32nd EACSL Annual Conference on Computer Science Logic

CSL 2024, February 19–23, 2024, Naples, Italy

Edited by

Aniello Murano

Alexandra Silva

LIPIcs – Vol. 288 – CSL 2024 www.dagstuhl.de/lipics

Editors

Aniello Murano
University of Naples Federico II, Italy
nello.murano@gmail.com

Alexandra Silva
Cornell University, Ithaca, NY, USA
alexandra.silva@gmail.com

ACM Classification 2012
Theory of computation → Logic

ISBN 978-3-95977-310-2

Published online and open access by
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at https://www.dagstuhl.de/dagpub/978-3-95977-310-2.

Publication date
February, 2024

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at https://portal.dnb.de.

Digital Object Identifier: 10.4230/LIPIcs.CSL.2024.0

LIPIcs – Leibniz International Proceedings in Informatics

LIPIcs is a series of high-quality conference proceedings across all fields in informatics. LIPIcs volumes are published according to the principle of Open Access, i.e., they are available online and free of charge.

ISSN 1868-8969

https://www.dagstuhl.de/lipics

Contents

Ackermann Award

Invited Talks

Regular Papers

32nd EACSL Annual Conference on Computer Science Logic (CSL 2024).
Editors: Aniello Murano and Alexandra Silva
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Preface

This volume contains the papers presented at CSL 2024, the 32nd meeting in the conference series Computer Science Logic (CSL), the annual conference of the European Association for Computer Science Logic (EACSL). CSL 2024 was held from 19th to 23th February 2024 in Naples, Italy.

CSL started as a series of international workshops, and became an international conference in 1992. Previous instalments of CSL were held in Warsaw (2023), Göttingen (2022, on-line), Ljubljana (2021, on-line), Barcelona (2020), Birmingham (2018), Stockholm (2017), Marseille (2016), Berlin (2015), Vienna (2014), Torino (2013), Fontainebleau (2012), Bergen(2011), Brno (2010), Coimbra (2009), Bologna (2008), Lausanne (2007), Szeged (2006), Oxford (2005), Karpacz (2004), Vienna (2003), Edinburgh (2002), Paris (2001), Munich (2000), Madrid (1999), Brno (1998), Aarhus (1997), Utrecht (1996), Paderborn (1995), Kazimierz (1994), Swansea (1993) and San Miniato (1992).

CSL is an interdisciplinary conference, spanning both basic and application-oriented research in mathematical logic and computer science. It is a forum for the presentation of research on all aspects of logic and its applications, including automated deduction and interactive theorem proving, constructive mathematics and type theory, equational logic and term rewriting, automata and games, game semantics, modal and temporal logic, logical aspects of computational complexity, finite model theory, computational proof theory, logic programming and constraints, lambda calculus and combinatory logic, domain theory, categorical logic and topological semantics, database theory, specification, extraction and transformation of programs, logical aspects of quantum computing, logical foundations of programming paradigms, verification and program analysis, linear logic, higher-order logic, and non-monotonic reasoning.

The conference received 106 abstracts of which 86 were followed up by full-paper submissions. The programme committee selected 42 papers for presentation at the conference. Each paper was reviewed by at least three members of the programme committee, with the help of external reviewers. The submission and reviewing process, programme committee discussion, and author notifications were all handled by the Easychair conference management system. In addition to the contributed papers, there were five invited talks, by: Balder ten Cate (University of Amsterdam, The Netherlands), Georg Gottlob (University of Calabria, Italy), Barbara König (University of Duisburg-Essen, Germany), Marta Kwiatkowska (Oxford University, United Kingdom), and Moshe Vardi (Rice University, United States). Barbara König was a joint invited speaker with the co-located workshop FICS (Fixpoints in Computer Science). We thank the invited speakers for their stimulating talks and papers, which greatly contributed to the success of the conference.

One of the major regular events at CSL conferences is the presentation of the Ackermann Award: the annual EACSL award for an outstanding dissertation in the area of logic in computer science. The recipients of the award are selected by jury from a field of international nominees, and the recipients receive their award at a ceremony at which they give a prize lecture on their dissertation. This year, the jury elected to give the Ackermann Award 2023 to Gabriele Vanoni for his thesis "On Reasonable Space and Time Cost Models for the λ-Calculus", defended at the University of Bologna (Italy) under the supervision of Ugo Dal Lago. The award was presented during the conference. The citation for the award is included in the proceedings. Another significant event at CSL 2024 was the presentation of the Helena Rasiowa Award, named after the eminent Polish mathematician and logician

32nd EACSL Annual Conference on Computer Science Logic (CSL 2024).
Editors: Aniello Murano and Alexandra Silva
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Helena Rasiowa (1917 – 1994) whose work had an essential impact on the emerging field of logic in computer science. The Helena Rasiowa Award, presented for the first time at CSL 2022, is given to the best paper, as decided by the programme committee, that is written solely by students or to which students were the main contributors. There was a strong field of candidates for this award edition, with 13 of the accepted papers eligible. From these, the programme committee selected Quentin Aristote as the recipient of the 2024 Helena Rasiowa Award, for his paper "Active Learning of Deterministic Transducers with Outputs in Arbitrary Monoids". Quentin Aristote is a PhD student at IRIF under the supervision of Daniela Petrisan.

CSL 2023 also had two affiliated workshops: Fixpoints in Computer Science (FICS) and the Logic Mentoring Workshop (LMW).

We are very grateful to all the members of the CSL 2024 programme committee and external reviewers for their careful and efficient evaluation of the papers submitted. We would like to thank also the members of the organisation committee – Marco Aruta, Davide Catta, Francesco Chiarello, Angelo Ferrando, Michał Tomasz Godziszewski, Alfredo Laino, Giulia Longo, Vadim Malvone, Munyque Mittelmann, Francesco Noviello – for taking care to ensure a smooth-running and enjoyable conference. It was as always a pleasure to work with Maribel Fernandez who, as the EACSL president, provided excellent guidance. The proceedings of CSL 2024 are published as a volume in the LIPIcs series. We thank Michael Wagner, Michael Didas and all the Dagstuhl/LIPIcs team for their ongoing support and for the high quality preparation of these proceedings. Last, but not least, we are very grateful to the Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione, the Università degli Studi di Napoli Federico II, and the Consorzio di Ricerca per l'Energia, l'Automazione e le Tecnologie dell'Elettromagnetismo (CREATE) for supporting the organisation of this conference.

Aniello Murano and Alexandra Silva 7th December 2023

Program Committee Members

Shaull Almagor	(Technion)
Christel Baier	(TU Dresden)
Filippo Bonchi	(University of Pisa)
James Brotherston	(University College London)
Valentina Castiglioni	(Reykjavik University)
Taolue Chen	(Birkbeck, University of London)
Diana Costa	(Universidade de Lisboa)
Silvia Ghilezan	(University of Novi Sad)
Nina Gierasimczuk	(Technical University of Denmark)
Sam van Gool	(Université Paris Cité)
Wojtek Jamroga	(Polish Academy of Sciences)
Benjamin Kaminski	(Saarland University)
Tobias Kappé	(Open University of the Netherlands and ILLC, University of Amsterdam)
Shin-Ya Katsumata	(National Institute of Informatics)
Marie Kerjean	(CNRS, LIPN, Université Sorbonne Paris Nord)
Juha Kontinen	(University of Helsinki)
Clemens Kupke	(University of Strathclyde)
Martin Lange	(University of Kassel)
Carsten Lutz	(Universität Bremen)
Nicolas Markey	(IRISA, CNRS & INRIA & Univ. Rennes 1)
Larry Moss	(Indiana University Bloomington)
Aniello Murano	(University of Naples Federico II, co-chair)
Pierre Ohlmann	(University of Warsaw)
Guillermo Perez	(University of Antwerp)
Nir Piterman	(Chalmers University of Technology)
Jurriaan Rot	(Radboud University)
Lutz Schröder	(Friedrich-Alexander-Universität Erlangen-Nürnberg)
Alexandra Silva	(Cornell University, co-chair)
Sonja Smets	(University of Amsterdam)
Pawel Sobocinski	(Tallinn University of Technology)
Luca Spada	(Università di Salerno)
Sam Staton	(University of Oxford)
Christine Tasson	(LIP6 - Sorbonne Université)
Andrea Turrini	(Institute of Software, Chinese Academy of Sciences)
Martin Zimmermann	(Aalborg University)

32nd EACSL Annual Conference on Computer Science Logic (CSL 2024).
Editors: Aniello Murano and Alexandra Silva
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

External Reviewers

Marco Abbadini
Matteo Acclavio
Yehia Alrahman
Amazigh Amrane
Eugene Asarin
Robert Atkey
Shaun Azzopardi
Henning Basold
Nicolas Basset
Emmanuel Beffara
Benno van den Berg
Valentin Blot
Ilario Bonacina
Flavien Breuvart
Alessandro Bruni
Florian Bruse
Simon Burton
Davide Catta
Aggeliki Chalki
James Chapman
Vikraman Choudhury
Andrea Corradini
Fredrik Dahlqvist
Tiziano Dalmonte
Luc Dartois
Anuj Dawar
Daniele Dell'Erba
Ryan Doenges
Peter Dybjer
Mirna Dzamonja
Kord Eickmeyer
Hugo Férée
Daniel Figueiredo
Marie Fortin
Zeinab Galal
Francesco Gavazzo
Guillaume Geoffroy
Lorenzo Gheri
Alessandro Di Giorgio
Jean Goubault-Larrecq
Mario Grobler
Ji Guan
Adrien Guatto
Shibashis Guha
Ronald de Haan
Amar Hadzihasanovic
Miika Hannula
Daniel Hausmann
Markus Hecher
Lauri Hella
Léo Henry
Tom Hirschowitz
Åsa Hirvonen
Mathieu Huot
Tomáš Jakl
Prabhat Jha
Makoto Kanazawa
Alex Kavvos
Yevgeny Kazakov
Yan Kim
Eryk Kopczynski
Louwe Kuijer
Stepan Kuznetsov
Ambroise Lafont
Victor Lagerkvist
Serafina Lapenta
Graham Leigh
Pierre Lescanne
Yong Li
Moritz Lichter
Luigi Liquori
Fosco Loregian
Tim Lyon
Alexandre Madeira
Alessio Mansutti
Radu Mardare
Sonia Marin
Manuel Martins
Samuele Maschio
Damiano Mazza
Arne Meier
Arne Meier
Matías Menni
Giuseppe Metere
Vincent Michielini
Lukasz Mikulski
Joe Moeller
Joshua Moerman
Sean Moss
Fredrik Nordvall Forsberg
Jovana Obradović
Federico Olimpieri
Jakub Opršal
Martin Otto
Paritosh Pandya
Luca Paolini
Nicolas Peltier
Wojciech Penczek
Diogo Poças
Cécilia Pradic
Ritam Raha
Gaurav Rattan
Mario Román
Neil Ross
Wojciech Rozowski
Jakub Rydval
Marco Sälzer
Matteo Sammartino
Alessio Santamaria
Alexis Saurin
Sven Schewe
Ezra Schoen
Pascal Schweitzer
Thomas Seiller
Paul Shafer
Ian Shillito
Teofil Sidoruk
Jesse Sigal
Matthieu Sozeau
Luca Di Stefano
Andrew Swan
Lê Thành Dũng Nguyen
Katrine Thoft
Stelios Tsampas
Ruben Turkenburg
Jouko Vaananen
Lena Verscht
Jonni Virtema
Jana Wagemaker
James Wood
Zhilin Wu
Tetsuo Yokoyama
Noam Zeilberger
Stanislav Živný

32nd EACSL Annual Conference on Computer Science Logic (CSL 2024).
Editors: Aniello Murano and Alexandra Silva
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The Ackermann Award 2023

Maribel Fernández
Department of Informatics, King's College London, UK

Jean Goubault-Larrecq
LSV, Ecole Normale Supérieure Paris-Saclay, France

Delia Kesner
IRIF (CNRS UMR 8243) – Université Paris Cité, France

Abstract

Report on the 2023 Ackermann Award.

2012 ACM Subject Classification Theory of computation → Logic and verification; Theory of computation → Lambda calculus; Theory of computation → Equational logic and rewriting

Keywords and phrases lambda-calculus, computational complexity, geometry of interaction, abstract machines, intersection types

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.1

Category Ackermann Award

1 The Ackermann Award

The Ackermann Award is the **EACSL Outstanding Dissertation Award for Logic in Computer Science**. It is presented at CSL, the annual conference of the EACSL (European Association for Computer Science Logic). This year, the 19th Ackermann Award is presented at CSL 2024 in Naples, Italy.

A call for nominations was issued in February 2023, open to any PhD dissertation (on any topic represented at the annual CSL and LICS conferences) formally accepted by a degree-granting institution in fulfilment of the PhD degree between 1 January 2022 and 31 December 2022. The Jury received ten nominations, which came from a number of different countries around the world: the nominees obtained their doctorates at institutions in Australia, Brazil, Germany, Italy, Spain and Sweden.

The topics covered a wide range of areas in Logic and Computer Science. All the nominated PhD theses were of a very high quality and contained significant contributions to their particular fields. On behalf of the Ackermann Jury, we extend our warmest congratulations to all the nominated candidates for their outstanding work.

All the submissions were evaluated by the Jury, and after two phases of reviewing and extensive discussion, the jury decided to grant the **2023 Ackermann Award** to **Gabriele Vanoni** for the PhD thesis entitled *On Reasonable Space and Time Cost Models for the λ-Calculus*, completed at the *University of Bologna*, Italy, in 2022.

2 Citation

Gabriele Vanoni receives the *2023 Ackermann Award* of the European Association of Computer Science Logic for the PhD thesis

On Reasonable Space and Time Cost Models for the λ-Calculus,

which provides the first reasonable space cost model for the λ-calculus, showing that the space complexity of functional programs is equivalent to that of Turing machines. To achieve this

ambitious goal, the thesis defines new abstract machines for call-by-name and call-by-value evaluation of λ-terms, as well as providing a characterisation of this new reasonable space measure by means of intersection types.

2.1 Background to the thesis

The thesis is in the domain of logical foundations of functional programming languages. More precisely, it studies space and time cost models for functional programs.

The use of resources (time and space consumption) during computation is typically measured on Turing machines (TMs), where time complexity refers to the number of steps the machine needs to perform until it stops, and space complexity is the number of tape cells used during the computation. However, TMs are a low-level model of computation, which does not provide an understanding of the time and space consumed on higher-level models of computation such as the ones that are at the core of modern programming languages.

The Slot and van Emde Boas Invariance Thesis states that a time (respectively, space) cost model is reasonable for a computation model C if there are mutual simulations between Turing machines and C such that the overhead is polynomial in time (respectively, linear in space). The rationale is that under the Invariance Thesis, complexity classes (such as LOGSPACE, P, PSPACE) are machine independent.

In his PhD thesis, Vanoni addresses the problem of finding out whether it is possible to define a reasonable space cost model for the λ-calculus, the paradigmatic computation model for functional programming languages.

2.2 Contributions of the thesis

In order to achieve the objective of the thesis, the work is organised in two stages. The first half of Vanoni's thesis is devoted to disproving a longstanding open conjecture in the field, originally stating that Girard's Geometry of Interaction is a reasonable space cost model for the λ-calculus. The second half of the thesis uses the intuitions gained with this first negative result to finally build a reasonable space cost model for the λ-calculus.

More precisely, the thesis starts by considering an evaluator for the λ-calculus, the λIAM, which is based on Girard's Geometry of Interaction. The latter was conjectured to be a good candidate to obtain a reasonable cost model for space. Vanoni disproved this conjecture by a detailed complexity analysis using new variants of non-idempotent intersection types. As a consequence, he changed the target of his analysis and considered a variant of Krivine's abstract machine (a standard evaluation mechanism for the call-by-name λ-calculus), which he optimised for space complexity and implemented without any pointer. By analysing a refined version of the encoding of Turing machines into the λ-calculus, Vanoni concluded that the space consumed by this machine is indeed a reasonable space cost model. In particular, for the first time it was also possible to measure sub-linear space complexities. This result translates also to the call-by-value case.

One should note that the challenges are rather formidable. For one, the various abstract machines need to implement computations in the λ-calculus with a fine control over garbage collection. The λIAM was promising precisely because it did not need any garbage collection. Vanoni's modified Krivine abstract machine instead resorts to eager garbage collection. Another subtle point is that, in order to obtain meaningful notions of sub-linear space complexity, one needs to separate the amount of work space from the size of the input, since the latter must not be counted, as in Turing machine models; and the usual abstract machines for the λ-calculus do not allow one to distinguish between input and non-input data.

Finally, Vanoni provided a characterisation of this new reasonable space measure by means of intersection types. This was instrumental in characterising the space complexity of each of the abstract machines considered, and was done through a minimal, yet non-trivial, modification of the original type system proposed by de Carvalho. Eventually, this leads Vanoni to obtain characterisations of reasonable space complexity that are modular and independent of the machine model.

2.3 Biographical sketch

Gabriele Vanoni carried out his PhD under the supervision of Ugo Dal Lago at the University of Bologna, Italy. He is the winner of the E.W. Beth Dissertation Prize 2023 and Best Italian PhD thesis in theoretical computer science 2023 (awarded by the Italian Chapter of EATCS). He won a Distinguished Paper Award at ICFP 2022, and his LICS 2022 paper was selected for the LMCS special issue. He was invited speaker at DCM 2023 and TLLA 2023. After a year as a postdoctoral researcher at Inria in Sophia Antipolis, he is now a postdoctoral researcher at IRIF in Paris.

3 Jury

The jury for the **Ackermann Award 2023** consisted of eight members, two of them *ex officio*, namely, the president and the vice-president of EACSL. In addition, the jury also included a representative of SIGLOG (the ACM Special Interest Group on Logic and Computation).

The members of the jury were:

- Christel Baier (Technical University Dresden);
- Maribel Fernández (King's College London), president of EACSL;
- Joost-Pieter Katoen (RWTH Aachen University), ACM SIGLOG representative;
- Delia Kesner (IRIF, Université Paris Cité);
- Slawomir Lasota (University of Warsaw);
- Jean Goubault-Larrecq (ENS Paris-Saclay);
- Florin Manea (University of Göttingen), vice-president of EACSL;
- James Worrell (University of Oxford).

4 Previous winners

Previous winners of the Ackermann Award were

2005, Oxford:
Mikołaj Bojańczyk from Poland,
Konstantin Korovin from Russia, and
Nathan Segerlind from the USA.

2006, Szeged:
Balder ten Cate from the Netherlands, and
Stefan Milius from Germany.

2007, Lausanne:
Dietmar Berwanger from Germany and Romania,
Stéphane Lengrand from France, and
Ting Zhang from the People's Republic of China.

2008, Bertinoro:
Krishnendu Chatterjee from India.

2009, Coimbra:
Jakob Nordström from Sweden.
2010, Brno:
no award given.
2011, Bergen:
Benjamin Rossman from USA.
2012, Fontainebleau:
Andrew Polonsky from Ukraine, and
Szymon Toruńczyk from Poland.
2013, Turin:
Matteo Mio from Italy.
2014, Vienna:
Michael Elberfeld from Germany.
2015, Berlin:
Hugo Férée from France, and
Mickael Randour from Belgium.
2016, Marseille:
Nicolai Kraus from Germany
2017, Stockholm:
Amaury Pouly from France.
2018, Birmingham:
Amina Doumane from France.
2019, Barcelona (conference in 2020):
Antoine Mottet from France.
2020, Ljubljana (conference online in 2021)
Benjamin Kaminski from Germany.
2021, Göttingen (conference online in 2022)
Marie Fortin from France, and
Sandra Kiefer from Germany
2022, Warsaw (conference in 2023)
Alexander Bentkamp from The Netherlands.

Detailed reports on their work appeared in the CSL proceedings and are also available on the EACSL homepage.

Craig Interpolation for Decidable Fragments of First-Order Logic

Balder ten Cate
ILLC, University of Amsterdam, The Netherlands

Abstract

The *Craig Interpolation Property* (CIP) is a property of logics. It states that, for all formulas φ and ψ, if $\varphi \models \psi$, then there exists an "interpolant" ϑ such that $\varphi \models \vartheta$ and $\vartheta \models \psi$, and such that all non-logical symbols occurring in ϑ occur both in φ and in ψ. Craig [6] proved in 1957 that first-order logic (FO) has this property. Since then, many refinements of Craig's result have been obtained (e.g., [12, 3]). These have found applications in various areas of computer science and AI, including formal verification, modular hard/software specification and automated deduction [11, 4, 7], and more recently prominently in databases [14, 2] and knowledge representation [10, 5, 9]. In this invited talk, we will survey recent work pertaining to Craig interpolation for various important decidable fragment of first-order logic, including guarded fragments, finite-variable fragments, and ordered fragments. Most of these fragments lack the CIP (the guarded-negation fragment GNFO being a notable exception [1]). We will discuss strategies that have been proposed in recent literature to deal with this lack of CIP, as well as recent results that shed light on where, within the landscape of decidable fragment of first-order logic, one may find logics that enjoy CIP [8, 13].

2012 ACM Subject Classification Theory of computation → Finite Model Theory; Theory of computation → Automated reasoning

Keywords and phrases First-Order Logic, Decidable Fragments, Craig Interpolation

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.2

Category Invited Talk

Funding *Balder ten Cate*: Supported by EU Horizon 2020 grant MSCA-101031081.

References

1 Vince Bárány, Michael Benedikt, and Balder ten Cate. Rewriting guarded negation queries. In *Proceedings of MFCS 2013*, pages 98–110, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

2 Michael Benedikt, Julien Leblay, Balder ten Cate, and Efthymia Tsamoura. *Generating plans from proofs : the interpolation-based approach to query reformulation*. Synthesis Lectures on Data Management. Morgan & Claypool, 2016.

3 Michael Benedikt, Balder ten Cate, and Efthymia Tsamoura. Generating plans from proofs. *ACM Trans. Database Syst.*, 40(4):22:1–22:45, 2016. doi:10.1145/2847523.

4 Diego Calvanese, Silvio Ghilardi, Alessandro Gianola, Marco Montali, and Andrey Rivkin. Combined covers and beth definability. In *Proceedings of the 10th International Joint Conference on Automated Reasoning, Part I, IJCAR 2020*, pages 181–200. Springer, 2020. doi:10.1007/978-3-030-51074-9_11.

5 Balder ten Cate, Enrico Franconi, and Inanç Seylan. Beth definability in expressive description logics. *J. Artif. Int. Res.*, 48(1):347–414, October 2013.

6 William Craig. Three uses of the herbrand-gentzen theorem in relating model theory and proof theory. *Journal of Symbolic Logic*, 22(3):269–285, 1957. doi:10.2307/2963594.

7 Krystof Hoder, Andreas Holzer, Laura Kovács, and Andrei Voronkov. Vinter: A Vampire-based tool for interpolation. In Ranjit Jhala and Atsushi Igarashi, editors, *Programming Languages and Systems - 10th Asian Symposium, APLAS 2012, Kyoto, Japan, December 11-13, 2012. Proceedings*, volume 7705 of *Lecture Notes in Computer Science*, pages 148–156. Springer, 2012. doi:10.1007/978-3-642-35182-2_11.

8 Jean Christoph Jung and Frank Wolter. Living without beth and craig: Definitions and interpolants in the guarded and two-variable fragments. In *Proceedings of LICS 2021*, pages 1–14. IEEE Computer Society, July 2021. doi:10.1109/LICS52264.2021.9470585.

9 Patrick Koopmann and Renate A. Schmidt. Uniform interpolation and forgetting for $\mathcal{ALC}$ ontologies with ABoxes. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence, AAAI 2015*, pages 175–181. AAAI Press, 2015. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9981.

10 Carsten Lutz and Frank Wolter. Foundations for uniform interpolation and forgetting in expressive description logics. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI 2011*, pages 989–995. IJCAI/AAAI, 2011. doi:10.5591/978-1-57735-516-8/IJCAI11-170.

11 Kenneth L. McMillan. Interpolation and model checking. In Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem, editors, *Handbook of Model Checking*, pages 421–446. Springer, 2018. doi:10.1007/978-3-319-10575-8_14.

12 Martin Otto. An interpolation theorem. *The Bulletin of Symbolic Logic*, 6(4):447–462, 2000. URL: http://www.jstor.org/stable/420966.

13 Balder ten Cate and Jesse Comer. Craig interpolation for decidable first-order fragments, 2023. arXiv:2310.08689.

14 David Toman and Grant E. Weddell. *Fundamentals of Physical Design and Query Compilation*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011. doi:10.2200/S00363ED1V01Y201105DTM018.

Artificial Intelligence and Artificial Ignorance

Georg Gottlob
University of Calabria, Italy
University of Oxford, UK

Abstract

This invited talk first delves into the division between the two primary branches of AI research: symbolic AI, which predominantly focuses on knowledge representation and logical reasoning, and sub-symbolic AI, primarily centered on machine learning employing neural networks. We explore both the notable accomplishments and the challenges encountered in each of these approaches. We provide instances where traditional deep learning encounters limitations, and we elucidate significant obstacles in achieving automated symbolic reasoning. We then discuss the recent groundbreaking advancements in generative AI, driven by language models such as ChatGPT. We showcase instances where these models excel and, conversely, where they exhibit shortcomings and produce erroneous information. We identify and illustrate five key reasons for potential failures in language models, which include:

- **(i)** information loss due to data compression,
- **(ii)** training bias,
- **(iii)** the incorporation of incorrect external data,
- **(iv)** the misordering of results, and
- **(v)** the failure to detect and resolve logical inconsistencies contained in a sequence of LLM-generated prompt-answers.

Lastly, we touch upon the Chat2Data project, which endeavors to leverage language models for the automated verification and enhancement of relational databases, all while mitigating the pitfalls (i)–(v) mentioned earlier.

2012 ACM Subject Classification Computing methodologies → Artificial intelligence; Computing methodologies → Machine learning; Computing methodologies → Knowledge representation and reasoning; Computing methodologies → Natural language generation; Information systems → World Wide Web

Keywords and phrases AI applications, symbolic AI, sub-symbolic AI, AI usefulness, achievements of symbolic AI, achievements of machine learning, machine learning errors and mistakes, large language models, LLMs, LLM usefulness, LLM mistakes, inaccuracies

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.3

Category Invited Talk

Funding *Georg Gottlob*: The author was supported by the Royal Society Research Professorship project "RAISON DATA" (Reference No. RP\R1\201074) and by the University of Calabria.

Approximating Fixpoints of Approximated Functions

Barbara König

University of Duisburg-Essen, Germany

Abstract

There is a large body of work on fixpoint theorems, guaranteeing the existence of fixpoints for certain functions and providing methods for computing them. This includes for instance Banachs's fixpoint theorem, the well-known result by Knaster-Tarski that is frequently employed in computer science and Kleene iteration.

It is less clear how to compute fixpoints if the function whose (least) fixpoint we are interested in is not known exactly, but can only be obtained by a sequence of subsequently better approximations. This scenario occurs for instance in the context of reinforcement learning, where the probabilities of a Markov decision process (MDP) – for which one wants to learn a strategy - are unknown and can only be sampled. There are several solutions to this problem where the fixpoint computation (for determining the value vector and the optimal strategy) and the exploration of the model are interleaved. However, these methods work only well for discounted MDPs, that is in the contractive setting, but not for general MDPs, that is for non-expansive functions.

After describing and motivating the problem, we will in particular concentrate on the non-expansive case. There are many interesting systems who value vectors can be obtained by determining the fixpoints of non-expansive functions. Other than contractive functions, they do not guarantee uniqueness of the fixpoint, making it more difficult to approximate the least fixpoint by methods other than Kleene iteration. And also Kleene iteration fails if the function under consideration is only approximated.

We hence describe a dampened Mann iteration scheme for (higher-dimensional) functions on the reals that converges to the least fixpoint from everywhere. This scheme can also be adapted to functions that are approximated, under certain conditions.

We will in particular study the case of MDPs and consider a related problem that arises when performing model-checking for quantitative μ-calculi, which involves the computation of nested fixpoints.

This is joint work with Paolo Baldan, Sebastian Gurke, Tommaso Padoan and Florian Wittbold.

2012 ACM Subject Classification Theory of computation → Logic and verification; Theory of computation → Program reasoning; Theory of computation → Reinforcement learning

Keywords and phrases fixpoints, approximation, Markov decision processes

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.4

Category Invited Talk

Strategy Synthesis for Partially Observable Stochastic Games with Neural Perception Mechanisms

Marta Kwiatkowska
Department of Computer Science, University of Oxford, UK

Abstract

Strategic reasoning is essential to ensure stable multi-agent coordination in complex environments, as it enables synthesis of optimal (or near-optimal) agent strategies and equilibria that guarantee expected outcomes, even in adversarial scenarios. *Partially-observable stochastic games* (POSGs) are a natural model for real-world settings involving multiple agents, uncertainty and partial information, but lack practical algorithms for computing or approximating optimal values and strategies. Recently, progress has been made for *one-sided POSGs*, a subclass of two-agent, zero-sum POSGs where only one agent has partial information while the other agent is assumed to have full knowledge of the state, with heuristic search value iteration (HSVI) proposed for computing approximately optimal values and strategies in one-sided POSGs [1]. This model is well suited to safety-critical applications, when making worst-case assumptions about one agent; examples include the attacker in a security application, modelled, e.g., as a patrolling or pursuit-evasion game.

However, many realistic autonomous coordination scenarios involve agents perceiving *continuous environments* using *data-driven* observation functions, typically implemented as neural networks (NNs). Examples include autonomous vehicles using NNs to perform object recognition or to estimate pedestrian intention, or NN-enabled vision in an airborne pursuit-evasion scenario. Such perception mechanisms bring new challenges, notably continuous environments, which are inherently tied to NN-enabled perception because of standard training regimes. This means that naive discretisation is difficult, since decision boundaries obtained for data-driven perception are typically irregular and can be misaligned with gridding schemes for discretisation, affecting the precision of the computed strategies.

This invited paper will discuss progress with developing a model class and algorithms for one-sided POSGs with neural perception mechanisms [2, 3] that work directly with their continuous state space. Building on continuous-state POMDPs with NN perception mechanisms [4], where the key idea is that ReLU neural network classifiers induce a finite decomposition of the continuous environment into polyhedra for each classification label, a piecewise constant representation for the value, reward and perception functions is developed that forms the basis for a variant of HSVI, a point-based solution method that computes a lower and upper bound on the value function from a given belief to compute an (approximately) optimal strategy. We extend these ideas from the single-agent (POMDP) setting [4] to zero-sum POSGs. In the game setting, this involves solving a normal form game at each stage and iteration, and goes significantly beyond HSVI for finite POSGs [1].

2012 ACM Subject Classification Theory of computation → Logic and verification; Computing methodologies → Neural networks

Keywords and phrases Stochastic games, neural networks, formal verification, strategy synthesis

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.5

Category Invited Talk

Related Version *Full Version*: `https://arxiv.org/abs/2310.11566`

Funding This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 834115).

References

1 Karel Horák, Branislav Bošanskỳ, Vojtěch Kovařík, and Christopher Kiekintveld. Solving zero-sum one-sided partially observable stochastic games. *Artificial Intelligence*, 316:103838, 2023.

2 R. Yan, G. Santos, G. Norman, D. Parker, and M. Kwiatkowska. Strategy synthesis for zero-sum neuro-symbolic concurrent stochastic games. *arXiv*, 2022. Accepted to Information and Computation. `arXiv:2202.06255`.

3 Rui Yan, Gabriel Santos, Gethin Norman, David Parker, and Marta Kwiatkowska. Partially observable stochastic games with neural perception mechanisms. *arXiv*, 2023. `arXiv:2310.11566`.

4 Rui Yan, Gabriel Santos, Gethin Norman, David Parker, and Marta Kwiatkowska. Point-based value iteration for neuro-symbolic POMDPs. *arXiv*, 2023. `arXiv:2306.17639`.

Logical Algorithmics: From Theory to Practice

Moshe Y. Vardi
Rice University, Houston, TX, USA

Abstract

The standard approach to algorithm development is to focus on a specific problem and develop for it a specific algorithm. Codd's introduction of the relational model in 1970 included two fundamental ideas: (1) Relations provide a universal data representation formalism, and (2) Relational databases can be queried using first-order logic. Realizing these ideas required the development of a meta-algorithm, which takes a declarative query and executes it with respect to a database. In this talk, I will describe this approach, which I call Logical Algorithmics, in detail, and explore its profound ramification.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography; Theory of computation → Design and analysis of algorithms

Keywords and phrases Logic, Algorithms

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.6

Category Invited Talk

Funding Work supported in part by NSF grants IIS-1527668, CCF-1704883, IIS-1830549, CNS-2016656, DoD MURI grant N00014-20-1-2787, and an award from the Maryland Procurement Office.

References

1 Jeffrey M. Dudek, Vu Phan, and Moshe Y. Vardi. ADDMC: weighted model counting with algebraic decision diagrams. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 1468–1476. AAAI Press, 2020. doi:10.1609/AAAI.V34I02.5505.

2 Phokion G. Kolaitis and Moshe Y. Vardi. Conjunctive-query containment and constraint satisfaction. *J. Comput. Syst. Sci.*, 61(2):302–332, 2000. doi:10.1006/JCSS.2000.1713.

3 Guoqiang Pan and Moshe Y. Vardi. Symbolic techniques in satisfiability solving. *J. Autom. Reason.*, 35(1-3):25–50, 2005. doi:10.1007/S10817-005-9009-7.

4 Vu H. N. Phan and Moshe Y. Vardi. DPO: dynamic-programming optimization on hybrid constraints. *CoRR*, abs/2205.08632, 2022. doi:10.48550/ARXIV.2205.08632.

5 Moshe Y. Vardi. The complexity of relational query languages (extended abstract). In Harry R. Lewis, Barbara B. Simons, Walter A. Burkhard, and Lawrence H. Landweber, editors, *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA*, pages 137–146. ACM, 1982. doi:10.1145/800070.802186.

6 Moshe Y. Vardi. On the complexity of bounded-variable queries. In Mihalis Yannakakis and Serge Abiteboul, editors, *Proceedings of the Fourteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 22-25, 1995, San Jose, California, USA*, pages 266–276. ACM Press, 1995. doi:10.1145/212433.212474.

Semantic Bounds and Multi Types, Revisited

Beniamino Accattoli
Inria & LIX, École Poytechnique, Palaiseau, France

Abstract

Intersection types are a standard tool in operational and semantical studies of the λ-calculus. De Carvalho showed how multi types, a quantitative variant of intersection types providing a handy presentation of the relational denotational model, allows one to extract precise bounds on the number of β-steps and the size of normal forms.

In the last few years, de Carvalho's work has been extended and adapted to a number of λ-calculi, evaluation strategies, and abstract machines. These works, however, only adapt the first part of his work, that extracts bounds from multi type *derivations*, while never consider the second part, which deals with extracting bounds from the multi types themselves. The reason is that this second part is more technical, and requires to reason up to *type substitutions*. It is however also the most interesting, because it shows that the bounding power is *inherent* to the relational model (which is induced by the types, without the derivations), independently of its presentation as a type system.

Here we dissect and clarify the second part of de Carvalho's work, establishing a link with principal multi types, and isolating a key property independent of type substitutions.

2012 ACM Subject Classification Theory of computation → Lambda calculus; Theory of computation → Denotational semantics; Theory of computation → Operational semantics

Keywords and phrases Lambda calculus, intersection types, denotational semantics, linear logic

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.7

Related Version *Paper with proof appendix hosted on arXiv*: `http://arxiv.org/abs/2311.18233` [1]

1 Introduction

Denotational semantics studies invariants of program evaluation. The typical way in which it is connected to the operational semantics of λ-calculi is at the *qualitative* level, via *adequacy*: the denotational interpretation $[\![t]\!]$ of a λ-term t is non-trivial (typically non-empty) if and only if the evaluation of t terminates.

At first sight, denotational semantics cannot provide *quantitative* operational insights such as evaluation lengths, because of its invariance by evaluation. Things are in fact not so black and white. Being invariant by evaluation, denotational semantics models normal forms, and in a *compositional* way: by composing the interpretations of two terms one can obtain the interpretation of the result of their application – therefore, denotational semantics does reflect the evaluation process *somehow*.

The aim of this paper is to revisit some overlooked – but we believe important – results for the λ-calculus by de Carvalho, about the extraction of bounds on evaluation lengths and the size of normal forms from the interpretation of terms into the relational model.

Relational Semantics and Multi Types. The relational model [37, 17] is a simple denotational semantics of the λ-calculus induced by the relational model of linear logic, via the representation of the λ-calculus in linear logic. It is a paradigmatic model, underlying many others [30, 31, 24, 45, 48], mainly studied by Ehrhard and his students and co-authors [38, 18, 21, 16, 32, 33, 34, 26, 47, 46, 49, 15], the importance of which has emerged slowly. One of its features is that it admits a handy presentation via an intersection type system.

The distinguished property of intersection types is that they *characterize* termination properties, in the sense that they not only enforce termination, but they also type *all* terminating terms. Additionally, by tuning the definition of the type system, one can capture

different notions of termination (weak head/head/leftmost termination, strong normalization, call-by-value/need, etc). Such a characterization usually induces a model: the set of types for a term t is an invariant of the characterized notion of evaluation $\to$, which gives rise to a denotational semantics which is adequate for $\to$. Therefore, intersection type systems are a tool halfway the operational and the denotational semantics of the λ-calculus.

Multi types are a *linear logic-related* variant of intersection types where intersections are *non-idempotent* (they are also known as *non-idempotent intersection types*), that is where $A \cap A \neq A$. The set of multi types judgements derivable for a term t provides a denotation $[\![t]\!]$ which coincides with the interpretation into the relational model.

Multi Types and Quantitative Bounds. In a seminal work, de Carvalho used the multi type system to obtain exact bounds on the evaluation lengths and the size of normal form for λ-terms [25, 27]. The relevance of his work was fully appreciated by the community only a decade later (as surveyed below), when it blossomed into a number of variations and extensions by other authors. De Carvalho developed *two* main results, but only the first one has widely permeated the community. The second one is arguably the most technical but also the deeper one. The aim of this paper is to make it accessible to a wider audience.

De Carvalho's original presentation in [25, 27] uses multi types to measure two forms of strong evaluations realized by abstract machines, implementing the head and leftmost call-by-name strategies. In this overview, we prefer to slightly depart from the (by now somewhat outdated) details of his work, forgetting about abstract machines, focussing on leftmost evaluation (the head case is treated at the end of the paper), and isolating three (rather than two) kinds of bounds:

1. *Bounds from type derivations.* The size $|\Phi|$ of a type derivation $\Phi \rhd \Gamma \vdash t : \sqsubset$ bounds the number of leftmost steps from t to its normal form $\mathtt{nf}(t)$ *plus* the size $|\mathtt{nf}(t)|$ of the normal form. Moreover, derivations of minimal size provide exact bounds.
2. *Size bounds from types.* The types in the final judgment – a point of the relational interpretation $[\![t]\!]$ – also provide a bound, independently of the derivation Φ. Being invariant by evaluation, they cannot bound evaluation lengths. They do however bound the size $|\mathtt{nf}(t)|$ of the normal form, and bounds are exact when types are minimal.
3. *Bounds from composable types.* De Carvalho shows that types can be used to bound evaluation lengths *compositionally*: from the types in $[\![t]\!]$ and $[\![u]\!]$ it is possible to extract bounds about the leftmost evaluation and the normal form of tu, with *no reference to type derivations*. Exact bounds rest on a complex construction involving type substitutions.

The third kind of bounds is de Carvalho's second result, and it is where the bounding power of the multi type system (which is just one possible way of defining relational semantics) is *lifted* to relational semantics. Therefore, the lifting guarantees that the bounding power is an *inherent* feature of the relational model – multi types are just a handy tool to show it.

Of the three results, the third one is the most technical. In particular, it requires to enrich the type system with an infinity of type variables and work up to type substitutions. The puzzling fact is that these extra features play no role in the two previous points.

De Carvalho's Legacy. De Carvalho developed his results in his PhD, defended in 2007 [25]. His work was known by the community thanks to a technical report, turned into a journal paper only much later, in 2018 [27]. Soon after his PhD, he adapted his work to linear logic, with Pagani and Tortora de Falco [28, 29]. Then, Bernadet and Graham-Lengrand adapted his work to measure the longest evaluation in the λ-calculus [14], but they only provided bounds of the first kind (*from type derivations*).

At the time, it was not known whether it would make sense to count the number of β-steps (or linear logic cut-elimination steps) as a reasonable measure of complexity. After the case of the λ-calculus was clarified in the positive by Accattoli and Dal Lago [2], de Carvalho's work has been revisited by Accattoli et al. [7]. The revisitation started a new wave of works adapting de Carvalho's study to many evaluation strategies and extensions of the λ-calculus, including call-by-value [8, 9], call-by-need [10], a linear logic presentation of call-by-push-value [19, 42], the $\lambda\mu$-calculus [41], the λ-calculus with pattern matching [13], generalized applications [35], fully lazy sharing [39], global state [12], the probabilistic λ-calculus [23], the abstract machine underlying the geometry of interaction [3], and even adapted as to measure *space* [4, 5]. All these works provide bounds of the first kind, and some of them also of the second kind, but *none of them* deals with those of the third kind (*bounds from composable types*).

Contributions. This paper revisits the *bounds from composable types*, appeared only in [27, 28]. Beyond providing cleaner proofs of the results, we have a very close look at these bounds, isolating the subtleties and decomposing the proofs in smaller steps. In particular:

- *Subtlety 1, minimality does not work*: when bounding a single term, both derivations and types provide exact bounds when they are minimal. When dealing with the application of t to u, every pair of composable types for them provides bounds. We stress that the technicalities for bounds from composable types are inherent to the problem, since the minimal composable pair in general does *not* provide exact bounds.
- *Subtlety 2, the gap between derivations and types*: we draw attention to the fact that the previous subtlety stems from a fact about derivations in *isolation*, namely that, for a normal term t, both the derivation $\Phi \rhd \Gamma \vdash t : \mathsf{L}$ and the types in Γ and L provide bounds for $|t|$, but they may not coincide. In general, the bound from types is *laxer*. The bounds gap hinders the possibility of lifting the bounding power from derivations to types, if bounds from some derivations are not reflected by any type in the interpretation $[\![t]\!]$.
- *Dry representation and type substitutions*: we isolate the property behind de Carvalho's solution of this problem, which rests on type variables and type substitutions, and we connect it with the study of principal types. The idea is that given a type derivation $\Phi \rhd \Gamma \vdash t : \mathsf{L}$ for which there is a gap between the bound from Φ and the bound from Γ and L, there always is a second *dry* derivation $\Psi \rhd \Delta \vdash t : \mathsf{L}'$, whose types Δ and L' give the same bound as the first derivation Φ, *plus* a type substitution σ turning Ψ into Φ. Then, all bounds coming from derivations can also be seen as coming from types, potentially from the types of other derivations – the dry ones – but related via type substitutions.
- *Removing substitutions*: lastly, we show that *weaker* bounds from composable types can be obtained *without* dealing with an infinity of type variables and type substitutions. This point provides both a simplified route to the (slightly weaker) bounds and an explanation of why these additional technicalities are needed for the full result.

Internal *vs* External View. The problem studied in the paper can also be seen in a more abstract way. Terms, or more generally programs, can be studied from two perspectives, which are distinct and yet entangled. The internal view considers programs as closed systems and looks at their internal evaluation in isolation. In the external view, programs are seen as parts of larger systems. What is relevant is how programs compose and interact with each other, their internal evaluation is instead secondary and hidden.

Cost analyses are usually done following the internal view, while denotational semantics and types are external-oriented tools. Normal forms can be seen as internalizing the external information, as they are normal for the internal process and thus retain only information for potential external interactions. Multi type derivations capture both the number of steps of typed terms (which is an internal information) and the structure of their normal forms (which is the internalization of external information). Multi types instead capture the external information only (each type capturing a potential interaction).

Bounds from composable types connect internal and external information, as they use types (that are external) to bound the length of evaluation (which is internal) of the isolated system given by the two composed terms. The bounds are obtained building over the connection between types and normal forms. The difficulty in this study stems from the fact that in general there is a gap between how the external information is represented in types (richly, distinguishing between different interactions) and how it is internalized in normal forms (in a raw way, all possible interactions are flattened on a single object).

Limits of de Carvalho's Approach. The aim of this paper is also to highlight a fundamental limitation of de Carvalho's work. As we ourselves suggested, bounds from composable types can be seen as lifting the bounding power from the type system to the relational model. There is however a delicate point, as the lifting does not cover the whole of the model. The relational model, indeed, does not only interpret full normal forms and terminating terms, but also *head* normal forms and terms that are *head normalizable*. In particular, there are terms that are *hereditarily* head normalizing and yet never fully normalize, such as fix-point combinators. For these terms, which have non-empty interpretation in the relational model, de Carvalho's study does not say anything, because it assumes that the terms to be composed are fully normal. De Carvalho's bounds for head reduction do not solve the issue, they rather make it worse, since his theorems for the head case still have to assume that the composed terms are fully normal, which seems an unnatural assumption that is nonetheless mandatory.

Consequently, de Carvalho's technique does not apply to all terms having non-empty interpretation in the relational model. Here, we point out the problem, which is a first essential step. We do not aim at solving it, because it seems to require the development of a new approach, not just a refinement of de Carvalho's. Abstractly, the limits of his technique come from the fact that external information is measured by reducing it to internal information, rather than measuring it *separately* from it, which would allow one to measure external information even when the internal evaluation process does not fully terminate.

Proofs. A few proofs are omitted and can be found in the Proof Appendix available on ArXiv [1].

2 Head and Leftmost Reductions and Normal Forms

Basics of λ. The set Λ of untyped λ-terms is defined by $t ::= x \mid \lambda x.t \mid t\,t$, which are considered as quotiented by α-equivalence. The capture-avoiding substitution of x by u in t is written $t\{x \leftarrow u\}$.

β-Reduction and Normal Forms. β-reduction $\to_\beta \subseteq \Lambda \times \Lambda$ is defined as follows:

β-Reduction

$$\frac{}{(\lambda x.t)u \to_\beta t\{x \leftarrow u\}}\,ax \qquad \frac{t \to_\beta t'}{\lambda x.t \to_\beta \lambda x.t'}\,\lambda \qquad \frac{t \to_\beta t'}{tu \to_\beta t'u}\,@l \qquad \frac{t \to_\beta t'}{ut \to_\beta ut'}\,@r$$

It is well known that β-reduction is non-deterministic but confluent, that is, a term can have at most one normal form. Normal forms (for β) are described by the following grammar relying on the mutually inductive notion of neutral term:

$$\text{Neutral terms} \quad n ::= x \mid nf \qquad\qquad \text{Normal forms} \quad f ::= n \mid \lambda x.f$$

An alternative streamlined definition of normal forms is $f := \lambda x_1 \dots \lambda x_n.(y f_1 \dots f_k)$ with $n, k \geq 0$, y possibly equal to one of the x_i, and the terms f_j normal forms themselves.

For our quantitative study, we need a notion of size of normal forms. We use the following *inner* one, which counts the number of inner nodes of a term, when seen as a tree having variables as leaves, as it is the one that best matches what shall be measured via multi types.

▶ **Definition 1** (Inner size). *The inner size of λ-terms is defined as follows:*

$$|x|_{\mathtt{in}} := 0 \qquad |\lambda x.u|_{\mathtt{in}} := |u|_{\mathtt{in}} + 1 \qquad |ur|_{\mathtt{in}} := |u|_{\mathtt{in}} + |r|_{\mathtt{in}} + 1$$

Head Reduction. A deterministic notion of evaluation for λ-terms is *head reduction*, which reduces only the left branch of a term, when seen as a tree. The definition follows (it is obtained by omitting rule $@r$ in the definition of β, and constraining t not to be an abstraction in rule $@l$):

$$\text{Head reduction}$$

$$\frac{}{(\lambda x.t)u \to_{\mathtt{h}} t\{x \leftarrow u\}}\, ax \qquad \frac{t \to_{\mathtt{h}} t'}{\lambda x.t \ \to_{\mathtt{h}} \lambda x.t'}\, \lambda \qquad \frac{t \to_{\mathtt{h}} t' \quad t \neq \lambda x.r}{tu \to_{\mathtt{h}} t'u}\, @l$$

Let $\mathsf{I} := \lambda z.z$ be the identity combinator, $\delta := \lambda x.xx$ be the duplicator, and consider the following examples: $\delta\mathsf{I}t \to_{\mathtt{h}} \mathsf{I}\mathsf{I}t$ and $\lambda y.(\delta\mathsf{I}) \to_{\mathtt{h}} \lambda y.\mathsf{I}\mathsf{I}$ but $(\lambda y.(\delta\mathsf{I}))t \not\to_{\mathtt{h}} (\lambda y.\mathsf{I}\mathsf{I})t$, as it rather reduces to $(\delta\mathsf{I})\{y \leftarrow t\} = \delta\mathsf{I}$.

Head reduction might not compute normal forms, since it does not evaluate arguments. Its notion of normal form follows:

$$\text{Head normal forms} \quad h ::= \lambda x_1 \dots \lambda x_n.(y t_1 \dots t_k) \text{ with } n, k \geq 0 \text{ and } y \text{ possibly equal to one of the } x_i.$$

We shall also need a notion of *head size* for head normal forms defined as $|h|_{\mathtt{h}} := n + k$ if $h = \lambda x_1 \dots \lambda x_n.(y t_1 \dots t_k)$.

Leftmost Reduction. Leftmost-outermost reduction $\to_{\mathtt{lo}}$ (shortened to *leftmost*) is a deterministic extension of head reduction as to reduce arguments and reach normal forms. The definition relies on the notion of neutral term n used to describe normal forms.

$$\text{Leftmost(-Outermost) Reduction}$$

$$\frac{}{(\lambda x.t)u \to_{\mathtt{lo}} t\{x \leftarrow u\}}\, ax \qquad \frac{t \to_{\mathtt{lo}} t'}{\lambda x.t \ \to_{\mathtt{lo}} \lambda x.t'}\, \lambda$$

$$\frac{t \to_{\mathtt{lo}} t' \quad t \neq \lambda x.r}{tu \to_{\mathtt{lo}} t'u}\, @l \qquad \frac{n \text{ is neutral} \quad t \to_{\mathtt{lo}} t'}{nt \to_{\mathtt{lo}} nt'}\, @r$$

Examples: $x(\mathsf{I}\mathsf{I})(\mathsf{I}\mathsf{I}) \to_{\mathtt{lo}} x\mathsf{I}(\mathsf{I}\mathsf{I})$ but $x(\mathsf{I}\mathsf{I})(\mathsf{I}\mathsf{I}) \not\to_{\mathtt{lo}} x(\mathsf{I}\mathsf{I})\mathsf{I}$ and $\delta(\mathsf{I}\mathsf{I})(\mathsf{I}\mathsf{I}) \not\to_{\mathtt{lo}} \delta\mathsf{I}(\mathsf{I}\mathsf{I})$.

Leftmost normal forms are simply normal forms and – crucially – leftmost reduction is *normalizing*, that is, if t has a β-reduction $t \to_\beta^* f$ to normal form then leftmost reduction reaches that normal form, that is, $t \to_{\mathtt{lo}}^* f$. For a recent simple proof of this classic result see Accattoli et al. [6].

$$\frac{}{x:[\mathsf{L}]\vdash x:\mathsf{L}}\,\mathsf{ax} \qquad \frac{\Gamma\vdash t:\mathsf{L}}{\Gamma\setminus\!\setminus x\vdash \lambda x.t:\Gamma(x)\multimap\mathsf{L}}\,\lambda \qquad \frac{\Gamma\vdash t:\mathsf{M}\multimap\mathsf{L}\quad \Delta\vdash u:\mathsf{M}}{\Gamma\uplus\Delta\vdash tu:\mathsf{L}}\,@ \qquad \frac{[\Gamma_i\vdash t:\mathsf{L}_i]_{i\in I}}{\biguplus_{i\in I}\Gamma_i\vdash t:[\mathsf{L}_i]_{i\in I}}\,\mathsf{many}$$

Figure 1 De Carvalho's multi type system.

3 Multi Types, Head Reduction, and Bounds From Type Derivations

In this section, we give our presentation of de Carvalho's system of multi types, and recall some results from the literature. In particular, we recall the characterization of head reduction, how multi types induce the relational model, and the bounds that can be extracted from type derivations for the length of head evaluations and the head size of head normal forms.

Multi Types. There are two layers of types, *linear* and *multi types*, built over a countably infinite set of (linear) type variables:

$$\begin{array}{rrcl} \textsc{Linear type variables} & \mathtt{TyVars} & ::= & \{\mathsf{X},\mathsf{Y},\mathsf{Z},\mathsf{W},\mathsf{X}_1,\mathsf{Y}',\mathsf{Z}_2,\dots\} \\ \textsc{Linear types} & \mathsf{L},\mathsf{L}' & ::= & \mathsf{X}\in\mathtt{TyVars}\mid \mathsf{M}\multimap\mathsf{L} \\ \textsc{Multi types} & \mathsf{M},\mathsf{N} & ::= & [\mathsf{L}_1,\dots,\mathsf{L}_n]\quad n\in\mathbb{N} \\ \textsc{Generic types} & \mathsf{T},\mathsf{T}' & ::= & \mathsf{L}\mid\mathsf{M} \end{array}$$

where $[\mathsf{L}_1,\dots,\mathsf{L}_n]$ is our notation for finite multisets. The *empty* multi type $[\,]$ obtained by taking $n=0$ is also denoted by $\mathbf{0}$. Often, multi types are presented using a single type variable X instead of countably many. Most results are unaffected, but we shall see that for de Carvalho's semantic bounds we need countably many type variables.

A multi type $[\mathsf{L}_1,\dots,\mathsf{L}_n]$ has to be intended as a conjunction $\mathsf{L}_1\wedge\cdots\wedge\mathsf{L}_n$ of linear types $\mathsf{L}_1,\dots,\mathsf{L}_n$, for a commutative, associative, non-idempotent conjunction $\wedge$ (morally a tensor $\otimes$), of neutral element $\mathbf{0}$. The intuition is that a linear type corresponds to a single use of a term t, and that t is typed with a multiset M of n linear types if it is going to be used (at most) n times, that is, if t is part of a larger term u, then a copy of t shall end up in evaluation position during the evaluation of u.

Typing Rules. The derivation rules for the multi types system are in Figure 1. *Judgments* have shape $\Gamma\vdash t:\mathsf{M}$ or $\Gamma\vdash t:\mathsf{L}$ where t is a term, M is a multi type, L is a linear type, and Γ is a *type context*, that is, a total function from variables to multi types such that $\mathsf{dom}(\Gamma):=\{x\mid\Gamma(x)\neq\mathbf{0}\}$ is finite, usually represented as $x_1:\mathsf{M}_1,\dots,x_n:\mathsf{M}_n$ (for some $n\in\mathbb{N}$) if $\mathsf{dom}(\Gamma)\subseteq\{x_1,\dots,x_n\}$ and $\Gamma(x_i)=\mathsf{M}_i$ for all $1\leq i\leq n$.

The abstraction rule λ uses the notation $\Gamma\setminus\!\setminus x$ for the type context defined as Γ on every variable but possibly x, for which $(\Gamma\setminus\!\setminus x)(x)=\mathbf{0}$. It is a compact way to express the rule in both the cases $x\in\mathsf{dom}(\Gamma)$ and $x\notin\mathsf{dom}(\Gamma)$. Note that the application rule @ requires the argument to be typed with a multi type M, which is necessarily introduced by rule many, the hypotheses of which are a multi set of derivations, indexed by a possibly empty set I. When I is empty, the rule has no premises and can type every term. For instance, $\vdash\Omega:\mathbf{0}$ is derivable, but no linear type can be assigned to Ω. Essentially, $\mathbf{0}$ is the type of erasable terms, and every term is erasable in the λ-calculus.

Technicalities about Types. The type context Γ is *empty* if $\mathsf{dom}(\Gamma)=\emptyset$. *Multi-set sum* $\uplus$ is extended to type contexts point-wise, *i.e.* $(\Gamma\uplus\Delta)(x):=\Gamma(x)\uplus\Delta(x)$ for each variable x. This notion is extended to a finite family of type contexts as expected, in particular $\biguplus_{i\in J}\Gamma_i$ is the empty context when $J=\emptyset$. Given two type contexts Γ and Δ such that

$\mathsf{dom}(\Gamma) \cap \mathsf{dom}(\Delta) = \emptyset$, the type context Γ, Δ is defined by $(\Gamma, \Delta)(x) := \Gamma(x)$ if $x \in \mathsf{dom}(\Gamma)$, $(\Gamma, \Delta)(x) := \Delta(x)$ if $x \in \mathsf{dom}(\Delta)$, and $(\Gamma, \Delta)(x) := \mathbf{0}$ otherwise. Note that $\Gamma, x : \mathbf{0} = \Gamma$, where we implicitly assume $x \notin \mathsf{dom}(\Gamma)$.

Type Derivations. We write $\Phi \rhd \Gamma \vdash t : \mathsf{T}$ if Φ is a (*type*) *derivation* (*i.e.* a tree constructed using the rules in Figure 1) with conclusion the multi judgment $\Gamma \vdash t : \mathsf{T}$. In particular, we write $\Phi \rhd \vdash t : \mathsf{T}$ when Γ is empty. We write $\Phi \rhd t$ if $\Phi \rhd \Gamma \vdash t : \mathsf{T}$ for some type context Γ and some type T.

We need a notion of size of type derivations, which shall be used to extract bounds for the number of evaluation steps and the size of normal forms.

▶ **Definition 2** (Inner size of derivations). *Let Φ be a type derivation. The* inner size $|\Phi|_{\mathtt{in}}$ *of Φ is the number of occurrences of rules λ and @ in Φ.*

Subject Reduction and Expansion, and Relational Semantics. The first properties of the type system that we recall are subject reduction and expansion, which hold for *every* β-step.

▶ **Proposition 3.** *Let $t \to_\beta t'$.*

1. Subject reduction*: if $\Phi \rhd \Gamma \vdash t : \mathsf{L}$ then there is a derivation $\Phi' \rhd \Gamma \vdash t' : \mathsf{L}$ such that $|\Phi'|_{\mathtt{in}} \leq |\Phi|_{\mathtt{in}}$. Moreover, if $t \to_{\mathtt{h}} t'$ then $|\Phi'|_{\mathtt{in}} = |\Phi|_{\mathtt{in}} - 2$.*
2. Subject expansion*: if $\Phi' \rhd \Gamma \vdash t' : \mathsf{L}$ then there is a derivation $\Phi \rhd \Gamma \vdash t : \mathsf{L}$.*

Together, subject reduction and expansion state that type judgements are *invariants* of β-reduction. Such invariants actually induce a denotational model of the λ-calculus, its (call-by-name) *relational semantics*.

Let t be a term and $x_1, \dots, x_n$ ($n \geq 0$) be pairwise distinct variables. The list $\vec{x} = (x_1, \dots, x_n)$ is *suitable for* t if $\mathsf{fv}(t) \subseteq \{x_1, \dots, x_n\}$. If $\vec{x} = (x_1, \dots, x_n)$ is suitable for t, the *relational semantics* $[\![t]\!]_{\vec{x}}$ *of* t *for* $\vec{x}$ is defined by:

$$[\![t]\!]_{\vec{x}} \quad := \quad \{((\mathsf{M}_1, \dots, \mathsf{M}_n), \mathsf{L}) \mid \exists\, \Phi \rhd x_1 : \mathsf{M}_1, \dots, x_n : \mathsf{M}_n \vdash t : \mathsf{L}\}.$$

The following property is an immediate corollary of subject reduction and expansion.

▶ **Proposition 4** (Invariance). *Let $\vec{x} = (x_1, \dots, x_n)$ be suitable for two terms t and u. If $t \to_\beta u$ then $[\![t]\!]_{\vec{x}} = [\![u]\!]_{\vec{x}}$.*

Characterizing Head Termination. Note the quantitative aspect of subject reduction (Prop. 3.1), stating that the derivation size *cannot increase* after a reduction step, and that it *decreases* with head steps (of 2 because removing a head β redex removes a λ and a @ rule). It does not say that it decreases at *every* step because steps occurring in sub-terms typed with rule many might not change the size. For instance, if $xt \to xt'$ and t is typed using a empty many rule (*i.e.* with 0 premises), which is a sub-derivation of size 0, then also t' is typed using a empty many rule, of size 0. Therefore, not all typable terms terminate, as for instance $x\Omega$ is typable as follows, for any linear type L, but it has no normal form:

$$\dfrac{\dfrac{}{x : [[\mathbf{0}] \multimap \mathsf{L}] \vdash x : [\mathbf{0}] \multimap \mathsf{L}}\,\mathsf{ax} \qquad \dfrac{}{\vdash \Omega : \mathbf{0}}\,\mathsf{many}}{x : [[\mathbf{0}] \multimap \mathsf{L}] \vdash x\Omega : \mathsf{L}}\,@ \tag{1}$$

[0] More precisely, such a model is the restriction of the relational model for lineal logic to the image of Girard's [36] call-by-name translation $(A \Rightarrow B)^{\mathsf{n}} = !A^{\mathsf{n}} \multimap B^{\mathsf{n}}$ of the intuitionistic arrow into linear logic.

Since the size of type derivations decreases at every head step, it provides a termination measure for the head reduction of typable terms. Therefore, typable terms are head terminating – this is also called *correctness* of the type system (with respect to head reduction). Conversely, every head normal forms is typable. Additionally, one can show that the size of type derivations bounds the head size of head normal forms, and that there exists derivations having exactly the head size of the normal form, as in the example (1) above.

▶ **Proposition 5** (Typability of head normal forms). *Let h be a head normal form.*
1. Lax bounds for all pairs*: if $\Phi \rhd \Gamma \vdash h : \mathsf{L}$ then $|h|_{\mathtt{h}} \leq |\Phi|_{\mathtt{in}}$;*
2. Existence and exact bounds*: there exists a derivation $\Phi \rhd \Gamma \vdash h : \mathsf{L}$ such that $|h|_{\mathtt{h}} = |\Phi|_{\mathtt{in}}$.*

Typability of all head normal forms (Prop. 5.2) together with subject expansion (Prop. 3.2) implies the *completeness* of the type system: every head terminating term is typable.

▶ **Theorem 6** (Typability characterizes head normalization).
1. Correctness*: if $\Phi \rhd t$ then there exists a head normalizing evaluation $t \to_{\mathtt{h}}^{n} h$ with h normal and $2n + |h|_{\mathtt{h}} \leq |\Phi|_{\mathtt{in}}$.*
2. Completeness*: if $t \to_{\mathtt{h}}^{n} h$ is a head normalizing evaluation, then there exists a derivation $\Phi \rhd t$. In particular, there is a derivation Φ for which $2n + |h|_{\mathtt{h}} = |\Phi|_{\mathtt{in}}$.*

The quantitative bounds involve $2n$ rather than n because every β redex is typed in a type derivation Φ using two rules (λ and @). The type derivation captures only the head size of the normal form, because in general it ignores arguments, and so it cannot catch the inner size. For instance, for the head normal form $x\Omega$ of head size $|x\Omega|_{\mathtt{h}} = 1$ (but inner size $|x\Omega|_{\mathtt{in}} = 6$) the derivation (1) has inner size 1.

The head characterization theorem implies the following property of the semantics.

▶ **Theorem 7** (Adequacy of relational semantics for head reduction). *Let $\vec{x} = (x_1, \dots, x_n)$ be suitable for t. Then $[\![t]\!]_{\vec{x}}$ is non-empty if and only if t is $\to_{\mathtt{h}}$-normalizing.*

Summing up, multi types naturally model head reduction. De Carvalho's bounds from composable types, however, rest on normal forms, which are reached by leftmost reduction, rather than on head normal forms and head reduction. Therefore, the next section recalls how multi types relate to leftmost reduction and normal forms.

4 Bounds From Derivations Via (Unitary) Shrinking

In this section, we recall how to extend the results of the previous section to leftmost reduction $\to_{\mathtt{lo}}$ and full normal forms, via the so called *shrinking* constraint. We follow the presentation of Accattoli et al. [7] (removing some of the aspects of their work that are not relevant here), but the definition of shrinking judgements is standard and not due to [7], see for instance Krivine's book [44], de Carvalho [44, 27], Kesner and Ventura [40], or Bucciarelli et al. [20].

The Need for Shrinking. Consider the derivation of end sequent $x : [\mathbf{0} \multimap \mathsf{L}] \vdash x\Omega : \mathsf{L}$ in (1). Since $x\Omega$ is $\to_{\mathtt{lo}}$-diverging, this derivation has to be excluded somehow. The problem here is that since x has an erasing type – that is an arrow type with $\mathbf{0}$ on the left – then the diverging subterm Ω does not get typed. Excluding the use of $\mathbf{0}$ is too drastic, because the paradigmatic erasing term $\lambda y.x$ is normal and can be typed only with $x : [\mathsf{L}] \vdash \lambda y.x : \mathbf{0} \multimap \mathsf{L}$.

The idea is that only *some* occurrences of $\mathbf{0}$ are dangerous. The given examples seem to suggest that if $\mathbf{0}$ occurs on the right side of $\vdash$ it is fine, while if it occurs in the typing context it is not. Things are subtler. Extending example (1) with an abstraction, one obtains the $\to_{\mathtt{lo}}$-diverging term $\lambda x.x\Omega$ and the typing $\vdash \lambda x.x\Omega : [\mathbf{0} \multimap \mathsf{L}] \multimap \mathsf{L}$, that show that $\mathbf{0}$ can be

dangerous also on the right of $\vdash$. The dangerous occurrences of $\mathbf{0}$ turn out to be those on the left of an *even number of arrows*, considering the $\vdash$ symbols as an arrow. This is formalized by the *shrinking* constraint, which allows one to characterize leftmost termination.

Defining Shrinking. There are two mutually defined notions of shrinking types, *left* and *right*, the key point of which is that right multi types *cannot be empty* (note $n \geq 1$), so that $\mathbf{0}$ is forbidden on the left of top arrows $\multimap$ for left linear types. Their definition follows:

Left and right (shrinking) types

$$\begin{array}{rrcl} \text{Right linear type} & \mathsf{L}^{\mathrm{R}} & ::= & \mathsf{X} \in \mathtt{TyVars} \mid \mathsf{M}^{\mathrm{L}} \multimap \mathsf{L}^{\mathrm{R}} \\ \text{Left linear type} & \mathsf{L}^{\mathrm{L}} & ::= & \mathsf{X} \in \mathtt{TyVars} \mid \mathsf{M}^{\mathrm{R}} \multimap \mathsf{L}^{\mathrm{L}} \\ \text{Right multi type} & \mathsf{M}^{\mathrm{R}} & ::= & [\mathsf{L}^{\mathrm{R}}_1, \dots, \mathsf{L}^{\mathrm{R}}_n] \quad n \geq 1 \\ \text{Left multi type} & \mathsf{M}^{\mathrm{L}} & ::= & [\mathsf{L}^{\mathrm{L}}_1, \dots, \mathsf{L}^{\mathrm{L}}_n] \quad n \geq 0 \end{array}$$

The notions extend to type contexts and to derivations as follows:

- A type context $x_1 : \mathsf{M}_1, \dots, x_n : \mathsf{M}_n$ is *left* if each M_i is left;
- A derivation $\Phi \rhd \Gamma \vdash t : \mathsf{L}$ is *shrinking* if Γ is left and L is right.

For instance, $[\mathsf{X}]$ is both left and right, while $\mathbf{0}$ is left but not right, and $[\mathbf{0} \multimap \mathsf{X}]$ is right but not left. Note that the derivation in (1) is not shrinking. By adding the shrinking constraint, we can now characterize leftmost normalization with multi types, with quantitative bounds involving the inner size of the normal form.

▶ **Theorem 8** (Shrinking typability characterizes leftmost normalization, [7]).

1. Correctness*: if $\Phi \rhd t$ is a shrinking derivation, then there exists a normalizing evaluation $t \to_{\mathtt{lo}}^{n} f$ with f normal and $2n + |f|_{\mathtt{in}} \leq |\Phi|_{\mathtt{in}}$.*
2. Completeness*: if $t \to_{\mathtt{lo}}^{*} f$ is a normalizing evaluation, then there exists a shrinking derivation $\Phi \rhd t$.*

Unitary Shrinking. Shrinking is enough to ensure termination, but not to capture the exact number of steps to normal form together with the exact size of the normal form. The point is somewhat dual to shrinkingness, as it concerns arguments that have to be typed, but that should not be typed *too many times*. Consider the evaluation $y(\mathsf{I}z) \to_{\mathtt{lo}} yz$ that involves 1 leftmost step and produces a normal form of inner size 1. The following shrinking derivation types the argument $\mathsf{I}z$ twice (the easy derivation of $z : [\mathsf{X}] \vdash \mathsf{I}z : \mathsf{X}$ of inner size 2 is omitted), instead of once, and it has size 5, instead of the required 3 (obtained as 2*1+1):

$$\dfrac{\dfrac{}{y : [[\mathsf{X},\mathsf{X}] \multimap \mathsf{Y}] \vdash y : [\mathsf{X},\mathsf{X}] \multimap \mathsf{Y}}\,\mathsf{ax} \qquad \dfrac{[z : [\mathsf{X}] \vdash \mathsf{I}z : \mathsf{X}]_{i=1,2}}{z : [\mathsf{X},\mathsf{X}] \vdash \mathsf{I}z : [\mathsf{X},\mathsf{X}]}\,\mathsf{many}}{y : [[\mathsf{X},\mathsf{X}] \multimap \mathsf{Y}], z : [\mathsf{X},\mathsf{X}] \vdash y(\mathsf{I}z) : \mathsf{Y}}\,@ \tag{2}$$

To obtain exact bounds, one needs *unitary shrinking* types and derivations, that type arguments of normal forms only once, obtained by constraining some multi-sets – the right ones – to be singletons. The definition follows:

Unitary left and right (shrinking) types

$$\begin{array}{rrcl} \text{Unitary right linear types} & \mathsf{L}^{\mathrm{UR}} & ::= & \mathsf{X} \in \mathtt{TyVars} \mid \mathsf{M}^{\mathrm{UL}} \multimap \mathsf{L}^{\mathrm{UR}} \\ \text{Unitary left linear types} & \mathsf{L}^{\mathrm{UL}} & ::= & \mathsf{X} \in \mathtt{TyVars} \mid \mathsf{M}^{\mathrm{UR}} \multimap \mathsf{L}^{\mathrm{UL}} \\ \text{Unitary right multi types} & \mathsf{M}^{\mathrm{UR}} & ::= & [\mathsf{L}^{\mathrm{UR}}] \\ \text{Unitary left multi types} & \mathsf{M}^{\mathrm{UL}} & ::= & [\mathsf{L}^{\mathrm{UL}}_1, \dots, \mathsf{L}^{\mathrm{UL}}_n] \quad n \geq 0 \end{array}$$

The notions extend to type contexts and to derivations as expected:

- A type context $x_1 : \mathsf{M}_1, \ldots, x_n : \mathsf{M}_n$ is *unitary left* if each M_i is unitary left;
- A derivation $\Phi \rhd \Gamma \vdash t : \mathsf{L}$ is *unitary shrinking* if Γ is unitary left and L is unitary right.

For instance, the derivation in (2) is not unitary shrinking, because the multi type $[[\mathsf{X}, \mathsf{X}] \multimap \mathsf{Y}]$ of y is not unitary left, since $[\mathsf{X}, \mathsf{X}]$ is not unitary right. A derivable unitary shrinking typing for $y(\mathsf{I}z)$ is $y : [[\mathsf{X}] \multimap \mathsf{Y}], z : [\mathsf{X}] \vdash y(\mathsf{I}z) : \mathsf{Y}$, obtained via a derivation of inner size 3.

The following refinement of the shrinking characterization theorem (Thm. 8) holds.

▶ **Theorem 9** (Unitary shrinking typability measures leftmost evaluation, [7]).

1. Correctness*: if $\Phi \rhd t$ is a unitary shrinking derivation, then there exists a normalizing evaluation $t \to_{\mathtt{lo}}^{n} f$ with f normal and $2n + |f|_{\mathtt{in}} = |\Phi|_{\mathtt{in}}$.*
2. Completeness*: if $t \to_{\mathtt{lo}}^{*} f$ is a normalizing evaluation, then there exists a unitary shrinking derivation $\Phi \rhd t$.*

Normal Forms. The proof of the last theorem rests on two properties of normal forms that it is useful to state explicitly, for comparison with the study of the next sections.

▶ **Proposition 10** (Unitary shrinking derivations and normal forms, [7]). *Let f be normal.*

1. Lax bounds*: if $\Phi \rhd f$ is a shrinking derivation then $|f|_{\mathtt{in}} \leq |\Phi|_{\mathtt{in}}$;*
2. Exact bounds*: there exists a unitary shrinking derivation $\Phi \rhd f$ such that $|f|_{\mathtt{in}} = |\Phi|_{\mathtt{in}}$.*

5 Bounds from Types

In this section, we recall the bounds on the size of normal forms that can be extracted from *types* rather than from *type derivations*.

Bounds from Types. The types appearing in the final judgement of a shrinking derivation for t bound the inner size $|f|_{\mathtt{in}}$ of the normal form f of t, according to a notion of *type size* given below, and independently of the derivation itself. For example, consider the easily derivable (unitary shrinking) derivation $\Phi \rhd\vdash \delta : [[\mathsf{X}] \multimap \mathsf{X}, \mathsf{X}] \multimap \mathsf{X}$ for $\delta = \lambda x.xx$. There are two arrows in the type (judgement) and the normal form has inner size two. Of course, one also has to take into account the arrow symbols appearing in the typing context, when present.

Note, however, that types – even unitary shrinking ones – in general do not provide exact bounds: taking the derivation of Φ for δ and substituting X with $[\mathsf{Y}] \multimap \mathsf{Y}$ everywhere in Φ one obtains a unitary shrinking derivation Ψ having the same size of Φ but final (still unitary shrinking) judgement:

$$\Psi \rhd \vdash \delta : [[[\mathsf{Y}] \multimap \mathsf{Y}] \multimap [\mathsf{Y}] \multimap \mathsf{Y}, [\mathsf{Y}] \multimap \mathsf{Y}] \multimap [\mathsf{Y}] \multimap \mathsf{Y}$$

which has six arrows while $|\delta|_{\mathtt{in}} = 2$.

▶ **Definition 11** (Type size). *The size $|\cdot|$ of types and typing contexts is defined as follows:*

$$\begin{array}{lllll} \textsc{Types} & |\mathsf{X}| := 0 & |\mathsf{M} \multimap \mathsf{L}| := |\mathsf{M}| + |\mathsf{L}| + 1 & |[\mathsf{L}_1, \ldots, \mathsf{L}_n]| := \sum_{i=1}^{n} |\mathsf{L}_i| \\ \textsc{Type ctxs} & |\epsilon| := 0 & |x : \mathsf{M}; \Gamma| := |\mathsf{M}| + |\Gamma| & \end{array}$$

Clearly, $|\mathsf{T}| \geq 0$ and $|\mathsf{M}| = 0$ if and only if M is a possibly empty multi set of type variables.

Given a type context $\Gamma = x_1 : \mathsf{M}_1, \ldots, x_n : \mathsf{M}_n$ we often consider the list of its types, noted $\hat{\Gamma} := (\mathsf{M}_1, \ldots, \mathsf{M}_n)$. Since any list of multi types $(\mathsf{M}_1, \ldots, \mathsf{M}_n)$ can be seen as extracted from a type context Γ, we use the notation $\hat{\Gamma}$ for lists of multi types. The *size* of a list of multi types is $|(\mathsf{M}_1, \ldots, \mathsf{M}_n)| := \sum_{i=1}^{n} |\mathsf{M}_i|$, and that of the conclusion of a derivation $\pi \rhd \Gamma \vdash e : \mathsf{L}$ is $|(\hat{\Gamma}, \mathsf{L})| := |\hat{\Gamma}| + |\mathsf{L}|$. Clearly, $\mathsf{dom}(\Gamma) = \emptyset$ implies $|\hat{\Gamma}| = 0$.

▶ **Proposition 12** (Shrinking types bound the size of normal forms, [7]). *Let f be a normal form.*

1. Lax bounds for all types*: if $\Phi \rhd \Gamma \vdash f : \mathsf{L}$ is a shrinking derivation then $|f|_{\mathtt{in}} \leq |(\hat{\Gamma}, \mathsf{L})|$;*
2. Exact bounds for special types*: there exists a unitary shrinking derivation $\Phi \rhd \Gamma \vdash f : \mathsf{L}$ such that $|f|_{\mathtt{in}} = |(\hat{\Gamma}, \mathsf{L})|$.*

6 Dissecting Bounds From Types via Skeletons and Dry Judgements

In this section, we decompose and elaborate over the bounds on the size of normal forms extracted from types given in the previous section. The analysis is the main contribution of this paper. In particular, we develop notions and tools that shall be used in the next section to understand the issues concerning how to extract exact bounds from composable types.

Types Bound the Size of Derivations. The first observation is that the lax bounds of Prop. 12.1 are a consequence of the more general fact that types bound the size of derivations, proved next, together with the already proved fact that derivations bound the size of normal forms (Prop. 10). The second point of the following proposition is the main statement, the first one is an auxiliary one that is needed for the proof to go through.

▶ **Proposition 13** (Types bound the size of derivations for normal forms). *Let $\Phi \rhd \Gamma \vdash t : \mathsf{T}$ be a derivation.*

1. Neutral*: if t is a neutral term then $|\Phi|_{\mathtt{in}} \leq |\hat{\Gamma}| - |\mathsf{T}|$.*
2. Normal*: if t is a normal form then $|\Phi|_{\mathtt{in}} \leq |\hat{\Gamma}| + |\mathsf{T}|$.*

Proof. By mutual induction on the definition of neutral and normal terms, followed by an induction on the type derivation Φ.

1. *t is a neutral term.* Cases of the last rule:
   - *Rule* many. Then T is a multi type $\mathsf{M} = [\mathsf{L}_i]_{i\in I}$ and the last rule is necessarily many. So, necessarily, for some finite set of indices I,

     $$\Phi = \dfrac{[\,\Phi_i \rhd \Gamma_i \vdash t : \mathsf{L}_i]_{i\in I}}{\uplus_{i\in I}\Gamma_i \vdash t : [\mathsf{L}_i]_{i\in I}}\,\mathsf{many}$$

     where $\Gamma = \uplus_{i\in I}\Gamma_i$. By *i.h.* (on Φ_i), $|\Phi_i|_{\mathtt{in}} \leq |\hat{\Gamma_i}| - |\mathsf{L}_i|$, thus $|\Phi|_{\mathtt{in}} = \sum_{i\in I} |\Phi_i|_{\mathtt{in}} \leq \sum_{i\in I} |\hat{\Gamma_i}| - \sum_{i\in I} |\mathsf{L}_i| = |\widehat{\uplus_{i\in I}\Gamma_i}| - |[\mathsf{L}_i]_{i\in I}| = |\hat{\Gamma}| - |\mathsf{M}|$.
   - *Rule* ax, that is, $t = x$. Then:

     $$\Phi = \dfrac{}{x : [\mathsf{L}] \vdash x : \mathsf{L}}\,\mathsf{ax}$$

     where $\mathsf{T} = \mathsf{L}$ and $\Gamma = x : [\mathsf{L}]$. Since $|\Phi|_{\mathtt{in}} = 0$ and $|\mathsf{T}| = |\mathsf{L}| = |[\mathsf{L}]| = |\hat{\Gamma}|$, then $|\Phi|_{\mathtt{in}} = 0 = |\hat{\Gamma}| - |\mathsf{T}|$.
   - *Rule* @, that is, $t = nf$. Then necessarily:

     $$\Phi = \dfrac{\Phi_n \rhd \Delta \vdash n : \mathsf{N} \multimap \mathsf{L} \qquad \Phi_f \rhd \Sigma \vdash f : \mathsf{N}}{\Delta \uplus \Sigma \vdash nf : \mathsf{L}}\,@$$

     where $\mathsf{T} = \mathsf{L}$ and $\Gamma = \Delta \uplus \Sigma$. By *i.h.* (on the definition of neutral terms and normal forms), $|\Phi_n|_{\mathtt{in}} \leq |\hat{\Delta}| - |\mathsf{N} \multimap \mathsf{L}| = |\hat{\Delta}| - |\mathsf{N}| - |\mathsf{L}| - 1$ and $|\Phi_f|_{\mathtt{in}} \leq |\hat{\Sigma}| + |\mathsf{N}|$. Therefore,

     $$\begin{array}{rcll} |\Phi|_{\mathtt{in}} & = & |\Phi_n|_{\mathtt{in}} + |\Phi_f|_{\mathtt{in}} + 1 & \leq_{i.h.} \; |\hat{\Delta}| - |\mathsf{N}| - |\mathsf{L}| - 1 + |\Phi_f|_{\mathtt{in}} + 1 \\ & = & |\hat{\Delta}| - |\mathsf{N}| - |\mathsf{L}| + |\Phi_f|_{\mathtt{in}} & \leq_{i.h.} \; |\hat{\Delta}| - |\mathsf{N}| - |\mathsf{L}| + |\hat{\Sigma}| + |\mathsf{N}| \\ & = & |\hat{\Delta}| + |\hat{\Sigma}| - |\mathsf{L}| & = \; |\hat{\Gamma}| - |\mathsf{T}|. \end{array}$$

2. *t is a normal form.* If t is a neutral term, then the statement follows from Point 1, which stronger than the statement that we need to prove. Otherwise, t is an abstraction. If the last rule is many then we reason exactly as for neutral terms. The remaining cases is when the last rule is λ, that is, $t = \lambda x.f$ and Φ is necessarily of the form:

$$\frac{\Phi' \rhd \Delta \vdash f : \mathsf{L}'}{\Delta \setminus\!\setminus x \vdash \lambda x.f : \Delta(x) \multimap \mathsf{L}'}\,\lambda$$

where $\mathsf{T} = \Delta(x) \multimap \mathsf{L}'$ and $\Gamma = \Delta \setminus\!\setminus x$. By *i.h.*,

$$|\Phi'|_{\mathtt{in}} \leq_{i.h.} \quad |\hat{\Delta}| + |\mathsf{L}'| \quad = \quad |\Delta \mathbin{\hat{\setminus\!\setminus}} x| + |\Delta(x)| + |\mathsf{L}'| \quad = \quad |\Delta \mathbin{\hat{\setminus\!\setminus}} x| + |\Delta(x) \multimap \mathsf{L}'| - 1$$

Therefore,

$$\begin{array}{rclcl} |\Phi|_{\mathtt{in}} & = & |\Phi'|_{\mathtt{in}} + 1 & \leq & |\Delta \mathbin{\hat{\setminus\!\setminus}} x| + |\Delta(x) \multimap \mathsf{L}'| - 1 + 1 \\ & = & |\Delta \mathbin{\hat{\setminus\!\setminus}} x| + |\Delta(x) \multimap \mathsf{L}'| & = & |\hat{\Gamma}| + |\mathsf{T}|. \end{array}$$

◀

Note that the bound holds for *every* derivation, without requiring them to be shrinking. This fact means that the connection between types and derivations is stronger than the one between derivations and normal forms. Note also that the bound does *not* hold for *head* normal forms, as can be seen by inspecting examples (1) (p. 7) and (2) (p. 9).

Exact Bounds from Types. We now turn our attention to exact bounds. Having showed that bounds for normal forms factor through bounds for derivations (Prop. 13), we actually turn to exact bounds for *derivations*, from types. The idea, as usual, is that exact bounds are given by types of minimal size. To describe such minimal types we shall use a modified *dry* typing system for normal forms related to principal judgements, that shall derive only minimal types. Additionally, we use a relation between derivations having the same structure but assigning possibly different types, also considered by de Carvalho.

Skeleton Equivalence. We formalize the notion of derivations having the same *skeleton*, that is, the same *mute* structure. Skeleton equivalence $\sim$ relates derivations having the same rules arranged in the same way, but not necessarily having the same types.

▶ **Definition 14** (Skeleton equivalence). *Let t be a term. Two derivations $\Phi \rhd t$ and $\Psi \rhd t$ are* skeleton equivalent*, noted $\Phi \sim \Psi$, if they end with the same kind of rule and the derivations on the premises are $\sim$-equivalent, namely they fall in one of the following cases:*

- *Both Φ and Ψ are axioms.*
- *Both Φ and Ψ end with rule @, their two left premises Φ_l and Ψ_l satisfy $\Phi_l \sim \Psi_l$, and their right premises Φ_r and Ψ_r satisfy $\Phi_r \sim \Psi_r$ – similarly for rules λ.*
- *Both Φ and Ψ end with a rule* many *with n premises and there is a permutation ρ of $\{1, \ldots, n\}$ such that the i-th premise Φ_i of Φ and the $\rho(i)$-th premise $\Psi_{\rho(i)}$ of Ψ satisfy $\Phi_i \sim \Psi_{\rho(i)}$ for $i \in \{1, \ldots, n\}$.*

The next lemma shows that skeleton equivalence preserves more or less everything one can imagine, but types. We denote by $\#m$ the cardinality of a multiset m.

▶ **Lemma 15** (Skeletal invariants). *Let $\Phi \rhd \Gamma \vdash t : \mathsf{T}$ and $\Psi \rhd \Delta \vdash t : \mathsf{T}'$ be two derivations such that $\Phi \sim \Psi$. Then $|\Phi|_{\mathtt{in}} = |\Psi|_{\mathtt{in}}$, $\mathsf{dom}(\Gamma) = \mathsf{dom}(\Delta)$, $\#(\Gamma(x)) = \#(\Delta(x))$ for every variable x, and T is a multi type if and only if T' is, and in that case $\#\mathsf{T} = \#\mathsf{T}'$. Moreover, Φ is shrinking (resp. unitary shrinking) if and only if Ψ is.*

Proof. Straightforward induction on Φ. ◀

$$\frac{}{x:[\mathsf{X}] \vdash^{\mathsf{d}} x:\mathsf{X}}\,\mathsf{ax}^* \qquad \frac{\Phi \rhd \Gamma \vdash^{\mathsf{d}} n:\mathsf{X} \quad \Psi \rhd \Delta \vdash^{\mathsf{d}} f:\mathsf{M} \quad \mathsf{Y} \text{ fresh}, \Phi \# \Psi}{(\Gamma\{\mathsf{X}\leftarrow(\mathsf{M} \multimap \mathsf{Y})\} \uplus \Delta) \vdash^{\mathsf{d}} nf:\mathsf{Y}}\,@^*$$

$$\frac{\Gamma \vdash^{\mathsf{d}} f:\mathsf{L}}{\Gamma \setminus\!\setminus x \vdash^{\mathsf{d}} \lambda x.f:\Gamma(x) \multimap \mathsf{L}}\,\lambda^* \qquad \frac{\left[\Phi_i \rhd \Gamma_i \vdash^{\mathsf{d}} f:\mathsf{L}_i\right]_{i\in I} \quad \#_{i\in I}\,\Phi_i}{\biguplus_{i\in I}\Gamma_i \vdash^{\mathsf{d}} f:[\mathsf{L}_i]_{i\in I}}\,\mathsf{many}^*$$

Figure 2 Dry multi type system for normal forms.

Principal and Dry Judgements. Simple types admits *principal judgements* (or typings), that is, for every term t there exists a principal judgement $\Gamma \vdash t:A$ such that for every other judgement $\Delta \vdash t:B$ for t there exists a type substitution σ such that $\Gamma\sigma = \Delta$ and $A\sigma = B$. Multi types do not have principal judgements, since there is no *single* judgement for a term that subsumes all others *up to substitutions*. The literature has studied a weakened notion of principal judgement, subsuming all judgements up to substitution *and* up to another (very technical) operation called *expansion*, which – roughly – duplicates multi sets [22, 50, 43].

What we are going to do next, intuitively, is following the other natural route when principal judgements do not exist: we study a notion of principal *set* of special judgements for a term t, called *dry* judgements, which are such that every ordinary judgement for t can be seen as a dry judgement up to substitution. In fact, we only study this property for normal forms, and we also relate the derivations producing those judgements. We need some definitions.

Supports and Substitutions. The *support* of a type derivation $\Phi \rhd \Gamma \vdash t:\mathsf{T}$ is the set $\mathtt{TyVars}(\Phi) := \{\mathsf{X} \mid \mathsf{X} \text{ occurs in } \Phi\}$ of type variables appearing in Φ, and the *final support* is the set $\mathtt{TyVarsF}(\Phi) := \{\mathsf{X} \mid \mathsf{X} \text{ occurs in } \Gamma \text{ or } \mathsf{T}\}$ of type variables appearing in the last judgement of Φ. We write $\Phi \# \Psi$ as a shortcut for $\mathtt{TyVars}(\Phi) \cap \mathtt{TyVars}(\Psi) = \emptyset$ and given $\{\Phi_i\}_{i\in I}$ we write $\#_{i\in I}\,\Phi_i$ when $\mathtt{TyVars}(\Phi_h) \cap \mathtt{TyVars}(\Phi_k) = \emptyset$ for any two distinct $h, k \in I$.

A type substitution σ is a function from type variables to linear types that is the identity but a for finite number of type variables. It is extended to act on types, multi types, type contexts, and derivations as expected.

Dry Judgements. Dry judgements for normal forms are derived using the rules in Fig. 2. There are three key points. Firstly, only normal forms are typable. Secondly, neutral terms are always typed with a type variable (which is minimal) and when they are applied (in rule $@^*$) their type is *enlarged* on-the-fly via a type substitution $\{\mathsf{X}\leftarrow(\mathsf{M} \multimap \mathsf{Y})\}$ depending on the type of the argument. Thirdly, the system uses many type variables, and for the rules with more than one premise (*i.e.* $@^*$ and many^*) it requires them to have disjoint supports. This is where having countably many type variables plays a role, as having only a finite number would not allow one to prove the *subsumption up to substitutions* property of dry derivations, given by Thm. 19.2 below.

Dry derivations can be seen as standard derivations, as the second point of the next lemma states. It is obtained using the straightforward fact that standard derivations are stable by type substitutions. Note the use of skeleton equivalence.

▶ **Lemma 16.** *Let t be a term and f be a normal form.*

1. Substitutivity for standard*: if $\Phi \rhd \Gamma \vdash t:\mathsf{L}$ then for any linear type L' there exists $\Phi_{\{\mathsf{X}\leftarrow\mathsf{L}'\}} \rhd \Gamma\{\mathsf{X}\leftarrow\mathsf{L}'\} \vdash t:\mathsf{L}\{\mathsf{X}\leftarrow\mathsf{L}'\}$ such that $\Phi \sim \Phi_{\{\mathsf{X}\leftarrow\mathsf{L}'\}}$.*
2. Dry derivations are standard*: if $\Phi \rhd \Gamma \vdash^{\mathsf{d}} f:\mathsf{L}$ then there exists $\Psi \rhd \Gamma \vdash f:\mathsf{L}$ such that $\Phi \sim \Psi$.*

Proof. The first point is a straightforward induction on Φ. The second point is by induction on Φ. The only rule of the dry system that is not a rule of the standard system is $@^*$:

$$\frac{\Phi_n \rhd \Gamma_n \vdash^{\mathsf{d}} n : \mathsf{X} \quad \Phi_{f'} \rhd \Gamma_{f'} \vdash^{\mathsf{d}} f' : \mathsf{M} \quad \mathsf{Y} \text{ fresh}, \Phi \# \Psi}{\Gamma_n\{\mathsf{X} \leftarrow (\mathsf{M} \multimap \mathsf{Y})\} \uplus \Gamma_{f'} \vdash^{\mathsf{d}} nf' : \mathsf{Y}} @^*$$

with $f = nf'$, $\Gamma = \Gamma_n\{\mathsf{X} \leftarrow (\mathsf{M} \multimap \mathsf{Y})\} \uplus \Gamma_{f'}$, $\mathsf{T} = \mathsf{Y}$. By *i.h.*, there exist $\Psi_n \rhd \Gamma_n \vdash n : \mathsf{X}$ and $\Psi_{f'} \rhd \Gamma_{f'} \vdash f' : \mathsf{M}$. By Point 1, there exists a derivation: $\Psi_n\{\mathsf{X} \leftarrow (\mathsf{M} \multimap \mathsf{Y})\} \rhd \Gamma_n\{\mathsf{X} \leftarrow (\mathsf{M} \multimap \mathsf{Y})\} \vdash n : \mathsf{M} \multimap \mathsf{Y}$ Then we build Ψ as follows:

$$\frac{\Psi_n\{\mathsf{X} \leftarrow (\mathsf{M} \multimap \mathsf{Y})\} \rhd \Gamma_n\{\mathsf{X} \leftarrow (\mathsf{M} \multimap \mathsf{Y})\} \vdash n : \mathsf{M} \multimap \mathsf{Y} \quad \Psi_{f'} \rhd \Gamma_{f'} \vdash f' : \mathsf{M}}{\Gamma_n\{\mathsf{X} \leftarrow (\mathsf{M} \multimap \mathsf{Y})\} \uplus \Gamma_{f'} \vdash nf' : \mathsf{Y}} @$$

Skeleton equivalence of Φ and Ψ follows immediately from the skeleton equivalences of the *i.h.* plus the one of Point 1. ◀

Next, we prove that types in dry judgements are always minimal and – crucially – capture the size of the derivation itself. This is obtained via a strong property for type variables in dry judgements, reminiscent of similar properties in multiplicative linear logic, and enforced by the requirements about disjoint supports in the derivation rules.

▶ **Proposition 17.** *Let f be a normal form.*

1. Dry derivations and variable types occurrences*: if $\Phi \rhd \Gamma \vdash^{\mathsf{d}} f : \mathsf{T}$ then X has exactly two occurrences in (Γ, T) for every $\mathsf{X} \in \mathtt{TyVarsF}(\Phi)$.*
2. Dry derivations are minimal*: if $\Phi \rhd \Gamma \vdash^{\mathsf{d}} f : \mathsf{L}$ then $|\Phi|_{\mathtt{in}} = |(\hat{\Gamma}, \mathsf{L})|$.*

Proof.

1. By induction on Φ, looking at its last rule. For ax^*, the statement evidently holds, and for λ^* and rule many^* it follows from the *i.h.*, since these rules preserve and do not introduce occurrences of type variable. If the last rule of Φ is $@^*$, then Φ has shape:

$$\Phi = \frac{\Phi_1 \rhd \Delta \vdash^{\mathsf{d}} n : \mathsf{X} \quad \Phi_2 \rhd \Sigma \vdash^{\mathsf{d}} f' : \mathsf{M} \quad \mathsf{Y} \text{ fresh}, \Phi \# \Psi}{\Delta\{\mathsf{X} \leftarrow (\mathsf{M} \multimap \mathsf{Y})\} \uplus \Sigma \vdash^{\mathsf{d}} nf' : \mathsf{Y}} @^*$$

   with $f = nf'$, $\mathsf{T} = \mathsf{Y}$, and $\Gamma = \Delta\{\mathsf{X} \leftarrow (\mathsf{M} \multimap \mathsf{Y})\} \uplus \Sigma$. By *i.h.*, X occurs exactly once in Δ, thus Y occurs exactly twice in (Γ, T). Note that $\mathtt{TyVarsF}(\Phi) = (\mathtt{TyVarsF}(\Phi_1) \setminus \{\mathsf{X}\}) \cup \mathtt{TyVarsF}(\Phi_2) \cup \{\mathsf{Y}\}$. By *i.h.*, each type variable in $\mathtt{TyVarsF}(\Phi_1) \setminus \{\mathsf{X}\}$ (resp. $\mathtt{TyVarsF}(\Phi_2)$) occurs exactly twice in Δ (resp. (Σ, M)). Moreover, by hypothesis $\mathtt{TyVars}(\Phi_1) \cap \mathtt{TyVars}(\Phi_2) = \emptyset$, so each such type variable occurs exactly twice in (Γ, T).
2. By induction on Φ, looking at its last rule. Cases:
   - *Rule* ax^*: the statement holds because $|\Phi|_{\mathtt{in}} = 0 = |([\mathsf{X}], \mathsf{X})|$.
   - *Rule* many^*: it follows from the *i.h.*
   - *Rule* λ^*: it follows from the *i.h.* because rule λ^* add 1 to the size of the derivation, but the the size of the judgement also grows of 1, because of the introduced arrow.
   - *Rule* $@^*$: then Φ has the following shape:

$$\frac{\Phi_1 \rhd \Delta \vdash^{\mathsf{d}} n : \mathsf{X} \quad \Phi_2 \rhd \Sigma \vdash^{\mathsf{d}} f : \mathsf{M} \quad \mathsf{Y} \text{ fresh}, \Phi \# \Psi}{\Delta\{\mathsf{X} \leftarrow (\mathsf{M} \multimap \mathsf{Y})\} \uplus \Sigma \vdash^{\mathsf{d}} nf : \mathsf{Y}} @^*$$

     with $\Gamma = \Delta\{\mathsf{X} \leftarrow (\mathsf{M} \multimap \mathsf{Y})\} \uplus \Sigma$. By Point 1, X occurs exactly once in Δ. Then we have:

$$\begin{array}{rclll} |\Phi|_{\mathtt{in}} & = & |\Phi_1|_{\mathtt{in}} + |\Phi_2|_{\mathtt{in}} + 1 & =_{i.h.} & |\Delta| + |\Sigma| + |\mathsf{M}| + 1 \\ & = & |\Delta| + |\Sigma| + |\mathsf{M} \multimap \mathsf{Y}| & =_{P.1} & |\Delta\{\mathsf{X} \leftarrow (\mathsf{M} \multimap \mathsf{Y})\}| + |\Sigma| \\ & = & |\Delta\{\mathsf{X} \leftarrow (\mathsf{M} \multimap \mathsf{Y})\}| + |\Sigma| + |\mathsf{Y}| & & \end{array}$$

◀

Dry Representation. We now prove the key property of the analysis, which also justifies seeing dry derivations as defining a set of principal judgements. The idea is that every (standard) derivation Φ admits a dry derivation Ψ that is skeleton equivalent to Φ – which by the skeletal invariants above entails that they have the same size – and such that there is a substitution turning Ψ into Φ. We need an auxiliary lemma that shall also be useful in the next section, about renamings of dry derivations.

▶ **Lemma 18** (Dry derivations are stable by renaming). *Let $\Phi \rhd \Gamma \vdash^{\mathsf{d}} f : \mathsf{T}$ be a dry type derivation $\mathtt{TyVars}(\Phi) = \{\mathsf{X}_1, \dots, \mathsf{X}_n\}$ and $\mathsf{Y}_1, \dots, \mathsf{Y}_n$ be distinct type variables Φ. Then the derivation $\Phi_{\{\mathsf{X}_1,\dots,\mathsf{X}_n \leftarrow \mathsf{Y}_1,\dots,\mathsf{Y}_n\}}$ obtained by simultaneously replacing X_i with Y_i in Φ for $i \in \{1, \dots, n\}$ is a dry type derivation such that $\Phi \sim \Phi_{\{\mathsf{X}_1,\dots,\mathsf{X}_n \leftarrow \mathsf{Y}_1,\dots,\mathsf{Y}_n\}}$.*

Proof. Straightforward induction on Φ. ◀

▶ **Theorem 19.** *Let f be a normal form and $\Phi \rhd \Gamma \vdash f : \mathsf{T}$ be a type derivation.*

1. Dry representation*: there exists a dry derivation $\Psi \rhd \Delta \vdash^{\mathsf{d}} f : \mathsf{T}'$ such that $\Phi \sim \Psi$;*
2. Type substitution*: there exists a type substitution σ such that $\Delta\sigma = \Gamma$ and $\mathsf{T}'\sigma = \mathsf{T}$.*

Proof. By induction on Φ. Cases of the last rule:

- *Rule* ax. Then Φ is a ax rule of conclusion $x : [\mathsf{L}] \vdash x : \mathsf{L}$. The dry representation Ψ of Φ is a ax rule of conclusion $x : [\mathsf{X}] \vdash^{\mathsf{d}} x : \mathsf{X}$. The type substitution of the statement is $\{\mathsf{X} \leftarrow \mathsf{L}\}$.
- *Rule* λ: it follows by the *i.h.*
- *Rule* many: it follows by the *i.h.* Note that, by stability of dry derivations under renaming (Lemma 18), we can assume that all the derivations Ψ_i given by the *i.h.* are on disjoint supports, so that the constraint $\#_{i \in I}\, \Psi_i$ for rule many^* is satisfied.
- *Rule* @: then $f = nf'$ and Φ has the following shape:

$$\frac{\Phi_n \rhd \Gamma_n \vdash n : \mathsf{M} \multimap \mathsf{L} \quad \Phi_{f'} \rhd \Gamma_{f'} \vdash f' : \mathsf{M}}{\Gamma_n \uplus \Gamma_{f'} \vdash nf' : \mathsf{L}}\,@$$

with $\Gamma = \Gamma_n \uplus \Gamma_{f'}$ and $\mathsf{T} = \mathsf{L}$. About the dry representation, by *i.h.* there are dry derivations $\Psi_n \rhd \Delta_n \vdash^{\mathsf{d}} n : \mathsf{X}$ and $\Psi_{f'} \rhd \Delta_{f'} \vdash^{\mathsf{d}} f' : \mathsf{N}$ such that $\Phi_n \sim \Psi_n$ and $\Phi_{f'} \sim \Psi_{f'}$. By stability of dry derivations under renaming (Lemma 18), we can assume that $\Psi_n \# \Psi_n$. Then Ψ is obtained as follows:

$$\frac{\Psi_n \rhd \Delta_n \vdash^{\mathsf{d}} n : \mathsf{X} \quad \Psi_{f'} \rhd \Delta_{f'} \vdash^{\mathsf{d}} f' : \mathsf{N} \quad \mathsf{Y} \text{ fresh}, \Psi_n \# \Psi_n}{\Delta_n\{\mathsf{X} \leftarrow (\mathsf{N} \multimap \mathsf{Y})\} \uplus \Delta_{f'} \vdash^{\mathsf{d}} nf' : \mathsf{Y}}\,@^*$$

About the type substitution, by *i.h.*, there are substitutions σ_n and $\sigma_{f'}$ such that $\Delta_n\sigma_n = \Gamma_n$ and $\mathsf{X}\sigma_n = \mathsf{M} \multimap \mathsf{L}$, and $\Delta_{f'}\sigma_{f'} = \Gamma_{f'}$ and $\mathsf{N}\sigma_{f'} = \mathsf{M}$. We can assume that $\mathsf{dom}(\sigma_n) = \mathsf{dom}(\Delta_n)$ and $\mathsf{dom}(\sigma_{f'}) = \mathsf{dom}(\Delta_{f'})$, and we know that $\mathsf{dom}(\sigma_n) \cap \mathsf{dom}(\sigma_{f'}) = \emptyset$. Define the substitution σ as σ_n on $\mathsf{dom}(\sigma_n) \setminus \{\mathsf{X}\}$, as $\sigma_{f'}$ on $\mathsf{dom}(\sigma_{f'})$, and as $\{\mathsf{Y} \leftarrow \mathsf{L}\}$ on Y. Note that $\sigma_n(\mathsf{X}) = \mathsf{M} \multimap \mathsf{L}$ and let σ'_n be σ_n without $\{\mathsf{X} \leftarrow (\mathsf{M} \multimap \mathsf{L})\}$. Then:

$$\begin{array}{lcl} (\Delta_n\{\mathsf{X} \leftarrow (\mathsf{N} \multimap \mathsf{Y})\} \uplus \Delta_{f'})\sigma & = & \Delta_n\sigma'_n\{\mathsf{X} \leftarrow (\mathsf{N}\sigma_{f'} \multimap \mathsf{Y}\{\mathsf{Y} \leftarrow \mathsf{L}\})\} \uplus \Delta_{f'}\sigma_{f'} \\ & =_{i.h.} & \Delta_n\sigma'_n\{\mathsf{X} \leftarrow (\mathsf{M} \multimap \mathsf{L})\} \uplus \Gamma_{f'} \\ & = & \Delta_n\sigma_n \uplus \Gamma_{f'} \quad =_{i.h.} \quad \Gamma_n \uplus \Gamma_{f'} \quad = \quad \Gamma \end{array}$$

and $\mathsf{Y}\sigma = \mathsf{Y}\{\mathsf{Y} \leftarrow \mathsf{L}\} = \mathsf{L} = \mathsf{T}$. ◀

Removing Substitutions. In the previous theorem, the substitution part rests on the properties of dry derivations enabled by the countable number of type variables in the type system. We now show that a slightly weaker result is possible even with only one type variable and without dry derivations. The type substitution part shall not be recoverable, but the representation and the quantitative bounds are.

Let $\Gamma \vdash^1 t : \mathsf{L}$ denote a (standard) type derivation built using only 1-*types*, that is, types built using a single fixed type variable X.

▶ **Theorem 20** (Size representation). *Let f be a normal term and $\Phi \rhd \Gamma \vdash f : \mathsf{L}$ be a derivation. Then there exists a derivation $\Psi \rhd \Delta \vdash^1 f : \mathsf{L}'$ such that $\Psi \sim \Phi$ and $|\Psi|_{\mathtt{in}} = |\hat{\Delta}| + |\mathsf{L}'|$.*

The proof of the theorem in fact requires a stronger statement for the induction to go through, in particular having a separate point about neutral terms, for which a stronger property holds. See the technical report [1].

7 Bounds From Composable Types

In this section, we finally study bounds from composable types for leftmost evaluation and normal forms, which is the technical and neglected part of de Carvalho's work. To ease the study, we restrict to closed terms, so that type contexts disappear – de Carvalho does the same. There are however no issues in dealing with open terms.

Composable Types. De Carvalho's idea is that, given two normal forms t and u, one can extract bounds for tu by looking only at the types of t and u – that is, at $[\![t]\!]$ and $[\![u]\!]$ – because a derivation for tu is just the application of a derivation for t and one for u. We need to give a formal status to composable types, and we also need a notion of types that compose up to a type substitution.

▶ **Definition 21** (Composable pairs). *Let t and u be closed terms.*

- *A* (shrinking) composable pair *(of types) for t and u is a pair $p = (\mathsf{L}, \mathsf{M})$ such that $\mathsf{L} = \mathsf{M} \multimap \mathsf{L}' \in [\![t]\!]$, $\mathsf{M} \in [\![u]\!]$, and L' is right. The set of composable pairs of t and u is noted* $\mathtt{ShComPairs}(t, u)$.
- *A* (shrinking) composable pair up to substitution *for t and u is a pair $p = (\mathsf{L}, \mathsf{M})$ such that there exists a type substitution σ such that $(\mathsf{L}\sigma, \mathsf{M}\sigma) \in \mathtt{ShComPairs}(t, u)$. The set of composable pairs up to substitution of t and u is noted as* $\mathtt{ShComPairsSub}(t, u)$.

Note that if $(\mathsf{M} \multimap \mathsf{L}', \mathsf{M}) \in \mathtt{ShComPairs}(t, u)$ only L' is required to be a right shrinking type (as it is the only type in the judgement for tu after composition), while $\mathsf{M} \multimap \mathsf{L}'$ might very well not be a right shrinking type (if M is not left). The constraint that L' is right in the definition of $\mathtt{ShComPairs}(t, u)$ ensures the following property.

▶ **Lemma 22** (Normalization and composable types). *Let t and u be closed terms. Then tu $\to_{\mathtt{lo}}$-normalizes if and only if $\mathtt{ShComPairs}(t, u) \neq \emptyset$.*

Proof. If tu normalizes then by shrinking completeness (Thm. 8) there exists a shrinking derivation $\Phi \vdash tu : \mathsf{L}$ with L right. The last rule of Φ is @ and its premises give a composable pair $(\mathsf{M} \multimap \mathsf{L}, \mathsf{M})$ for t and u, for some M. Therefore, $\mathtt{ShComPairs}(t, u) \supset \{(\mathsf{M} \multimap \mathsf{L}, \mathsf{M})\} \neq \emptyset$.

Vice versa, if $\mathtt{ShComPairs}(t, u) \neq \emptyset$ then every composable pair induces a shrinking derivation for tu, by connecting the two derivations producing the composable pair via rule @. Thus, tu is typable. By shrinking correctness (Thm. 8), tu is $\to_{\mathtt{lo}}$-normalizing. ◀

Normal Forms and Composable Types. To warm up, we first show how to bound the size of normal forms from composable pairs.

▶ **Theorem 23** (Normal form bounds from composable types). *Let t and u be closed terms such that there is a normalizing evaluation $d : tu \to_{\mathtt{lo}}^{*} f$.*

1. Lax bounds*: $|f|_{\mathtt{in}} \leq |\mathsf{L}|$ for every composable pair $(\mathsf{M} \multimap \mathsf{L}, \mathsf{M}) \in \mathtt{ShComPairs}(t, u)$.*
2. Exact bounds from special pairs*: moreover, there exists a composable pair $(\mathsf{M} \multimap \mathsf{L}, \mathsf{M}) \in \mathtt{ShComPairs}(t, u)$ such that $|f|_{\mathtt{in}} = |\mathsf{L}|$.*

Proof.

1. Let $(\mathsf{M} \multimap \mathsf{L}, \mathsf{M}) \in \mathtt{ShComPairs}(t, u)$ be a composable pair. Then, there are derivations $\Phi_t \rhd\vdash t : \mathsf{M} \multimap \mathsf{L}$ and $\Phi_u \rhd\vdash u : \mathsf{M}$. We compose them as a derivation Φ for tu via rule @:

$$\Phi := \dfrac{\Phi_t \rhd\vdash t : \mathsf{M} \multimap \mathsf{L} \qquad \Phi_u \rhd\vdash u : \mathsf{M}}{\vdash tu : \mathsf{L}} @$$

   Note that the definition of composable pair guarantees that L is right (shrinking), so that Φ is shrinking. By shrinking correctness (Thm. 8), there is a derivation $\vdash f : \mathsf{L}$. Since shrinking types bound the size of normal forms (Prop. 12), $|f|_{\mathtt{in}} \leq |\mathsf{L}|$.
2. By Prop. 12.2, there exists a unitary shrinking derivation $\Psi \rhd\vdash f : \mathsf{L}$ for f such that $|\mathsf{L}| = |f|_{\mathtt{in}}$. Pulling back the final judgement of Ψ using subject expansion (Prop. 3.2), we obtain a derivation $\Theta \vdash tu : \mathsf{L}$. The last rule of Θ is @ and its premises give a composable pair $(\mathsf{M} \multimap \mathsf{L}, \mathsf{M})$ for t and u, for some M. ◀

Lax Evaluation Bounds from Composable Types. Now, we study how to additionally extract (bounds on) the number of leftmost step from a composable pair. Obtaining lax bounds is easy. The idea is that the composed type bounds the size of a derivation for tu, which in turns provides bounds about tu, as shown in Sect. 4. We also show that even composable pairs up to substitution yield bounds.

▶ **Theorem 24** (Lax bounds from composable types). *Let f and f' be closed normal terms such that $d : ff' \to_{\mathtt{lo}}^{*} f''$ with f'' normal. Then:*

1. Lax bounds and types*: $2|d| + |f''|_{\mathtt{in}} \leq |\mathsf{L}| + |\mathsf{M}| + 1$ for every composable pair $(\mathsf{L}, \mathsf{M}) \in \mathtt{ShComPairs}(f, f')$.*
2. Lax bounds and types, up to substitutions*: $2|d| + |f''|_{\mathtt{in}} \leq |\mathsf{L}| + |\mathsf{M}| + 1$ for every composable pair up to substitution $(\mathsf{L}, \mathsf{M}) \in \mathtt{ShComPairsSub}(f, f')$.*

Proof.

1. Let $(\mathsf{L}, \mathsf{M}) \in \mathtt{ShComPairs}(f, f')$ be a composable pair, which implies $\mathsf{L} = \mathsf{M} \multimap \mathsf{L}'$ for some right L'. Then, there are two derivations $\Phi_f \rhd\vdash f : \mathsf{M} \multimap \mathsf{L}'$ and $\Phi_{f'} \rhd\vdash f' : \mathsf{M}$. We compose them via a @ rule into a derivation Φ for ff':

$$\Phi := \dfrac{\Phi_f \rhd\vdash f : \mathsf{M} \multimap \mathsf{L}' \qquad \Phi_{f'} \rhd\vdash f' : \mathsf{M}}{\vdash ff' : \mathsf{L}'} @$$

   By definition of composable pair, L' is right shrinking, so that Φ is shrinking. By shrinking correctness (Thm. 8), $2|d| + |f''|_{\mathtt{in}} \leq |\Phi|_{\mathtt{in}} = |\Phi_f|_{\mathtt{in}} + |\Phi_{f'}|_{\mathtt{in}} + 1$. Now, since types bound the size of the derivation for normal terms (Prop. 13), we obtain $|\Phi_f|_{\mathtt{in}} \leq |\mathsf{M} \multimap \mathsf{L}'|$ and $|\Phi_{f'}|_{\mathtt{in}} \leq |\mathsf{M}|$. Therefore, $2|d| + |f''|_{\mathtt{in}} \leq |\Phi|_{\mathtt{in}} \leq |\mathsf{M} \multimap \mathsf{L}'| + |\mathsf{M}| + 1$, as required.
2. Let $(\mathsf{L}, \mathsf{M}) \in \mathtt{ShComPairsSub}(f, f')$ be a composable pair up to substitution, which implies that there exist a type substitution σ such that $\mathsf{L}\sigma = \mathsf{N} \multimap \mathsf{L}'$, $\mathsf{M}\sigma = \mathsf{N}$ for some right L' and some N. Then, there are two derivations $\Phi_f \rhd\vdash f : \mathsf{L}$ and $\Phi_{f'} \rhd\vdash f' : \mathsf{M}$. By

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x:[[\mathsf{W}^2]\multimap\mathsf{W}^2]\vdash x:[\mathsf{W}^2]\multimap\mathsf{W}^2}\,\mathsf{ax}\quad\dfrac{\dfrac{}{x:[\mathsf{W}^2]\vdash x:\mathsf{W}^2}\,\mathsf{ax}}{x:[\mathsf{W}^2]\vdash x:[\mathsf{W}^2]}\,\mathsf{many}}{x:[[\mathsf{W}^2]\multimap\mathsf{W}^2,\mathsf{W}^2]\vdash xx:\mathsf{W}^2}\,@}{\Psi_\delta\rhd\vdash\lambda x.xx:[[\mathsf{W}^2]\multimap\mathsf{W}^2,\mathsf{W}^2]\multimap\mathsf{W}^2}\,\lambda\qquad\dfrac{\dfrac{\dfrac{}{y:[\mathsf{W}^2]\vdash y:\mathsf{W}^2}\,\mathsf{ax}}{\vdash\lambda y.y:[\mathsf{W}^2]\multimap\mathsf{W}^2}\,\lambda\quad\dfrac{\dfrac{}{y:[\mathsf{W}]\vdash y:\mathsf{W}}\,\mathsf{ax}}{\vdash\lambda y.y:[\mathsf{W}]\multimap\mathsf{W}}\,\lambda}{\Psi_{\mathsf{I}}\rhd\vdash\lambda y.y:[[\mathsf{W}^2]\multimap\mathsf{W}^2,\mathsf{W}^2]}\,\mathsf{many}}{\vdash\delta\mathsf{I}:\mathsf{W}^2}\,@$$

Figure 3 Unitary shrinking derivation $\Psi_{\delta\mathsf{I}}$ of minimal type for $\delta\mathsf{I}$, where $\mathsf{W}^2 \coloneqq [\mathsf{W}]\multimap\mathsf{W}$.

Lemma 16.1, applying σ to Φ_f and $\Phi_{f'}$ we obtain two derivations $\Psi_f \rhd\vdash f:\mathsf{N}\multimap\mathsf{L}'$ and $\Psi_{f'}\rhd\vdash f':\mathsf{N}$ such that $\Phi_f\sim\Psi_f$ and $\Phi_{f'}\sim\Psi_{f'}$. We compose them via a @ rule into a derivation Ψ for ff':

$$\Psi \coloneqq \dfrac{\Psi_f\rhd\vdash f:\mathsf{N}\multimap\mathsf{L}'\qquad\Psi_{f'}\rhd\vdash f':\mathsf{N}}{\vdash ff':\mathsf{L}'}\,@$$

Since L' is right, Ψ is shrinking. By shrinking correctness (Thm. 8), $2|d|+|f''|_{\mathtt{in}}\le|\Psi|_{\mathtt{in}}=|\Psi_f|_{\mathtt{in}}+|\Psi_{f'}|_{\mathtt{in}}+1=|\Phi_f|_{\mathtt{in}}+|\Phi_{f'}|_{\mathtt{in}}+1$. By the skeletal invariants (Lemma 15), we obtain $|\Psi_f|_{\mathtt{in}}+|\Psi_{f'}|_{\mathtt{in}}+1=|\Phi_f|_{\mathtt{in}}+|\Phi_{f'}|_{\mathtt{in}}+1$. Since types bound the size of the derivation for normal terms (Prop. 13), $|\Phi_f|_{\mathtt{in}}\le|\mathsf{L}|$ and $|\Phi_{f'}|_{\mathtt{in}}\le|\mathsf{M}|$. Therefore, $2|d|+|f''|_{\mathtt{in}}\le|\mathsf{L}|+|\mathsf{M}|+1$, as required. ◀

Note that the hypotheses for bounding evaluation lengths are stronger than for bounding normal forms (Thm. 23), as the two applied terms f and f' are required to be normal. If the two composed terms are not normal, then there is no way to measure the extra steps to their normal forms using their types, because types are invariant by reduction. The stronger hypotheses limit the scope of the result: if $t\coloneqq\lambda x.x\Omega f$ with f normal and $u\coloneqq\lambda y.\lambda z.z$ then $tu\to^3_{\mathtt{lo}} f$ and Thm. 23 can be applied, while Thm. 24 cannot, because of the diverging Ω sub-term in t.

Exact Bounds from Composable Types. Now, the tricky point is how to obtain *exact* bounds. The problem is that for the application of two normal terms f and f', the minimal composable pair in general does *not* provide exact bounds, and – dually – minimal types for f and f' do *not* compose. The following example pinpoints the subtleties.

Key Example. Consider the unitary shrinking derivations Φ_δ and Φ_{I} of minimal types for $\delta\coloneqq\lambda x.xx$ and for $\mathsf{I}\coloneqq\lambda y.y$:

$$\Phi_\delta = \dfrac{\dfrac{\dfrac{}{x:[[\mathsf{Z}]\multimap\mathsf{Z}]\vdash x:[\mathsf{Z}]\multimap\mathsf{Z}}\,\mathsf{ax}\quad\dfrac{}{x:[\mathsf{Z}]\vdash x:\mathsf{Z}}\,\mathsf{ax}}{x:[[\mathsf{Z}]\multimap\mathsf{Z},\mathsf{Z}]\vdash xx:\mathsf{Z}}\,@}{\vdash\lambda x.xx:[[\mathsf{Z}]\multimap\mathsf{Z},\mathsf{Z}]\multimap\mathsf{Z}}\,\lambda \qquad \Phi_{\mathsf{I}} = \dfrac{\dfrac{}{y:[\mathsf{W}]\vdash y:\mathsf{W}}\,\mathsf{ax}}{\vdash\lambda y.y:[\mathsf{W}]\multimap\mathsf{W}}\,\lambda$$

Note that, pleasantly, $|\Phi_\delta|_{\mathtt{in}}=2=|[[\mathsf{Z}]\multimap\mathsf{Z},\mathsf{Z}]\multimap\mathsf{Z}|=|\delta|_{\mathtt{in}}$, and $|\Phi_{\mathsf{I}}|_{\mathtt{in}}=1=|[\mathsf{W}]\multimap\mathsf{W}|=|\mathsf{I}|_{\mathtt{in}}$. Now, consider the application $\delta\mathsf{I}$. Note that, unfortunately, the two obtained minimal types do *not* compose, and not just because they use different type variables: identifying Z and W would not be enough, one actually needs to identify Z with $\mathsf{W}^2\coloneqq[\mathsf{W}]\multimap\mathsf{W}$. The unitary shrinking derivation $\Psi_{\delta\mathsf{I}}$ for $\delta\mathsf{I}$ with minimal types (which provides exact information for $\delta\mathsf{I}$) obtained in this way is in Fig. 3. Note that its sub-derivations Ψ_δ and Ψ_{I} for δ and I do *not* derive minimal types. The derivation $\Psi_{\delta\mathsf{I}}$ indeed is obtained by composing:

1. The variant Ψ_δ of Φ_δ which has the same exact structure of Φ_δ and where every occurrence of Z has been replaced by W^2, obtaining the type $[[[\mathsf{W}^2]\multimap\mathsf{W}^2,\mathsf{W}^2]]\multimap\mathsf{W}^2$,

2. With Ψ_{I}, which puts together two derivations for I, one being Φ_{I} (of type W^2), and one being the variant Φ'_{I} of Φ_{I} (of type $[\mathsf{W}^2] \multimap \mathsf{W}^2$) where W has been replaced by W^2.

Note that there is a *gap* between:

- The length of the evaluation $d : \delta\mathsf{I} \to_{\mathtt{lo}} \mathsf{II} \to_{\mathtt{lo}} \mathsf{I}$, that takes 2 steps, plus the size of the normal form $|\mathsf{I}|_{\mathtt{in}} = 1$, so that $2|d| + |\mathsf{I}|_{\mathtt{in}} = 5$, and
- The size of the composable pair $p = ([[\mathsf{Z}^2] \multimap \mathsf{Z}^2, \mathsf{Z}^2] \multimap \mathsf{Z}^2, [[\mathsf{Z}^2] \multimap \mathsf{Z}^2, \mathsf{Z}^2])$, which is 10.

The point is that the types derived by Ψ_δ and Ψ_{I} are *not* minimal, so their sizes are bigger than $|\Psi_\delta|_{\mathtt{in}}$ and $|\Psi_{\mathsf{I}}|_{\mathtt{in}}$, and do not provide exact bounds for $\delta\mathsf{I}$. For instance, the size of $[[\mathsf{W}^2] \multimap \mathsf{W}^2, \mathsf{W}^2] \multimap \mathsf{W}^2$, which is the type of Ψ_δ, is 6, while $|\Psi_\delta|_{\mathtt{in}} = 2$ – this is an instance of the mentioned gap. Summing up, *minimal types do not compose*, and *composable types do not give exact bounds*.

Out of the Impasse. De Carvalho solves this *cul-de-sac* using composable pairs *up to substitution*. With respect to our example, he considers the composable pair p given by Ψ, but computes the bound using the pair $p' = ([[\mathsf{Z}] \multimap \mathsf{Y}, \mathsf{Z}] \multimap \mathsf{Y}, [[\mathsf{X}] \multimap \mathsf{X}, [\mathsf{X}'] \multimap \mathsf{X}'])$, which is minimal and non-composable. It is obtained by collecting the types of the dry version of Ψ_δ (of type $[[\mathsf{Z}] \multimap \mathsf{Y}, \mathsf{Z}] \multimap \mathsf{Y}$) and the dry version of Ψ_{I} (typing I twice thus having type $[[\mathsf{X}] \multimap \mathsf{X}, [\mathsf{X}'] \multimap \mathsf{X}']$). The last bit is noting that p' is composable *up to the substitution* $\sigma := \{\mathsf{Z} \leftarrow \mathsf{W}^2\}\{\mathsf{Y} \leftarrow \mathsf{W}^2\}\{\mathsf{X} \leftarrow \mathsf{W}^2\}\{\mathsf{X}' \leftarrow \mathsf{W}\}$, since $p'\sigma = p$.

Roughly, for minimal types to compose, they usually have to be expanded, as we have done in the example when substituting Z with W^2. Such an expansion introduces some noise in the measures, so that even minimal composable pairs might not provide exact bounds. De Carvalho's trick is to reverse the expansion, considering composable pairs up to substitution. Our notion of dry derivation makes the expansion reversal technically clean.

Main Result. We can now prove the main result of the paper, namely de Carvalho's exact bounds from composable types.

▶ **Theorem 25** (Exact bounds from composable types). *Let f and f' be normal. If $d : ff' \to^*_{\mathtt{lo}} f''$ and f'' is normal. Then:*

1. Exact bounds*: there exist $\mathsf{L} \in [\![f]\!]$ and $\mathsf{M} \in [\![f']\!]$ such that $2|d| + |f''|_{\mathtt{in}} = |\mathsf{L}| + |\mathsf{M}| + 1$, and L and M are obtained by drying the composable pair induced by a unitary shrinking derivation for ff'.*
2. From types composable up to substitution*: moreover, (L, M) are composable up to substitution, that is, $(\mathsf{L}, \mathsf{M}) \in \mathtt{ShComPairsSub}(f, f')$.*

Proof.

1. By unitary shrinking completeness (Thm. 9), there is a unitary shrinking derivation $\Phi \rhd ff'$ which, by unitary shrinking correctness, satisfies $2|d| + |f''|_{\mathtt{in}} = |\Phi|_{\mathtt{in}}$. The last rule of Φ is an @ rule, that is, Φ has the following shape:

$$\frac{\Phi_f \rhd\vdash f : \mathsf{N} \multimap \mathsf{L}' \qquad \Phi_{f'} \rhd\vdash f' : \mathsf{N}}{\vdash ff' : \mathsf{L}'}\ @$$

With L' right linear type. Note that $|\Phi|_{\mathtt{in}} = |\Phi_f|_{\mathtt{in}} + |\Phi_{f'}|_{\mathtt{in}} + 1$. By dry representation (Thm. 19.1), there are dry derivations $\Psi_f \rhd\vdash f : \mathsf{L}$ and $\Psi_{f'} \rhd\vdash f' : \mathsf{M}$ such that $\Phi_f \sim \Psi_f$ and $\Phi_{f'} \sim \Psi_{f'}$. By the properties of skeletal invariants (Lemma 15), $|\Phi_f|_{\mathtt{in}} = |\Psi_f|_{\mathtt{in}}$ and $|\Phi_{f'}|_{\mathtt{in}} = |\Psi_{f'}|_{\mathtt{in}}$. By the fact that dry derivations have minimal types (Prop. 17), $|\Psi_f|_{\mathtt{in}} = |\mathsf{L}|$ and $|\Psi_{f'}|_{\mathtt{in}} = |\mathsf{M}|$. Putting it all together, we obtain:

$$\begin{array}{lll} 2|d| + |f''|_{\mathtt{in}} & =_{T.9} & |\Phi_f|_{\mathtt{in}} + |\Phi_{f'}|_{\mathtt{in}} + 1 \\ & =_{L.15} & |\Psi_f|_{\mathtt{in}} + |\Psi_{f'}|_{\mathtt{in}} + 1 \quad =_{Pr.\ 17} \quad |\mathsf{L}| + |\mathsf{M}| + 1 \end{array}$$

2. By the *type substitution* part of the dry representation theorem (Thm. 19.2), there exist substitutions σ_f and $\sigma_{f'}$ such that $\mathsf{L}\sigma_f = \mathsf{N} \multimap \mathsf{L}'$ and $\mathsf{M}\sigma_{f'} = \mathsf{N}$. By stability of dry derivations under renaming (Lemma 18), we can assume that the supports of Ψ_f and $\Psi_{f'}$ are disjoint, so that the domains of σ_f and $\sigma_{f'}$ are disjoint, allowing us to define σ as $\sigma_f \cup \sigma_{f'}$, for which $\mathsf{L}\sigma = \mathsf{L}\sigma_f = \mathsf{N} \multimap \mathsf{L}'$ and $\mathsf{M}\sigma = \mathsf{M}\sigma_{f'} = \mathsf{N}$. Finally, note that L' is right by hypothesis (because Φ is shrinking), so that $(\mathsf{L}, \mathsf{M}) \in \mathtt{ShComPairsSub}(f, f')$. ◀

8 The Less Satisfying Head Case

We conclude our study by adapting the bounds from composable types to the case of head reduction. The study is slightly different in that we omit the study of type substitutions and dry derivations. We do so to show that one can obtain the first part of the de Carvalho's result – which in our opinion is the important one – by resting only on the simpler *size representation theorem* of Sect. 6, with no need of bothering about countably many type variables and dry derivations.

We give only the proof of the main theorem, as the other ones are variants of those in the previous section. They can be found in the technical report [1].

The Head Case. Let $\mathtt{ComPairs}(t, u)$ and $\mathtt{ComPairsSub}(t, u)$ be the analogous sets of $\mathtt{ShComPairs}(t, u)$ and $\mathtt{ShComPairsSub}(t, u)$ but without asking that the composed type is right shrinking, which is not needed for characterizing head termination.

▶ **Lemma 26** (Head normalization and composable types). *Let t and u be closed terms. Then tu $\to_{\mathtt{h}}$-normalizes if and only if $\mathtt{ComPairs}(t, u) \neq \emptyset$.*

The next theorem adapts lax bounds.

▶ **Theorem 27** (Lax bounds for head reduction from composable types). *Let f and f' be closed normal terms such that $d : ff' \to^*_{\mathtt{lo}} h$ with h head normal. Then:*

1. Lax bounds and types*:* $2|d| + |h|_{\mathtt{h}} \leq |\mathsf{L}| + |\mathsf{M}| + 1$ *for every composable pair* $(\mathsf{L}, \mathsf{M}) \in \mathtt{ComPairs}(f, f')$.
2. Lax bounds and types, up to substitutions*:* $2|d| + |h|_{\mathtt{h}} \leq |\mathsf{L}| + |\mathsf{M}| + 1$ *for every composable pair up to substitution* $(\mathsf{L}, \mathsf{M}) \in \mathtt{ComPairsSub}(f, f')$.

▶ **Theorem 28** (Exact bounds for head reduction from composable types). *Let f and f' be normal and such that $d : ff' \to^*_{\mathtt{h}} h$ with h head normal. Then there exist $\mathsf{L} \in [\![f]\!]$ and $\mathsf{M} \in [\![f']\!]$ such that $2|d| + |h|_{\mathtt{h}} = |\mathsf{L}| + |\mathsf{M}| + 1$.*

Proof. By head completeness (Thm. 6), there is a derivation $\Phi \rhd ff'$ satisfying $2|d| + |h|_{\mathtt{h}} = |\Phi|_{\mathtt{in}}$. The last rule of Φ is an @ rule, that is, Φ has the following shape:

$$\dfrac{\Phi_f \rhd\vdash f : \mathsf{N} \multimap \mathsf{L}' \qquad \Phi_{f'} \rhd\vdash f' : \mathsf{N}}{\vdash ff' : \mathsf{L}'}\ @$$

Note that $|\Phi|_{\mathtt{in}} = |\Phi_f|_{\mathtt{in}} + |\Phi_{f'}|_{\mathtt{in}} + 1$. By size representation (Thm. 20), there are $\Psi_f \rhd\vdash f : \mathsf{L}$ and $\Psi_{f'} \rhd\vdash f' : \mathsf{M}$ such that $\Phi_f \sim \Psi_f$ and $|\Psi_f|_{\mathtt{in}} = |\mathsf{L}|$, and $\Phi_{f'} \sim \Psi_{f'}$ and $|\Psi_{f'}|_{\mathtt{in}} = |\mathsf{M}|$.

By the properties of skeletal invariants (Lemma 15), $|\Phi_f|_{\mathtt{in}} = |\Psi_f|_{\mathtt{in}}$ and $|\Phi_{f'}|_{\mathtt{in}} = |\Psi_{f'}|_{\mathtt{in}}$. Putting it all together, we obtain:

$$\begin{array}{lllll} 2|d| + |h|_{\mathtt{h}} & =_{T.6} & |\Phi_f|_{\mathtt{in}} + |\Phi_{f'}|_{\mathtt{in}} + 1 & & \\ & =_{L.15} & |\Psi_f|_{\mathtt{in}} + |\Psi_{f'}|_{\mathtt{in}} + 1 & =_{T.20} & |\mathsf{L}| + |\mathsf{M}| + 1. \end{array}$$ ◀

Note that the hypotheses *f and f' are normal* is not a typo, we do mean *normal*, not *head normal*. The stronger hypotheses are required because the evaluation d leading to h might involve arbitrary sub-terms of f and f', not just their heads. For instance if $f := \lambda x.xt$ and $f' := \mathsf{I}$ then $ff' \to_{\mathtt{h}} \mathsf{I}t \to_{\mathtt{h}} t$, and t is not a head sub-term of f.

This is where the results are less satisfying. Having to assume that f and f' are normal might be acceptable for leftmost reduction but it limits considerably the value of de Carvalho's analysis in the head case, which is the case naturally corresponding to relational semantics. Indeed, there are many head normalizing terms that have no normal form – the paradigmatic example begin given by *fix-point operators* – and about which Thm. 28 does not say anything.

9 Conclusions

In 2007, de Carvalho developed a sharp quantitative analysis of the λ-calculus using multi types and the relational model. His study has been influential, leading to a recent new wave of studies in the λ-calculus halfway between operational and denotational semantics. Only the first and simpler half of de Carvalho's results, however, has really permeated the community. The second more technical – and probably even more interesting – part, which lifts the quantitative analysis to the relational model, has instead been ignored by the recent literature. This paper dissects it and revisits it, pointing out the underlying subtleties and clarifying the concepts and tools for its proof.

A preliminary version of this work led to the adaption of de Carvalho's compositional bounds to call-by-value, which can be found in the technical report [11] by Accattoli et al. Hopefully, further adaptations to other λ-calculi will be developed.

About future work, a sharper study for the head case should be developed, as to avoid the normal form hypotheses. The main weakness of de Carvalho's results, indeed, is that they really work only for strong reduction, at present. As we hinted at in the introduction, probably a finer understanding of the external *vs* internal behaviour of terms is needed.

References

1. Beniamino Accattoli. Semantic bound and multi types, revisited, 2023. arXiv:2311.18233.
2. Beniamino Accattoli and Ugo Dal Lago. (Leftmost-outermost) Beta reduction is invariant, indeed. *Logical Methods in Computer Science*, 12(1), 2016. doi:10.2168/LMCS-12(1:4)2016.
3. Beniamino Accattoli, Ugo Dal Lago, and Gabriele Vanoni. The (in)efficiency of interaction. *Proc. ACM Program. Lang.*, 5(POPL):1–33, 2021. doi:10.1145/3434332.
4. Beniamino Accattoli, Ugo Dal Lago, and Gabriele Vanoni. The space of interaction. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 – July 2, 2021*, pages 1–13. IEEE, 2021. doi:10.1109/LICS52264.2021.9470726.
5. Beniamino Accattoli, Ugo Dal Lago, and Gabriele Vanoni. Multi types and reasonable space. *Proc. ACM Program. Lang.*, 6(ICFP):799–825, 2022. doi:10.1145/3547650.
6. Beniamino Accattoli, Claudia Faggian, and Giulio Guerrieri. Factorization and normalization, essentially. In Anthony Widjaja Lin, editor, *Programming Languages and Systems – 17th Asian Symposium, APLAS 2019, Nusa Dua, Bali, Indonesia, December 1-4, 2019, Proceedings*, volume 11893 of *Lecture Notes in Computer Science*, pages 159–180. Springer, 2019. doi:10.1007/978-3-030-34175-6_9.
7. Beniamino Accattoli, Stéphane Graham-Lengrand, and Delia Kesner. Tight typings and split bounds, fully developed. *J. Funct. Program.*, 30:e14, 2020. doi:10.1017/S095679682000012X.
8. Beniamino Accattoli and Giulio Guerrieri. Types of fireballs. In *Programming Languages and Systems – 16th Asian Symposium, APLAS 2018, Wellington, New Zealand, December 2-6, 2018, Proceedings*, volume 11275 of *Lecture Notes in Computer Science*, pages 45–66. Springer, 2018. doi:10.1007/978-3-030-02768-1_3.

9 Beniamino Accattoli and Giulio Guerrieri. The theory of call-by-value solvability. *Proc. ACM Program. Lang.*, 6(ICFP):855–885, 2022. doi:10.1145/3547652.

10 Beniamino Accattoli, Giulio Guerrieri, and Maico Leberle. Types by need. In *Programming Languages and Systems – 28th European Symposium on Programming, ESOP 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings*, volume 11423 of *Lecture Notes in Computer Science*, pages 410–439. Springer, 2019. doi:10.1007/978-3-030-17184-1_15.

11 Beniamino Accattoli, Giulio Guerrieri, and Maico Leberle. Semantic bounds and strong call-by-value normalization. *CoRR*, abs/2104.13979, 2021. URL: https://arxiv.org/abs/2104.13979.

12 Sandra Alves, Delia Kesner, and Miguel Ramos. Quantitative global memory. In Helle Hvid Hansen, Andre Scedrov, and Ruy J. G. B. de Queiroz, editors, *Logic, Language, Information, and Computation – 29th International Workshop, WoLLIC 2023, Halifax, NS, Canada, July 11-14, 2023, Proceedings*, volume 13923 of *Lecture Notes in Computer Science*, pages 53–68. Springer, 2023. doi:10.1007/978-3-031-39784-4_4.

13 Sandra Alves, Delia Kesner, and Daniel Ventura. A quantitative understanding of pattern matching. In *25th International Conference on Types for Proofs and Programs, TYPES 2019, June 11-14, 2019, Oslo, Norway*, volume 175 of *LIPIcs*, pages 3:1–3:36. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.TYPES.2019.3.

14 Alexis Bernadet and Stéphane Lengrand. Non-idempotent intersection types and strong normalisation. *Logical Methods in Computer Science*, 9(4), 2013. doi:10.2168/LMCS-9(4:3)2013.

15 Flavien Breuvart, Giulio Manzonetto, and Domenico Ruoppolo. Relational graph models at work. *Log. Methods Comput. Sci.*, 14(3), 2018. doi:10.23638/LMCS-14(3:2)2018.

16 Antonio Bucciarelli, Alberto Carraro, Thomas Ehrhard, and Giulio Manzonetto. Full abstraction for resource calculus with tests. In Marc Bezem, editor, *Computer Science Logic, 25th International Workshop / 20th Annual Conference of the EACSL, CSL 2011, September 12-15, 2011, Bergen, Norway, Proceedings*, volume 12 of *LIPIcs*, pages 97–111. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2011. doi:10.4230/LIPIcs.CSL.2011.97.

17 Antonio Bucciarelli and Thomas Ehrhard. On phase semantics and denotational semantics: the exponentials. *Ann. Pure Appl. Logic*, 109(3):205–241, 2001. doi:10.1016/S0168-0072(00)00056-7.

18 Antonio Bucciarelli, Thomas Ehrhard, and Giulio Manzonetto. A relational model of a parallel and non-deterministic lambda-calculus. In Sergei N. Artëmov and Anil Nerode, editors, *Logical Foundations of Computer Science, International Symposium, LFCS 2009, Deerfield Beach, FL, USA, January 3-6, 2009. Proceedings*, volume 5407 of *Lecture Notes in Computer Science*, pages 107–121. Springer, 2009. doi:10.1007/978-3-540-92687-0_8.

19 Antonio Bucciarelli, Delia Kesner, Alejandro Ríos, and Andrés Viso. The bang calculus revisited. In *Functional and Logic Programming – 15th International Symposium, FLOPS 2020, Akita, Japan, September 14-16, 2020, Proceedings*, pages 13–32. Springer, 2020. doi:10.1007/978-3-030-59025-3_2.

20 Antonio Bucciarelli, Delia Kesner, and Daniel Ventura. Non-idempotent intersection types for the lambda-calculus. *Log. J. IGPL*, 25(4):431–464, 2017. doi:10.1093/JIGPAL/JZX018.

21 Alberto Carraro, Thomas Ehrhard, and Antonino Salibra. Exponentials with infinite multiplicities. In Anuj Dawar and Helmut Veith, editors, *Computer Science Logic, 24th International Workshop, CSL 2010, 19th Annual Conference of the EACSL, Brno, Czech Republic, August 23-27, 2010. Proceedings*, volume 6247 of *Lecture Notes in Computer Science*, pages 170–184. Springer, 2010. doi:10.1007/978-3-642-15205-4_16.

22 Mario Coppo, Mariangiola Dezani-Ciancaglini, and Betti Venneri. Functional characters of solvable terms. *Math. Log. Q.*, 27(2-6):45–58, 1981. doi:10.1002/malq.19810270205.

23 Ugo Dal Lago, Claudia Faggian, and Simona Ronchi Della Rocca. Intersection types and (positive) almost-sure termination. *Proc. ACM Program. Lang.*, 5(POPL):1–32, 2021. doi:10.1145/3434313.

24 Vincent Danos and Thomas Ehrhard. Probabilistic coherence spaces as a model of higher-order probabilistic computation. *Inf. Comput.*, 209(6):966–991, 2011. doi:10.1016/j.ic.2011.02.001.

25 Daniel de Carvalho. *Sémantiques de la logique linéaire et temps de calcul*. Thèse de doctorat, Université Aix-Marseille II, 2007.

26 Daniel de Carvalho. The relational model is injective for multiplicative exponential linear logic. In Jean-Marc Talbot and Laurent Regnier, editors, *25th EACSL Annual Conference on Computer Science Logic, CSL 2016, August 29 – September 1, 2016, Marseille, France*, volume 62 of *LIPIcs*, pages 41:1–41:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.CSL.2016.41.

27 Daniel de Carvalho. Execution time of λ-terms via denotational semantics and intersection types. *Math. Str. in Comput. Sci.*, 28(7):1169–1203, 2018. doi:10.1017/S0960129516000396.

28 Daniel de Carvalho, Michele Pagani, and Lorenzo Tortora de Falco. A semantic measure of the execution time in linear logic. *Theor. Comput. Sci.*, 412(20):1884–1902, 2011. doi:10.1016/j.tcs.2010.12.017.

29 Daniel de Carvalho and Lorenzo Tortora de Falco. A semantic account of strong normalization in linear logic. *Inf. Comput.*, 248:104–129, 2016. doi:10.1016/j.ic.2015.12.010.

30 Thomas Ehrhard. Hypercoherences: A strongly stable model of linear logic. *Math. Struct. Comput. Sci.*, 3(4):365–385, 1993. doi:10.1017/S0960129500000281.

31 Thomas Ehrhard. Finiteness spaces. *Math. Struct. Comput. Sci.*, 15(4):615–646, 2005. doi:10.1017/S0960129504004645.

32 Thomas Ehrhard. Collapsing non-idempotent intersection types. In *Computer Science Logic (CSL'12) – 26th International Workshop/21st Annual Conference of the EACSL, CSL 2012, September 3-6, 2012, Fontainebleau, France*, volume 16 of *LIPIcs*, pages 259–273. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2012. doi:10.4230/LIPIcs.CSL.2012.259.

33 Thomas Ehrhard. The scott model of linear logic is the extensional collapse of its relational model. *Theor. Comput. Sci.*, 424:20–45, 2012. doi:10.1016/j.tcs.2011.11.027.

34 Thomas Ehrhard. Non-idempotent intersection types in logical form. In Jean Goubault-Larrecq and Barbara König, editors, *Foundations of Software Science and Computation Structures – 23rd International Conference, FOSSACS 2020, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2020, Dublin, Ireland, April 25-30, 2020, Proceedings*, volume 12077 of *Lecture Notes in Computer Science*, pages 198–216. Springer, 2020. doi:10.1007/978-3-030-45231-5_11.

35 José Espírito Santo, Delia Kesner, and Loïc Peyrot. A faithful and quantitative notion of distant reduction for generalized applications. In Patricia Bouyer and Lutz Schröder, editors, *Foundations of Software Science and Computation Structures – 25th International Conference, FOSSACS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings*, volume 13242 of *Lecture Notes in Computer Science*, pages 285–304. Springer, 2022. doi:10.1007/978-3-030-99253-8_15.

36 Jean-Yves Girard. Linear Logic. *Theoretical Computer Science*, 50:1–102, 1987. doi:10.1016/0304-3975(87)90045-4.

37 Jean-Yves Girard. Normal functors, power series and the λ-calculus. *Annals of Pure and Applied Logic*, 37:129–177, 1988. doi:10.1016/0168-0072(88)90025-5.

38 Martin Hyland, Misao Nagayama, John Power, and Giuseppe Rosolini. A category theoretic formulation for engeler-style models of the untyped λ-calculus. *Electronic Notes in Theoretical Computer Science*, 161:43–57, 2006. Proceedings of the Third Irish Conference on the Mathematical Foundations of Computer Science and Information Technology (MFCSIT 2004). doi:10.1016/j.entcs.2006.04.024.

39 Delia Kesner, Loïc Peyrot, and Daniel Ventura. The spirit of node replication. In Stefan Kiefer and Christine Tasson, editors, *Foundations of Software Science and Computation Structures – 24th International Conference, FOSSACS 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 – April 1, 2021, Proceedings*, volume 12650 of *Lecture Notes in Computer Science*, pages 344–364. Springer, 2021. doi:10.1007/978-3-030-71995-1_18.

40 Delia Kesner and Daniel Ventura. Quantitative types for the linear substitution calculus. In Josep Díaz, Ivan Lanese, and Davide Sangiorgi, editors, *Theoretical Computer Science – 8th IFIP TC 1/WG 2.2 International Conference, TCS 2014, Rome, Italy, September 1-3, 2014. Proceedings*, volume 8705 of *Lecture Notes in Computer Science*, pages 296–310. Springer, 2014. doi:10.1007/978-3-662-44602-7_23.

41 Delia Kesner and Pierre Vial. Consuming and persistent types for classical logic. In *LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020*, pages 619–632, 2020. doi:10.1145/3373718.3394774.

42 Delia Kesner and Andrés Viso. Encoding tight typing in a unified framework. In Florin Manea and Alex Simpson, editors, *30th EACSL Annual Conference on Computer Science Logic, CSL 2022, February 14-19, 2022, Göttingen, Germany (Virtual Conference)*, volume 216 of *LIPIcs*, pages 27:1–27:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.CSL.2022.27.

43 A. J. Kfoury and J. B. Wells. Principality and type inference for intersection types using expansion variables. *Theor. Comput. Sci.*, 311(1-3):1–70, 2004. doi:10.1016/j.tcs.2003.10.032.

44 Jean-Louis Krivine. *Lambda-calculus, types and models*. Ellis Horwood series. Masson, 1993.

45 Jim Laird, Giulio Manzonetto, Guy McCusker, and Michele Pagani. Weighted relational models of typed lambda-calculi. In *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*, pages 301–310. IEEE Computer Society, 2013. doi:10.1109/LICS.2013.36.

46 Giulio Manzonetto. A general class of models of H^*. In Rastislav Královic and Damian Niwinski, editors, *Mathematical Foundations of Computer Science 2009, 34th International Symposium, MFCS 2009, Novy Smokovec, High Tatras, Slovakia, August 24-28, 2009. Proceedings*, volume 5734 of *Lecture Notes in Computer Science*, pages 574–586. Springer, 2009. doi:10.1007/978-3-642-03816-7_49.

47 Giulio Manzonetto and Domenico Ruoppolo. Relational graph models, taylor expansion and extensionality. In Bart Jacobs, Alexandra Silva, and Sam Staton, editors, *Proceedings of the 30th Conference on the Mathematical Foundations of Programming Semantics, MFPS 2014, Ithaca, NY, USA, June 12-15, 2014*, volume 308 of *Electronic Notes in Theoretical Computer Science*, pages 245–272. Elsevier, 2014. doi:10.1016/j.entcs.2014.10.014.

48 C.-H. Luke Ong. Quantitative semantics of the lambda calculus: Some generalisations of the relational model. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–12. IEEE Computer Society, 2017. doi:10.1109/LICS.2017.8005064.

49 Luca Paolini, Mauro Piccolo, and Simona Ronchi Della Rocca. Essential and relational models. *Math. Struct. Comput. Sci.*, 27(5):626–650, 2017. doi:10.1017/S0960129515000316.

50 Simona Ronchi Della Rocca. Principal type scheme and unification for intersection type discipline. *Theor. Comput. Sci.*, 59:181–209, 1988. doi:10.1016/0304-3975(88)90101-6.

Infinitary Cut-Elimination via Finite Approximations

Matteo Acclavio
University of Southern Denmark, Odense, Denmark
University of Sussex, Department of Informatics, Brighton, UK

Gianluca Curzi
University of Birmingham, UK
University of Gothenburg, Sweden

Giulio Guerrieri
University of Sussex, Department of Informatics, Brighton, UK

Abstract

We investigate non-wellfounded proof systems based on parsimonious logic, a weaker variant of linear logic where the exponential modality ! is interpreted as a constructor for streams over finite data. Logical consistency is maintained at a global level by adapting a standard progressing criterion. We present an infinitary version of cut-elimination based on finite approximations, and we prove that, in presence of the progressing criterion, it returns well-defined non-wellfounded proofs at its limit. Furthermore, we show that cut-elimination preserves the progressing criterion and various regularity conditions internalizing degrees of proof-theoretical uniformity. Finally, we provide a denotational semantics for our systems based on the relational model.

2012 ACM Subject Classification Theory of computation → Linear logic; Theory of computation → Proof theory

Keywords and phrases cut-elimination, non-wellfounded proofs, parsimonious logic, linear logic, proof theory, approximation, sequent calculus, non-uniform proofs

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.8

Related Version *Full Version*: `https://arxiv.org/abs/2308.07789`

Funding *Matteo Acclavio*: Supported by Villum Fonden, grant no. 50079

Acknowledgements We would like to thank Anupam Das, Abhishek De, Farzad Jafar-Rahmani, Alexis Saurin, Tito (Lê Thành Dung Nguyên), Damiano Mazza and the anonymous reviewers for their useful comments and suggestions.

1 Introduction

Non-wellfounded proof theory studies proofs as possibly infinite (but finitely branching) trees, where logical consistency is maintained via global conditions called *progressing* (or *validity*) *criteria*. In this setting, the so-called *regular* (also called *circular*) proofs receive a special attention, as they admit a finite description in terms of (possibly cyclic) directed graphs.

This area of proof theory makes its first appearance (in its modern guise) in the modal μ-calculus [29, 14]. Since then, it has been extensively investigated from many perspectives (see, e.g., [8, 34, 13, 23]), establishing itself as an ideal setting for manipulating least and greatest fixed points, and hence for modeling induction and coinduction principles.

Non-wellfounded proof theory has been applied to constructive fixed point logics i.e., with a computational interpretation based on the *Curry-Howard correspondence* [35]. A key example can be found in the context of *linear logic* (LL) [21], a logic implementing a finer control on resources thanks to the *exponential* modalities ! and ?. In this framework, the most extensively studied fixed point logic is μMALL, defined as the exponential-free fragment of LL with least and greatest fixed point operators (respectively, μ and its dual ν) [7, 6].

In [7] Baelde and Miller have shown that the exponentials can be recovered in μMALL by exploiting the fixed points operators, i.e., by defining $!A \coloneqq \nu X.(\mathbf{1} \,\&\, A \,\&\, (X \otimes X))$ and $?A \coloneqq \mu X.(\bot \oplus A \oplus (X \parr X))$. As these authors notice, the fixed point-based definition of !

and ? can be regarded as a more permissive variant of the standard exponentials, since a proof of $\nu X.(\mathbf{1} \,\&\, A \,\&\, (X \otimes X))$ could be constructed using different proofs of A, whereas in LL a proof of $!A$ is constructed uniformly using a single proof of A. This proof-theoretical notion of *non-uniformity* is indeed a central feature of the fixed-point exponentials.

However, the above encoding is not free of issues. First, as discussed in full detail in [16], the encoding of the exponentials does not verify the Seely isomorphisms, syntactically expressed by the equivalence $!(A \,\&\, B) \multimap\!\!\circ (!A \otimes !B)$, an essential property for modeling exponentials in LL. Specifically, the fixed-point definition of ! relies on the multiplicative connective $\otimes$, which forces an interpretation of $!A$ based on lists rather than multisets. Secondly, as pointed out in [7], there is a neat mismatch between cut-elimination for the exponentials of LL and the one for the fixed point exponentials of μMALL. While the first problem is related to syntactic deficiencies of the encoding, and does not undermine further investigations on fixed point-based definitions of the exponential modalities, the second one is more critical. These apparent differences between the two exponentials contribute to stressing an important aspect in linear logic modalities, i.e., their *non-canonicity* [31, 12][1].

On a parallel research thread, Mazza [25, 26, 27] studied *parsimonious logic*, a variant of linear logic where the exponential modality ! satisfies Milner's law (i.e., $!A \multimap\!\!\circ A \otimes !A$) and invalidates the implications $!A \multimap !!A$ (*digging*) and $!A \multimap !A \otimes !A$ (*contraction*). In parsimonious logic, a proof of $!A$ can be interpreted as a *stream* over (a finite set of) proofs of A, i.e., as a greatest fixed point, where the linear implications $A \otimes !A \multimap !A$ (*co-absorption*) and $!A \multimap A \otimes !A$ (*absorption*) can be read computationally as the *push* and *pop* operations on streams. More specifically, a formula $!A$ is introduced by an *infinitely branching rule* that takes a finite set of proofs $\mathcal{D}_1, \ldots, \mathcal{D}_n$ of A and a (possibly non-recursive) function $f : \mathbb{N} \to \{1, \ldots, n\}$ as premises, and constructs a proof of $!A$ representing a stream of proofs of the form $\mathfrak{S} = (\mathcal{D}_{f(0)}, \mathcal{D}_{f(1)}, \ldots, \mathcal{D}_{f(n)}, \ldots)$. Hence, parsimonious logic exponential modalities exploit in an essential way the above-mentioned proof-theoretical non-uniformity, which in turn deeply interfaces with notions of non-uniformity from computational complexity [27].

The analysis of parsimonious logic conducted in [26, 27] reveals that fixed point definitions of the exponentials are better behaving when digging and contraction are discarded. On the other hand, the co-absorption rule cannot be derived in LL, and so it prevents parsimonious logic becoming a genuine subsystem of the latter. This led the authors of the present paper to introduce *parsimonious linear logic*, a subsystem of linear logic (in particular, *co-absorption-free*) that nonetheless allows a stream-based interpretation of the exponentials.

We present two finitary proof systems for parsimonious linear logic: the system nuPLL, supporting non-uniform exponentials, and PLL, a fully uniform version. We investigate non-wellfounded counterparts of nuPLL and PLL, adapting to our setting the progressing criterion to maintain logical consistency. To recover the proof-theoretical behavior of nuPLL and PLL, we identify further global conditions on non-wellfounded proofs, that is, some forms of regularity to capture the notions of uniformity and non-uniformity. This leads us to two main non-wellfounded proof systems: *regular parsimonious linear logic* (rPLL^∞), defined via the *regularity* condition and corresponding to PLL, and *weakly regular parsimonious linear logic* (wrPLL^∞), defined via a *weak regularity* condition and corresponding to nuPLL.

The major contribution of this paper is the study of continuous cut-elimination in the setting of non-wellfounded parsimonious linear logic. We first introduce Scott-domains of partially defined non-wellfounded proofs, ordered by an approximation relation. Here, undefinedness in proofs is expressed by the use of an axiom introducing an arbitrary sequent;

[1] One can construct LL proof systems with alternative (not equivalent) exponential modalities, see [28].

this approach is analogous to the one used to define Böhm trees in the λ-calculus: intuitively, a non-wellfounded proof is kind of like a Böhm tree that may be described by its finite approximations, with the difference that – in the λ-calculus – Böhm trees, and therefore their finite approximations, are normal (that is, cut-free) by definition, whereas here proofs need not be cut-free and so the approximations too may contain cuts. Then, we define special infinitary proof rewriting strategies called *maximal and continuous infinitary cut-elimination strategies* (mc-ices) which compute (Scott-)continuous functions. Productivity in this framework is established by showing that, in presence of a good global condition (progressing, regularity or weak regularity), these continuous functions return totally defined cut-free non-wellfounded proofs and preserve the global condition: progressing (Theorem 33.1), and regularity or weak regularity (Theorem 33.2).

On a technical side, we stress that our methods and results distinguish from previous approaches to cut-elimination in a non-wellfounded setting in many respects. First, we get rid of many technical notions typically introduced to prove infinitary cut-elimination, such as the *multicut rule* or the *fairness conditions* (as in, e.g., [20, 6]), as these notions are subsumed by a *finitary approximation* approach to cut-elimination. Furthermore, we prove productivity of cut-elimination and preservation of the progressing condition in a more direct and constructive way, i.e., without going through auxiliary proof systems and avoiding arguments by contradiction (see, e.g., [6]). Finally, we prove for the first time preservation of regularity properties under continuous cut-elimination, essentially exploiting methods for compressing transfinite rewriting sequences to ω-long ones from [36, 25, 33].

Finally, we define a denotational semantics for non-wellfounded parsimonious logic based on the relational model, with a standard multiset-based interpretation of the exponentials, and we show that this semantics is preserved under continuous cut-elimination (Theorem 38). We also prove that extending non-wellfounded parsimonious linear logic with digging prevents the existence of a cut-elimination result preserving the semantics (Theorem 40). Therefore, the impossibility of a stream-based definition of ! that validates digging (and contraction).

Additional details of the proofs are provided in the extended version of this paper [2].

2 Preliminary notions

In this section we recall some basic notions from (non-wellfounded) proof theory, fixing the notation that will be adopted in this paper.

2.1 Derivations and coderivations

We assume that the reader is familiar with the syntax of sequent calculus, e.g. [37]. Here we specify some conventions adopted to simplify the content of this paper.

We consider (**sequent**) **rules** of the form $\mathsf{r}\,\dfrac{}{\Gamma}$ or $\mathsf{r}\,\dfrac{\Gamma_1}{\Gamma}$ or $\mathsf{r}\,\dfrac{\Gamma_1 \quad \Gamma_2}{\Gamma}$, and we refer to the sequents Γ_1 and Γ_2 as the **premises**, and to the sequent Γ as the **conclusion** of the rule r. To avoid technicalities of the sequents-as-lists presentation, we follow [6] and we consider **sequents** as *sets of occurrences of formulas* from a given set of formulas. In particular, when we refer to a formula in a sequent we always consider a *specific occurrence* of it.

▶ **Definition 1.** *A (binary, possibly infinite)* ***tree*** $\mathcal{T}$ *is a subset of words in* $\{1,2\}^*$ *that contains the empty word* ϵ *(the* ***root*** *of* $\mathcal{T}$*) and is* ordered-prefix-closed *(i.e., if* $n \in \{1,2\}$ *and* $vn \in \mathcal{T}$*, then* $v \in \mathcal{T}$*, and if moreover* $v2 \in \mathcal{T}$*, then* $v1 \in \mathcal{T}$*). The elements of* $\mathcal{T}$ *are called* ***nodes*** *and their* ***height*** *is the length of the word. A* ***child*** *of* $v \in \mathcal{T}$ *is any* $vn \in \mathcal{T}$ *with* $n \in \{1,2\}$*. The*

$$\mathsf{ax}\,\frac{}{A, A^\perp} \quad \mathsf{cut}\,\frac{\Gamma, A \quad A^\perp, \Delta}{\Gamma, \Delta} \quad \otimes\,\frac{\Gamma, A \quad B, \Delta}{\Gamma, \Delta, A \otimes B} \quad \parr\,\frac{\Gamma, A, B}{\Gamma, A \parr B} \quad 1\,\frac{}{1} \quad \bot\,\frac{\Gamma}{\Gamma, \bot} \quad \mathsf{f!p}\,\frac{\Gamma, A}{?\Gamma, !A} \quad ?\mathsf{w}\,\frac{\Gamma}{\Gamma, ?A} \quad ?\mathsf{b}\,\frac{\Gamma, A, ?A}{\Gamma, ?A}$$

Figure 1 Sequent calculus rules of PLL.

prefix order *is a partial order* $\leq_{\mathcal{T}}$ *on* $\mathcal{T}$ *defined by: for any* $v, v' \in \mathcal{T}$, $v \leq_{\mathcal{T}} v'$ *if* $v' = vw$ *for some* $w \in \{1,2\}^*$. *A maximal element of* $\leq_{\mathcal{T}}$ *is a* ***leaf*** *of* $\mathcal{T}$. *A* ***branch*** *of* $\mathcal{T}$ *is a set* $\mathcal{B} \subseteq \mathcal{T}$ *such that* $\epsilon \in \mathcal{B}$ *and if* $w \in \mathcal{B}$ *is not a leaf of* $\mathcal{T}$ *then* w *has exactly one child in* $\mathcal{B}$.

A ***coderivation*** *over a set of rules* $\mathcal{S}$ *is a labeling* $\mathcal{D}$ *of a tree* $\mathcal{T}$ *by sequents such that if* v *is a node of* $\mathcal{T}$ *with children* $v_1, \dots, v_n$ *(with* $n \in \{0,1,2\}$*), then there is an occurrence of a rule* r *in* $\mathcal{S}$ *with conclusion the sequent* $\mathcal{D}(v)$ *and premises the sequents* $\mathcal{D}(v_1), \dots, \mathcal{D}(v_n)$. *The* ***height*** *of* r *in* $\mathcal{D}$ *is the height of the node* $v \in \mathcal{T}$ *such that* $\mathcal{D}(v)$ *is the conclusion of* r.

The ***conclusion*** *of* $\mathcal{D}$ *is the sequent* $\mathcal{D}(\epsilon)$. *If* v *is a node of the tree, the* ***sub-coderivation*** *of* $\mathcal{D}$ *rooted at* v *is the coderivation* $\mathcal{D}_v$ *defined by* $\mathcal{D}_v(w) = \mathcal{D}(vw)$.

A coderivation $\mathcal{D}$ *is* r***-free*** *(for a rule* $\mathsf{r} \in \mathcal{S}$*) if it contains no occurrence of* r. *It is* ***regular*** *if it has finitely many distinct sub-coderivations; it is* ***non-wellfounded*** *if it labels an infinite tree, and it is a* ***derivation*** *(with* ***size*** $|\mathcal{D}| \in \mathbb{N}$*) if it labels a finite tree (with* $|\mathcal{D}|$ *nodes).*

Given a set of coderivations X, *a sequent* Γ *is* ***provable*** *in* X *(noted* $\vdash_{\mathsf{X}} \Gamma$*) if there is a coderivation in* X *with conclusion* Γ.

While derivations are represented as finite trees, regular coderivations (also called circular or cyclic) can be represented as *finite* directed (possibly cyclic) graphs: a cycle is created by linking the roots of two identical subcoderivations.

▶ **Definition 2.** *Let* $\mathcal{D}$ *be a coderivation labeling a tree* $\mathcal{T}$. *A* ***bar*** *(resp.* ***prebar****) of* $\mathcal{D}$ *is a set* $\mathcal{V} \subseteq \mathcal{T}$ *where:*

- *any branch (resp. infinite branch) of the tree* $\mathcal{T}$ *underlying* $\mathcal{D}$ *contains a node in* $\mathcal{V}$;
- *any pair of nodes in* $\mathcal{V}$ *are mutually incomparable with respect to the prefix order* $\leq_{\mathcal{T}}$.

The ***height*** *of a prebar* $\mathcal{V}$ *of* $\mathcal{D}$ *is the minimal height of the nodes of* $\mathcal{V}$.

3 Parsimonious Linear Logic

In this paper we consider the set of **formulas** for propositional multiplicative-exponential linear logic with units (MELL). These are generated by a countable set of propositional variables $\mathcal{A} = \{X, Y, \dots\}$ using the following grammar:

$$A, B ::= X \mid X^\perp \mid A \otimes B \mid A \parr B \mid !A \mid ?A \mid 1 \mid \bot$$

A **!-formula** (resp. **?-formula**) is a formula of the form $!A$ (resp. $?A$). **Linear negation** $(\cdot)^\perp$ is defined by De Morgan's laws $(A^\perp)^\perp = A$, $(A \otimes B)^\perp = A^\perp \parr B^\perp$, $(!A)^\perp = ?A^\perp$, and $(1)^\perp = \bot$ while **linear implication** is defined as $A \multimap B := A^\perp \parr B$.

▶ **Definition 3.** ***Parsimonious linear logic****, denoted by* PLL*, is the set of rules in Figure 1, that is,* ***axiom*** *(*ax*),* ***cut*** *(*cut*),* ***tensor*** *(*$\otimes$*),* ***par*** *(*$\parr$*),* ***one*** *(*1*),* ***bottom*** *(*$\bot$*),* ***functorial promotion*** *(*f!p*),* ***weakening*** *(*?w*),* ***absorption*** *(*?b*). Rules* ax, $\otimes$, $\parr$, 1 *and* $\bot$ *are called* ***multiplicative****, while rules* f!p, ?w *and* ?b *are called* ***exponential****. We also denote by* PLL *the set of derivations over the rules in* PLL.

▶ **Example 4.** Figure 2 gives some examples of derivation in PLL. The (distinct) derivations $\underline{0}$ and $\underline{1}$ prove the same formula $\mathbf{N} := !(X \multimap X) \multimap X \multimap X$. The derivation $\mathcal{D}_{\mathsf{abs}}$ proves the *absorption law* $!A \multimap A \otimes !A$; the derivation $\mathcal{D}_{\mathsf{der}}$ proves the *dereliction law* $!A \multimap A$.

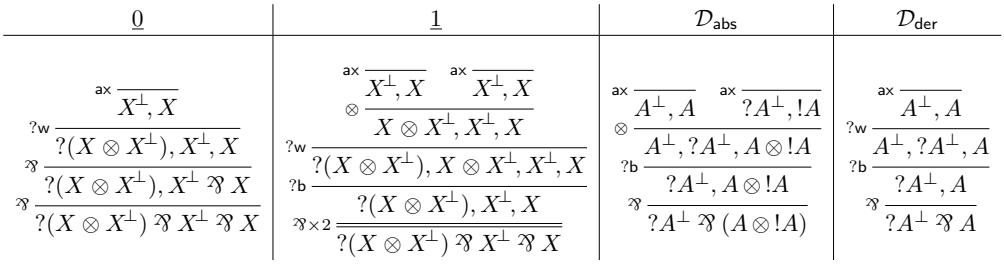


Figure 2 Examples of derivations in PLL.

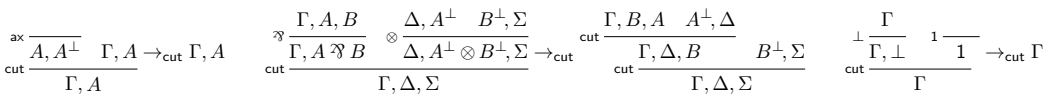


Figure 3 Multiplicative cut-elimination steps in PLL.

The **cut-elimination** relation $\to_{\mathsf{cut}}$ in PLL is the union of **principal** cut-elimination steps in Figure 3 (**multiplicative**) and Figure 4 (**exponential**) and **commutative** cut-elimination steps in Figure 5. The reflexive-transitive closure of $\to_{\mathsf{cut}}$ is noted $\to^*_{\mathsf{cut}}$.

▶ **Theorem 5.** *For every $\mathcal{D} \in \mathsf{PLL}$, there is a cut-free $\mathcal{D}' \in \mathsf{PLL}$ such that $\mathcal{D} \to^*_{\mathsf{cut}} \mathcal{D}'$.*

Sketch of proof. We associate with any derivation $\mathcal{D}$ in PLL a derivation $\mathcal{D}^{\spadesuit}$ in MELL sequent calculus. Thanks to additional commutative cut-elimination steps, we prove that cut-elimination in MELL rewrites $\mathcal{D}^{\spadesuit}$ to the translation of a derivation in PLL. The termination of cut-elimination in PLL follows from strong normalisation of (second-order) MELL [30]. ◀

Akin to light linear logic [22, 24, 32], the exponential rules of PLL are weaker than those in MELL: the usual promotion rule is replaced by f!p (*functorial promotion*), and the usual contraction and dereliction rules by ?b. As a consequence, the *digging* formula $!A \multimap !!A$ and the *contraction* formula $!A \multimap !A \otimes !A$ are not provable in PLL (unlike the dereliction formula, Example 4). This allows us to interpret computationally these weaker exponentials in terms of streams, as well as to control the complexity of cut-elimination [26, 27].

It is easy to show that MELL = PLL + digging: if we add the digging formula as an axiom (or equivalently, the *digging rule* ??d in Figure 13) to the set of rules in Figure 1, then the contraction formula becomes provable, and the obtained proof system coincides with MELL.

4 Non-wellfounded Parsimonious Linear Logic

In linear logic, a formula $!A$ is interpreted as the availability of A at will. This intuition still holds in PLL. Indeed, the Curry-Howard correspondence interprets rule f!p introducing the modality ! as an operator taking a derivation $\mathcal{D}$ of A and creating a (infinite) *stream* $(\mathcal{D}, \mathcal{D}, \dots, \mathcal{D}, \dots)$ of copies of the proof $\mathcal{D}$. Each element of the stream is accessed via the cut-elimination step f!p vs ?b in Figure 4: rule ?b is interpreted as an operator *popping* one copy of $\mathcal{D}$ out of the stream. Pushing these ideas further, Mazza [26] introduced *parsimonious logic* **PL**, a type system (comprising rules f!p and ?b) characterizing the logspace decidable problems.

Mazza and Terui then introduced in [27] another type system, $\mathbf{nuPL}_{\forall \ell}$, based on parsimonious logic and capturing the complexity class $\mathbf{P}/\mathsf{poly}$ (i.e., the problems decidable by polynomial size families of Boolean circuits [5]). Their system is endowed with a *non-uniform* version of the functorial promotion, which takes a finite set of proofs $\mathcal{D}_1, \dots, \mathcal{D}_n$ of A and a

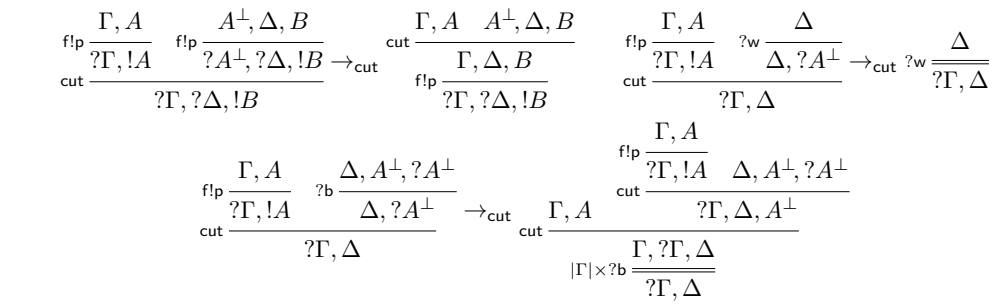


Figure 4 Exponential cut-elimination steps in PLL.

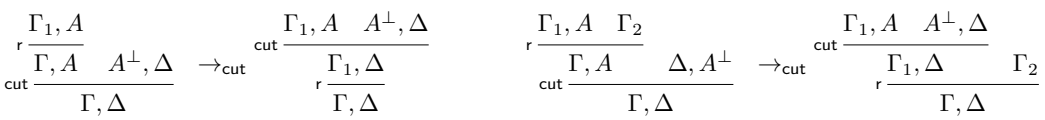


Figure 5 Commutative cut-elimination steps in PLL, where $\mathsf{r} \neq \mathsf{cut}$.

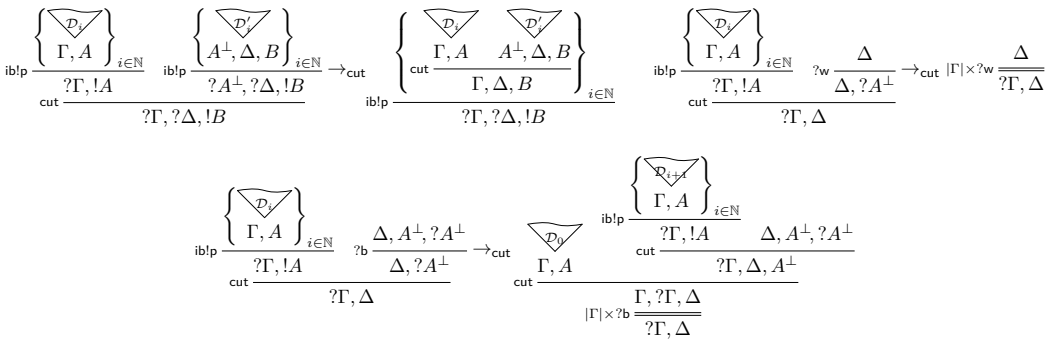
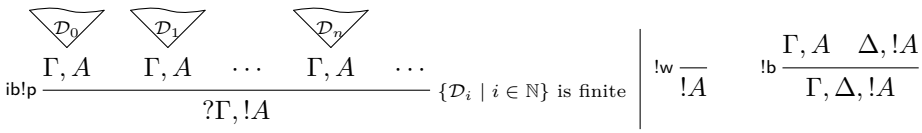


Figure 6 Exponential cut-elimination steps in nuPLL.

(possibly non-recursive) function $f\colon \mathbb{N} \to \{1, \dots, n\}$ as premises, and constructs a proof of $!A$ modeling the stream $(\mathcal{D}_{f(0)}, \mathcal{D}_{f(1)}, \dots, \mathcal{D}_{f(n)}, \dots)$. This typing rule is the key tool to encode the so-called *advices* for Turing machines, an essential step to show completeness for $\mathbf{P}/\mathsf{poly}$.

In a similar vein, we can endow PLL with a non-uniform version of f!p called **infinitely branching promotion** (ib!p), which constructs a stream $(\mathcal{D}_0, \mathcal{D}_1, \dots, \mathcal{D}_n, \dots)$ with finite support, i.e., made of *finitely* many distinct derivations (of the same conclusion):[2]



(1)

The side condition on ib!p provides a proof theoretic counterpart to the function $f\colon \mathbb{N} \to \{1, \dots, n\}$ in $\mathbf{nuPL}_{\forall \ell}$. Clearly, f!p is subsumed by the rule ib!p, as it corresponds to the special (uniform) case where $\mathcal{D}_i = \mathcal{D}_{i+1}$ for all $i \in \mathbb{N}$.

[2] Rule ib!p is reminiscent of the ω-rule used in (first-order) Peano arithmetic to derive formulas of the form $\forall x \phi$ that cannot be proven in a uniform way.

$$\mathcal{D}_{\lightning} \coloneqq \dfrac{\dfrac{}{A^{\perp}, A}\,\mathsf{ax} \qquad \dfrac{\dfrac{}{A^{\perp}, A}\,\mathsf{ax} \qquad \dfrac{\vdots}{\Gamma, A}}{\Gamma, A}\,\mathsf{cut}}{\Gamma, A}\,\mathsf{cut} \qquad \mathcal{D}_{?} \coloneqq \dfrac{\dfrac{\dfrac{\vdots}{A, A, ?A}\,?\mathsf{b}}{A, ?A}\,?\mathsf{b}}{?A}\,?\mathsf{b}$$

Figure 7 Two non-wellfounded and non-progressing coderivations in PLL^{∞}.

▶ **Definition 6.** *We define the set of rules* $\mathsf{nuPLL} \coloneqq \{\mathsf{ax}, \otimes, \parr, 1, \perp, \mathsf{cut}, ?\mathsf{b}, ?\mathsf{w}, \mathsf{ib!p}\}$. *We also denote by* nuPLL *the set of derivations over the rules in* nuPLL.[3]

There are some notable differences between nuPLL and Mazza and Terui's original system $\mathbf{nuPL}_{\forall\ell}$ [27]. As opposed to nuPLL, $\mathbf{nuPL}_{\forall\ell}$ is formulated as an intuitionistic (type) system. Furthermore, to achieve completeness for $\mathbf{P}/\mathsf{poly}$, these authors introduced second-order quantifiers and the co-absorption (!b) and co-weakening (!w) rules displayed in (1).

Cut-elimination steps for nuPLL are in Figures 3, 5, and 6. In particular, the step ib!p-vs-?b in Figure 6 *pops* the first premise $\mathcal{D}_0$ of ib!p out of the stream $(\mathcal{D}_0, \mathcal{D}_1, \dots, \mathcal{D}_n, \dots)$.

4.1 From infinitely branching proofs to non-wellfounded proofs

In this paper we explore a dual approach to the one of $\mathbf{nuPL}_{\forall\ell}$ (and nuPLL): instead of considering (wellfounded) derivations with infinite branching, we consider (non-wellfounded) coderivations with finite branching. For this purpose, the infinitary rule ib!p of nuPLL is replaced by the binary rule below, called **conditional promotion** (c!p):

$$\mathsf{c!p}\,\frac{\Gamma, A \quad ?\Gamma, !A}{?\Gamma, !A} \tag{2}$$

▶ **Definition 7.** *We define the set of rules* $\mathsf{PLL}^{\infty} \coloneqq \{\mathsf{ax}, \otimes, \parr, 1, \perp, \mathsf{cut}, ?\mathsf{b}, ?\mathsf{w}, \mathsf{c!p}\}$. *We also denote by* PLL^{∞} *the set of coderivations over the rules in* PLL^{∞}.

In other words, PLL^{∞} is the set of coderivations generated by the same rules as PLL, except that f!p is replaced by c!p. From now on, we will only consider coderivations in PLL^{∞}.

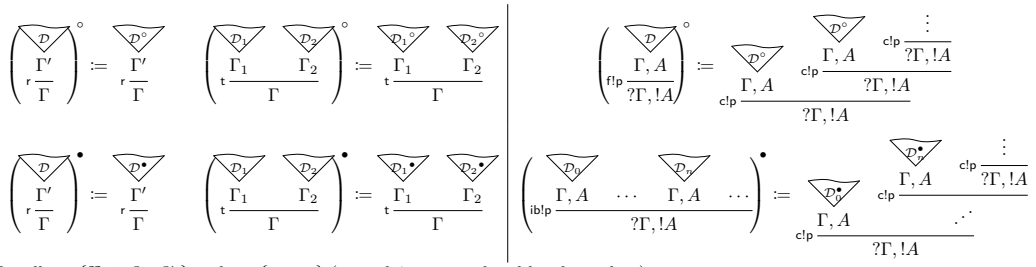
▶ **Example 8.** Figure 7 shows two non-wellfounded coderivations in PLL^{∞}: $\mathcal{D}_{\lightning}$ (resp. $\mathcal{D}_{?}$) has an infinite branch of cut (resp. ?b) rules, and is (resp. is not) regular.

We can embed PLL and nuPLL into PLL^{∞} via the conclusion-preserving translations $(\cdot)^{\circ} \colon \mathsf{PLL} \to \mathsf{PLL}^{\infty}$ and $(\cdot)^{\bullet} \colon \mathsf{nuPLL} \to \mathsf{PLL}^{\infty}$ defined in Figure 8 by induction on derivations: they map all rules to themselves except f!p and ib!p, which are "unpacked" into non-wellfounded coderivations that iterate infinitely many times the rule c!p.

An infinite chain of c!p rules (Figure 9) is a structure of interest in itself in PLL^{∞}.

▶ **Definition 9.** *A* ***non-wellfounded box*** *(nwb for short) is a coderivation* $\mathcal{D} \in \mathsf{PLL}^{\infty}$ *with an infinite branch* $\{\epsilon, 2, 22, \dots\}$ *(the* ***main branch*** *of* $\mathcal{D}$*) all labeled by* c!p *rules as in Figure 9, where* $!A$ *in the conclusion is the* ***principal formula*** *of* $\mathcal{D}$*, and* $\mathcal{D}_0, \mathcal{D}_1, \dots$ *are the* ***calls*** *of* $\mathcal{D}$*. We denote* $\mathcal{D}$ *by* $\mathsf{c!p}_{(\mathcal{D}_0, \dots, \mathcal{D}_n, \dots)}$.

[3] To be rigorous, this requires a slight change in Definition 1: the tree labeled by a derivation in nuPLL must be over $\mathbb{N}^{\omega}$ instead of $\{1, 2\}^*$, in order to deal with infinitely branching derivations.



for all $\mathsf{r} \in \{⅋, \bot, ?\mathsf{w}, ?\mathsf{b}\}$ and $\mathsf{t} \in \{\mathsf{cut}, \otimes\}$ (ax and $\mathbf{1}$ are translated by themselves).

Figure 8 Translations $(\cdot)^\circ$ from PLL to PLL$^\infty$, and $(\cdot)^\bullet$ from nuPLL to PLL$^\infty$.

$$\mathcal{D} = \mathsf{c!p}_{(\mathcal{D}_0,\dots,\mathcal{D}_n,\dots)} = \dfrac{\dfrac{\dfrac{\mathcal{D}_0}{\Gamma, A}}{\mathsf{c!p}\dfrac{\Gamma, A \qquad \cdots}{?\Gamma, !A}} \qquad \dfrac{\dfrac{\mathcal{D}_1}{\Gamma, A}}{\mathsf{c!p}\dfrac{\Gamma, A \qquad \cdots}{?\Gamma, !A}} \qquad \dfrac{\dfrac{\mathcal{D}_n}{\Gamma, A}}{\mathsf{c!p}\dfrac{\Gamma, A \qquad \vdots}{?\Gamma, !A}} \qquad \cdots}{?\Gamma, !A}$$

Figure 9 A non-wellfounded box in PLL$^\infty$.

*Let $\mathfrak{S} = \mathsf{c!p}_{(\mathcal{D}_0,\dots,\mathcal{D}_n,\dots)}$ be a nwb. We may write $\mathfrak{S}(i)$ to denote $\mathcal{D}_i$. We say that $\mathfrak{S}$ has **finite support** (resp. is **periodic** with **period** k) if $\{\mathfrak{S}(i) \mid i \in \mathbb{N}\}$ is finite (resp. if $\mathfrak{S}(i) = \mathfrak{S}(k+i)$ for any $i \in \mathbb{N}$). A coderivation $\mathcal{D}$ has **finite support** (resp. is **periodic**) if any nwb in $\mathcal{D}$ has finite support (resp. is periodic).*

▶ **Example 10.** The only cut-free derivations of the formula $\mathbf{N} \coloneqq !(X \multimap X) \multimap X \multimap X$ are of the form $\underline{n}$ below on the right, for all $n \in \mathbb{N}$, up to permutations of the rules $?\mathsf{w}$, $?\mathsf{b}$ and $\otimes$ (the derivations $\underline{0}$ and $\underline{1}$ in Example 4 are special cases of it)

$$\mathsf{c!p}_{(\underline{i_0},\dots,\underline{i_n},\dots)} = \dfrac{\dfrac{\dfrac{\dfrac{i_0}{\mathbf{N}}}{\mathsf{c!p}\dfrac{\mathbf{N} \qquad \cdots}{!\mathbf{N}}} \qquad \dfrac{\dfrac{i_1}{\mathbf{N}}}{\mathsf{c!p}\dfrac{\mathbf{N} \qquad \cdots}{!\mathbf{N}}} \qquad \dfrac{\dfrac{i_n}{\mathbf{N}}}{\mathsf{c!p}\dfrac{\mathbf{N} \qquad \vdots}{!\mathbf{N}}} \qquad \cdots}{!\mathbf{N}}$$

$$\underline{n} \coloneqq \dfrac{\dfrac{\dfrac{\dfrac{\mathsf{ax}\dfrac{}{X^\bot, X} \quad \mathsf{ax}\dfrac{}{X^\bot, X}}{\otimes\dfrac{}{X \otimes X^\bot, X^\bot, X}} \quad \mathsf{ax}\dfrac{}{X^\bot, X}}{\otimes\times(n-1)\dfrac{}{X \otimes X^\bot, \dots, X \otimes X^\bot, X^\bot, X}}}{?\mathsf{w}\dfrac{}{?(X \otimes X^\bot), X \otimes X^\bot, \dots, X \otimes X^\bot, X^\bot, X}}}{?\mathsf{b}\times n\dfrac{}{?(X \otimes X^\bot), X^\bot, X}}}{⅋\times 2\dfrac{}{?(X \otimes X^\bot) ⅋ X^\bot ⅋ X}} \tag{3}$$

Consider the nwb $\mathsf{c!p}_{(\underline{i_0},\dots,\underline{i_n},\dots)}$ above on the left, proving the formula $!\mathbf{N}$, where $i_j \in \{0,1\}$ for all $j \in \mathbb{N}$. Thus $\mathsf{c!p}_{(\underline{i_0},\dots,\underline{i_n},\dots)}$ has finite support, as its only calls can be $\underline{0}$ or $\underline{1}$, and it is periodic if and only if so is the infinite sequence $(i_0, \dots, i_n, \dots) \in \{0,1\}^\omega$.

The *cut-elimination* steps $\to_{\mathsf{cut}}$ for PLL$^\infty$ are in Figures 3, 5, and 10. Computationally, they allow the $\mathsf{c!p}$ rule to be interpreted as a *coinductive* definition of a stream of type $!A$ from a stream of the same type to which an element of type A is prepended. In particular, the cut-elimination step $\mathsf{c!p}$ vs $?\mathsf{b}$ accesses the head of a stream: rule $?\mathsf{b}$ acts as a *pop* operator.

As a consequence, the nwb in Figure 9 constructs a stream $(\mathcal{D}_0, \mathcal{D}_1, \dots, \mathcal{D}_n, \dots)$ similarly to $\mathsf{ib!p}$ but, unlike the latter, all the $\mathcal{D}_i$'s may be pairwise distinct. The reader expert in linear logic can see a nwb as a box with possibly *infinitely many* distinct contents (its calls), while usual linear logic boxes (and $\mathsf{f!p}$ in PLL) provide infinitely many copies of the *same* content.

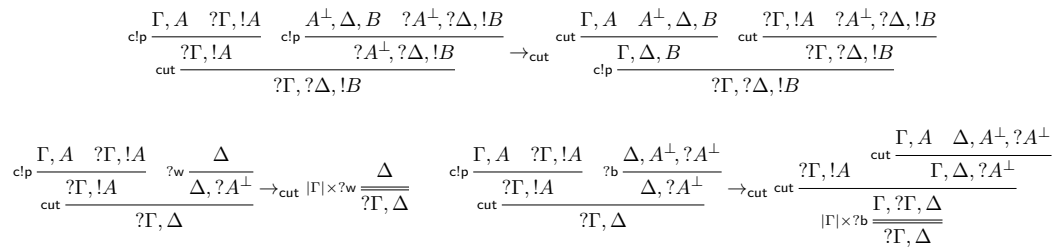


Figure 10 Exponential cut-elimination steps for coderivations of PLL^∞.

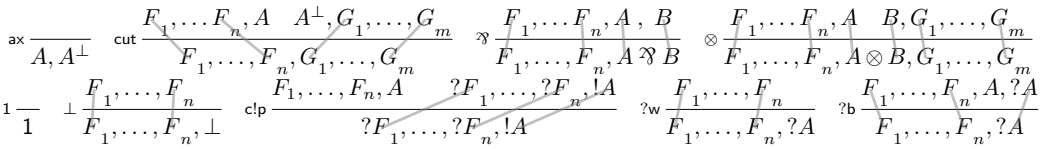


Figure 11 PLL^∞ rules: edges connect a formula in the conclusion with its parent(s) in a premise.

Rules f!p in PLL and ib!p in nuPLL are mapped by $(\cdot)^\circ$ and $(\cdot)^\bullet$ into nwbs, which are non-wellfounded coderivations. Hence, the cut-elimination steps f!p vs f!p in PLL and ib!p vs ib!p in nuPLL can only be simulated by infinitely many cut-elimination steps in PLL^∞.

Note that $\mathcal{D}_\natural \in \mathsf{PLL}^\infty$ in Figure 7 is not cut-free, and if $\mathcal{D}_\natural \to_{\mathsf{cut}} \mathcal{D}$ then $\mathcal{D} = \mathcal{D}_\natural$: thus $\mathcal{D}_\natural$ cannot reduce to a cut-free coderivation, and so the cut-elimination theorem fails in PLL^∞.

4.2 Consistency via a progressing criterion

In a non-wellfounded setting such as PLL^∞, any sequent is provable. Indeed, the (non-wellfounded) coderivation $\mathcal{D}_\natural$ in Figure 7 shows that any non-empty sequent (in particular, any formula) is provable in PLL^∞, and the empty sequent is provable in PLL^∞ by applying the cut rule on the conclusions B and $B^\perp$ (for any formula B) of two derivations $\mathcal{D}_\natural$.

The standard way to recover logical consistency in non-wellfounded proof theory is to introduce a global soundness condition on coderivations, called *progressing criterion* [23, 13]. In PLL^∞, this criterion relies on tracking occurrences of !-formulas in a coderivation.

▶ **Definition 11.** *Let $\mathcal{D}$ be a coderivation in PLL^∞. It is **weakly progressing** if every infinite branch contains infinitely many right premises of* c!p*-rules.*

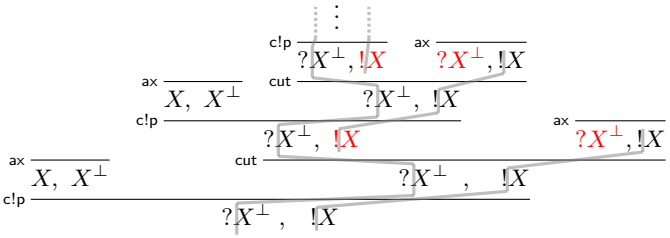
An occurrence of a formula in a premise of a rule r *is the **parent** of an occurrence of a formula in the conclusion if they are connected according to the edges depicted in Figure 11.*

*A **!-thread** (resp. **?-thread**) in $\mathcal{D}$ is a maximal sequence $(A_i)_{i\in I}$ of !-formulas (resp. ?-formulas) for some downward-closed $I \subseteq \mathbb{N}$ such that A_{i+1} is the parent of A_i for all $i \in I$. A !-thread $(A_i)_{i\in I}$ is **progressing** if A_j is in the conclusion of a* c!p *for infinitely many $j \in I$. $\mathcal{D}$ is **progressing** if every infinite branch contains a progressing !-thread. We define* pPLL^∞ *(resp.* wpPLL^∞*) as the set of progressing (resp. weak-progressing) coderivations in* PLL^∞.

▶ Remark 12. Clearly, any progressing coderivation is weakly progressing too, but the converse fails (Example 13), therefore $\mathsf{pPLL}^\infty \subsetneq \mathsf{wpPLL}^\infty$. Moreover, the main branch of any nwb contains by definition a progressing !-thread of its principal formula.

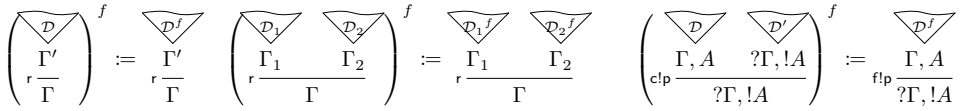
▶ **Example 13.** Coderivations in Figure 7 are not weakly progressing (hence, not progressing): the rightmost branch of $\mathcal{D}_\natural$, i.e., the branch $\{\epsilon, 2, 22, \dots\}$, and the unique branch of $\mathcal{D}_?$ are infinite and contain no c!p-rules. In contrast, the nwb $\mathsf{c!p}_{(\underline{i_0},\dots,\underline{i_n},\dots)}$ in Example 10 is

progressing by Remark 12, since its main branch is the only infinite branch. Below, a regular, weakly progressing but not progressing coderivation ($!X$ in the conclusion of c!p is a cut formula, so the branch $\{\epsilon, 2, 21, 212, 2121, \dots\}$ is infinite but has no progressing !-thread).



▶ **Lemma 14.** *Let* Γ *be a sequent. Then,* $\vdash_{\mathsf{PLL}} \Gamma$ *if and only if* $\vdash_{\mathsf{wpPLL}^\infty} \Gamma$.

Proof. Given $\mathcal{D} \in \mathsf{PLL}$, $\mathcal{D}^\circ \in \mathsf{PLL}^\infty$ preserves the conclusion and is progressing, hence weakly progressing (see Remark 12). Conversely, given a weakly progressing coderivation $\mathcal{D}$, we define a derivation $\mathcal{D}^f \in \mathsf{PLL}$ with the same conclusion by applying, bottom-up, the translation:



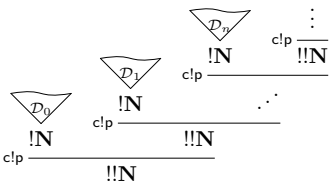
with $\mathsf{r} \neq \mathsf{c!p}$. Note that the derivation $\mathcal{D}^f$ is well-defined because $\mathcal{D}$ is weakly progressing. ◀

▶ **Corollary 15.** *The empty sequent is not provable in* wpPLL^∞ *(and hence in* pPLL^∞*).*

Proof. If the empty sequent were provable in wpPLL^∞, then there would be a cut-free derivation $\mathcal{D} \in \mathsf{PLL}$ of the empty sequent by Lemma 14 and Theorem 5, but this is impossible since cut is the only rule in PLL that could have the empty sequent in its conclusion. ◀

4.3 Recovering (weak forms of) regularity

The progressing criterion cannot capture the finiteness condition of the rule ib!p in the derivations in nuPLL. By means of example, consider the nwb below, which is progressing but cannot be the image of the rule ib!p via $(\cdot)^\bullet$ (see Figure 8) since $\{\mathcal{D}_i \mid i \in \mathbb{N}\}$ is infinite.



with $\mathcal{D}_i = \mathsf{c!p}_{(\underbrace{1,\dots,1}_{i},0,\dots)}$ for each $i \in \mathbb{N}$. (4)

To identify in pPLL^∞ the coderivations corresponding to derivations in nuPLL and in PLL via the translations $(\cdot)^\bullet$ and $(\cdot)^\circ$, respectively, we need additional conditions.

▶ **Definition 16.** *A coderivation is* ***weakly regular*** *if it has only finitely many distinct sub-coderivations whose conclusions are left premises of* c!p*-rules; it is* ***finitely expandable*** *if any branch contains finitely many* cut *and* ?b *rules. We denote by* wrPLL^∞ *(resp.* rPLL^∞*) the set of weakly regular (resp. regular) and finitely expandable coderivations in* pPLL^∞.

▶ Remark 17. Regularity implies weak regularity and the converse fails as shown in Example 18 below, so $\mathsf{rPLL}^\infty \subsetneq \mathsf{wrPLL}^\infty$. Given $\mathcal{D} \in \mathsf{PLL}^\infty$ progressing and finitely expandable, it is regular (resp. weakly regular) if and only if any nwb in $\mathcal{D}$ is periodic (resp. has finite support).

▶ **Example 18.** Coderivations $\mathcal{D}_{\not i}$ and $\mathcal{D}_{?}$ in Figure 7 are not finitely expandable, as their infinite branch has infinitely many cut or ?b, but they are weakly regular, since they have no c!p rules. The coderivation in (4) is not weakly regular because $\{\mathcal{D}_i \mid i \in \mathbb{N}\}$ is infinite.

An example of a weakly regular but not regular coderivation is the nwb $\mathsf{c!p}_{(\underline{i_0},\ldots,\underline{i_n},\ldots)}$ in Example 10 when the infinite sequence $(i_j)_{j\in\mathbb{N}} \in \{0,1\}^\omega$ is not periodic: $\underline{0}$ and $\underline{1}$ are the only coderivations ending in the left premise of a c!p rule (so the nwb is weakly regular), but there are infinitely many distinct coderivations ending in the right premise of a c!p rule (so the nwb is not regular). Moreover, that nwb is finitely expandable, as it contains no ?b or cut.

The sets rPLL^∞ and wrPLL^∞ are the non-wellfounded counterparts of PLL and nuPLL, respectively. Indeed, we have the following correspondence via the translations $(\cdot)^\circ$ and $(\cdot)^\bullet$.

▶ **Proposition 19.**

1. *If* $\mathcal{D} \in \mathsf{PLL}$ *(resp.* $\mathcal{D} \in \mathsf{nuPLL}$*) with conclusion* Γ*, then* $\mathcal{D}^\circ \in \mathsf{rPLL}^\infty$ *(resp.* $\mathcal{D}^\bullet \in \mathsf{wrPLL}^\infty$*) with conclusion* Γ*, and every* c!p *in* $\mathcal{D}^\circ$ *(resp.* $\mathcal{D}^\bullet$*) belongs to a* nwb*.*
2. *If* $\mathcal{D}' \in \mathsf{rPLL}^\infty$ *(resp.* $\mathcal{D}' \in \mathsf{wrPLL}^\infty$*) and every* c!p *in* $\mathcal{D}'$ *belongs to a* nwb*, then there is* $\mathcal{D} \in \mathsf{PLL}$ *(resp.* $\mathcal{D} \in \mathsf{nuPLL}$*) such that* $\mathcal{D}^\circ = \mathcal{D}'$ *(resp.* $\mathcal{D}^\bullet = \mathcal{D}'$*).*

Progressing and weak progressing coincide in finitely expandable coderivations.

▶ **Lemma 20.** *Let* $\mathcal{D} \in \mathsf{PLL}^\infty$ *be finitely expandable. If* $\mathcal{D} \in \mathsf{wpPLL}^\infty$ *then any infinite branch contains the main branch of a* nwb*. Moreover,* $\mathcal{D} \in \mathsf{pPLL}^\infty$ *if and only if* $\mathcal{D} \in \mathsf{wpPLL}^\infty$*.*

Proof. Let $\mathcal{D} \in \mathsf{wpPLL}^\infty$ be finitely expandable, and let $\mathcal{B}$ be an infinite branch in $\mathcal{D}$. By finite expandability there is $h \in \mathbb{N}$ such that $\mathcal{B}$ contains no conclusion of a cut or ?b with height greater than h. Moreover, by weakly progressing there is an infinite sequence $h \leq h_0 < h_1 < \ldots < h_n < \ldots$ such that the sequent of $\mathcal{B}$ at height h_i has shape $?\Gamma_i, !A_i$. By inspecting the rules in Figure 1, each such $?\Gamma_i, !A_i$ can be the conclusion of either a ?w or a c!p (with right premise $?\Gamma_i, !A_i$). So, there is a k large enough such that, for any $i \geq k$, only the latter case applies (and, in particular, $\Gamma_i = \Gamma$ and $A_i = A$ for some Γ, A). Therefore, h_k is the root of a nwb. This also shows $\mathcal{D} \in \mathsf{pPLL}^\infty$. By Remark 12, $\mathsf{pPLL}^\infty \subseteq \mathsf{wpPLL}^\infty$. ◀

By inspecting the steps in Figures 3, 5, and 10, we prove the following preservations.

▶ **Proposition 21.** *Cut elimination preserves weak-regularity, regularity and finite expandability. Therefore, if* $\mathcal{D} \in \mathsf{X}$ *with* $\mathsf{X} \in \{\mathsf{rPLL}^\infty, \mathsf{wrPLL}^\infty\}$ *and* $\mathcal{D} \to_{\mathsf{cut}} \mathcal{D}'$*, then also* $\mathcal{D}' \in \mathsf{X}$*.*

5 Continuous cut-elimination

Cut-elimination for (finitary) sequent calculi proceeds by introducing a proof rewriting strategy that stepwise decreases an appropriate termination ordering (see, e.g, [37]). Typically, these proof rewriting strategies consist on pushing upward the topmost cuts via the cut-elimination steps in order to eventually eliminate them.

A somewhat dual approach is investigated in the context of non-wellfounded proofs [6, 20]. It consists on *infinitary* proof rewriting strategies that gradually push upward the bottommost cuts. In this setting, the progressing condition is essential to guarantee *productivity*, i.e., that such proof rewriting strategies construct strictly increasing approximations of the cut-free proof, which can thus be obtained as a (well-defined) *limit*.

A major obstacle of this approach arises when the bottommost cut r is below another one r′. In this case, no cut-elimination step can be applied to r, so proof rewriting runs into an apparent stumbling block. To circumvent this problem, in [6, 20] a special cut-elimination step is introduced, which merges r and r′ in a single, generalized cut rule called *multicut*.

In this section we study a continuous cut-elimination method that does not rely on multicut rules, following an alternative idea in which the notion of approximation plays an even more central rule, inspired by the topological approaches to infinite trees [9]. To this end, we assume the reader familiar with basic definitions on domain-theory (see, e.g., [4]).

5.1 Approximating coderivations

We introduce *open coderivations* to approximate coderivations. They form Scott-domains, on top of which we define *continuous cut elimination*. We also exploit them to *decompose* a finitely expandable and progressing coderivation into a *finite* approximation beneath nwbs.

▶ **Definition 22.** *We define the set of rules* $\mathsf{oPLL}^\infty := \mathsf{PLL}^\infty \cup \{\mathsf{hyp}\}$*, where* $\mathsf{hyp} := \dfrac{}{\Gamma}\,\mathsf{hyp}$ *for any sequent* Γ.[4] *We will also refer to* oPLL^∞ *as the set of coderivations over* oPLL^∞*, which we call* ***open coderivations****. An open coderivation is* ***normal*** *if no cut-elimination step can be applied to it, that is, if one premise of each* cut *is a* hyp*. An* ***open derivation*** *is a derivation in* oPLL^∞*. We denote by* $\mathsf{oPLL}^\infty(\Gamma)$ *the set of open coderivations with conclusion* Γ*.*

▶ **Definition 23.** *Let* $\mathcal{D}$ *be an open coderivation,* $\mathcal{V} \subseteq \{1,2\}^*$ *be a set of mutually incomparable (w.r.t. the prefix order) nodes of* $\mathcal{D}$*, and* $\{\mathcal{D}'_\nu\}_{\nu\in\mathcal{V}}$ *be a set of open coderivations where* $\mathcal{D}'_\nu$ *has the same conclusion as the subderivation* $\mathcal{D}_\nu$ *of* $\mathcal{D}$*. We denote by* $\mathcal{D}\{\mathcal{D}'_\nu/\nu\}_{\nu\in\mathcal{V}} = \mathcal{D}(\mathcal{D}'_{\nu_1}/\nu_1, \ldots, \mathcal{D}'_{\nu_n}/\nu_n)$*, the open coderivation obtained by replacing each* $\mathcal{D}_\nu$ *with* $\mathcal{D}'_\nu$*.*

The ***pruning*** *of* $\mathcal{D}$ *over* $\mathcal{V}$ *is the open coderivation* $\lfloor\mathcal{D}\rfloor_\mathcal{V} = \mathcal{D}\{\mathsf{hyp}/\nu\}_{\nu\in\mathcal{V}}$*. If* $\mathcal{D}$ *and* $\mathcal{D}'$ *are two open coderivations, then we say that* $\mathcal{D}$ *is an* ***approximation*** *of* $\mathcal{D}'$ *(noted* $\mathcal{D} \preceq \mathcal{D}'$*) iff* $\mathcal{D} = \lfloor\mathcal{D}'\rfloor_\mathcal{V}$ *for some* $\mathcal{V} \subseteq \{1,2\}^*$*. An approximation is* ***finite*** *if it is an open derivation.*

We denote by $\mathcal{K}(\mathcal{D})$ *the set of finite approximations of* $\mathcal{D}$*.*

Note that $\mathcal{D}$ and $\lfloor\mathcal{D}\rfloor_\mathcal{V}$ (and hence $\mathcal{D}'$ if $\mathcal{D} \preceq \mathcal{D}'$) have the same conclusion. Any open coderivation $\mathcal{D}$ is the supremum of its finite approximations, i.e. $\mathcal{D} = \bigsqcup_{\mathcal{D}'\in\mathcal{K}(\mathcal{D})} \mathcal{D}'$. Indeed:

▶ **Proposition 24.** *For any sequent* Γ*, the poset* $(\mathsf{oPLL}^\infty(\Gamma), \preceq)$ *is a Scott-domain with least element the open derivation* hyp *and with maximal elements the coderivations (in* PLL^∞*) with conclusion* Γ*. The compact elements are precisely the open derivations in* $\mathsf{oPLL}^\infty(\Gamma)$*.*

Cut-elimination steps essentially do not increase the size of open derivations, hence:

▶ **Lemma 25.** $\to_{\mathsf{cut}}$ *over open derivations is strongly normalizing and confluent.*

Progressing and finitely expandable coderivations can be approximated in a canonical way. Indeed, by Lemma 20 we have:

▶ **Proposition 26.** *If* $\mathcal{D} \in \mathsf{pPLL}^\infty$ *is finitely expandable, then there is a prebar* $\mathcal{V} \subseteq \{1,2\}^*$ *of* $\mathcal{D}$ *such that each* $v \in \mathcal{V}$ *is the root of a* nwb *in* $\mathcal{D}$*.*

▶ **Definition 27.** *Let* $\mathcal{D} \in \mathsf{pPLL}^\infty$ *be finitely expandable. The* ***decomposition prebar*** *of* $\mathcal{D}$ *is the minimal prebar* $\mathcal{V}$ *of* $\mathcal{D}$ *such that, for all* $\nu \in \mathcal{V}$*,* $\mathcal{D}_\nu$ *is a* nwb*. We denote with* $\mathsf{border}(\mathcal{D})$ *such a bar and we set* $\mathsf{base}(\mathcal{D}) := \lfloor\mathcal{D}\rfloor_{\mathsf{border}(\mathcal{D})}$*.*

Note that, by weak König lemma, in the above definition $\mathsf{border}(\mathcal{D})$ is finite and $\mathsf{base}(\mathcal{D})$ is a finite approximation of $\mathcal{D}$.

[4] Previously introduced notions and definitions on coderivations extend to open coderivations in the obvious way, e.g. the global conditions of Definitions 11 and 16 and the cut-elimination relation $\to_{\mathsf{cut}}$.

5.2 Domain-theoretic approach to continuous cut-elimination

In this subsection we define *maximal and continuous infinitary cut-elimination strategies* (mc-ices), special rewriting strategies that stepwise generate ω-chains approximating the cut-free version of an open coderivation. In other words, a mc-ices computes a (Scott-)continuous function from open coderivations to cut-free open coderivations. Then, we introduce the *height-by-height* mc-ices, a notable example of mc-ices that will be used for our results, and we show that any two mc-icess compute the same (Scott-)continuous function.

In what follows, σ denotes a countable sequence of coderivations, and $\sigma(i)$ denotes the $(i+1)$-th coderivation in σ. We denote the length of a sequence σ by $\ell(\sigma) \leq \omega$.

▶ **Definition 28.** *An **infinitary cut elimination strategy** (or* ices *for short) is a family $\sigma = \{\sigma_{\mathcal{D}}\}_{\mathcal{D}\in \mathsf{oPLL}^\infty}$ where, for all $\mathcal{D} \in \mathsf{oPLL}^\infty$, $\sigma_{\mathcal{D}}$ is a sequence of open coderivations such that $\sigma_{\mathcal{D}}(0) = \mathcal{D}$ and $\sigma_{\mathcal{D}}(i) \to_{\mathsf{cut}} \sigma_{\mathcal{D}}(i+1)$ for all $0 \leq i < \ell(\sigma_{\mathcal{D}})$. Given an* ices *$\sigma$, we define the function $f_\sigma \colon \mathsf{oPLL}^\infty(\Gamma) \to \mathsf{oPLL}^\infty(\Gamma)$ as $f_\sigma(\mathcal{D}) \coloneqq \bigsqcup_{i=0}^{\ell(\sigma_{\mathcal{D}})} \mathsf{cf}(\sigma_{\mathcal{D}}(i))$ where $\mathsf{cf}(\mathcal{D}_i)$ is the greatest cut-free approximation of $\mathcal{D}_i$ (w.r.t. $\preceq$).* [5]*. An* ices *σ is a* mc-ices *if it is:*

- ***maximal**: $\sigma_{\mathcal{D}}(\ell(\sigma_{\mathcal{D}}))$ is normal for any open derivation $\mathcal{D}$ ($\ell(\sigma_{\mathcal{D}}) < \omega$ by Lemma 25);*
- ***(Scott)-continuous**: f_σ is Scott-continuous.*

Roughly, a maximal ices is an ices that applies cut-elimination steps to open derivations (i.e., finite approximations) until a normal (possibly cut-free) open derivation is reached. The following property states that all mc-icess induce the same continuous function, an easy consequence of Lemma 25 and continuity.

▶ **Proposition 29.** *If σ and σ' are two* mc-icess*, then $f_\sigma = f_{\sigma'}$.*

Therefore, we define a specific mc-ices we use in our proofs, where cut-elimination steps are applied in a deterministic way to the minimal reducible cut-rules.

▶ **Definition 30.** *The **height-by-height** ices is defined as $\sigma^\infty = \{\sigma^\infty_{\mathcal{D}}\}_{\mathcal{D}\in\mathsf{oPLL}^\infty}$ where $\sigma^\infty_{\mathcal{D}}(0) = \mathcal{D}$ for each $\mathcal{D} \in \mathsf{oPLL}^\infty$, and $\sigma^\infty_{\mathcal{D}}(i+1)$ is the open coderivation obtained by applying a cut-elimination step to the rightmost reducible* cut*-rule with minimal height in $\sigma^\infty_{\mathcal{D}}(i)$.*

▶ **Proposition 31.** *The* ices *σ^∞ is a* mc-ices*.*

Proof. By Lemma 25, any open derivation $\mathcal{D}$ normalizes in $n_{\mathcal{D}} \in \mathbb{N}$ steps; so, if $\mathcal{D}$ is an open derivation, $\ell(\sigma^\infty_{\mathcal{D}}) = n_{\mathcal{D}}$ with $\sigma^\infty_{\mathcal{D}}(n_{\mathcal{D}})$ normal by definition of σ^∞. Hence, σ^∞ is maximal.

Since $\sigma^\infty_{\mathcal{D}}(i)$ is defined by applying a finite number of cut-eliminations steps to $\mathcal{D}$, then there is $\mathcal{D}' \in \mathcal{K}(\mathcal{D})$ such that $\sigma^\infty_{\mathcal{D}}(i) = \sigma^\infty_{\mathcal{D}'}(i)$, and therefore $\mathsf{cf}(\sigma^\infty_{\mathcal{D}}(i)) = \mathsf{cf}(\sigma^\infty_{\mathcal{D}'}(i)) \preceq f_{\sigma^\infty}(\mathcal{D}')$ for all $0 \leq i \leq \ell(\sigma^\infty)$. Thus $f_{\sigma^\infty}(\mathcal{D}) \preceq \bigsqcup_{\mathcal{D}'\in\mathcal{K}(\mathcal{D})} f_{\sigma^\infty}(\mathcal{D}')$. Moreover $\bigsqcup_{\mathcal{D}'\in\mathcal{K}(\mathcal{D})} f_{\sigma^\infty}(\mathcal{D}') \preceq f_{\sigma^\infty}(\mathcal{D})$ because σ^∞ is monotone by construction. Therefore, f_{σ^∞} is continuous. ◀

In order to prove our results, we introduce the notion of chain of cut-rules, which allows us to keep track of the dynamic of cut-elimination steps during infinitary rewriting. Note that the definition of cut-chain is the analogue of the *multi-cut reduction sequences* from [6].

▶ **Definition 32** (Chains). *Let $\sigma = \{\sigma_{\mathcal{D}}\}_{\mathcal{D}\in\mathsf{oPLL}^\infty}$ be an* ices*. We write $\mathsf{r}_i \mapsto_\sigma \mathsf{r}_{i+1}$ if r_{i+1} is a* cut*-rule in $\sigma_{\mathcal{D}}(i+1)$ produced by applying a cut-elimination step to the* cut*-rule r_i in $\sigma_{\mathcal{D}}(i)$.*

A cut***-chain** in $\sigma_{\mathcal{D}}$ is a sequence $(\mathsf{r}_i)_{i<\alpha}$ of* cut *rules with $\alpha \leq \ell(\sigma_{\mathcal{D}})$, such that r_i a rule in $\sigma_{\mathcal{D}}(i)$, and either $\mathsf{r}_i = \mathsf{r}_{i+1}$ or $\mathsf{r}_i \mapsto_\sigma \mathsf{r}_{i+1}$. We say that a chain **starts** at r_0 and that each r_{i+1} is a **descendant** of r_i.*

[5] f_σ is well-defined, as $(\mathsf{cf}(\sigma_{\mathcal{D}}(i)))_{0\leq i<\ell(\sigma_{\mathcal{D}})}$ is an ω-chain in oPLL^∞ and so its sup exists by Proposition 24.

We conclude this section by providing the sketch of proof for the continuous cut-elimination theorem, the main contribution of this paper, establishing a productivity result and showing that continuous cut-elimination preserves all global conditions.

▶ **Theorem 33** (Continuous Cut-Elimination)**.**
1. *If* $\mathcal{D} \in \mathsf{pPLL}^\infty$*, then so is* $f_{\sigma^\infty}(\mathcal{D})$*.*
2. *If* $\mathcal{D} \in \mathsf{wrPLL}^\infty$ *(resp.* $\mathcal{D} \in \mathsf{rPLL}^\infty$*), then so is* $f_{\sigma^\infty}(\mathcal{D})$*.*

Sketch of the proof.
1. We have to prove that $f_{\sigma^\infty}(\mathcal{D})$ is hyp-free (i.e., *productivity*) and that any of its infinite branches contains a progressing !-thread. To facilitate our argument, leveraging on symmetry of the cut rules, we assume w.l.o.g. that !-formulas can only be cut in the left-hand premise of a cut-rule.
   We first show that, for any infinite cut-chain $(\mathsf{r}_i)_i$ there is a descendant r_i in $\sigma^\infty_{\mathcal{D}}(i)$ whose right premise is the conclusion of a c!p-rule. Since $\mathcal{B}$ has infinitely many c!p rules by progressing condition, every cut-rule with a premise in $\mathcal{B}$ is eventually reducible, so that there are infinitely many $i \geq i_0$ such that $\mathsf{r}_i \mapsto_\sigma \mathsf{r}_{i+1}$. Therefore, if the right-premise of r_i did not eventually become conclusion of a c!p-rule we could identify an infinite branch of $\mathcal{D}$ that has no progressing !-thread.
   Now, let $\mathcal{B}^*$ be a branch of $f_{\sigma^\infty}(\mathcal{D})$. If $\mathcal{B}^*$ has been obtained from $\mathcal{D}$ after finitely many cut-elimination steps then it is clearly hyp-free and, if infinite, it has a progressing !-thread (Proposition 21). Otherwise, $\mathcal{B}^*$ has been constructed by an infinite cut-chain $(\mathsf{r}_i)_i$ with minimal height. By repeatedly applying the above property, we have that there are infinitely many r_i whose rightmost premise is the conclusion of a c!p-rule r^*, and such that $\mathsf{r}_i \mapsto_\sigma \mathsf{r}_{i+1}$ is a step permuting r^* downward (since r^* it is on the left premise of r_i, its principal !-formula cannot be a cut-formula of r_i by assumption). This means that $\mathcal{B}^*$ contains infinitely many c!p rules, and so it is hyp-free. To prove that there is a progressing !-thread in $\mathcal{B}^*$ it suffices to show that infinitely many c!p rules of $\mathcal{B}^*$ are descendants of the same branch $\mathcal{B}$ of $\mathcal{D}$, as the existence of a progressing !-thread of $\mathcal{B}^*$ would follow directly from the existence of a (unique) progressing !-thread of $\mathcal{B}$.
2. Akin to linear logic, we define the *depth* of a coderivation as the maximal number of nested nwb*s*, and we prove that the depth of (weakly) regular coderivations is always finite. Moreover, by Proposition 26, a progressing and finitely expandable coderivation $\mathcal{D}$ can be decomposed to a nwb-free finite approximation $\mathsf{base}(\mathcal{D})$ and a series of nwb*s* whose calls have smaller depth. Using this property we define, by induction on the depth of $\mathcal{D}$, a maximal and *transfinite* ices reducing the calls of the nwb*s* one by one. The proof of preservation of (weak) regularity under cut-elimination for such an ices follows by construction since, by Remark 17, if we reduce a nwb with finite support (resp. a periodic nwb) via our transfinite ices, then we obtain in the limit a cut-free nwb with finite support (resp. a periodic nwb). We then show that this transfinite ices can be compressed to a (ω-long) mc-ices using methods studied in [36, 33], and we conclude the proof by Item 1 and by the fact that $f_{\sigma^\infty}(\mathcal{D})$ is finitely expandable and (weakly) regular for such a mc-ices. ◀

By definition (as the sup of cut-free open coderivations) $f_{\sigma^\infty}(\mathcal{D})$ is cut-free. Each item of Theorem 33 says in particular that $f_{\sigma^\infty}(\mathcal{D})$ is hyp-free, which means that $f_{\sigma^\infty}(\mathcal{D})$ is obtained by eliminating *all* the cuts in $\mathcal{D}$. This may not be the case if $\mathcal{D}$ does not fulfill any of the global conditions in the hypotheses of Theorem 33: $f_{\sigma^\infty}(\mathcal{D})$ is still cut-free but may contain some "truncating" hyp that "prevented" eliminating some cut in $\mathcal{D}$, as in the example below.

$$\left[\!\!\left[\mathsf{ax}\,\frac{}{A, A^\perp} \right]\!\!\right]_n = \{\, (x,x) \mid x \in [\![A]\!] \,\} \qquad \left[\!\!\left[\mathsf{cut}\,\frac{\overset{\mathcal{D}'}{\Gamma, A} \quad \overset{\mathcal{D}''}{\Delta, A^\perp}}{\Gamma, \Delta} \right]\!\!\right]_n = \left\{ (\vec{x}, \vec{y}) \;\middle|\; \exists z \in [\![A]\!] \text{ s.t. } \begin{array}{c} (\vec{x}, z) \in [\![\mathcal{D}']\!]_{n-1} \\ \text{and} \\ (z, \vec{y}) \in [\![\mathcal{D}'']\!]_{n-1} \end{array} \right\}$$

$$\left[\!\!\left[\perp\,\frac{\overset{\mathcal{D}'}{\Gamma}}{\Gamma, \perp} \right]\!\!\right]_n = \{\, (\vec{x}, *) \mid \vec{x} \in [\![\mathcal{D}']\!]_{n-1} \,\} \qquad \left[\!\!\left[\parr\,\frac{\overset{\mathcal{D}'}{\Gamma, A, B}}{\Gamma, A \parr B} \right]\!\!\right]_n = \{\, (\vec{x}, (y,z)) \mid (\vec{x}, y, z) \in [\![\mathcal{D}']\!]_{n-1} \,\}$$

$$\left[\!\!\left[1\,\frac{}{1} \right]\!\!\right]_n = \{*\} \qquad \left[\!\!\left[\otimes\,\frac{\overset{\mathcal{D}'}{\Gamma, A} \quad \overset{\mathcal{D}''}{\Delta, B}}{\Gamma, \Delta, A \otimes B} \right]\!\!\right]_n = \left\{ (\vec{x}, \vec{y}, (x,y)) \;\middle|\; \begin{array}{c} (\vec{x}, x) \in [\![\mathcal{D}']\!]_{n-1} \\ \text{and} \\ (\vec{y}, y) \in [\![\mathcal{D}'']\!]_{n-1} \end{array} \right\} \qquad \left[\!\!\left[\mathsf{hyp}\,\frac{}{\Gamma} \right]\!\!\right]_n = \varnothing$$

$$\left[\!\!\left[?\mathsf{w}\,\frac{\overset{\mathcal{D}'}{\Gamma}}{\Gamma, ?A} \right]\!\!\right]_n = \{\, (\vec{x}, [\,]) \mid \vec{x} \in [\![\mathcal{D}']\!]_{n-1} \,\} \qquad \left[\!\!\left[?\mathsf{b}\,\frac{\overset{\mathcal{D}'}{\Gamma, A, ?A}}{\Gamma, ?A} \right]\!\!\right]_n = \{\, (\vec{x}, [y] + \mu) \mid (\vec{x}, y, \mu) \in [\![\mathcal{D}']\!]_{n-1} \,\}$$

$$\left[\!\!\left[\mathsf{c!p}\,\frac{\overset{\mathcal{D}'}{\Gamma, A} \quad \overset{\mathcal{D}''}{?\Gamma, !A}}{?\Gamma, !A} \right]\!\!\right]_n = \left\{ (\vec{[\,]}, [\,]) \right\} \cup \left\{ ([x_1] + \mu_1, \dots, [x_k] + \mu_k, [x] + \mu) \;\middle|\; \begin{array}{c} (x_1, \dots, x_k, x) \in [\![\mathcal{D}']\!]_{n-1} \\ \text{and} \\ (\mu_1, \dots, \mu_k, \mu) \in [\![\mathcal{D}'']\!]_{n-1} \end{array} \right\}$$

Figure 12 Inductive definition of the set $[\![\mathcal{D}]\!]_n$, for $n > 0$.

▶ **Example 34.** For any finite approximation $\mathcal{D}$ of the (non-weakly progressing, non-finitely expandable) open coderivation $\mathcal{D}_\natural$, we have $f_{\sigma^\infty}(\mathcal{D}) = \mathsf{hyp}$, so $f_{\sigma^\infty}(\mathcal{D}_\natural) = \mathsf{hyp}$ by continuity.

6 Relational semantics for non-wellfounded proofs

Here we define a denotational model for oPLL^∞ based on *relational semantics*, which interprets an open coderivation as the union of the interpretations of its finite approximations, as in [17]. We show that relational semantics is sound for oPLL^∞, but not for its extension with digging.

Relational semantics interprets exponential by finite multisets, denoted by brackets, e.g., $[x_1, \dots, x_n]$; $+$ denotes the *multiset union*, and $\mathcal{M}_f(X)$ denotes the set of finite multisets over a set X. To correctly define the semantics of a coderivation, we need to see sequents as *finite sequences* of formulas (taking their order into account), which means that we have to add an *exchange* rule to oPLL^∞ to swap the order of two consecutive formulas in a sequent.

▶ **Definition 35.** *We associate with each formula A a* ***set*** $[\![A]\!]$ *defined as follows:*

$$[\![X]\!] \coloneqq D_X \quad [\![1]\!] \coloneqq \{*\} \quad [\![A \otimes B]\!] \coloneqq [\![A]\!] \times [\![B]\!] \quad [\![!A]\!] \coloneqq \mathcal{M}_f([\![A]\!]) \quad [\![A^\perp]\!] \coloneqq [\![A]\!]$$

where D_X is an arbitrary set. For a sequent $\Gamma = A_1, \dots, A_n$, we set $[\![\Gamma]\!] \coloneqq [\![A_1 \parr \cdots \parr A_n]\!]$.

Given $\mathcal{D} \in \mathsf{PLL} \cup \mathsf{oPLL}^\infty$ with conclusion Γ, we set $[\![\mathcal{D}]\!] \coloneqq \bigcup_{n \geq 0} [\![\mathcal{D}]\!]_n \subseteq [\![\Gamma]\!]$, where $[\![\mathcal{D}]\!]_0 = \varnothing$ and, for all $i \in \mathbb{N} \setminus \{0\}$, $[\![\mathcal{D}]\!]_i$ is defined inductively according to Figure 12.

▶ **Example 36.** For the coderivations $\mathcal{D}_\natural$ and $\mathcal{D}_?$ in Figure 7, $[\![\mathcal{D}_\natural]\!] = [\![\mathcal{D}_?]\!] = \varnothing$. For the derivations $\underline{0}$ and $\underline{1}$ in Figure 2, $[\![\underline{0}]\!] = \{([\,], (x,x)) \mid x \in D_X\}$ and $[\![\underline{1}]\!] = \{([(x,y)], (x,y)) \mid x, y \in D_X\}$. For the coderivation $\mathsf{c!p}_{(\underline{i_0}, \dots, \underline{i_n}, \dots)}$ in Example 10 (with $i_j \in \{0,1\}$ for all $j \in \mathbb{N}$), $[\![\mathsf{c!p}_{(\underline{i_0}, \dots, \underline{i_n}, \dots)}]\!] = \{[\,]\} \cup \left\{ [x_{i_0}, \dots, x_{i_n}] \in \mathcal{M}_f([\![\mathbf{N}]\!]) \mid n \in \mathbb{N},\ x_{i_j} \in [\![\underline{i_j}]\!]\ \forall\, 0 \leq j \leq n \right\}$. For the derivation $\underline{n}$ in Example 10 (for any $n \in \mathbb{N}$), $[\![\underline{n}]\!] = \{([(x_1, x_2), \dots, (x_n, x_{n+1})], (x_1, x_{n+1})) \mid x_1, \dots, x_{n+1} \in D_X\}$. Note that $[\![\underline{n}]\!] \cap [\![\underline{m}]\!] = \varnothing$ for all $n, m \in \mathbb{N}$ such that $n \neq m$, and that $[\![n]\!]$ is stable under permutations of the rules ?w, ?b and $\otimes$ in $\underline{n}$ (that is, if $\mathcal{D}$ is obtained from $\underline{n}$ by permuting the rules ?w, ?b or $\otimes$, then $[\![\mathcal{D}]\!] = [\![\underline{n}]\!]$).

$$\mathsf{??d}\,\frac{\Gamma, ??A}{\Gamma, ?A} \qquad \left[\!\!\left[\mathsf{??d}\,\frac{\overset{\mathcal{D'}}{\Gamma, ??A}}{\Gamma, ?A} \right]\!\!\right]_0 = \varnothing \qquad \left[\!\!\left[\mathsf{??d}\,\frac{\overset{\mathcal{D'}}{\Gamma, ??A}}{\Gamma, ?A} \right]\!\!\right]_n = \left\{ \left(\vec{x}, \sum_{i=1}^{m} \mu_i \right) \;\middle|\; (\vec{x}, [\mu_1, \dots, \mu_m]) \in [\![\mathcal{D'}]\!]_{n-1},\ m \in \mathbb{N} \right\}$$

Figure 13 The rule ??d and its interpretation in the relational semantics ($n > 0$).

By inspecting the cut-elimination steps and by continuity, we can prove the soundness of relational semantics with respect to cut-elimination (Theorem 38), thanks to the fact the interpretation of a coderivation is the union the interpretations of its finite approximation.

▶ **Lemma 37.** *Let $\mathcal{D} \in \mathsf{oPLL}^\infty$. Then, $[\![\mathcal{D}]\!] = [\![\bigsqcup_{\mathcal{D'} \in \mathcal{K}(\mathcal{D})} \mathcal{D'}]\!] = \bigcup_{\mathcal{D'} \in \mathcal{K}(\mathcal{D})} [\![\mathcal{D'}]\!]$.*

▶ **Theorem 38** (Soundness)**.**
1. *Let $\mathcal{D} \in \mathsf{oPLL}^\infty$. If $\mathcal{D} \to_{\mathsf{cut}} \mathcal{D'}$, then $[\![\mathcal{D}]\!] = [\![\mathcal{D'}]\!]$.*
2. *Let $\mathcal{D} \in \mathsf{oPLL}^\infty$. If σ is a* mc-ices*, then $[\![\mathcal{D}]\!] = [\![f_\sigma(\mathcal{D})]\!]$.*

By Theorem 38 and since cut-free coderivations have non-empty semantics, we have:

▶ **Corollary 39.** *Let $\mathcal{D} \in \mathsf{wpPLL}^\infty$. Then $[\![\mathcal{D}]\!] \neq \varnothing$.*

We define the set of rules $\mathsf{MELL}^\infty := \mathsf{PLL}^\infty \cup \{\mathsf{??d}\}$ where the rule ??d (**digging**) is defined in Figure 13. We also denote by MELL^∞ the set of coderivations over the rules in MELL^∞. Relational semantics is naturally extended to MELL^∞ as shown in Figure 13.

The proof system MELL^∞ can be seen as a non-wellfounded version of MELL. We show that, as opposed to several fragments of PLL^∞, in any good fragment of MELL^∞ with digging, cut-elimination cannot reduce to cut-free coderivations *and* preserve both the progressing condition and relational semantics.

▶ **Theorem 40.** *Let $\mathsf{X} \subseteq \mathsf{MELL}^\infty$ contain non-wellfounded coderivations with ??d. Let $\to_{\mathsf{cut}+}$ be a cut-elimination relation on X preserving the progressing condition, containing $\to_{\mathsf{cut}}$ in Figures 3, 5, and 10 and reducing every coderivation in X to a cut-free one. Then, $\to_{\mathsf{cut}+}$ does not preserve relational semantics.*

Proof. Consider the coderivations $\mathcal{D}_{??\mathsf{d}}$ and $\widehat{\mathcal{D}_{??\mathsf{d}}}$ below, where $\mathcal{D} = \mathsf{c!p}_{(\underline{0},\underline{1},\underline{0},\underline{1},\dots)}$ and, for all $i \in \mathbb{N}$, $\mathcal{D}_i \in \{\mathsf{c!p}_{(\underline{k^i_0},\dots,\underline{k^i_n},\dots)} \mid k^i_j \in \mathbb{N} \text{ for all } j \in \mathbb{N}\}$ ($\underline{n}$ is defined in Example 10 for all $n \in \mathbb{N}$).

$$\mathcal{D}_{??\mathsf{d}} := \cfrac{\cfrac{\overset{\mathcal{D}}{\mathbf{!N}} \qquad \cfrac{}{??\mathbf{N}^\perp, \mathbf{!!N}}\,\mathsf{ax}}{?\mathbf{N}^\perp, \mathbf{!!N}}\,??\mathsf{d}}{\mathbf{!!N}}\,\mathsf{cut} \qquad \widehat{\mathcal{D}_{??\mathsf{d}}} := \cfrac{\cfrac{\overset{\mathcal{D}_0}{\mathbf{!N}} \qquad \cfrac{\overset{\mathcal{D}_1}{\mathbf{!N}} \qquad \cfrac{\overset{\mathcal{D}_n}{\mathbf{!N}} \qquad \cfrac{\vdots}{\mathbf{!!N}}\,\mathsf{c!p}}{\mathbf{!!N}}\,\mathsf{c!p} \qquad \iddots}{\mathbf{!!N}}\,\mathsf{c!p}}{\mathbf{!!N}}\,\mathsf{c!p}$$

Coderivations $\widehat{\mathcal{D}_{??\mathsf{d}}}$ are the only cut-free and progressing ones with conclusion $\mathbf{!!N}$. Indeed, any cut-free coderivation of $\mathbf{!!N}$ or $\mathbf{!N}$ must end with a c!p, and the only cut-free and progressing coderivations of $\mathbf{N}$ are the derivations of the form $\underline{n}$ for any $n \in \mathbb{N}$, up to permutations of the rules ?w, ?b and $\otimes$ (other cut-free coderivations of $\mathbf{N}$ exist, but they have an infinite branch containing infinitely many ?b rules and no c!p rules, hence they are not progressing). Therefore, for whatever definition of the cut-elimination steps concerning ??d that preserves the progressing condition, necessarily $\mathcal{D}_{??\mathsf{d}}$ will reduce to $\widehat{\mathcal{D}_{??\mathsf{d}}}$, since $\mathcal{D}_{??\mathsf{d}}$ is progressing.

We show that $[\![\widehat{\mathcal{D}_{??\mathsf{d}}}]\!] \not\subseteq [\![\mathcal{D}_{??\mathsf{d}}]\!]$. First, it can be easily shown that if, in one of the $\mathcal{D}_i = \mathsf{c!p}_{(\underline{k^i_0},\dots,\underline{k^i_n},\dots)}$ in $\widehat{\mathcal{D}_{??\mathsf{d}}}$, one of the k^i_j is different from 0 or 1, then there is $x \in [\![\widehat{\mathcal{D}_{??\mathsf{d}}}]\!] \setminus [\![\mathcal{D}_{??\mathsf{d}}]\!]$ (this basically follows from the fact that $[\![\underline{n}]\!] \cap [\![\underline{m}]\!] = \varnothing$ for all $n, m \in \mathbb{N}$ such that $n \neq m$,

see Example 36). Let us now suppose that in $\widehat{\mathcal{D}_{??\mathsf{d}}}$, for all $i \in \mathbb{N}$, $\mathcal{D}_i = \mathsf{c!p}_{(\underline{k_0^i},\dots,\underline{k_n^i},\dots)}$ with $k_j^i \in \{0,1\}$ for all $j \in \mathbb{N}$. Let $\hat{0}$ and $\hat{1}$ be any element of $[\![\underline{0}]\!]$ and $[\![\underline{1}]\!]$, respectively (see Example 36). Note that $\hat{0} \neq \hat{1}$. It is easy to verify that $[[\hat{0}],[\hat{0}]],[[\hat{1}],[\hat{1}]] \notin [\![\mathcal{D}_{??\mathsf{d}}]\!]$, since $[\hat{0},\hat{0}],[\hat{1},\hat{1}] \notin [\![\mathcal{D}]\!]$ (see Example 36). Concerning $[\![\widehat{\mathcal{D}_{??\mathsf{d}}}]\!]$, notice that, since $k_0^0,k_0^1,k_0^2 \in \{0,1\}$, either $k_0^0 = k_0^1$ or $k_0^1 = k_0^2$ or $k_0^2 = k_0^0$. In the first case, we have $[[k_0^0],[k_0^1]] \in [\![\widehat{\mathcal{D}_{??\mathsf{d}}}]\!]$, in the second case we have $[[k_0^1],[k_0^2]] \in [\![\widehat{\mathcal{D}_{??\mathsf{d}}}]\!]$, and in the last case we have $[[k_0^2],[k_0^0]] \in [\![\widehat{\mathcal{D}_{??\mathsf{d}}}]\!]$. ◀

7 Conclusion and future work

For future research, we envisage extending our contributions in many directions. First, our notion of finite approximation seems intimately related with that of Taylor expansion from *differential linear logic* (DiLL) [18, 19, 15], where the rule hyp (quite like the rule 0 from DiLL, [3]) serves to model approximations of *boxes*. This connection with Taylor expansions becomes even more apparent in Mazza's original systems for parsimonious logic [26, 27], which comprise co-absorption and co-weakening rules typical of DiLL. These considerations deserve further investigations. Secondly, building on a series of recent works in *Cyclic Implicit Complexity*, i.e., implicit computational complexity in the setting of circular and non-wellfounded proof theory [11, 10], we are currently working on second-order extensions of wrPLL^∞ and rPLL^∞ to characterize the complexity classes $\mathbf{P}/\mathsf{poly}$ and $\mathbf{P}$ (see [1]). These results would reformulate in a non-wellfounded setting the characterization of $\mathbf{P}/\mathsf{poly}$ presented in [27].

References

1 Matteo Acclavio, Gianluca Curzi, and Giulio Guerrieri. Non-uniform polynomial time via non-wellfounded parsimonious proofs. Unpublished. URL: http://gianlucacurzi.com/Non-uniform-polynomial-time-via-non-wellfounded-parsimonious-proofs.pdf.

2 Matteo Acclavio, Gianluca Curzi, and Giulio Guerrieri. Infinitary cut-elimination via finite approximations (extended version). *CoRR*, abs/2308.07789, 2023. doi:10.48550/ARXIV.2308.07789.

3 Matteo Acclavio and Giulio Guerrieri. A deep inference system for differential linear logic. In *Proceedings Second Joint International Workshop on Linearity & Trends in Linear Logic and Applications, Linearity&TLLA@IJCAR-FSCD 2020*, volume 353 of *EPTCS*, pages 26–49, 2020. doi:10.4204/EPTCS.353.2.

4 Roberto M. Amadio and Pierre-Louis Curien. *Domains and lambda-calculi*, volume 46 of *Cambridge tracts in theoretical computer science*. Cambridge University Press, 1998.

5 Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009. doi:10.1017/CBO9780511804090.

6 David Baelde, Amina Doumane, and Alexis Saurin. Infinitary proof theory: the multiplicative additive case. In Jean-Marc Talbot and Laurent Regnier, editors, *25th EACSL Annual Conference on Computer Science Logic, CSL 2016, August 29 – September 1, 2016, Marseille, France*, volume 62 of *LIPIcs*, pages 42:1–42:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.CSL.2016.42.

7 David Baelde and Dale Miller. Least and greatest fixed points in linear logic. In Nachum Dershowitz and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning, 14th International Conference, LPAR 2007, Yerevan, Armenia, October 15-19, 2007, Proceedings*, volume 4790 of *Lecture Notes in Computer Science*, pages 92–106. Springer, 2007. doi:10.1007/978-3-540-75560-9_9.

8 James Brotherston and Alex Simpson. Sequent calculi for induction and infinite descent. *Journal of Logic and Computation*, 21(6):1177–1216, 2011.

9 Bruno Courcelle. Fundamental properties of infinite trees. *Theoretical Computer Science*, 25(2):95–169, 1983. doi:10.1016/0304-3975(83)90059-2.

10 Gianluca Curzi and Anupam Das. Cyclic implicit complexity. In *Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '22, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3531130.3533340.

11 Gianluca Curzi and Anupam Das. Non-uniform complexity via non-wellfounded proofs. In *31st EACSL Annual Conference on Computer Science Logic, CSL 2023*, volume 252 of *LIPIcs*, pages 16:1–16:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.CSL.2023.16.

12 Vincent Danos and Jean-Baptiste Joinet. Linear logic and elementary time. *Inf. Comput.*, 183(1):123–137, 2003. doi:10.1016/S0890-5401(03)00010-5.

13 Anupam Das. On the logical strength of confluence and normalisation for cyclic proofs. In *6th International Conference on Formal Structures for Computation and Deduction, FSCD 2021*, volume 195 of *LIPIcs*, pages 29:1–29:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.FSCD.2021.29.

14 Christian Dax, Martin Hofmann, and Martin Lange. A proof system for the linear time μ-calculus. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 273–284. Springer, 2006.

15 Thomas Ehrhard. An introduction to differential linear logic: proof-nets, models and antiderivatives. *Math. Struct. Comput. Sci.*, 28(7):995–1060, 2018. doi:10.1017/S0960129516000372.

16 Thomas Ehrhard and Farzad Jafar-Rahmani. On the denotational semantics of linear logic with least and greatest fixed points of formulas. *CoRR*, abs/1906.05593, 2019. arXiv:1906.05593.

17 Thomas Ehrhard, Farzad Jafar-Rahmani, and Alexis Saurin. On relation between totality semantic and syntactic validity. In *5th International Workshop on Trends in Linear Logic and Applications (TLLA 2021)*, June 2021. URL: https://hal-lirmm.ccsd.cnrs.fr/lirmm-03271408.

18 Thomas Ehrhard and Laurent Regnier. Differential interaction nets. *Theor. Comput. Sci.*, 364(2):166–195, 2006. doi:10.1016/J.TCS.2006.08.003.

19 Thomas Ehrhard and Laurent Regnier. Uniformity and the Taylor expansion of ordinary lambda-terms. *Theor. Comput. Sci.*, 403(2-3):347–372, 2008. doi:10.1016/J.TCS.2008.06.001.

20 Jérôme Fortier and Luigi Santocanale. Cuts for circular proofs: semantics and cut-elimination. In *Computer Science Logic 2013, CSL 2013*, volume 23 of *LIPIcs*, pages 248–262. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2013. doi:10.4230/LIPIcs.CSL.2013.248.

21 Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–101, 1987. doi:10.1016/0304-3975(87)90045-4.

22 Jean-Yves Girard. Light linear logic. *Information and Computation*, 143(2):175–204, 1998. doi:10.1006/inco.1998.2700.

23 Denis Kuperberg, Laureline Pinault, and Damien Pous. Cyclic proofs, system T, and the power of contraction. *Proc. ACM Program. Lang.*, 5(POPL):1–28, 2021. doi:10.1145/3434282.

24 Yves Lafont. Soft linear logic and polynomial time. *Theoretical Computer Science*, 318(1):163–180, 2004. Implicit Computational Complexity. doi:10.1016/j.tcs.2003.10.018.

25 Damiano Mazza. Non-uniform polytime computation in the infinitary affine lambda-calculus. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming – 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part II*, volume 8573 of *Lecture Notes in Computer Science*, pages 305–317. Springer, 2014. doi:10.1007/978-3-662-43951-7_26.

26 Damiano Mazza. Simple parsimonious types and logarithmic space. In *24th EACSL Annual Conference on Computer Science Logic, CSL 2015*, volume 41 of *LIPIcs*, pages 24–40. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.CSL.2015.24.

27 Damiano Mazza and Kazushige Terui. Parsimonious types and non-uniform computation. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Automata, Languages, and Programming – 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part II*, volume 9135 of *Lecture Notes in Computer Science*, pages 350–361. Springer, 2015. doi:10.1007/978-3-662-47666-6_28.

28 Vivek Nigam and Dale Miller. Algorithmic specifications in linear logic with subexponentials. In António Porto and Francisco Javier López-Fraguas, editors, *Proceedings of the 11th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming*, pages 129–140. ACM, 2009. doi:10.1145/1599410.1599427.

29 Damian Niwiński and Igor Walukiewicz. Games for the μ-calculus. *Theoretical Computer Science*, 163(1-2):99–116, 1996. doi:10.1016/0304-3975(95)00136-0.

30 Michele Pagani and Lorenzo Tortora de Falco. Strong normalization property for second order linear logic. *Theor. Comput. Sci.*, 411(2):410–444, January 2010. doi:10.1016/j.tcs.2009.07.053.

31 Myriam Quatrini. *Sémantique cohérente des exponentielles: de la logique linéaire à la logique classique*. PhD thesis, Aix-Marseille 2, 1995.

32 Luca Roversi and Luca Vercelli. Safe recursion on notation into a light logic by levels. In Patrick Baillot, editor, *Proceedings International Workshop on Developments in Implicit Computational complExity, DICE 2010, Paphos, Cyprus, 27-28th March 2010*, volume 23 of *EPTCS*, pages 63–77, 2010. doi:10.4204/EPTCS.23.5.

33 Alexis Saurin. A linear perspective on cut-elimination for non-wellfounded sequent calculi with least and greatest fixed points (extended version). working paper or preprint, 2023. URL: https://hal.science/hal-04169137.

34 Alex Simpson. Cyclic arithmetic is equivalent to peano arithmetic. In Javier Esparza and Andrzej S. Murawski, editors, *Foundations of Software Science and Computation Structures – 20th International Conference, FOSSACS 2017, Proceedings*, volume 10203 of *Lecture Notes in Computer Science*, pages 283–300, 2017. doi:10.1007/978-3-662-54458-7_17.

35 Morten Heine Sørensen and Pawel Urzyczyn. *Lectures on the Curry-Howard Isomorphism, Volume 149 (Studies in Logic and the Foundations of Mathematics)*. Elsevier Science Inc., USA, 2006.

36 Terese. *Term rewriting systems*, volume 55 of *Cambridge tracts in theoretical computer science*. Cambridge University Press, 2003.

37 A. S. Troelstra and H. Schwichtenberg. *Basic Proof Theory*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2 edition, 2000. doi:10.1017/CBO9781139168717.

Descriptive Complexity for Neural Networks via Boolean Networks

Veeti Ahvonen
Tampere University, Finland

Damian Heiman
Tampere University, Finland

Antti Kuusisto
Tampere University, Finland

Abstract

We investigate the descriptive complexity of a class of neural networks with unrestricted topologies and piecewise polynomial activation functions. We consider the general scenario where the running time is unlimited and floating-point numbers are used for simulating reals. We characterize these neural networks with a rule-based logic for Boolean networks. In particular, we show that the sizes of the neural networks and the corresponding Boolean rule formulae are polynomially related. In fact, in the direction from Boolean rules to neural networks, the blow-up is only linear. We also analyze the delays in running times due to the translations. In the translation from neural networks to Boolean rules, the time delay is polylogarithmic in the neural network size and linear in time. In the converse translation, the time delay is linear in both factors. We also obtain translations between the rule-based logic for Boolean networks, the diamond-free fragment of modal substitution calculus and a class of recursive Boolean circuits where the number of input and output gates match.

2012 ACM Subject Classification Computing methodologies → Neural networks; Theory of computation → Finite Model Theory; Mathematics of computing → Numerical analysis; Computer systems organization → Parallel architectures

Keywords and phrases Descriptive complexity, neural networks, Boolean networks, floating-point arithmetic, logic

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.9

Related Version *Full Version*: `https://arxiv.org/abs/2308.06277`

Funding Antti Kuusisto was supported by the Academy of Finland project Theory of computational logics, grant numbers 352419, 352420, 353027, 324435, 328987. Damian Heiman was supported by the same project, grant number 353027. Antti Kuusisto was also supported by the Academy of Finland consortium project Explaining AI via Logic (XAILOG), grant number 345612. Veeti Ahvonen was supported by the Vilho, Yrjö and Kalle Väisälä Foundation of the Finnish Academy of Science and Letters.

1 Introduction

This article investigates the descriptive complexity of neural networks, giving a logical characterization for a class of general neural networks which have the topology of directed graphs and unlimited running time. The characterization is based on *Boolean networks* [5, 14]. Boolean networks have a long history, originating from the work of Kauffman in the 1960s [10]. Current applications include a wide variety of research relating to topics varying from biology and medicine to telecommunications and beyond, see, e.g., [16, 15, 14].

Boolean networks are usually not defined via a logical syntax, but it is easy to give them one as follows. Consider the set $\mathcal{T} = \{X_1, \dots, X_k\}$ of Boolean variables. A *Boolean rule* over $\mathcal{T}$ is an expression of the form $X_i :- \varphi$ where $X_i \in \mathcal{T}$ is a *head predicate* and φ is a Boolean

formula over the syntax $\varphi ::= \top \mid X_j \mid \neg\varphi \mid \varphi \wedge \varphi$, where $X_j \in \mathcal{T}$. A *Boolean program* over $\mathcal{T}$ is then a set of Boolean rules (over $\mathcal{T}$), one rule for each X_i. Given an input $f : \mathcal{T} \to \{0,1\}$ and executing the rules in parallel, the program then produces a time-series of k-bit strings in a natural way (see the preliminaries section for the full details). An extended Boolean program over $\mathcal{T}$ is a Boolean program over some $\mathcal{S} \supseteq \mathcal{T}$ together with a *terminal clause* $X_j(0) :- b$ for each $X_j \in \mathcal{S} \setminus \mathcal{T}$, where $b \in \{\top, \bot\}$. Extended programs produce time-series just like regular programs, but they also contain *auxiliary variables* $X_j \in \mathcal{S} \setminus \mathcal{T}$ whose initial value is not part of the input but is instead given via a terminal clause (cf. the preliminaries section). The logic used in this paper, **Boolean network logic** BNL, consists of extended Boolean programs.

It turns out that BNL is also closely related to the diamond-free fragment of *modal substitution calculus* MSC used in [11] to characterize distributed message passing automata. Calling that fragment SC (for *substitution calculus*), we prove that programs of SC and BNL can be translated to each other with only a linear increase in program size. Thereby our characterization via BNL can alternatively be obtained via SC. Moreover, we also show that BNL is closely related to *self-feeding circuits*. Informally, self-feeding circuits are a class of Boolean circuits where the number of input and output gates match. Each self-feeding circuit is associated with an initializing function. An initializing function fixes the input for some subset of the set of input gates. The fixed gates are intuitively "auxiliary gates"; this is analogous to the terminal clauses for auxiliary variables in a BNL-program. To execute a self-feeding circuit, it is given an input that consists of values for the non-auxiliary input gates. With a given input, a self-feeding circuit induces a time-series of bit strings (whose length matches the number of output gates) as follows. In round zero, the bit string is obtained as a combination of the values given by the initializing function and input. In each subsequent round n, the string in round n is obtained by feeding the string from the previous round $n-1$ to the circuit. We prove that programs of BNL and self-feeding circuits can likewise be translated to each other with only a linear increase in size.

The neural network (NN) model we consider is very general. We allow unrestricted topologies for the underlying directed graphs, including loops, thereby considering the recurrent setting. The reals are modeled via floating-point numbers and the running times are unlimited. We show that for each NN, there exists a corresponding program of BNL that simulates the time series of the NN for each input, and vice versa, BNL-programs can – likewise – be simulated by NNs. Furthermore, the sizes of the NNs and BNL-programs are shown to be polynomially related.

In a bit more detail, let $S = (p, q, \beta)$ denote a floating-point system with *fraction precision* p, *exponent precision* q and *base* β (see Section 3.2 for the definitions). Let N denote the number of nodes in an NN and Δ the maximum degree of the underlying graph. Modeling activation functions via piecewise polynomial functions, let P denote the number of pieces required and Ω the maximum order of the involved polynomials. Then the following holds.

▶ **Theorem 14.** *Given a general neural network $\mathcal{N}$ for $S = (p, q, \beta)$ with N nodes, degree Δ, piece-size P and order Ω, we can construct a* BNL*-program Λ such that $\mathcal{N}$ and Λ are asynchronously equivalent in S where for $r = \max\{p, q\}$,*

1. *the size of Λ is $\mathcal{O}(N(\Delta + P\Omega^2)(r^4 + r^3\beta^2 + r\beta^4))$, and*
2. *the computation delay of Λ is $\mathcal{O}((\log(\Omega) + 1)(\log(r) + \log(\beta)) + \log(\Delta))$.*

Here (and also in the below theorem) *asynchronous equivalence* means that the modeled time series can be repeated but with a delay between *significant computation rounds*. The time delays in our results are not arbitrary but rather modest. For modeling Boolean network

logic via a general neural network, let the depth of a program refer to the maximum nesting depth of Boolean formulas appearing in rules. Our result is for NNs that use the activation function $\mathrm{ReLU}(x) = \max\{0, x\}$, but it can be generalized for other activation functions.

▶ **Theorem 15.** *Given a* BNL*-program* Λ *of size* s *and depth* d*, we can construct a general neural network* $\mathcal{N}$ *for any floating-point system* S *with at most* s *nodes, degree at most* 2*,* ReLU *activation functions and computation delay* $\mathcal{O}(d)$ *(or* $\mathcal{O}(s)$ *since* $s > d$*) such that* Λ *and* $\mathcal{N}$ *are asynchronously equivalent in binary.*

It is worth noting that in our setting, while we allow for general topologies and unlimited running times, our systems have inherently finite input spaces. In the framework of NN models, this is a well-justified assumption for a wide variety of modeling purposes. Our results stress the close relations between the size and time resources of general NNs and BNL-programs. Furthermore, as outputs we consider time series rather than a single-output framework. Indeed, it is worth noting that trivially a single Boolean function suffices to model any NN with a finite input space when limiting to single outputs only and not caring about size and time blow-ups in translations.

Related work. The closely related topic of descriptive complexity of graph neural networks (or GNNs) has been studied by Barceló et al. in [3], and by Grohe in [7], [6]. In [7], the GNNs operate via feedforward neural networks, and a natural connection between these models and the circuit complexity class TC^0 is established via logic. The feedforward model in [7] uses *rational piecewise linear approximable* activation functions, meaning that the parameters of the linear representations of activation functions are finitely representable in a binary floating-point system. In the current paper, we allow floating-point systems with an arbitrary base, which can be useful, as a change of base often allows inadmissible reals to become admissible. Moreover, our activation functions are piecewise polynomially definable, meaning that most of the widely used activation functions are directly representable in our framework, e.g., ReLU. Furthermore, practically all activation functions are reasonably approximable.

Neural networks are special kinds of distributed systems, and descriptive complexity of distributed computing was initiated in Hella et al. [8], Kuusisto [11] and Hella et al. [9] by relating distributed complexity classes to modal logics. While [8] and [9] gave characterizations to constant-time distributed complexity classes via modal logics, [11] lifted the work to general non-constant-time algorithms by characterizing finite distributed message passing automata via the modal substitution calculus MSC. This was recently lifted to circuit-based networks with identifiers in Ahvonen et al. [1], also utilizing modal substitution calculus. The logic MSC has been linked to a range of other logics. For example, in [11], it is shown to contain the μ-fragment of the modal μ-calculus, and it is easy to translate MSC into a fragment of partial fixed-point logic. Building on [11], Reiter shows in [13] that this fragment of the modal μ-calculus captures finite message passing automata in the asynchronous setting. It is worth noting here that also [3] utilizes modal logic, establishing a match between aggregate-combine graph neural networks and graded modal logic. The logics BNL and SC used in this article are rule-based systems. Rule-based logics are used widely in various applications, involving systems such as Datalog, answer-set programming (ASP) formalisms, and many others.

2 Preliminaries

First we introduce some basic concepts. For any set S, we let $\wp(S)$ denote the power set of S and we let $|S|$ denote the size (or cardinality) of S. We let $\mathbb{N}$ and $\mathbb{Z}_+$ denote the sets of non-negative and positive integers respectively. For every $n \in \mathbb{Z}_+$, we let $[n] = \{1, \dots, n\}$ and $[0; n] = \{0, \dots, n\}$. We let bold lower-case letters $\mathbf{a}, \mathbf{b}, \mathbf{c}, \dots$ denote strings. The letters of a

string are written directly next to each other, i.e. abc, or with dots in-between, i.e. $a \cdot b \cdot c$, or a mix of both, i.e. $abc \cdot def$. Omitted segments of strings are represented with three dots, i.e. $abcd \cdots wxyz$. If $\mathbf{s} = s_0 \cdots s_{k-1}$ is a string of length k, then for any $j \in [0; k-1]$, we let $\mathbf{s}(j)$ denote the letter s_j. The alphabet for the strings will depend on the context. We let $\mathrm{VAR} = \{ V_i \mid i \in \mathbb{N} \}$ denote the (countably infinite) set of all **schema variables**. Mostly, we will use meta variables X, Y, Z and so on, to denote symbols in VAR. We assume a linear order $<^{\mathrm{VAR}}$ over the set VAR. Moreover, for any set $\mathcal{T} \subseteq \mathrm{VAR}$, a linear order $<^{\mathcal{T}}$ is induced by $<^{\mathrm{VAR}}$. We let $\mathrm{PROP} = \{ p_i \mid i \in \mathbb{N} \}$ denote the (countably infinite) set of **proposition symbols** that is associated with the linear order $<^{\mathrm{PROP}}$, inducing a linear order $<^{P}$ over any subset $P \subseteq \mathrm{PROP}$. We let $\Pi \subseteq \mathrm{PROP}$ denote a finite subset of proposition symbols. When we talk about **rounds** in any context, we refer to non-negative integers that are interpreted as discrete steps of a computation.

2.1 Discrete time series

Next we consider infinite sequences of bit strings, i.e., we consider *discrete time series* of strings over the alphabet $\{0, 1\}$. To separate important strings from less important ones, we need to define when a time series produces an output; importantly, we allow an arbitrary number of outputs. We will define two separate general output conditions for time series. In the first approach, special bits indicate when to output. In the second approach the output rounds are fixed, and we do not include bits that indicate when to output.

The formal definition for the first approach is as follows. Let $k \in \mathbb{Z}_+$ and let B denote an infinite sequence $(\mathbf{b}_j)_{j \in \mathbb{N}}$ of k-bit strings $\mathbf{b}_j \in \{0, 1\}^k$. Let $A \subseteq [k]$ and $P \subseteq [k]$ be subsets of bits called **attention** bits and **print** bits respectively (or bit positions, strictly speaking). The sets A and P induce corresponding sequences $(\mathbf{a}_j)_{j \in \mathbb{N}}$ and $(\mathbf{p}_j)_{j \in \mathbb{N}}$ of substrings of the strings in B. More formally, $(\mathbf{a}_j)_{j \in \mathbb{N}}$ records the substrings with positions in A, and $(\mathbf{p}_j)_{j \in \mathbb{N}}$ records the substrings with positions in P. Next we define output conditions for B with respect to attention and print bits. If at least one bit in $\mathbf{a}_n$ is 1 (for some $n \in \mathbb{N}$), then we say that B **outputs** $\mathbf{p}_n$ in round n and that n is an **output round**. More precisely, B **outputs in round n with respect to (k, A, P)**, and $\mathbf{p}_n$ is the **output of B in round n with respect to (k, A, P)**. Let $O \subseteq \mathbb{N}$ be the set of output rounds induced by B w.r.t. (k, A, P); they induce a subsequence $(\mathbf{b}_j)_{j \in O}$ of B. We call the sequence $(\mathbf{p}_j)_{j \in O}$ the **output sequence** of B (w.r.t. (k, A, P)).

Next we define an output condition where output rounds are fixed by a set $O \subseteq \mathbb{N}$ and attention bits are excluded. We say that S **outputs** in rounds O (and also, in any particular round $n \in O$). Outputs and output sequences w.r.t. (k, O, P) are defined analogously.

We study the two approaches for the sake of generality. The difference between the two output frameworks is that the output rounds are induced internally from within the sequence in the first approach, while they are given externally from the outside in the second one. For instance, it is natural to indicate output conditions within a program if it is part of the program's design. Retroactively, it might be more natural to augment a program to draw attention to rounds the original design doesn't account for, and a different mechanism could be used to compute the output rounds, e.g., a Turing machine.

2.2 Modal substitution calculus MSC and Boolean network logic BNL

We next define modal substitution calculus MSC introduced in [11]. Let $\Pi \subseteq \mathrm{PROP}$ be a finite set of proposition symbols and $\mathcal{T} \subseteq \mathrm{VAR}$. A **terminal clause** (over $(\Pi, \mathcal{T})$) is a string of the form $V_i(0) :- \varphi$, where $V_i \in \mathcal{T}$ and φ is defined over the language $\varphi ::= \top \mid p_i \mid \neg\varphi \mid \varphi \wedge \varphi \mid \Diamond\varphi$ where $p_i \in \Pi$ (i.e., φ is a formula of modal logic over Π). An **iteration clause** (over $(\Pi, \mathcal{T})$) is

a string of the form $V_i :- \psi$ where ψ is a $(\Pi, \mathcal{T})$**-formula of modal substitution calculus** (or MSC) defined over the language $\psi ::= \top \mid p_i \mid V_i \mid \neg\psi \mid \psi \wedge \psi \mid \Diamond\psi$ where $p_i \in \Pi$ and $V_i \in \mathcal{T}$. We also use symbols $\bot, \vee, \rightarrow$ and $\leftrightarrow$ as shorthand in the usual way. In a terminal clause $V_i(0) :- \varphi$, the symbol V_i is the **head predicate** and φ is the **body** of the clause. Analogously, for an iteration clause $V_i :- \psi$, we say that V_i is the head predicate and ψ is the body of the iteration clause.

Let $\mathcal{T} = \{X_1, \ldots, X_n\} \subseteq \mathrm{VAR}$ be a finite, nonempty set of $n \in \mathbb{Z}_+$ distinct schema variables. A $(\Pi, \mathcal{T})$-**program** of MSC is defined as a list

$$\begin{array}{ll} X_1(0) :- \varphi_1 & X_1 :- \psi_1, \\ \vdots & \vdots \\ X_n(0) :- \varphi_n & X_n :- \psi_n, \end{array}$$

where each schema variable in $\mathcal{T}$ has precisely one terminal and one iteration clause. Informally, the terminal and iteration clauses of a program can be seen as the "rules" of the program. The diamond-free fragment of MSC called **substitution calculus** SC simply restricts the terminal and iteration clauses, not allowing diamonds $\Diamond$. Note that in SC the bodies of terminal clauses are thereby formulae of propositional logic. Moreover, the programs of MSC (and SC) are also associated with a set $\mathcal{P} \subseteq \mathcal{T}$ of **print predicates** and either a set $\mathcal{A} \subseteq \mathcal{T}$ of **attention predicates** or an **attention function** $A\colon \{0,1\}^k \to \wp(\mathbb{N})$, where k is the number of distinct proposition symbols that appear in the program. Informally, the attention predicates and the attention function are analogous to the two output conditions discussed for time series. We will later discuss how either can be used to determine a set of output rounds for the program. We use attention predicates by default, and only discuss the attention function when specified.

Usually, a run of a program of MSC is defined over a Kripke-model, but a run of a $(\Pi, \mathcal{T})$-program of SC is defined over a **model** M of propositional logic, i.e., M is a valuation $\Pi \to \{0,1\}$ assigning a truth value to each proposition symbol in Π. The semantics of formulae of propositional logic in model M are defined as follows: $M \models p_i$ (read: p_i is true in M) iff the valuation of p_i is 1, and the semantics of $\wedge$, $\neg$ and $\top$ is the usual one. If $\Pi' \subseteq \Pi$ is the set of proposition symbols that appear in the program, the linear order $<^{\Pi'}$ and the model M induce a binary string $\mathbf{i} \in \{0,1\}^{|\Pi'|}$ that serves as input, i.e., the ith bit of $\mathbf{i}$ is 1 iff the valuation of the ith proposition in Π' is 1. Let Λ be a $(\Pi, \mathcal{T})$-program of MSC. The truth of a $(\Pi, \mathcal{T})$-formula ψ in round $n \in \mathbb{N}$ (written $M \models \psi^n$) is defined as follows: **1)** $M \models \top^n$ always holds, **2)** $M \models p_i^n$ iff $M \models p_i$ (where $p_i \in \Pi$), **3)** if $\psi := \neg\theta$, then $M \models (\neg\theta)^n$ iff $M \not\models \theta^n$, **4)** if $\psi := (\chi \wedge \theta)$, then $M \models (\chi \wedge \theta)^n$ iff $M \models \chi^n$ and $M \models \theta^n$, and **5)** the truth of a head predicate X_i is defined separately as follows. We define that $M \models X_i^0$ if $M \models \varphi_i$ where φ_i is the body of the terminal clause of X_i in Λ. Assume we have defined the truth of all $(\Pi, \mathcal{T})$-formulae in round n. We define that $M \models X_i^{n+1}$ iff $M \models \psi_i^n$ where ψ_i is the body of the iteration clause of X_i in Λ.

We then define **Boolean network logic** (or BNL) which we will later show to be equivalent to the fragment SC. Boolean network logic gets its name from Boolean networks, which are discrete dynamical systems commonly used in various fields, e.g., biology, telecommunications and various others. For example, they are used to describe genetic regulatory networks (e.g., [10]). A Boolean network consists of a set of Boolean variables, i.e. variables that only get Boolean values 0 or 1. Each variable is given an initial Boolean value called the "seed". The Boolean values of all variables are updated in discrete steps starting with the seed. In each step, each variable updates its Boolean value using its own Boolean function. The updated value is determined from the Boolean values of all variables in the previous step. There is no general syntax for Boolean networks, but BNL will give us a suitable one.

Let $\mathcal{T} \subseteq \mathrm{VAR}$. A **$\mathcal{T}$-formula of Boolean network logic** (or BNL) is defined over the language $\psi ::= \top \mid V_i \mid \neg\psi \mid \psi \wedge \psi$, where $V_i \in \mathcal{T}$ (i.e. we do not include propositions). Assume now that $\mathcal{T}$ is finite and nonempty. There are three main differences between $\mathcal{T}$-programs of BNL and SC: **1)** The terminal clauses of BNL are either of the form $X(0) :- \top$ or $X(0) :- \bot$. **2)** The bodies of iteration clauses of BNL are $\mathcal{T}$-formulae of BNL. **3)** Each schema variable in a BNL-program has exactly one iteration clause and either one or zero terminal clauses. We let $\mathcal{I}$ denote the predicates that do not have terminal clauses, which we call **input predicates**. For example, consider a BNL-program with the terminal clause $X(0) :- \top$ and the iteration clauses $Y :- Y \wedge X$ and $X :- \neg X$. Here Y is the sole input predicate and X acts as an auxiliary predicate. Bodies and head predicates of clauses are defined analogously to SC (and MSC). A BNL-program also includes print predicates and either attention predicates or an attention function $A\colon \{0,1\}^k \to \wp(\mathbb{N})$, where $k = |\mathcal{I}|$.

The run of a program of BNL is defined over a **model** $\mathcal{M}$, i.e. $\mathcal{M}$ is a valuation $\mathcal{I} \to \{0,1\}$. Analogously to a model of SC, any model for BNL and the set $\mathcal{I}$ induce a binary string $\mathbf{i} \in \{0,1\}^{|\mathcal{I}|}$ that serves as input. (Note that vice versa each string $\mathbf{i} \in \{0,1\}^{|\mathcal{I}|}$ induces a model with valuation $\mathcal{I} \to \{0,1\}$ such that $I_j \mapsto \mathbf{i}(j)$ if $I_0, \ldots, I_{|I|-1}$ enumerates the set $\mathcal{I}$ in the order $<^{\mathrm{VAR}}$.) The truth of a $\mathcal{T}$-formula is defined analogously to SC except for the truth value of head predicates in round 0. If $X \in \mathcal{I}$, we define that $\mathcal{M} \models X^0$ if the valuation of X in $\mathcal{M}$ is 1. If $X \notin \mathcal{I}$, then $\mathcal{M} \models X^0$ if the body of the terminal clause of X is $\top$.

Let $X_1, \ldots, X_n$ enumerate the set $\mathcal{T}$ of schema variables (in the order $<^{\mathrm{VAR}}$). Let Λ be a $\mathcal{T}$-program of SC (or BNL), and M a model for SC (or respectively $\mathcal{M}$ for BNL) that induces an input $\mathbf{i} \in \{0,1\}^k$, where k is the number of proposition symbols (or resp. the number of input predicates). Each time step (or round) $t \in \mathbb{N}$ defines a **global configuration** $g_t\colon \mathcal{T} \to \{0,1\}$. The global configuration at time step t is induced by the values of head predicates, i.e., $g_t(X_i) = 1$ iff $M \models X_i^t$ (or resp. $\mathcal{M} \models X_i^t$), for each $X_i \in \mathcal{T}$. Thus, an SC-program (or BNL-program) also induces an infinite sequence $(\mathbf{s}_t)_{t\in\mathbb{N}}$ called the **global configuration sequence** (with input $\mathbf{i}$), where $\mathbf{s}_t = g_t(X_1) \cdots g_t(X_n)$. The set of print predicates $\mathcal{P}$ corresponds to the set of print bits $\{\, i \mid X_i \in \mathcal{P} \,\}$. If the program has attention predicates $\mathcal{A}$, then $\mathcal{A}$ corresponds to the set of attention bits $\{\, i \mid X_i \in \mathcal{A} \,\}$. If the program has an attention function A, then the output rounds are given by $A(\mathbf{i})$. Therefore, analogously to the general output conditions defined for infinite sequences of bit strings, a program of SC or BNL with an input $\mathbf{i}$ also induces **output rounds** and an **output sequence** w.r.t. $(n, \mathcal{A}, \mathcal{P})$ (or resp. w.r.t. $(n, A(\mathbf{i}), \mathcal{P})$).

We say that a $(\Pi, \mathcal{T})$-program of SC and a $\mathcal{T}'$-program of BNL (or likewise, two BNL-programs) are **asynchronously equivalent** if they have the same output sequences with every input. We say that they are **globally equivalent** if they *also* have the same global configuration sequences and output rounds with each input (note that identical inputs require that $|\Pi'| = |\mathcal{I}|$, where $\Pi' \subseteq \Pi$ is the set of proposition symbols that appear in the SC-program and $\mathcal{I}$ is the set of input predicates of the BNL-program). We define the delay between two asynchronously equivalent objects x and y. Let $x_1, x_2, \ldots$ and $y_1, y_2, \ldots$ enumerate their (possibly infinite) sets of output rounds in ascending order. Assume that the cardinality of the sets of output rounds is the same and $x_n \geq y_n$ for every $n \in \mathbb{N}$. If T is the smallest amount of time steps (that might depend on x or y) such that $T \cdot y_n \geq x_n$ for every $n \in \mathbb{N}$, then we say that the **computation delay** of x is T. The case for $y_n \geq x_n$ is analogous.

The **size** of a program of SC or BNL is defined as the number of appearances of $\top$, proposition symbols p_i, head predicates V_i and logical connectives $\neg$ and $\wedge$ in its terminal and iteration clauses. The **depth** $d(\psi)$ of a BNL-formula or SC-formula is defined recursively:

1) $d(p_i) = d(\top) = d(X) = 0$, where p_i is a proposition symbol and X is a schema variable, **2)** $d(\neg\psi) = d(\psi) + 1$ and **3)** $d(\psi \wedge \theta) = \max\{d(\psi), d(\theta)\} + 1$. The **depth** of a BNL-program is the maximum depth of the bodies of its iteration clauses.

BNL-programs inherit a number of properties from Boolean networks. Each reachable combination of truth values for the head predicates (i.e., each reachable global configuration) is called a **state** and together they form a **state space**. Note that certain global configurations may not be reachable, because neither they nor their preceding states are possible states at round 0 due to the terminal clauses of the BNL-program. Given that the number of states is finite, a BNL-program will eventually either reach a single stable state or begin looping through a sequence of states. A stable state is called a **point attractor**, a **fixed-point attractor** or simply a **fixed point**, whereas a looping sequence of multiple states is a **cycle attractor**. The smallest amount of time it takes to reach an attractor from a given state is called the **transient time** of that state. The **transient time** of a BNL-program is the maximum transient time of a state in its state space [5]. The concept of transient time is also applicable to SC, since it is also deterministic and eventually stabilizes with each input.

Consider the fragment BNL_0 where no head predicate of a program is allowed to have a terminal clause. The programs of this logic BNL_0 are an exact match with Boolean networks; each program encodes a Boolean network, and vice versa. The logic BNL extends this framework by allowing terminal clauses.

A BNL-program that only has fixed points (i.e., no input leads to a cycle attractor) and outputs precisely at fixed points, is called a **halting** BNL**-program**. For a halting BNL-program Λ with input predicates $\mathcal{I}$ and print predicates $\mathcal{P}$, each input $\mathbf{i} \in \{0,1\}^{|\mathcal{I}|}$ results in a single (repeating) **output** denoted by $\Lambda(\mathbf{i})$, which is the output string determined by the fixed-point values of the print predicates. In this sense, a halting BNL-program is like a function $\Lambda\colon \{0,1\}^{|\mathcal{I}|} \to \{0,1\}^{|\mathcal{P}|}$. We say that Λ **specifies** a function $f\colon \{0,1\}^{\ell} \to \{0,1\}^{k}$ if $|\mathcal{I}| = \ell$, $|\mathcal{P}| = k$ and $\Lambda(\mathbf{i}) = f(\mathbf{i})$ for all $\mathbf{i} \in \{0,1\}^{\ell}$. The **computation time** of a halting BNL-program is its transient time.

We introduce two useful tools that are used when constructing BNL-programs (these tools are also definable via MSC or SC). Flagging is one of the most useful tools similar to adding "if-else" conditions in programming. Given two formulae φ and χ, and a rule $X :\!- \psi$, **flagging** X (w.r.t. φ and χ) means rewriting the rule $X :\!- \psi$ as $X :\!- (\varphi \wedge \psi) \vee (\neg\varphi \wedge \chi)$. Now, if φ is true then the truth value of X depends on the truth value of ψ, and if φ is false then the truth value of X depends on the truth value of χ. We call φ the **flag** and χ the **backup**. Often χ is X itself meaning that the truth value of X does not change if φ is false. Using flags, it is possible to create branches in a program, and thereby combine subprograms into a single, bigger program.

A **one-hot counter** is defined as a sequence of schema variables $T_0, T_1, \ldots, T_n$ with the terminal clauses $T_0(0) :\!- \top$ and $T_i(0) :\!- \bot$ for all $i \geq 1$, and iteration clauses $T_0 :\!- T_n$ and $T_i :\!- T_{i-1}$ for all $i \geq 1$. Exactly one of these schema variables is true in any one time step, and they turn on in a looping sequence from left to right. T_t is true in round t for all $t \leq n$. In round $n+1$, T_0 is true again and the cycle continues. This is ideal for flagging: T_n can be used as a flag for attention predicates to trigger an output round once every n time steps.

We are ready to prove that BNL is equivalent to SC.

▶ **Theorem 1.** *Each* SC*-program of transient time T has an asynchronously equivalent* BNL*-program of linear size and transient time $T+1$, and each* BNL*-program has a globally equivalent* SC*-program of linear size.*

Proof sketch. For the full proof, see [2]. From SC to BNL, we create a BNL-program that uses one time step to compute the terminal clauses of the SC-program; the terminal clauses of the SC-program are embedded into the iteration clauses of the BNL-program using a flag. From BNL to SC, we amend the BNL-program with the missing terminal clauses using proposition symbols. ◀

2.3 Link to self-feeding circuits

In this section we introduce self-feeding circuits. We will show that for every BNL-program, we can construct an equivalent self-feeding circuit and vice versa. We also pay special attention to the size and time complexities in the translations.

We first recall some basics related to circuits and then define a related self-feeding circuit model. A **Boolean circuit** is a directed, acyclic graph where at least each node of non-zero in-degree is labeled by one of the symbols $\wedge, \vee, \neg$. The nodes of a circuit are called **gates**. The in-degree of a gate u is called the **fan-in** of u, and the out-degree of u is the **fan-out**. The **input gates** of a circuit are precisely the gates that have zero fan-in and no label $\wedge, \vee$ or $\neg$. The **output gates** are the ones with fan-out zero; we allow multiple output gates in a circuit. The fan-in of every gate labeled with $\neg$ is 1.

The **size** $|C|$ of a circuit C is the number of gates in C. The **depth** $\mathsf{depth}(C)$ (or the **computation/evaluation time**) of C is the length of the longest path (number of edges) from an input gate to an output gate. The **height** $\mathsf{height}(G)$ of a gate G in C is the length of the longest path from an input gate to the gate G. The sets of input and output gates of a circuit are both linearly ordered. A circuit with n input gates and k output gates then **computes** (or specifies) a function of type $\{0,1\}^n \to \{0,1\}^k$. This is done in the natural way, analogously to the Boolean operators corresponding to $\wedge, \vee, \neg$; see, for example, [12] for the formal definition. The output of the circuit is the *binary string* determined by the bits of the output gates. Note that gates with the labels $\wedge, \vee$ can have any fan-in (also 0), meaning that by default, circuits have **unbounded** fan-in. In the elaborations below, we say a circuit is **fan-in bounded** (or the circuit has a bounded fan-in) if the fan-in of every $\wedge$-gate and $\vee$-gate of the circuit is at most 2. The $\wedge$-gates that have zero fan-in always output 1, and therefore correspond to the symbol $\top$. The $\vee$-gates that have zero fan-in always output 0 and therefore, similarly, correspond to the symbol $\bot$.

Analogously to circuits, a Boolean formula φ with n variables specifies a function of type $\{0,1\}^n \to \{0,1\}$. Let $\mathcal{B}$ denote the set of all Boolean formulas, and let $\mathcal{C}$ denote the set of all circuits. Given x and y in the set $\mathcal{B} \cup \mathcal{C}$, we say that x and y are **equivalent** if they specify the same function.

Let $k \in \mathbb{Z}_+$. A **self-feeding circuit for** k is a circuit C that specifies a function

$$f : \{0,1\}^k \to \{0,1\}^k.$$

The circuit C is associated with a set of **input positions** $I \subseteq [k]$ and an **initializing function** $\pi\colon [k] \setminus I \to \{0,1\}$. The elements of $[k] \setminus I$ are called **auxiliary positions**. Moreover, C is also associated with a set $P \subseteq [k]$ of **print positions** and either with a set $A \subseteq [k]$ of **attention positions** or an **attention function** $a\colon \{0,1\}^{|I|} \to \wp(\mathbb{N})$.

Informally, a self-feeding circuit computes as follows. The non-auxiliary input gates are fed with the input and the auxiliary input gates are fed with the values given by the initializing function; then the circuit produces an output in the ordinary way. After that in each round n the output from the previous round $n-1$ is fed back to the circuit itself to produce a new binary string. We then define the computation of self-feeding circuits formally. Let C be a self-feeding circuit for k with input positions I and input $i\colon I \to \{0,1\}$

(or the corresponding bit string $\mathbf{i} \in \{0,1\}^{|I|}$). Respectively, the function $\pi \cup i$ corresponds to the binary string $\mathbf{s}_{\pi \cup i} \in \{0,1\}^k$, where for each $j \in [k]$, if $j \in I$, then $\mathbf{s}_{\pi \cup i}(j-1) = i(j)$ and if $j \notin I$, then $\mathbf{s}_{\pi \cup i}(j-1) = \pi(j)$. Each round $n \in \mathbb{N}$ defines a **global configuration** $\mathbf{g}_n \in \{0,1\}^k$. The configuration of round 0 is the k-bit binary string $\mathbf{g}_0 = \mathbf{s}_{\pi \cup i}$. Recursively, assume we have defined $\mathbf{g}_n$. Then $\mathbf{g}_{n+1}$ is the output string of C when it is fed with the string $\mathbf{g}_n$. Now, consider the sequence $(\mathbf{g}_n)_{n \in \mathbb{N}}$ of k-bit strings that C produces. The circuit C with input i (or $\mathbf{i}$) also induces a set of **output rounds** and an **output sequence** w.r.t. (k, A, P) (or $(k, a(\mathbf{i}), P)$). Analogously to SC and BNL, we define asynchronous equivalence, global equivalence and computation delay between two self-feeding circuits or between a self-feeding circuit and a program.

We recall a well-known fact. The lemma below is related to the fact that the Boolean functions in the circuit complexity class NC^1 (with one output gate) are equivalent to the Boolean functions in the class of polynomial-size Boolean formulas.

▶ **Lemma 2** ([4]). *Given a Boolean formula of size n, there exists an equivalent circuit with bounded fan-in, one output gate, size $\mathcal{O}(n^2)$ and formula depth $\mathcal{O}(\log n)$.*

It is easy to obtain the following theorems.

▶ **Theorem 3.** *Given a* BNL*-program of size n and depth d, we can construct a globally equivalent self-feeding circuit with bounded fan-in, size $\mathcal{O}(n)$ and depth d. Moreover, we can also construct a globally equivalent self-feeding circuit with bounded fan-in, size $\mathcal{O}(n^3)$ and depth $\mathcal{O}(\log n)$.*

Proof. We prove both claims at once. Let Λ be a BNL-program of size n. We construct a globally equivalent self-feeding circuit C_Λ as follows. Let $X_1, \ldots, X_m$ enumerate the head predicates and $\psi_1, \ldots, \psi_m$ the corresponding rules of the head predicates of Λ. Let $\mathcal{I}$ denote the set of input predicates of Λ. Each rule ψ_i is transformed into a corresponding circuit C_i with bounded fan-in as follows. To prove the first claim, each C_i is obtained in a straightforward way from the tree representation of ψ_i, and therefore the size of each circuit C_i is linear in the size of ψ_i. Respectively to prove the second claim, each C_i is obtained by applying Lemma 2. We combine each circuit C_i to one circuit C_Λ such that they share the common input gates. The initializing function π is defined as follows. If $X_i \notin \mathcal{I}$, then $\pi(i) = 1$ if the rule of the terminal clause of X_i is $\top$ and respectively $\pi(i) = 0$ if the rule of the terminal clause of X_i is $\bot$. The depth of the obtained circuit C_Λ is $\mathcal{O}(\log n)$ if each circuit C_i is obtained by applying Lemma 2 and otherwise the depth is d, since combining circuits does not affect the depth. The size of C_Λ is $\mathcal{O}(n^3)$ if Lemma 2 was applied to circuits C_i since there are at most n head predicates and the size of each C_i is $\mathcal{O}(n^2)$. Otherwise the size of C is $\mathcal{O}(n)$ since each C_i was linear in the size of the corresponding rule ψ_i. The corresponding input positions, print positions and attention positions (or attention function) are straightforward to define. Clearly C_Λ is globally equivalent to Λ. ◀

▶ **Theorem 4.** *Given a self-feeding circuit C with size n, depth d and m edges, we can construct an asynchronously equivalent* BNL*-program of size $\mathcal{O}(n+m)$ and computation delay $\mathcal{O}(d)$. Moreover, if C has bounded fan-in then the size of the program is $\mathcal{O}(n)$.*

Proof. The proof is heavily based on the proof of Lemma 3 and Theorem 4 in [1], but we give a sketch of the proof. We assume that $d > 0$. The case for $d = 0$ is trivial. First we modify C so that we obtain a globally equivalent circuit C' of size $\mathcal{O}(n)$ such that the height of each output gate is $\mathcal{O}(d)$, see for example [1]. We define a one-hot counter $T_0, \ldots, T_{\mathsf{depth}(C')}$ as defined before. We define a head predicate X_G for each gate G in C' as follows. If G is an

$\wedge$-gate at height h and $Y_1, \ldots, Y_k$ are corresponding head predicates of gates that connect to G, then $X_G :- (T_h \wedge Y_1 \wedge \cdots \wedge Y_k) \vee (\neg T_h \wedge \psi_G)$, where ψ_G is X_G if G is not an output gate and otherwise ψ_G is $\bot$. Moreover, if the fan-in of G is zero, then $X_G :- (T_h \wedge \top) \vee (\neg T_h \wedge \psi_G)$. The cases for $\vee$-gates and $\neg$-gates are analogous. Intuitively, the one-hot counter is used as a flag to make sure that each X_G evaluates in the correct time. Let π be the initializing function of C'. If G is the ith input gate and G' is the ith output gate, then we define $X_G :- (T_0 \wedge X_{G'}) \vee (\neg T_0 \wedge X_G)$.

The input, print and attention predicates are the predicates corresponding to the output gates in input, print and attention positions respectively. If C' has an attention function a, then we define the attention function a' such that $a'(\mathbf{i}) = \{\, (\mathsf{depth}(C') + 1)n \mid n \in a(\mathbf{i}) \,\}$. The constructed program is clearly asynchronously equivalent to C. Moreover the computation delay is $\mathcal{O}(d)$ since it takes $\mathsf{depht}(C')$ rounds to "simulate" each round of C. The size is also clearly $\mathcal{O}(n+m)$ and $\mathcal{O}(n)$ if C is fan-in bounded. ◀

3 Arithmetic with BNL

In this section we first show how to carry out integer addition and multiplication in Boolean network logic in parallel. We then extend this demonstration to floating-point arithmetic, including floating-point polynomials and piecewise polynomial functions.

The algorithms we use for integers are mostly well known and thus some of the formal details are spared; they can be found in [2]. Informally, the idea is to split both addition and multiplication into simple steps that are executed in parallel. We will show that we can simulate integer arithmetic (respectively, floating-point arithmetic) by programs whose size is polynomial in the size of the integers (respectively, in the size of the floating-point system). We also analyze the time delays of the constructed programs. The time delay is polylogarithmic in the size of the integers (and resp. in the size of the floating-point system) and sometimes even a constant. Ultimately, the same applies to floating-point polynomials and piecewise polynomial functions.

3.1 Integer arithmetic

We next define how a *halting* BNL-program simulates integer functions in an arbitrary base $\beta \in \mathbb{Z}$, $\beta \geq 2$. Informally, we represent integers with bit strings that are split into substrings of length β, where exactly one bit in each substring is 1 and the others are 0. Formally, let $\mathbf{s}_1, \ldots, \mathbf{s}_k \in \{0,1\}^\beta$ be **one-hots**, i.e. bit strings with exactly one 1. We say that $\mathbf{s} = \mathbf{s}_1 \cdots \mathbf{s}_k$ **corresponds** to $b_1 \cdots b_k \in [0; \beta-1]^k$ if for every b_i, we have $\mathbf{s}_i(b_i) = 1$ (and other values in $\mathbf{s}_i$ are zero). For example, if $\beta = 5$, then $00100 \cdot 01000 \cdot 00001 \in \{0,1\}^{\beta \cdot 3}$ corresponds to $2 \cdot 1 \cdot 4 \in [0;4]^3$. We say that $\mathbf{s}$ is a **one-hot representation** of $b_1 \cdots b_k$.

Using the binary one-hot representations, we can present integers in BNL by assigning each bit with a head predicate that is true if and only if the bit is 1. The sign ($+$ or $-$) of a number can likewise be handled with a single bit that is true iff the sign is positive.

▶ **Definition 5.** *Let $\beta \in \mathbb{Z}$, $\beta \geq 2$, be a base. We say that a halting* BNL*-program Λ **simulates** (or computes) a function $f\colon [0;\beta-1]^\ell \to [0;\beta-1]^k$ if for each input string $\mathbf{i} \in \{0,1\}^{\ell\beta}$ that corresponds to $\mathbf{b} \in [0;\beta-1]^\ell$, the output $\Lambda(\mathbf{i})$ also corresponds to $f(\mathbf{b})$.*

We note that *comparison of two p-length integers in base β can be simulated* with a BNL-program of size $\mathcal{O}(p\beta^2 + p^2)$ and computation time 2. The one-hot representations of the numbers are first coded into input predicates. Then in round 1, auxiliary predicates determine which number has the higher digit in each position, which requires β^2 space for

each of the p positions. Finally in round 2, the attention/output predicates check that if the first number had a lower digit in some position i, then it has a greater digit in some position j to the left of i; this requires $\mathcal{O}(p^2)$ space.

Parallel addition

In this section we construct a parallel integer addition algorithm via BNL-programs. The algorithm is mostly well known and is based on how integer addition is computed in Nick's class NC^1 (sometimes called the carry-lookahead method), i.e., we parallelize the textbook method (sometimes called the long addition algorithm). Here the main difference to integer addition in Nick's class is that we generalize the algorithm for arbitrary bases.

To illustrate our method of carrying out integer addition, consider the following example of adding $\mathbf{x} = 614$ and $\mathbf{y} = 187$ in base 10. Let c_1, c_2 and c_3 denote the carry over digits and let s_1, s_2, s_3 and s_4 denote the digits of the sum $\mathbf{x} + \mathbf{y}$ from right to left. We have

$$c_1 = 1 = \lfloor (4+7)/10 \rfloor, \quad c_2 = 1 = \lfloor (1+8+c_1)/10 \rfloor, \quad c_3 = 0 = \lfloor (6+1+c_2)/10 \rfloor$$

and therefore $s_1 = 1$, $s_2 = 0$, $s_3 = 8$ and $s_4 = 0 = c_3$, since $(4+7) \equiv 1 (\bmod 10)$, $(1+8+1) \equiv 0 (\bmod 10)$ and $(6+1+1) \equiv 8 (\bmod 10)$. Therefore $\mathbf{x} + \mathbf{y} = 0801$, as wanted. As we can see, in order to know that $c_2 = 1$ we have to first check if $c_1 = 1$. In other words, we have to check if a carry from a previous position has been propagated forward.

Now we are ready to prove the following lemma.

▶ **Lemma 6.** *Given a base $\beta \in \mathbb{Z}$, $\beta \geq 2$ and $p \in \mathbb{Z}_+$, adding two numbers in $[0; \beta-1]^p$ can be simulated with a (halting)* BNL*-program of size $\mathcal{O}(p^3 + p\beta^2)$ and computation time $\mathcal{O}(1)$.*

Proof. We start with an informal description. Consider the addition of two p-digit integers $\mathbf{x} = x_p \cdots x_1$ and $\mathbf{y} = y_p \cdots y_1$ in a base $\beta \geq 2$ (we also allow leading zeros). We assume that the signs of both $\mathbf{x}$ and $\mathbf{y}$ are positive since this can be easily generalized for arbitrary signs. For $i \in [p]$, we let $c_1 = \left\lfloor \frac{x_1 + y_1}{\beta} \right\rfloor$ and $c_{i+1} = \left\lfloor \frac{x_{i+1} + y_{i+1} + c_i}{\beta} \right\rfloor$ denote the *carry digits*. We let $\mathbf{s}$ denote the result of the addition, that is, $\mathbf{x} + \mathbf{y} = (x_p \cdots x_1) + (y_p \cdots y_1) = s_{p+1} \cdots s_1 =: \mathbf{s}$, where for $j \in [p]$, $x_j + y_j + c_{j-1} \equiv s_j \bmod \beta$ (if $j = 1$, then c_{j-1} is omitted), and $s_{p+1} = c_p$. The hard part is to compute the carry digits c_i. We note that c_i is 1 if and only if the sum of x_i, y_i and c_{i-1} is at least β. The problem is that the sum of x_i and y_i might be less than β. Therefore, we have to also check if c_{i-1} is 1. To compute c_{i-1} we have to check if the sum of x_{i-1}, y_{i-1} and c_{i-2} is at least β and so on. So in order to compute c_i, we might have to check all previous carry digits.

So, in the worst case for c_i there are $\mathcal{O}(p)$ possibilities where adding x_j and y_j $(j < i)$ might lead the carry digit c_j to become 1 and in the worst case there are $\mathcal{O}(p)$ digits between j and i for whom we need to check if they carry c_j further. Since we are going to use one-hot representations this requires $\mathcal{O}(p^3 + p\beta^2)$ space in total but it can be done in $\mathcal{O}(1)$ time steps, as we will show next.

We will write a BNL-program of size $\mathcal{O}(p^3 + p\beta^2)$ that computes the sum of two integers (where β is the base and p is the length of the integers) in $\mathcal{O}(1)$ steps. We assume that integers $\mathbf{x} = x_p \cdots x_1$ and $\mathbf{y} = y_p \cdots y_1$ in $[0; \beta-1]^p$ (where we allow leading zeros) are encoded to variables $X_{j,m}$ and $Y_{j,m}$, where $j \in [p]$ and $m \in [0; \beta-1]$, using one-hot representation. For example, consider the integer 13. It can be represented in base 10 with the following variables: $Z_{1,0}, \ldots, Z_{1,9}$ and $Z_{2,0}, \ldots, Z_{2,9}$, where precisely $Z_{2,1}$ and $Z_{1,3}$ are true and the others are false. For $i \in [p]$ and $b \in \{0, 1\}$, we have

$$O_{i,b} :- \underbrace{\bigvee_{\lfloor (n+m+b)/\beta \rfloor = 1} (X_{i,n} \wedge Y_{i,m})}_{\mathcal{O}(\beta)}, \qquad C_i :- \underbrace{\bigvee_{1 \le j \le i} \left(O_{j,0} \wedge \bigwedge_{j<k<i} \left(O_{k,1} \right) \right)}_{\mathcal{O}(p^2)}.$$

The predicates $O_{i,0}$ determine whether the sum of the digits in position i will result in a carry-over digit. The predicates $O_{i,1}$ determine whether the sum of the digits in position i will result in a carry-over digit *presuming that the sum of digits in position* $i-1$ has resulted in a carry-over digit. Finally, the predicates C_i determine whether a carry-over digit is created in position i taking into account the whole sum.

Therefore we can write rules for variables $S_{i,k}$ ($i \in [p+1]$ and $k \in [0; \beta - 1]$) that will represent the sum of $\mathbf{x}$ and $\mathbf{y}$ in one-hot representation. For $i = 1$ and $k \in [0; \beta - 1]$, we have

$$S_{1,k} :- \underbrace{\bigvee_{n+m \equiv k \pmod{\beta}} (X_{1,n} \wedge Y_{1,m})}_{\mathcal{O}(\beta)}$$

and for $i \in \{2, \ldots, p\}$, we have

$$S_{i,k} :- \underbrace{\bigvee_{n+m \equiv k \pmod{\beta}} (X_{i,n} \wedge Y_{i,m} \wedge \neg C_{i-1})}_{\mathcal{O}(\beta)} \vee \underbrace{\bigvee_{n+m+1 \equiv k \pmod{\beta}} (X_{i,n} \wedge Y_{i,m} \wedge C_{i-1})}_{\mathcal{O}(\beta)}.$$

For $i = p+1$, we have $S_{p+1,0} :- \neg C_p$, $S_{p+1,1} :- C_p$, and $S_{p+1,k} :- \bot$ for every $k \in [\beta - 1]$. After three iteration rounds, the values of predicates $S_{i,k}$ have been computed. The program could be timed by using one-hot counters and flags correctly to avoid unwanted values for the predicates in steps one and two, but this is trivial to add and does not affect the size and time complexity. We have $\mathcal{O}(p)$ predicates $O_{i,b}$ of size $\mathcal{O}(\beta)$, $\mathcal{O}(p)$ predicates C_i of size $\mathcal{O}(p^2)$ and $\mathcal{O}(p\beta)$ predicates $S_{i,k}$ of size $\mathcal{O}(\beta)$. The total size of the program is thus $\mathcal{O}(p\beta + p^3 + p\beta^2) = \mathcal{O}(p^3 + p\beta^2)$.

If $\mathbf{x}$ and $\mathbf{y}$ both have negative signs, we can use the same addition algorithm; the output simply includes a negative sign. If $\mathbf{x}$ and $\mathbf{y}$ have opposite signs, we need to use a subtraction algorithm instead. First, we need to compare $\mathbf{x}$ and $\mathbf{y}$ with signs omitted; in other words, we compare which number has a greater absolute value (this also determines the sign of the output). As stated before, this requires $\mathcal{O}(p\beta^2 + p^2)$ space and 2 iteration rounds. Then, we modify the addition algorithm above in the following way. Instead of adding digits together, we subtract them; the digits of the number with the smaller absolute value are subtracted from the digits of the number with the greater absolute value. If the subtraction of two digits goes below 0, it results in a negative carry -1. Otherwise the algorithm works in the same exact way, and thus adds nothing to the size and time complexity of the program. ◀

Parallel multiplication

In this section we introduce a parallel multiplication algorithm. The parallelization method is mostly well known and is based on splitting the multiplication into simple addition tasks.

▶ **Lemma 7.** *Given a base $\beta \in \mathbb{Z}$, $\beta \ge 2$, multiplication of any two numbers in $[0; \beta - 1]^p$ can be simulated with a (halting)* BNL*-program of size $\mathcal{O}(p^4 + p^3\beta^2 + p\beta^4)$ and computation time $\mathcal{O}(\log(p) + \log(\beta))$.*

Proof sketch. The formal explanations and examples are in [2]. Assume that we have two p-digit integers (we allow leading zeros, i.e. the leftmost digits can be zeros): a multiplicand $\mathbf{x}$ and a multiplier $\mathbf{y} = y_p \cdots y_1$ in an arbitrary base $\beta \in \mathbb{Z}$, $\beta \geq 2$. The parallel multiplication algorithm computes in the following two steps. **(1)** We run p different multiplications in parallel where the multiplicand $\mathbf{x}$ is multiplied by $y_i 0 \cdots 0$ with $i-1$ zeros on the right (for each $i \in [p]$ in base β). Each multiplication is actually also computed in parallel by using the parallel addition algorithm to obtain relatively small space and time complexities. As a result we obtain p different numbers of length $2p$. **(2)** We add the numbers obtained in the first step together in parallel using the parallel addition algorithm. ◀

3.2 Floating-point arithmetic

In this section we consider floating-point arithmetic, including polynomials and piecewise polynomial functions. We show that BNL-programs can simulate these in polynomial space and in polylogarithmic time, and some simple arithmetic operations even in constant time.

Floating-point system

A **floating-point number** in a system $S = (p, q, \beta)$ (where $p, q, \beta \in \mathbb{Z}_+$, $\beta \geq 2$) is a number that can be represented in the form $\pm 0.d_1 d_2 \cdots d_p \times \beta^{\pm e_1 \cdots e_q}$, where $d_i, e_i \in [0; \beta - 1]$. For such a number in system S, we call $f = 0.d_1 d_2 \ldots d_p$ the **fraction**, the dot between 0 and d_1 the **radix point**, p the **fraction precision**, $e = \pm e_1 \cdots e_q$ the **exponent**, q the **exponent precision** and β the **base** (or **radix**).

A floating-point number in a system S may have many different representations such as 0.10×10^1 and 0.01×10^2 which are both representations of the number 1. To ensure that our calculations are well defined, we desire a single form for all non-zero numbers. We say that a floating-point number (or more specifically, a floating-point representation) is **normalized**, if **1)** $d_1 \neq 0$, or **2)** $f = 0$, e is the smallest possible value and the sign of the fraction is $+$.

For a floating-point system $S = (p, q, \beta)$, we define an extended system of **raw floating-point numbers** $S^+(p', q')$ (where $p' \geq p$ and $q' \geq q$) that possess a representation of the form $\pm d_0.d_1 d_2 \cdots d_{p'} \times \beta^{\pm e_1 \cdots e_{q'}}$. When performing floating-point arithmetic, the precise outcomes of the calculations may be raw numbers, i.e., no longer in the same system as the operands strictly speaking. Therefore, in practical scenarios, we have $p' = \mathcal{O}(p)$ and $q' = \mathcal{O}(q)$. Consider, e.g., the numbers 99 and 2 which are both in the system $S = (2, 1, 10)$, but their sum 101 is not, because 3 digits are required to represent the fraction precisely. For this purpose, we must round numbers.

The easiest way to round a number is **truncation**, where the least significant digits of the number are simply omitted, rounding the number toward zero. On the other hand, the most common method is to round to the nearest floating-point number, with ties rounding to the number with an even least significant digit. This is called **round-to-nearest ties-to-even**.

Representing floating-point numbers in binary

Our way of representing floating-points of arbitrary base in binary is based on international standards (e.g. IEEE 754). Informally, if $\mathbf{b}$ represents a floating-point number in a system $S = (p, q, \beta)$, then the first two bits encode the signs of the exponent and fraction. The next $q\beta$ bits encode the exponent in base β, and the last $p\beta$ bits encode the fraction in base β.

Before we go into the details, we have to define simulation of functions that compute with floating-point numbers in a system $S = (p, q, \beta)$. Let $F = \pm f \times \beta^{\pm e}$ be a floating-point number in system S. Let $\mathbf{p}_1, \mathbf{p}_2 \in \{0,1\}$ and $\mathbf{s}_1, \ldots, \mathbf{s}_q, \mathbf{s}'_1, \ldots, \mathbf{s}'_p \in \{0,1\}^\beta$. We say that

$\mathbf{s} = \mathbf{p}_1\mathbf{p}_2\mathbf{s}_1 \cdots \mathbf{s}_q\mathbf{s}'_1 \cdots \mathbf{s}'_p$ **corresponds** to F (or $\mathbf{s}$ is a **one-hot representation** of F) if **(1)** $\mathbf{p}_1 = 1$ iff the sign of the exponent is $+$, **(2)** $\mathbf{p}_2 = 1$ iff the sign of the fraction is $+$, **(3)** $\mathbf{s}_1 \cdots \mathbf{s}_q$ corresponds to $e = e_1 \cdots e_q$, and **(4)** $\mathbf{s}'_1 \cdots \mathbf{s}'_p$ corresponds to $f = 0.d_1d_2 \cdots d_p$ (or, more precisely, to $d_1 \cdots d_p$). Correspondence is defined analogously for raw floating-point numbers; we simply replace p and q with p' and q', and add one more bit string $\mathbf{s}_0 \in \{0,1\}^\beta$ that must correspond to the digit d_0 to the left of the radix point. Likewise, we say that a bit string $\mathbf{s}$ **corresponds** to a sequence $(F_1, \ldots, F_k)$ of floating-point numbers if $\mathbf{s}$ is the concatenation of the bit strings that correspond to $F_1, \ldots, F_k$ from left to right. For example, in the system $S = (4, 3, 3)$ the number $-0.2001 \times 3^{+120}$ has the corresponding string

$$\underbrace{1}_{\mathbf{p}_1} \cdot \underbrace{0}_{\mathbf{p}_2} \cdot \underbrace{010 \cdot 001 \cdot 100}_{\mathbf{s}_1\mathbf{s}_2\mathbf{s}_3} \underbrace{001 \cdot 100 \cdot 100 \cdot 010}_{\mathbf{s}'_1\mathbf{s}'_2\mathbf{s}'_3\mathbf{s}'_4}.$$

▶ **Definition 8.** *Let $S = (p, q, \beta)$ be a floating-point system, and let $S^+ = (p', q')$ be a raw floating-point system. We say that a halting* BNL*-program Λ **simulates** a function $f\colon S^\ell \to S^k$ (or respectively $f\colon (S^+)^\ell \to S^k$) if the output $\Lambda(\mathbf{i}_1 \cdots \mathbf{i}_\ell)$ corresponds to $f(F_1, \ldots, F_\ell)$ for any $F_1, \ldots, F_\ell \in S$ (or resp. $F_1, \ldots, F_\ell \in S^+$) and the corresponding inputs $\mathbf{i}_1, \ldots, \mathbf{i}_\ell \in \{0,1\}^{2+\beta(p+q)}$ (or resp. $\mathbf{i}_1, \ldots, \mathbf{i}_\ell \in \{0,1\}^{2+\beta(p'+1+q')}$).*

Later when we construct programs for the floating-point operations, e.g. normalization, we will use a tool called **shifting**, which means moving each digit of a fraction to the left or right by one (e.g. shifting a fraction 0.012 once to the left leads to 0.120)

Consider a raw floating-point number $\pm 0.d_1d_2 \cdots d_{p'} \times \beta^{\pm e_1 \cdots e_{q'}}$ where $d_1 \neq 0$, which we seek to round to the system $S = (p, q, \beta)$ (where $p \leq p'$ and $q \leq q'$) using round-to-nearest ties-to-even. First, we check whether $e_1 \cdots e_{q'-q} = 0 \cdots 0$; if not, then we have exceeded the maximum exponent and output the highest or lowest possible number depending on the sign of the fraction. If yes, we set $e' = e_{q'-q+1} \cdots e_{q'}$. Next, we check the value of d_{p+1}. If $d_{p+1} < \frac{\beta}{2}$, then we let $f' = d_1 \cdots d_p$. If $d_{p+1} > \frac{\beta}{2}$, then we let $f' = d_1 \cdots d_p + 0^{p-1}1$ using integer addition. (We round to the nearest number in both cases.) If $d_{p+1} = \frac{\beta}{2}$, then we let $f' = d_1 \cdots d_p$ if d_p is even and $f' = d_1 \cdots d_p + 0 \cdots 01$ if d_p is odd. (In other words, in the case of a tie we round to the nearest number whose rightmost digit is even.) Finally, if f' has precision $p + 1$, then we must shift the fraction to the right and round again; otherwise we output $\pm 0.f' \times \beta^{\pm e'}$ where the signs are the same as before rounding. A BNL-program that carries out the rounding clearly takes as much space as integer addition for the fractions, i.e., $\mathcal{O}(p^3 + p\beta^2)$. Instead of integer addition, we could use a different method using carries that would result in size $\mathcal{O}(p^2\beta)$, but this does not affect our other results.

Normalizing a floating-point number

We informally describe how the normalization of raw floating-point numbers can be done. By normalization we mean that a raw floating-point number is normalized as described above.

▶ **Lemma 9.** *Let $S = (p, q, \beta)$ be a floating-point system. Normalization of a raw floating-point number in $S^+(p', q')$ to the floating-point system S, where $p' = \mathcal{O}(p)$ and $q' = \mathcal{O}(q)$, can be simulated with a (halting)* BNL*-program of size $\mathcal{O}(r^3 + r^2\beta^2)$ and computation time $\mathcal{O}(1)$, where $r = \max\{p, q\}$.*

Proof sketch. The full proof can be found in [2]. Let $S = (p, q, \beta)$ be a floating-point system. The normalization of a raw floating-point number $f \times \beta^e$ (we do not write down the signs here) in system $S^+(p', q')$ to the system S, where $p' = \mathcal{O}(p)$ and $q' = \mathcal{O}(q)$ can be split into the following cases.

1. If $f = 0$, we only set e to the smallest possible value and the sign of the fraction to $+$.
2. If $0 < |f| < 1$, then we can calculate in a few steps how much we have to shift the fraction to the left (and decrease the exponent).
3. If $|f| \geq 1$, we shift the fraction to the right by one (and decrease the exponent by one) and, after that, round the number to match fraction precision p. The rounding might lead to a non-normalized floating-point number, but we only have to shift the number to the right again at most once (because after rounding, $|f| \leq 1$).

The hard part is to keep the time complexity as low as possible. We do not go into the details here (full proofs are in [2]), but the main idea is to apply parallel integer addition specified in Section 3.1. ◀

Addition of floating-point numbers

In this section we show that we can simulate floating-point addition via BNL-programs, which is possible even in constant time.

▶ **Lemma 10.** *Addition of two (normalized) floating-point numbers in $S = (p, q, \beta)$ can be simulated with a (halting)* BNL*-program of size $\mathcal{O}(r^3 + r^2\beta^2)$ and computation time $\mathcal{O}(1)$, where $r = \max\{p, q\}$.*

Proof sketch. The full proof can be found in [2]. In the parts where we add or normalize numbers, we apply the results obtained in earlier sections. The addition is done in the following steps. **(1)** We compare which of the exponents is greater and store it. **(2)** We determine the difference d between the exponents. If d is greater than the length of the fractions, we are done and output the number with the greater exponent. If d is smaller than the length of the fractions, then we shift the fraction of the number with the smaller exponent to the right d times. We then perform integer addition on the fractions and store the result. **(3)** We obtain a number whose exponent was obtained in the first step and whose fraction was obtained in the second step. We normalize this number. ◀

Multiplication of floating-point numbers

In this section we show that we can simulate floating-point multiplication via BNL-programs. The multiplication requires logarithmic time, since the proof applies the result obtained for integer multiplication in Lemma 7.

▶ **Lemma 11.** *Multiplication of two (normalized) floating-point numbers in $S = (p, q, \beta)$ can be simulated with a (halting)* BNL*-program of size $\mathcal{O}(r^4 + r^3\beta^2 + r\beta^4)$ and computation time $\mathcal{O}(\log(r) + \log(\beta))$, where $r = \max\{p, q\}$.*

Proof sketch. The full proof can be found in [2]. Informally, we do the following.

1. We add the exponents together by using the parallel (integer) addition algorithm and store the result. If the result is less than the maximum exponent, we move to the next step. Otherwise, we are done and output the largest possible number, i.e. the number with the highest possible fraction and exponent in the system.
2. We multiply the fractions using the integer multiplication algorithm and store the product.
3. We obtain a number whose exponent was obtained in the first step and whose fraction was obtained in the second step. We normalize this number.

Applying the results of parallel (integer) addition, parallel (integer) multiplication, and normalization described in the previous sections, we obtain the wanted results. ◀

Floating-point polynomials and piecewise polynomial functions

Next we consider floating-point polynomials and activation functions that are piecewise polynomial. A **piecewise polynomial** function (with a single variable) is defined as separate polynomials over certain intervals of real numbers. For instance, the function "$f(x) = x^2$ when $x \geq 0$ and $f(x) = -x$ when $x < 0$" is piecewise polynomial; the intervals are the sets of non-negative and negative numbers and the attached polynomials are x^2 and $-x$. In a floating-point system, a piecewise polynomial function is an approximation, much like addition and multiplication. We perform approximations after each addition and multiplication; as a result, the calculations must be performed in some canonical order because the order of approximations will influence the result. By the number of pieces, we refer to the number of intervals that the piecewise polynomial function is defined over; our example above has 2 pieces. We obtain the following theorem.

▶ **Theorem 12.** *Assume we have a piecewise polynomial function $\alpha\colon S \to S$, where each polynomial is of the form $a_n x^n + \cdots + a_1 x + a_0$ where $n \in \mathbb{N}$, $a_i \in S = (p, q, \beta)$ for each $0 \leq i \leq n$ and $r = \max\{p, q\}$ (addition and multiplication approximated in S). Let Ω be the highest order of the polynomials (or 1 if the highest order is 0) and let $P \in \mathbb{Z}_+$ be the number of pieces. We can construct a* BNL*-program Λ that simulates $\alpha(x)$ such that*

1. *the size of Λ is $\mathcal{O}(P\Omega^2(r^4 + r^3\beta^2 + r\beta^4))$, and*
2. *the computation time of Λ is $\mathcal{O}((\log(\Omega) + 1)(\log(r) + \log(\beta)))$.*

Proof sketch. The full proof can be found in [2]. We obtain BNL-programs that simulate the polynomials in polynomial space and polylogarithmic time. When calculating a floating-point polynomial $a_n x^n + \cdots + a_1 x + a_0$, the order of calculations is as follows: Multiplications are handled first. When carrying out the multiplication $x_1 \cdot x_2 \cdot \ldots \cdot x_k$, we simultaneously calculate the products $y_1 = x_1 \cdot x_2$, $y_2 = x_3 \cdot x_4$, etc. (If k is an odd number, the multiplicand x_k has no pair. In this case we define $y_{(k+1)/2} = x_k$.) Then, in similar fashion we calculate the products $z_1 = y_1 \cdot y_2$, $z_2 = y_3 \cdot y_4$, etc. We continue this until we have calculated the whole product. After multiplications, we handle the sums in identical fashion. We obtain the wanted results by simulating the additions and multiplications of each polynomial as described in Lemmas 10 and 11. ◀

4 Descriptive complexity for general neural networks

In this section, we establish connections between Boolean network logic and neural networks. Informally, we define a general neural network as a weighted directed graph (with any topology) operating on floating-point numbers in some system S. Each node receives either a fixed initial value or an input as its first activation value. In each communication round a node sends its activation value to its neighbours and calculates a new activation value as follows. Each node multiplies the activation values of its neighbours with associated weights, adds them together with a node-specific bias and feeds the result into a node-specific activation function. Note that floating-point systems are bounded, and the input space of a neural network is thus finite.

Before specifying neural networks formally, we introduce some concepts for infinite sequences of floating-point numbers analogous to infinite sequences of bit strings. Let $k \in \mathbb{Z}_+$ and let $S = (p, q, \beta)$ be a floating-point system. Let F denote an infinite sequence $(\mathbf{f}_j)_{j\in\mathbb{N}}$ of k-floating-point strings $\mathbf{f}_j \in S^k$. Let $A \subseteq [k]$ and $P \subseteq [k]$ be subsets of positions called **attention** positions and **print** positions respectively. The sets A and P induce corresponding sequences $(\mathbf{a}_j)_{j\in\mathbb{N}}$ and $(\mathbf{p}_j)_{j\in\mathbb{N}}$ of substrings of the strings in F. More formally, $(\mathbf{a}_j)_{j\in\mathbb{N}}$

records the substrings with positions in A, and $(\mathbf{p}_j)_{j\in\mathbb{N}}$ records the substrings with positions in P. Next we define output conditions for F with respect to attention and print positions. Let $\mathbf{t} \in S^{|A|}$ denote a set of thresholds for each attention position. If at least one floating-point number in $\mathbf{a}_n$ exceeds the threshold in the same position in $\mathbf{t}$ (for some $n \in \mathbb{N}$), then we say that $\mathcal{F}$ **outputs** $\mathbf{p}_n$ in round n and that n is an **output round**. More precisely, F **outputs in round** n **with respect to** $(k, A, P, \mathbf{t})$, and $\mathbf{p}_n$ is the **output of** F **in round** n **with respect to** $(k, A, P, \mathbf{t})$. Let $O \subseteq \mathbb{N}$ be the set of output rounds; they induce a subsequence $(\mathbf{f}_j)_{j\in O}$ of F. We call the sequence $(\mathbf{p}_j)_{j\in O}$ the **output sequence** of F.

Next we define an output condition where output rounds are fixed by a set $O \subseteq \mathbb{N}$ and attention bits are excluded along with thresholds. We say that S **outputs** in rounds O (and also, in any particular round $n \in O$). Outputs and output sequences w.r.t. (k, O, P) are defined analogously.

4.1 General neural networks

Next we define neural networks formally. A **(directed) graph** is a tuple (V, E), where V is a finite set of **nodes** and $E \subseteq V \times V$ is a set of **edges**. Note that we allow self-loops on graphs, i.e. edges $(v, v) \in E$. A **general neural network** $\mathcal{N}$ (for floating-point system S) is defined as a tuple $(G, \mathfrak{a}, \mathfrak{b}, \mathfrak{w}, \pi)$, where $G = (V, E, <^V)$ is a directed graph associated with a linear order $<^V$ for nodes in V. The network $\mathcal{N}$ contains sets $I, O \subseteq V$ of **input** and **output** nodes respectively, and a set $H = V \setminus (I \cup O)$ of **hidden nodes**. The tuples $\mathfrak{a} = (\alpha_v)_{v\in V}$ and $\mathfrak{b} = (b_v)_{v\in V}$ are assignments of a piecewise polynomial **activation function** $\alpha_v \colon S \to S$ and a **bias** $b_v \in S$ for each node. Likewise, $\mathfrak{w} = (w_e)_{e\in E}$ is an assignment of a **weight** $w_e \in S$ for each edge. The function $\pi \colon (V \setminus I) \to S$ assigns an initial value to each non-input node.

The computation of a general neural network is defined with a given input function $i \colon I \to S$. Similar to BNL-programs, an input function i also induces a floating-point string $\mathbf{i} \in S^{|I|}$, and respectively a floating-point string induces an input function. **The state of the network at time** t is a function $g_t \colon V \to S$, which is defined recursively as follows. For $t = 0$, we have $g_0(v) = i(v)$ for input nodes and $g_0(v) = \pi(v)$ for non-input nodes. Now assume we have defined the state at time t. The state at time $t + 1$ is defined as follows:

$$g_{t+1}(v) = \alpha_v\Big(b_v + \sum_{(u,v)\in E} \big(g_t(u) \cdot w_{(u,v)}\big)\Big).$$

More specifically, the sum is unfolded from left to right according to the order $<^V$ of the nodes $u \in V$. For each piece of an activation function, we assume a normal form $a_n x^n + \cdots + a_1 x + a_0$, which designates the order of operations (as in the proof sketch of Theorem 12). If we designate that $u_1, \ldots, u_k$ enumerate the set O of output nodes in the order $<^V$, then the state of the system induces an output tuple $o_t = (g_t(u_1), \ldots, g_t(u_k))$ at time t for all $t \geq 0$.

We once again define two frameworks for designating output rounds, one machine-internal and one machine-external framework. In the first framework, the set V contains a set A of **attention nodes** u, each of which is associated with a *threshold* $s_u \in S$; the order of the nodes induces a threshold string $\mathbf{t} \in S^{|A|}$. In the second framework, attention nodes are excluded and the neural network is associated instead with an **attention function** $a \colon S^{|I|} \to \wp(\mathbb{N})$.

Next we define how the output rounds and output sequence are obtained. Let $v_1, \ldots, v_n$ enumerate the nodes of the neural network (in the order $<^V$). Let $i \colon I \to S$ be an input function that induces an input $\mathbf{i} \in S^{|I|}$. A neural network induces an infinite sequence $(\mathbf{s}_t)_{t\in\mathbb{N}}$ called the **network state sequence** (with input $\mathbf{i}$), where $\mathbf{s}_t = g_t(v_1) \cdots g_t(v_n)$. The set of

output nodes O corresponds to the set of print positions $\{\, i \mid v_i \in O \,\}$. If the network has attention nodes A, then A corresponds to the set of attention positions $\{\, i \mid v_i \in A \,\}$. If the network has an attention function a, then the output rounds are given by $a(\mathbf{i})$. Therefore, analogously to the general output conditions defined for infinite sequences of floating-point strings, a neural network with an input $\mathbf{i}$ also induces **output rounds** and an **output sequence** w.r.t. $(n, A, O, \mathbf{t})$ (or resp. w.r.t. $(n, a(\mathbf{i}), O)$).

We then define some parameters that will be important when describing how neural networks and BNL-programs are related in terms of space and time complexity. The in-degree of a node v is the number of nodes u such that there is an edge $(u, v) \in E$; we say that u is a **neighbour** of v. Note that we allow reflexive loops so a node might be its own "neighbour". The **degree** of a general neural network $\mathcal{N}$ is the maximum in-degree of the underlying graph. The **piece-size** of $\mathcal{N}$ is the maximum number of "pieces" across all its piecewise polynomial activation functions. The **order** of $\mathcal{N}$ is the highest order of a "piece" of its piecewise polynomial activation functions.

A general neural network can easily emulate typical *feedforward neural networks*. This requires that the graph of the general neural network is connected and acyclic, the sets I, O and H are chosen correctly and the graph topology is as required, with all paths from an input node to an output node being of the same length. Unlike in a classical feedforward neural network, the hidden and output nodes of a general neural network have an initial value, but they are erased as the calculations flow through the network, so this is an inconsequential, essentially syntactic phenomenon. The inputs are also erased in the same way, likewise an inconsequential syntactic phenomenon. Finally, there is a round t where the general neural network outputs the same values as a corresponding feedforward network would.

In general, our neural network models are *recurrent* in the sense that they allow loops. They are *one-to-many* networks, in other words, they can map each input to a sequence of outputs unlike feedforward neural networks which always map each input to a single output.

4.2 Equivalence and time series problems

In order to translate neural networks to BNL-programs and vice versa, we define time series problems for both floating-point numbers and binary numbers, and two types of corresponding equivalence relations. The reason for this is obvious, as BNL-programs operate with binary numbers and neural networks with floating-point numbers. Informally, in the below asynchronous equivalence means that the modeled time series can be repeated but with a delay between output rounds. The time delays in our results are not arbitrary but rather modest. Moreover, we do not fix the attention mechanism for the programs or neural networks, and our definitions work in both cases.

First we define notions for floating-points. Let $k, \ell \in \mathbb{N}$, $P \subseteq [k]$ and let $S = (p, q, \beta)$ be a floating-point system. We let $\mathcal{F}(k, P, S)$ denote the family of sequences $F = (\mathbf{f}_n)_{n\in\mathbb{N}}$ of k-strings $\mathbf{f}_n \in S^k$ of numbers in S with print position set P. A **(floating-point) time series problem $\mathfrak{P}$ for** (ℓ, k, P) **in** S is a function $\mathfrak{P}\colon S^\ell \to \mathcal{F}(k, P, S) \times \wp(\mathbb{N})$. With a given input $(F_1, \ldots, F_\ell) \in S^\ell$, $\mathfrak{P}$ gives a sequence $(\mathbf{f}_n)_{n\in\mathbb{N}} \in \mathcal{F}(k, P, S)$ and a subset $O \in \wp(\mathbb{N})$ and therefore $\mathfrak{P}$ **induces** the output sequence of $(\mathbf{f}_n)_{n\in\mathbb{N}}$ w.r.t. (k, O, P) (P induces a subsequence $(\mathbf{p}_n)_{n\in\mathbb{N}}$, and O further induces the output sequence $(\mathbf{p}_n)_{n\in O}$). Let Λ be a BNL-program with $(\beta(p+q)+2)|P|$ print predicates and $(\beta(p+q)+2)\ell$ input predicates. We say that Λ **simulates a solution** for time series problem $\mathfrak{P}$ if for every input $\mathbf{i} \in \{0,1\}^{(\beta(p+q)+2)\ell}$ corresponding to $(F_1, \ldots, F_\ell) \in S^\ell$, the output sequence of Λ with input $\mathbf{i}$ corresponds to the output sequence induced by $\mathfrak{P}(F_1, \ldots, F_\ell)$, i.e., the output strings of Λ correspond to the output strings of $\mathfrak{P}$. A neural network $\mathcal{N}$ with ℓ input nodes and $|P|$ output nodes

solves $\mathfrak{P}$ if the output sequence of $\mathcal{N}$ with input $(F_1, \dots, F_\ell)$ is the output sequence induced by $\mathfrak{P}(F_1, \dots, F_\ell)$. We say that a BNL-program Λ and a neural network $\mathcal{N}$ (for S) are **asynchronously equivalent in** S if the time series problems in S simulated by Λ are exactly the ones solved by $\mathcal{N}$.

We define notions for binaries in similar fashion. Recall that $k, \ell \in \mathbb{N}$, $P \subseteq [k]$. Similarly, let $\mathcal{S}(k, P)$ denote the family of k-bit string sequences $B = (\mathbf{b}_n)_{n\in\mathbb{N}}$ with print bit set P. A **(binary) time series problem** $\mathfrak{P}$ **for** (ℓ, k, P) is a function $\mathfrak{P}\colon \{0,1\}^\ell \to \mathcal{S}(k, P) \times \wp(\mathbb{N})$ that assigns a k-bit string sequence and a set $O \in \wp(\mathbb{N})$ of output rounds to every input $\mathbf{i} \in \{0,1\}^\ell$; together they **induce** an output sequence w.r.t. (k, O, P). We say that a BNL-program Λ with ℓ input predicates and $|P|$ print predicates **solves** $\mathfrak{P}$ if the output sequence of Λ with any input $\mathbf{i} \in \{0,1\}^\ell$ is the output sequence induced by $\mathfrak{P}(\mathbf{i})$. We say that a neural network $\mathcal{N}$ for floating point system S with ℓ input nodes and $|P|$ output nodes **simulates a solution** for binary time series problem $\mathfrak{P}$ if for every input $(F_1, \dots, F_\ell) \in S^\ell$ that represents $\mathbf{i} \in \{0,1\}^\ell$ (i.e. every F_i represents 0 or 1), the output sequence of $\mathcal{N}$ with input $(F_1, \dots, F_\ell)$ corresponds to the output sequence induced by $\mathfrak{P}(\mathbf{i})$ (in the same fashion, where every floating-point number represents 0 or 1). We say that a BNL-program Λ and a general neural network $\mathcal{N}$ are **asynchronously equivalent in binary** if the time series problems in binary simulated by $\mathcal{N}$ are exactly the ones solved by Λ.

We define the **computation delay** between two objects that are asynchronously equivalent (in binary or in floating-point system S) analogously to the computation delay defined in the preliminaries.

▶ Remark 13. Asynchronous equivalence in binary could be extended to two BNL-programs; this is consistent with the asynchronous equivalence defined in preliminaries. Therefore asynchronous equivalence in binary could also extend for SC and self-feeding circuits. We could also define equivalence between two neural networks. Informally, two neural networks are asynchronously equivalent if they solve exactly the same floating-point time series problems. It is also possible to define a weakened equivalence relation for neural networks, where a neural network simulates another neural network in "binary" as follows. Let $\mathfrak{P}$ be a floating-point time series problem for (ℓ, k, P) in $S = (p, q, \beta)$ and let $\mathfrak{P}'$ be a binary time series problem for $((\beta(p+q)+2)\ell, (\beta(p+q)+2)k, P')$, where P' is the set of bit positions which corresponds to positions in P. We say that $\mathfrak{P}'$ corresponds to $\mathfrak{P}$ if for each $\mathbf{f} \in S^\ell$ and the unique bit string $\mathbf{i} \in \{0,1\}^{(\beta(p+q)+2)\ell}$ that corresponds to $\mathbf{f}$, we have that the output sequence induced by $\mathfrak{P}'(\mathbf{i})$ corresponds to the one induced by $\mathfrak{P}(\mathbf{f})$. We say that neural networks $\mathcal{N}$ and $\mathcal{N}'$ are weakly asynchronously equivalent in S if the time series problems in S solved by $\mathcal{N}$ are exactly the ones with a corresponding binary time series problem simulated by $\mathcal{N}'$, or respectively the time series problems in S solved by $\mathcal{N}'$ are exactly the ones with a corresponding binary time series problem simulated by $\mathcal{N}$.

4.3 From NN to BNL

We provide a translation from general neural networks to Boolean network logic. The proof is based on the results obtained for floating-point arithmetic in Section 3.2.

▶ **Theorem 14.** *Given a general neural network $\mathcal{N}$ for $S = (p, q, \beta)$ with N nodes, degree Δ, piece-size P and order Ω (or $\Omega = 1$ if the order is 0), we can construct a* BNL*-program Λ such that $\mathcal{N}$ and Λ are asynchronously equivalent in S where for $r = \max\{p, q\}$,*

1. *the size of Λ is $\mathcal{O}(N(\Delta + P\Omega^2)(r^4 + r^3\beta^2 + r\beta^4))$, and*
2. *the computation delay of Λ is $\mathcal{O}((\log(\Omega) + 1)(\log(r) + \log(\beta)) + \log(\Delta))$.*

Proof. First we consider the framework where output rounds are defined by attention nodes and attention predicates. We consider the setting where output rounds are fixed as a corollary.

We use separate head predicates $S_{u,e}$, $S_{u,f}$, $E_{u,i,b}$, and $F_{u,j,b}$ ($i \in [q]$, $j \in [p]$, $b \in [0; \beta - 1]$) for each node u of $\mathcal{N}$. Together, they encode the **1)** exponent sign, **2)** fraction sign, **3)** exponent and **4)** fraction of the activation values of u in one-hot representation as described in Section 3.2. These calculations are done using the arithmetic algorithms from the same section. The program can not calculate a new activation value in one step like a neural network does, as each arithmetic operation takes some time to compute. The input of a single node is a floating-point number with q digits for the exponent, p digits for the fraction, and a sign for both. Its one-hot representation therefore has $(p+q)\beta + 2$ bits; exactly the number of head predicates assigned for each node. Each of these predicates receives a corresponding bit as input. For instance, if the input floating-point number of u is $-0.314 \times 10^{+01}$, then the head predicates $S_{u,e}$, $E_{u,1,0}$, $E_{u,2,1}$, $F_{u,1,3}$, $F_{u,2,1}$ and $F_{u,3,4}$ get the input 1 while all the other head predicates for u get the input 0.

After receiving these inputs, the rest of the program is built by applying the programs for floating-point addition and multiplication constructed in Section 3.2 to the aggregations and activation functions of each node in the established canonical order of operations. The calculations are timed with a one-hot counter, i.e., predicates $T_0, \ldots, T_n$ as described in Section 2.2. Here n is the worst-case number of rounds required for the algorithms to calculate an activation value for a node in the network (based on the number of neighbours, as well as the order and number of pieces of the activation function). The predicates in this counter are used to stall the head predicates for each node such that they receive the bits corresponding to the new activation values at the same time (this includes the print predicates, which are all the predicates corresponding to output nodes). The attention nodes have additional predicates that correspond to the threshold values; during rounds where the activation values have been calculated, an attention predicate turns true if this value is exceeded.

We compute additions and multiplications for each node in the network; this can be done simultaneously for each node. Each node requires at most Δ multiplications and additions in the aggregation before the use of the activation function. Multiplications can be done simultaneously and sums in parallel as described in section 3. These steps require size $\mathcal{O}(N\Delta(r^4 + r^3\beta^2 + r\beta^4))$ (each of the N nodes performs $\mathcal{O}(\Delta)$ multiplications/additions; the size of the multiplication is $\mathcal{O}(r^4 + r^3\beta^2 + r\beta^4)$ which dwarfs the addition size $\mathcal{O}(r^3 + r^2\beta^2)$) and the overall time required is $\mathcal{O}(\log(r) + \log(\beta)) + \mathcal{O}(\log(\Delta))$ (multiplication + addition).

After the aggregation come the activation functions. Since they are piecewise polynomial, we may apply Theorem 12, using the piece-size and order of the network. If $\Omega = 0$ we are done, so assume that $\Omega \in \mathbb{Z}_+$. Each of the N nodes calculates at most P polynomial pieces of order at most Ω, which gives us a size of $\mathcal{O}(NP\Omega^2(r^4 + r^3\beta^2 + r\beta^4))$. This requires only $\mathcal{O}((\log(\Omega) + 1)(\log(r) + \log(\beta)))$ time. The same predicates are used for the calculation of each subsequent global configuration of the network. Timing the calculations does not increase the size and time complexity. Adding the sizes and times together, the size of the program is $\mathcal{O}(N(\Delta + P\Omega^2)(r^4 + r^3\beta^2 + r\beta^4))$ and computing each global configuration of $\mathcal{N}$ requires time $\mathcal{O}(\log(r) + \log(\beta)) + \mathcal{O}(\log(\Delta)) + \mathcal{O}((\log(\Omega) + 1)(\log(r) + \log(\beta))) = \mathcal{O}((\log(\Omega) + 1)(\log(r) + \log(\beta)) + \log(\Delta))$; the first $\mathcal{O}(\log(r) + \log(\beta))$ is not dwarfed if $\Omega = 1$.

The case for the second framework, where output rounds are given by an attention function, is obtained as a corollary. We simply take the worst case time for calculating a new activation value with the aggregations and piecewise polynomial functions in BNL; let's say the worst case is T rounds. If $a \colon S^k \to \wp(\mathbb{N})$ is the attention function of $\mathcal{N}$, then the attention function of Λ is the function $a' \colon \{0,1\}^{k(\beta(p+q)+2)} \to \wp(\mathbb{N}), a'(\mathbf{i}) = Ta(\mathbf{i}') = \{Tn \mid n \in a(\mathbf{i}')\}$ where $\mathbf{i} \in \{0,1\}^{k(\beta(p+q)+2)}$ corresponds to $\mathbf{i}' \in S^k$. ◀

4.4 From BNL to NN

Before the formal translation from BNL-programs to general neural networks, we introduce two typical piecewise polynomial activation functions with just two pieces and order at most 1. These are the well-known rectified linear unit and the Heaviside step function. Recall that an activation function is a function $S \to S$, where S is a floating-point system. The **rectified linear unit** ReLU is defined by $\mathrm{ReLU}(x) = \max\{0, x\}$ and the **Heaviside step function** H is defined by $H(x) = 1$ if $x > 0$, and $H(x) = 0$, otherwise. It is easy to generalize our results for other activation functions.

▶ **Theorem 15.** *Given a* BNL*-program Λ of size s and depth d, we can construct a general neural network $\mathcal{N}$ for any floating-point system S with at most s nodes, degree at most* 2*,* ReLU *(or Heaviside) activation functions and computation delay $\mathcal{O}(d)$ (or $\mathcal{O}(s)$ since $s > d$) such that Λ and $\mathcal{N}$ are asynchronously equivalent in binary.*

Proof sketch. The full proof can be found in [2]. The aggregation each node performs on the activation values of its neighbours weakens neural networks in the sense that much of the information related to specific neighbours is lost. Due to this, a single node of a neural network can't imitate an arbitrary iteration clause where each predicate has a precise role. Instead, the program Λ is first turned into an asynchronously equivalent "fully-open" program Λ' that is described in [2]. Informally, that means each body of the iteration clauses of Λ' includes at most one logical connective. This is turned into a neural network by creating a node for each predicate of Λ'. The network only uses the floating-point numbers $-1, 0, 1, 2$, and the iteration clauses can all be calculated with ReLU or Heaviside by choosing the weights and biases appropriately. ◀

We have shown a translation from neural networks to BNL-programs and vice versa. Using the translations in succession, it is possible to transform a neural network into a weakly asynchronously equivalent neural network that only uses 1 and 0 as activation values, and either ReLU or Heaviside activation functions in every node. Generalizing our result for other activation functions is possible.

The match between BNL and neural networks provides a concrete demonstration of the obvious fact that – in some relevant sense – there is no difference between symbolic and non-symbolic approaches. Under reasonable background assumptions, non-symbolic approaches can be technically reduced to symbolic ones. More than to the differences between the symbolic and non-symbolic realms, the clear advantages of modern AI methods relate to the difference between systems based on explicit programming and systems that involve an aspect of learning not based on explicit and fully controlled programming steps.

5 Conclusion

We have shown a strong equivalence between a general class of one-to-many neural networks and Boolean network logic in terms of discrete time series. The translations are simple in both directions, with reasonable time and size blow-ups. We receive similar results for the logic SC due to Theorem 1 and self-feeding circuits due to Theorem 3. The link to self-feeding circuits is novel, since it allows us to apply circuit based methods to reason about neural networks in the recurrent setting. Interesting future directions involve investigating extensions with randomization as well as studying the effects of using alternatives to floating-point numbers, such as, for example, fixed-point arithmetic.

References

1 Veeti Ahvonen, Damian Heiman, Lauri Hella, and Antti Kuusisto. Descriptive complexity for distributed computing with circuits. In Jérôme Leroux, Sylvain Lombardy, and David Peleg, editors, *48th International Symposium on Mathematical Foundations of Computer Science, MFCS 2023, August 28 to September 1, 2023, Bordeaux, France*, volume 272 of *LIPIcs*, pages 9:1–9:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.MFCS.2023.9.

2 Veeti Ahvonen, Damian Heiman, and Antti Kuusisto. Descriptive complexity for neural networks via boolean networks. *CoRR*, abs/2308.06277, 2023. doi:10.48550/arXiv.2308.06277.

3 Pablo Barceló, Egor V. Kostylev, Mikaël Monet, Jorge Pérez, Juan L. Reutter, and Juan Pablo Silva. The logical expressiveness of graph neural networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

4 Maria Luisa Bonet and Samuel R. Buss. Size-depth tradeoffs for boolean formulae. *Information Processing Letters*, 49(3):151–155, 1994. doi:10.1016/0020-0190(94)90093-0.

5 Daizhan Cheng and Hongsheng Qi. A linear representation of dynamics of boolean networks. *IEEE Transactions on Automatic Control*, 55(10):2251–2258, 2010.

6 Martin Grohe. The logic of graph neural networks. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 – July 2, 2021*, pages 1–17. IEEE, 2021.

7 Martin Grohe. The descriptive complexity of graph neural networks, 2023. arXiv:2303.04613.

8 Lauri Hella, Matti Järvisalo, Antti Kuusisto, Juhana Laurinharju, Tuomo Lempiäinen, Kerkko Luosto, Jukka Suomela, and Jonni Virtema. Weak models of distributed computing, with connections to modal logic. In *Proceedings of the 2012 ACM Symposium on Principles of distributed computing*, pages 185–194, 2012.

9 Lauri Hella, Matti Järvisalo, Antti Kuusisto, Juhana Laurinharju, Tuomo Lempiäinen, Kerkko Luosto, Jukka Suomela, and Jonni Virtema. Weak models of distributed computing, with connections to modal logic. *Distributed Comput.*, 28(1):31–53, 2015.

10 Stuart Kauffman. Homeostasis and differentiation in random genetic control networks. *Nature*, 224(5215):177–178, 1969.

11 Antti Kuusisto. Modal Logic and Distributed Message Passing Automata. In *Computer Science Logic 2013 (CSL 2013)*, volume 23 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 452–468, 2013.

12 Leonid Libkin. *Elements of Finite Model Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004.

13 Fabian Reiter. Asynchronous distributed automata: A characterization of the modal mu-fragment. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 100:1–100:14, 2017.

14 Julian D Schwab, Silke D Kühlwein, Nensi Ikonomi, Michael Kühl, and Hans A Kestler. Concepts in boolean network modeling: What do they all mean? *Computational and structural biotechnology journal*, 18:571–582, 2020.

15 Massimiliano Zanin and Alexander N Pisarchik. Boolean networks for cryptography and secure communication. *Nonlinear Science Letters B: Chaos, Fractal and Synchronization. Vol*, 1(1):27–34, 2011.

16 Ranran Zhang, Mithun Vinod Shah, Jun Yang, Susan B Nyland, Xin Liu, Jong K Yun, Réka Albert, and Thomas P Loughran Jr. Network model of survival signaling in large granular lymphocyte leukemia. *Proceedings of the National Academy of Sciences*, 105(42):16308–16313, 2008.

Enumerating Error Bounded Polytime Algorithms Through Arithmetical Theories

Melissa Antonelli
Helsinki Institute for Information Technology, Finland

Ugo Dal Lago
Bologna University, Italy
Inria, Université Côte d'Azur, Sophia Antipolis, France

Davide Davoli
Inria, Université Côte d'Azur, Sophia Antipolis, France

Isabel Oitavem
Center for Mathematics and Applications (NOVA Math), NOVA FCT, Caparica, Portugal
Department of Mathematics, NOVA FCT, Caparica, Portugal

Paolo Pistone
Bologna University, Italy

Abstract

We consider a minimal extension of the language of arithmetic, such that the bounded formulas provably total in a suitably-defined theory *à la Buss* (expressed in this new language) precisely capture polytime *random* functions. Then, we provide two new characterizations of the semantic class **BPP** obtained by internalizing the error-bound check *within* a logical system: the first relies on measure-sensitive quantifiers, while the second is based on standard first-order quantification. This leads us to introduce a family of effectively enumerable subclasses of **BPP**, called $\mathbf{BPP}_{\mathrm{T}}$ and consisting of languages captured by those probabilistic Turing machines whose underlying error can be proved bounded in T. As a paradigmatic example of this approach, we establish that polynomial identity testing is in $\mathbf{BPP}_{\mathrm{T}}$, where $\mathrm{T} = \mathrm{I}\Delta_0 + \mathrm{Exp}$ is a well-studied theory based on bounded induction.

2012 ACM Subject Classification Theory of computation → Complexity theory and logic; Theory of computation → Proof theory

Keywords and phrases Bounded Arithmetic, Randomized Computation, Implicit Computational Complexity

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.10

Related Version *Extended Version*: `https://arxiv.org/abs/2311.15003` [1]

Funding The first, second, third and fifth authors' work is supported by the European Research Council through the project DIAPASoN ERC COoG 818616, and by the French "Agence Nationale de la Recherche" through the project PPS ANR-19-C48-0014. The first author's work is supported by the Helsinki Institute for Information Technology. The third author's work is supported by the French "Agence Nationale de la Recherche" through the project UCA DS4H ANR-17-EURE-0004. The fourth author's work is supported by national funds through the "FCT-Fundação para a Ciência e a Tecnologia, I.P.", through the projects UIDB/00297/2020 and UIDP/00297/2020 (Center for Mathematics and Applications).

1 Introduction

Since the early days of computer science, numerous and profound interactions with mathematical logic have emerged (think of the seminal works by Turing [55] and Church [9]). Among the sub-fields of computer science that have benefited the most from this dialogue, we should

certainly mention the theory of programming languages (e.g. through the Curry-Howard correspondence [17, 38, 54]), the theory of databases (e.g. through Codd's Theorem [11]) and computational complexity (e.g. through descriptive complexity [4, 39]). In particular, this last discipline deals with complexity classes [36, 10, 3], the nature of which still remains today, more than fifty years after the introduction of **P** and **NP** [12, 36], somewhat obscure.

The possibility of describing fundamental classes within the language of mathematical logic offered a better understanding of their nature: since the seventies [22, 15], but especially from the eighties and nineties [7, 32, 4, 39, 42], the logical characterization of several crucial classes has made it possible to consider them from a new viewpoint, less dependent on concrete machine models and explicit resource bounds. Characterizing complexity classes by way of a simple enough proof-of-recursion theoretical system also means being able to *enumerate* the problems belonging to them, and thus to devise sound and complete languages for the class, from which type systems and static analysis methodologies can be derived [37].

Among the various classes of problems considered in computational complexity, those defined on the basis of *randomized* algorithms [49] have appeared difficult to capture with the tools of logic. These include important and well-studied classes like **BPP** or **ZPP**. The former, in particular, is often considered as *the* class of feasible problems, and most complexity theorists conjecture that it actually coincides with **P**. One might thus expect it to be possible to obtain an enumeration of **BPP**, along the lines of the many examples known for classes like **P**, or even **PP** [18, 19]. However, by simply looking at its definition, **BPP** looks pretty different from **P**. Notably, the former, but not the latter, is an example of what is usually called a *semantic* class: the definition of **BPP** relies on algorithms which are both efficient and not *too erratic*: once an input is fixed, one of the two possible outputs must clearly prevail over the other; in other words, there is some fixed probability p, bounded away from $\frac{1}{2}$, such that, on any input x, the machine outputs some value $b_x \in \{0, 1\}$ with probability at least p. The existence of an effective enumerable family of algorithms deciding *all and only* the problems in **BPP** is still an open question.

In this paper we make a step towards a logical understanding of semantic complexity classes, and in particular of the logical and proof-theoretic complexity involved in keeping error-bounds under control. Our contributions can be divided in three parts. First, we generalize to the probabilistic setting the path indicated by *bounded arithmetic* [7, 24], a well-known approach to capture polynomial time algorithms, by extending usual arithmetical languages with a distinguished unary predicate $\mathtt{Flip}(x)$, playing the role of a source of randomness. We define a bounded theory $R\Sigma^b_1$-NIA as the randomized analogue of Buss' S^1_2 [7] and Ferreira's Σ^b_1-NIA [25], and show that the functions which can be proved total in $R\Sigma^b_1$-NIA are precisely the polytime *random* functions [53], i.e. those functions from strings to *distributions* of strings which can be computed by polytime probabilistic Turing machines (PTM, for short). Then, we move towards proper randomized classes by considering ways to keep the probability of error under control *from within* the logic. We first consider *measure quantifiers* [48, 46, 2], well-studied second-order quantifiers capable of measuring *the extent* to which a formula is true; we then show that these quantifiers, when applied to bounded formulas, can be encoded via *standard* first-order quantification. This way we obtain two characterizations of the problems in **BPP**, yet still semantic in nature: the error-bound check is translated into conditions which are not based on *provability* in some formal system, but rather on the *truth* of some formula in the standard model of first-order arithmetic.

While these results, which rely on semantic conditions, do not shed light on the enumeration problem for **BPP** directly, they set the conditions for a proof-theoretic investigation of this class: our last contribution is the introduction of a family of new *syntactic* subclasses of

BPP, each called $\mathbf{BPP}_{\mathrm{T}}$, and consisting of those languages for which the error-bounding condition is not only true, but also *provable* in some (non necessarily bounded) theory T. This reduces the enumeration problem to that of finding a recursively enumerable (r.e., for short) arithmetical theory T such that $\mathbf{BPP} = \mathbf{BPP}_{\mathrm{T}}$. To witness the difficulty of this problem, we show that the error-bounding condition is Π^0_1-*complete* and that establishing that **BPP** *cannot* be enumerated would be at least *as hard* as refuting the $\mathbf{BPP} = \mathbf{P}$ conjecture. At the same time, we show that *polynomial identity testing* (PIT), one of the few problems in **BPP** currently not known to be in **P** lies in $\mathbf{BPP}_{\mathrm{T}}$, where $\mathrm{T} = \mathrm{I}\Delta_0 + \mathrm{Exp}$ is a well-studied [35] sub-theory of PA, thus identifying an interesting and effectively enumerable subclass of **BPP**.

The main technical contributions of this paper can thus be summarized as follows:

- We introduce the arithmetical theory $R\Sigma^b_1$-NIA and prove that the random functions which are Σ^b_1-representable in it are precisely those which can be computed in polynomial time. The proof of the correspondence goes through the definition of a class of *oracle recursive* functions, called $\mathcal{POR}$, which is shown equivalent to the class of probabilistic polytime random functions **RFP**. The overall structure of the proof is described in Section 3, while further details can be found in the extended version of this paper [1].
- We exploit this result to obtain a new syntactic characterization of **PP** and, more interestingly, two semantic characterizations of **BPP**, the first based on measure quantifiers and the second relying on standard, first-order quantification. This is in Section 4.
- Finally, we introduce a family of syntactic subclasses $\mathbf{BPP}_{\mathrm{T}} \subseteq \mathbf{BPP}$ of *provable* **BPP**-problems, relative to a theory T. After showing that the property of being non-erratic is Π^0_1-complete, we establish that PIT is in $\mathbf{BPP}_{(\mathrm{I}\Delta_0+\mathrm{Exp})}$.We conclude by showing how our approach relates to existing works capturing **BPP** languages in bounded arithmetic [41]. All this can be found in Section 5 and Section 7.

Related Work. While a recursion-theoretic characterization of the syntactic class **PP** can be found in [18], most existing characterizations of **BPP** are based on some external, semantic condition [20, 47]. In particular, Eickmeyer and Grohe [21] provide a semantic characterization of **BPP** in a logic with fixed-point operators and a special counting quantifier, associated with a probabilistic semantics not too different from the quantitative interpretation we present in Section 3. On the other hand, [41] and [40] uses bounded arithmetic to provide characterizations of (both syntactic and semantic) randomized classes, such as **ZPP**, **RP** and **coRP**, and also provides a semantic characterization of **BPP**. An in-depth comparison is thus in order, and can be found in Section 7. Finally, [47] defines a higher-order language for polytime oracle recursive functions based on an adaptation of Bellantoni-Cook's safe recursion.

2 On the Enumeration of Complexity Classes

Before delving into the technical details, it is worth spending a few words on the problem of enumerating complexity classes, and on the reasons why it is more difficult for semantic classes than for syntactic ones.

First of all, it is worth observing that, although the distinction between syntactic and semantic classes appears in many popular textbooks (e.g. in [3, 50]), in the literature these notions are not defined in a precise way. Roughly speaking, syntactic classes are those which can be defined via limitations on the *amount of resources* (i.e. units of either time or space) that the underlying algorithm is allowed to use. Typical examples are the class **P** of problems

solvable in polynomial time and the class **PSPACE** of problems solvable in polynomial space. Instead, the definition of a semantic class usually requires, beyond some resource condition, an additional condition, sometimes called a *promise*, typically expressing that the underlying algorithm returns the correct answer *often enough*. A typical example here is the class **BPP** considered in this paper (cf. Definition 12), corresponding to problems solvable in polynomial time by probabilistic algorithms with some fixed error bound strictly smaller than $\frac{1}{2}$. Sometimes the distinction between syntactic and semantic classes may be subtle. For instance, as we discuss in Section 4, the class **PP**, whose definition also comprises a promise, is generally considered a syntactic class.

Notice that the sense of the terms "syntactic" and "semantic", when referred to complexity classes, is not clearly related to the sense that these terms have in mathematical logic. To a certain extent, the analysis that we develop in this paper with the tools of bounded arithmetic may help to clarify this point. On the one hand, well-known results in bounded arithmetic (cf. [7, 8]) provide a characterization of syntactic classes like **P** in terms of purely *proof-theoretic* conditions (i.e. provability in some weak fragment of Peano Arithmetic); on the other hand, we establish that, for a semantic class like **BPP**, an arithmetical characterization can be obtained by employing *both* proof-theoretic and *model-theoretic* conditions (i.e. truth in the standard model of Peano Arithmetic).

A natural question is whether such genuinely semantic (i.e. model-theoretic) conditions can somehow be eliminated in favor of purely syntactic (i.e. proof-theoretic) ones. In fact, this is a non-trivial problem, since, as proved in Section 5 (cf. Proposition 21), the promise underlying **BPP** is expressed by a Π^0_1-*complete* arithmetical formula. One should of course recall, however, that the distinction between semantic and syntactic classes refers to *how* a class is defined and not to the underlying set of problems. It is thus of *intensional* nature. In other words, even if **P** and **BPP** are defined in a different way, it could well be that someday we discover that $\mathbf{P} = \mathbf{BPP}$: in this case **BPP** would become a syntactic class, and, as we show in Section 5 (cf. Proposition 20), a purely proof-theoretic characterization of **BPP** would be available.

The problem of showing that a complexity class can be enumerated (i.e. that one can devise a recursive enumeration of, say, Turing Machines solving all and only the problems in the class) provides a different, and useful, angle to look at the distinction between syntactic and semantic classes. Ordinary syntactic classes, such as **P**, **PP**, and **PSPACE**, are quite simple to enumerate. While verifying resource bounds for *arbitrary* programs is very difficult, it is surprisingly easy to define an enumeration of resource bounded algorithms containing at least *one* algorithm for any problem in one of the aforementioned classes. To clarify what we mean, suppose we want to characterize the class **P**. On the one hand, the class of *all* algorithms working in polynomial time is recursion-theoretically very hard, actually Σ^0_2-complete [34]. On the other hand, the class of those programs consisting of a `for` loop executed a polynomial number of times, whose body itself consists of conditionals and simple enough instructions manipulating string variables, is both trivial to enumerate and big enough to characterize **P**, at least in an extensional sense: every problem in this class is decided by *at least one* program in the class and every algorithm in this class works in polytime. Many characterizations of **P** (and of other syntactic classes), as those based on safe-recursion [4, 45], light and soft linear logic [31, 30, 44], and bounded arithmetic [7], can be seen as instances of the just described pattern, where the precise class of polytime *programs* varies, while the underlying class of *problems* remains unchanged.

But what about semantic classes? Being resource bounded is not sufficient for an algorithm to solve a problem in some semantic class, since there can well be algorithms getting it wrong too often. For instance, it may well be that some probabilistic Turing Machine running in

polynomial time does not solve *any* problem in **BPP**. For this reason, unfortunately, the enumeration strategy sketched above does not seem to be readily applicable to semantic classes. How can we isolate a simple enough subclass of algorithms – which are not only resource bounded, but also not too erratic – at the same time saturating the class?

We think that the results in this paper, concerning proof-theoretic and model-theoretic characterizations of probabilistic complexity classes, may provide new insights on the nature of this problem, without giving a definite answer. Indeed, observe that the existence of a purely proof-theoretic characterization of some complexity class $\mathcal{C}$ via some recursively enumerable theory T directly leads to providing an enumeration of $\mathcal{C}$ (by enumerating the theorems of T). In this way, the problem of enumerating a semantic class $\mathcal{C}$ is directly related to the existence of some strong enough theory T.

In the following sections we do *not* prove **BPP** to be effectively enumerable, which is still out of reach. On the one hand we show that proving the non-enumerability of **BPP** is *as hard as* proving that **P** is different from **BPP**. On the other hand, we show that there exist subclasses of **BPP** which are large enough to include interesting problems in **BPP** and still "syntactic enough" to be effectively enumerable via some arithmetical theory.

3 Bounded Arithmetic and Polytime Random Functions

In this section we discuss our first result, namely, the characterization of polytime random functions via bounded arithmetic.

3.1 From Arithmetic to Randomized Computation, Subrecursively

We introduce the two main ingredients on which our characterization of polytime random functions relies: a randomized bounded theory of arithmetic $R\Sigma^b_1$-NIA, and a Cobham-style function algebra for polytime oracle recursive functions, called $\mathcal{POR}$.

Recursive Functions and Arithmetical Formulas. The study of so-called bounded theories of arithmetic, i.e. subsystems of PA in which only *bounded quantifications* are admitted, initiated by Parikh and Buss, has led to characterize several complexity classes [51, 14, 7, 8, 24, 43]. At the core of these characterizations lies the well-known fact (dating back to Gödel's [33]) that recursive functions can be *represented* in PA by means of Σ^0_1-formulas, i.e. formulas of the form $\exists x_1.\ldots.\exists x_n.A$, where A is a bounded formula. For example, the formula

$$A(x_1, x_2, y) := \exists x_3.x_1 \times x_2 = x_3 \wedge y = succ(x_3)$$

represents the function $f(x_1, x_2) = (x_1 \times x_2) + 1$. Indeed, in PA one can prove that $\forall x_1.\forall x_2.\exists!y.A(x_1, x_2, y)$, namely that A expresses a *functional* relation, and check that for all $n_1, n_2, m \in \mathbb{N}$, $A(\overline{n_1}, \overline{n_2}, \overline{m})$ holds (in the standard model $\mathcal{N}$) precisely when $m = f(n_1, n_2)$. Buss' intuition was then that, by considering theories *weaker* than PA, it becomes possible to capture functions computable within given resource bounds [7, 8].

In order to extend this approach to classes of *random* computable functions, we rely on a simple correspondence between first-order predicates over natural numbers and *oracles* from the Cantor space $\{0,1\}^{\mathbb{N}}$, following [2]. Indeed, suppose the aforementioned recursive function f has now the ability to observe (part of) an infinite sequence of bits. For instance, f might observe the first bit and return $(x_1 \times x_2) + 1$ if this is 0, and return 0 otherwise. Our idea is that we can capture the call by f to the oracle by adding to the standard language of PA a new unary predicate $\texttt{Flip}(x)$, to be interpreted as a stream of (random) bits. Our function f can then be represented by the following formula:

$$B(x_1, x_2, y) := \big(\texttt{Flip}(\overline{0}) \wedge \exists x_3.x_1 \times x_2 = x_3 \wedge y = succ(x_3)\big) \vee \big(\neg\texttt{Flip}(\overline{0}) \wedge y = \overline{0}\big).$$

As in the case above, it is possible to prove that $B(x_1, x_2, y)$ is functional, that is, that $\forall x_1.\forall x_2.\exists! y.B(x_1, x_2, y)$. However, since B now contains the unary predicate symbol $\texttt{Flip}(x)$, the actual numerical function that B represents depends on the choice of an interpretation for $\texttt{Flip}(x)$, i.e. on the choice of an oracle for f.

In the rest of this section we develop this idea in detail, establishing a correspondence between polytime random functions and a class of *oracle-recursive* functions which are provably total in a suitable bounded theory relying on the predicate `Flip`.

The Language $\mathcal{RL}$. We let $\mathbb{B} := \{0, 1\}$, $\mathbb{S} := \mathbb{B}^*$ indicate the set of finite words from $\mathbb{B}$, and $\mathbb{O} := \mathbb{B}^{\mathbb{S}}$. We introduce a language for first-order arithmetic incorporating the new predicate symbol $\texttt{Flip}(x)$ and its interpretation in the standard model. Following [26], we consider a first-order signature for natural numbers *in binary notation*. Consistently, formulas will be interpreted over $\mathbb{S}$ rather than $\mathbb{N}$. Working with strings is not essential and all results below could be spelled out in a language for natural numbers. Indeed, bounded theories may be formulated in both ways equivalently, e.g. Ferreira's Σ^b_1-NIA and Buss' S^1_2 [26].

▶ **Definition 1.** *The terms and formulas of $\mathcal{RL}$ are defined by the grammars below:*

$$t, s ::= x \mid \epsilon \mid \texttt{0} \mid \texttt{1} \mid t \frown s \mid t \times s$$
$$F, G ::= \texttt{Flip}(t) \mid t = s \mid t \subseteq s \mid \neg F \mid F \wedge G \mid F \vee G \mid \exists x.F \mid \forall x.F.$$

The function symbol $\frown$ stands for string concatenation, while $t \times u$ indicates the concatenation of t with itself a number of times corresponding to the length of u. The binary predicate $\subseteq$ stands for the initial substring relation. As usual, we let $A \to B := \neg A \vee B$.

We adopt the following abbreviations: ts for $t \frown s$; $\texttt{1}^t$ for $\texttt{1} \times t$; $t \preceq s$ for $\texttt{1}^t \subseteq \texttt{1}^s$, i.e. the length of t is smaller than that of s; $t|_r = s$ for $(\texttt{1}^r \subseteq \texttt{1}^t \wedge s \subseteq t \wedge \texttt{1}^r = \texttt{1}^s) \vee (\texttt{1}^t \subseteq \texttt{1}^r \wedge s = t)$, i.e s is the *truncation* of t at the length of r. For each string $\sigma \in \mathbb{S}$, we let $\overline{\sigma}$ be the term of $\mathcal{RL}$ representing it (e.g. $\overline{\epsilon} = \epsilon$, $\overline{\sigma 0} = \overline{\sigma}\texttt{0}$ and $\overline{\sigma 1} = \overline{\sigma}\texttt{1}$).

As for standard bounded arithmetics [7, 23], a defining feature of our theory is the focus on so-called *bounded quantification*. In $\mathcal{RL}$, *bounded quantifications* are of the forms $\forall x.\texttt{1}^x \subseteq \texttt{1}^t \to F$ and $\exists x.\texttt{1}^x \subseteq \texttt{1}^t \wedge F$, abbreviated as $\forall x \preceq t.F$ and $\exists x \preceq t.F$. Following [23], we adopt *subword quantifications* as those quantifications of the forms $\forall x.(\exists w \subseteq t.wx \subseteq t) \to F$ and $\exists x.\exists w \subseteq t.wx \subseteq t \wedge F$, abbreviated as $\forall x \subseteq^* t.F$ and $\exists x \subseteq^* t.F$. An $\mathcal{RL}$-formula F is said to be a Σ^b_1*-formula* if it is of the form $\exists x_1 \preceq t_1 \ldots . \exists x_n \preceq t_n.G$, where the only quantifications in G are subword ones. The distinction between bounded and subword quantifications is relevant for complexity reasons: if $\sigma \in \mathbb{S}$ is a string of length k, the witness of a subword existentially quantified formula $\exists y.y \subseteq^* \overline{\sigma} \wedge H$ is to be looked for among all possible *sub-strings* of σ, i.e. within a space of size $\mathcal{O}(k^2)$, while the witness of a bounded formula $\exists y \preceq \overline{\sigma}.H$ is to be looked for among all possible strings *of length* k, i.e. within a space of size $\mathcal{O}(2^k)$.

The Borel Semantics of $\mathcal{RL}$. We introduce a *quantitative* semantics for formulas of $\mathcal{RL}$, inspired by the one introduced in [2]. While the function symbols of $\mathcal{RL}$, as well as the predicate symbols "=" and "$\subseteq$", have a standard interpretation as relations over $\mathbb{S}$, the idea is that the predicate symbol `Flip` may stand for *an arbitrary* subset of $\mathbb{S}$, that is, an arbitrarily chosen $\omega \in \mathbb{O}$. For this reason, we take as the interpretation of a $\mathcal{RL}$-formula

F the set $[\![F]\!] \subseteq \mathbb{O}$ of *all* possible interpretations of `Flip` satisfying F. Importantly, such sets $[\![F]\!]$ can be proved to be *measurable*, a fact that will turn out essential in Section 4. Indeed, the canonical first-order model of $\mathcal{RL}$ over $\mathbb{S}$ can be extended to a probability space $(\mathbb{O}, \sigma(\mathcal{C}), \mu)$ defined in a standard way: here $\sigma(\mathcal{C}) \subseteq \wp(\mathbb{O})$ is the Borel σ-algebra generated by *cylinders* $\mathsf{C}^b_\sigma = \{\omega \mid \omega(\sigma) = b\}$, with $b \in \{0,1\}$, and μ is the *unique* measure such that $\mu(\mathsf{C}^b_\sigma) = \frac{1}{2}$ (see [5]). While the interpretation of terms is standard, the interpretation of formulas is defined below.

▶ **Definition 2** (Borel Semantics of $\mathcal{RL}$). *Given a term t, a formula F and an environment $\xi : \mathcal{G} \to \mathbb{S}$, where $\mathcal{G}$ is the set of term variables, the* interpretation of F under ξ *is the measurable set $[\![F]\!]_\xi \in \sigma(\mathcal{C})$ inductively defined as follows:*

$$[\![t = s]\!]_\xi := \begin{cases} \mathbb{O} & \text{if } [\![t]\!]_\xi = [\![s]\!]_\xi \\ \emptyset & \text{otherwise} \end{cases} \qquad [\![\mathtt{Flip}(t)]\!]_\xi := \{\omega \mid \omega([\![t]\!]_\xi) = 1\} \qquad [\![\exists x.G]\!]_\xi := \bigcup_{i \in \mathbb{S}} [\![G]\!]_{\xi\{x \leftarrow i\}}$$

$$[\![t \subseteq s]\!]_\xi := \begin{cases} \mathbb{O} & \text{if } [\![t]\!]_\xi \subseteq [\![s]\!]_\xi \\ \emptyset & \text{otherwise} \end{cases} \qquad \begin{array}{l} [\![\neg G]\!]_\xi := \mathbb{O} - [\![G]\!]_\xi \\ [\![G \vee H]\!]_\xi := [\![G]\!]_\xi \cup [\![H]\!]_\xi \\ [\![G \wedge H]\!]_\xi := [\![G]\!]_\xi \cap [\![H]\!]_\xi \end{array} \qquad [\![\forall x.G]\!]_\xi := \bigcap_{i \in \mathbb{S}} [\![G]\!]_{\xi\{x \leftarrow i\}}.$$

This semantics is well-defined as the sets $[\![\mathtt{Flip}(t)]\!]_\xi$, $[\![t = s]\!]_\xi$ and $[\![t \subseteq s]\!]_\xi$ are measurable and measurability is preserved by all the logical operators.

Observe that an interpretation of the language $\mathcal{RL}$, in the usual first-order sense, requires some ξ as above as well as an interpretation ω for $\mathtt{Flip}(x)$. One can easily check by induction that, for any formula F and interpretation ξ, $\omega \in [\![F]\!]_\xi$ precisely when F is satisfied in the first-order environment formed by ξ *and* ω.

The Bounded Theory $R\Sigma^b_1$-NIA. We now introduce a bounded theory in the language $\mathcal{RL}$, called $R\Sigma^b_1$-NIA, which can be seen as a probabilistic counterpart to Ferreira's Σ^b_1-NIA [25]. The theory $R\Sigma^b_1$-NIA is defined by axioms belonging to two classes:

- *Basic axioms* (where $\mathtt{b} \in \{\mathtt{0}, \mathtt{1}\}$):

$$x\epsilon = x \qquad x \times \epsilon = \epsilon \qquad x \subseteq \epsilon \leftrightarrow x = \epsilon \qquad x\mathtt{b} = y\mathtt{b} \to x = y$$
$$x(y\mathtt{b}) = (xy)\mathtt{b} \qquad x \times y\mathtt{b} = (x \times y)x \qquad x \subseteq y\mathtt{b} \leftrightarrow x \subseteq y \vee x = y\mathtt{b} \qquad x\mathtt{0} \neq y\mathtt{1} \qquad x\mathtt{b} \neq \epsilon.$$

- *Axiom schema for induction on notation*: $B(\epsilon) \wedge \forall x.\big(B(x) \to B(x\mathtt{0}) \wedge B(x\mathtt{1})\big) \to \forall x.B(x)$, where B is a Σ^b_1-formula in $\mathcal{RL}$.

The axiom schema for induction on notation adapts the usual induction schema of PA to the binary representation. As standard in bound arithmetic, restriction to Σ^b_1-formulas, is essential to characterize algorithms with *bounded* resources. Indeed, more general instances of this schema would lead to represent functions which are not polytime computable.

An Algebra of Polytime Oracle Recursive Functions. We now introduce a Cobham-style function algebra, called $\mathcal{POR}$, for polytime *oracle* recursive functions, and show that it is captured by a class of bounded formulas provably representable in the theory $R\Sigma^b_1$-NIA. This algebra is inspired by Ferreira's PTCA [23, 25]. Yet, a fundamental difference is that the functions we define are of the form $f : \mathbb{S}^k \times \mathbb{O} \to \mathbb{S}$, i.e. they carry an additional argument $\omega : \mathbb{S} \to \mathbb{B}$, to be interpreted as the underlying stream of random bits. Furthermore, our class includes the basic *query* function, which can be used to observe any bit from ω.

The *class* $\mathcal{POR}$ is the smallest class of functions from $\mathbb{S}^k \times \mathbb{O}$ to $\mathbb{S}$, containing the *empty* function $E(x, \omega) = \epsilon$, the *projection* functions $P^n_i(x_1, \ldots, x_n, \omega) = x_i$, the *word-successor* function $S_b(x, \omega) = xb$, the *conditional* function $C(\epsilon, y, z_0, z_1, \omega) = y$ and $C(xb, y, z_0, z_1, \omega) = z_b$, where $b \in \mathbb{B}$ (corresponding to $b \in \{0,1\}$), the *query* function $Q(x, \omega) = \omega(x)$, and closed under the following schemata:

- *Composition*, where f is defined from $g, h_1, \ldots, h_k$ as $f(\vec{x}, \omega) = g(h_1(\vec{x}, \omega), \ldots, h_k(\vec{x}, \omega), \omega)$.
- *Bounded recursion on notation*, where f is defined from g, h_0, h_1 as

$$\begin{aligned} f(\vec{x}, \epsilon, \omega) &= g(\vec{x}, \omega) \\ f(\vec{x}, y0, \omega) &= h_0\big(\vec{x}, y, f(\vec{x}, y, \omega), \omega\big)|_{t(\vec{x},y)} \\ f(\vec{x}, y1, \omega) &= h_1\big(\vec{x}, y, f(\vec{x}, y, \omega), \omega\big)|_{t(\vec{x},y)}, \end{aligned}$$

  with t obtained from $\epsilon, 0, 1, \frown, \times$ by explicit definition, i.e. by applying $\frown$ and $\times$ on constants ϵ, 0, 1, and variables $\vec{x}$ and y.

We now show that functions of $\mathcal{POR}$ are precisely those which are Σ_1^b-representable in $R\Sigma_1^b$-NIA. To do so, we slightly modify Buss' representability conditions by adding a constraint relating the quantitative semantics of formulas in $\mathcal{RL}$ and the additional functional parameter ω of oracle recursive functions.

▶ **Definition 3.** *A function $f : \mathbb{S}^k \times \mathbb{O} \to \mathbb{S}$ is* Σ_1^b-representable in $R\Sigma_1^b$-NIA *if there exists a Σ_1^b-formula $G(\vec{x}, y)$ of $\mathcal{RL}$ such that:*

1. *$R\Sigma_1^b$-NIA $\vdash \forall \vec{x}.\exists! y.G(\vec{x}, y)$,*
2. *for all $\sigma_1, \ldots, \sigma_k, \tau \in \mathbb{S}$ and $\omega \in \mathbb{O}$, $f(\sigma_1, \ldots, \sigma_k, \omega) = \tau$ iff $\omega \in [\![G(\overline{\sigma_1}, \ldots \overline{\sigma_k}, \overline{\tau})]\!]$.*

Condition 1. above does *not* say that the unique value y is obtained as a function of $\vec{x}$ *only*. Indeed, the truth-value of a formula depends both on the value of its first-order variables and on the value assigned to the random predicate `Flip`. Hence this condition says that y is uniquely determined as a function *both* of its first-order inputs and of an oracle from $\mathbb{O}$, precisely as functions of $\mathcal{POR}$.

▶ **Theorem 4.** *For any $f : \mathbb{S}^k \times \mathbb{O} \to \mathbb{S}$, f is Σ_1^b-representable in $R\Sigma_1^b$-NIA iff $f \in \mathcal{POR}$.*

Proof sketch. ($\Leftarrow$) The desired Σ_1^b-formula is constructed by induction on the structure of oracle recursive functions. Observe that the formula $\forall \vec{x}.\exists! y.G(\vec{x}, y)$ occurring in Condition 1. of Definition 3 is *not* Σ_1^b, since it is universally quantified while the existential quantifier is not bounded. Hence, in order to apply the inductive steps (corresponding to functions defined by composition and bounded recursion on notation), we need to adapt Parikh's theorem [51] (which holds for S_2^1 and Σ_1^b-NIA) to $R\Sigma_1^b$-NIA, to state that if $R\Sigma_1^b$-NIA $\vdash \forall \vec{x}.\exists y.G(\vec{x}, y)$, where $G(\vec{x}, y)$ is a Σ_1^b-formula, then we can find a term t such that $R\Sigma_1^b$-NIA $\vdash \forall \vec{x}.\exists y \preceq t.G(\vec{x}, y)$. ($\Rightarrow$) The proof consists in adapting Cook and Urquhart's argument for system IPV$^\omega$ [13], and this goes through a *realizability interpretation* of the intuitionistic version of $R\Sigma_1^b$-NIA, called $IR\Sigma_1^b$-NIA. Further details can be found in the extended version of this paper [1]. ◀

3.2 Characterizing Polytime Random Functions

Theorem 4 shows that it is possible to characterize $\mathcal{POR}$ by means of a system of bounded arithmetic. Yet, this is not enough to deal with classes, like **BPP** or **RP**, which are defined in terms of functions computed by PTMs. Observe that there is a crucial difference in the way in which probabilistic machines and oracle recursive functions access randomness, so our next goal is to fill the gap, by relating these classes of functions.

Let $\mathbb{D}(\mathbb{S})$ indicate the set of *distributions over* $\mathbb{S}$, that is, those functions $\lambda : \mathbb{S} \to [0, 1]$ such that $\sum_{\sigma \in \mathbb{S}} \lambda(\sigma) = 1$. By a *random function* we mean a function of the form $f : \mathbb{S}^k \to \mathbb{D}(\mathbb{S})$. Observe that any (polytime) PTM $\mathcal{M}$ computes a random function $f_{\mathcal{M}}$, where, for every $\sigma_1, \ldots, \sigma_k, \tau \in \mathbb{S}$, $f_{\mathcal{M}}(\sigma_1, \ldots, \sigma_k)(\tau)$ coincides with the probability that $\mathcal{M}(\sigma_1 \sharp \ldots \sharp \sigma_k) \Downarrow \tau$. However, a random function needs not be computed by a PTM in general. We define the following class of *polytime random functions*:

▶ **Definition 5** (Class **RFP**). *The class* **RFP** *is made of all random functions* $f : \mathbb{S}^k \to \mathbb{D}(\mathbb{S})$ *such that* $f = f_{\mathcal{M}}$*, for some PTM* $\mathcal{M}$ *running in polynomial time.*

Functions of **RFP** are closed under *monadic composition* $\diamond$, where $(g \diamond f)(\sigma)(\tau) = \sum_{\rho \in \mathbb{S}} g(\rho)(\tau) \cdot f(\sigma)(\rho)$ (one can check $f_{\mathcal{N}} \diamond f_{\mathcal{M}} = f_{\mathcal{N} \circ \mathcal{M}}$, where $\circ$ indicates PTM composition).

Since functions of **RFP** have a different shape from those of $\mathcal{POR}$, we must adapt the notion of Σ_1^b-representability for them, relying on the fact that any closed $\mathcal{RL}$-formula F generates a *measurable* set $[\![F]\!] \subseteq \mathbb{B}^{\mathbb{N}}$.

▶ **Definition 6.** *A function* $f : \mathbb{S}^k \to \mathbb{D}(\mathbb{S})$ *is* Σ_1^b-representable in $R\Sigma_1^b$-NIA *if there exists a* Σ_1^b*-formula* $G(\vec{x}, y)$ *of* $\mathcal{RL}$ *such that:*

1. $R\Sigma_1^b$*-NIA* $\vdash \forall \vec{x}.\exists! y.G(\vec{x}, y)$*,*
2. *for all* $\sigma_1, \ldots, \sigma_k, \tau \in \mathbb{S}$*,* $f(\sigma_1, \ldots, \sigma_k, \tau) = \mu\big([\![G(\overline{\sigma_1}, \ldots, \overline{\sigma}_k, \overline{\tau})]\!]\big)$*.*

Notice that any Σ_1^b-formula $G(\vec{x}, y)$ satisfying Condition 1. from Definition 6 actually defines a random function $\langle G \rangle : \mathbb{S} \to \mathbb{D}(\mathbb{S})$ given by $\langle G \rangle(\vec{\sigma})(\tau) = \mu([\![G(\overline{\vec{\sigma}}, \overline{\tau})]\!])$, where $\langle G \rangle$ is Σ_1^b-represented by G. Moreover, if G represents some $f \in \mathbf{RFP}$, then $f = \langle G \rangle$. In analogy with Theorem 4, we can now prove the following result:

▶ **Theorem 7.** *For any* $f : \mathbb{S}^k \to \mathbb{D}(\mathbb{S})$*,* f *is* Σ_1^b*-representable in* $R\Sigma_1^b$*-NIA iff* $f \in \mathbf{RFP}$*.*

Thanks to Theorem 4, the proof of the result above simply consists in showing that $\mathcal{POR}$ and **RFP** can be related as stated below.

▶ **Lemma 8.** *For all functions* $f : \mathbb{S}^k \times \mathbb{O} \to \mathbb{S}$ *in* $\mathcal{POR}$ *there exists* $g : \mathbb{S}^k \to \mathbb{D}(\mathbb{S})$ *in* **RFP** *such that for all* $\sigma_1, \ldots, \sigma_k, \tau \in \mathbb{S}$*,* $\mu(\{\omega \mid f(\vec{\sigma}, \omega) = \tau\}) = g(\sigma_1, \ldots, \sigma_k, \tau)$*, and conversely.*

Proof sketch. The first step of our proof consists in replacing the class **RFP** by an intermediate class **SFP** corresponding to functions computed by polytime *stream Turing machines* (STM, for short). These are defined as deterministic TM with one extra read-only tape: at the beginning of the computation the extra tape is sampled from $\mathbb{B}^{\mathbb{N}}$, and at each computation step the machine reads one new bit from this tape. Then we show that for any function $f : \mathbb{S}^k \to \mathbb{D}(\mathbb{S})$ computed by some polytime PTM there is a function $g : \mathbb{S}^k \times \mathbb{B}^{\mathbb{N}} \to \mathbb{S}$ computed by a polytime STM such that for all $\sigma_1, \ldots, \sigma_k, \tau \in \mathbb{S}$, and $\eta \in \mathbb{B}^{\mathbb{N}}$, $f(\sigma_1, \ldots, \sigma_k, \tau) = \mu(\{\eta \mid g(\sigma_1, \ldots, \sigma_k, \eta) = \tau\})$, and conversely. To conclude, we prove the correspondence between the classes $\mathcal{POR}$ and **SFP**:

(SFP $\Rightarrow$ $\mathcal{POR}$) The encoding relies on the remark that, given an input $x \in \mathbb{S}$ and an extra-tape $\eta \in \mathbb{B}^{\mathbb{N}}$, an STM $\mathcal{S}$ running in polynomial time can only access a *finite* portion of η, bounded by some polynomial $p(|x|)$. This way the behavior of $\mathcal{S}$ is encoded by a $\mathcal{POR}$-function $h(x, y)$, where the second string y corresponds to $\eta_{p(|x|)}$, and we can define $f^\sharp(x, \omega) = h(x, e(x, \omega))$, where $e : \mathbb{S} \times \mathbb{O} \to \mathbb{S}$ is a function of $\mathcal{POR}$ which mimics the prefix extractor $\eta \mapsto \eta_{p(|x|)}$, in the sense that its outputs have the same distributions of all possible η's prefixes (yet over $\mathbb{O}$ rather than $\mathbb{B}^{\mathbb{N}}$).

($\mathcal{POR}$ $\Rightarrow$ SFP) Here we must consider that these two models not only invoke oracles of different shape, but also that functions of $\mathcal{POR}$ can manipulate such oracles in a much more liberal way than STMs. Notably, the STM accesses oracle bits in a *linear* way: each bit is used exactly once and cannot be re-invoked. Moreover, at each step of computation the STM queries a new oracle bit, while functions of $\mathcal{POR}$ can access the oracle, so to say, *on demand*. The argument rests then on a chain of simulations, making use of a class of imperative languages inspired by Winskell's IMP [56], each one taking care of one specific oracle access policy: first non-linear and on-demand (as for $\mathcal{POR}$), then linear but still on-demand, and finally linear and not on-demand (as for STMs). ◀

4 Semantic Characterizations of BPP

We now turn our attention to randomized complexity classes. This requires us to consider how random functions (and thus PTMs) may correspond to languages, i.e. subsets of $\mathbb{S}$. The language computed by a random function can naturally be defined via a majority rule:

▶ **Definition 9.** *Let $f : \mathbb{S} \to \mathbb{D}(\mathbb{S})$ be a random function. The language* $\mathrm{Lang}(f) \subseteq \mathbb{S}$ *is defined by $\sigma \in \mathrm{Lang}(f)$ iff $f(\sigma)(\epsilon) > \frac{1}{2}$.*

It is instructive to first take a look at the case of the class **PP**, recalled below:

▶ **Definition 10 (PP).** *Given a language $L \subseteq \mathbb{S}$, $L \in$ **PP** iff there is a polynomial time PTM $\mathcal{M}$ such that for any $\sigma \in \mathbb{S}$, $\Pr[\mathcal{M}(\sigma) = \chi_L(\sigma)] > \frac{1}{2}$, where, $\chi_L : \mathbb{S} \to \{0, 1\}$ is the characteristic function of L.*

At first glance, **PP** might be considered a semantic class, since its definition comprises *both* a resource condition and a promise. However, **PP** is generally considered a syntactic class, due to the fact that, when trying to capture the machines solving languages in **PP**, the promise condition can actually be eliminated. Indeed, *any* PTM $\mathcal{M}$ running in polynomial time recognizes *some* language in **PP**, namely the language $L = \mathrm{Lang}(f)$, where f is the polytime random function computed by $\mathcal{M}$. Furthermore, the class **PP** *can* be enumerated (see e.g. [18]).

Using Theorem 7, the remarks above readily lead to a proof-theoretic characterization of **PP** via $R\Sigma^b_1$-NIA.

▶ **Proposition 11** (Syntactic Characterization of **PP**). *For any language $L \subseteq \mathbb{S}$, $L \in$ **PP** iff there is a Σ^b_1-formula $G(x, y)$ such that:*

1. *$R\Sigma^b_1$-NIA $\vdash \forall x.\exists! y.G(x, y)$,*
2. *$L = \mathrm{Lang}(\langle G \rangle)$.*

The characterization above provides an enumeration of **PP** (by enumerating the pairs made of a formula G and a proof in $R\Sigma^b_1$-NIA of Condition 1). However, while a majority rule is enough to capture the problems in **PP**, the definition of a semantic class like **BPP** requires a different condition.

▶ **Definition 12 (BPP).** *Given a language $L \subseteq \mathbb{S}$, $L \in$ **BPP** iff there is a polynomial time PTM $\mathcal{M}$ such that for any $\sigma \in \mathbb{S}$, $\Pr[\mathcal{M}(\sigma) = \chi_L(\sigma)] \geq \frac{2}{3}$.*

The class **BPP** can be captured by "non-erratic" probabilistic algorithms, i.e. such that, for a fixed input, one possible output is definitely more likely than the others.

▶ **Definition 13.** *A random function $f : \mathbb{S} \to \mathbb{D}(\mathbb{S})$ is* non-erratic *if for all $\sigma \in \mathbb{S}$, $f(\sigma)(\tau) \geq \frac{2}{3}$ holds for some value $\tau \in \mathbb{S}$.*

▶ **Lemma 14.** *For any language $L \subseteq \mathbb{S}$, $L \in$ **BPP** iff $L = \mathrm{Lang}(f)$, for some non-erratic random function $f \in$ **RFP**.*

Proof. For any non-erratic **RFP**-function f, let $\mathcal{M}$ be the PTM computing $k \diamond f$, where $k(\epsilon) = 1$ and $k(\sigma \neq \epsilon) = 0$; then $\mathcal{M}$ computes $\chi_{\mathrm{Lang}(f)}$ with error $\leq \frac{1}{3}$. Conversely, if $L \in$ **BPP**, let $\mathcal{M}$ be a PTM accepting L with error $\leq \frac{1}{3}$; then $L = \mathrm{Lang}(h \diamond f_{\mathcal{M}})$, where $h(1) = \epsilon$ and $h(\sigma \neq 1) = 0$. ◀

Lemma 14 suggests that, in order to characterize **BPP** in the spirit of Proposition 11, a new condition has to be added, corresponding to the fact that G represents a non-erratic random function. In the rest of this section we discuss two approaches to measure error bounds for probabilistic algorithms, leading to two different characterizations of **BPP**: first via measure quantifiers [2], then by purely arithmetical means. While both such methods ultimately consist in showing that the *truth* of a formula in the standard model of $R\Sigma_1^b$-NIA, they also suggest a more proof-theoretic approach, that we explore in Section 5.

BPP via Measure Quantifiers. As we have seen, any $\mathcal{RL}$-formula F is associated with a measurable set $[\![F]\!] \subseteq \mathbb{O}$. So, a natural idea, already explored in [2], consists in enriching $\mathcal{RL}$ with *measure-quantifiers* [48, 46], that is, second-order quantifiers of the form $\mathbf{C}^q F$, where $q \in [0,1] \cap \mathbb{Q}$, intuitively expressing that the measure of $[\![F]\!]$ is greater than (or equal to) q. Then, let $\mathcal{RL}^{\mathsf{MQ}}$ be the extension of $\mathcal{RL}$ with measure-quantified formulas $\mathbf{C}^{t/s}F$, where t, s are terms. The Borel semantics of $\mathcal{RL}$ naturally extends to $\mathcal{RL}^{\mathsf{MQ}}$ letting $[\![\mathbf{C}^{t/s}F]\!]_\xi = \mathbb{O}$ when $|[\![s]\!]_\xi| > 0$ and $\mu([\![F]\!]_\xi) \geq \frac{|[\![t]\!]_\xi|}{|[\![s]\!]_\xi|}$ both hold, and $[\![\mathbf{C}^{t/s}F]\!]_\xi = \emptyset$ otherwise. To improve readability, for all $n, m \in \mathbb{N}$, we abbreviate $\mathbf{C}^{\mathtt{1}^n/\mathtt{1}^m}F$ as $\mathbf{C}^{n/m}F$.

Measure quantifiers can now be used to express that the formula representing a random function is non-erratic, as shown below.

▶ **Theorem 15** (First Characterization of **BPP**). *For any language $L \subseteq \mathbb{S}$, $L \in$ **BPP** iff there is a Σ_1^b-formula $G(x,y)$ such that:*
1. *$R\Sigma_1^b$-NIA $\vdash \forall x \exists! y.G(x,y)$,*
2. *$\vDash \forall x.\exists y.\mathbf{C}^{2/3}G(x,y)$,*
3. *$L = \mathrm{Lang}(\langle G \rangle)$.*

Proof. Let $L \in$ **BPP** and $g : \mathbb{S} \to \mathbb{D}(\mathbb{S})$ be a function of **RFP** computing L with uniform error-bound (which, thanks to Lemma 14, we can suppose to be non-erratic). By Theorem 7, there is a Σ_1^b-formula $G(x,y)$ such that $g = \langle G \rangle$. So, for all $\sigma \in \mathbb{S}$, $\mu([\![G(\overline{\sigma}, \overline{\tau})]\!]) = g(\sigma)(\tau) \geq \frac{2}{3}$ holds for some $\tau \in \mathbb{S}$, which shows that Condition 2. holds. Conversely, if Conditions 1.-3. hold, then $\langle G \rangle$ computes L with the desired error bound, so $L \in$ **BPP**. ◀

Arithmetizing Measure Quantifiers. Theorem 15 relies on the tight correspondence between arithmetic and probabilistic computation; yet, Condition 2. involves formulas which are not in the language of first-order arithmetic. Lemma 16 below shows that measure quantification over bounded formulas of $\mathcal{RL}$ can be expressed arithmetically.

▶ **Lemma 16** (De-Randomization of Bounded Formulas). *For any Σ_1^b-formula $F(\vec{x})$ of $\mathcal{RL}$, there exists a* `Flip`*-free Π_1^0-formula* $\mathsf{TwoThirds}[F](\vec{x})$ *such that for any $\vec{\sigma} \in \mathbb{S}$, $\vDash \mathsf{TwoThirds}[F](\overline{\vec{\sigma}})$ holds iff $\mu([\![F(\overline{\vec{\sigma}})]\!]) \geq \frac{2}{3}$.*

Proof. First, observe that for any bounded $\mathcal{RL}$-formula $F(\vec{x})$, strings $\vec{\sigma}$ and $\omega \in \mathbb{O}$, to check whether $\omega \in [\![F(\overline{\vec{\sigma}})]\!]$ only a *finite* portions of bits of ω has to be observed. More precisely, we can construct a $\mathcal{RL}$-term $t_F(\vec{x})$ such that for any $\vec{\sigma} \in \mathbb{S}$ and $\omega, \omega' \in \mathbb{O}$, if ω and ω' agree on all strings shorter than $t_F(\vec{\sigma})$, then $\omega \in [\![F(\sigma)]\!]$ iff $\omega' \in [\![F(\sigma)]\!]$. Now, all finitely many relevant bits $\omega(\tau)$, for $|\tau| \leq t_F(\vec{\sigma})$ can be encoded as a *unique* string w of length $\leq 2^{|t_F(\vec{\sigma})|}$ where the bit w_i corresponds to the value $\omega(\tau)$, where τ is obtained by stripping the right-most bit from the binary representation of i. We obtain in this way a `Flip`-free formula $F^*(\vec{x}, y)$ such that measuring $[\![F(\sigma)]\!]$ corresponds to *counting* the strings y of length $\leq 2^{|t_F(\vec{\sigma})|}$ making $F^*(\vec{x}, y)$ true, i.e. to showing

$$\left| \left\{ \tau \preceq 2^{|t_F(\vec{\sigma})|} \;\middle|\; F^*(\vec{\sigma}, \tau) \right\} \right| \geq \frac{2}{3} \cdot N, \qquad (\star)$$

where $2^{\epsilon} = 1$ and $2^{\sigma b} = 2^{\sigma}2^{\sigma}$ is an exponential function on strings and $N = 2^{\left(2^{|t_F(\sigma)|}\right)}$ is the total amount of the strings to be counted. $(\star)$ can be encoded in a standard way yielding a bounded formula $F^{\sharp}(\vec{x})$ in the language of arithmetic *extended* with the function symbol 2^x. Finally, the function symbol 2^x can be eliminated using a Δ_0^0-formula $\exp(x, y)$ defining the exponential function (see [28]), yielding a `Flip`-free Π_1^0-formula of $\mathcal{RL}$ of the form $\forall z_1 \ldots . \forall z_k . \exp(t_1, z_1) \wedge \cdots \wedge \exp(t_k, z_k) \rightarrow F^{\sharp}(\vec{x}, z_1, \ldots, z_k)$. ◀

▶ Remark 17. It is important to observe at this point that the elimination of `Flip` via counting takes us *beyond* the usual machinery of bounded arithmetic, since we employ some operation which is not polytime. This is indeed not surprising, since the counting problems associated with polytime problems (generating the class $\sharp\mathsf{P}$) are not even known to belong to the polynomial hierarchy PH (while, by Toda's theorem, we know that $\mathsf{PH} \subseteq \mathsf{P}^{\sharp\mathsf{P}}$).

Theorem 15 and Lemma 16 together yield a purely arithmetical characterization of **BPP**. Let $\mathsf{NotErratic}[G]$ indicate the arithmetical formula $\forall x . \exists y \preceq 0 . \mathsf{TwoThirds}[G](x, y)$.

▶ **Theorem 18** (Second Characterization of **BPP**). *For any language $L \subseteq \mathbb{S}$, $L \in$ **BPP** when there is a Σ_1^b-formula $G(x, y)$ such that:*

1. *$R\Sigma_1^b$-NIA $\vdash \forall x . \exists! y . G(x, y)$,*
2. *$\models \mathsf{NotErratic}[G]$,*
3. *$L = \mathrm{Lang}(\langle G \rangle)$.*

5 Provably BPP Problems

The characterization provided by Theorem 18 is still semantic in nature, as it provides no way to effectively enumerate **BPP**: the crucial Condition 2 is not checked within a formal system, but over the standard model of $\mathcal{RL}$. Yet, since the condition is now expressed in purely arithmetical terms, it makes sense to consider *syntactic* variants of Condition 2, where the model-theoretic check is replaced by provability in some sufficiently expressive theory.

We will work in extensions of $R\Sigma_1^b$-NIA+Exp, where $\mathrm{Exp} = \forall x . \exists y . \exp(x, y)$ is the formula expressing the totality of the exponential function (which is used in the de-randomization of Lemma 16). This naturally leads to the following definition:

▶ **Definition 19** (Class $\mathbf{BPP}_{\mathrm{T}}$). *Let $\mathrm{T} \supseteq R\Sigma_1^b$-NIA + Exp be a theory in the language $\mathcal{RL}$. The* class **BPP** relative to T, *denoted $\mathbf{BPP}_{\mathrm{T}}$, contains all languages $L \subseteq \mathbb{S}$ such that for some Σ_1^b-formula $G(x, y)$ the following hold:*

1. *$R\Sigma_1^b$-NIA $\vdash \forall x . \exists! y . G(x, y)$,*
2. *$\mathrm{T} \vdash \mathsf{NotErratic}[G]$,*
3. *$L = \mathrm{Lang}(\langle G \rangle)$.*

Whenever T is sound (i.e. $\mathrm{T} \vdash F$ implies that F is true in the standard model), it is clear that $\mathbf{BPP}_{\mathrm{T}} \subseteq \mathbf{BPP}$. However, a crucial difference between the *syntactic* class $\mathbf{BPP}_{\mathrm{T}}$ and the semantic class **BPP** is that, when T is recursively enumerable, $\mathbf{BPP}_{\mathrm{T}}$ can be *enumerated* (by enumerating the proofs of Condition 1. and 2. in T). Hence, the enumerability problem for **BPP** translates into the question whether one can find a sound r.e. theory T such that $\mathbf{BPP}_{\mathrm{T}} = \mathbf{BPP}$. Let us first observe that the relevance of this problem is tightly related to the question $\mathbf{BPP} = \mathbf{P}$:

▶ **Proposition 20.** *If $\mathbf{BPP} = \mathbf{P}$, then there exists a r.e. theory T such that $\mathbf{BPP} = \mathbf{BPP}_{\mathrm{T}}$.*

Proof. If $\mathbf{BPP} = \mathbf{P}$, and $L \in \mathbf{BPP}$, then there is a polytime deterministic TM μ accepting it. μ yields then a PTM μ^* in a trivial way. Since the corresponding formula G of $\mathcal{RL}$ does not contain `Flip`, $\mathsf{NotErratic}[G]$ can be proved in e.g. $R\Sigma_1^b$-NIA + Exp. ◀

The counter-positive of the result above is even more interesting, as it says that establishing that *no* r.e. theory T is such that $\mathbf{BPP}_{\mathrm{T}} = \mathbf{BPP}$ is *at least as hard as* establishing that $\mathbf{BPP} \neq \mathbf{P}$. Yet, without knowing whether $\mathbf{BPP} = \mathbf{P}$, how hard may it be to find a theory T such that $\mathbf{BPP} = \mathbf{BPP}_{\mathrm{T}}$?

Observe that, when G is Σ_1^b, $\mathsf{NotErratic}[G]$ is expressed by a Π_1^0-formula: as $\mathsf{TwoThirds}[G](\vec{x})$ is of the form $\forall \vec{z}.\, \wedge_i \exp(t_i, z_i) \to F^\sharp(\vec{x}, \vec{z})$, the condition is expressed by the Π_1^0-formula $\forall x. \forall \vec{z}.\, \wedge_i \exp(t_i, z_i) \to \exists y \preceq \mathsf{0}. F^\sharp(\vec{x}, \vec{z})$. Hence, if we us fix some recursive enumeration $(\mathcal{M}_n)_{n\in\mathbb{N}}$ of polytime PTM as well as a recursive coding $\sharp\mathcal{M}$ of such machines as natural numbers, the fact that $\mathcal{M}$ is non-erratic is expressed by some Π_1^0-formula $\varphi_{\mathsf{NotErratic}}(\sharp\mathcal{M})$. The Π_1^0-set $\mathsf{NotErratic} = \{e \mid \varphi_{\mathsf{NotErratic}}(e)\}$ indicates then the sets of codes corresponding to non-erratic machines.

The possibility of finding a theory strong enough to prove *all* positive instances of Condition 2 is then ruled out by the following result.

▶ **Proposition 21.** $\mathsf{NotErratic}$ *is* Π_1^0*-complete.*

Proof. We reduce to $\mathsf{NotErratic}$ the Π_1^0-complete problem HALT_{n^2} consisting of codes of TM halting in time at most n^2 (see [29]). With any TM μ associate a polytime PTM μ^* that, on input x, yields TRUE with prob. $\frac{1}{2}$, and otherwise simulates $\mu(x)$ on $|x|^2$ steps, yielding TRUE if the computation of $\mu(x)$ terminated, and FALSE otherwise. Then it is easily seen that $\mu \in \mathsf{HALT}_{n^2}$ iff $\mu^* \in \mathsf{NotErratic}$. ◀

▶ **Corollary 22.** *Codes of poly-time and non-erratic PTMs form a* Σ_2^0*-complete set.*

Proof. As we say, for a PTM, solving some $\mathbf{BPP}$-problem is equivalent to being polytime and non-erratic. Being the code of a polytime (P)TM is a Σ_2^0-complete property [34]. By Proposition 21, checking non-erraticity does not increase the logical complexity. ◀

Proposition 21 implies that for any consistent theory T one can always find some non-erratic polytime PTM whose non-erraticity is *not provable* in T. Indeed, since $\mathsf{NotErratic}$ is Π_1^0-complete, we can reduce to it the Π_1^0-set of codes of *consistent* r.e. theories. Hence, if T is some consistent theory such that for *any* code $e \in \mathsf{NotErratic}$, T proves $\varphi_{\mathsf{NotErratic}}(e)$, then T can prove *all* Π_1^0-statement expressing the consistency of some consistent r.e. theory, and thus, in particular, the one expressing its own consistency, contradicting (Rosser's variant of) Gödel's second incompleteness theorem.

Observe that Corollary 22 suggests that the enumerability problem *might* be very difficult, but it *does not* provide a negative answer to it. Indeed, recall that what we are interested in is not an enumeration of *all* non-erratic polytime PTM, but an enumeration containing *at least one* machine for each problem in $\mathbf{BPP}$. In other words, the question remains open whether, for any non-erratic polytime PTM, it is possible to find a machine solving the same problem but whose non-erratic behavior can be proved in some fixed theory T. While we do not know the answer to this question, we can still show that a relatively weak arithmetical theory is capable of proving the non-erraticity of a machine solving one of the (very few) problems in $\mathbf{BPP}$ which are currently not known to be in $\mathbf{P}$.

6 Polynomial Zero Testing is Provably BPP

In this section we establish that PIT is in $\mathbf{BPP}_{(\mathrm{I}\Delta_0+\mathrm{Exp})}$. We recall that $\mathrm{I}\Delta_0 + \mathrm{Exp}$ is the fragment of Peano Arithmetics with induction restricted to *bounded formulas*, together with the totality of the exponential function.

▶ Remark 23. While $\mathrm{I}\Delta_0 + \mathrm{Exp}$ is a theory in the usual language of PA, here we work in a language for binary strings. Indeed, what we here call $\mathrm{I}\Delta_0 + \mathrm{Exp}$ is actually the corresponding theory Δ_0-NIA + Exp, formulated for the language $\mathcal{RL}$ *without* `Flip`, and defined as Σ_1^b-NIA + Exp with induction extended to *all* bounded formulas, plus the axiom Exp. Based on [26] Δ_0-NIA corresponds to Buss' theory S_2, which, in turn, is known to correspond to $\mathsf{I}\Delta_0 + \Omega_1$, indeed a sub-theory of $\mathrm{I}\Delta_0 + \mathrm{Exp}$.

The PIT problem asks to decide the identity of the polynomial computed by two arithmetical circuits. These are basically DAGs whose nodes can be labeled so as to denote an input, an output, the constants $0, 1$ or an arithmetic operation. These structures can easily be encoded, e.g. using lists, as terms of $\mathcal{RL}$.

▶ **Definition 24** (cf. [3]). *The problem PIT asks to decide whether two arithmetical circuits p, q encoded as lists of nodes describe the same polynomial, i.e. $\mathbb{Z} \models p = q$.*

Usually, PIT is reduced to another problem: the so-called Polynomial Zero Testing (PZT) problem, which asks to decide whether a polynomial computing a circuit over $\mathbb{Z}$ is zero, i.e. to check whether $\mathbb{Z} \models p = 0$. Indeed, $\mathbb{Z} \models p = q$ if and only if $\mathbb{Z} \models p - q = 0$. Our proof of the fact that the language PZT is in $\mathbf{BPP}_{(\mathrm{I}\Delta_0+\mathrm{Exp})}$ is structured as follows:

- We identify a Σ_1^b-formula $G(x, y)$ of $\mathcal{RL}$ characterizing the polytime algorithm `PZT` from [3], and we turn it into a `Flip`-free formula $G^*(x, y, z)$ as in Lemma 16, where the variable z stands for the source of randomness;
- We identify a `Flip`-free Δ_0^0-formula $H(x, y)$ which represents the naïve deterministic algorithm for PZT.
- We show that $\mathrm{I}\Delta_0 + \mathrm{Exp}$ proves a statement showing that the formulas G^* and H are equivalent in at least $\frac{2}{3}$ of all (finitely many) relevant values of z. In other words, we establish $\mathrm{I}\Delta_0 + \mathrm{Exp} \vdash \forall x.\forall y.\mathsf{TwoThirds}[G(x, y) \leftrightarrow H(x, y)]$.

From the last step, since the totality of H is provable in $\mathrm{I}\Delta_0 + \mathrm{Exp}$, we can deduce $\mathrm{I}\Delta_0 + \mathrm{Exp} \vdash \forall x.\exists y.\mathsf{TwoThirds}[G](x, y)$, as required in Definition 19.

Each of the aforementioned steps will be described in one of the forthcoming paragraphs, although the details are discussed in the extended version of this paper [1].

The Randomized Algorithm. Our algorithm for PZT takes an input x, which encodes a circuit p of size m on the variables $v_1, \ldots, v_n$, it draws $r_1, \ldots, r_n$ uniformly at random from $\{0, \ldots, 2^{m+3} - 1\}$ and k from $\{1, \ldots, 2^{2m}\}$, then it computes the value of $p(r_1, \ldots, r_n)$ mod k, so to ensure that during the evaluation no overflow can take place. This is done linearly many times in $|x|$ (we call this value s), as to ensure that, if the polynomial is not identically zero, the probability to evaluate p on values witnessing this property at least once grows over $\frac{2}{3}$. Finally, if all the evaluations returned 0 as output the input is accepted; otherwise, it is rejected.

The procedure described above is correct only when the size of the input circuit x is greater than some constant ϱ. If this is not the case, our algorithm queries a table T storing all the pairs $(x_i, \chi_{\mathrm{PZT}}(x_i))$ for $|x_i| < \varrho$, to obtain $\chi_{\mathrm{PZT}}(x_i)$. The table T can be pre-computed, having just a constant number of entries. This algorithm, which we call `PZT`, is inspired by [3] and described in detail in the extended version of this paper [1].

As the input circuit is evaluated modulo some $k \in \mathbb{Z}$, the algorithm works in time polynomial with respect to $|x|$. Therefore, as a consequence of Theorem 4 and Lemma 8, there is a Σ_1^b-formula $G(x, y)$ of $\mathcal{RL}$ that represents it. The extended version of this paper [1] also contains a lower-level description of this formula G.

The Underlying Language. We show that there is a predicate H of $\mathcal{RL}$ such that $H(x,\epsilon)$ holds if and only if x is the encoding of a circuit in PZT; otherwise, $H(x,0)$ holds. This predicate realizes the function h described by the following algorithm:

1. Take in input x, and check whether it is a polynomial circuit with one output; if it is not, reject it. Otherwise:
2. Compute the polynomial term p represented by x, and reduce it to a normal form $\overline{p}$.
3. Check whether all the coefficients of the terms are null. If this is true, output ϵ, otherwise output 1 and terminate.

For reasonable encodings of polynomial circuits and expressions, h is elementary recursive and therefore there is a predicate H which characterizes it, and $\mathrm{I}\Delta_0 + \mathrm{Exp}$ proves the totality of h. Moreover, we have $h = \chi_{\mathrm{PZT}}$, as for every polynomial p with coefficients in $\mathbb{Z}$, $\mathbb{Z} \models \forall \vec{x}.p(\vec{x}) = 0$ iff all the monomials in the normal form of p have zero as coefficient.

Proving the Error Bound. We now show that the formula G is not-erratic and that it decides $\mathrm{Lang}(\langle G \rangle)$. With the notations G^* and t_G from the the proof of Lemma 16, this can be reduced to proving in $\mathrm{I}\Delta_0 + \mathrm{Exp}$ the following two claims:

$$\vdash \forall z.\,(|z| = t_G(x) \wedge G^*(x,0,z)) \to H(x,0), \tag{$\dagger$}$$

$$\vdash \forall x.\left|\left\{z \preceq 2^{t_G(x)} \;\middle|\; G^*(x,\epsilon,z) \to H(x,\epsilon)\right\}\right| \geq \frac{2}{3}\cdot 2^{|2^{t_G(x)}|}. \tag{$\ddagger$}$$

($\dagger$) states that whenever the randomized algorithm rejects an input, then so does the deterministic one, while ($\ddagger$), which is reminiscent of ($\star$), states that in at least $\frac{2}{3}$ of all possible cases, if the randomized algorithm accepts the circuit, the deterministic one accepts it too. Jointly, ($\dagger$) and ($\ddagger$) imply that the equivalence $G^*(x,y,z) \leftrightarrow H(x,y)$ holds in at least $2/3$ of all possible cases.

While Claim ($\dagger$) is a consequence of the compatibility of the mod k function with addition and multiplication, which are easily proved in $\mathrm{I}\Delta_0 + \mathrm{Exp}$, the proof of Claim ($\ddagger$) is more articulated and relies on the Schwartz-Zippel Lemma, providing a lower bound to the probability of evaluating the polynomial on values witnessing that it is not identically zero, and the Prime Number Theorem (whose provability in $\mathrm{I}\Delta_0 + \mathrm{Exp}$ is known [16]) which bounds the probability to choose a *bad* value for k, i.e. one of those values causing `PZT` to return the wrong value. Detailed arguments are provided in the extended version of this paper [1].

Closure under Polytime Reduction. Only assessing that a problem belongs to $\mathbf{BPP}_{\mathrm{T}}$ does not tell us anything about other languages of this class; for this reason, we are interested in showing that $\mathbf{BPP}_{\mathrm{T}}$ is closed under polytime reduction. This allows us to start from $\mathrm{PZT} \in \mathbf{BPP}_{(\mathrm{I}\Delta_0+\mathrm{Exp})}$ to conclude that all problems which can be reduced to PZT in polynomial time belong to this class, and in particular that $\mathrm{PIT} \in \mathbf{BPP}_{(\mathrm{I}\Delta_0+\mathrm{Exp})}$. This is assessed by the following proposition, proved in the the extended version of this paper:

▶ **Proposition 25.** *For any theory* $\mathrm{T} \supseteq R\Sigma^b_1\text{-}NIA + \mathrm{Exp}$*, language* $L \in \mathbf{BPP}_{\mathrm{T}}$ *and language* $M \subseteq \mathbb{S}$*, if there is a polytime reduction from* M *to* L*, then* $M \in \mathbf{BPP}_{\mathrm{T}}$*.*

▶ **Corollary 26.** *PIT is in* $\mathbf{BPP}_{(\mathrm{I}\Delta_0+\mathrm{Exp})}$*.*

7 On Jeřábek's Characterization of BPP

As mentioned in Section 1, a semantic characterization of **BPP** based on bounded arithmetic was already provided by Jeřábek in [41]. This approach relies on checking, against the standard model, the truth of a formula which, rather than expressing that some machine is non-erratic, expresses what can be seen as a second totality condition (beyond the formula expressing the totality of the algorithm). Hence, also within this approach we think it makes sense to investigate which problems can be proved to be in **BPP** within some given theory.

In this section, we relate the two approaches by showing that the problems in $\mathbf{BPP}_{\mathrm{T}}$ are provably definable **BPP** problems, in the sense of [41], *within some suitable extension* of the bounded theory PV_1[13].

A PTM is represented in this setting by two provably total functions (A, r), where the machine accepts on input x with probability less than p/q when $\Pr_{w<r(x)}(A(x,w)) \leq p/q$. Jeřábek focuses on the theory PV_1, extended with an axiom schema dWPHP (PV_1)called the *dual weak pigeonhole principle* (cf. [41, pp. 962ff.]) for PV_1 (i.e. the axiom stating that for every PV_1-definable function f, f is not a surjection from x to x^2). The reason is that this theory is capable of proving *approximate* counting formulas of the form $\Pr_{w<r(x)}(A(x,w)) \preceq_0 p/q$, where "$\preceq_0$" is a relation equivalent to "$\leq$" up to some polynomially small error (recall that, in order to establish *exact* counting results, we were forced to use non-polytime operations, cf. Remark 17). The representation of **BPP** problems hinges on the definition, for any probabilistic algorithm (A, r), of $L^+_{A,r}(x) := \Pr_{w<r(x)}(\neg A(x,w)) \leq 1/3$ and $L^-_{A,r}(x) := \Pr_{w<r(x)}(A(x,w)) \leq 1/3$. Checking if the algorithm (A, r) solves some problem in **BPP** reduces then to checking the "totality" formula $\vDash \forall x.L^+_{A,r}(x) \vee L^-_{A,r}(x)$.

Now, first observe that, modulo an encoding of strings via numbers, everything which is provable in $R\Sigma^b_1$-NIA *without* the predicate `Flip` can be proved in the theory $S^1_2(PV)$ [13], which extends both PV_1 and Buss' S^1_2. Moreover, by arguing as in the proof of Lemma 14, in our characterization of **BPP** we can w.l.o.g. suppose that the formula G satisfies $\mathrm{EpsZero}[G] := \forall x.\forall y.G(x,y) \to y = \epsilon \vee y = 0$. Under this assumption, the de-randomization procedure described in the proof of Lemma 16 turns G into a pair (A, r), where $A = G^*$ is `Flip`-free and $r(x) = t_G(x)$, and the languages $L^+_{A,r}(x)$ and $L^-_{A,r}(x)$ correspond then to the formulas $L^+_G(x) := \mathsf{TwoThirds}[G(-,\epsilon)](x)$, and $L^-_G(x) := \mathsf{TwoThirds}[G(-,0)](x)$.

Now, since from $\mathrm{T} \vdash \forall x.\exists y.\mathsf{TwoThirds}[G](x,y)$ and $\mathrm{EpsZero}[G]$ one can deduce $\mathrm{T} \vdash \forall x.L^+_G(x) \vee L^-_G(x)$, we arrive at the following:

▶ **Proposition 27.** *Let L be a language with $L = \mathrm{Lang}(\langle G \rangle)$. If $L \in \mathbf{BPP}_{\mathrm{T}}$, then $\forall x.L^+_G(x) \vee L^-_G(x)$ is provable in some recursively enumerable extension of PV_1. Conversely, if $PV_1 + dWPHP(PV_1) \vdash \forall x.L^+_G(x) \vee L^-_G(x)$, then $L \in \mathbf{BPP}_{R\Sigma^b_1\text{-}NIA+\mathrm{Exp}}$.*

The second statement above relies on the fact that approximate counting can be replaced by exact counting in $R\Sigma^b_1$-NIA + Exp (i.e. "$\preceq_0$" can be replaced by "$\leq$").

8 Future Work

The authors see this work as a starting point for a long-term study on the logical nature of semantic classes. From this point of view, many ideas for further work naturally arise.

An exciting direction is the study of the expressiveness of the new syntactic classes $\mathbf{BPP}_{\mathrm{T}}$, that is, an investigation on the kinds of error bounds which can be proved in the arithmetical theories lying *between* standard bounded theories like S^1_2, PV and PA, but also in theories which are *more expressive* than PA (like e.g. second-order theories). Surely, classes of the form $\mathbf{BPP}_{\mathrm{T}}$ could be analyzed also as for the existence of complete problems and hierarchy theorems for them, since such results are not known to hold for **BPP** itself [27].

Our approach to **BPP** suggests that extensions to other complexity classes of randomized algorithms like **ZPP**, **RP** and **coRP** could make sense. Notice that this requires to deal not only with error-bounds, but also with either average class complexity or with failure in decision procedures.

Finally, given the tight connections between bounded arithmetics and proof complexity, another natural direction is the study of applications of our work to randomized variations on the theme, for example recent investigations on *random resolution refutations* [41, 6, 52], i.e. resolution systems where proofs may make errors but are correct most of the time.

9 Conclusion

The logical characterization of randomized complexity classes, in particular those having a semantic nature, is a great challenge. This paper contributes to the understanding of this problem by showing not only how resource bounded randomized computation can be captured within the language of arithmetic, but also that the latter offers convenient tools to control error bounds, the essential ingredient in the definition of classes like **BPP** and **ZPP**.

We believe that the main contribution of this work is a first example of a sort of *reverse* computational complexity for probabilistic algorithms. As we discussed in Section 5, while the restriction to bounded theories is crucial in order to capture polytime algorithms via a totality condition, it is not necessary to prove error bounds for probabilistic (even polynomial time) algorithms. In particular, the (difficult) challenge of enumerating **BPP** translates into the challenge of proving $\mathbf{BPP} = \mathbf{BPP}_{\mathrm{T}}$ for some strong enough r.e. theory T. So, it is worth exploring how much can be proved within expressive arithmetical theories. For this reason we focused here on a well-known problem, PIT, which is known to be in **BPP**, but not in **P**, showing that the whole argument for PIT $\in$ **BPP** can be formalized in a fragment of PA, namely $\mathrm{I}\Delta_0 + \mathrm{Exp}$.

References

1 M. Antonelli, U. Dal Lago, D. Davoli, I. Oitavem, and P. Pistone. Enumerating error bounded polytime algorithms through arithmetical theories, 2023. `arXiv:2311.15003`.

2 M. Antonelli, U. Dal Lago, and P. Pistone. On measure quantifiers in first-order arithmetic. In *Proc. of CiE 2021*, pages 12–24. Springer-Verlag, 2021.

3 S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

4 S. Bellantoni and S. Cook. A New Recursion-Theoretic Characterization of the Polytime Functions. *Computational Complexity*, 2:97–110, 1992.

5 P. Billingsley. *Probability and Measure*. Wiley, 1995.

6 S. Buss, A.L. Kolodziejczyk, and N. Thapen. Fragments of Approximate Counting. *Journal of Symbolic Logic*, 79(2):496–525, 2014.

7 S.R. Buss. *Bounded Arithmetic*. PhD thesis, Princeton University, 1986.

8 S.R. Buss. First-Order Proof Theory of Arithmetic. In S.R: Buss, editor, *Handbook of Proof Theory*. Elsavier, 1998.

9 A. Church. An Unsolvable Problem of Elementary Number Theory. *American J. of Mathematics*, 58(2):345–363, 1992.

10 A. Cobham. The intrinsic computational difficulty of functions. In *Proc. of the 1964 International Congress on Logics, Methodology and Philosophy of Science*, pages 24–30. North-Holland Publishing, 1965.

11 E.F. Codd. Relational Completeness of Data Base Sublanguages. In *Proc. of 6th Courant Computer Science Symposium.*, pages 65–98, 1972.

12 S. Cook. The Complexity of Theorem Proving Procedures. In *Proc. of STOC 1971*, pages 151–158, 1971.

13 S. Cook and A. Urquhart. Functional Interpretations of Feasibly Constructive Arithmetic. *Annals of Pure and Applied Logic*, 63(2):103–200, 1993.

14 S.A. Cook. Feasibly constructive proofs and the propositional calculus. In ACM Press, editor, *Proc. of STOC 1975*, pages 83–97, 1975.

15 S.A. Cook and R.A. Reckhow. Efficiency of Propositional Proof Systems. *Journal of Symbolic Logic*, 44(1):36–50, 1979.

16 C. Cornaros and C. Dimitracopoulos. The Prime Number Theorem and Fragments of PA. *Archive for Mathematical Logic*, 33:265–281, August 1994.

17 H. B. Curry. Functionality in Combinatory Logic. *Proceedings of the National Academy of Sciences*, 20(11):584–590, 1934.

18 U. Dal Lago, R. Kahle, and I. Oitavem. A Recursion-Theoretic Characterization of the probabilistic Class PP. In *Proc. of MFCS 2021*, pages 1–12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

19 U. Dal Lago, R. Kahle, and I. Oitavem. Implicit Recursion-Theoretic Characterizations of Counting Classes. *Archive for Mathematical Logic*, May 2022.

20 U. Dal Lago and P. Parisen Toldin. A Higher-Order Characterization of Probabilistic Polynomial Time. *Information and Computation*, 241:114–141, 2015.

21 K. Eickmeyer and M. Grohe. Randomisation and Derandomisation in Descriptive Complexity Theory. In *Proc. of CSL 2010*. Springer, 2010.

22 R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. *Complexity of computation*, 7:43–73, 1974.

23 F. Ferreira. Polynomial-Time Computable Arithmetic and Conservative Extesions. Ph.D. Dissertation, December 1988.

24 F. Ferreira. Polynomial-Time Computable Arithmetic. In W. Sieg, editor, *Logic and Computation*, volume 106 of *Contemporary Mathematics*, pages 137–156. AMS, 1990.

25 F. Ferreira. *Stockmeyer induction*, pages 161–180. Birkhäuser Boston, Boston, MA, 1990. doi:10.1007/978-1-4612-3466-1_9.

26 G. Ferreira and I. Oitavem. An Interpretation of S_2^1 in Σ_1^b-NIA. *Portugaliae Mathematica*, 63:137–156, 2006.

27 L. Fortnow. Comparing notions of full derandomization. In *Proceedings of the 16th Annual IEEE Conference on Computational Complexity*, pages 28–34, Chicago, IL, USA, 2001. IEEE Computer Society.

28 H. Gaifman and C. Dimitracopoulos. Fragments of Peano's arithmetic and the MRDP theorem. *Logic and Algorithmic, Monograph. Enseign. Math.*, 30:187–206, 1982.

29 D. Gajser. Verifying Time Complexity of Turing Machines. *Informatica*, 40:369–370, 2016.

30 J.-Y. Girard. Light Linear Logic. *Information and Computation*, 2(143):175–204, 1998.

31 J.-Y. Girard and Y. Lafont. *Advances in Linear Logic*. Cambridge University Press, 1995.

32 J.-Y. Girard, A. Scedrov, and P. Scott. Bounded Linear Logic: A Modular Approach to Polynomial-Time Computability. *Theoretical Computer Science*, 97(1):1–66, 1992.

33 K. Gödel. Über Formal Unentscheidbare Sätze der Principia Mathematica and Verwandter Systeme. *Monatshefte für Mathematik und Physik*, 38:173–198, 1931.

34 P. Hájek. Arithmetical Hierarchy and Complexity of Computation. *Theoretical Computer Science*, 8(2):227–237, 1979.

35 P. Hájek and P. Pudlák. *Metamathematics of First-Order Arithmetic*. Springer, Berlin-Heidelberg, 1998.

36 J. Hartmanis and R.E. Stearns. On the Computational Complexity of Algorithms. *Transactions of the AMS*, 117:285–306, 1965.

37 M. Hofmann. Programming Languages Capturing Complexity Classes. *SIGACT News*, 31(1):31–42, March 2000.

38 H. A. Howard. The Formulae-as-Types Notion of Construction. In *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus, and Formalism*. Academic Press, 1980.
39 N. Immerman. *Descriptive Complexity*. Springer, 1999.
40 E. Jeřábek. Dual Weak Pigeonhole Principle, Boolean Complexity, and Derandomization. *Annals of Pure and Applied Logic*, 129(1):1–37, 2004.
41 E. Jeřábek. Approximate Counting in Bounded Arithmetic. *Journal of Symbolic Logic*, 72(3):959–993, 2007.
42 J. Krajíček and P. Pudlák. Propositional Proof Systems, the Consistency of First-Order Theories and the Complexity of Computations. *Journal of Symboic Logic*, 54(3):1063–1079, 1989.
43 J. Krajíček, P. Pudlák, and G. Takeuti. Bounded Arithmetic and the Polynomial Hierarchy. *Annals of Pure and Applied Logic*, 52:143–153, 1991.
44 Y. Lafont. Soft Linear Logic and Polynomial Time. *Theoretical Computer Science*, 1/2(318):163–180, 2004.
45 D. Leivant. Ramified Recurrence and Computational Complexity I: Word Recurrence and Polytime. In P. Clote and J. Remmel, editors, *Feasible Mathematics II*, pages 320–343. Springer, 1995.
46 H. Michalewski and M. Mio. Measure Quantifiers in Monadic Second Order Logic. In *Proc. of LFCS*, pages 267–282, Cham, 2016. Springer.
47 J. Mitchell, M. Mitchell, and A. Scedrov. A Linguistic Characterization of Bounded Oracle Computation and Probabilistic Polynomial Time. In *Proc. of FOCS 1998*, pages 725–733. IEEE Computer Society, 1998.
48 C. Morgenstern. The Measure Quantifier. *Journal of Symbolic Logic*, 44(1):103–108, 1979.
49 R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, Cambridge; NY, 1995.
50 C.H. Papadimitriou. *Computational Complexity*. Pearson Education, 1993.
51 R. Parikh. Existence and Feasibility in Arithmetic. *Journal of Symbolic Logic*, 36:494–508, 1971.
52 P. Pudlák and N. Thapen. Random Resolution Refutations. *Computational Complexity*, 28:185–239, 2019.
53 E.S. Santos. Probabilistic Turing Machines and Computability. *AMS*, 22(3):704–710, 1969.
54 M.H. Sørensen and P. Urzyczyn. *Lectures on the Curry-Howard Isomorphism*. Elsevier, 2006.
55 A. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proc. London Mathematical Society*, pages 2–42, 230–265, 1936-37.
56 G. Winskel. *The Formal Semantics of Programming Languages: An Introduction*. MIT press, 1993.

Active Learning of Deterministic Transducers with Outputs in Arbitrary Monoids

Quentin Aristote
École Normale Supérieure de Paris, PSL University, France
Université Paris Cité, CNRS, Inria, IRIF, F-75013, Paris, France

Abstract

We study monoidal transducers, transition systems arising as deterministic automata whose transitions also produce outputs in an arbitrary monoid, for instance allowing outputs to commute or to cancel out. We use the categorical framework for minimization and learning of Colcombet, Petrişan and Stabile to recover the notion of minimal transducer recognizing a language, and give necessary and sufficient conditions on the output monoid for this minimal transducer to exist and be unique (up to isomorphism). The categorical framework then provides an abstract algorithm for learning it using membership and equivalence queries, and we discuss practical aspects of this algorithm's implementation.

2012 ACM Subject Classification Theory of computation → Transducers

Keywords and phrases transducers, monoids, active learning, category theory

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.11

Related Version *Extended Version*: https://ens.hal.science/hal-04172251v2

1 Introduction

Transducers are (possibly infinite) transition systems that take input words over an input alphabet and translate them to some output words over an output alphabet. They are numerous ways to implement them, but here we focus on *subsequential transducers*, i.e deterministic automata whose transitions also produce an output (see Figure 1 for an example). They are used in diverse fields such as compilers [11], linguistics [13], or natural language processing [14].

Two subsequential transducers are considered equivalent when they *recognize* the same *subsequential function*, that is if, given the same input, they always produce the same output. A natural question is thus whether there is a (unique) minimal transducer recognizing a given function (a transducer with a minimal number of states and which produces its ouput as early as possible), and whether this minimal transducer is computable. The answer to both these questions is positive when there exists a finite subsequential transducer recognizing this function: the minimal transducer can then for example be computed through minimization [6].

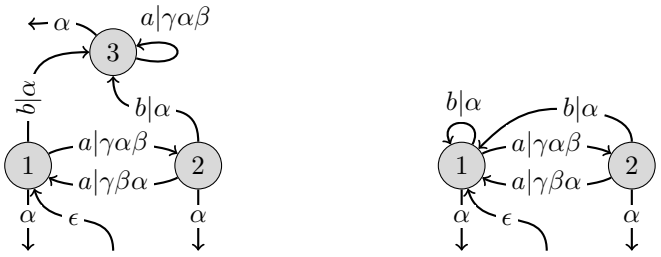
Active learning of transducers

Another method for computing a minimal transducer is to learn it through Vilar's algorithm [21], a generalization to transducers of Angluin's L*-algorithm, which learns the minimal deterministic automaton recognizing a language [1]. Vilar's algorithm thus relies on the existence of an oracle which may answer two types of queries, namely:

- *membership queries*: when queried with an input word, the oracle answers with the corresponding expected output word;
- *equivalence queries*: when queried with a *hypothesis transducer*, the oracle answers whether this transducer recognizes the target function, and, if not, provides a counter-example input word for which this transducer is wrong.

The basic idea of the algorithm is to use the membership queries to infer partial knowledge of the target function on a finite subset of input words, and, when some *closure* and *consistency* conditions are fulfilled, use this partial knowledge to build a hypothesis transducer to submit to the oracle through an equivalence query: the oracle then either confirms this transducer is the right one, or provides a counter-example input word on which more knowledge of the target function should be inferred.

Figure 1 Two transducers: unlike automata, the transitions are also labelled with output words.



(a) A monoidal transducer $\mathcal{A}$. **(b)** A hypothesis transducer built when learning $\mathcal{A}$.

Consider for instance the partial function recognized by the minimal transducer $\mathcal{A}$ of Figure 1a over the input alphabet $A = \{a, b\}$ and output alphabet $\Sigma = \{\alpha, \beta, \gamma\}$. We write this function $\mathcal{L}(\triangleright - \triangleleft) : A^* \to \Sigma^* \sqcup \{\bot\}$, and let $e \in A^*$ and $\epsilon \in \Sigma^*$ stand for the respective empty words over these two alphabets. To learn $\mathcal{A}$, the algorithm maintains a subset $Q \subset A^*$ of prefixes of input words and a subset $T \subset A^*$ of suffixes of input words, and keeps track of the restriction of $\mathcal{L}(\triangleright - \triangleleft)$ to words in $QT \cup QAT$. The prefixes in Q will be made into states of the hypothesis transducer, and two prefixes $q, q' \in Q$ will correspond to two different states if there is a suffix $t \in T$ such that $\mathcal{L}(\triangleright qt \triangleleft) \neq \mathcal{L}(\triangleright q't \triangleleft)$. Informally, closure then holds when for any state $q \in Q$ and input letter $a \in A$ an a-transition to some state $q' \in Q$ can always be built; consistency holds when there is always at most one consistent choice for such a q' and when the newly-built a-transition can be equipped with an output word. The execution of the learning algorithm for the function recognized by $\mathcal{A}$ would thus look like the following.

The algorithm starts with $Q = T = \{e\}$ only consisting of the empty input word. In a hypothesis transducer, we would want $e \in Q$ to correspond to the initial state, and the output value produced by the initial transition to be the longest common prefix $\Lambda(e)$ of each $\mathcal{L}(\triangleright et \triangleleft)$ for $t \in T$, here $\Lambda(e) = \alpha$. But the longest common prefix $\Lambda(a)$ of each $\mathcal{L}(\triangleright at \triangleleft)$ for $t \in T$ is $\gamma\alpha\beta\alpha$, of which $\Lambda(e)$ is not a prefix: it is not possible to make the output of the first a-transition so that following the initial transition and then the a-transition produces a prefix of $\Lambda(a)$! This is a first kind of consistency issue, which we solve by adding a to T, turning $\Lambda(e)$ into the empty output word ϵ and $\Lambda(a)$ into $\gamma\alpha\beta$.

Now $Q = \{e\}$ and $T = \{e, a\}$. The initial transition should go into the state corresponding to e and output $\Lambda(e) = \epsilon$, the final transition from this state should output $\Lambda(e)^{-1}\mathcal{L}(\triangleright e \triangleleft) = \alpha$, the a-transition from this state should output $\Lambda(e)^{-1}\Lambda(a) = \gamma\alpha\beta$, and this a-transition followed by a final transition should output $\Lambda(e)^{-1}\mathcal{L}(\triangleright a \triangleleft) = \gamma\alpha\beta\alpha$. This a-transition should moreover lead to a state from which another a-transition followed by a final transition outputs $\Lambda(a)^{-1}\mathcal{L}(\triangleright aa \triangleleft) = \gamma\beta\alpha^2$: in particular, it cannot lead back to the state corresponding to e, because $\gamma\beta\alpha^2 \neq \gamma\alpha\beta\alpha$. But this state is the only state accounted for by Q, so now we have no candidate for its successor when following the a-transition! This is a closure issue, which we solve by adding a to Q, the corresponding new state then being the candidate successor we were looking for.

Once $Q = T = \{e, a\}$, there are no closure nor consistency issues and we may thus build the hypothesis transducer given by Figure 1b: it coincides with $\mathcal{A}$ on $QA \cup QAT$. Submitting it to the oracle we learn that this transducer is not the one we are looking for, and we get as counter-example the input word bb, which indeed satisfies $\mathcal{L}(\triangleright bb \triangleleft) = \bot$ and yet for which our hypothesis transducer produced the output word α^3: we thus add bb and its prefixes to Q.

With $Q = \{e, a, b, bb\}$ and $T = \{e, a\}$ there is another kind of consistency issue, because the states corresponding to e and b are not distinguished by T ($\Lambda(e)^{-1}\mathcal{L}(\triangleright et \triangleleft) = \Lambda(b)^{-1}\mathcal{L}(\triangleright bt \triangleleft)$ for all $t \in T$) and should thus be merged in the hypothesis transducer, yet this is not the case of their candidate successors when following an additional b-transition ($\mathcal{L}(\triangleright ebe \triangleleft) = \alpha$ yet $\mathcal{L}(\triangleright bbe \triangleleft) = \bot$ is undefined)! This issue is solved by adding b to T, after which there are again no closure nor consistency issues and we may thus build $\mathcal{A}$ as our new hypothesis transducer. The algorithm finally stops as the oracle confirms that we found the right transducer.

Transducers with outputs in arbitrary monoids

In the example above we assumed the output of the transducer consisted of words over the output alphabet $\Sigma = \{\alpha, \beta, \gamma\}$, that is of elements of the free monoid Σ^*. But in some contexts it may be relevant to assume that certain output words can be swapped or can cancel each other out. In other words, transducers may be considered to be *monoidal* and have output not in a free monoid, but in a quotient of a free monoid. An example of a non-trivial family of monoids that should be interesting to use as the output of a transducer is the family of trace monoids, that are used in concurrency theory to model sequences of executions where some jobs are independent of one another and may thus be run asynchronously: transducers with outputs in trace monoids could be used to programatically schedule jobs. Algebraically, trace monoids are just free monoids where some pairs of letters are allowed to commute. For instance, the transducers of Figure 1 could be considered under the assumption that $\alpha\beta = \beta\alpha$, in which case the states 1 and 2 would have the same behavior.

This raises the question of the existence and computability of a minimal monoidal transducer recognizing a function with output in an arbitrary monoid. In [12], Gerdjikov gave some conditions on the output monoid for minimal monoidal transducers to exist and be unique up to isomorphism, along with a minimization algorithm that generalizes the one for (non-monoidal) transducers. This question had also been addressed in [10], although in a less satisfying way as the minimization algorithm relied on the existence of stronger oracles. Yet, to the best of the author's knowledge, no work has addressed the problem of learning minimal monoidal transducers through membership and equivalence queries.

As all monoids are quotients of free monoids, a first solution would of course be to consider the target function to have output in a free monoid, learn the minimal (non-monoidal) transducer recognizing this function using Vilar's algorithm, and only then consider the resulting transducer to have output in a non-free monoid and minimize it using Gerdjikov's minimization algorithm. But this solution is unsatisfactory as, during the learning phase, it may introduce states that will be optimized away during the minimization phase. For instance, learning the function recognized by the transducer $\mathcal{A}$ of Figure 1a with the assumption that $\alpha\beta = \beta\alpha$ would first produce $\mathcal{A}$ itself before having its states 1 and 2 merged during the minimization phase. Worse still, it is possible to find a partial function with output in a finitely generated quotient monoid $\Sigma^*/\sim$ and recognized by a finite monoidal transducer, and yet so that, when this function is considered to have output in Σ^*, Vilar's algorithm may not even terminate if, when answering membership queries, the oracle does not carefully choose the representatives in Σ^* of each equivalence class in $\Sigma^*/\sim$ (this is made formal by Lemma 34 in the appendix; the idea of finding such an example was suggested by an anonymous reviewer whom the author thanks).

A more satisfactory solution would hence do away with the minimization phase and instead use the assumptions on the output monoid during the learning phase to directly produce the minimal monoidal transducer.

Structure and contributions

In this work we thus study the problem of generalizing Vilar's algorithm to monoidal transducers. To this aim, we first recall in Section 2 the categorical framework of Colcombet, Petrişan and Stabile for learning minimal transition systems [7]. This framework encompasses both Angluin's and Vilar's algorithms, as well as a similar algorithm for weighted automata [4, 5]. We use this specific framework because, while others exist, they either do not encompass transducers or require stronger assumptions [3, 19, 20]. In Section 3 we then instantiate this framework to retrieve monoidal transducers as transition systems whose state-spaces live in a certain category (Section 3.2). Studying the existence of specific structures in this category – namely, powers of the terminal object (Section 3.3) and factorization systems (Section 3.4) – we then give conditions for the framework to apply and hence for the minimal monoidal transducers to exist and be computable.

This paper's contributions are thus the following:

- necessary and sufficient conditions on the output monoid for the categorical framework of Colcombet, Petrişan and Stabile to apply to monoidal transducers are given;
- these conditions mostly overlap those of Gerdjikov, but are nonetheless not equivalent: in particular, they extend the class of output monoids for which minimization is known to be possible, although with a possibly worse complexity bound;
- practical details on the implementation of the abstract monoidal transducer-learning algorithm that results from the categorical framework are given;
- in particular, additional structure on the category in which the framework is instantiated provides a neat categorical explanation to both the different kinds of consistency issues that arise in the learning algorithm and the main steps that are taken in every transducer minimization algorithm.

2 Categorical approach to learning minimal automata

In this section we recall (and extend) the definitions and results of Colcombet, Petrişan and Stabile [16, 8]. We assume basic knowledge of category theory [15], but we also focus on the example of deterministic complete automata and on the counter-example of non-deterministic automata. We do not explain it here but the framework also applies to weighted automata [4, 5, 17] and (non-monoidal) transducers (as generalized in Section 3).

2.1 Automata and languages as functors

Let $\mathcal{I}$, the *input category*, be the category freely generated by the diagram $\mathtt{in} \xrightarrow{\triangleright} \mathtt{st} \overset{a}{\circlearrowleft} \xrightarrow{\triangleleft} \mathtt{out}$ where a ranges in the *input alphabet* A: the objects are the vertices of the graph and the morphisms paths between two vertices. $\mathcal{I}$ represents the basic structure of automata as transition systems: $\mathtt{st}$ represents the state-space, $\triangleright$ the initial configuration, each $a : \mathtt{st} \to \mathtt{st}$ the transition along the corresponding letter, and $\triangleleft$ the output values associated to each state. An automaton is then an instantiation of $\mathcal{I}$ in some output category:

▶ **Definition 1** ($(\mathcal{C}, X, Y)$-automaton)**.** *Given an output category $\mathcal{C}$, a $(\mathcal{C}, X, Y)$-automaton is a functor $\mathcal{A} : \mathcal{I} \to \mathcal{C}$ such that $\mathcal{A}(\mathtt{in}) = X$ and $\mathcal{A}(\mathtt{out}) = Y$.*

▶ **Example 2** ((non-)deterministic automata). If $1 = \{*\}$ and $2 = \{\bot, \top\}$, a $(\mathbf{Set}, 1, 2)$-automaton $\mathcal{A}$ is a (possibly infinite) deterministic complete automaton: it is given by a state-set $S = \mathcal{A}(\mathtt{st})$, transition functions $\mathcal{A}(a) : S \to S$ for each $a \in A$, an initial state $s_0 = \mathcal{A}(\triangleright)(*) \in S$ and a set of accepting states $F = \{s \in S \mid \mathcal{A}(\triangleleft)(s) = \top\} \subseteq S$. Similarly, a $(\mathbf{Rel}, 1, 1)$-automaton (where **Rel** is the category of sets and relations between them) is a (possibly infinite) non-deterministic automaton: it is given by a state-set $S = \mathcal{A}(\mathtt{st})$, transition relations $\mathcal{A}(a) \subseteq S \times S$, a set of initial states $\mathcal{A}(\triangleright) \subseteq 1 \times S \cong S$ and a set of accepting states $\mathcal{A}(\triangleleft) \subseteq S \times 1 \cong S$.

▶ **Definition 3** ($(\mathcal{C}, X, Y)$-language). *In the same way, we define a* language *to be a functor $\mathcal{L} : \mathcal{O} \to \mathcal{C}$, where $\mathcal{O}$, the* category of observable inputs*, is the full subcategory of $\mathcal{I}$ on* `in` *and* `out`*. In other words, a language is the data of two objects $X = \mathcal{L}(\mathtt{in})$ and $Y = \mathcal{L}(\mathtt{out})$ in $\mathcal{C}$ (in which case we speak of a $(\mathcal{C}, X, Y)$-*language*), and, for each word $w \in A^*$, of a morphism $\mathcal{L}(\triangleright w \triangleleft) : X \to Y$. In particular, composing an automaton $\mathcal{A} : \mathcal{I} \to \mathcal{C}$ with the embedding $\iota : \mathcal{O} \hookrightarrow \mathcal{I}$, we get the language $\mathcal{L}_\mathcal{A} = \mathcal{A} \circ \iota$ recognized by $\mathcal{A}$.*

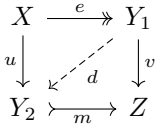
▶ **Example 4** (languages). The language recognized by a $(\mathbf{Set}, 1, 2)$-automaton $\mathcal{A}$ is the language recognized by the corresponding complete deterministic automaton: for a given $w \in A^*$, $\mathcal{L}_\mathcal{A}(\triangleright w \triangleleft)(*) = \top$ if and only if w belongs to the language, and the equality $\mathcal{L}_\mathcal{A}(\triangleright w \triangleleft) = \mathcal{A}(\triangleright w \triangleleft) = \mathcal{A}(\triangleright)\mathcal{A}(w)\mathcal{A}(\triangleleft)$ means that we can decide whether w is in the language by checking whether the state we get in by following w from the initial state is accepting. Such a language could also be seen as a set of relations $\mathcal{L}(\triangleright w \triangleleft) \subseteq 1 \times 1$, where $*$ is related to itself if and only if w belongs to $\mathcal{L}$. The language recognized by a $(\mathbf{Rel}, 1, 1)$-automaton is thus the language recognized by the corresponding non-deterministic automaton.

▶ **Definition 5** (category of automata recognizing a language). *Given a category $\mathcal{C}$ and a language $\mathcal{L} : \mathcal{O} \to \mathcal{C}$, we define the category $\mathbf{Auto}(\mathcal{L})$ whose objects are $(\mathcal{C}, \mathcal{L}(\mathtt{in}), \mathcal{L}(\mathtt{out}))$-automata $\mathcal{A}$ recognizing $\mathcal{L}$, and whose morphisms $\mathcal{A} \to \mathcal{A}'$ are natural transformations whose components on $\mathcal{L}(\mathtt{in})$ and $\mathcal{L}(\mathtt{out})$ are the identity. In other words, a morphism of automata is given by a morphism $f : \mathcal{A}(\mathtt{st}) \to \mathcal{A}'(\mathtt{st})$ in $\mathcal{C}$ such that $\mathcal{A}'(\triangleright) = f \circ \mathcal{A}(\triangleright)$ (it preserves the initial configuration), $\mathcal{A}'(a) \circ f = f \circ \mathcal{A}(a)$ (it commutes with the transitions), and $\mathcal{A}'(\triangleleft) \circ f = \mathcal{A}(\triangleleft)$ (it preserves the output values).*

2.2 Factorization systems and the minimal automaton recognizing a language

▶ **Definition 6** (factorization system). *In a category $\mathcal{C}$, a* factorization system $(\mathcal{E}, \mathcal{M})$ *is the data of a class of $\mathcal{E}$-*morphisms *(represented with $\twoheadrightarrow$) and a class of $\mathcal{M}$-*morphisms *(represented with $\rightarrowtail$) $\mathcal{M}$ such that*

- *every arrow f in $\mathcal{C}$ may be* factored *as $f = m \circ e$ with $m \in \mathcal{M}$ and $e \in \mathcal{E}$;*
- *$\mathcal{E}$ and $\mathcal{M}$ are both stable under composition;*
- *for every commuting diagram as below where $m \in \mathcal{M}$ and $e \in \mathcal{E}$ there is a unique diagonal fill-in $d : Y_1 \to Y_2$ such that $u = d \circ e$ and $v = m \circ d$.*



▶ **Example 7.** In **Set** surjective and injective functions form a factorization system (Surj, Inj) such that a map $f : X \to Y$ factors through its image $f(X) \subseteq Y$. In **Rel** a factorization system is given by $\mathcal{E}$-morphisms those relations $r : X \to Y$ such that every $y \in Y$ is related

to some $x \in X$ by r, and $\mathcal{M}$-morphisms the graphs of injective functions (i.e. $m \in \mathcal{M}$ if and only if there is an injective function f such that $(x, y) \in m \iff y = f(x)$). A relation $r : X \to Y$ then factors through the subset $\{y \in Y \mid \exists x \in X, (x, y) \in R\} \subseteq Y$.

▶ **Lemma 8** (factorization system on $\mathbf{Auto}(\mathcal{L})$ [16, Lemma 2.8])**.** *Given $\mathcal{L} : \mathcal{I} \to \mathcal{C}$ and $(\mathcal{E}, \mathcal{M})$ a factorization system on $\mathcal{C}$, $\mathbf{Auto}(\mathcal{L})$ has a factorization given by those natural transformations whose components are respectively $\mathcal{E}$- and $\mathcal{M}$-morphisms.*

Because of this last result, we use $(\mathcal{E}, \mathcal{M})$ to refer both to a factorization on $\mathcal{C}$ and to its extensions to categories of automata.

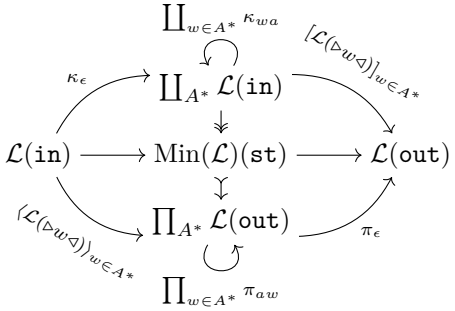
▶ **Definition 9** (minimal object)**.** *When a category $\mathcal{C}$, equipped with a factorization system $(\mathcal{E}, \mathcal{M})$, has both an initial object I and a final object F (for every object X there is exactly one morphism $I \to X$ and one morphism $X \to F$), we define its* $(\mathcal{E}, \mathcal{M})$-minimal object Min *to be the one that $(\mathcal{E}, \mathcal{M})$-factors the unique arrow $I \to F$ as $I \twoheadrightarrow \mathrm{Min} \rightarrowtail F$. For every object X we also define* $\mathrm{Reach}\, X$ *and* $\mathrm{Obs}\, X$ *by the $(\mathcal{E}, \mathcal{M})$-factorizations $I \twoheadrightarrow \mathrm{Reach}\, X \rightarrowtail X$ and $X \twoheadrightarrow \mathrm{Obs}\, X \rightarrowtail F$.*

▶ **Proposition 10** (uniqueness of the minimal object [16, Lemma 2.3])**.** *The minimal object of a category $\mathcal{C}$ is unique up to isomorphism, so that for every other object X,* $\mathrm{Min} \cong \mathrm{Obs}(\mathrm{Reach}\, X) \cong \mathrm{Reach}(\mathrm{Obs}\, X)$*: there are in particular spans $X \leftarrowtail \mathrm{Reach}\, X \twoheadrightarrow \mathrm{Min}$ and co-spans $\mathrm{Min} \rightarrowtail \mathrm{Obs}\, X \twoheadleftarrow X$. It is in that last sense that* Min *is $(\mathcal{E}, \mathcal{M})$-smaller than every other object X, and is thus minimal.*

▶ **Example 11** (initial, final and minimal automata [16, Example 3.1])**.** Since **Set** is complete and cocomplete, the category of $(\mathbf{Set}, 1, 2)$-automata recognizing a language $\mathcal{L} : \mathcal{I} \to \mathbf{Set}$ has an initial, a final and a minimal object. The initial automaton has state-set A^*, initial state $\epsilon \in A^*$, transition functions $\delta_a(w) = wa$ and accepting states the $w \in A^*$ such that w is in $\mathcal{L}$. Similarly, the final automaton has state-set 2^{A^*}, initial state $\mathcal{L}$, transition functions $\delta_a(L) = a^{-1}L$ and accepting states the $L \in 2^{A^*}$ such that ϵ is in L. The minimal automaton for the factorization system of Example 7 thus has the Myhill-Nerode equivalence classes for its states. It is unique up to isomorphism and its $(\mathcal{E}, \mathcal{M})$-minimality ensures that it is the complete deterministic automaton with the smallest state-set that recognizes $\mathcal{L}$: it is in particular finite as soon as $\mathcal{L}$ is recognized by a finite automaton.

On the contrary, there is no good notion of a unique minimal non-deterministic automaton recognizing a regular ($(\mathbf{Rel}, 1, 1)$-) language $\mathcal{L}$. $\mathbf{Auto}(\mathcal{L})$ does have an initial and a final object: the initial automaton is the initial deterministic automaton recognizing $\mathcal{L}$, and the final automaton is the (non-deterministic) transpose of this initial automaton. But there is no factorization system that gives rise to a meaningful minimal object: for instance, the minimal object for the factorization system described in Example 7 has for state-set the set of suffixes of words in $\mathcal{L}$.

Notice how in Example 11 the initial and final $(\mathbf{Set}, 1, 2)$-automata have for respective state-sets A^*, the disjoint union of $|A^*|$ copies of 1, and 2^{A^*}, the cartesian product of $|A^*|$ copies of 2. A similar result holds for non-deterministic automata and generalizes as Theorem 12, itself summarized by the diagram below, where κ and π are the canonical inclusion and projections and $[-]$ and $\langle - \rangle$ are the copairing and pairing of arrows.

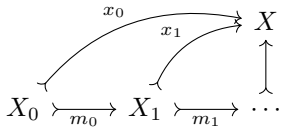


▶ **Theorem 12** ([16, Lemma 3.2]). *Given a countable alphabet A and a language $\mathcal{L} : \mathcal{I} \to \mathcal{C}$,*

- *if $\mathcal{C}$ has all countable copowers of $\mathcal{L}(\mathtt{in})$ then $\mathbf{Auto}(\mathcal{L})$ has an initial object $\mathcal{A}^{init}(\mathcal{L})$ with $\mathcal{A}^{init}(\mathcal{L})(\mathtt{st}) = \coprod_{A^*} \mathcal{L}(\mathtt{in})$;*
- *dually if $\mathcal{C}$ has all countable powers of $\mathcal{L}(\mathtt{out})$ then $\mathbf{Auto}(\mathcal{L})$ has a final object $\mathcal{A}^{final}(\mathcal{L})$ with $\mathcal{A}^{final}(\mathcal{L})(\mathtt{st}) = \prod_{A^*} \mathcal{L}(\mathtt{out})$;*
- *hence when both of the previous items hold and $\mathcal{C}$ comes equipped with a factorization system $(\mathcal{E}, \mathcal{M})$, $\mathbf{Auto}(\mathcal{L})$ has an $(\mathcal{E}, \mathcal{M})$-minimal object* $\mathrm{Min}\,\mathcal{L}$.

We now have all the ingredients to define algorithms for computing the minimal automaton recognizing a language. But since we will also want to prove the termination of these algorithms, we need an additional notion of finiteness.

▶ **Definition 13** ($\mathcal{E}$-artinian and $\mathcal{M}$-noetherian objects [7, Definition 24]). *In a category $\mathcal{C}$ equipped with a factorization system $(\mathcal{E}, \mathcal{M})$, an object X is said to be* $\mathcal{M}$-noetherian *if every strict chain of $\mathcal{M}$-subobjects is finite: if $(x_n : X_n \rightarrowtail X)_{n\in\mathbb{N}}$ and $(m_n : X_n \rightarrowtail X_{n+1})$ form the commutative diagram*



then only finitely many of the m_n's may not be isomorphisms. Dually, X is $\mathcal{E}$-artinian *if X^{op} is $\mathcal{E}^{\mathrm{op}}$-noetherian in $\mathcal{C}^{\mathrm{op}}$, that is if every strict cochain of $\mathcal{E}$-quotients of X is finite.*

While Colcombet, Petrişan and Stabile do not give complexity results for their algorithm, it is straightforward to do so, hence we extend their definition so that it also measures the size of an object in $\mathcal{C}$.

▶ **Definition 14** (co-$\mathcal{E}$- and $\mathcal{M}$-lengths). *For a fixed $x_0 : X_0 \rightarrowtail X$, we call* $\mathcal{M}$-length *of x_0, written* $\mathrm{length}_{\mathcal{M}}\, x_0$*, the (possibly infinite) supremum of the lengths (the number of pairs of consecutive subobjects) of strict chains of $\mathcal{M}$-subobjects of X that start with x_0. Dually, we call* co-$\mathcal{E}$-length *of an $\mathcal{E}$-quotient $x_0 : X \twoheadrightarrow X_0$ the (possibly infinite) quantity* $\mathrm{colength}_{\mathcal{E}}\, x_0 = \mathrm{length}_{\mathcal{E}^{\mathrm{op}}}\, x_0^{\mathrm{op}}$.

▶ **Example 15.** In **Set**, X is finite if and only if it is Inj-noetherian iff it is Surj-artinian, and in that case for $Y \subseteq X$ we have $\mathrm{colength}_{\mathrm{Surj}}(X \twoheadrightarrow Y) = \mathrm{length}_{\mathrm{Inj}}(Y \rightarrowtail X) = |X - Y|$.

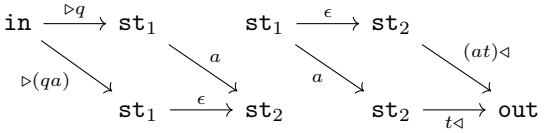
Note that the co-E- and $\mathcal{M}$-lengths need not be equal: see for instance the factorization system we define for monoidal transducers in Section 3.4, for which the co-$\mathcal{E}$- and $\mathcal{M}$-lengths are computed in Lemma 33.

2.3 Learning

In this section, we fix a language $\mathcal{L} : \mathcal{O} \to \mathcal{C}$ and a factorization system $(\mathcal{E}, \mathcal{M})$ of $\mathcal{C}$ that extends to $\mathbf{Auto}(\mathcal{L})$, and we assume that $\mathcal{C}$ has countable copowers of $\mathcal{L}(\mathtt{in})$ and countable powers of $\mathcal{L}(\mathtt{out})$ so that Theorem 12 applies. Our goal is to compute $\operatorname{Min}\mathcal{L}$ with the help of an oracle answering two types of queries: the function $\textsc{Eval}_{\mathcal{L}}$ processes membership queries, and, for a given input word $w \in A^*$, outputs $\mathcal{L}(\triangleright w \triangleleft)$; while $\textsc{Equiv}_{\mathcal{L}}$ processes equivalence queries, and, for a given hypothesis $(\mathcal{C}, \mathcal{L}(\mathtt{in}), \mathcal{L}(\mathtt{out}))$-automaton $\mathcal{A}$, decides whether $\mathcal{A}$ recognizes $\mathcal{L}$, and, if not, outputs a counter-example $w \in A^*$ such that $\mathcal{L}(\triangleright w \triangleleft) \neq (\mathcal{A} \circ \iota)(\triangleright w \triangleleft)$.

For $(\mathbf{Set}, 1, 2)$-automata, if the language is regular this problem is solved using Angluin's L* algorithm [1]. It works by maintaining a set of prefixes Q and of suffixes T and, using $\textsc{Eval}_{\mathcal{L}}$, incrementally building a table $L : Q \times (A \cup \{\epsilon\}) \times T \to 2$ that represents partial knowledge of $\mathcal{L}$ until it can be made into a (minimal) automaton. This automaton is then submitted to $\textsc{Equiv}_{\mathcal{L}}$ when some closure and consistency conditions hold: if the automaton is accepted it must be $\operatorname{Min}\mathcal{L}$, otherwise the counter-example is added to Q and the algorithm loops over. The FunL* algorithm generalizes this to arbitrary $(\mathcal{C}, \mathcal{L}(\mathtt{in}), \mathcal{L}(\mathtt{out}))$, and in particular also encompasses Vilar's algorithm for learning (non-monoidal) transducers, which was described in Section 1 [7].

Instead of maintaining a table, the FunL* algorithm maintains a biautomaton: if $Q \subseteq A^*$ is prefix-closed ($wa \in Q \Rightarrow w \in Q$) and $T \subseteq A^*$ is suffix-closed ($aw \in T \Rightarrow w \in T$), a (Q,T)-*biautomaton* is, similarly to an automaton, a functor $\mathcal{A} : \mathcal{I}_{Q,T} \to \mathcal{C}$, where $\mathcal{I}_{Q,T}$ is now the category freely generated by the graph $\mathtt{in} \xrightarrow{\triangleright q} \mathtt{st}_1 \underset{\epsilon}{\overset{a}{\rightrightarrows}} \mathtt{st}_2 \xrightarrow{t\triangleleft} \mathtt{out}$ where a, q and t respectively range in A, Q and T, and where we also require the diagrams below to commute, the left one whenever $qa \in Q$ and the right one whenever $at \in T$.



A (Q,T)-biautomaton may thus process a prefix in Q and get in a state in $\mathcal{A}(\mathtt{st}_1)$, follow a transition along $A \cup \{\epsilon\}$ to go in $\mathcal{A}(\mathtt{st}_2)$, and output a value for each suffix in T. The category of biautomata recognizing $\mathcal{L}_{Q,T}$ ($\mathcal{L}$ restricted to words in $QT \cup QAT$) is written $\mathbf{Auto}_{Q,T}(\mathcal{L})$. A result similar to Theorem 12 also holds for biautomata [7, Lemma 18], and the initial and final biautomata are then made of finite copowers of $\mathcal{L}(\mathtt{in})$ and finite powers of $\mathcal{L}(\mathtt{out})$ (when these exist). Writing Q/T for the $(\mathcal{E}, \mathcal{M})$-factorization of the canonical morphism $\langle [\mathcal{L}(\triangleright qt \triangleleft)]_{q \in Q} \rangle_{t \in T} : \coprod_Q \mathcal{L}(\mathtt{in}) \to \prod_T \mathcal{L}(\mathtt{out})$, the minimal biautomaton recognizing $\mathcal{L}_{Q,T}$ then has state-spaces $(\operatorname{Min}\mathcal{L}_{Q,T})(\mathtt{st}_1) = Q/(T \cup AT)$ and $(\operatorname{Min}\mathcal{L}_{Q,T})(\mathtt{st}_2) = (Q \cup QA)/T$. The table, represented by the morphism $\langle [\mathcal{L}(\triangleright qt \triangleleft)]_{q \in Q} \rangle_{t \in T}$, may be fully computed using $\textsc{Eval}_{\mathcal{L}}$, and hence so can be the minimal (Q,T)-biautomaton.

A biautomaton $\mathcal{B}$ can then be merged into a hypothesis $(\mathcal{C}, \mathcal{L}(\mathtt{in}), \mathcal{L}(\mathtt{out}))$-automaton precisely when $\mathcal{B}(\epsilon)$ is an isomorphism, i.e. both an $\mathcal{E}$- and an $\mathcal{M}$-morphism (a factorization system necessarily satisfies that $\mathrm{Iso} = \mathcal{E} \cap \mathcal{M}$): this encompasses respectively the closure and consistency conditions that need to hold in the L*-algorithm (and its variants) for the table that is maintained to be merged into a hypothesis automaton.

The FunL* algorithm terminates as soon as $(\operatorname{Min}\mathcal{L})(\mathtt{st})$ is finite, which in this framework is expressed as it being $\mathcal{E}$-artinian and $\mathcal{M}$-noetherian. While Colcombet, Petrişan and Stabile do not give a bound on the actual running time of their algorithm, it would be straightforward to extend their proof to show that the number of updates to Q and T (hence in particular of calls to $\textsc{Equiv}_{\mathcal{L}}$) is linear in the size of $(\operatorname{Min}\mathcal{L})(\mathtt{st})$, itself defined through Definition 14.

Algorithm 1 The FunL*-algorithm.

Input: $\text{Eval}_{\mathcal{L}}$ and $\text{Equiv}_{\mathcal{L}}$
Output: $\text{Min}(\mathcal{L})$

$Q = T = \{\epsilon\}$
loop
  while $\epsilon^{min}_{Q,T}$ is not an isomorphism **do**
    if $\epsilon^{min}_{Q,T}$ is not an $\mathcal{E}$-morphism **then**
      find $qa \in QA$ such that $Q/T \rightarrowtail (Q \cup \{qa\})/T$ is not an $\mathcal{E}$-morphism; add it to Q
    else if $\epsilon^{min}_{Q,T}$ is not an $\mathcal{M}$-morphism **then**
      find $at \in AT$ such that $Q/(T \cup \{at\}) \twoheadrightarrow Q/T$ is not an $\mathcal{M}$-morphism; add it to T
    end if
  end while
  merge $\text{Min}\,\mathcal{L}_{Q,T}$ into $\mathcal{H}_{Q,T}\mathcal{L}$
  if $\text{Equiv}_{\mathcal{L}}(\mathcal{H}_{Q,T}\mathcal{L})$ outputs some counter-example w **then**
    add w and its prefixes to Q
  else
    return $\mathcal{H}_{Q,T}\mathcal{L}$
  end if
end loop

3 The category of monoidal transducers

We now study a specific family of transition systems, monoidal transducers, through the lens of category theory, so as to be able to apply the framework of Colcombet, Petrişan and Stabile. In Section 3.1, we first rapidly recall the notion of monoid. We then define the category of monoidal transducers recognizing a language in Section 3.2, and study how it fits into the framework of Section 2: the initial transducer is given in Corollary 25, conditions for the final transducer to exist are described in Section 3.3, and factorization systems are tackled in Section 3.4.

3.1 Monoids

Let us first recall definitions and notations related to monoids. Most of these are standard in the monoid literature, only coprime-cancellativity (Definition 19) and noetherianity (Definition 20) are uncommon.

▶ **Definition 16** (monoid). *A* monoid $(M, \epsilon_M, \otimes_M)$ *is a set* M *equipped with a binary operation* $\otimes_M$ *(often called the* product*) that is associative* $(\forall \upsilon, \nu, \omega \in M, \upsilon \otimes_M (\nu \otimes_M \omega) = (\upsilon \otimes_M \nu) \otimes_M \omega)$ *and has* ϵ_M *as* unit *element* $(\forall \upsilon \in M, \upsilon \otimes_M \epsilon_M = \epsilon_M \otimes_M \upsilon = \upsilon)$*. When non-ambiguous, it is simply written* $(M, \epsilon, \otimes)$ *or even* M*, and the symbol for the binary operation may be omitted. The* dual *of* $(M, \epsilon, \otimes)$*, written* $(M^{op}, \epsilon^{op}, \otimes^{op})$*, has underlying set* $M^{op} = M$ *and identity* $\epsilon^{op} = \epsilon$*, but symmetric binary operation :* $\forall \upsilon, \nu \in M, \upsilon \otimes^{op} \nu = \nu \otimes \upsilon$.

▶ **Definition 17** (invertibility). *An element* χ *of a monoid* M *is* right-invertible *when there is a* $\chi' \in M$ *such that* $\chi\chi' = \epsilon$*, and* χ' *is then called the* right-inverse *of* χ*.* χ *is* left-invertible *when it is right-invertible in* M^{op}*, and the corresponding right-inverse is called its* left-inverse*. When* χ *is both right- and left-invertible, we say it is* invertible*. In that case its right- and*

left-inverse are equal: this defines its inverse*, written* χ^{-1}*. The set of invertible elements of* M *is written* $M^\times$*. Two families* $(\upsilon_i)_{i\in I}$ *and* $(\nu_i)_{i\in I}$ *indexed by some non-empty set* I *are equal* up to invertibles on the left *when there is some invertible* $\chi \in M$ *such that* $\forall i \in I, \upsilon_i = \chi\nu_i$*.*

▶ **Definition 18** (divisibility)**.** *An element* υ *of a monoid* M left-divides *a family* $\omega = (\omega_i)_{i\in I}$ *of* M *indexed by some set* I *when there is a family* $(\nu_i)_{i\in I}$ *such that* $\forall i \in I, \upsilon\nu_i = \omega_i$*, and we say that* υ *is a* left-divisor *of* ω*.* υ right-divides ω *when it left-divides it in* M^{op}*, and in that case* υ *is called a* right-divisor *of* ω*. A* greatest common left-divisor *(or* left-gcd*) of the family* ω *is a left-divisor of* ω *that is left-divided by all others left-divisors of* ω*. A family* ω *is said to be* left-coprime *if* ϵ_M *is one of its left-gcds, that is if all its left-divisors (or equivalently one of its left-gcds when there is one) are trivial in the sense that they are right-invertible.*

We speak of *greatest* common left-divisors because, while there may be many such elements for a fixed family ω, they all left-divide one another and are thus equivalent in some sense.

▶ **Definition 19** (cancellativity)**.** *A monoid* M *is said to be* left-cancellative *when for any families* $(\upsilon_i)_{i\in I}$ *and* $(\nu_i)_{i\in I}$ *of* M *indexed by some set* I *and for any* $\omega \in M$*,* $\upsilon = \nu$ *as soon as* $\omega\upsilon_i = \omega\nu_i$ *for all* $i \in I$*. If this only implies* $\upsilon_i = \chi\nu_i$ *for some* $\chi \in M^\times$ *that does not depend on* $i \in I$*, we instead say that* M *is* left-cancellative up to invertibles on the left*. Similarly,* M *is said to be* right-coprime-cancellative *when for any* $\upsilon, \nu \in M$ *and any left-coprime family* $(\omega_i)_{i\in I}$ *indexed by some set* I*,* $\upsilon = \nu$ *as soon as* $\upsilon\omega_i = \nu\omega_i$ *for all* $i \in I$*.*

▶ **Definition 20** (noetherianity)**.** *A monoid* M *is* right-noetherian *when for any sequences* $(\upsilon_n)_{n\in\mathbb{N}}$ *and* $(\nu_n)_{n\in\mathbb{N}}$ *of* M *such that* $\nu_n = \nu_{n+1}\upsilon_n$ *for all* $n \in \mathbb{N}$*, there is some* $n \in \mathbb{N}$ *such that* υ_n *is invertible. In this case, we write* $\mathrm{rk}\,\nu$ *for the* rank *of* ν*, the (possibly infinite) supremum of the number of non-invertibles in a sequence* $(\upsilon_n)_{n\in\mathbb{N}}$ *that satisfies* $\nu_n = \nu_{n+1}\upsilon_n$ *for some sequence* $(\nu_n)_{n\in\mathbb{N}}$ *with* $\nu_0 = \nu$*.*

In other words, a monoid is right-noetherian when it has no strict infinite chain of right-divisors (in the definition above, each $\upsilon_n \cdots \upsilon_0$ right-divides ν_0). This condition is the one that will ensure our algorithms terminate (notice the resemblance with Definition 13).

▶ **Example 21.** The canonical example of a monoid is the *free monoid* A^* over an alphabet A, whose elements are words with letters in A, whose product is the concatenation of words and whose unit is the empty word. Notice that the alphabet A may be infinite. The left-divisibility relation is the prefix one, and the left-gcd is the longest common prefix.

The *free commutative monoid* $A^\circledast$ over A has elements the functions $A \to \mathbb{N}$ with finite support, product $(f \otimes g)(a) = f(a)+g(a)$ and unit the zero function $a \mapsto 0$. It is commutative ($f \otimes g = g \otimes f$) hence is its own dual : the divisibility relation is the pointwise order inherited from $\mathbb{N}$ and the greatest common divisor is the pointwise infimum.

These two monoids are examples of *trace monoids* over some A, defined as quotients of A^* by commutativity relations on letters (for $A^\circledast$, all the pairs of letters are required to commute, and for A^* none are). Trace monoids have no non-trivial right- or left-invertible elements, are all left-cancellative, right-coprime-cancellative and right-noetherian, and the rank of a word is simply its number of letters.

Another family of examples is that of *groups*, monoids where all elements are invertible. Again, all groups are left-cancellative, right-coprime-cancellative and right-noetherian.

A final family of monoids of interest is given by $(E, \vee, \bot)$ for E any join-semilattice with a bottom element $\bot$. In these commutative monoids, the divisibility relation is the partial order on E, and the gcd, when it exists, is the infimum. This example shows that a monoid can be coprime-cancellative without being cancellative nor noetherian: this is for instance the case when $E = \mathbb{R}_+$.

3.2 Monoidal transducers as functors

In the rest of this paper we fix a countable *input alphabet* A and an *output monoid* $(M, \epsilon, \otimes)$. To differentiate between elements of A^* and elements of M, we write the former with Latin letters ($a, b, c, \dots$ for letters and $u, v, w, \dots$ for words) and the latter with Greek letters ($\alpha, \beta, \gamma, \dots$ for generating elements and $\upsilon, \nu, \omega, \dots$ for general elements). In particular the empty word over A is denoted e while the unit of M is still written ϵ. We now define our main object of study, M-monoidal transducers.

▶ **Definition 22** (monoidal transducer). *A* monoidal transducer *is a tuple* $(S, (\upsilon_0, s_0), t, (- \odot a)_{a \in A})$ *where* S *is a set of* states*;* $(\upsilon_0, s_0) \in M \times S \sqcup \{\bot\}$ *is the (possibly undefined) pair of the* initialization value *and* initial state*;* $t : S \to M \sqcup \{\bot\}$ *is the partial* termination function*;* $s \odot a \in M \times S \sqcup \{\bot\}$ *for* $a \in A$ *may be undefined, and its two components,* $- \bigcirc a : S \to M \sqcup \{\bot\}$ *and* $- \cdot a : S \to S \sqcup \{\bot\}$*, are respectively called the partial* production function *and the partial* transition function *along* a*.*

▶ **Example 23.** Figure 1a is a graphical representation of a monoidal transducer that takes its input in the alphabet $A = \{a, b\}$ and has output in any monoid that is a quotient of Σ^* with $\Sigma = \{\alpha, \beta, \gamma\}$. Formally, it is given by $S = \{1, 2, 3\}$; $(\upsilon_0, s_0) = (\epsilon, 1)$; $t(1) = t(2) = t(3) = \alpha$; and finally $1 \odot a = (\gamma\alpha\beta, 2)$, $2 \odot a = (\gamma\beta\alpha, 1)$, $3 \odot a = (\gamma\alpha\beta, 3)$, $1 \odot b = 2 \odot b = (\alpha, 3)$ and $3 \odot b = \bot$.

To apply the framework of Section 2 we first need to model monoidal transducers as functors. We thus design a tailored output category that in particular matches the one that instantiates classical transducers when M is a free monoid [16, Section 4]. We write $\mathcal{T}_M$ for the monad on **Set** given by $\mathcal{T}_M X = M \times X + 1 = (M \times X) \sqcup \{\bot\}$ (in Haskell, this monad is the composite of the `Maybe` monad and a `Writer` monad). Its unit $\eta : \mathrm{Id} \Rightarrow \mathcal{T}_M$ is given by $\eta_X(x) = (\epsilon, x)$ and its multiplication $\mu : \mathcal{T}_M^2 \Rightarrow \mathcal{T}_M$ is given by $\mu_X((\upsilon, (\nu, x))) = (\nu\upsilon, x)$, $\mu_X((\upsilon, \bot)) = \bot$ and $\mu_X(\bot) = \bot$. Recall that the Kleisli category $\mathbf{Kl}(\mathcal{T}_M)$ for the monad $\mathcal{T}_M$ has sets for objects and arrows $X \multimap Y$ (notice the different symbol) those functions $f^\dagger : \mathcal{T}_M X \to \mathcal{T}_M Y$ such that $f^\dagger(\bot) = \bot$, $f^\dagger(\upsilon, x) = (\upsilon\nu, y)$ when $f^\dagger(\epsilon, x) = (\nu, y)$ and $f^\dagger(\upsilon, x) = \bot$ when $f(\epsilon, x) = \bot$: in particular, such an arrow is entirely determined by its restriction $f : X \to \mathcal{T}_M Y$, and we will freely switch between these two points of view for the sake of conciseness. The identity on X is then given by the identity function $\mathrm{id}_{\mathcal{T}_M X} = \eta_X^\dagger : \mathcal{T}_M X \to \mathcal{T}_M X$, and the composition of two arrows $X \multimap Y \multimap Z$ is given by the composition of the underlying functions $\mathcal{T}_M X \to \mathcal{T}_M Y \to \mathcal{T}_M Z$.

M-transducers are in one-to-one correspondance with $(\mathbf{Kl}(\mathcal{T}_M), 1, 1)$-automata, i.e. functors $\mathcal{A} : \mathcal{I} \to \mathbf{Kl}(\mathcal{T}_M)$ such that $\mathcal{A}(\mathtt{in}) = \mathcal{A}(\mathtt{out}) = 1$: $(S, (\upsilon_0, s_0), t, (- \odot a)_{a \in A})$ is modelled by the functor $\mathcal{A} : \mathcal{I} \to \mathbf{Kl}(\mathcal{T}_M)$ given by $\mathcal{A}(\mathtt{st}) = S$, $\mathcal{A}(\triangleright) = * \mapsto (\upsilon_0, s_0) : 1 \to M \times S + 1$, $\mathcal{A}(w) = s \mapsto s \odot w = (((s \odot a_1) \odot^\dagger a_2) \odot^\dagger \cdots) \odot^\dagger a_n : S \to M \times S + 1$ for $w = a_1 \cdots a_n \in A^*$, and $\mathcal{A}(\triangleleft) = t : S \to M + 1 \cong M \times 1 + 1$.

▶ **Definition 24.** *We write* $\mathbf{Trans}_M$ *for the category of* $(\mathbf{Kl}(\mathcal{T}_M), 1, 1)$*-automata: the objects are* M*-transducers seen as functors and the morphisms are natural transformations between them. Given a* $(\mathbf{Kl}(\mathcal{T}_M), 1, 1)$*-language* $\mathcal{L} : \mathcal{O} \to \mathbf{Kl}(\mathcal{T}_M)$*, we write* $\mathbf{Trans}_M(\mathcal{L})$ *for the subcategory of* M*-transducers* $\mathcal{A} : \mathcal{I} \to \mathbf{Kl}(\mathcal{T}_M)$ *that recognize* $\mathcal{L}$*, i.e. such that* $\mathcal{A} \circ \iota = \mathcal{L}$*.*

Under this correspondance, the language recognized by a transducer $(S, (\upsilon_0, s_0), t, \odot)$ is thus a function $L : A^* \to M + 1$ given by $L(w) = t^\dagger((\upsilon_0, s_0) \odot^\dagger w)$, and a morphism between two transducers $(S_1, (\upsilon_1, s_1), t_1, \odot_1)$ and $(S_2, (\upsilon_2, s_2), t_2, \odot_2)$ is a function $f : S_1 \to M \times S_2 + 1$ such that $f^\dagger(\upsilon_1, s_1) = (\upsilon_2, s_2)$, $t_1(s) = t_2^\dagger(f(s))$ and $f^\dagger(s \odot a) = f^\dagger(s) \odot^\dagger a$.

When $M = B^*$ for some alphabet B, M-transducers coincide with the classical notion of transducers and the minimal transducer is given by Definition 9 [6, 16]. To study the notion of minimal monoidal transducer, it is thus natural to try to follow this framework as well.

3.3 The initial and final monoidal transducers recognizing a function

To apply the framework of Section 2 to $(\mathbf{Kl}(\mathcal{T}_M), 1, 1)$-automata, we need three ingredients in $\mathbf{Kl}(\mathcal{T}_M)$: countable copowers of 1, countable powers of 1, and a factorization system.

We start with the first ingredient, countable copowers of 1. Since **Set** has arbitrary coproducts, $\mathbf{Kl}(\mathcal{T}_M)$ has arbitrary coproducts as well as any Kleisli category for a monad over a category with coproducts does [18, Proposition 2.2]. Hence Theorem 12 applies:

▶ **Corollary 25** (initial transducer). *For any $(\mathbf{Kl}(\mathcal{T}_M), 1, 1)$-language $\mathcal{L}$, $\mathbf{Trans}_M(\mathcal{L})$ has an initial object $\mathcal{A}^{init}(\mathcal{L})$ with state-set $S^{init} = A^*$, initial state $s_0^{init} = e$, initialization value $\upsilon_0^{init} = \epsilon$, termination function $t^{init}(w) = \mathcal{L}(\triangleright w \triangleleft)(*)$ and transition function $w \odot^{init} a = (\epsilon, wa)$. Given any other transducer $\mathcal{A} = (S, (\upsilon_0, s_0), t, \odot)$ recognizing $\mathcal{L}$, the unique transducer morphism $f : \mathcal{A}^{init}(\mathcal{L}) \Longrightarrow \mathcal{A}$ is given by the function $f : A^* \to M \times S + 1$ such that $f(w) = \mathcal{A}(\triangleright w)(*) = (\upsilon_0, s_0) \odot^\dagger w$.*

Similarly, to get a final transducer in $\mathbf{Trans}_M(\mathcal{L})$ for some $\mathcal{L}$, Theorem 12 tells us that it is enough for $\mathbf{Kl}(\mathcal{T}_M)$ to have all countable powers of 1. This is in particular what happens for classical transducers, when M is a free monoid [16, Lemma 4.7]. Hence we study conditions on the monoid M for $\mathbf{Kl}(\mathcal{T}_M)$ to have these powers.

▶ **Theorem 26.** *If M is both left-cancellative up to invertibles on the left and right-coprime-cancellative, and all non-empty countable subsets of M have a unique left-gcd up to invertibles on the right, then $\mathbf{Kl}(\mathcal{T}_M)$ has all countable powers of 1. This is moreover a necessary condition as soon as M is right-noetherian.*

In practice, to build the final transducer we need some additional technical tools hidden in the proof of Theorem 26. Given a countable set I, we consider partial functions $\Lambda : I \to M+1$. We write $\bot^I$ for the nowhere defined function $i \mapsto \bot$ and $(M+1)^I_* = (M+1)^I - \{\bot^I\}$ for the set of partial functions that are defined somewhere. If $I \subseteq J$, $(M+1)^I_*$ may thus be identified with the subset of partial functions of $(M+1)^J_*$ that are undefined on $J - I$. We extend the product $\otimes : M^2 \to M$ of M to a function $M \times (M+1)^I_* \to (M+1)^I_*$ by setting $(\upsilon \otimes \Lambda)(i) = \upsilon \otimes \Lambda(i)$ for $i \in I$ such that $\Lambda(i) \neq \bot$ and $(\upsilon \otimes \Lambda)(i) = \bot$ otherwise.

$\mathbf{Kl}(\mathcal{T}_M)$ has all countable powers of 1 if and only if there are two functions lgcd : $(M+1)^{\mathbb{N}}_* \to M$ and red : $(M+1)^{\mathbb{N}}_* \to (M+1)^{\mathbb{N}}_*$ that uniquely decompose a partial function into its left-gcd and the corresponding left-coprime reduced function: they satisfy that

- $\Lambda = \mathrm{lgcd}(\Lambda)\,\mathrm{red}(\Lambda)$ for all $\Lambda \in (M+1)^{\mathbb{N}}_*$;
- $\upsilon\,\mathrm{red}(\Gamma) = \nu\,\mathrm{red}(\Lambda)$ implies $\upsilon = \nu$ and $\mathrm{red}\,\Gamma = \mathrm{red}\,\Lambda$ for all $\Gamma, \Lambda \in (M+1)^{\mathbb{N}}_*$ and $\upsilon, \nu \in M$.

We also write $\mathrm{lgcd}(\bot^{\mathbb{N}}) = \bot$, $\mathrm{red}(\bot^{\mathbb{N}}) = \bot^{\mathbb{N}} = \bot$ and do not distinguish between $(\bot, \bot)$ and $\bot$. For every countable I, lgcd and red can be extended to $(M+1)^I_*$ as I may be embedded into $\mathbb{N}$. We then get $\prod_I 1 = \mathrm{Irr}(I, M) = \{\mathrm{red}(\Lambda) \mid \Lambda \in (M+1)^I_*\} \subseteq (M+1)^I_*$.

▶ **Corollary 27.** *When the functions* lgcd *and* red *satisfying the two conditions above exist, the final transducer $\mathcal{A}^{final}(\mathcal{L})$ recognizing a $(\mathbf{Kl}(\mathcal{T}_M), 1, 1)$-language $\mathcal{L}$ exists and has state-set $S^{final} = \mathrm{Irr}(A^*, M)$, initial state $s_0^{final} = \mathrm{red}\,\mathcal{L}$, initialization value $\upsilon_0^{final} = \mathrm{lgcd}\,\mathcal{L}$, termination function $t^{final}(\Lambda) = \Lambda(e)$ and transition functions $\Lambda \odot^{final} a = (\mathrm{lgcd}(a^{-1}\Lambda), \mathrm{red}(a^{-1}\Lambda))$ where we write $(a^{-1}\Lambda)(w) = \Lambda(aw)$ for $a \in A$. Given any other transducer $\mathcal{A} = (S, (\upsilon_0, s_0), t, \odot)$ recognizing $\mathcal{L}$, the unique transducer morphism $f : \mathcal{A} \Longrightarrow \mathcal{A}^{final}(\mathcal{L})$ is given by the function $f : S \to M \times \mathrm{Irr}(A^*, M) + 1$ such that $f(s) = (\mathrm{lgcd}\,\mathcal{L}_s, \mathrm{red}\,\mathcal{L}_s)$ where $\mathcal{L}_s(\triangleright w \triangleleft)(*) = \mathcal{A}(w \triangleleft)(s)$ is the function recognized by $\mathcal{A}$ from the state s.*

▶ **Example 28.** When M is a group it is cancellative (because every element is invertible) and all countable families have a unique left-gcd up to invertibles on the right (ϵ itself) hence Theorem 26 applies and $\mathbf{Trans}_M(\mathcal{L})$ always has a final object. The same is true when M is a trace monoid (the left-gcd then being the longest common prefix, whose existence is guaranteed by [9, Proposition 1.3]).

Conversely, the monoids given by join semi-lattices are not left-cancellative up to invertibles in general. In $\mathbb{R}_+$ for instance, there are ways to define the functions lgcd and red but they may not satisfy that $\mathrm{red}(\upsilon\Lambda) = \mathrm{red}\,\Lambda$ (which should be implied by $\upsilon\,\mathrm{lgcd}(\Lambda)\,\mathrm{red}(\Lambda) = \upsilon\Lambda = \mathrm{lgcd}(\upsilon\Lambda)\,\mathrm{red}(\upsilon\Lambda)$). This is expected, as there may be several non-isomorphic ways to minimize transducers with outputs in these monoids, which is incompatible with the framework of Definition 9.

Theorem 26 provides sufficient conditions that are reminiscent of those developed in [12] for the minimization of monoidal transducers. These conditions are stronger than ours but still similar: the output monoid is assumed to be both left- and right-cancellative, which in particular implies the unicity up to invertibles on the right of the left-gcd whose existence is also assumed. They do only require the existence of left-gcds for finite families (whereas we ask for left-gcds of countable families), which would not be enough for our sake since the categorical framework also encompasses the existence of minimal (infinite) automata for non-regular languages, but in practice our algorithms will only use binary left-gcds as well. We conjecture that, when only those binary left-gcds exist, the existence of a unique minimal transducer is explained categorically by the existence of a final transducer in the category of transducers whose states all recognize functions that are themselves recognized by finite transducers. Where the two sets of conditions really differ is in the conditions required for the termination of the algorithms: where we will require right-noetherianity of M, they require that if some ν left-divides both some ω and $\upsilon\omega$ for some υ, then ν should also left-divide $\upsilon\nu$. This last condition leads to better complexity bounds than right-noetherianity, but misses some otherwise simple monoids that satisfy right-noetherianity, e.g. $\{\alpha, \beta\}^*$ but where we also let α and β^2 commute. Conversely, Gerdjikov's main non-trivial example, the tropical monoid $(\mathbb{R}_+, 0, +)$, is not right-noetherian. It can still be dealt with in our context by considering submonoids (finitely) generated by the output values of a finite transducer's transitions, these monoids themselves being right-noetherian.

3.4 Factorization systems

The last ingredient we need in order to be able to apply the framework of Section 2 is a factorization system on $\mathbf{Trans}_M(\mathcal{L})$. By Lemma 8, it is enough to find one on $\mathbf{Kl}(\mathcal{T}_M)$. When M is a free monoid, [16, Section 4.5], provides such a factorization system for which the minimal $(\mathbf{Kl}(\mathcal{T}_M), 1, 1)$-automaton recognizing a language is the usual notion of minimal (non-monoidal) transducer. We thus simply generalize this factorization system to arbitrary output monoids. Define the classes of $\mathbf{Kl}(\mathcal{T}_M)$-morphisms Surj, Inj, Inv and Tot as follows: for $f : X \to M \times Y + 1$ and writing $f_1 : X \to M + 1$ for its projection on M and $f_2 : X \to Y + 1$ for its projection on Y, we let $f \in \mathrm{Surj}$ whenever f_2 is surjective on Y, and say f is surjective; let $f \in \mathrm{Inj}$ whenever f_2 is injective when corestricted to Y, and say f is injective; let $f \in \mathrm{Tot}$ whenever it is total, i.e. $f(x) \neq \bot$ for all $x \in X$; and let $f \in \mathrm{Inv}$ whenever it only produces invertible elements, i.e. whenever $f_1(X) \subseteq M^\times + 1$. Then,

▶ **Lemma 29.** $(\mathcal{E}, \mathcal{M}) = (\mathrm{Surj}, \mathrm{Inj} \cap \mathrm{Inv} \cap \mathrm{Tot})$ *is a factorization system on* $\mathbf{Kl}(\mathcal{T}_\mathcal{M})$.

Theorem 12 and Proposition 10 show that $(\mathcal{E}, \mathcal{M})$ indeed gives rise to a useful notion of minimal transducer.

▶ **Corollary 30.** *When* $\mathbf{Kl}(\mathcal{T}_M)$ *has all countable powers of* 1*, the* $(\mathcal{E},\mathcal{M})$*-minimal transducer recognizing a* $(\mathbf{Kl}(\mathcal{T}_M),1,1)$*-language* $\mathcal{L}$ *is well-defined and has state-set* $S^{min} = \{\mathrm{red}(w^{-1}\mathcal{L}) \mid w \in A^*\} \cap (M+1)_*^{A^*}$*, initial state* $s_0^{min} = \mathrm{red}\,\mathcal{L}$*, initialization value* $v_0^{min} = \mathrm{lgcd}\,\mathcal{L}$*, termination function* $t^{min}(\Lambda) = \Lambda(e)$ *and transition functions* $\mathrm{red}(w^{-1}\mathcal{L}) \odot^{min} a = (\mathrm{lgcd}((wa)^{-1}\mathcal{L}), \mathrm{red}((wa)^{-1}\mathcal{L}))$*. It is characterized by the property that all its states are reachable from the initial state and recognize distinct left-coprime functions.*

▶ **Example 31.** When $M = \{\alpha,\beta,\gamma\}^*$ is the free monoid, the transducer of Figure 1a is minimal for the function it recognizes. But when we allow α and β to commute in M this transducer is not minimal anymore because the states 1 and 2 recognize the same left-coprime function and need to be merged. If we also allow α to commute with γ then the resulting transducer (where the states 1 and 2 have been merged) is again not minimal, because the function recognized by the merged state (and previously by the states 1 and 2) is not left-coprime: it is left-divisible by α.

More interestingly, $(\mathrm{Surj}\cap\mathrm{Inj}\cap\mathrm{Tot},\mathrm{Inv})$ and $(\mathrm{Surj}\cap\mathrm{Inj},\mathrm{Tot}\cap\mathrm{Inv})$ are also factorization systems on $\mathbf{Kl}(\mathcal{T}_M)$, and since $\mathrm{Surj}\cap\mathrm{Inv}\cap\mathrm{Tot} \subset \mathrm{Surj}\cap\mathrm{Inj} \subset \mathrm{Surj}$, together with $(\mathcal{E},\mathcal{M})$ they thus form what is called a *quaternary factorization system*. In practice, this means that the computation of any morphism f in $\mathbf{Kl}(\mathcal{T}_M)$ can be uniquely factored into four orthogonal parts, as $f = f_4 \circ f_3 \circ f_2 \circ f_1$: f_1 is the part that forgets some inputs (it belongs to $\mathrm{Surj}\cap\mathrm{Inj}\cap\mathrm{Inv}$ but need not belong to Tot), f_2 the one that produces non-invertible elements of the output monoid (it belongs to $\mathrm{Surj}\cap\mathrm{Inj}\cap\mathrm{Tot}$ but need not belong to Inv), f_3 the one that merges some inputs together (it belongs to $\mathrm{Surj}\cap\mathrm{Inv}\cap\mathrm{Tot}$ but need not belong to Inj) and f_4 embeds the result into a bigger set (it belongs to $\mathrm{Inv}\cap\mathrm{Inj}\cap\mathrm{Tot}$ but need not belong to Surj).

In particular, for a $(\mathbf{Kl}(\mathcal{T}_M),1,1)$-automaton $\mathcal{A}$ recognizing a language $\mathcal{L}$, the final arrow $\mathrm{Reach}\,\mathcal{A} \multimap \mathcal{A}^{final}(\mathcal{L})$ can be factored in this way into $\mathrm{Reach}\,\mathcal{A} \multimap\!\!\!\!\rightarrow \mathrm{Total}\,\mathcal{A} \multimap\!\!\!\!\rightarrow \mathrm{Prefix}\,\mathcal{A} \multimap\!\!\!\!\rightarrow \mathrm{Min}\,\mathcal{L} \rightarrowtail\!\!\multimap \mathcal{A}^{final}(\mathcal{L})$, and we retrieve the four main steps of every algorithm that minimizes (monoidal or non-monoidal) transducers [6, 12]. In practice, minimizing $\mathcal{A} = (S,(u_0,s_0),t,\odot)$ indeed involves computing $\mathrm{Reach}\,\mathcal{A}$, $\mathrm{Total}\,\mathcal{A}$, $\mathrm{Prefix}\,\mathcal{A}$ and $\mathrm{Min}\,\mathcal{L}$ one after the other:

- $\mathrm{Reach}\,\mathcal{A}$ has state-set the set of states in S that are reachable from s_0;
- $\mathrm{Total}\,\mathcal{A}$ has state-set S' the set of states in S that recognize a function defined for at least one input word (in particular if $\mathcal{A}$ recognizes $\bot^{A^*}$ then (v_0,s_0) is set to $\bot$);
- $\mathrm{Prefix}\,\mathcal{A} = (S',(v_0\,\mathrm{lgcd}(\mathcal{L}_{s_0}),s_0),t',\odot')$, where $\mathcal{L}_s$ is the function recognized from a state $s \in S$ in $\mathcal{A}$, is obtained from $\mathrm{Total}\,\mathcal{A}$ by setting $t'(s) = \mathrm{lgcd}(\mathcal{L}_s)^{-1}t(s)$ and $s \odot' a = (\mathrm{lgcd}(\mathcal{L}_s)^{-1}(s \circ a)\,\mathrm{lgcd}(\mathcal{L}_{s\cdot a}), s\cdot a)$;
- $\mathrm{Obs}(\mathrm{Reach}\,A) \cong \mathrm{Min}\,\mathcal{L}$ is obtained from $\mathrm{Prefix}\,\mathcal{A}$ by merging two states s_1 and s_2 whenever they recognize functions that are equal up to invertibles on the left in $\mathrm{Prefix}\,\mathcal{A}$, that is when $\mathrm{red}(\mathcal{L}_{s_1}) = \mathrm{red}(\mathcal{L}_{s_2})$ in $\mathcal{A}$.

▶ **Example 32.** Starting from the monoidal transducer $\mathcal{A}$ of Figure 1a considered within the output monoid where we allow α to commute with both β and γ, $\mathrm{Reach}\,\mathcal{A} = \mathcal{A}$ (every state is reachable from the initial state), $\mathrm{Total}\,\mathcal{A} = \mathcal{A}$ (every state produces an output when following at least one input word, e.g. the empty input word), $\mathrm{Prefix}\,\mathcal{A}$ is obtained from $\mathcal{A}$ by pulling the output letter α from the terminal transitions to the initial transition (because α commutes with every output word hence left-divides the functions recognized from each state), and $\mathrm{Min}\,\mathcal{L}$ is then obtained from $\mathrm{Prefix}\,\mathcal{A}$ by merging the states 1 and 2 (they recognize the same function because $\gamma\alpha\beta = \gamma\beta\alpha$).

4 Active learning of minimal monoidal transducers

Let M be a monoid satisfying the conditions of Theorem 26 and consider now a function $A^* \to M+1$ seen as a $(\mathbf{Kl}(\mathcal{T}_M), 1, 1)$-language $\mathcal{L}$. Theorem 12 tells us that the minimal M-transducer recognizing $\mathcal{L}$ exists, is unique up to isomorphism and is given by Corollary 30, but does not tell us whether this minimal transducer is computable. For this to hold we need that the product in M, the left-gcd of two elements in M – written $\wedge$ – and the function LeftDivide – that takes as input $\delta, \upsilon \in M$ and outputs a ν such that $\upsilon = \delta\nu$ or fails if there is none – be all computable, and that equality up to invertibles on the left be decidable (and that the corresponding invertible be computable as well). We extend these operations to $M+1$ by means of $u\bot = \bot u = \bot$, $u \wedge \bot = \bot \wedge u = \bot$ and $\textsc{LeftDivide}(\delta, \bot) = \bot$. For the computations to terminate we additionally require that $\operatorname{Min}\mathcal{L}$ have finite state-set and M be right-noetherian, so that $(\operatorname{Min}\mathcal{L})(\mathtt{st})$ is noetherian for the factorization system $(\mathcal{E}, \mathcal{M})$ of Lemma 29:

▶ **Lemma 33.** *An object X of $\mathbf{Kl}(\mathcal{T}_M)$ is $\mathcal{M}$-noetherian if and only if it is a finite set, in which case* $\operatorname{length}_{\mathcal{M}}(m : Y \rightarrowtail\!\!\!\!\circ\!\!\rightarrow X) = |X| - |Y|$*, and it is $\mathcal{E}$-artinian if and only if it is a finite set and either M is right-noetherian or $X = \varnothing$, in which case* $\operatorname{colength}_{\mathcal{E}}(e : X \twoheadrightarrow\!\!\!\!\circ\!\!\rightarrow Y) = |X| - |Y| + \sum_{e(x)=(\upsilon,y)} \operatorname{rk}\upsilon$.

The categorical framework of Section 2 can be extended with an abstract minimization algorithm [2]. With the output category described in Section 3 this instantiates in particular Gerdjikov's algorithm for minimizing monoidal transducers [12], and even shows that the latter is still valid under the conditions discussed in Section 3.3 and terminates as soon as M is right-noetherian. However, we focus here on a second way to compute the minimal transducer recognizing $\mathcal{L}$, namely learning it through membership and equivalence queries, that is relying on a function $\textsc{Eval}_{\mathcal{L}}$ that outputs the value of $\mathcal{L}$ on input words and a function $\textsc{Equiv}_{\mathcal{L}}$ that checks whether the hypothesis transducer is $\operatorname{Min}\mathcal{L}$ or outputs a counterexample otherwise. The FunL* algorithm described in Section 2.3 instantiates such an algorithm, that terminates as soon as $(\operatorname{Min}\mathcal{L})(\mathtt{st})$ is $(\mathcal{E}, \mathcal{M})$-noetherian. We now give a practical description of this categorical algorithm: we explain how to keep track of the minimal biautomaton and how to check whether $\epsilon^{min}_{Q,T}$ is in $\mathcal{E}$ and $\mathcal{M}$. This is summarized by Algorithm 2 in the appendix.

The algorithm for learning the minimal monoidal transducer recognizing $\mathcal{L}$ is very similar to Vilar's algorithm (described in Section 1), the main difference being that the longest common prefix is now the left-gcd and that, in some places, testing for equality is now testing for equality up to invertibles on the left. It maintains two sets Q and T that are respectively prefix-closed and suffix-closed, and tables $\Lambda : Q \times (A \cup \{e\}) \to M+1$ and $R : Q \times (A \cup \{e\}) \times T \to M+1$. They satisfy that, for all $a \in A \cup \{e\}$, $\Lambda(q,a)R(q,a,t) = \mathcal{L}(\triangleright qat \triangleleft)$ and $R(q,a,\cdot)$ is left-coprime, hence $\Lambda(q,a)$ is a left-gcd of $(\mathcal{L}(\triangleright qat\triangleleft))_{t\in T}$. The algorithm then extends Q and T until some closure and consistency conditions are satisfied, and builds a hypothesis transducer $\mathcal{H}(Q,T)$ using Λ and R: its state-set S can be constructed by, starting with $e \in Q$, picking as many $q \in Q$ such that $R(q,e,\cdot)$ is not $\bot^T$ and such that, for any other $q' \in S$, $R(q,e,\cdot)$ and $R(q',e,\cdot)$ are not equal up to invertibles on the left; it then has initial state $e \in Q$, initialization value $\Lambda(e,e)$, termination function $t = q \in S \mapsto R(q,e,e)$ and transition functions given by $q \odot a = (\textsc{LeftDivide}(\Lambda(q,e), \Lambda(q,a))\chi, q')$ for $q, q' \in S$ such that $R(q,a,\cdot) = \chi R(q',e,\cdot)$. The algorithm then adds the counter-example given by $\textsc{Equiv}_{\mathcal{L}}(\mathcal{H}(Q,T))$ to Q and builds a new hypothesis automaton until no counter-example is returned and $\mathcal{H}(Q,T) = \operatorname{Min}\mathcal{L}$.

Closure issues happen when $\epsilon^{min}_{Q,T}$ is not in $\mathcal{E} = \mathrm{Surj}$, that is when there is a $qa \in QA$ such that $R(q,a,\cdot) \neq \chi R(q',e,\cdot)$ for every other $q' \in Q$ and $\chi \in M^\times$, and in that case qa should be added to Q. Consistency issues happen when the $\mathcal{E}$-factor of $\epsilon^{min}_{Q,T}$ is not in $\mathcal{M} = \mathrm{Inj} \cap \mathrm{Inv} \cap \mathrm{Tot}$, i.e. if it is not in Tot, in Tot but not in $\mathrm{Inv} \cap \mathrm{Tot}$, or in $\mathrm{Inv} \cap \mathrm{Tot}$ but not in $\mathrm{Inj} \cap \mathrm{Inv} \cap \mathrm{Tot}$: the quaternary factorization system described in Section 3.4 thus also explains the different kinds of consistency issues we may face. In practice, there is hence a consistency issue if there is an $at \in AT$ such that respectively: either there is a $q \in Q$ such that $R(q,a,t) \neq \bot$ but $R(q,e,T) = \bot^T$; or there is a $q \in Q$ such that $\Lambda(q,e)$ does not left-divide $\Lambda(q,a)R(q,a,t)$; or there are some $q,q' \in Q$ and $\chi \in M^\times$ such that $R(q,e,T) = \chi R(q',e,T)$ but $\textsc{LeftDivide}(\Lambda(q,e), \Lambda(q,a)R(q,a,t)) \neq \chi \textsc{LeftDivide}(\Lambda(q',e), \Lambda(q',a)R(q',a,t))$. In each of these cases at should be added to T.

The number of updates to Q and T, hence in particular of calls to $\textsc{Equiv}_{\mathcal{L}}$, is bounded linearly by the the number of states in $\mathrm{Min}\,\mathcal{L}$ and the sum of the ranks of the left-gcds of the functions recognized by each of these states (although this latter quantity is not necessarily finite). Our algorithm also differs from Vilar's original one in a small additional way: the latter also keeps track of the left-gcds of every $\Lambda(q,\tilde{a})$ where $\tilde{a}$ ranges over $A \cup \{e\}$ and $q \in Q$ is fixed, and checks for consistency issues accordingly. This is a small optimization of the algorithm that does not follow immediately from the categorical framework. In Section 1 we thus actually provided an example run of our version of the algorithm when the output monoid is a free monoid. This also provides example runs of our algorithm for non-free output monoids, as quotienting the output monoid will only remove closure and consistency issues and make the run simpler. For instance letting α commute with β for the transducer of Figure 1a would have removed the closure issue and the need to add a to Q while learning the corresponding monoidal transducer, and letting α also commute with γ would have removed the first consistency issue to arise and the need to add a to T.

5 Summary and future work

In this work, we instantiated Colcombet, Petrişan and Stabile's active learning categorical framework with monoidal transducers. We gave some simple sufficient conditions on the output monoid for the minimal transducer to exist and be unique, which in particular extend Gerdjikov's conditions for minimization to be possible [12]. Finally, we described what the active learning algorithm of the categorical framework instantiated to in practice under these conditions, relying in particular on the quaternary factorization system in the output category.

This work was mainly a theoretical excursion and was not motivated by practical examples where monoidal transducers are used. One particular application that could be further explored is the use of transducers with outputs in trace monoids (and their learning) to programatically schedule jobs, as mentioned in the introduction. We also leave the search for other interesting examples for future work.

Some intermediate results of this work go beyond what the categorical framework currently provides and could be generalized. The use of a quaternary factorization system (or any n-ary factorization system) would split the algorithms into several substeps that should be easier to work with. Here our factorization systems seemed to arise as the image of the factorization system on **Set** through the monad $\mathcal{T}_M$; generalizing this to other monads could provide meaningful examples of factorization systems in any Kleisli category. Finally, we mentioned in Section 3.3 that a problem with the current framework is that it may only account for the minimization of both finite and infinite transition systems at the same time,

and conjectured that we could restrict to only the finite case by working in a subcategory of well-behaved transducers: this subcategory is perhaps an instance of a general construction that has its own version of Theorem 12, so as to still have a generic way to build the initial, final and minimal objects.

References

1. Dana Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75(2):87–106, November 1987. doi:10.1016/0890-5401(87)90052-6.
2. Quentin Aristote. Functorial approach to minimizing and learning deterministic transducers with outputs in arbitrary monoids, November 2023. URL: https://ens.hal.science/hal-04172251v2.
3. Simone Barlocco, Clemens Kupke, and Jurriaan Rot. Coalgebra Learning via Duality. In Mikołaj Bojańczyk and Alex Simpson, editors, *Foundations of Software Science and Computation Structures*, Lecture Notes in Computer Science, pages 62–79, Cham, 2019. Springer International Publishing. doi:10.1007/978-3-030-17127-8_4.
4. F. Bergadano and S. Varricchio. Learning behaviors of automata from multiplicity and equivalence queries. In M. Bonuccelli, P. Crescenzi, and R. Petreschi, editors, *Algorithms and Complexity*, Lecture Notes in Computer Science, pages 54–62, Berlin, Heidelberg, 1994. Springer. doi:10.1007/3-540-57811-0_6.
5. Francesco Bergadano and Stefano Varricchio. Learning Behaviors of Automata from Multiplicity and Equivalence Queries. *SIAM Journal on Computing*, 25(6):1268–1280, December 1996. doi:10.1137/S009753979326091X.
6. Christian Choffrut. Minimizing subsequential transducers: A survey. *Theoretical Computer Science*, 292(1):131–143, January 2003. doi:10.1016/S0304-3975(01)00219-5.
7. Thomas Colcombet, Daniela Petrişan, and Riccardo Stabile. Learning automata and transducers: A categorical approach, October 2020. doi:10.48550/arXiv.2010.13675.
8. Thomas Colcombet, Daniela Petrişan, and Riccardo Stabile. Learning Automata and Transducers: A Categorical Approach. In Christel Baier and Jean Goubault-Larrecq, editors, *29th EACSL Annual Conference on Computer Science Logic (CSL 2021)*, volume 183 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 15:1–15:17, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.CSL.2021.15.
9. Robert Cori and Dominique Perrin. Automates et commutations partielles. *RAIRO. Informatique théorique*, 19(1):21–32, 1985. doi:10.1051/ita/1985190100211.
10. Jason Eisner. Simpler and more general minimization for weighted finite-state automata. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology – Volume 1*, NAACL '03, pages 64–71, USA, May 2003. Association for Computational Linguistics. doi:10.3115/1073445.1073454.
11. Charles N. Fischer, Ron K. Cytron, and Richard J. LeBlanc. *Crafting a Compiler*. Crafting a Compiler with C. Addison-Wesley, Boston, 2010.
12. Stefan Gerdjikov. A General Class of Monoids Supporting Canonisation and Minimisation of (Sub)sequential Transducers. In Shmuel Tomi Klein, Carlos Martín-Vide, and Dana Shapira, editors, *Language and Automata Theory and Applications*, Lecture Notes in Computer Science, pages 143–155, Cham, 2018. Springer International Publishing. doi:10.1007/978-3-319-77313-1_11.
13. Ronald M. Kaplan and Martin Kay. Regular Models of Phonological Rule Systems. *Computational Linguistics*, 20(3):331–378, 1994. URL: https://aclanthology.org/J94-3001.
14. Kevin Knight and Jonathan May. Applications of Weighted Automata in Natural Language Processing. In Manfred Droste, Werner Kuich, and Heiko Vogler, editors, *Handbook of Weighted Automata*, Monographs in Theoretical Computer Science. An EATCS Series, pages 571–596. Springer, Berlin, Heidelberg, 2009. doi:10.1007/978-3-642-01492-5_14.

15 Saunders Mac Lane. *Categories for the Working Mathematician*, volume 5 of *Graduate Texts in Mathematics*. Springer, New York, NY, 1978. doi:10.1007/978-1-4757-4721-8.

16 Daniela Petrişan and Thomas Colcombet. Automata Minimization: A Functorial Approach. *Logical Methods in Computer Science*, Volume 16, Issue 1, March 2020. doi:10.23638/LMCS-16(1:32)2020.

17 M. P. Schützenberger. On the definition of a family of automata. *Information and Control*, 4(2):245–270, September 1961. doi:10.1016/S0019-9958(61)80020-X.

18 Jenö Szigeti. On limits and colimits in the Kleisli category. *Cahiers de topologie et géométrie différentielle*, 24(4):381–391, 1983. URL: http://www.numdam.org/item/?id=CTGDC_1983__24_4_381_0.

19 Henning Urbat and Lutz Schröder. Automata Learning: An Algebraic Approach. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '20, pages 900–914, New York, NY, USA, July 2020. Association for Computing Machinery. doi:10.1145/3373718.3394775.

20 Gerco van Heerdt, Matteo Sammartino, and Alexandra Silva. Learning Automata with Side-Effects. In Daniela Petrişan and Jurriaan Rot, editors, *Coalgebraic Methods in Computer Science*, Lecture Notes in Computer Science, pages 68–89, Cham, 2020. Springer International Publishing. doi:10.1007/978-3-030-57201-3_5.

21 Juan Miguel Vilar. Query learning of subsequential transducers. In Laurent Miclet and Colin de la Higuera, editors, *Grammatical Interference: Learning Syntax from Sentences*, Lecture Notes in Computer Science, pages 72–83, Berlin, Heidelberg, 1996. Springer. doi:10.1007/BFb0033343.

A Why Vilar's algorithm is not enough for monoidal transducers

▶ **Lemma 34.** *Let $A = \{a\}$, $\Sigma = \{\alpha, \beta, \gamma\}$, let $\Sigma^*/\!\sim$ be the monoid given by the presentation $\langle \Sigma \mid \alpha\beta = \beta\alpha \rangle$ and let $\pi : \Sigma^* \to \Sigma^*/\!\sim$ be the corresponding quotient. Consider the function $f : A^* \to \Sigma^*/\!\sim$ that maps a^n to $\alpha^n\beta^n\gamma = (\alpha\beta)^n\gamma$.*

f is recognized by a finite transducer with outputs in $\Sigma^*/\!\sim$, yet learning a transducer that recognizes any function $f' : A^* \to \Sigma^*$ such that $f = f' \circ \pi$ with Vilar's algorithm will never terminate if the oracle replies to the membership query for a^n with $\alpha^n\beta^n\gamma \in \Sigma^*$ (which differs from $(\alpha\beta)^n\gamma$ in Σ^* but not in $\Sigma^*/\!\sim$).

Proof. f is recognized by the $(\Sigma^*/\!\sim)$-transducer (Definition 22) with one state s that is initial, initial value $\upsilon_0 = \epsilon$, transition function $a \odot s = (\alpha\beta, s)$ and termination function $t(s) = \gamma$.

Consider now the run of Vilar's algorithm (Algorithm 2 with $M = \Sigma^*$ and $M^\times = \{e\}$) with the oracle answering the membership query for a^n with $f'(a^n) = \alpha^n\beta^n\gamma$. We start with $Q = T = \{e\}$, $\Lambda(e) = \gamma$, $\Lambda(a) = \alpha\beta\gamma$ and $R(e, e) = R(a, e) = \epsilon$, where $\Lambda(w)$ for $w \in Q \cup QA$ is the longest common prefix of the $\{f'(wt) \mid t \in T\}$ and $R(w, t)$ is the suffix such that $f'(wt) = \Lambda(w)R(w, t)$.

Since $\Lambda(e) = \gamma$ is not a prefix of $\Lambda(a) = \alpha\beta\gamma$, there is a consistency issue and we add a to T. Taking $n = 0$, we are now in the configuration C_n given by $Q = Q_n = \{a^k \mid k \leq n\}$, $T = \{e, a\}$, and $\Lambda(a^k) = \alpha^k$, $R(a^k, e) = \beta^k\gamma$ and $R(a^k, a) = \alpha\beta^{k+1}\gamma$ for every $k \leq n + 1$ (since the oracle replies to the membership query for a^k with $\alpha^k\beta^k\gamma$ and to that for $a^k a$ with $\alpha^{k+1}\beta^{k+1}\gamma$).

Suppose now we are in the configuration C_n for some $n \in \mathbb{N}$. Then there is a closure issue, since for all $k \leq n$, $R(a^k, e) = \beta^k\gamma \neq \beta^{n+1}\gamma = R(a^{n+1}, e)$. We thus add a^{n+1} to Q. To compute $\Lambda(a^{n+2})$, $R(a^{n+2}, e)$ and $R(a^{n+2}, a)$, we make a membership query for a^{n+3}: the oracle answers with $f'(a^{n+3}) = \alpha^{n+3}\beta^{n+3}\gamma$. The longest common prefix of

$f'(a^{n+2}) = \Lambda(a^{n+1})R(a^{n+1}, a) = \alpha^{n+2}\beta^{n+2}\gamma$ and $f'(a^{n+2}a) = \alpha^{n+3}\beta^{n+3}\gamma$ is thus α^{n+2}, and the corresponding suffixes are $R(a^{n+2}, e) = \beta^{n+2}\gamma$ and $R(a^{n+2}, a) = \alpha\beta^{n+3}\gamma$: we are now in the configuration C_{n+1}.

Hence the run of the algorithm never terminates and never even reaches an equivalence query, as it must first go through all the configurations C_n for $n \in \mathbb{N}$. ◀

B The learning algorithm

Algorithm 2 The FunL*-algorithm for monoidal transducers.

Input: $\text{Eval}_{\mathcal{L}}$ and $\text{Equiv}_{\mathcal{L}}$
Output: $\text{Min}_M(\mathcal{L})$

$Q = T = \{e\}$
for $a \in A \cup \{e\}$ **do**
  $\Lambda(e, a) = \text{Eval}_{\mathcal{L}}(a)$
  $R(e, a, e) = \epsilon$
end for
loop
  if there is a $qa \in QA$ such that $\forall q' \in Q, \chi \in M^\times, R(q, a, \cdot) \neq \chi R(q', e, \cdot)$ **then**
    add qa to Q
  else if there is an $at \in AT$ such that
  - **either** there is a $q \in Q$ such that $R(q, a, t) \neq \bot$ but $R(q, e, T) = \bot^T$;
  - **or** there is a $q \in Q$ such that $\Lambda(q, e)$ does not left-divide $\Lambda(q, a)R(q, a, t)$;
  - **or** there are $q, q' \in Q$ and $\chi \in M^\times$ such that $R(q, e, T) = \chi R(q', e, T)$ but $\text{LeftDivide}(\Lambda(q, e), \Lambda(q, a)R(q, a, t)) \neq \chi\text{LeftDivide}(\Lambda(q', e), \Lambda(q', a)R(q', a, t))$

  then
    add at to T
  else
    build $\mathcal{H}(Q, T) = (S, (\upsilon_0, s_0), t, \odot)$ given by:
  - $S \subseteq Q$ is built by starting with $e \in S$ and adding as many $q \in Q$ as long as $R(q, e, \cdot) \neq \bot$ and $\forall q' \in S, \chi \in M^\times, R(q, e, \cdot) \neq \chi R(q', e, \cdot)$;
  - $(\upsilon_0, s_0) = (\Lambda(e), e)$
  - $q \odot a = (\text{LeftDivide}(\Lambda(q, e), \Lambda(q, a))\chi, q')$ with $q \in S$, $\chi \in M^\times$ given by $R(q, a, \cdot) = \chi R(q', e, \cdot)$
  - $t(q) = R(q, e, e)$

    if $\text{Equiv}_{\mathcal{L}}(\mathcal{H}_{Q,T}(\mathcal{L}))$ outputs some counter-example w **then**
      add w and its prefixes to Q
    else
      return $\mathcal{H}(Q, T)$
    end if
  end if
  update Λ and R using $\text{Eval}_{\mathcal{L}}$
end loop

If $\mathcal{A}$ is a monoidal transducer, write $|\mathcal{A}|_{\mathtt{st}} = |\mathcal{A}(\mathtt{st})|$ and $\text{rk}(\mathcal{A}) = \sum_{s \in \mathcal{A}(\mathtt{st})} \text{rk}(\text{lgcd}(\mathcal{L}_s))$ (where $\mathcal{L}_s$ is the partial function recognized by $\mathcal{A}$ when $s \in \mathcal{A}(\mathtt{st})$ is chosen to be the initial state).

▶ **Theorem 35.** *Algorithm 2 is correct and terminates as soon as* $\operatorname{Min}\mathcal{L}$ *has finite state-set and* M *is right-noetherian. It makes at most* $3\left|\operatorname{Min}\mathcal{L}\right|_{\mathtt{st}} + \operatorname{rk}(\operatorname{Min}\mathcal{L})$ *updates to* Q *(Lines 8 and 14) and at most* $\operatorname{rk}(\operatorname{Min}\mathcal{L}) + \left|\operatorname{Min}\mathcal{L}\right|_{\mathtt{st}}$ *updates to* T *(Line 10).*

Extending the WMSO+U Logic with Quantification over Tuples

Anita Badyl
Institute of Informatics, University of Warsaw, Poland

Paweł Parys
Institute of Informatics, University of Warsaw, Poland

Abstract

We study a new extension of the weak MSO logic, talking about boundedness. Instead of a previously considered quantifier U, expressing the fact that there exist arbitrarily large finite sets satisfying a given property, we consider a generalized quantifier U, expressing the fact that there exist tuples of arbitrarily large finite sets satisfying a given property. First, we prove that the new logic WMSO+$\mathsf{U}_{\mathsf{tup}}$ is strictly more expressive than WMSO+U. In particular, WMSO+$\mathsf{U}_{\mathsf{tup}}$ is able to express the so-called simultaneous unboundedness property, for which we prove that it is not expressible in WMSO+U. Second, we prove that it is decidable whether the tree generated by a given higher-order recursion scheme satisfies a given sentence of WMSO+$\mathsf{U}_{\mathsf{tup}}$.

2012 ACM Subject Classification Theory of computation → Logic and verification; Theory of computation → Verification by model checking; Theory of computation → Rewrite systems

Keywords and phrases Boundedness, logic, decidability, expressivity, recursion schemes

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.12

Related Version *Extended Version*: `http://arxiv.org/abs/2311.16607`

Funding Work supported by the National Science Centre, Poland (grant no. 2016/22/E/ST6/00041).

1 Introduction

In the field of logic in computer science, one of the goals is to find logics that, on the one hand, have decidable properties and, on the other hand, are as expressive as possible. An important example of such a logic is the monadic second-order logic, MSO, which defines exactly all regular properties of finite and infinite words [11, 18, 35] and trees [31], and is decidable over these structures.

A natural question that arises is whether MSO can be extended in a decidable way. Particular hopes were connected with expressing boundedness properties. Bojańczyk [3] introduced a logic called MSO+U, which extends MSO with a new quantifier U, with $\mathsf{U}X.\varphi$ saying that the subformula φ holds for arbitrarily large finite sets X. Originally, it was only shown that satisfiability over infinite trees is decidable for formulae where the U quantifier is only used once and not under the scope of set quantification. A significantly more powerful fragment of the logic, albeit for infinite words, was shown decidable by Bojańczyk and Colcombet [6] using automata with counters. These automata were further developed into the theory of cost functions initiated by Colcombet [15].

The difficulty of MSO+U comes from the interaction between the U quantifier and quantification over possibly infinite sets. This motivated the study of WMSO+U, which is a variant of MSO+U where set quantification is restricted to finite sets (the "W" in the name stands for *weak*). On infinite words, satisfiability of WMSO+U is decidable, and the logic has an automaton model [4]. Similar results hold for infinite trees [7], and have been used to decide properties of CTL* [12]. Currently, the strongest decidability result in this line is about WMSO+U over infinite trees extended with quantification over infinite paths [5]. The

latter result entails decidability of problems such as the realisability problem for prompt LTL [26], deciding the winner in cost parity games [20], or deciding certain properties of energy games [8].

The results mentioned so far concern mostly the satisfiability problem (is there a model in which a given formula is true?), but arguably the problem more relevant in practice is the model-checking problem: is a given formula satisfied in a given model? In a typical setting, the model represents (possible computations of) some computer system, and the formula expresses some desired property of the system, to be verified. The model is thus usually infinite, although described in a finite way. In this paper, as the class of considered models we choose trees generated by higher-order recursion schemes, which is a very natural and highly expressive choice.

Higher-order recursion schemes (recursion schemes in short) are used to faithfully represent the control flow of programs in languages with higher-order functions [17, 23, 27, 24]. This formalism is equivalent via direct translations to simply-typed λY-calculus [34]. Collapsible pushdown systems [22] and ordered tree-pushdown systems [13] are other equivalent formalisms. Recursion schemes easily cover finite and pushdown systems, but also some other models such as indexed grammars [1] and ordered multi-pushdown automata [9].

A classic result, with several proofs and extensions, says that model-checking trees generated by recursion schemes against MSO formulae is decidable: given a recursion scheme $\mathcal{G}$ and a formula $\varphi \in \mathrm{MSO}$, one can say whether φ holds in the tree generated by $\mathcal{G}$ [27, 22, 25, 32, 10, 33]. When it comes to boundedness properties, one has to first mention decidability of the simultaneous unboundedness property (a.k.a. diagonal property) [21, 14, 30]. In this problem one asks whether, in the tree generated by a given recursion scheme $\mathcal{G}$, there exist branches containing arbitrarily many occurrences of each of the labels $a_1, \ldots, a_k$ (i.e., whether for every $n \in \mathbb{N}$ there exists a branch on which every label from $\{a_1, \ldots, a_k\}$ occurs at least n times). This result turns out to be interesting, because it entails other decidability results for recursion schemes, concerning in particular computability of the downward closure of recognized languages [36], and the problem of separability by piecewise testable languages [16]. Then, we also have decidability for logics talking about boundedness. Namely, it was shown recently that model-checking for recursion schemes is decidable against formulae from WMSO+U [28] (and even from a mixture of MSO and WMSO+U, where quantification over infinite sets is allowed but cannot be arbitrarily nested with the U quantifier [29]). Another paper [2] shows decidability of model-checking for a subclass of recursion schemes against alternating B-automata and against weak cost monadic second-order logic (WCMSO); these are other formalisms allowing to describe boundedness properties, but in a different style than the U quantifier.

Interestingly, the decidability of model-checking for WMSO+U is obtained by a reduction to (a variant of) the simultaneous unboundedness problem. On the other hand, it seems that the simultaneous unboundedness property cannot be expressed in WMSO+U (except for the case of a single distinguished letter a_1), which is very intriguing.

Our contribution. As a first contribution, we prove the fact that was previously only a hypothesis: WMSO+U is indeed unable to express the simultaneous unboundedness property. Then, we define a new logic, WMSO+$\mathsf{U}_{\mathsf{tup}}$; it is an extension of WMSO+U, where the U quantifier can be used with a tuple of set variables, instead of just one variable. A construct with the extended quantifier, $\mathsf{U}(X_1, \ldots, X_k).\varphi$, says that the subformula φ holds for tuples of sets in which each of $X_1, \ldots, X_k$ is arbitrarily large. This logic is capable of easily expressing properties in which multiple quantities are simultaneously required to be unbounded. In particular, it can express the simultaneous unboundedness property, and thus it is strictly more expressive than the standard WMSO+U logic:

▶ **Theorem 1.1.** *The* WMSO+$\mathsf{U}_{\mathsf{tup}}$ *logic can express some properties of trees that are not expressible in* WMSO+U*; in particular, this is the case for the simultaneous unboundedness property.*

In fact, to separate the two logics it is enough to consider WMSO+$\mathsf{U}_{\mathsf{tup}}$ only with U quantifiers for pairs of variables (i.e., with $k = 2$). Actually, we are convinced that the proof of Theorem 1.1 contained in this paper can be modified for showing that, for every $k \geq 2$, WMSO+$\mathsf{U}_{\mathsf{tup}}$ without U quantifiers for tuples of length at least k is less expressive than WMSO+$\mathsf{U}_{\mathsf{tup}}$ with such quantifiers (cf.Remark 5.6).

Our main theorem says that the model-checking procedure for WMSO+U can be extended to the new logic:

▶ **Theorem 1.2.** *Given an* WMSO+$\mathsf{U}_{\mathsf{tup}}$ *sentence φ and a recursion scheme $\mathcal{G}$ one can decide whether φ is true in the tree generated by $\mathcal{G}$.*

2 Preliminaries

The powerset of a set X is denoted $\mathcal{P}(X)$. For $i, j \in \mathbb{N}$ we define $[i, j] = \{k \in \mathbb{N} \mid i \leq k \leq j\}$. The domain of a function f is denoted $\mathrm{dom}(f)$. When f is a function, by $f[x \mapsto y]$ we mean the function that maps x to y and every other $z \in \mathrm{dom}(f)$ to $f(z)$.

Trees. We consider rooted, potentially infinite trees, where children are ordered. For simplicity of the presentation, we consider only binary trees, where every node has at most two children. This is not really a restriction. Indeed, it is easy to believe that our proofs can be generalized to trees of arbitrary bounded finite arity without any problem (except for notational complications). Alternatively, a tree of arbitrary bounded finite arity can be converted into a binary tree using the first child / next sibling encoding, and a logical formula can be translated as well to a formula talking about the encoding; this means that the WMSO+$\mathsf{U}_{\mathsf{tup}}$ model-checking problem over trees of arbitrary bounded finite arity can be reduced to such a problem over binary trees.

Formally, a *tree domain* (a set of tree nodes) is a set $D \subseteq \{\mathsf{L}, \mathsf{R}\}^*$ that is closed under taking prefixes (i.e., if $uv \in D$ then also $u \in D$). A *tree* over an alphabet $\mathbb{A}$ is a function $T \colon D \to \mathbb{A}$, for some tree domain D. The set of trees over $\mathbb{A}$ is denoted $\mathcal{T}(\mathbb{A})$. The *subtree* of T starting in a node v is denoted $T{\restriction}v$ and is defined by $(T{\restriction}v)(u) = T(vu)$ (with domain $\{u \in \{\mathsf{L}, \mathsf{R}\}^* \mid vu \in \mathrm{dom}(T)\}$). For nodes we employ the usual notions of child, parent, ancestor, descendant, etc. (where we assume that a node is also an ancestor and a descendant of itself).

For trees T_1, T_2, and for $a \in \mathbb{A}$ we write $a\langle T_1, T_2\rangle$ for the tree T such that $T{\restriction}\mathsf{L} = T_1$, $T{\restriction}\mathsf{R} = T_2$, and $T(\varepsilon) = a$. We also write $\bot$ for the tree with empty domain.

Recursion schemes. Recursion schemes are grammars used to describe some infinite trees in a finitary way. We introduce recursion schemes only by giving an example, rather than by defining them formally. This is enough, because this paper does not work with recursion schemes directly; it only uses some facts concerning them.

A recursion scheme is given by a set of rules, like this:

$$\mathsf{S} \to \mathsf{F}\,\mathsf{G}\,, \qquad \mathsf{D}\,\mathsf{g}\,\mathsf{x} \to \mathsf{g}\,(\mathsf{g}\,\mathsf{x})\,,$$
$$\mathsf{F}\,\mathsf{g} \to \mathsf{a}\langle \mathsf{g}\,\bot, \mathsf{F}\,(\mathsf{D}\,\mathsf{g})\rangle\,, \qquad \mathsf{G}\,\mathsf{x} \to \mathsf{b}\langle \mathsf{x}, \bot\rangle\,.$$

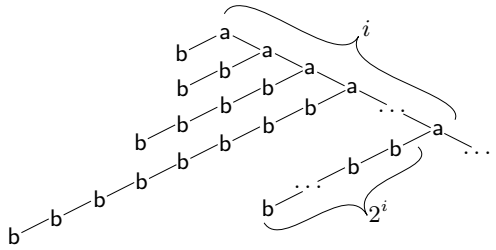


Figure 1 The tree generated by the example recursion scheme.

Here $\mathsf{S}, \mathsf{D}, \mathsf{F}, \mathsf{G}$ are nonterminals, with S being the starting nonterminal, x, g are variables, and a, b are letters from $\mathbb{A}$. To generate a tree, we start with S, which reduces to $\mathsf{F}\,\mathsf{G}$ using the first rule. We now use the rule for F, where the parameter g is instantiated to be G; we obtain $\mathsf{a}\langle \mathsf{G}\,\bot, \mathsf{F}\,(\mathsf{D}\,\mathsf{G})\rangle$. This already defines the root of the tree, which should be a-labeled; its two subtrees should be generated from $\mathsf{G}\,\bot$ and $\mathsf{F}\,(\mathsf{D}\,\mathsf{G})$, respectively. We see that $\mathsf{G}\,\bot$ reduces to $\mathsf{b}\langle\bot,\bot\rangle$, which is a tree with a single b-labeled node. On the other hand, $\mathsf{F}\,(\mathsf{D}\,\mathsf{G})$ reduces to $\mathsf{a}\langle \mathsf{D}\,\mathsf{G}\,\bot, \mathsf{F}\,(\mathsf{D}\,(\mathsf{D}\,\mathsf{G}))\rangle$, which means that the right child of the root is a-labeled, and its left subtree generated from $\mathsf{D}\,\mathsf{G}\,\bot$ (which reduces to $\mathsf{G}\,(\mathsf{G}\,\bot)$, then to $\mathsf{b}\langle \mathsf{G}\bot, \bot\rangle$, and then to $\mathsf{b}\langle \mathsf{b}\langle\bot,\bot\rangle,\bot\rangle$) is a path consisting of two b-labeled nodes. Continuing like this, when going right we always obtain a next a-labeled node (we thus have an infinite a-labeled branch), and to the left of the i-th such node we have a tree generated from $\underbrace{\mathsf{D}\,(\mathsf{D}\,(\ldots(\mathsf{D}}_{i-1}\,\mathsf{G})\ldots))\,\bot$, which is a finite branch consisting of 2^{i-1} b-labeled nodes (note that every D applies its argument twice, and hence doubles the number of produced b-labeled nodes). The resulting tree is depicted on Figure 1.

For a formal definition of recursion schemes consult prior work (e.g., [23, 24, 32, 28]). Some of these papers use a lambda-calculus notation, where our rule for D would be rather written as $\mathsf{D} \to \lambda\mathsf{g}.\lambda\mathsf{x}.\mathsf{g}\,(\mathsf{g}\,\mathsf{x})$. Sometimes it is also allowed to have λ inside a rule, like $\mathsf{S} \to \mathsf{F}\,(\lambda\mathsf{x}.\mathsf{b}\langle\mathsf{x},\bot\rangle)$; this does not make the definition more general, because subterms starting with λ can be always extracted to separate nonterminals.

3 The WMSO+$\mathsf{U}_{\mathsf{tup}}$ logic

In this section we introduce the logic under consideration: the WMSO+$\mathsf{U}_{\mathsf{tup}}$ logic.

Definition. For technical convenience, we use a syntax in which there are no first-order variables. It is easy to translate a formula from a more standard syntax to ours: first-order variables may be simulated by set variables for which we check that they contain exactly one node (i.e., that they are nonempty and that every subset thereof is either empty or equal to the whole set).

We assume an infinite set $\mathcal{V}$ of variables, which can be used to quantify over finite sets of tree nodes. In order to distinguish variables from sets to which these variables are valuated, we denote variables using Sans Serif font (e.g., $\mathsf{X}, \mathsf{Y}, \mathsf{Z}$). In the syntax of WMSO+$\mathsf{U}_{\mathsf{tup}}$ we have the following constructions:

$$\varphi ::= a(\mathsf{X}) \mid \mathsf{X} \not\downarrow_d \mathsf{Y} \mid \mathsf{X} \subseteq \mathsf{Y} \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi' \mid \exists_{\mathsf{fin}}\mathsf{X}.\varphi' \mid \mathsf{U}(\mathsf{X}_1,\ldots,\mathsf{X}_k).\varphi'\,,$$

where $a \in \mathbb{A}$, $d \in \{\mathsf{L},\mathsf{R}\}$, $k \in \mathbb{N}$, and $\mathsf{X},\mathsf{Y},\mathsf{X}_1,\ldots,\mathsf{X}_k \in \mathcal{V}$. Free variables of a formula are defined as usual; in particular $\mathsf{U}(\mathsf{X}_1,\ldots,\mathsf{X}_k)$ is a quantifier that bounds the variables $\mathsf{X}_1,\ldots,\mathsf{X}_k$.

We evaluate formulae of WMSO+$\mathsf{U}_{\mathsf{tup}}$ in $\mathbb{A}$-labeled trees. In order to evaluate a formula φ in a tree T, we also need a *valuation*, that is, a function ν from $\mathcal{V}$ to finite sets of nodes of T (its values are meaningful only for free variables of φ). The semantics of formulae is defined as follows:

- $a(\mathsf{X})$ holds when every node in $\nu(\mathsf{X})$ is labeled with a,
- $\mathsf{X} \swarrow_d \mathsf{Y}$ holds when both $\nu(\mathsf{X})$ and $\nu(\mathsf{Y})$ are singletons, and the unique node in $\nu(\mathsf{Y})$ is the left (if $d = \mathsf{L}$) / right (if $d = \mathsf{R}$) child of the unique node in $\nu(\mathsf{X})$,
- $\mathsf{X} \subseteq \mathsf{Y}$ holds when $\nu(\mathsf{X}) \subseteq \nu(\mathsf{Y})$,
- $\varphi_1 \wedge \varphi_2$ holds when both φ_1 and φ_2 hold,
- $\neg\varphi'$ holds when φ' does not hold,
- $\exists_{\mathsf{fin}}\mathsf{X}.\varphi'$ holds if there exists a finite set X of nodes of T for which φ' holds under the valuation $\nu[\mathsf{X} \mapsto X]$, and
- $\mathsf{U}(\mathsf{X}_1, \ldots, \mathsf{X}_k).\varphi'$ holds if for every $n \in \mathbb{N}$ there exist finite sets $X_1, \ldots, X_k$ of nodes of T, each of cardinality at least n, such that φ' holds under the valuation $\nu[\mathsf{X}_1 \mapsto X_1, \ldots, \mathsf{X}_k \mapsto X_k]$.

We write $T, \nu \models \varphi$ to denote that φ holds in T under the valuation ν.

Logical types. In proofs of both our results, Theorem 1.1 and Theorem 1.2, we use logical types, which we now define.

Let φ be a formula of WMSO+$\mathsf{U}_{\mathsf{tup}}$, let T be a tree, and let ν be a valuation. We define the *φ-type* of T under valuation ν, denoted $[\![T]\!]^\nu_\varphi$, by induction on the size of φ as follows:

- if φ is of the form $a(\mathsf{X})$ (for some letter $a \in \mathbb{A}$) or $\mathsf{X} \subseteq \mathsf{Y}$ then $[\![T]\!]^\nu_\varphi$ is the logical value of φ in T, ν, that is, tt if $T, \nu \models \varphi$ and ff otherwise,
- if φ is of the form $\mathsf{X} \swarrow_d \mathsf{Y}$, then $[\![T]\!]^\nu_\varphi$ equals:
  - tt if $T, \nu \models \varphi$,
  - empty if $\nu(\mathsf{X}) = \nu(\mathsf{Y}) = \emptyset$,
  - root if $\nu(\mathsf{X}) = \emptyset$ and $\nu(Y) = \{\varepsilon\}$, and
  - ff otherwise,
- if $\varphi = (\psi_1 \wedge \psi_2)$, then $[\![T]\!]^\nu_\varphi = ([\![T]\!]^\nu_{\psi_1}, [\![T]\!]^\nu_{\psi_2})$,
- if $\varphi = (\neg\psi)$, then $[\![T]\!]^\nu_\varphi = [\![T]\!]^\nu_\psi$,
- if $\varphi = \exists_{\mathsf{fin}}\mathsf{X}.\psi$, then

$$[\![T]\!]^\nu_\varphi = \{\sigma \mid \exists X.\, [\![T]\!]^{\nu[\mathsf{X}\mapsto X]}_\psi = \sigma\},$$

  where X ranges over finite sets of nodes of T, and
- if $\varphi = \mathsf{U}(\mathsf{X}_1, \ldots, \mathsf{X}_k).\psi$, then

$$[\![T]\!]^\nu_\varphi = \big(\{\sigma \mid \forall n \in \mathbb{N}.\, \exists X_1. \cdots . \exists X_k.\, [\![T]\!]^{\nu[\mathsf{X}_1\mapsto X_1,\ldots,\mathsf{X}_k\mapsto X_k]}_\psi = \sigma \wedge \forall i \in I.\, |X_i| \geq n\}\big)_{I\subseteq[1,k]},$$

  where $X_1, \ldots, X_k$ range over finite sets of nodes of T (the above φ-type is a tuple of 2^k sets, indexed by subsets I of $[1, k]$).

For each φ, let Typ_φ denote the set of all potential φ-types. Namely, $\mathit{Typ}_\varphi = \{\mathsf{tt}, \mathsf{ff}\}$ if $\varphi = a(\mathsf{X})$ or $\varphi = (\mathsf{X} \subseteq \mathsf{Y})$, $\mathit{Typ}_\varphi = \{\mathsf{tt}, \mathsf{empty}, \mathsf{root}, \mathsf{ff}\}$ if $\varphi = \mathsf{X} \swarrow_d \mathsf{Y}$, $\mathit{Typ}_\varphi = \mathit{Typ}_{\psi_1} \times \mathit{Typ}_{\psi_2}$ if $\varphi = (\psi_1 \wedge \psi_2)$, $\mathit{Typ}_\varphi = \mathit{Typ}_\psi$ if $\varphi = (\neg\psi)$; $\mathit{Typ}_\varphi = \mathcal{P}(\mathit{Typ}_\psi)$ if $\varphi = \exists_{\mathsf{fin}}\mathsf{X}.\psi$, and $\mathit{Typ}_\varphi = (\mathcal{P}(\mathit{Typ}_\psi))^{2^k}$ if $\varphi = \mathsf{U}(\mathsf{X}_1, \ldots, \mathsf{X}_k).\psi$.

The following two facts can be shown by a straightforward induction on the structure of a considered formula:

▶ **Fact 3.1.** *For every* WMSO+$\mathsf{U}_{\mathsf{tup}}$ *formula* φ *the set* Typ_φ *is finite.*

The second fact says that whether or not φ holds in T, ν is determined by $[\![T]\!]^\nu_\varphi$:

▶ **Fact 3.2.** *For every* WMSO+$\mathsf{U}_{\mathsf{tup}}$ *formula* φ *there is a computable function* $tv_\varphi \colon Typ_\varphi \to \{\mathsf{tt}, \mathsf{ff}\}$ *such that for every tree* $T \in \mathcal{T}(\mathbb{A})$ *and every valuation* ν *in* T*, it holds that* $tv_\varphi([\![T]\!]^\nu_\varphi) = \mathsf{tt}$ *if, and only if,* $T, \nu \models \varphi$*.*

Next, we observe that types behave in a compositional way, as formalized below. Here, for a node w we write $X{\restriction}w$ and $\nu{\restriction}w$ to denote the restriction of a set X and of a valuation ν to the subtree starting at w; formally, $X{\restriction}w = \{u \mid wu \in X\}$ and $\nu{\restriction}w$ maps every variable $\mathsf{X} \in \mathcal{V}$ to $\nu(\mathsf{X}){\restriction}w$.

▶ **Proposition 3.3.** *For every letter* $a \in \mathbb{A}$ *and every formula* φ*, one can compute a function* $Comp_{a,\varphi} \colon \mathcal{P}(\mathcal{V}) \times Typ_\varphi \times Typ_\varphi \to Typ_\varphi$ *such that for every tree* T *whose root has label* a *and for every valuation* ν*,*

$$[\![T]\!]^\nu_\varphi = Comp_{a,\varphi}(\{\mathsf{X} \mid \varepsilon \in \nu(\mathsf{X})\}, [\![T{\restriction}\mathsf{L}]\!]^{\nu{\restriction}\mathsf{L}}_\varphi, [\![T{\restriction}\mathsf{R}]\!]^{\nu{\restriction}\mathsf{R}}_\varphi)\,. \tag{1}$$

We remark that *a priori* the first argument of $Comp_{a,\varphi}$ is an arbitrary subset of $\mathcal{V}$, but in fact we only need to know which free variables of φ it contains; in consequence, $Comp_{a,\varphi}$ can be seen as a finite object.

Proof of Proposition 3.3. We proceed by induction on the size of φ.

When φ is of the form $b(\mathsf{X})$ or $\mathsf{X} \subseteq \mathsf{Y}$, then we see that φ holds in T, ν if, and only if, it holds in the subtrees $T{\restriction}\mathsf{L}, \nu{\restriction}\mathsf{L}$ and $T{\restriction}\mathsf{R}, \nu{\restriction}\mathsf{R}$, and in the root of T. Thus for $\varphi = b(\mathsf{X})$ as $Comp_{a,\varphi}(S, \tau_\mathsf{L}, \tau_\mathsf{R})$ we take tt when $\tau_\mathsf{L} = \tau_\mathsf{R} = \mathsf{tt}$ and either $a = b$ or $\mathsf{X} \notin S$. For $\varphi = (\mathsf{X} \subseteq \mathsf{Y})$ the last part of the condition is replaced by "if $\mathsf{X} \in S$ then $\mathsf{Y} \in S$".

Next, suppose that $\varphi = (\mathsf{X} \not\downarrow_d \mathsf{Y})$. Then as $Comp_{a,\varphi}(S, \tau_\mathsf{L}, \tau_\mathsf{R})$ we take

- tt if $\mathsf{X} \notin S$, $\mathsf{Y} \notin S$, and either $\tau_\mathsf{L} = \mathsf{tt}$ and $\tau_\mathsf{R} = \mathsf{empty}$ or $\tau_\mathsf{L} = \mathsf{empty}$ and $\tau_\mathsf{R} = \mathsf{tt}$,
- tt also if $\mathsf{X} \in S$, $\mathsf{Y} \notin S$, $\tau_d = \mathsf{root}$, and $\tau_i = \mathsf{empty}$ for the direction i other than d,
- empty if $\mathsf{X} \notin S$, $\mathsf{Y} \notin S$, and $\tau_L = \tau_S = \mathsf{empty}$,
- root if $\mathsf{X} \notin S$, $\mathsf{Y} \in S$, and $\tau_L = \tau_S = \mathsf{empty}$, and
- ff otherwise.

By comparing this definition with the definition of the type we immediately see that Equality (1) is satisfied.

When $\varphi = (\neg\psi)$, we simply take $Comp_{a,\varphi} = Comp_{a,\psi}$, and when $\varphi = (\psi_1 \wedge \psi_2)$, as $Comp_{a,\varphi}(S, (\tau^1_\mathsf{L}, \tau^2_\mathsf{L}), (\tau^1_\mathsf{R}, \tau^2_\mathsf{R}))$ we take the pair of $Comp_{a,\psi_i}(S, \tau^i_\mathsf{L}, \tau^i_\mathsf{R})$ for $i \in \{1, 2\}$.

Suppose now that $\varphi = \exists_{\mathsf{fin}}\mathsf{X}.\psi$. We define $Comp_{a,\varphi}(S, \tau_\mathsf{L}, \tau_\mathsf{R})$ to be

$$\{Comp_{a,\psi}(S', \sigma_\mathsf{L}, \sigma_\mathsf{R}) \mid S \setminus \{\mathsf{X}\} \subseteq S' \subseteq S \cup \{\mathsf{X}\}, \sigma_\mathsf{L} \in \tau_\mathsf{L}, \sigma_\mathsf{R} \in \tau_\mathsf{R}\}\,.$$

Let us check Equality (1) in details. Denote $S = \{\mathsf{Y} \mid \varepsilon \in \nu(\mathsf{Y})\}$. In order to show the left-to-right inclusion recall that, by definition, $[\![T]\!]^\nu_\varphi$ is a set of ψ-types, whose every element is of the form $[\![T]\!]^{\nu[\mathsf{X}\mapsto X]}_\psi$ for some finite set of nodes X. For every such X by the induction hypothesis we have $[\![T]\!]^{\nu[\mathsf{X}\mapsto X]}_\psi = Comp_{a,\psi}(S', [\![T{\restriction}\mathsf{L}]\!]^{\nu[\mathsf{X}\mapsto X]{\restriction}\mathsf{L}}_\varphi, [\![T{\restriction}\mathsf{R}]\!]^{\nu[\mathsf{X}\mapsto X]{\restriction}\mathsf{R}}_\varphi)$, where $S' = S \cup \{\mathsf{X}\}$ if $\varepsilon \in X$ and $S' = S \setminus \{\mathsf{X}\}$ if $\varepsilon \notin X$; moreover $[\![T{\restriction}\mathsf{L}]\!]^{\nu[\mathsf{X}\mapsto X]{\restriction}\mathsf{L}}_\psi \in [\![T{\restriction}L]\!]^{\nu{\restriction}\mathsf{L}}_\varphi$ and $[\![T{\restriction}\mathsf{R}]\!]^{\nu[\mathsf{X}\mapsto X]{\restriction}\mathsf{R}}_\psi \in [\![T{\restriction}L]\!]^{\nu{\restriction}\mathsf{R}}_\varphi$, which implies that $[\![T]\!]^{\nu[\mathsf{X}\mapsto X]}_\psi \in Comp_{a,\varphi}(S, [\![T{\restriction}\mathsf{L}]\!]^{\nu{\restriction}\mathsf{L}}_\varphi, [\![T{\restriction}\mathsf{R}]\!]^{\nu{\restriction}\mathsf{R}}_\varphi)$, as required. For the opposite inclusion take some $\sigma \in Comp_{a,\varphi}(S, [\![T{\restriction}\mathsf{L}]\!]^{\nu{\restriction}\mathsf{L}}_\varphi, [\![T{\restriction}\mathsf{R}]\!]^{\nu{\restriction}\mathsf{R}}_\varphi)$; it is of the form $Comp_{a,\psi}(S', \sigma_\mathsf{L}, \sigma_\mathsf{R})$ for some $\sigma_\mathsf{L} \in [\![T{\restriction}\mathsf{L}]\!]^{\nu{\restriction}\mathsf{L}}_\varphi$ and $\sigma_\mathsf{R} \in [\![T{\restriction}\mathsf{R}]\!]^{\nu{\restriction}\mathsf{R}}_\varphi$, where S' is either $S \cup \{\mathsf{X}\}$ or $S \setminus \{\mathsf{X}\}$. Then, by definition, σ_L and σ_R are of the form $[\![T{\restriction}\mathsf{L}]\!]^{(\nu{\restriction}\mathsf{L})[\mathsf{X}\mapsto X_\mathsf{L}]}_\psi$

and $[\![T{\restriction}\mathsf{R}]\!]_\psi^{(\nu{\restriction}\mathsf{R})[\mathsf{X}\mapsto X_\mathsf{R}]}$, respectively, for some finite sets of nodes X_L and X_R. We now take X such that $X{\restriction}\mathsf{L} = X_\mathsf{L}$ and $X{\restriction}\mathsf{R} = X_\mathsf{R}$, and $\varepsilon \in X$ if, and only if, $S' = S \cup \{\mathsf{X}\}$; we have $(\nu{\restriction}\mathsf{L})[\mathsf{X}\mapsto X_\mathsf{L}] = \nu[\mathsf{X}\mapsto X]{\restriction}\mathsf{L}$ and $(\nu{\restriction}\mathsf{R})[\mathsf{X}\mapsto X_\mathsf{R}] = \nu[\mathsf{X}\mapsto X]{\restriction}\mathsf{R}$. By the induction hypothesis we then have $\sigma = [\![T]\!]_\psi^{\nu[\mathsf{X}\mapsto X]}$, which by definition is an element of $[\![T]\!]_\varphi^\nu$, as required.

Finally, suppose that $\varphi = \mathsf{U}(\mathsf{X}_1,\dots,\mathsf{X}_k).\psi$. For $\tau_\mathsf{L} = (\rho_{\mathsf{L},I})_{I\subseteq[1,k]}$ and $\tau_\mathsf{R} = (\rho_{\mathsf{R},I})_{I\subseteq[1,k]}$ we define $\mathit{Comp}_{a,\varphi}(S,\tau_\mathsf{L},\tau_\mathsf{R})$ to be $(\rho_I)_{I\subseteq[1,k]}$, where

$$\rho_I = \{\mathit{Comp}_{a,\psi}(S',\sigma_\mathsf{L},\sigma_\mathsf{R}) \mid S\setminus\{\mathsf{X}_1,\dots,\mathsf{X}_k\} \subseteq S' \subseteq S\cup\{\mathsf{X}_1,\dots,\mathsf{X}_k\},\\ \sigma_\mathsf{L}\in\rho_{\mathsf{L},I_\mathsf{L}}, \sigma_\mathsf{R}\in\rho_{\mathsf{R},I_\mathsf{R}}, I_\mathsf{L}\cup I_\mathsf{R} = I\}\,.$$

In order to check Equality (1), denote $[\![T]\!]_\varphi^\nu = (\rho'_I)_{I\subseteq[1,k]}$, $[\![T{\restriction}\mathsf{L}]\!]_\varphi^{\nu{\restriction}\mathsf{L}} = (\rho_{\mathsf{L},I})_{I\subseteq[1,k]}$, $[\![T{\restriction}\mathsf{R}]\!]_\varphi^{\nu{\restriction}\mathsf{R}} = (\rho_{\mathsf{R},I})_{I\subseteq[1,k]}$, and $S = \{\mathsf{Y} \mid \varepsilon\in\nu(\mathsf{Y})\}$; we have to prove that $\rho'_I = \rho_I$ for all $I\subseteq[1,k]$ (where ρ_I is as defined above).

For the left-to-right inclusion, take some $\sigma\in\rho'_I$. By definition, it is a ψ-type such that for every $n\in\mathbb{N}$ there exist finite sets $X_{n,1},\dots,X_{n,k}$ for which $[\![T]\!]_\psi^{\nu[\mathsf{X}_1\mapsto X_{n,1},\dots,\mathsf{X}_k\mapsto X_{n,k}]} = \sigma$, where the cardinality of the sets $X_{n,i}$ with $i\in I$ is at least n. To every n let us assign the following information, called *characteristic*, and consisting of $2k$ bits and 2 ψ-types:

- for every $i\in[1,k]$, does the root ε belong to $X_{n,i}$?
- for every $i\in[1,k]$, is $X_{n,i}{\restriction}\mathsf{L}$ larger than $X_{n,i}{\restriction}\mathsf{R}$?
- the ψ-types $[\![T{\restriction}\mathsf{L}]\!]_\psi^{\nu[\mathsf{X}_1\mapsto X_{n,1},\dots,\mathsf{X}_k\mapsto X_{n,k}]{\restriction}\mathsf{L}}$ and $[\![T{\restriction}\mathsf{R}]\!]_\psi^{\nu[\mathsf{X}_1\mapsto X_{n,1},\dots,\mathsf{X}_k\mapsto X_{n,k}]{\restriction}\mathsf{R}}$.

Because there are only finitely many possible characteristics, by the pigeonhole principle we may find an infinite set $G\subseteq\mathbb{N}$ of indices n such that the same characteristic is assigned to every $n\in G$. We then take

$$S' = S\setminus\{\mathsf{X}_1,\dots,\mathsf{X}_k\}\cup\{\mathsf{X}_i \mid \varepsilon\in X_{n,i} \text{ for } n\in G\}\,,\\ I_\mathsf{L} = \{i\in I \mid |X_{n,i}{\restriction}\mathsf{L}| > |X_{n,i}{\restriction}\mathsf{R}| \text{ for } n\in G\}\,,\qquad I_\mathsf{R} = I\setminus I_\mathsf{L}\,,\\ \sigma_\mathsf{L} = [\![T{\restriction}\mathsf{L}]\!]_\psi^{\nu[\mathsf{X}_1\mapsto X_{n,1},\dots,\mathsf{X}_k\mapsto X_{n,k}]{\restriction}\mathsf{L}}\,,\qquad \sigma_\mathsf{R} = [\![T{\restriction}\mathsf{R}]\!]_\psi^{\nu[\mathsf{X}_1\mapsto X_{n,1},\dots,\mathsf{X}_k\mapsto X_{n,k}]{\restriction}\mathsf{R}} \qquad \text{for } n\in G.$$

The induction hypothesis (used with the valuation $\nu[\mathsf{X}_1\mapsto X_{n,1},\dots,\mathsf{X}_k\mapsto X_{n,k}]$ for any $n\in G$) gives us $\sigma = \mathit{Comp}_{a,\psi}(S',\sigma_\mathsf{L},\sigma_\mathsf{R})$. For every $m\in\mathbb{N}$ we can find $n\in G$ such that $n\geq 2m+1$; then $[\![T{\restriction}\mathsf{L}]\!]_\psi^{\nu[\mathsf{X}_1\mapsto X_{n,1},\dots,\mathsf{X}_k\mapsto X_{n,k}]{\restriction}\mathsf{L}} = \sigma_\mathsf{L}$ and $|X_{n,i}{\restriction}\mathsf{L}|\geq m$ for all $i\in I_\mathsf{L}$ ($X_{n,i}$ has at least $2m+1$ elements because $i\in I$, one of them may be the root, and at least half of the other elements is in the left subtree by definition of I_L). This implies that $\sigma_\mathsf{L}\in\rho_{\mathsf{L},I}$, by definition of the φ-type. Likewise $\sigma_\mathsf{R}\in\rho_{\mathsf{R},I}$. By definition of ρ_I this gives us $\sigma\in\rho_I$ as required.

The right-to-left inclusion is completely straightforward. Indeed, take some $\sigma\in\rho_I$. The definition of ρ_I gives us a set S' such that $S\setminus\{\mathsf{X}_1,\dots,\mathsf{X}_k\}\subseteq S'\subseteq S\cup\{\mathsf{X}_1,\dots,\mathsf{X}_k\}$, sets $I_\mathsf{L},I_\mathsf{R}\subseteq[1,k]$ such that $I_\mathsf{L}\cup I_\mathsf{R} = I$, and types $\sigma_\mathsf{L}\in\rho_{\mathsf{L},I_\mathsf{L}}$ and $\sigma_\mathsf{R}\in\rho_{\mathsf{R},I_\mathsf{R}}$ such that $\sigma = \mathit{Comp}_{a,\psi}(S',\sigma_\mathsf{L},\sigma_\mathsf{R})$. By definition of the two ψ-types, σ_L and σ_R, for every n there are sets $X_{\mathsf{L},1},\dots,X_{\mathsf{L},k}$ and $X_{\mathsf{R},1},\dots,X_{\mathsf{R},k}$ such that $[\![T{\restriction}\mathsf{L}]\!]_\psi^{(\nu{\restriction}\mathsf{L})[\mathsf{X}_1\mapsto X_{\mathsf{L},1},\dots,\mathsf{X}_k\mapsto X_{\mathsf{L},k}]} = \sigma_\mathsf{L}$, and $[\![T{\restriction}\mathsf{R}]\!]_\psi^{(\nu{\restriction}\mathsf{R})[\mathsf{X}_1\mapsto X_{\mathsf{R},1},\dots,\mathsf{X}_k\mapsto X_{\mathsf{R},k}]} = \sigma_\mathsf{R}$, and $|X_{\mathsf{L},i}|\geq n$ for all $i\in I_\mathsf{L}$, and $|X_{\mathsf{R},i}|\geq n$ for all $i\in I_\mathsf{R}$. We now take $X_1,\dots,X_k$ such that $X_i{\restriction}\mathsf{L} = X_{\mathsf{L},i}$, and $X_i{\restriction}\mathsf{R} = X_{\mathsf{R},i}$, and $\varepsilon\in X_i$ if, and only if, $\mathsf{X}_i\in S'$, for all $i\in[1,k]$. By the induction hypothesis we then have $[\![T]\!]_\psi^{\nu[\mathsf{X}_1\mapsto X_1,\dots,\mathsf{X}_k\mapsto X_k]} = \mathit{Comp}_{a,\psi}(S',\sigma_\mathsf{L},\sigma_\mathsf{R}) = \sigma$. Because additionally $|X_i|\geq n$ for all $i\in I = I_\mathsf{L}\cup I_\mathsf{R}$, we obtain that $\sigma\in\rho'_I$, as required. ◀

Figure 2 An example tree T (left), and the corresponding tree $\mathsf{refl}_\varphi(T)$ (right) obtained for an MSO sentence φ saying "the right child of the root has label a".

The next fact says that one can find a type of the empty tree. In the empty tree, a valuation has to map every variable to the empty set; we denote such a valuation by $\varnothing$. This fact is trivial: we simply follow the definition of $[\![\bot]\!]^{\varnothing}_{\varphi}$.

▶ **Fact 3.4.** *For ever formula φ, one can compute $[\![\bot]\!]^{\varnothing}_{\varphi}$.*

4 Decidability of model-checking

In this section we show how to evaluate WMSO+$\mathsf{U}_{\mathsf{tup}}$ formulae over trees generated by recursion schemes, that is, we prove Theorem 1.2. To this end, we first introduce three kinds of operations on recursion schemes, known to be computable. Then, we show how a sequence of these operations can be used to evaluate our formulae.

MSO reflection. The property of logical reflection for recursion schemes comes from Broadbent, Carayol, Ong, and Serre [10]. They state it for sentences of μ-calculus, but μ-calculus and MSO are equivalent over infinite trees [19].

Consider a tree T, and an MSO sentence φ (we skip a formal definition of MSO, assuming that it is standard). We define $\mathsf{refl}_\varphi(T)$ to be the tree having the same domain as T, and such that every node u thereof is labeled by the pair (a_u, b_u), where a_u is the label of u in T, and b_u is tt if φ is satisfied in $T{\restriction}u$ and ff otherwise. In other words, $\mathsf{refl}_\varphi(T)$ adds, in every node of T, a mark saying whether φ holds in the subtree starting in that node. Consult Figure 2 for an example.

▶ **Theorem 4.1** (MSO reflection [10, Theorem 7.3(2)])**.** *Given a recursion scheme $\mathcal{G}$ generating a tree T, and an* MSO *sentence φ, one can construct a recursion scheme $\mathcal{G}_\varphi$ generating the tree $\mathsf{refl}_\varphi(T)$.*

SUP reflection. The SUP reflection is the heart of our proof (where "SUP" stands for simultaneous unboundedness property). In order to talk about this property, we need a few more definitions. By $\#_a(V)$ we denote the number of a-labeled nodes in a (finite) tree V. For a set of (finite) trees $\mathcal{L}$ and a set of letters A, we define a predicate $SUP_A(\mathcal{L})$, which holds if for every $n \in \mathbb{N}$ there is some $V_n \in \mathcal{L}$ such that for all $a \in A$ it holds that $\#_a(V_n) \geq n$.

Originally, in the simultaneous unboundedness property we consider devices recognizing a set of finite trees, unlike recursion schemes, which generate a single infinite tree. We use here an equivalent formulation, in which the set of finite trees is encoded in a single infinite tree. To this end, we use two special letters: nd, denoting a nondeterministic choice (disjunction between two children), and $\mathsf{nd}_\bot$, denoting that there is no choice (empty disjunction). We write $T \to_{\mathsf{nd}} V$ if V is obtained from T by choosing some nd-labeled node u and some its child v, and attaching $T{\restriction}v$ in place of $T{\restriction}u$. In other words, $\to_{\mathsf{nd}}$ is the smallest relation such that $\mathsf{nd}\langle T_\mathsf{L}, T_\mathsf{R}\rangle \to_{\mathsf{nd}} T_d$ for $d \in \{\mathsf{L}, \mathsf{R}\}$, and $a\langle T_\mathsf{L}, T_\mathsf{R}\rangle \to_{\mathsf{nd}} a\langle T'_\mathsf{L}, T_\mathsf{R}\rangle$ if $T_\mathsf{L} \to_{\mathsf{nd}} T'_\mathsf{L}$, and $a\langle T_\mathsf{L}, T_\mathsf{R}\rangle \to_{\mathsf{nd}} a\langle T_\mathsf{L}, T'_\mathsf{R}\rangle$ if $T_\mathsf{R} \to_{\mathsf{nd}} T'_\mathsf{R}$. For a tree T, $\mathcal{L}(T)$ is the set of all finite trees V such

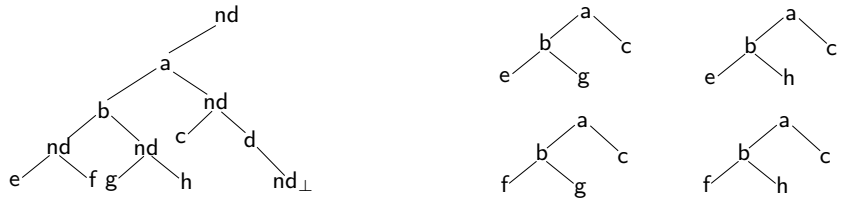


Figure 3 An example tree T, and four trees in $\mathcal{L}(T)$ (right). Additionally, $\mathcal{L}(T)$ contains $\bot$, the tree with empty domain, obtained by choosing the right child in the topmost nd-labeled node. Note that no tree in $\mathcal{L}(T)$ contains d, because d in T is followed by $\mathsf{nd}_\bot$, which is forbidden in trees in $\mathcal{L}(T)$.

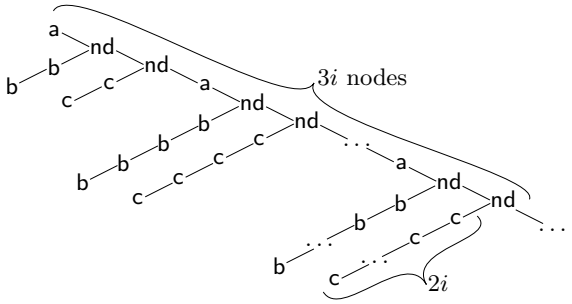


Figure 4 A tree T illustrating SUP reflection.

that $\#_{\mathsf{nd}}(V) = \#_{\mathsf{nd}_\bot}(V) = 0$ and $T \to^*_{\mathsf{nd}} V$. See Figure 3 for an example. We then say that T satisfies the *simultaneous unboundedness property with respect to* a set of letters A if $SUP_A(\mathcal{L}(T))$ holds, that is, if for every $n \in \mathbb{N}$ there are trees in $\mathcal{L}(T)$ having at least n occurrences of every letter from A.

Let T be a tree over an alphabet $\mathbb{A}$. We define $\mathsf{refl}_{SUP}(T)$ to be the tree having the same domain as T, and such that every node u thereof, having in T label a_u, is labeled by

- the pair $(a_u, \{A \subseteq \mathbb{A} \mid SUP_A(\mathcal{L}(T{\restriction}u))\})$, if $a_u \notin \{\mathsf{nd}, \mathsf{nd}_\bot\}$, and
- the original letter a_u, if $a_u \in \{\mathsf{nd}, \mathsf{nd}_\bot\}$.

In other words, $\mathsf{refl}_{SUP}(T)$ adds, in every node u of T (except for nd- and $\mathsf{nd}_\bot$-labeled nodes) and for every set A of letters, a mark saying whether $T{\restriction}u$ has the simultaneous unboundedness property with respect to A.

Consider, for example, the tree T from Figure 4. The tree $\mathsf{refl}_{SUP}(T)$ has the same shape as T. Every node u having label a in T gets label $(\mathsf{a}, \{\emptyset, \{\mathsf{a}\}, \{\mathsf{b}\}, \{\mathsf{c}\}, \{\mathsf{a}, \mathsf{b}\}, \{\mathsf{a}, \mathsf{c}\}\})$. Note that the set does not contain $\{\mathsf{b}, \mathsf{c}\}$ nor $\{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$: in $\mathcal{L}(T{\restriction}u)$ there are no trees having simultaneously many occurrences of b and many occurrences of c. Nodes u having in T label b or c are simply relabeled to $(\mathsf{b}, \{\emptyset\})$ or $(\mathsf{c}, \{\emptyset\})$, respectively, because $\mathcal{L}(T{\restriction}u)$ contains only a single tree, with a fixed number of nodes.

▶ **Theorem 4.2** (SUP reflection [30, Theorem 10.1]). *Given a recursion scheme $\mathcal{G}$ generating a tree T, one can construct a recursion scheme $\mathcal{G}_{SUP}$ generating the tree $\mathsf{refl}_{SUP}(T)$.*

▶ Remark 4.3. In the introduction we have described an easier variant of the simultaneous unboundedness property, called a *word variant*. In this variant, every node with label other than nd has at most one child; then choosing a tree in $\mathcal{L}(T)$ corresponds to choosing a branch of T (and trees in $\mathcal{L}(T)$ consist of single branches, hence they can be seen as words). Although the word variant of SUP is more commonly known than the *tree variant* described in this section, Theorem 4.2 holds also for the more general tree variant, as presented above.

Transducers. A *(deterministic, top-down) finite tree transducer* is a tuple $\mathcal{F} = (\mathbb{A}, \mathbb{B}, Q, q_0, \delta)$, where $\mathbb{A}$ is a finite input alphabet, $\mathbb{B}$ is a finite output alphabet, Q is a finite set of states, $q_0 \in Q$ is an initial state, and δ is a transition function mapping $Q \times (\mathbb{A} \cup \{\bot\})$ to finite trees over the alphabet $\mathbb{B} \cup (Q \times \{\mathsf{L}, \mathsf{R}\})$. Letters from $Q \times \{\mathsf{L}, \mathsf{R}\}$ are allowed to occur only in leaves of trees $\delta(q, a)$ with $a \in \mathbb{A}$ (internal nodes of these trees, and all nodes of trees $\delta(q, \bot)$ are labeled by letters from $\mathbb{B}$). Moreover, it is assumed that that there is no sequence $(q_1, a_1, d_1), (q_2, a_2, d_2), \ldots, (q_n, a_n, d_n)$ such that $\delta(q_i, a_i) = (q_{(i \bmod n)+1}, d_i)\langle \bot, \bot \rangle$ for all $i \in [1, n]$.

For an input tree T over $\mathbb{A}$ and a state $q \in Q$, we define an output tree $\mathcal{F}_q(T)$ over $\mathbb{B}$. Namely $\mathcal{F}_q(a\langle T_{\mathsf{L}}, T_{\mathsf{R}} \rangle)$ is the tree obtained from $\delta(q, a)$ by substituting $\mathcal{F}_r(T_d)$ for every leaf labeled with $(r, d) \in Q \times \{\mathsf{L}, \mathsf{R}\}$; additionally, $\mathcal{F}_q(\bot)$ simply equals $\delta(q, \bot)$ (recall that this tree has no labels from $Q \times \{\mathsf{L}, \mathsf{R}\}$). In other words, while being in state q over an a-labeled node of the input tree, the transducer produces a tree prefix specified by $\delta(q, a)$, where instead of outputting an (r, L)-labeled (or (r, R)-labeled) leaf, it rather continues by going to the left (respectively, right) child in the input tree, in state r; when $\mathcal{F}$ leaves the domain of the input tree, it still has a chance to output something, namely $\delta(q, \bot)$, and then it stops. In the root we start from the initial state, that is, we define $\mathcal{F}(T) = \mathcal{F}_{q_0}(T)$. To make the above definition formal, we can define $\mathcal{F}_q(T)(v)$, the label of $\mathcal{F}_q(T)$ in a node $v \in \{\mathsf{L}, \mathsf{R}\}^k$, by induction on the depth k, simultaneously for all input trees T and states $q \in Q$. Transitions $\delta(q, a)$ with (r, d) immediately in the root are a bit problematic, because we go down along the input tree without producing anything in the output tree; we have assumed, however, that such transitions do not form a cycle, so after a few (at most $|Q|$) steps we necessarily advance in the output tree.

Note that transducers need not be linear. For example, we may have $\delta(q, a) = a\langle a\langle (q, \mathsf{L}), (q, \mathsf{L}) \rangle, a\langle (q, \mathsf{R}), (q, \mathsf{R}) \rangle \rangle$, which creates two copies of the tree produced out of the left subtree, and two copies of the tree produced out of the right subtree.

We have the following theorem:

▶ **Theorem 4.4.** *Given a finite tree transducer $\mathcal{F} = (\mathbb{A}, \mathbb{B}, Q, q_0, \delta)$ and a recursion scheme $\mathcal{G}$ generating a tree T over the alphabet $\mathbb{A}$, one can construct a recursion scheme $\mathcal{G}_{\mathcal{F}}$ generating the tree $\mathcal{F}(T)$.*

This theorem follows from the equivalence between recursion schemes and collapsible pushdown systems [22], as it is straightforward to compose a collapsible pushdown system with $\mathcal{F}$. A formal proof can be found for instance in Parys [30, Appendix A].

Sequences of operations. We consider sequences of operations of the form $O_1, O_2, \ldots, O_n$, where every O_i is either an MSO sentence φ, or the string "SUP", or a finite tree transducer $\mathcal{F}$. Having a tree T, we can apply such a sequence of operations to it. Namely, we take $T_0 = T$, and for every $i \in [1, n]$, as T_i we take

- $\mathsf{refl}_{\varphi}(T_{i-1})$ if $O_i = \varphi$ is an MSO sentence,
- $\mathsf{refl}_{SUP}(T_{i-1})$ if $O_i = SUP$, and
- $\mathcal{F}(T_{i-1})$ if $O_i = \mathcal{F}$ is a finite tree transducer.

As the result we take T_n. We implicitly assume that whenever we apply a finite tree transducer to some tree, then the tree is over the input alphabet of the transducer; likewise, we assume that while computing $\mathsf{refl}_{\varphi}(T_{i-1})$, the formula uses letters from the alphabet of T_{i-1}.

Using the aforementioned closure properties (Theorems 4.1, 4.2, and 4.4) we can apply the operations on the level of recursion schemes generating our tree:

▶ **Proposition 4.5.** *Given a recursion scheme $\mathcal{G}$ generating a tree T, and a sequence of operations $O_1, O_2, \ldots, O_n$ as above, one can construct a recursion scheme $\mathcal{G}'$ generating the result of applying $O_1, O_2, \ldots, O_n$ to T.*

Main theorem. Let $\mathbb{A}$ be the alphabet used by WMSO+$\mathsf{U}_{\mathsf{tup}}$ formulae under consideration. We prove the following theorem:

▶ **Theorem 4.6.** *Given a* WMSO+$\mathsf{U}_{\mathsf{tup}}$ *sentence φ, one can compute a sequence of operations $O_1, O_2, \ldots, O_n$, such that for every tree T over $\mathbb{A}$, by applying $O_1, O_2, \ldots, O_n$ to T we obtain $\mathsf{tt}\langle\bot,\bot\rangle$ if φ is true in T, and $\mathsf{ff}\langle\bot,\bot\rangle$ otherwise.*

Having a recursion scheme generating either $\mathsf{tt}\langle\bot,\bot\rangle$ or $\mathsf{ff}\langle\bot,\bot\rangle$, we can easily check what is generated: we just repeatedly apply rules of the recursion scheme. Thus Theorem 1.2 is an immediate consequence of Theorem 4.6 and Proposition 4.5.

▶ Remark 4.7. Note that in Theorem 4.6 we do not assume that T is generated by a recursion scheme; the theorem holds for any tree T. Thus our decidability result, Theorem 1.2, can be immediately generalized from the class of trees generated by recursion schemes to any class of trees that is effectively closed under the considered three types of operations (i.e., any class for which Theorems 4.1, 4.2, and 4.4 remain true).

We now formulate a variant of Theorem 4.6 suitable for induction. On the input side, we have to deal with formulae with free variables (subformulae of our original sentence). On the output side, it is not enough to produce the truth value; we rather need to produce trees decorated by logical types. While logical types in general depend on the valuation of free variables, we consider here only a very special valuation mapping all variables to the empty set; recall that we denote such a valuation by $\varnothing$. Additionally, in the input tree we have to allow presence of some additional labels (used to store types with respect to other subformulae): we suppose that we have a tree T over an alphabet $\mathbb{A} \times \mathbb{B}$, where $\mathbb{A}$ is our fixed alphabet used by WMSO+$\mathsf{U}_{\mathsf{tup}}$ formulae, and $\mathbb{B}$ is some other auxiliary alphabet. Then by $\pi_{\mathbb{A}}(T)$ we denote the tree over $\mathbb{A}$ having the same domain as T, with every node thereof relabeled from $(a,b) \in \mathbb{A} \times \mathbb{B}$ to a.

▶ **Lemma 4.8.** *Given a* WMSO+$\mathsf{U}_{\mathsf{tup}}$ *formula φ and an auxiliary alphabet $\mathbb{B}$, one can compute a sequence of operations $O_1, O_2, \ldots, O_n$, such that for every tree T over $\mathbb{A} \times \mathbb{B}$, by applying $O_1, O_2, \ldots, O_n$ to T we obtain a tree having the same domain as T, such that every node u thereof is labeled by the pair $(\ell_u, [\![\pi_{\mathbb{A}}(T)\restriction u]\!]^{\varnothing}_{\varphi})$, where $\ell_u \in \mathbb{A} \times \mathbb{B}$ is the label of u in T.*

Theorem 4.6 is an immediate consequence of Lemma 4.8. Indeed, let us use Lemma 4.8 with a singleton alphabet $\mathbb{B}$; for such an alphabet we identify $\mathbb{A}$ with $\mathbb{A} \times \mathbb{B}$. By applying operations $O_1, \ldots, O_n$ obtained from Lemma 4.8 we obtain a tree with the root labeled by (a, τ) for $\tau = [\![T]\!]^{\varnothing}_{\varphi}$. Recall that, by Fact 3.2, we have a function tv_{φ} such that $tv_{\varphi}([\![T]\!]^{\varnothing}_{\varphi}) = \mathsf{tt}$ if, and only if, $T, \varnothing \models \varphi$. Thus, after all the operations $O_1, \ldots, O_n$, we can simply apply a transducer $\mathcal{F}$ that reads the root's label (a, τ) and returns the tree $\mathsf{tt}\langle\bot,\bot\rangle$ if $tv_{\varphi}(\tau) = \mathsf{tt}$, and the tree $\mathsf{ff}\langle\bot,\bot\rangle$ otherwise. There is a small exception if the original tree T has empty domain: then there is no root at all, in particular no root from which we can read the φ-type τ. Thus, if the transducer $\mathcal{F}$ sees an empty tree, it should rather use $\tau = [\![\bot]\!]^{\varnothing}_{\varphi}$, which is known by Fact 3.4.

Proof of Lemma 4.8. The proof is by induction on the structure of φ. We have several cases depending on the shape of φ.

Recall that in this lemma we only consider the valuation $\varnothing$ mapping all variables to the empty set. Because of that, if φ is of the form $a(X)$ or $X \subseteq Y$, then the φ-type $[\![\pi_{\mathbb{A}}(T)\restriction u]\!]^{\varnothing}_{\varphi}$ is tt for every tree T and node u thereof. It is thus enough to return (as the only operation O_1) a transducer that appends tt to the label of every node of T. Similarly, if $\varphi = (X \not\perp_d Y)$, then the φ-type $[\![\pi_{\mathbb{A}}(T)\restriction u]\!]^{\varnothing}_{\varphi}$ is always empty. For $\varphi = (\neg\psi)$ the situation is also trivial: we can directly use the induction hypothesis since $[\![\pi_{\mathbb{A}}(T)\restriction u]\!]^{\varnothing}_{\varphi} = [\![\pi_{\mathbb{A}}(T)\restriction u]\!]^{\varnothing}_{\psi}$.

Suppose that $\varphi = (\psi_1 \wedge \psi_2)$. The induction hypothesis for ψ_1 gives us a sequence of operations $O_1, O_2, \ldots, O_n$ that appends $[\![\pi_{\mathbb{A}}(T)\restriction u]\!]^{\varnothing}_{\psi_1}$ to the label of every node u of T. The resulting tree T' is over the alphabet $\mathbb{A} \times \mathbb{B} \times \mathit{Typ}_{\psi_1}$, which can be seen as $\mathbb{A} \times \mathbb{B}'$ for $\mathbb{B}' = \mathbb{B} \times \mathit{Typ}_{\psi_1}$; we have $\pi_{\mathbb{A}}(T') = \pi_{\mathbb{A}}(T)$. We can thus apply the induction hypothesis for ψ_2 to the resulting tree T'; it gives us a sequence of operations $O_{n+1}, O_{n+2}, \ldots, O_{n+m}$ that appends $[\![\pi_{\mathbb{A}}(T)\restriction u]\!]^{\varnothing}_{\psi_2}$ to the label of every node u of T'. The tree obtained after applying all the $n+m$ operations is as needed: in every node thereof we have appended the pair containing the ψ_1-type and the ψ_2-type, and such a pair is precisely the φ-type.

The case of $\varphi = \exists_{\mathsf{fin}}\mathsf{X}.\psi$ is handled by a reduction to the case of $\varphi' = \mathsf{U}\mathsf{X}.\psi$. Indeed, recall that the type for $\mathsf{U}(\mathsf{X}_1, \ldots, \mathsf{X}_k)$ is a tuple of 2^k coordinates indexed by sets $I \subseteq [1,k]$; in the case of a single variable $\mathsf{X}_1 = \mathsf{X}$, there are only two coordinates, one for $I = \emptyset$, and the other for $I = \{1\}$. The coordinate for $I = \emptyset$ in $[\![T']\!]^{\varnothing}_{\mathsf{U}\mathsf{X}.\psi}$ is simply $\{\sigma \mid \exists X. [\![T']\!]^{\nu[\mathsf{X}\mapsto X]}_{\psi} = \sigma\}$, that is, the φ-type $[\![T']\!]^{\nu}_{\exists_{\mathsf{fin}}\mathsf{X}.\psi}$. Thus, we can take the sequence of operations $O_1, O_2, \ldots, O_n$ from the forthcoming case of $\varphi' = \mathsf{U}\mathsf{X}.\psi$, which appends the φ'-type, and then add a simple transducer that removes the second coordinate of this type.

Finally, suppose that $\varphi = \mathsf{U}(\mathsf{X}_1, \ldots, \mathsf{X}_k).\psi$. By the induction hypothesis we have a sequence of operations $O_1, O_2, \ldots, O_n$ that appends the ψ-type $[\![\pi_{\mathbb{A}}(T)\restriction u]\!]^{\varnothing}_{\psi}$ to the label of every node u of T. Let T^1 be the tree obtained from T by applying these operations.

As a first step, to T^1 we apply a transducer $\mathcal{F}$ defined as follows. Its input alphabet is $\mathbb{A}' = \mathbb{A} \times \mathbb{B} \times \mathit{Typ}_{\psi}$, the alphabet of T^1, its output alphabet is $\mathbb{A}' \cup \{?, \#, \mathsf{nd}, \mathsf{nd}_{\perp}, \mathsf{X}_1, \ldots, \mathsf{X}_k\}$, and its set of states is $\{q_0\} \cup \mathit{Typ}_{\psi}$. Having a letter $\ell = (a, b, \tau) \in \mathbb{A}'$, let $\pi_{\mathbb{A}}(\ell) = a$ and $\pi_{\mathit{Typ}_{\psi}}(\ell) = \tau$. Coming to transitions, first for every triple $(S, \tau_{\mathsf{L}}, \tau_{\mathsf{R}})$, where $S = \{\mathsf{X}_{i_1}, \ldots, \mathsf{X}_{i_m}\} \subseteq \{\mathsf{X}_1, \ldots, \mathsf{X}_k\}$ and $\tau_{\mathsf{L}}, \tau_{\mathsf{R}} \in \mathit{Typ}_{\psi}$ we define

$$\mathit{sub}(S, \tau_{\mathsf{L}}, \tau_{\mathsf{R}}) = \mathsf{X}_{i_1}\langle \perp, \mathsf{X}_{i_2}\langle \perp, \ldots \mathsf{X}_{i_m}\langle \perp, \#\langle(\tau_{\mathsf{L}}, \mathsf{L}), (\tau_{\mathsf{R}}, \mathsf{R})\rangle\rangle \ldots \rangle\rangle\,.$$

Moreover, for every $\ell \in \mathbb{A}'$ and $\tau \in \mathit{Typ}_{\psi}$, let $\mathit{here}(\ell, \tau) = \perp$ if $\tau = \pi_{\mathit{Typ}_{\psi}}(\ell)$ and $\mathit{here}(\ell, \tau) = \mathsf{nd}_{\perp}\langle \perp, \perp\rangle$ otherwise. In order to define $\delta(\tau, \ell)$, we consider all triples $(S_1, \tau_{\mathsf{L},1}, \tau_{\mathsf{R},1}), \ldots, (S_s, \tau_{\mathsf{L},s}, \tau_{\mathsf{R},s})$ for which $\mathit{Comp}_{\pi_{\mathbb{A}}(\ell),\psi}(S_i, \tau_{\mathsf{L},i}, \tau_{\mathsf{R},i}) = \tau$ (assuming some fixed order in which these triples are listed). Then, we take

$$\delta(\tau, \ell) = ?\langle \perp, \mathsf{nd}\langle \mathit{sub}(S_1, \tau_{\mathsf{L},1}, \tau_{\mathsf{R},1}), \mathsf{nd}\langle \mathit{sub}(S_2, \tau_{\mathsf{L},2}, \tau_{\mathsf{R},2}), \ldots \\ \mathsf{nd}\langle \mathit{sub}(S_s, \tau_{\mathsf{L},s}, \tau_{\mathsf{R},s}), \mathit{here}(\ell, \tau)\rangle \ldots \rangle\rangle\,.$$

Additionally, we consider the list $\tau_1, \ldots, \tau_r$ of all ψ-types from Typ_{ψ} (listed in some fixed order), and we define

$$\delta(q_0, \ell) = \ell\langle(q_0, \mathsf{L}), \#\langle(q_0, \mathsf{R}), \#\langle\delta(\tau_1, \ell), \#\langle\delta(\tau_2, \ell), \ldots \#\langle\delta(\tau_r, \ell), \perp\rangle \ldots\rangle\rangle\rangle\,.$$

For the empty tree we define

$$\delta(q_0, \perp) = \perp, \qquad \delta([\![\perp]\!]^{\varnothing}_{\psi}, \perp) = \perp, \qquad \text{and} \qquad \delta(\tau, \perp) = \mathsf{nd}_{\perp} \qquad \text{for } \tau \neq [\![\perp]\!]^{\varnothing}_{\psi}\,.$$

The "main part" of the result $\mathcal{F}(T^1)$, produced using the state q_0 is an almost unchanged copy of T^1; there is only a technical change, that a new #-labeled node is inserted between every node and its right child, so that the right child is moved to the left child of this new

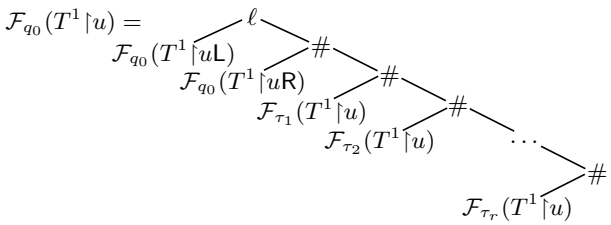


Figure 5 An illustration of $\mathcal{F}_{q_0}(T^1\restriction u)$. Here, ℓ is the label of u in T^1, and $\tau_1, \ldots, \tau_r$ are all possible ψ-types.

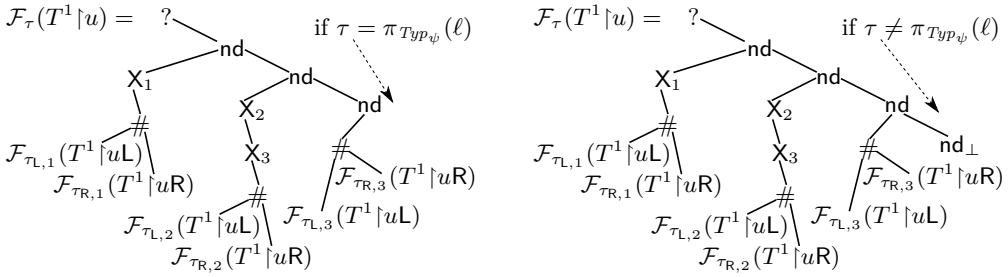


Figure 6 An illustration of $\mathcal{F}_\tau(T^1\restriction u)$. We assume that there are exactly three triples $(S, \tau_\mathsf{L}, \tau_\mathsf{R})$ such that $Comp_{\pi_\mathbb{A}(\ell),\psi}(S, \tau_\mathsf{L}, \tau_\mathsf{R}) = \tau$, namely $(\{\mathsf{X}_1\}, \tau_{\mathsf{L},1}, \tau_{\mathsf{R},1})$, $(\{\mathsf{X}_2, \mathsf{X}_3\}, \tau_{\mathsf{L},2}, \tau_{\mathsf{R},2})$, and $(\emptyset, \tau_{\mathsf{L},3}, \tau_{\mathsf{R},3})$, for ℓ being the label of u in T^1. We have two cases depending on whether the ψ-type written in ℓ is τ or not.

right child. But additionally, below the new #-labeled right child of every node u of T^1, there are $|Typ_\psi|$ modified copies of $T^1\restriction u$, attached below a branch of #-labeled nodes (cf. Figure 5). For each ψ-type τ we have such a copy, namely $\mathcal{F}_\tau(T^1\restriction u)$, responsible for checking whether the type of $\pi_\mathbb{A}(T)\restriction u$ can be τ. The tree $\mathcal{F}_\tau(T^1\restriction u)$ is a disjunction (formed by nd-labeled nodes) of all possible triples $(S, \tau_\mathsf{L}, \tau_\mathsf{R})$ such that types τ_L and τ_R in children of u, together with S being the set of those variables among $\mathsf{X}_1, \ldots, \mathsf{X}_k$ that contain u, result in type τ in u (cf. Figure 6). We output the variables from S in the resulting tree, so that they can be counted, and then we have subtrees $\mathcal{F}_{\tau_\mathsf{L}}(T^1\restriction u\mathsf{L})$ and $\mathcal{F}_{\tau_\mathsf{R}}(T^1\restriction u\mathsf{R})$, responsible for checking whether the type in the children of u can be τ_L and τ_R. Additionally, the *here* subtree allows to finish immediately if τ is the ψ-type of $T^1\restriction u$ under the empty valuation. Formally, we have the following claim:

▷ Claim 4.9. For every ψ-type τ, numbers $n_1, \ldots, n_k \in \mathbb{N}$, and node u, the following two statements are equivalent:

- there exist sets $X_1, \ldots, X_k$ of nodes of $T\restriction u$ such that $[\![\pi_\mathbb{A}(T)\restriction u]\!]_\psi^{\varnothing[\mathsf{X}_1\mapsto X_1,\ldots,\mathsf{X}_k\mapsto X_k]} = \tau$ and $|X_i| = n_i$ for $i \in [1,k]$, and
- there exists a tree $V \in \mathcal{L}(\mathcal{F}_\tau(T^1\restriction u))$ such that $\#_{\mathsf{X}_i}(V) = n_i$ for $i \in [1,k]$.

Proof. Let us concentrate on the left-to-right implication. The proof is by induction on the maximal depth of nodes in the X_i sets. We have three cases. First, it is possible that u is not a node of T. Then, all the sets X_i have to be empty, so we have $\tau = [\![\bot]\!]_\psi^\varnothing$, and hence $\mathcal{F}_\tau(T^1\restriction u) = \delta(\tau, \bot) = \bot$ (recall that T and T^1 have the same domain). The set $\mathcal{L}(\bot)$ contains the tree $\bot$ which indeed has no X_i labeled nodes, as needed.

Second, it is possible that u is a node of T, but all the sets X_i are empty. Let ℓ be the label of u in T^1. By construction of T^1, we have $\pi_{Typ_\psi}(\ell) = [\![\pi_\mathbb{A}(T)\restriction u]\!]_\psi^\varnothing = \tau$. On the rightmost branch of $\mathcal{F}_\tau(T^1\restriction u)$, after a ?-labeled node and a few nd-labeled nodes, we have the subtree $here(\ell, \tau)$, which is $\bot$ by the above equality. We can return the tree $?\langle\bot, \bot\rangle$, which belongs to $\mathcal{L}(\mathcal{F}_\tau(T^1\restriction u))$.

Finally, suppose that our sets are not all empty. Then necessarily u is inside T (and T^1); let ℓ be the label of u in T^1 (by construction of T^1, the label of u in T consists of the first two coordinates of ℓ). Consider $S = \{\mathsf{X}_i \mid \varepsilon \in X_i\}$ and $\tau_d = [\![\pi_{\mathbb{A}}(T)\restriction ud]\!]_\psi^{\varnothing[\mathsf{X}_1\mapsto X_1,\dots,\mathsf{X}_k\mapsto X_k]\restriction d}$ for $d \in \{\mathsf{L},\mathsf{R}\}$. By the induction hypothesis, there are trees $V_d \in \mathcal{L}(\mathcal{F}_{\tau_d}(T^1\restriction ud))$ such that $\#_{\mathsf{X}_i}(V_d) = |X_i\restriction d|$ for $i \in [1,k]$. Due to Equality (1) we have $\tau = Comp_{\pi_{\mathbb{A}}(\ell),\psi}(S,\tau_{\mathsf{L}},\tau_{\mathsf{R}})$. This means that $\delta(\tau,\ell)$, below a ?-labeled node and a few nd-labeled, produces a subtree using $sub(S,\tau_{\mathsf{L}},\tau_{\mathsf{R}})$. We define V by choosing this subtree. Then, there are some X_i-labeled nodes, for all $\mathsf{X}_i \in S$ (that is, for those sets X_i that contain the root of $T\restriction u$). Below them, we have the tree $\#\langle \mathcal{F}_{\tau_{\mathsf{L}}}(T^1\restriction u\mathsf{L}), \mathcal{F}_{\tau_{\mathsf{R}}}(T^1\restriction u\mathsf{R})\rangle$; in its left subtree we choose V_{L}, and in its right subtree we choose V_{R}. This way, we obtain a tree $V \in \mathcal{L}(\mathcal{F}_\tau(T^1\restriction u))$, where the number of X_i-labeled nodes is indeed $|X_i| = n_i$, for all $i \in [1,k]$.

We skip the proof of the right-to-left implication, as it is analogous (this time, the induction is on the height of the tree V). ◁

Let $T^2 = \mathcal{F}(T^1)$. As the next operation after $\mathcal{F}$, we use SUP. Let $T^3 = \mathsf{refl}_{SUP}(T^2)$. The SUP operation attaches a label to every node of T^3 (except for nd-labeled nodes), but we are interested in these labels only in nodes originally (i.e., in T_2) labeled by "?". Every such node is the root of a subtree $\mathsf{refl}_{SUP}(\mathcal{F}_\tau(T^1\restriction u))$ for some node u of T^1; it becomes labeled by $(?,\mathcal{U})$, where $\mathcal{U} = \{A \subseteq \mathbb{A}' \mid SUP_A(\mathcal{L}(\mathcal{F}_\tau(T^1\restriction u)))\}$. Recall that $\varphi = \mathsf{U}(\mathsf{X}_1,\dots,\mathsf{X}_k).\psi$ and that the φ-type is a tuple of 2^k coordinates, indexed by sets $I \subseteq [1,k]$. Consider such a set I, and take $A_I = \{\mathsf{X}_i \mid i \in I\}$. By definition of SUP_{A_I}, the label $\mathcal{U}$ contains A_I if, and only if, for every $n \in \mathbb{N}$ the language $\mathcal{L}(\mathcal{F}_\tau(T^1\restriction u))$ contains trees with at least n occurrences of every element of A_I. By Claim 4.9 this is the case if, and only if, for every $n \in \mathbb{N}$ there exist sets $X_1,\dots,X_k$ of nodes of $T\restriction u$ such that $[\![\pi_{\mathbb{A}}(T)\restriction u]\!]_\psi^{\varnothing[\mathsf{X}_1\mapsto X_1,\dots,\mathsf{X}_k\mapsto X_k]} = \tau$ and $|X_i| \geq n$ for all $i \in I$. This, in turn, holds if, and only if, the I-coordinate of the φ-type $[\![\pi_{\mathbb{A}}(T)\restriction u]\!]_\varphi^\varnothing$ contains τ. (The case of $I = \emptyset$ is a bit delicate, but one can see that the proof works without any change also in this case.)

The above means that all the φ-types we wished to compute are already present in T^3, we only have to move them to correct places. To this end, for every ψ-type τ_i, and for every set $I \subseteq [1,k]$ we append to our sequence of operations a formula $\theta_{i,I}$ saying that the node $\mathsf{R}^{i+1}\mathsf{L}$ has label of the form $(?,\mathcal{U},\dots)$ with $A_I \in \mathcal{U}$ (note that this node in $\mathcal{F}_{q_0}(T^1\restriction u)$ is the ?-labeled root of $\mathcal{F}_{\tau_i}(T^1\restriction u)$; the operation SUP appends a set $\mathcal{U}$ to this label, and operations $\theta_{i',I'}$ applied so far append some additional coordinates that we ignore).

After that, we already have all φ-types in correct nodes, but in a wrong format; we also have additional nodes not present in the original tree T. To deal with this, at the end we apply a transduction $\mathcal{F}'$, which

- removes all nodes labeled by $(\#,\dots)$ and their right subtrees, hence leaving only nodes present in the original tree T;
- the remaining nodes have labels of the form $(a,b,\tau,\mathcal{U},v_{i_1,I_1},\dots,v_{i_s,I_s})$; we relabel them to $(a,b,(\{\tau_i \mid v_{i,I} = \mathsf{tt}\})_{I\subseteq[1,k]})$.

This last transduction produces a tree exactly as needed. ◀

5 Expressivity

In this section we prove our second main result, Theorem 1.1, saying that the simultaneous unboundedness property can be expressed in WMSO+$\mathsf{U}_{\mathsf{tup}}$, but not in WMSO+$\mathsf{U}$. The positive part of this statement is easy:

▶ **Proposition 5.1.** *For every set of letters A there exists a* WMSO+$\mathsf{U}_{\mathsf{tup}}$ *sentence φ which holds in a tree T if, and only if, $SUP_A(\mathcal{L}(T))$ holds.*

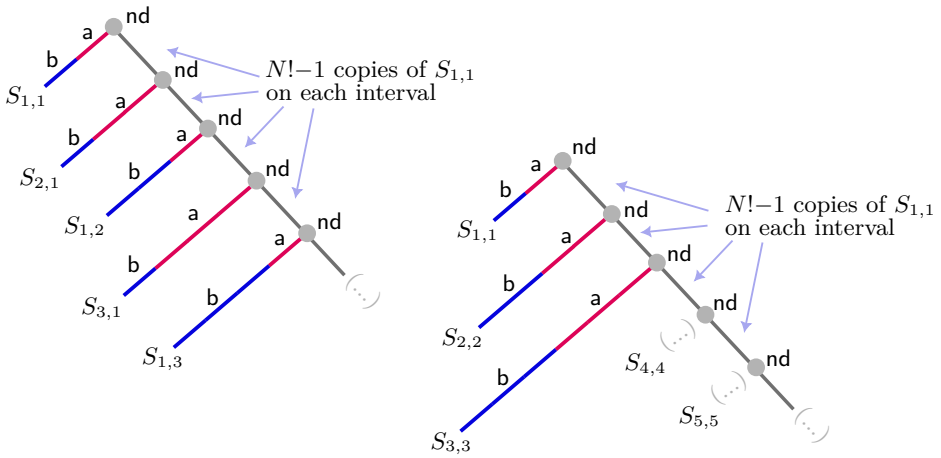


Figure 7 T_1 (left) and T_2 (right).

Proof. Let $A = \{a_1, \dots, a_k\}$. We take

$$\varphi = \mathsf{U}(\mathsf{X}_1, \dots, \mathsf{X}_k).\exists_{\mathsf{fin}}\mathsf{Y}.(a_1(\mathsf{X}_1) \wedge \dots \wedge a_k(\mathsf{X}_k) \wedge \mathsf{X}_1 \subseteq \mathsf{Y} \wedge \dots \wedge \mathsf{X}_k \subseteq \mathsf{Y} \wedge \psi(\mathsf{Y})),$$

where $\psi(\mathsf{Y})$ expresses the fact that Y contains nodes of a single tree from $\mathcal{L}(T)$, together with their nd-labeled ancestors, that is, that for every node v of Y,

- the parent of v, if exists, belongs to Y;
- if v has label nd, then exactly one child of v belongs to Y (strictly speaking: there is a direction $d \in \{\mathsf{L}, \mathsf{R}\}$ such that a child in this direction, if exists, belongs to Y, and the child in the opposite direction, if exists, does not belong to Y);
- v does not have label $\mathsf{nd}_\perp$; and
- if v has label other than nd, then all children of v belong to Y.

It is easy to write the above properties in WMSO+$\mathsf{U}_{\mathsf{tup}}$. Then φ expresses that for every $n \in \mathbb{N}$ there exist sets $X_1, \dots, X_k$ of nodes of some $V \in \mathcal{L}(T)$ such that $|X_i| \geq n$ and nodes of X_i have label a_i, for all $i \in [1, k]$; this is precisely the simultaneous unboundedness property with respect to the set $A = \{a_1, \dots, a_k\}$. ◀

In the remaining part of this section we prove that SUP with respect to $\{\mathsf{a}, \mathsf{b}\}$ cannot be expressed in WMSO+U (i.e., without using the U quantifier for tuples of variables). We prove this already for the word variant of SUP (cf. Remark 4.3), which is potentially easier to be expressed than SUP in its full generality.

Our proof is by contradiction. Assume thus that there is a sentence φ_{SUP} of WMSO+U that holds exactly in those trees T for which $SUP_{\{\mathsf{a},\mathsf{b}\}}(T)$ is true. Having φ_{SUP} fixed, we take a number N such that $|Typ_\varphi| \leq N$ and $|Typ_{\exists_{\mathsf{fin}}\mathsf{X}.\varphi}| \leq N$ for all subformulae φ of φ_{SUP} (recall that Typ_φ is a set containing all possible φ-types).

Based on N, we now define two trees, T_1 and T_2, such that $SUP_{\{\mathsf{a},\mathsf{b}\}}(T_2)$ but not $SUP_{\{\mathsf{a},\mathsf{b}\}}(T_1)$, and we show that they are indistinguishable by φ_{SUP}. We achieve that by demonstrating their type equality as stated in Lemma 5.5, which by Fact 3.2 gives their indistinguishability by the WMSO+U sentence φ_{SUP}.

▶ **Definition 5.2** (T_1 and T_2). *We define T_1 as a tree with an infinite rightmost path (that we call its* trunk*), containing* nd*-labeled nodes. For each integer $k \geq 0$, there is a leftward path called* vault *attached to the $(kN! + 1)$-th node of the trunk. If k is even, we denote the*

vault as $S_{1,\frac{k}{2}+1}$*, and otherwise as* $S_{\frac{k+1}{2}+1,1}$*. Each vault* $S_{m,n}$ *consists of two parts: the upper sub-path of length* $mN!$*, where every node has label* a*, and the lower sub-path of length* $nN!$*, where every node has label* b *(cf. Figure 7).*

To each node of the trunk that does not have a vault attached we attach a copy of $S_{1,1}$*. Note that we do not call these copies vaults; only the original* $S_{1,1}$ *starting at the root of* T_1 *is a vault.*

The definition of T_2 *is similar to that of* T_1*, except that this time the vault associated with each* k *is* $S_{k,k}$*, still starting at depth* $kN!+1$ *and having* $kN!$ *nodes with label* a *followed by* $kN!$ *nodes with label* b*.*

The technical core of our proof lies in the following two lemmata:

▶ **Lemma 5.3.** *Let* ψ *be such that* $|\mathit{Typ}_{\exists_{\mathsf{fin}}\mathsf{X}.\psi}| \leq N$*. If for all* $k', \ell' \in \mathbb{N}$ *we have* $[\![T_1 \restriction \mathsf{R}^{k'N!}]\!]^{\varnothing}_{\psi} = [\![T_2 \restriction \mathsf{R}^{\ell'N!}]\!]^{\varnothing}_{\psi}$*, then for all* $k, \ell \in \mathbb{N}$ *and* $\tau \in \mathit{Typ}_{\psi}$ *there exists a function* $f\colon \mathbb{N} \to \mathbb{N}$ *such that* $\lim_{n\to\infty} f(n) = \infty$ *and for all* $n \in \mathbb{N}$*,*

$$\exists X_1 \subseteq \mathrm{dom}(T_1 \restriction \mathsf{R}^{kN!}).\, |X_1| = n \wedge [\![T_1 \restriction \mathsf{R}^{kN!}]\!]^{\varnothing[\mathsf{X}\mapsto X_1]}_{\psi} = \tau$$
$$\implies \quad \exists X_2 \subseteq \mathrm{dom}(T_2 \restriction \mathsf{R}^{\ell N!}).\, f(n) \leq |X_2| < \infty \wedge [\![T_2 \restriction \mathsf{R}^{\ell N!}]\!]^{\varnothing[\mathsf{X}\mapsto X_2]}_{\psi} = \tau.$$

▶ **Lemma 5.4.** *Let* ψ *be such that* $|\mathit{Typ}_{\exists_{\mathsf{fin}}\mathsf{X}.\psi}| \leq N$*. If for all* $k', \ell' \in \mathbb{N}$ *we have* $[\![T_1 \restriction \mathsf{R}^{k'N!}]\!]^{\varnothing}_{\psi} = [\![T_2 \restriction \mathsf{R}^{\ell'N!}]\!]^{\varnothing}_{\psi}$*, then for all* $k, \ell \in \mathbb{N}$ *and* $\tau \in \mathit{Typ}_{\psi}$ *there exists a function* $f\colon \mathbb{N} \to \mathbb{N}$ *such that* $\lim_{n\to\infty} f(n) = \infty$ *and for all* $n \in \mathbb{N}$*,*

$$\exists X_1 \subseteq \mathrm{dom}(T_1 \restriction \mathsf{R}^{kN!}).\, f(n) \leq |X_1| < \infty \wedge [\![T_1 \restriction \mathsf{R}^{kN!}]\!]^{\varnothing[\mathsf{X}\mapsto X_1]}_{\psi} = \tau$$
$$\impliedby \quad \exists X_2 \subseteq \mathrm{dom}(T_2 \restriction \mathsf{R}^{\ell N!}).\, |X_2| = n \wedge [\![T_2 \restriction \mathsf{R}^{\ell N!}]\!]^{\varnothing[\mathsf{X}\mapsto X_2]}_{\psi} = \tau.$$

Note that the function f in Lemmata 5.3 and 5.4 may depend on k and ℓ. We only sketch here the proof of the above lemmata; a full proof can be found in Appendix A of the extended version.

Lemma 5.3 is slightly easier. Indeed, suppose first that $k = \ell = 0$. By assumption, in T_1 we have a finite set of nodes X_1 resulting in a ψ-type τ; based on X_1, we have to produce a finite set of nodes X_2 in T_2, producing the same ψ-type τ. The non-vault nodes of X_1 are transferred to X_2 without any change; note that the trees T_1, T_2 are identical outside of vaults. When in T_1 we have some vault $S_{1,i}$ (or $S_{i,1}$, handled in the same way), then in the analogous place of T_2 we have a vault $S_{j,j}$ with $j \geq i$. We use a form of pumping to convert $S_{1,i}$ with some nodes marked as elements of X_1 into $S_{j,j}$ with marked nodes, which we take to X_2; this is done so that the ψ-type does not change. Namely, we concentrate on ψ-types of subtrees of $S_{1,i}$ starting on different levels. Already in the bottom, b-labeled part of $S_{1,i}$ we can find two levels in distance at most N, where the ψ-type repeats (by the pigeonhole principle; recall that the number of possible ψ-types is at most N). We then repeat the fragment of $S_{1,i}$ between these two places (together with the set elements marked in it), so that $(j-i)N!$ new nodes are created, and we obtain $S_{1,j}$. Note that the repeated length, being at most N, necessarily divides $N!$. Because of Proposition 3.3, such a pumping does not change the ψ-type. In a similar way, we can pump the upper, a-labeled part of $S_{1,j}$, and obtain $S_{j,j}$. In this way, we convert a finite top part of T_1 (with a set X_1) into T_2 (with a set X_2) without changing the ψ-type; the infinite parts located below (where the sets X_1, X_2 do not contain any elements) have the same ψ-type by the assumption $[\![T_1 \restriction \mathsf{R}^{k'N!}]\!]^{\varnothing}_{\psi} = [\![T_2 \restriction \mathsf{R}^{\ell'N!}]\!]^{\varnothing}_{\psi}$. All nodes originally in X_1 remained in X_2 (possibly shifted), so we have $|X_2| \geq |X_1|$; the lemma holds with $f(n) = n$ in this case.

When k, ℓ are arbitrary (and we want to change $T_1{\restriction}\mathsf{R}^{kN!}$ into $T_2{\restriction}\mathsf{R}^{\ell N!}$), we proceed in a similar way, but there is a potential problem that a vault $S_{1,i}$ (or $S_{i,1}$) should be mapped to $S_{j,j}$ with $j < i$; then we should not stretch the vault, but rather contract it. But contracting is also possible: this time we look on $\exists_{\mathsf{fin}}\mathsf{X}.\psi$-types (instead of ψ-types) on the shorter target vault $S_{1,j}$; we can pump the vault as previously, so $S_{1,i}$ and $S_{1,j}$ have the same $\exists_{\mathsf{fin}}\mathsf{X}.\psi$-type. Because $\exists_{\mathsf{fin}}\mathsf{X}.\psi$-type is a set of all possible ψ-types, we can choose elements of $S_{1,j}$ (and later of $S_{j,j}$) to X_2, so that the ψ-type is the same as originally in $S_{1,i}$, although without any guarantees on the size of the new set. Anyway, the length of vaults in T_2 grows two times faster than in T_1, so the above problem concerns only the first $\max(0, k-2\ell)$ vaults, where the number of elements of X_1 is bounded by a constant $c_{k,\ell}$ (depending on k and ℓ). All further elements of X_1 contribute to the size of X_2; the lemma holds with $f(n) = n - c_{k,\ell}$.

Consider now Lemma 5.4, where we have to create a set X_1 in T_1 based on a set X_2 in T_2. There are two cases. Suppose first that at least half of elements of X_2 lie outside of the vaults. In this case we proceed as previously, appropriately stretching and/or contracting the vaults. While there is no size guarantee for vault elements, already by counting elements outside of the vaults we obtain $|X_2| \geq \frac{|X_1|}{2}$.

In the opposite case, we check which label is more frequent among the (at least $\frac{|X_2|}{2}$) vault elements of X_2. Suppose this is a (the case of b is analogous), and that $k = \ell = 0$. We then map every vault $S_{i,i}$ into $S_{i,1}$, contracting only the b-labeled part; all the a-labeled vault elements of X_2 remain in X_1. Because the distance between $S_{i,i}$ and $S_{i+1,i+1}$ in T_2 is $N!$, while the distance between $S_{i,1}$ and $S_{i+1,1}$ in T_1 is $2N!$, we also need to stretch the trunk, which is possible using a similar pumping argument (and we stretch some $S_{1,1}$ into the vault $S_{1,i}$ that should be in the middle between $S_{i,1}$ and $S_{i+1,1}$).

This is almost the end, except that we need to handle arbitrary k, ℓ. To this end, we either stretch the initial fragment of the trunk of length $N!$ into multiple such fragments, or we contract the initial fragment of appropriate length into a fragment of length $N!$, so that the vault lengths become synchronized.

Having Lemmata 5.3 and 5.4 we can conclude that the trees T_1 and T_2 (cf. Definition 5.2) have the same types:

▶ **Lemma 5.5.** *Let φ be a subformula of φ_{SUP}. Then for all $k, \ell \in \mathbb{N}$ we have $[\![T_1{\restriction}\mathsf{R}^{kN!}]\!]^{\varnothing}_{\varphi} = [\![T_2{\restriction}\mathsf{R}^{\ell N!}]\!]^{\varnothing}_{\varphi}$.*

Proof. We proceed by induction on φ, considering all possible forms of the formula. First, note that we only consider the valuation $\varnothing$, mapping all variables to the empty set, so for atomic formulae of the form $a(\mathsf{X})$ or $\mathsf{X} \subseteq \mathsf{Y}$ the φ-type is always tt, and for $\mathsf{X} \between_d \mathsf{Y}$ the φ-type is always empty. For $\varphi = \psi_1 \wedge \psi_2$ the φ-type is just the pair containing the ψ_1-type and the ψ_2-type; for them we have the equality $[\![T_1{\restriction}\mathsf{R}^{kN!}]\!]^{\varnothing}_{\psi_i} = [\![T_2{\restriction}\mathsf{R}^{\ell N!}]\!]^{\varnothing}_{\psi_i}$ by the induction hypothesis. Likewise, for $\varphi = \neg\psi$ the φ-type equals the ψ-type, and we immediately conclude by the induction hypothesis $[\![T_1{\restriction}\mathsf{R}^{kN!}]\!]^{\varnothing}_{\psi} = [\![T_2{\restriction}\mathsf{R}^{\ell N!}]\!]^{\varnothing}_{\psi}$.

Suppose now that $\varphi = \exists_{\mathsf{fin}}\mathsf{X}.\psi$. Then the φ-type of $T_1{\restriction}\mathsf{R}^{kN!}$ is the set of ψ-types $[\![T_1{\restriction}\mathsf{R}^{kN!}]\!]^{\varnothing[\mathsf{X}\mapsto X_1]}_{\psi}$ over all possible finite sets $X_1 \subseteq \mathrm{dom}(T_1{\restriction}\mathsf{R}^{kN!})$, and likewise for T_2. By Lemma 5.3, for every ψ-type of $T_1{\restriction}\mathsf{R}^{kN!}$ there exists a set X_2 giving the same ψ-type for $T_2{\restriction}\mathsf{R}^{\ell N!}$, and conversely by Lemma 5.4, so the two φ-types are equal (recall that N was chosen such that $|Typ_{\exists_{\mathsf{fin}}\mathsf{X}.\psi}| \leq N$ whenever ψ is a subformula of φ_{SUP}, hence the two lemmata can indeed be applied).

Finally, suppose that $\varphi = \mathsf{U}\mathsf{X}.\psi$. Then the φ-type consists of two coordinates. On the first coordinate we simply have the $\exists_{\mathsf{fin}}\mathsf{X}.\psi$-type – these types are equal for $T_1{\restriction}\mathsf{R}^{kN!}$ and $T_2{\restriction}\mathsf{R}^{\ell N!}$ by the previous case. On the second coordinate we have the set of ψ-types τ such

that $[\![T_1 \restriction \mathsf{R}^{kN!}]\!]_\psi^{\varnothing[\mathsf{X}\mapsto X_1]} = \tau$ for arbitrarily large finite sets X_1, and likewise for X_2. But $[\![T_1 \restriction \mathsf{R}^{kN!}]\!]_\psi^{\varnothing[\mathsf{X}\mapsto X_1]} = \tau$ for arbitrarily large finite sets X_1 if and only if $[\![T_2 \restriction \mathsf{R}^{\ell N!}]\!]_\psi^{\varnothing[\mathsf{X}\mapsto X_2]} = \tau$ for arbitrarily large finite sets X_2, by Lemmata 5.3 and 5.4. This gives us equality of the two φ-types.

Recall that by assumption φ_{SUP} is a formula of WMSO+U, without quantification over tuples, so the above exhausts all possible cases. ◀

Lemma 5.5 implies in particular that $[\![T_1]\!]_{\varphi_{SUP}}^{\varnothing} = [\![T_2]\!]_{\varphi_{SUP}}^{\varnothing}$, which by Fact 3.2 means that φ_{SUP} is satisfied in T_1 if and only if it is satisfied in T_2. This way we reach a contradiction with the fact that φ_{SUP} should be true in T_1, but not in T_2. Thus, the simultaneous unboundedness property for two letters cannot be expressed by a formula φ_{SUP} not involving the U quantifiers for tuples of variables; we obtain Theorem 1.1.

▶ Remark 5.6. We have shown that SUP with respect to a two-element set $\{\mathsf{a}, \mathsf{b}\}$ cannot be expressed without quantification over pairs of variables. It is easy to believe that using a very similar proof one can show that SUP with respect to a k-element set cannot be expressed without quantification over k-tuples of variables, for every $k \geq 2$.

References

1 Alfred V. Aho. Indexed grammars – an extension of context-free grammars. *J. ACM*, 15(4):647–671, 1968. doi:10.1145/321479.321488.

2 David Barozzini, Lorenzo Clemente, Thomas Colcombet, and Paweł Parys. Cost automata, safe schemes, and downward closures. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 109:1–109:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.109.

3 Mikołaj Bojańczyk. A bounding quantifier. In Jerzy Marcinkowski and Andrzej Tarlecki, editors, *Computer Science Logic, 18th International Workshop, CSL 2004, 13th Annual Conference of the EACSL, Karpacz, Poland, September 20-24, 2004, Proceedings*, volume 3210 of *Lecture Notes in Computer Science*, pages 41–55. Springer, 2004. doi:10.1007/978-3-540-30124-0_7.

4 Mikołaj Bojańczyk. Weak MSO with the unbounding quantifier. *Theory Comput. Syst.*, 48(3):554–576, 2011. doi:10.1007/s00224-010-9279-2.

5 Mikołaj Bojańczyk. Weak MSO+U with path quantifiers over infinite trees. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming – 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part II*, volume 8573 of *Lecture Notes in Computer Science*, pages 38–49. Springer, 2014. doi:10.1007/978-3-662-43951-7_4.

6 Mikołaj Bojańczyk and Thomas Colcombet. Bounds in ω-regularity. In *21th IEEE Symposium on Logic in Computer Science (LICS 2006), 12-15 August 2006, Seattle, WA, USA, Proceedings*, pages 285–296. IEEE Computer Society, 2006. doi:10.1109/LICS.2006.17.

7 Mikołaj Bojańczyk and Szymon Toruńczyk. Weak MSO+U over infinite trees. In Christoph Dürr and Thomas Wilke, editors, *29th International Symposium on Theoretical Aspects of Computer Science, STACS 2012, February 29th – March 3rd, 2012, Paris, France*, volume 14 of *LIPIcs*, pages 648–660. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2012. doi:10.4230/LIPIcs.STACS.2012.648.

8 Tomás Brázdil, Krishnendu Chatterjee, Antonín Kucera, and Petr Novotný. Efficient controller synthesis for consumption games with multiple resource types. In P. Madhusudan and Sanjit A. Seshia, editors, *Computer Aided Verification – 24th International Conference, CAV 2012, Berkeley, CA, USA, July 7-13, 2012 Proceedings*, volume 7358 of *Lecture Notes in Computer Science*, pages 23–38. Springer, 2012. doi:10.1007/978-3-642-31424-7_8.

9 Luca Breveglieri, Alessandra Cherubini, Claudio Citrini, and Stefano Crespi-Reghizzi. Multi-push-down languages and grammars. *Int. J. Found. Comput. Sci.*, 7(3):253–292, 1996. doi:10.1142/S0129054196000191.

10 Christopher H. Broadbent, Arnaud Carayol, C.-H. Luke Ong, and Olivier Serre. Higher-order recursion schemes and collapsible pushdown automata: Logical properties. *ACM Trans. Comput. Log.*, 22(2):12:1–12:37, 2021. doi:10.1145/3452917.

11 Julius Richard Büchi. On a decision method in restricted second order arithmetic. In *Proceedings of the 1960 International Congress on Logic, Methodology and Philosophy of Science*, pages 1–11. Stanford University Press, 1962.

12 Claudia Carapelle, Alexander Kartzow, and Markus Lohrey. Satisfiability of CTL* with constraints. In Pedro R. D'Argenio and Hernán C. Melgratti, editors, *CONCUR 2013 – Concurrency Theory – 24th International Conference, CONCUR 2013, Buenos Aires, Argentina, August 27-30, 2013. Proceedings*, volume 8052 of *Lecture Notes in Computer Science*, pages 455–469. Springer, 2013. doi:10.1007/978-3-642-40184-8_32.

13 Lorenzo Clemente, Paweł Parys, Sylvain Salvati, and Igor Walukiewicz. Ordered tree-pushdown systems. In Prahladh Harsha and G. Ramalingam, editors, *35th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2015, December 16-18, 2015, Bangalore, India*, volume 45 of *LIPIcs*, pages 163–177. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.FSTTCS.2015.163.

14 Lorenzo Clemente, Paweł Parys, Sylvain Salvati, and Igor Walukiewicz. The diagonal problem for higher-order recursion schemes is decidable. In Martin Grohe, Eric Koskinen, and Natarajan Shankar, editors, *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 96–105. ACM, 2016. doi:10.1145/2933575.2934527.

15 Thomas Colcombet. The theory of stabilisation monoids and regular cost functions. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris E. Nikoletseas, and Wolfgang Thomas, editors, *Automata, Languages and Programming, 36th Internatilonal Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part II*, volume 5556 of *Lecture Notes in Computer Science*, pages 139–150. Springer, 2009. doi:10.1007/978-3-642-02930-1_12.

16 Wojciech Czerwiński, Wim Martens, Lorijn van Rooijen, Marc Zeitoun, and Georg Zetzsche. A characterization for decidable separability by piecewise testable languages. *Discret. Math. Theor. Comput. Sci.*, 19(4), 2017. doi:10.23638/DMTCS-19-4-1.

17 Werner Damm. The IO- and OI-hierarchies. *Theor. Comput. Sci.*, 20:95–207, 1982. doi:10.1016/0304-3975(82)90009-3.

18 Calvin C. Elgot. Decision problems of finite automata design and related arithmetics. *Trans. Amer. Math. Soc.*, 98(1):21–51, 1961. doi:10.2307/1993511.

19 E. Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1-4 October 1991*, pages 368–377. IEEE Computer Society, 1991. doi:10.1109/SFCS.1991.185392.

20 Nathanaël Fijalkow and Martin Zimmermann. Cost-parity and cost-Streett games. In Deepak D'Souza, Telikepalli Kavitha, and Jaikumar Radhakrishnan, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2012, December 15-17, 2012, Hyderabad, India*, volume 18 of *LIPIcs*, pages 124–135. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2012. doi:10.4230/LIPIcs.FSTTCS.2012.124.

21 Matthew Hague, Jonathan Kochems, and C.-H. Luke Ong. Unboundedness and downward closures of higher-order pushdown automata. In Rastislav Bodík and Rupak Majumdar, editors, *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20 – 22, 2016*, pages 151–163. ACM, 2016. doi:10.1145/2837614.2837627.

22 Matthew Hague, Andrzej S. Murawski, C.-H. Luke Ong, and Olivier Serre. Collapsible pushdown automata and recursion schemes. *ACM Trans. Comput. Log.*, 18(3):25:1–25:42, 2017. doi:10.1145/3091122.

23 Teodor Knapik, Damian Niwiński, and Paweł Urzyczyn. Higher-order pushdown trees are easy. In Mogens Nielsen and Uffe Engberg, editors, *Foundations of Software Science and Computation Structures, 5th International Conference, FOSSACS 2002. Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2002 Grenoble, France, April 8-12, 2002, Proceedings*, volume 2303 of *Lecture Notes in Computer Science*, pages 205–222. Springer, 2002. doi:10.1007/3-540-45931-6_15.

24 Naoki Kobayashi. Model checking higher-order programs. *J. ACM*, 60(3):20:1–20:62, 2013. doi:10.1145/2487241.2487246.

25 Naoki Kobayashi and C.-H. Luke Ong. A type system equivalent to the modal mu-calculus model checking of higher-order recursion schemes. In *Proceedings of the 24th Annual IEEE Symposium on Logic in Computer Science, LICS 2009, 11-14 August 2009, Los Angeles, CA, USA*, pages 179–188. IEEE Computer Society, 2009. doi:10.1109/LICS.2009.29.

26 Orna Kupferman, Nir Piterman, and Moshe Y. Vardi. From liveness to promptness. *Formal Methods Syst. Des.*, 34(2):83–103, 2009. doi:10.1007/s10703-009-0067-z.

27 C.-H. Luke Ong. On model-checking trees generated by higher-order recursion schemes. In *21th IEEE Symposium on Logic in Computer Science (LICS 2006), 12-15 August 2006, Seattle, WA, USA, Proceedings*, pages 81–90. IEEE Computer Society, 2006. doi:10.1109/LICS.2006.38.

28 Paweł Parys. Recursion schemes and the WMSO+U logic. In Rolf Niedermeier and Brigitte Vallée, editors, *35th Symposium on Theoretical Aspects of Computer Science, STACS 2018, February 28 to March 3, 2018, Caen, France*, volume 96 of *LIPIcs*, pages 53:1–53:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.STACS.2018.53.

29 Paweł Parys. Recursion schemes, the MSO logic, and the U quantifier. *Log. Methods Comput. Sci.*, 16(1), 2020. doi:10.23638/LMCS-16(1:20)2020.

30 Paweł Parys. A type system describing unboundedness. *Discret. Math. Theor. Comput. Sci.*, 22(4), 2020. doi:10.23638/DMTCS-22-4-2.

31 Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Trans. Amer. Math. Soc.*, 141:1–35, 1969. doi:10.2307/1995086.

32 Sylvain Salvati and Igor Walukiewicz. Krivine machines and higher-order schemes. *Inf. Comput.*, 239:340–355, 2014. doi:10.1016/j.ic.2014.07.012.

33 Sylvain Salvati and Igor Walukiewicz. A model for behavioural properties of higher-order programs. In Stephan Kreutzer, editor, *24th EACSL Annual Conference on Computer Science Logic, CSL 2015, September 7-10, 2015, Berlin, Germany*, volume 41 of *LIPIcs*, pages 229–243. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.CSL.2015.229.

34 Sylvain Salvati and Igor Walukiewicz. Simply typed fixpoint calculus and collapsible pushdown automata. *Math. Struct. Comput. Sci.*, 26(7):1304–1350, 2016. doi:10.1017/S0960129514000590.

35 Boris Trakhtenbrot. Finite automata and the logic of monadic predicates. *Doklady Akademii Nauk SSSR*, 140:326–329, 1961.

36 Georg Zetzsche. An approach to computing downward closures. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Automata, Languages, and Programming – 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part II*, volume 9135 of *Lecture Notes in Computer Science*, pages 440–451. Springer, 2015. doi:10.1007/978-3-662-47666-6_35.

A Natural Intuitionistic Modal Logic: Axiomatization and Bi-Nested Calculus

Philippe Balbiani
CNRS-INPT-UT3, IRIT, Toulouse, France

Han Gao
Aix-Marseille University, CNRS, LIS, Marseille, France

Çiğdem Gencer
CNRS-INPT-UT3, IRIT, Toulouse, France

Nicola Olivetti
Aix-Marseille University, CNRS, LIS, Marseille, France

Abstract

We introduce **FIK**, a natural intuitionistic modal logic specified by Kripke models satisfying the condition of forward confluence. We give a complete Hilbert-style axiomatization of this logic and propose a bi-nested calculus for it. The calculus provides a decision procedure as well as a countermodel extraction: from any failed derivation of a given formula, we obtain by the calculus a finite countermodel of it directly.

2012 ACM Subject Classification Theory of computation → Modal and temporal logics; Theory of computation → Proof theory

Keywords and phrases Intuitionistic Modal Logic, Axiomatization, Completeness, Sequent Calculus

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.13

Related Version *Full Version*: `https://doi.org/10.48550/arXiv.2309.06309`

Acknowledgements This paper is originated from a discussion started by Anupam Das and Sonia Marin in the proof theory blog (see the link `https://prooftheory.blog/2022/08/19/`), we are grateful to them as well as all other contributors to the discussion. In particular, Example 24 was reported in the blog by Alex Simpson, who learnt it in 1996 by Carsten Grefe in a private communication. Example 55 was suggested first by Anupam Das and Sonia Marin in the blog. Finally we thank the reviewers for their very helpful criticisms and insightful remarks.

1 Introduction

Intuitionistic modal logic (**IML**) has a long history, starting from the pioneering work by Fitch [5] in the late 40's and Prawitz [12] in the 60's. Along the time, two traditions emerged that led to the study of two different families of systems. The first tradition, called *intuitionistic modal logics*, has been introduced by Fischer Servi [13, 14, 15], Plotkin and Stirling [11] and then systematized by Simpson [16]. Its main goal is to define an analogous of classical modalities justified from an intuitionistic meta-theory. The basic modal logic in this tradition, **IK**, is intended to be the intuitionistic counterpart of the minimal normal modal logic **K**. The second tradition leads to so-called *constructive modal logics* that are mainly motivated by their applications in computer science such as type-theoretic interpretations, verification and knowledge representation (contextual reasoning). This second tradition has been developed independently, first by Wijesekera [17] who proposed the system **CCDL** (Constructive Concurrent Dynamic logic), and then by Bellin, De Paiva, and Ritter [2], among others who proposed the logic **CK** (Constructive **K**) as the basic system for a constructive account of modality.

But putting aside the historical perspective, we can consider naively the following question: how can we build "from scratch" an **IML**? Since both modal logic and intuitionistic logic enjoy Kripke semantics, we can think of combining them together in order to define an intuitionistic modal logic. The simplest proposal is to consider Kripke models equipped with two relations, $\leq$ for intuitionistic implication and R for modalities. Propositional intuitionistic connectives (in particular implication) have their usual interpretations. We request that every valid formula or rule scheme of propositional intuitionistic logic **IPL** is also valid in **IML**. To reach this goal, we must ensure the *hereditary property*, which means for any formula A, if A is forced by a world, it will also be forced also by all its uppers worlds, namely:

if $x \Vdash A$ and $x \leq y$ then also $y \Vdash A$.

Thus the question becomes how to define modalities in order to ensure this property. The simplest solution is to build the hereditary property in the forcing conditions for $\Box$ and $\Diamond$:

(1) $x \Vdash \Box A$ iff for all x' with $x' \geq x$, for all y with $Rx'y$ it holds $y \Vdash A$ and
(1') $x \Vdash \Diamond A$ iff for all x' with $x' \geq x$, there exists y with $Rx'y$ s.t. $y \Vdash A$.

Observe that the definition of $\Box A$ is reminiscent of the definition of $\forall$ in intuitionistic first-order logic. This logic is nothing else than the propositional part of Wijeskera's **CCDL** mentioned above and is *non-normal* as it does not contain all formulas of the form

(DP) $\Diamond(A \vee B) \supset \Diamond A \vee \Diamond B$.

Moreover, the logic does not satisfy the maximality criteria, one of the criteria stated by Simpson [16, Chapter 3] for a "good" **IML** since by adding any classical principle to it, we cannot get the classical normal modal logic **K**. In addition, **CCDL** has also been criticized for being *too strong*, as it still satisfies the *nullary* $\Diamond$ distribution: $\Diamond\bot \supset \bot$. By removing this last axiom, the constructive modal logic **CK** is obtained.

However, the opposite direction is also possible: we can make local the definition of $\Diamond$ (pursuing the analogy with $\exists$ in intuitionistic first-order logic **FOIL**) exactly as in classical **K**, that is:

(2) $x \Vdash \Diamond A$ iff there exists y with Rxy s.t. $y \Vdash A$.

In this way we recover $\Diamond(A \vee B) \supset \Diamond A \vee \Diamond B$, making the logic *normal*. But there is a price to pay: nothing ensures that the hereditary property holds for $\Diamond$-formulas. In order to solve this problem, we need to postulate some frame conditions. The most natural (and maybe the weakest) condition is simply that if $x' \geq x$ and x has an R-accessible y then also x' must have an R-accessible y' which refines y, which means $y' \geq y$. This condition is called *Forward Confluence* in [1]. It is not new as it is also called (F1) by Simpson [16, Chapter 3] and together with another frame conditions (F2) characterizes the very well-known system **IK** by Fischer-Servi and Simpson. Although from a meta-theoretical point of view **IK** can be justified by its standard translation in first-order intuitionistic logic, it does not seem to be the minimal system allowing the definition of modalities as in (1) and (2) above.

This paper attempts to fill the gap by studying a weaker logic for which the forcing conditions for modalities are just (1) and (2) above and we assume *only* Forward Confluence for the frames. We call this logic **FIK** for *forward confluenced* **IK**. As far as we know, this logic has never been studied before. And we think it is well worth being studied since it seems to be the minimal logic defined by bi-relational models with forcing conditions (1) and (2) which preserves intuitionistic validity.

In the following sections, we first give a sound and complete Hilbert axiomatization of **FIK**. We show that **FIK** finds its place in the **IML**/constructive family: it is strictly stronger than **CCDL** (whence than **CK**) and strictly weaker than **IK**. At the same time **FIK** seems acceptable to be regarded as an **IML** since it satisfies *all* criteria proposed by Simpson, including the one about maximality, which means by adding any classical principle to **FIK**, we can get the classical normal modal logic **K**. All in all **FIK** seems to be a respectable intuitionistic modal logic and is a kind of "third way" between intuitionistic **IK** and constructive **CCDL**/**CK**.

We then investigate **FIK** from a proof-theoretic viewpoint. We propose a nested sequent calculus $\mathbf{C_{FIK}}$ which makes use of two kinds of nestings, one for representing $\geq$-upper worlds and the other for R-related worlds. A nested sequent calculus for (first-order) intuitionistic logic that exploits the first type of nesting has been proposed in [6], so our calculus can be seen as an extension of the propositional part of it. More recently in [4], the authors present a sequent calculus with the same kind of nesting to capture the **IML** logic given by $\mathbf{CCDL} + (DP)$ [1].

As mentioned, our calculus contains a double type of nesting. The use of this double nesting is somewhat analogous to the labelled calculus proposed in [10] which introduces two kinds of relations on labels in the syntax. However, the essential ingredient of our calculus $\mathbf{C_{FIK}}$ is the *interactive rule* between the two kinds of nested sequents that captures the specific Forward Confluence condition.

We also prove that the calculus $\mathbf{C_{FIK}}$ provides a decision procedure for the logic **FIK**. In addition, since the rules of $\mathbf{C_{FIK}}$ are invertible, we show that from a failed derivation under a suitable strategy, it is possible to extract a finite countermodel of the formula or sequent at the root of the derivation. This result allows us to obtain a constructive proof of the finite model property, which means if a formula is not valid then it has a finite countermodel.

2 A natural intuitionistic modal logic

Firstly, we present the syntax and semantics of forward confluenced intuitionistic modal logic **FIK**. Secondly, we present an axiom system and we prove its soundness and completeness. Thirdly, we discuss whether **FIK** satisfies the properties that are expected from intuitionistic modal logics.

▶ **Definition 1** (Formulas). *The set $\mathcal{L}$ of all formulas (denoted as A, B, etc.) is generated by the following grammar: $A ::= p \mid \bot \mid \top \mid (A \wedge A) \mid (A \vee A) \mid (A \supset A) \mid \Box A \mid \Diamond A$ where p ranges over a countable set of atomic propositions At. We omit parentheses for readability. For all formulas A, we write $\neg A$ instead of $A \supset \bot$. For all formulas A, B, we write $A \equiv B$ instead of $(A \supset B) \wedge (B \supset A)$. The size of a formula A is denoted $|A|$.*

▶ **Definition 2** (Bi-relational model). *A bi-relational model is a quadruple $\mathcal{M} = (W, \leq, R, V)$ where W is a nonempty set of worlds, $\leq$ is a pre-order on W, R is a binary relation on W and $V : W \longrightarrow \wp(\mathsf{At})$ is a valuation on W satisfying the following hereditary condition:*

$$\forall x, y \in W, \ (x \leq y \ \Rightarrow \ V(x) \subseteq V(y)).$$

The triple $(W, \leq, R)$ is called a frame. For all $x, y \in W$, we write $x \geq y$ instead of $y \leq x$. Moreover, we say "y is a successor of x" when Rxy.

[1] A calculus for **IK** with the same kind of nesting was also preliminarily considered in [9]

It is worth mentioning that an upper world of a successor of a world is not necessarily a successor of an upper world of that world. However, from now on in this paper, we only consider models $\mathcal{M} = (W, \leq, R, V)$ that satisfy the following condition called *Forward Confluence* as in [1]:

(FC) $\forall x, y \in W,\ (\exists z \in W,\ (x \geq z\ \&\ Rzy)\ \Rightarrow\ \exists t \in W,\ (Rxt\ \&\ t \geq y))$.

▶ **Definition 3** (Forcing relation). *Let $\mathcal{M} = (W, \leq, R, V)$ be a bi-relational model and $w \in W$. The forcing conditions are the usual ones for atomic propositions and for formulas constructed by means of the connectives $\bot, \top, \wedge$ and $\vee$. For formulas constructed by means of the connectives $\supset$, $\Box$ and $\Diamond$, the forcing conditions are as follows:*

- *$\mathcal{M}, w \Vdash B \supset C$ iff for all $w' \in W$ with $w \leq w'$ and $\mathcal{M}, w' \Vdash B$, $\mathcal{M}, w' \Vdash C$;*
- *$\mathcal{M}, w \Vdash \Box B$ iff for all $w', v' \in W$ with $w \leq w'$ and $Rw'v'$, $v' \Vdash B$;*
- *$\mathcal{M}, w \Vdash \Diamond B$ iff there exists $v \in W$ with Rwv and $\mathcal{M}, v \Vdash B$.*

We also abbreviate $\mathcal{M}, w \Vdash A$ as $w \Vdash A$ if the model is clear from the context.

▶ **Proposition 4.** *Let $(W, \leq, R, V)$ be a bi-relational model. For all formulas A in $\mathcal{L}$ and for all $x, y \in W$ with $x \leq y$, $x \Vdash A$ implies $y \Vdash A$.*

Proposition 4 is proved by induction on the size of A using (FC) for the case of $A = \Diamond B$.

▶ **Definition 5** (Validity). *A formula A in $\mathcal{L}$ is valid, denoted $\Vdash A$, if for any bi-relational model $\mathcal{M}$ and any world w in it, $\mathcal{M}, w \Vdash A$. Let **FIK** be the set of all valid formulas.*

Obviously, **FIK** contains all standard axioms of **IPL**. Moreover, **FIK** is closed with respect to the following inference rules:

$$\frac{p \supset q, p}{q}\ (\mathbf{MP}) \qquad \frac{p}{\Box p}\ (\mathbf{NEC})$$

Finally, **FIK** contains the following formulas:

($\mathbf{K}_\Box$) $\Box(p \supset q) \supset (\Box p \supset \Box q)$,
($\mathbf{K}_\Diamond$) $\Box(p \supset q) \supset (\Diamond p \supset \Diamond q)$,
(**N**) $\neg\Diamond\bot$,
(**DP**) $\Diamond(p \vee q) \supset \Diamond p \vee \Diamond q$,
(**wCD**) $\Box(p \vee q) \supset ((\Diamond p \supset \Box q) \supset \Box q)$.

We only show the validity of (**wCD**). Suppose $\nVdash \Box(p \vee q) \supset ((\Diamond p \supset \Box q) \supset \Box q)$. Hence, there exists a model $(W, \leq, R, V)$ and $w \in W$ such that $w \Vdash \Box(p \vee q)$, $w \Vdash \Diamond p \supset \Box q$ and $w \nVdash \Box q$. Thus, let $u, v \in W$ be such that $w \leq u$, Ruv and $v \nVdash q$. Since $w \Vdash \Box(p \vee q)$, $v \Vdash p \vee q$. Since $v \nVdash q$, $v \Vdash p$. Since Ruv, $u \Vdash \Diamond p$. Since $w \Vdash \Diamond p \supset \Box q$ and $w \leq u$, $u \Vdash \Diamond p \supset \Box q$. Since $u \Vdash \Diamond p$, $u \Vdash \Box q$. Since Ruv, $v \Vdash q$: a contradiction.

▶ **Definition 6** (Axiom system). *Let $\mathbf{D}_{\boldsymbol{FIK}}$ be the Hilbert-style axiom system consisting of all standard axioms of **IPL**, the inference rules* (**MP**) *and* (**NEC**) *and the formulas* ($\mathbf{K}_\Box$), ($\mathbf{K}_\Diamond$), (**N**), (**DP**) *and* (**wCD**) *considered as axioms. Derivations are defined as usual. For all formulas A, we write $\vdash A$ when A is $\mathbf{D}_{\boldsymbol{FIK}}$-derivable. The set of all $\mathbf{D}_{\boldsymbol{FIK}}$-derivable formulas will also be denoted $\mathbf{D}_{\boldsymbol{FIK}}$.*

The formulas ($\mathbf{K}_\Box$), ($\mathbf{K}_\Diamond$), (**DP**) and (**N**) are not new, seeing that they have already been used by many authors as axioms in multifarious variants of **IML**. As for the formula (**wCD**), as far as we are aware, it is used here for the first time as an axiom of an **IML** variant. Indeed, (**wCD**) is derivable in **IK**. Moreover, it is a weak form of the *Constant Domain*

axiom (**CD**) : $\Box(p \vee q) \supset \Diamond p \vee \Box q$ used in [1]. In other respect, (**wCD**) is derivable in **IK**, whereas it is not derivable in **CCDL**/**CK**. As for the **IK** axiom $(\Diamond p \supset \Box q) \supset \Box(p \supset q)$, it is not in **FIK** as it will be also constructively shown by using the calculus presented in next section. Therefore, we get **CK**⊂**CCDL**⊂**FIK**⊂**IK**. We can consider also the logic **CCDL** + (**DP**) (= **CK** + (**N**) + (**DP**)) recently studied in [4], according to the results in that paper, we get that **CCDL** + (**DP**) ⊂ **FIK**.

▶ **Theorem 7** (Soundness). $\mathbf{D}_{\boldsymbol{FIK}} \subseteq \boldsymbol{FIK}$, *i.e. for all formulas* A, *if* $\vdash A$ *then* $\Vdash A$.

Theorem 7 can be proved by induction on the length of the derivation of A. Later, we will prove the converse inclusion (Completeness) saying that $\mathbf{FIK} \subseteq \mathbf{D}_{\mathbf{FIK}}$. At the heart of our proof of completeness, will be the concept of theory.

▶ **Definition 8** (Theories). *A theory is a set of formulas containing* $\mathbf{D}_{\boldsymbol{FIK}}$ *and closed with respect to* **MP**. *A theory* Γ *is proper if* $\bot \notin \Gamma$. *A proper theory* Γ *is prime if for all formulas* A, B, *if* $A \vee B \in \Gamma$ *then either* $A \in \Gamma$, *or* $B \in \Gamma$. *For all theories* Γ *and for all formulas* A, *let* $\Gamma + A = \{B \in \mathcal{L} : \ A \supset B \in \Gamma\}$ *and* $\Box\Gamma = \{A \in \mathcal{L} : \ \Box A \in \Gamma\}$.

Obviously, $\mathbf{D}_{\mathbf{FIK}}$ is the least theory and $\mathcal{L}$ is the greatest theory. Moreover, for all theories Γ, Γ is proper if and only if $\Gamma \neq \mathcal{L}$ if and only if $\Diamond\bot \notin \Gamma$.

▶ **Lemma 9.** *For all theories* Γ *and for all formulas* A, *(i)* $\Gamma + A$ *is the least theory containing* Γ *and* A; *(ii)* $\Gamma + A$ *is proper if and only if* $\neg A \notin \Gamma$; *(iii)* $\Box\Gamma$ *is a theory.*

Lemma 9 can be proved by using standard axioms of **IPL**, inference rules (**MP**) and (**NEC**) and axiom $\mathbf{K}_{\Box}$.

▶ **Lemma 10** (Lindenbaum's Lemma). *Let* A *be a formula. If* $A \notin \mathbf{D}_{\boldsymbol{FIK}}$ *then there exists a prime theory* Γ *such that* $A \notin \Gamma$.

▶ **Definition 11** (Canonical model). *Let* $\bowtie$ *be the binary relation between sets of formulas such that for all sets* Δ, Λ *of formulas,* $\Delta \bowtie \Lambda$ *iff for all formulas* B, *the following conditions hold: (i) if* $\Box B \in \Delta$ *then* $B \in \Lambda$ *and (ii) if* $B \in \Lambda$ *then* $\Diamond B \in \Delta$.

Let $(W_c, \leq_c, R_c)$ *be the frame such that* W_c *is the set of all prime theories,* $\leq_c$ *is the inclusion relation on* W_c *and* R_c *is the restriction of* $\bowtie$ *to* W_c. *For all* $\Gamma, \Delta \in W_c$, *we write "*$\Gamma \geq_c \Delta$*" instead of "*$\Delta \leq_c \Gamma$*". Let* $V_c : \ W_c \longrightarrow \wp(\mathsf{At})$ *be the valuation on* W_c *such that for all* Γ *in* W_c, $V_c(\Gamma) = \Gamma \cap \mathsf{At}$.

By Theorem 7, $\bot \notin \mathbf{D}_{\mathbf{FIK}}$. Hence, by Lemma 10, W_c is nonempty.

▶ **Lemma 12.** $(W_c, \leq_c, R_c, V_c)$ *satisfies the frame condition* **(FC)**.

The proof of the completeness will be based on the following lemmas.

▶ **Lemma 13** (Existence Lemma). *Let* Γ *be a prime theory and* B, C *be formulas.*

1. *If* $B \supset C \notin \Gamma$ *then there exists a prime theory* Δ *such that* $\Gamma \subseteq \Delta$, $B \in \Delta$ *and* $C \notin \Delta$,
2. *if* $\Box B \notin \Gamma$ *then there exists prime theories* Δ, Λ *such that* $\Gamma \subseteq \Delta$, $\Delta \bowtie \Lambda$ *and* $B \notin \Lambda$,
3. *if* $\Diamond B \in \Gamma$ *then there exists a prime theory* Δ *such that* $\Gamma \bowtie \Delta$ *and* $B \in \Delta$.

▶ **Lemma 14** (Truth Lemma). *For all formulas* A *and for all* $\Gamma \in W_c$, $A \in \Gamma$ *if and only if* $\Gamma \Vdash A$.

The proof of Lemma 14 can be done by induction on the size of A. The case when A is an atomic proposition is by definition of V_c. The cases when A is of the form $\bot, \top$, $B \wedge C$ and $B \vee C$ are as usual. The cases when A is of the form $B \supset C$, $\Box B$ and $\Diamond B$ use the Existence Lemma.

As for the proof of Theorem 15, it can be done by contraposition. Indeed, if $\nvdash A$ then by Lemma 10, there exists a prime theory Γ such that $A \notin \Gamma$. Thus, by Lemma 14, $\Gamma \nVdash A$. Consequently, $\nVdash A$.

▶ **Theorem 15** (Completeness). ***FIK*** $\subseteq \mathbf{D}_{\boldsymbol{FIK}}$*, i.e. for all formulas* A*, if* $\Vdash A$ *then* $\vdash A$*.*

As mentioned above, there exists many variants of **IML**. Therefore, one may ask how *natural* is the variant we consider here. Simpson [16, Chapter 3] discusses the formal features that might be expected of an **IML** **L**:

(C_1) **L** is conservative over **IPL**,
(C_2) **L** contains all substitution instances of **IPL** and is closed under (**MP**),
(C_3) for all formulas A, B, if $A \vee B$ is in **L** then either A is in **L**, or B is in **L**,
(C_4) the addition of the law of excluded middle to **L** yields modal logic **K**,
(C_5) $\Box$ and $\Diamond$ are independent in **L**.

The fact that $\mathbf{D_{FIK}}$ satisfies features (C_1) and (C_2) is an immediate consequence of Theorems 7 and 15. The fact that $\mathbf{D_{FIK}}$ satisfies feature (C_3) will be proved in Section 3. Concerning feature (C_4), let $\mathbf{D_{FIK}}^+$ be the Hilbert-style axiom system consisting of $\mathbf{D_{FIK}}$ plus the law $p \vee \neg p$ of excluded middle. The set of all $\mathbf{D_{FIK}}^+$-derivable formulas will also be denoted $\mathbf{D_{FIK}}^+$. Obviously, $\mathbf{D_{FIK}}^+$ contains all substitution instances of **CPL** and is closed under (**MP**). Moreover, it contains all substitution instances of ($\mathbf{K}_\Box$) and is closed under (**NEC**). Therefore, in order to prove that $\mathbf{D_{FIK}}$ satisfies feature (C_4), it suffices to prove

▶ **Lemma 16.** $\Diamond p \equiv \neg\Box\neg p$ *is in* $\mathbf{D}_{\boldsymbol{FIK}}{}^+$*.*

The fact that $\mathbf{D_{FIK}}$ satisfies feature (C_5) is a consequence of

▶ **Lemma 17.** *Let* p *be an atomic proposition. There exists no* $\Box$*-free* A *such that* $\Box p \equiv A$ *is in* $\mathbf{D}_{\boldsymbol{FIK}}$ *and there exists no* $\Diamond$*-free* A *such that* $\Diamond p \equiv A$ *is in* $\mathbf{D}_{\boldsymbol{FIK}}$*.*

Consequently, $\mathbf{D_{FIK}}$ can be considered as a natural intuitionistic modal logic. [2]

3 A bi-nested sequent calculus

In this section, we present a bi-nested calculus for **FIK**. The calculus is two-sided and it makes use of two kinds of nestings, also called blocks $\langle\cdot\rangle$ and $[\cdot]$. The former is called an *implication* block and the latter a *modal* block. The intuition is that implication blocks correspond to upper worlds while modal blocks correspond to R-successors in a bi-relational model. The calculus we present is a conservative extension (with some notational difference) of the nested sequent calculus for **IPL** presented in [6].

[2] Simpson considers a further requirement (C6), in our opinion more controversial: "there is an intuitionistically comprehensible explanation of the meaning of the modalities, relative to which IML is sound and complete". He interprets this as the requirement of soundness and completeness with respect to the obvious (the same as in the classical case) translation of the modalities into first-order intuitionistic logic. The logic **IK** is sound and complete with respect to such a translation, whereas evidently no weaker logic, whence neither **CK**, **CCDL**, nor **FIK** is. However, this does not mean that any other translation is impossible. A wider discussion will be deferred to further work.

▶ **Definition 18** (Bi-nested sequent). *A bi-nested sequent S is defined as follows:*

- $\Rightarrow$ *is a bi-nested sequent (the empty sequent);*
- $\Gamma \Rightarrow B_1, \ldots, B_k, [S_1], \ldots, [S_m], \langle T_1 \rangle, \ldots, \langle T_n \rangle$ *is a bi-nested sequent if* $S_1, \ldots, S_m$, $T_1, \ldots, T_n$ *are bi-nested sequents where* $m, n \geq 0$, *and* Γ *is a finite (possibly empty) multi-set of formulas and* $B_1, \ldots, B_k$ *are formulas.*

We use S, T to denote bi-nested sequents and to simplify wording we will call bi-nested sequents simply by sequents in the rest of this paper. We denote by $|S|$ *the size* of a sequent S intended as the length of S as a string of symbols.

As usual with nested calculi, we need the notion of context in order to specify the rules, as they can be applied to sequents occurring inside other sequents. A *context* is of the form $G\{\}$, in which G is a part of a sequent, $\{\cdot\}$ is regarded as a placeholder that needs to be filled by another sequent in order to complete G. $G\{S\}$ is the sequent obtained by replacing the occurrence of the symbol $\{\}$ in $G\{\}$ by the sequent S.

▶ **Definition 19** (Context). *A context $G\{\}$ is inductively defined as follows:*

- $\{\}$ *is a context (the empty context).*
- *if* $\Gamma \Rightarrow \Delta$ *is a sequent and* $G'\{\}$ *is a context then* $\Gamma \Rightarrow \Delta, \langle G'\{\} \rangle$ *is a context.*
- *if* $\Gamma \Rightarrow \Delta$ *is a sequent and* $G'\{\}$ *is a context then* $\Gamma \Rightarrow \Delta, [G'\{\}]$ *is a context.*

For example, given a context $G\{\} = A \wedge B, \Box C \Rightarrow \langle \Box A \Rightarrow [\Rightarrow B] \rangle, [\{\}]$ and a sequent $S = A \Rightarrow \Delta, [C \Rightarrow B]$, we have $G\{S\} = A \wedge B, \Box C \Rightarrow \langle \Box A \Rightarrow [\Rightarrow B] \rangle, [A \Rightarrow \Delta, [C \Rightarrow B]]$.

The two types of blocks interact by the (inter) rule. In order to define this rule, we need the following:

▶ **Definition 20** ($*$-operator). *Let $\Lambda \Rightarrow \Theta$ be a sequent, we define Θ^* as follows:*

- $\Theta^* = \emptyset$ *if* Θ *is* $[\cdot]$*-free;*
- $\Theta^* = [\Phi_1 \Rightarrow \Psi_1^*], \ldots, [\Phi_k \Rightarrow \Psi_k^*]$ *if* $\Theta = \Theta_0, [\Phi_1 \Rightarrow \Psi_1], \ldots, [\Phi_k \Rightarrow \Psi_k]$ *and* Θ_0 *is* $[\cdot]$*-free.*

By definition, given a sequent $\Lambda \Rightarrow \Theta$, Θ^* is a multi-set of modal blocks. Denote the sequent $G\{S\}$ in the previous example for context by $\Lambda \Rightarrow \Theta$, then by definition, we can see $\Lambda \Rightarrow \Theta^* = A \wedge B, \Box C \Rightarrow [A \Rightarrow [C \Rightarrow]]$.

Now we can give a bi-nested sequent calculus for **FIK** as follows.

▶ **Definition 21.** *The calculus* $\mathbf{C}_{\boldsymbol{FIK}}$ *is given in Figure 1.*

Here is a brief explanation of these rules. As usual, the (id) axiom can be generalized from atoms to formulas. The logical rules, except $(\supset_R)$, are just the standard rules of intuitionistic logic in their nested version. From a backward direction and a semantic point of view, the rule $(\supset_R)$ introduces an implication block, which corresponds to an upper world (in the pre-order). The modal rules create new modal blocks or propagate modal formulas into existing ones, which correspond to R-accessible worlds. The (trans) rule transfers formulas (forced by) lower worlds to upper worlds following the pre-order. This rule is called (Lift) in [6]. Finally, the (inter) rule encodes the (FC) frame condition. It partially transfers "accessible" modal blocks from lower worlds to upper ones and creates new accessible worlds from upper worlds fulfilling the (FC) condition.

We define the modal degree of a sequent, which will be useful when discussing termination.

▶ **Definition 22** (Modal degree). *Modal degree for a formula F, denoted as $md(F)$, is defined as usual:*

- $md(p) = md(\bot) = md(\top) = 0$*;*
- $md(A \circ B) = max(md(A), md(B))$*, for* $\circ \in \{\wedge, \vee, \supset\}$*;*
- $md(\Box A) = md(\Diamond A) = md(A) + 1$.

Axioms:

$$\frac{}{G\{\Gamma, \bot \Rightarrow \Delta\}}\,(\bot_L) \qquad \frac{}{G\{\Gamma \Rightarrow \top, \Delta\}}\,(\top_R) \qquad \frac{}{G\{\Gamma, p \Rightarrow \Delta, p\}}\,(\mathrm{id})$$

Logical rules:

$$\frac{G\{A, B, \Gamma \Rightarrow \Delta\}}{G\{A \wedge B, \Gamma \Rightarrow \Delta\}}\,(\wedge_L) \qquad \frac{G\{\Gamma \Rightarrow \Delta, A\} \qquad G\{\Gamma \Rightarrow \Delta, B\}}{G\{\Gamma \Rightarrow \Delta, A \wedge B\}}\,(\wedge_R)$$

$$\frac{G\{\Gamma, A \Rightarrow \Delta\} \qquad G\{\Gamma, B \Rightarrow \Delta\}}{G\{\Gamma, A \vee B \Rightarrow \Delta\}}\,(\vee_L) \qquad \frac{G\{\Gamma \Rightarrow \Delta, A, B\}}{G\{\Gamma \Rightarrow \Delta, A \vee B\}}\,(\vee_R)$$

$$\frac{G\{\Gamma, A \supset B \Rightarrow A, \Delta\} \qquad G\{\Gamma, B \Rightarrow \Delta\}}{G\{\Gamma, A \supset B \Rightarrow \Delta\}}\,(\supset_L) \qquad \frac{G\{\Gamma \Rightarrow \Delta, \langle A \Rightarrow B\rangle\}}{G\{\Gamma \Rightarrow \Delta, A \supset B\}}\,(\supset_R)$$

$$\frac{G\{\Gamma, \Box A \Rightarrow \Delta, [\Sigma, A \Rightarrow \Pi]\}}{G\{\Gamma, \Box A \Rightarrow \Delta, [\Sigma \Rightarrow \Pi]\}}\,(\Box_L) \qquad \frac{G\{\Gamma \Rightarrow \Delta, \langle \Rightarrow [\Rightarrow A]\rangle\}}{G\{\Gamma \Rightarrow \Delta, \Box A\}}\,(\Box_R)$$

$$\frac{G\{\Gamma \Rightarrow \Delta, [A \Rightarrow]\}}{G\{\Gamma, \Diamond A \Rightarrow \Delta\}}\,(\Diamond_L) \qquad \frac{G\{\Gamma \Rightarrow \Delta, \Diamond A, [\Sigma \Rightarrow \Pi, A]\}}{G\{\Gamma \Rightarrow \Delta, \Diamond A, [\Sigma \Rightarrow \Pi]\}}\,(\Diamond_R)$$

Transferring and interactive rules:

$$\frac{G\{\Gamma, \Gamma' \Rightarrow \Delta, \langle \Gamma', \Sigma \Rightarrow \Pi\rangle\}}{G\{\Gamma, \Gamma' \Rightarrow \Delta, \langle \Sigma \Rightarrow \Pi\rangle\}}\,(\mathrm{trans}) \qquad \frac{G\{\Gamma \Rightarrow \Delta, \langle \Sigma \Rightarrow \Pi, [\Lambda \Rightarrow \Theta^*]\rangle, [\Lambda \Rightarrow \Theta]\}}{G\{\Gamma \Rightarrow \Delta, \langle \Sigma \Rightarrow \Pi\rangle, [\Lambda \Rightarrow \Theta]\}}\,(\mathrm{inter})$$

Figure 1 $\mathbf{C_{FIK}}$.

Further, let Γ *be a finite set of formulas, define* $md(\Gamma) = md(\bigwedge \Gamma)$. *As for a nested sequent* S *of the following form*

$$S = \Gamma \Rightarrow \Delta, [S_1], \ldots, [S_m], \langle T_1\rangle, \ldots, \langle T_n\rangle,$$

we set $md(S) = \max\{md(\Gamma), md(\Delta), md(S_1) + 1, \ldots, md(S_m) + 1, md(T_1), \ldots, md(T_n)\}$.

▶ **Example 23.** Axiom (**wCD**) in $\mathbf{D_{FIK}}$ is provable in $\mathbf{C_{FIK}}$.

Proof. To prove this, it suffices to prove $S = \Diamond p \supset \Box q, \Box(p \vee q) \Rightarrow \Box q$. Let $\Gamma = \Diamond p \supset \Box q, \Box(p \vee q)$ and then a derivation for S, i.e. $\Gamma \Rightarrow \Box q$ is given as below.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \Rightarrow \langle \Gamma \Rightarrow \Diamond p, [p \Rightarrow q, p]\rangle}\,(\mathrm{id})}{\Gamma \Rightarrow \langle \Gamma \Rightarrow \Diamond p, [p \Rightarrow q]\rangle}\,(\Diamond_R) \qquad \dfrac{\dfrac{}{\Gamma \Rightarrow \langle \Box q, \Box(p \vee q) \Rightarrow [q, p \Rightarrow q]\rangle}\,(\mathrm{id})}{\Gamma \Rightarrow \langle \Box q, \Box(p \vee q) \Rightarrow [p \Rightarrow q]\rangle}\,(\Box_L)}{\Gamma \Rightarrow \langle \Gamma \Rightarrow [p \Rightarrow q]\rangle}\,(\supset_L) \qquad \dfrac{}{\Gamma \Rightarrow \langle \Gamma \Rightarrow [q \Rightarrow q]\rangle}\,(\mathrm{id})}{\Gamma \Rightarrow \langle \Gamma \Rightarrow [p \vee q \Rightarrow q]\rangle}\,(\vee_L)}{\Gamma \Rightarrow \langle \Gamma \Rightarrow [\Rightarrow q]\rangle}\,(\Box_L)}{\Gamma \Rightarrow \langle \Rightarrow [\Rightarrow q]\rangle}\,(\mathrm{trans})}{\Gamma \Rightarrow \Box q}\,(\Box_R)$$

Then we are done. ◀

▶ **Example 24.** The formula $(\neg\Box\bot \supset \Box\bot) \supset \Box\bot$ is provable in $\mathbf{C_{FIK}}$.[3].

[3] Note that this $\Diamond$-free formula is unprovable in **CK** (whence the $\Diamond$-free fragments of these two logics are different, see [4]).

Proof. To prove this, it suffices to prove $S = \neg\Box\bot \supset \Box\bot \Rightarrow \Box\bot$. Let $\Gamma = \neg\Box\bot \supset \Box\bot$ and then a derivation for S, i.e. $\Gamma \Rightarrow \Box\bot$ is given as below.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \Rightarrow \langle \Gamma \Rightarrow \langle \Box\bot \Rightarrow \bot, [\bot \Rightarrow] \rangle, [\Rightarrow \bot] \rangle}(\bot_L)}{\Gamma \Rightarrow \langle \Gamma \Rightarrow \langle \Box\bot \Rightarrow \bot, [\Rightarrow] \rangle, [\Rightarrow \bot] \rangle}(\Box_L)}{\Gamma \Rightarrow \langle \Gamma \Rightarrow \langle \Box\bot \Rightarrow \bot \rangle, [\Rightarrow \bot] \rangle}(\text{inter})}{\Gamma \Rightarrow \langle \Gamma \Rightarrow \neg\Box\bot, [\Rightarrow \bot] \rangle}(\supset_R) \qquad \dfrac{\dfrac{}{\Gamma \Rightarrow \langle \Box\bot \Rightarrow [\bot \Rightarrow \bot] \rangle}(\bot_L)}{\Gamma \Rightarrow \langle \Box\bot \Rightarrow [\Rightarrow \bot] \rangle}(\Box_L)}{\Gamma \Rightarrow \langle \Gamma \Rightarrow [\Rightarrow \bot] \rangle}(\supset_L)}{\Gamma \Rightarrow \langle \Rightarrow [\Rightarrow \bot] \rangle}(\text{trans})}{\Gamma \Rightarrow \Box\bot}(\Box_R)$$

Then we are done. ◀

We now show that the calculus $\mathbf{C_{FIK}}$ enjoys the disjunctive property, which means if $A \vee B$ is provable, then either A or B is provable. This fact is an immediate consequence of the following lemma. Its general form is due to the fact that backwards expansion of a sequent with empty antecedent will (only) treat/introduce formulas and implication blocks in the consequent.

▶ **Lemma 25.** *Suppose that a sequent $S = \Rightarrow A_1, \ldots, A_m, \langle G_1 \rangle, \ldots, \langle G_n \rangle$ is provable in $\mathbf{C_{FIK}}$, where the A_i's are formulas. Then either for some A_i, sequent $\Rightarrow A_i$ is provable or for some G_j, sequent $\Rightarrow \langle G_j \rangle$ is provable.*

Since $\Rightarrow A \vee B$ is provable if and only if $\Rightarrow A, B$ from the lemma we immediately obtain:

▶ **Proposition 26.** *For any formulas A, B, if $\Rightarrow A \vee B$ is provable in $\mathbf{C_{FIK}}$, then either $\Rightarrow A$ or $\Rightarrow B$ is provable.*

By the soundness and completeness of $\mathbf{C_{FIK}}$ with respect to **FIK** proved in the following, we will conclude that the logic **FIK** enjoys the disjunctive property.

Next, we prove the soundness of the calculus $\mathbf{C_{FIK}}$. To achieve this aim, we need to define the semantic interpretation of sequents, whence their validity. We first extend the forcing relation $\Vdash$ to sequents and blocks therein.

▶ **Definition 27.** *Let $\mathcal{M} = (W, \leq, R, V)$ be a bi-relational model and $x \in W$. The relation $\Vdash$ is extended to sequents as follows:*

$\mathcal{M}, x \nVdash \emptyset$
$\mathcal{M}, x \Vdash [T]$ *if for every y with Rxy, $\mathcal{M}, y \Vdash T$*
$\mathcal{M}, x \Vdash \langle T \rangle$ *if for every x' with $x \leq x'$, $\mathcal{M}, x' \Vdash T$*
$\mathcal{M}, x \Vdash \Gamma \Rightarrow \Delta$ *if either $\mathcal{M}, x \nVdash A$ for some $A \in \Gamma$ or $\mathcal{M}, x \Vdash \mathcal{O}$ for some $\mathcal{O} \in \Delta$*

We say S is valid *in $\mathcal{M}$ iff $\forall w \in W$, we have $\mathcal{M}, w \Vdash S$. S is* valid *iff it is valid in every bi-relational model.*

Whenever the model $\mathcal{M}$ is clear, we omit it and write simply $x \Vdash \mathcal{O}$, where $\mathcal{O}$ is either formula, or a sequent, or a block. Moreover, given a sequent $S = \Gamma \Rightarrow \Delta$, we write $x \Vdash \Delta$ if there is an $\mathcal{O} \in \Delta$ s.t. $x \Vdash \mathcal{O}$ and write $x \nVdash \Delta$ if the previous condition does not hold.

The following lemma gives a semantic meaning to the $*$-operation used in (inter).

▶ **Lemma 28.** *Let $\mathcal{M} = (W, \leq, R, V)$ be a bi-relational model and $x, x' \in W$ with $x \leq x'$. Let $S = \Gamma \Rightarrow \Delta$ be any sequent, if $x \nVdash \Delta$ then $x' \nVdash \Delta^*$.*

In order to prove soundness we first show that the all rules are *forcing-preserving*.

▶ **Lemma 29.** *Given a model $\mathcal{M} = (W, \leq, R, V)$ and $x \in W$, for any rule (r) of the form $\frac{G\{S_1\} \quad G\{S_2\}}{G\{S\}}$ or $\frac{G\{S_1\}}{G\{S\}}$, if $x \Vdash G\{S_i\}$, then $x \Vdash G\{S\}$.*

Proof of this lemma proceeds by induction on the structure of the context $G\{\}$. The the base of the induction (that is $G = \emptyset$) is the important one, we check rule by rule and in the case of (inter) we make use of Lemma 28.

By Lemma 29, the soundness of $\mathbf{C_{FIK}}$ is proved as usual by a straightforward induction on the length of derivations.

▶ **Theorem 30** (Soundness). *If a sequent S is provable in $\mathbf{C_{FIK}}$, then it is valid.*

4 Termination and completeness for $\mathbf{C_{FIK}}$

In this section, we provide a terminating proof-search procedure based on $\mathbf{C_{FIK}}$, whence a decision procedure for **FIK**; it will then be used to prove that $\mathbf{C_{FIK}}$ is complete with respect to **FIK** bi-relational semantics. Here is a roadmap. First we introduce a set-based variant of the calculus where all rules are cumulative (or kleen'ed), in the sense that principal formulas are kept in the premises. With this variant, we formulate saturation conditions on a sequent associated to each rule. Saturation conditions are needed for both termination and completeness in order to prevent "redundant" application of the rules as the source of non-termination. In the meantime saturation conditions also ensure that a saturated sequent satisfies the truth conditions specified by the semantics (which will be presented in the truth lemma), so it can be seen as a countermodel.

The reformulation of the calculus by means of set-based sequents is motivated as usual by the following consideration: while multisets are the natural data-structure for any proof-system (at least with commutative $\wedge$, $\vee$), set-based sequents are needed to bound the size of sequents occurring in a derivation in terms of *subsets of subformulas* of the formula or sequent at the root of the derivation (see for instance [3]).

With this in mind, we first present the following $\mathbf{CC_{FIK}}$, a variant of $\mathbf{C_{FIK}}$ where sequents are set-based rather than multi-set based and the rules are cumulative.

▶ **Definition 31.** $\mathbf{CC_{FIK}}$ *acts on set-based sequents, where a set-based sequent $S = \Gamma \Rightarrow \Delta$ is defined as in definition 18, but Γ is a* set *of formulas and Δ is a* set *of formulas and/or blocks (containing set-based sequents). The rules are as follows:*

- *It keeps the rules $(\bot_L)$, $(\top_R)$, (id), $(\Box_L)$, $(\Diamond_R)$, (trans) and (inter) of $\mathbf{C_{FIK}}$.*
- *$(\supset_R)$ is replaced by the two rules dealing with cases of $A \in \Gamma$ and $A \notin \Gamma$ respectively,*

$$\frac{G\{\Gamma \Rightarrow \Delta, A \supset B, B\}}{G\{\Gamma \Rightarrow \Delta, A \supset B\}}\,(\supset_{R_1}) \qquad \frac{G\{\Gamma \Rightarrow \Delta, A \supset B, \langle A \Rightarrow B\rangle\}}{G\{\Gamma \Rightarrow \Delta, A \supset B\}}\,(\supset_{R_2})$$

- *Other rules $(\wedge_L)$, $(\wedge_R)$, $(\vee_L)$, $(\vee_R)$, $(\supset_L)$, $(\Box_R)$ and $(\Diamond_L)$ in $\mathbf{C_{FIK}}$ are modified by keeping the principal formula in the premise(s). For example, the cumulative version of $(\supset_L)$ is*

$$\frac{G\{\Gamma, A \supset B \Rightarrow A, \Delta\} \qquad G\{\Gamma, A \supset B, B \Rightarrow \Delta\}}{G\{\Gamma, A \supset B \Rightarrow \Delta\}}\,(\supset_L)$$

and the cumulative versions of $(\wedge_L)$ and $(\Box_R)$ are

$$\frac{G\{A, B, A \wedge B, \Gamma \Rightarrow \Delta\}}{G\{A \wedge B, \Gamma \Rightarrow \Delta\}}\,(\wedge_L) \qquad \frac{G\{\Gamma \Rightarrow \Delta, \Box A, \langle \Rightarrow [\Rightarrow A]\rangle\}}{G\{\Gamma \Rightarrow \Delta, \Box A\}}\,(\Box_R)$$

The following proposition is a consequence of the admissibility of weakening and contraction of $\mathbf{C_{FIK}}$ which can be done by a standard proof.

▶ **Proposition 32.** *A sequent S is provable in $\mathbf{C}_{FIK}$ if and only if S is provable in $\mathbf{CC_{FIK}}$.*

From now on we only consider $\mathbf{CC_{FIK}}$. We introduce the notion of *structural inclusion* between sequents. It is used in the definition of saturation conditions as well as the model construction presented at the end of the section.

▶ **Definition 33** (Structural inclusion $\subseteq^{\mathbf{S}}$)**.** *Let $\Gamma_1 \Rightarrow \Delta_1, \Gamma_2 \Rightarrow \Delta_2$ be two sequents. $\Gamma_1 \Rightarrow \Delta_1$ is said to be structurally included in $\Gamma_2 \Rightarrow \Delta_2$, denoted as $\Gamma_1 \Rightarrow \Delta_1 \subseteq^{\mathbf{S}} \Gamma_2 \Rightarrow \Delta_2$, if:*

- $\Gamma_1 \subseteq \Gamma_2$ *and*
- *for each $[\Lambda_1 \Rightarrow \Theta_1] \in \Delta_1$, there exists $[\Lambda_2 \Rightarrow \Theta_2] \in \Delta_2$ such that $\Lambda_1 \Rightarrow \Theta_1 \subseteq^{\mathbf{S}} \Lambda_2 \Rightarrow \Theta_2$.*

It is easy to see that $\subseteq^{\mathbf{S}}$ is reflexive and transitive; moreover if $\Gamma_1 \Rightarrow \Delta_1 \subseteq^{\mathbf{S}} \Gamma_2 \Rightarrow \Delta_2$, then $\Gamma_1 \subseteq \Gamma_2$.

We define now the saturation conditions associated to each rule of $\mathbf{CC_{FIK}}$.

▶ **Definition 34** (Saturation conditions)**.** *Let $\Gamma \Rightarrow \Delta$ be a sequent where Γ is a set of formulas and Δ is a set of formulas and blocks. Saturation conditions associated to a rule in the calculus are given as below.*

($\bot_L$) $\bot \notin \Gamma$.
($\top_R$) $\top \notin \Delta$.
(id) $\mathsf{At} \cap (\Gamma \cap \Delta)$ *is empty.*
($\wedge_R$) If $A \wedge B \in \Delta$, then $A \in \Delta$ or $B \in \Delta$.
($\wedge_L$) If $A \wedge B \in \Gamma$, then $A \in \Gamma$ and $B \in \Gamma$.
($\vee_R$) If $A \vee B \in \Delta$, then $A \in \Delta$ and $B \in \Delta$.
($\vee_L$) If $A \vee B \in \Gamma$, then $A \in \Gamma$ or $B \in \Gamma$.
($\supset_R$) If $A \supset B \in \Delta$, then either $A \in \Gamma$ and $B \in \Delta$, or there is $\langle \Sigma \Rightarrow \Pi \rangle \in \Delta$ with $A \in \Sigma$ and $B \in \Pi$.
($\supset_L$) If $A \supset B \in \Gamma$, then $A \in \Delta$ or $B \in \Gamma$.
($\Box_R$) If $\Box A \in \Delta$, then either there is $[\Lambda \Rightarrow \Theta] \in \Delta$ with $A \in \Theta$, or there is $\langle \Sigma \Rightarrow [\Lambda \Rightarrow \Theta], \Pi \rangle \in \Delta$ with $A \in \Theta$.
($\Box_L$) If $\Box A \in \Gamma$ and $[\Sigma \Rightarrow \Pi] \in \Delta$, then $A \in \Sigma$.
($\Diamond_R$) If $\Diamond A \in \Delta$ and $[\Sigma \Rightarrow \Pi] \in \Delta$, then $A \in \Pi$.
($\Diamond_L$) If $\Diamond A \in \Gamma$, then there is $[\Sigma \Rightarrow \Pi] \in \Delta$ with $A \in \Sigma$.
(trans) *If Δ is of form $\Delta', \langle \Sigma \Rightarrow \Pi \rangle$, then $\Gamma \subseteq \Sigma$.*
(inter) *If Δ is of form $\Delta', \langle \Sigma \Rightarrow \Pi \rangle, [\Lambda \Rightarrow \Theta]$, then there is $[\Phi \Rightarrow \Psi] \in \Pi$ with $\Lambda \Rightarrow \Theta \subseteq^{\mathbf{S}} \Phi \Rightarrow \Psi$.*

Concerning the (inter)-saturation, observe that in the (inter) rule we have $\Lambda \Rightarrow \Theta \subseteq^{\mathbf{S}} \Lambda \Rightarrow \Theta^*$, thus the saturation condition actually generalizes the situation.

▶ **Proposition 35.** *Let $\Gamma \Rightarrow \Delta$ be a sequent saturated with respect to both (trans) and (inter). If $\langle \Sigma \Rightarrow \Pi \rangle \in \Delta$, then $\Gamma \Rightarrow \Delta \subseteq^{\mathbf{S}} \Sigma \Rightarrow \Pi$.*

In order to define a terminating proof-search procedure based on $\mathbf{CC_{FIK}}$ (like for any calculus with cumulative rules), as usual we say that the backward application of a rule (R) to a sequent S is *redundant* if S satisfies the corresponding saturation condition for that application of (R) and we impose the following constraints:

(i) *No rule is applied to an axiom* and
(ii) *No rule is applied redundantly.*

However, the restrictions above are not sufficient to ensure the termination of the procedure.

▶ **Example 36.** Let us consider the sequent $S = \Box a \supset \bot, \Box b \supset \bot \Rightarrow p$, where we abbreviate by Γ the antecedent of S. Consider the following derivation, we only show the leftmost branch (the others succeed), we collapse some steps:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{(3)\ \Gamma \Rightarrow p, \Box a, \Box b, \langle \Gamma \Rightarrow \Box a, \Box b, [\Rightarrow a], \langle \Gamma \Rightarrow \Box a, \Box b, [\Rightarrow b]\rangle\rangle, \langle \Gamma \Rightarrow \Box a, \Box b, [\Rightarrow b]\rangle}}{\vdots}}{(2)\ \Gamma \Rightarrow p, \Box a, \Box b, \langle \Gamma \Rightarrow \Box a, \Box b, [\Rightarrow a], \langle \Rightarrow [\Rightarrow b]\rangle\rangle, \langle \Gamma \Rightarrow \Box a, \Box b, [\Rightarrow b]\rangle}\,(\Box_R)}{(1)\ \Gamma \Rightarrow p, \Box a, \Box b, \langle \Gamma \Rightarrow \Box a, \Box b, [\Rightarrow a]\rangle, \langle \Gamma \Rightarrow \Box a, \Box b, [\Rightarrow b]\rangle}\,(\supset_L) \times 4}{\Gamma \Rightarrow p, \Box a, \Box b, \langle \Gamma \Rightarrow [\Rightarrow a]\rangle, \langle \Gamma \Rightarrow [\Rightarrow b]\rangle}\,(trans) \times 2}{\Gamma \Rightarrow p, \Box a, \Box b, \langle \Rightarrow [\Rightarrow a]\rangle, \langle \Rightarrow [\Rightarrow b]\rangle}\,(\Box_R) \times 2}{\Gamma \Rightarrow p, \Box a, \Box b}\,(\supset_L) \times 2}{\Gamma \Rightarrow p}$$

Observe that in the first implication block of sequent (1) ($\Box_R$) can only be applied to $\Box b$, creating the nested block $\langle \Rightarrow [\Rightarrow b]\rangle$ in (2), as it satisfies the saturation condition for $\Box a$. This block will be further expanded to $\langle \Gamma \Rightarrow \Box a, \Box b, [\Rightarrow b]\rangle$ in (3) that satisfies the saturation condition for $\Box b$, but not for $\Box a$, whence it will be further expanded, and so on. Thus the branch does not terminate.

In order to deal with this case of non-termination, intuitively we need to block the expansion of a sequent that occurs nested in another sequent whenever the former has already been expanded and the latter is "equivalent" in some sense to the former. To realize this purpose we first introduce a few notions.

▶ **Definition 37** ($\in^{\langle\cdot\rangle}, \in^{[\cdot]}, \in^+$-relation). *Let $\Gamma_1 \Rightarrow \Delta_1, \Gamma_2 \Rightarrow \Delta_2$ be two sequents. We denote $\Gamma_1 \Rightarrow \Delta_1 \in_0^{\langle\cdot\rangle} \Gamma_2 \Rightarrow \Delta_2$ if $\langle \Gamma_1 \Rightarrow \Delta_1\rangle \in \Delta_2$ and let $\in^{\langle\cdot\rangle}$ be the transitive closure of $\in_0^{\langle\cdot\rangle}$. Relations $\in_0^{[\cdot]}$ and $\in^{[\cdot]}$ for modal blocks are defined similarly. Besides, let $\in_0^+ = \in_0^{\langle\cdot\rangle} \cup \in_0^{[\cdot]}$ and finally let $\in^+$ be the reflexive-transitive closure of $\in_0^+$.*

Observe that when we say $S' \in^+ S$, it is equivalent to say that for some context G, $S = G\{S'\}$.

We introduce the operator $\sharp$ for the succedent of a sequent, it is used to remove implication blocks but retain all the other formulas and modal blocks.

▶ **Definition 38** ($\sharp$-operator). *Let $\Lambda \Rightarrow \Theta$ be a sequent. We define $\Theta^\sharp$ as follows:*

(i) $\Theta^\sharp = \Theta$ *if* Θ *is block-free;*
(ii) $\Theta^\sharp = \Theta_0^\sharp, [\Phi \Rightarrow \Psi^\sharp]$ *if* $\Theta = \Theta_0, [\Phi \Rightarrow \Psi]$*;*
(iii) $\Theta^\sharp = \Theta_0^\sharp$ *if* $\Theta = \Theta_0, \langle \Phi \Rightarrow \Psi\rangle$*.*

We can compare this $\sharp$-operator with $*$ in Definition 20. For example, let $\Delta = b, [c \Rightarrow d, [e \Rightarrow f], \langle g \Rightarrow h\rangle], \langle t \Rightarrow [p \Rightarrow q]\rangle, [m \Rightarrow n]$, then $\Delta^\sharp = b, [c \Rightarrow d, [e \Rightarrow f]], [m \Rightarrow n]$, while $\Delta^* = [c \Rightarrow [e \Rightarrow]], [m \Rightarrow]$.

Intuitively speaking, if a sequent $S = \Lambda \Rightarrow \Theta$ describes a model rooted in S and specifies formulas forced and not forced in S, then $\Lambda \Rightarrow \Theta^\sharp$, describes the chains of R-related worlds to S by specifying all formulas forced and not forced in each one of them, but ignores upper worlds in the pre-order, the latter being represented by implication blocks.

We use the $\sharp$-operator to define an equivalence relation between sequents. The equivalence relation will be used to detect loops in a derivation as in the example above.

▶ **Definition 39** ($\sharp$-equivalence). *Let S_1, S_2 be two sequents where $S_1 = \Gamma_1 \Rightarrow \Delta_1, S_2 = \Gamma_2 \Rightarrow \Delta_2$. We say S_1 is $\sharp$-equivalent to S_2, denoted as $S_1 \simeq S_2$, if $\Gamma_1 = \Gamma_2$ and $\Delta_1^\sharp = \Delta_2^\sharp$.*

In order to define a proof-search procedure, we divide rules of $\mathbf{CC_{FIK}}$ into three groups and define correspondingly three levels of saturation.

(R1) basic rules: all propositional and modal rules except $(\supset_R)$ and $(\Box_R)$;
(R2) rules that transfer formulas and blocks into implication blocks: (trans) and (inter);
(R3) rules that create implication blocks: $(\Box_R)$ and $(\supset_R)$.

▶ **Definition 40** (Saturation). *Let $S = \Gamma \Rightarrow \Delta$ be a sequent and not an axiom. S is called:*

- R1-saturated *if $\Gamma \Rightarrow \Delta^\sharp$ satisfies all the saturation conditions of R1 rules;*
- R2-saturated *if S is R1-saturated and S satisfies saturation conditions of R2 rules for blocks $S_1 \in_0^{\langle\cdot\rangle} S$ and $S_2 \in_0^{[\cdot]} S$.*
- R3-saturated *if S is R2-saturated and S satisfies saturation conditions of R3 rules for formulas $\Box A, B \supset C \in \Delta$.*

We can finally define when a sequent is blocked, the intention is that it will not be expanded anymore by the proof-search procedure.

▶ **Definition 41** (Blocked sequent). *Given a sequent S and $S_1, S_2 \in^+ S$, with $S_1 = \Gamma_1 \Rightarrow \Delta_1, S_2 = \Gamma_2 \Rightarrow \Delta_2$. We say S_2 is blocked by S_1 in S, if S_1 is R3-saturated, $S_2 \in^{\langle\cdot\rangle} S_1$ and $S_1 \simeq S_2$. We say that a sequent S' is* blocked *in S if there exists $S_1 \in^+ S$ such that S' is blocked by S_1 in S.*

Observe that if S is finite, then for any $S' \in^+ S$ checking whether S' is blocked in S can be effectively decided. We will say just that S' is blocked when S is clear.

▶ **Example 42.** We reconsider the example 36. The sequent (3) will be further expanded to

$$\begin{aligned}(4)\ \Gamma \Rightarrow p, \Box a, \Box b,\\ &\langle \Gamma \Rightarrow \Box a, \Box b, [\Rightarrow a], \langle \Gamma \Rightarrow \Box a, \Box b, [\Rightarrow b], \langle \Gamma \Rightarrow \Box a, \Box b, [\Rightarrow a]\rangle^{(ii)}\rangle\rangle^{(i)},\\ &\langle \Gamma \Rightarrow \Box a, \Box b, [\Rightarrow b]\rangle\end{aligned}$$

We have marked by (i) and (ii) the relevant blocks. Observe that the sequent $S_2 = \Gamma \Rightarrow \Box a, \Box b, [\Rightarrow a]$ in the block marked (ii) is blocked by the sequent $S_1 = \Gamma \Rightarrow \Box a, \Box b, [\Rightarrow a], \langle \Gamma \Rightarrow \Box a, \Box b, [\Rightarrow b], \langle \Gamma \Rightarrow \Box a, \Box b, [\Rightarrow a]\rangle\rangle$ marked (i), since S_1 is R3-saturated, $S_2 \in^{\langle\cdot\rangle} S_1$ and $S_1 \simeq S_2$, as in particular $(\Box a, \Box b, [\Rightarrow a], \langle \Gamma \Rightarrow \Box a, \Box b, [\Rightarrow b], \langle \Gamma \Rightarrow \Box a, \Box b, [\Rightarrow a]\rangle\rangle)^\sharp = (\Gamma \Rightarrow \Box a, \Box b, [\Rightarrow a])^\sharp$.

We finally define three global saturation conditions.

▶ **Definition 43** (Global saturation). *Let S be a sequent and not an axiom. S is called :*

- global-R1-saturated *if for each $T \in^+ S$, T is either R1-saturated or blocked;*
- global-R2-saturated *if for each $T \in^+ S$, T is either R2-saturated or blocked;*
- global-saturated *if for each $T \in^+ S$, T is either R3-saturated or blocked.*

In order to specify the proof-search procedure, we make use of three sub-procedures that extend a given derivation $\mathcal{D}$ by expanding a leaf S, each procedure applies rules *non-redundantly* to some $T := \Gamma \Rightarrow \Delta \in^+ S$, that we recall it means that $S = G\{T\}$, for some context G . We define :

1. $\mathbf{EXP1}(\mathcal{D}, S, T) = \mathcal{D}'$ where $\mathcal{D}'$ is the extension of $\mathcal{D}$ obtained by applying R1 rules to every formula in $\Gamma \Rightarrow \Delta^\sharp$.
2. $\mathbf{EXP2}(\mathcal{D}, S, T) = \mathcal{D}'$ where $\mathcal{D}'$ is the extension of $\mathcal{D}$ obtained by applying R2-rules to blocks $\langle T_i\rangle, [T_j] \in \Delta$.
3. $\mathbf{EXP3}(\mathcal{D}, S, T) = \mathcal{D}'$ where $\mathcal{D}'$ is the extension of $\mathcal{D}$ obtained by applying R3-rules to formulas $\Box A, A \supset B \in \Delta$.

The three procedures are used as macro-steps in the proof search procedure defined next. We are going to prove that the three sub-procedures terminate, this is stated in Proposition 46 below. The claim is obvious for $\mathbf{EXP2}(\mathcal{D}, S, T)$, $\mathbf{EXP3}(\mathcal{D}, S, T)$ as only finitely many blocks or formulas in T are processed. For $\mathbf{EXP1}(\mathcal{D}, S, T)$, the claim is not so trivial, since the rules are applied also deeply within $\Gamma \Rightarrow \Delta^\sharp$. But notice that **EXP1** only applies the rules (both left and right) for $\wedge, \vee, \Diamond$ and $\supset_L, \Box_L$ while ignores implication blocks, we can see $\mathbf{EXP1}(\mathcal{D}, S, T)$ produces exactly the same expansion of $\mathcal{D}$ that we would obtain by applying the same rules of a nested sequent calculus for classical modal logic **K** and we know that that procedure terminates (see [3], Lemma 7).

In order to give a proof of the claim for $\mathbf{EXP1}(\mathcal{D}, S, T)$ precisely we introduce the following definition.

▶ **Definition 44.** *Given a sequent S, the tree $\mathcal{T}_S$ is defined as follows: (i) the root of $\mathcal{T}_S$ is S; (ii) if $S_1 \in_0^{[\cdot]} S_2$, then S_1 is a child of S_2.*

We denote the height of $\mathcal{T}_S$ as $h(\mathcal{T}_S)$. It is easy to verify that $h(\mathcal{T}_S) \leq md(S)$. Moreover we denote by $Sub(A)$ the set of subformulas of a formula A and for a sequent $S = \Gamma \Rightarrow \Delta$ we use the corresponding notations $Sub(\Gamma)$, $Sub(\Delta)$, $Sub(S)$. Finally, we recall that $Card(Sub(S)) = O(|S|)$.

By estimating the size of the tree associated to a sequent, we can get the following *rough* bound of the size of any sequent occurring in a derivation by R1-rules.

▶ **Proposition 45.** *Let $\mathcal{D}o$ be a derivation with root a non-axiomatic sequent $T = \Gamma \Rightarrow \Delta$ obtained by applying R1-rules to $\Gamma \Rightarrow \Delta^\sharp$, then any T' occurring in $\mathcal{D}o$ has size $O(|T|^{|T|+1})$.*

We can now prove the following proposition.

▶ **Proposition 46.** *Given a finite derivation $\mathcal{D}$, a finite leaf S of $\mathcal{D}$ and $T \in^+ S$, then each $\boldsymbol{EXP1}(\mathcal{D}, S, T)$, $\boldsymbol{EXP2}(\mathcal{D}, S, T)$,$\boldsymbol{EXP3}(\mathcal{D}, S, T)$ terminates by producing a finite expansion of $\mathcal{D}$ where all sequents in it are finite.*

We present below the proof-search procedure PROCEDURE(A), that given an input formula A it returns either a proof of A or a finite derivation tree in which all non-axiomatic leaves are global-saturated.

Note that the proof-search algorithm we give is breadth-first, as we can see in line 8, we expand all such non-axiomatic leaves in parallel. As a result, in line 5, the output is a fully-saturated derivation, which means each non-axiomatic leaf in it is global-saturated. Actually it is also possible to redesign the algorithm in a depth-first way by working with one leaf exhaustively at each time and then the procedure for a unprovable formulas terminates once the first global-saturated leaf is constructed.

An important property of the proof-search procedure is that saturation and blocking are preserved through sequent expansion, in other words, they are *invariant* of the repeated loop of the procedure.

▶ **Lemma 47** (Invariant). *Let S be a leaf of a derivation $\mathcal{D}$ with root $\Rightarrow A$:*

1. *Let $T \in^+ S$, where $T = \Gamma \Rightarrow \Delta$, for every rule (R) if T satisfies the R-saturation condition on some formulas A_i and/or blocks $\langle T_j \rangle, [T_k]$* before *the execution of (the body of) the repeat loop (lines 3-14), then T satisfies the R-condition on the involved $A_i, \langle T_j \rangle, [T_k]$* after *the execution of it.*
2. *Let $T \in^+ S$, if T is blocked in S* before *the execution of (the body of) the repeat loop, then it is still so* after *it.*

Algorithm 1 PROCEDURE(A).

Input: $\mathcal{D}_0 = \Rightarrow A$
initialization $\mathcal{D} = \mathcal{D}_0$;
repeat
  if *all the leaves of* $\mathcal{D}$ *are axiomatic* **then**
    return "PROVABLE" and $\mathcal{D}$
  else if *all the non-axiomatic leaves of* $\mathcal{D}$ *are global-saturated* **then**
    return "UNPROVABLE" and $\mathcal{D}$
  else
    for all non-axiomatic leaves S of $\mathcal{D}$ that are not global-saturated
    if *S is global-R2-saturated* **then**
      for all $T \in^+ S$ such that T is a $\in^{\langle\cdot\rangle}$-minimal and not R3-saturated, check whether T is blocked in S, if not, let $\mathcal{D} = \mathbf{EXP3}(\mathcal{D}, S, T)$
    else if *S is global-R1-saturated* **then**
      for all $T \in^+ S$ such that T is not R2-saturated, let $\mathcal{D} = \mathbf{EXP2}(\mathcal{D}, S, T)$
    else
      for all $T \in^+ S$ such that T is not R1-saturated, let $\mathcal{D} = \mathbf{EXP1}(\mathcal{D}, S, T)$
until *FALSE*;

The last ingredient in order to prove termination is that in a derivation of a formula, there can only be finitely many non-blocked sequents.

▶ **Lemma 48.** *Given a formula A, let $\boldsymbol{Seq}(A)$ be the set of sequents that may occur in any possible derivation with root $\Rightarrow A$. Let $\boldsymbol{Seq}(A)/_{\simeq}$ be the quotient of $\boldsymbol{Seq}(A)$ with respect to $\sharp$-equivalence $\simeq$ as defined in Definition 39. Then $\boldsymbol{Seq}(A)/_{\simeq}$ is finite.*

Intuitively, the termination of the procedure is based on the fact that the procedure cannot run forever by building an infinite derivation. The reason is that the built derivation cannot contain any infinite branch, because (i) once that a sequent satisfies a saturation condition for a rule (R), further expansions of it will still satisfy that condition (whence will not be reconsidered for the application of (R)); (ii) if a sequent is blocked, further application or rules cannot "unblock" it; (iii) the number of non-equivalent, whence unblocked sequents is finite.

▶ **Theorem 49** (Termination). *Let A be a formula. Proof-search for the sequent $\Rightarrow A$ terminates with a finite derivation in which any leaf is either an axiom or global-saturated.*

Next, we prove the completeness of $\mathbf{CC_{FIK}}$. Given a finite global-saturated leaf S of the derivation $\mathcal{D}$ produced by PROCEDURE(A), we can define a model $\mathcal{M}_S$ as follows, which will be shown as a countermodel for A.

▶ **Definition 50.** *The model $\mathcal{M}_S = (W_S, \leq_S, R_S, V_S)$ determined by S is defined as follows:*

- $W_S = \{x_{\Phi \Rightarrow \Psi} \mid \Phi \Rightarrow \Psi \in^+ S\}$.
- $x_{S_1} \leq_S x_{S_2}$ *if* $S_1 \subseteq^{\mathbf{S}} S_2$.
- $R_S x_{S_1} x_{S_2}$ *if* $S_2 \in_0^{[\cdot]} S_1$.
- *for each* $x_{\Phi \Rightarrow \Psi} \in W_S$, *let* $V_S(x_{\Phi \Rightarrow \Psi}) = \{p \mid p \in \Phi\}$.

We give some brief remarks on the model construction. Obviously $\mathcal{M}_S$ is finite, and each world in W_S corresponds to either a R3-saturated or a blocked sequent, that is nonetheless saturated with respect to (inter) and (trans). Moreover, by Proposition 35, we can see if $x_{\Gamma\Rightarrow\Delta',\langle\Sigma\Rightarrow\Pi\rangle} \in W_S$ then $x_{\Sigma\Rightarrow\Pi} \in W_S$, and $x_{\Gamma\Rightarrow\Delta',\langle\Sigma\Rightarrow\Pi\rangle} \leq_S x_{\Sigma\Rightarrow\Pi}$. Lastly, by the property of structural inclusion $\subseteq^{\mathbf{S}}$, we have that $\leq_S$ is a pre-order.

▶ **Proposition 51.** *$\mathcal{M}_S$ satisfies the hereditary property (HP) and forward confluence (FC).*

▶ **Lemma 52** (Truth Lemma)**.** *Let S be a global-saturated sequent and $\mathcal{M}_S$ be defined as above. (a). If $A \in \Phi$, then $\mathcal{M}_S, x_{\Phi\Rightarrow\Psi} \Vdash A$; (b). If $A \in \Psi$, then $M_S, x_{\Phi\Rightarrow\Psi} \nVdash A$.*

From the truth lemma we immediately obtain the completeness of $\mathbf{CC_{FIK}}$.

▶ **Theorem 53.** *For any formula $A \in \mathcal{L}$, if $\Vdash A$, then $\Rightarrow A$ is provable in $\mathbf{CC_{FIK}}$.*

▶ **Example 54.** We show how to build a countermodel of the formula $(\Diamond p \supset \Box q) \supset \Box(p \supset q)$ by $\mathbf{CC_{FIK}}$ (due to space limit, we do not present the full derivation here). By backward application of the rules, one branch of the derivation ends up with the the saturated sequent S_0:

$$S_0 = \ \Diamond p \supset \Box q \Rightarrow \Diamond p, \Box(p \supset q), \langle \Diamond p \supset \Box q \Rightarrow \Diamond p, [\Rightarrow p \supset q, \langle p \Rightarrow q\rangle, p]\rangle \text{ and let}$$

$$S_1 = \Diamond p \supset \Box q \Rightarrow \Diamond p, [\Rightarrow p \supset q, \langle p \Rightarrow q\rangle, p], \ \ S_2 = \Rightarrow p \supset q, \langle p \Rightarrow q\rangle, p, \ \ S_3 = p \Rightarrow q$$

We then get the model $M_{S_0} = (W, \leq, R, V)$ where
- $W = \{x_{S_0}, x_{S_1}, x_{S_2}, x_{S_3}\}$,
- $x_{S_0} \leq x_{S_1}$, $x_{S_2} \leq x_{S_0}$, $x_{S_2} \leq x_{S_3}$,
- $Rx_{S_1}x_{S_2}$,
- $V(x_{S_0}) = V(x_{S_1}) = V(x_{S_2}) = \emptyset$ and $V(x_{S_3}) = \{p\}$.

It is easy to see that $x_{S_0} \nVdash (\Diamond p \supset \Box q) \supset \Box(p \supset q)$.

▶ **Example 55.** Consider another example $\neg\neg\Box\neg p \supset \Box\neg p$ which shows that the $\Diamond$-free fragment of **FIK** is weaker than the same fragment of **IK**. The formula is presented in [4] and is provable in **IK**. By building a derivation with the root $\Rightarrow ((\Box(p \supset \bot) \supset \bot) \supset \bot) \supset \Box(p \supset \bot)$, we generate a saturated sequent

$$S_0 = F_1 \Rightarrow \Box(p \supset \bot), F_2, \langle S_1\rangle, \langle S_6\rangle,$$

where $F_1 = (\Box(p \supset \bot) \supset \bot) \supset \bot, F_2 = \Box(p \supset \bot) \supset \bot$, and

$$S_1 = F_1 \Rightarrow F_2, [\Rightarrow \langle p \Rightarrow \bot\rangle], \langle S_4\rangle, \quad S_4 = F_1, \Box(p \supset \bot) \Rightarrow \bot, F_2, [p \supset \bot \Rightarrow p],$$
$$S_6 = F_1, \Box(p \supset \bot) \Rightarrow \bot, F_2,$$
$$S_2 = \Rightarrow \langle p \Rightarrow \bot\rangle, \quad S_3 = p \Rightarrow \bot, \quad S_5 = p \supset \bot \Rightarrow p.$$

We then get the model $M_{S_0} = (W, \leq, R, V)$ where
- $W = \{x_{S_0}, \ldots, x_{S_6}\}$,
- $x_{S_0} \leq x_{S_1}, x_{S_0} \leq x_{S_6}, x_{S_1} \leq x_{S_4}, x_{S_6} \leq x_{S_4}$, $x_{S_2} \leq x_{S_3}$, $x_{S_2} \leq x_{S_5}$, $x_{S_2} \leq x_{S_0}$,
- $Rx_{S_1}x_{S_2}$, $Rx_{S_4}x_{S_5}$,
- $V(x_{S_i}) = \emptyset$ if $i \neq 3$ and $V(x_{S_3}) = \{p\}$.

It is easy to see that $x_{S_0} \nVdash ((\Box(p \supset \bot) \supset \bot) \supset \bot) \supset \Box(p \supset \bot)$.

5 Conclusion and future work

We have proposed **FIK**, a natural variant of Intuitionistic modal logic characterized by forward confluent bi-relational models. **FIK** is intermediate between constructive modal logic **CK** and intuitionistic modal logic **IK** and it satisfies all the expected criteria for **IML**. We have presented a sound and complete axiomatization of it and a bi-nested calculus $\mathbf{C_{FIK}}$ which provides a decision procedure together with a finite countermodel extraction.

There are many topics for further research. First we may study extensions of **FIK** with the standard axioms from the modal cube. To obtain decidability and terminating proof systems for transitive logics (e.g. the **4**-extension) might be difficult and it may be worthwhile to study an embedding of our nested sequent calculus into a labelled calculus and then adapt the techniques and results in [7]. More generally, we can also explore extensions of **FIK** whose accessibility relation is defined by Horn properties and the nested sequent calculi might be obtained by means of the refinement technique proposed in [8]. Lastly we can consider other bi-relational frame conditions relating to the pre-order and the accessible (including the one for **IK**) and see how they can be captured uniformly in bi-nested calculi with suitable "interactive rules".

References

1 Philippe Balbiani, Martín Diéguez, and David Fernández-Duque. Some constructive variants of S4 with the finite model property. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS*, pages 1–13. IEEE, 2021. doi:10.1109/LICS52264.2021.9470643.

2 Gianluigi Bellin, Valeria De Paiva, and Eike Ritter. Extended curry-howard correspondence for a basic constructive modal logic. In *Proceedings of methods for modalities*, volume 2, 2001.

3 Kai Brünnler. Deep sequent systems for modal logic. *Arch. Math. Log.*, 48(6):551–577, 2009. doi:10.1007/S00153-009-0137-3.

4 Anupam Das and Sonia Marin. On intuitionistic diamonds (and lack thereof). In Revantha Ramanayake and Josef Urban, editors, *Automated Reasoning with Analytic Tableaux and Related Methods – 32nd International Conference, TABLEAUX 2023*, volume 14278 of *Lecture Notes in Computer Science*, pages 283–301. Springer, 2023. doi:10.1007/978-3-031-43513-3_16.

5 Frederic B. Fitch. Intuitionistic modal logic with quantifiers. *Portugaliae mathematica*, 7(2):113–118, 1948. URL: http://eudml.org/doc/114664.

6 Melvin Fitting. Nested sequents for intuitionistic logics. *Notre Dame J. Formal Log.*, 55(1):41–61, 2014. doi:10.1215/00294527-2377869.

7 Marianna Girlando, Roman Kuznets, Sonia Marin, Marianela Morales, and Lutz Straßburger. Intuitionistic S4 is decidable. In *2023 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–13, 2023. doi:10.1109/LICS56636.2023.10175684.

8 Tim S. Lyon. Nested sequents for intuitionistic modal logics via structural refinement. In Anupam Das and Sara Negri, editors, *Automated Reasoning with Analytic Tableaux and Related Methods – 30th International Conference, TABLEAUX 2021*, volume 12842 of *Lecture Notes in Computer Science*, pages 409–427. Springer, 2021. doi:10.1007/978-3-030-86059-2_24.

9 Sonia Marin and Marianela Morales. Fully structured proof theory for intuitionistic modal logics. In *AiML 2020 – Advances in Modal Logic*, 2020. URL: https://hal.science/hal-03048959.

10 Sonia Marin, Marianela Morales, and Lutz Straßburger. A fully labelled proof system for intuitionistic modal logics. *J. Log. Comput.*, 31(3):998–1022, 2021. doi:10.1093/LOGCOM/EXAB020.

11 Gordon Plotkin and Colin Stirling. A framework for intuitionistic modal logics. In *Proceedings of the 1st Conference on Theoretical Aspects of Reasoning about Knowledge (TARK)*, pages 399–406, 1986. doi:10.5555/1029786.1029823.

12 Dag Prawitz. *Natural Deduction: A Proof-Theoretical Study*. Dover Publications, Mineola, N.Y., 1965. doi:10.2307/2271676.

13 Gisèle Fischer Servi. On modal logic with an intuitionistic base. *Studia Logica*, 36(3):141–149, 1977. doi:10.1007/BF02121259.

14 Gisèle Fischer Servi. *Semantics for a Class of Intuitionistic Modal Calculi*, pages 59–72. Springer Netherlands, Dordrecht, 1981. doi:10.1007/978-94-009-8937-5_5.

15 Gisèle Fischer Servi. Axiomatizations for some intuitionistic modal logics. *Rendiconti del Seminario Matematico Università e Politecnico di Torino*, 42, 1984. URL: https://cir.nii.ac.jp/crid/1371132818982119684.

16 Alex K. Simpson. *The proof theory and semantics of intuitionistic modal logic*. Ph.D. Thesis, University of Edinburgh, 1994. URL: https://api.semanticscholar.org/CorpusID:2309858.

17 Duminda Wijesekera. Constructive modal logics I. *Ann. Pure Appl. Log.*, 50(3):271–301, 1990. doi:10.1016/0168-0072(90)90059-B.

A Appendix

The appendix includes the proofs of some of our results.

Proof of Lemma 12. Let $\Gamma, \Delta, \Lambda \in W_c$ be such that $\Gamma \geq_c \Delta$ and $\Delta R_c \Lambda$. Hence, $\Gamma \supseteq \Delta$ and $\Delta \bowtie \Lambda$. Let $A_1, A_2, \ldots$ be an enumeration of $\Box\Gamma$ and $B_1, B_2, \ldots$ be an enumeration of Λ. Obviously, for all $n \in \mathbb{N}$, $\Box(A_1 \wedge \ldots \wedge A_n) \in \Gamma$ and $B_1 \wedge \ldots \wedge B_n \in \Lambda$. Since $\Delta \bowtie \Lambda$, for all $n \in \mathbb{N}$, $\Diamond(B_1 \wedge \ldots \wedge B_n) \in \Delta$. For all $n \in \mathbb{N}$, let $\Theta_n = \mathbf{D_{FIK}} + A_1 \wedge \ldots \wedge A_n \wedge B_1 \wedge \ldots \wedge B_n$. Obviously, $(\Theta_n)_{n\in\mathbb{N}}$ is a chain of theories such that $\bigcup\{\Theta_n : n \in \mathbb{N}\} \supseteq \Lambda$.

We claim that for all formulas C, if $\Box C \in \Gamma$ then $C \in \bigcup\{\Theta_n : n \in \mathbb{N}\}$. If not, there exists a formula C such that $\Box C \in \Gamma$ and $C \notin \bigcup\{\Theta_n : n \in \mathbb{N}\}$. Thus, $C \in \Box\Gamma$. Consequently, let $n \in \mathbb{N}$ be such that $A_n = C$. Hence, $A_1 \wedge \ldots \wedge A_n \wedge B_1 \wedge \ldots \wedge B_n \to C$ is in $\mathbf{D_{FIK}}$. Thus, $C \in \Theta_n$. Consequently, $C \in \bigcup\{\Theta_n : n \in \mathbb{N}\}$: a contradiction. Hence, for all formulas C, if $\Box C \in \Gamma$ then $C \in \bigcup\{\Theta_n : n \in \mathbb{N}\}$.

We claim that for all formulas C, if $C \in \bigcup\{\Theta_n : n \in \mathbb{N}\}$ then $\Diamond C \in \Gamma$. If not, there exists $n \in \mathbb{N}$ and there exists a formula C such that $C \in \Theta_n$ and $\Diamond C \notin \Gamma$. Thus, $A_1 \wedge \ldots \wedge A_n \wedge B_1 \wedge \ldots \wedge B_n \to C$ is in $\mathbf{D_{FIK}}$. Consequently, $B_1 \wedge \ldots \wedge B_n \to (A_1 \wedge \ldots \wedge A_n \to C)$ is in $\mathbf{D_{FIK}}$. Hence, $\Diamond(B_1 \wedge \ldots \wedge B_n) \supset \Diamond(A_1 \wedge \ldots \wedge A_n \supset C)$ is in $\mathbf{D_{FIK}}$. Since $\Diamond(B_1 \wedge \ldots \wedge B_n) \in \Delta$, $\Diamond(A_1 \wedge \ldots \wedge A_n \supset C) \in \Delta$. Since $\Gamma \supseteq \Delta$, $\Diamond(A_1 \wedge \ldots \wedge A_n \supset C) \in \Gamma$. Thus, $\Box(A_1 \wedge \ldots \wedge A_n) \supset \Diamond C \in \Gamma$. Since $\Box(A_1 \wedge \ldots \wedge A_n) \in \Gamma$, $\Diamond C \in \Gamma$: a contradiction. Consequently, for all formulas C, if $C \in \bigcup\{\Theta_n : n \in \mathbb{N}\}$ then $\Diamond C \in \Gamma$.

Let $\mathcal{S} = \{\Theta : \Theta \text{ is a theory such that } \mathbf{(1)}\ \Gamma \bowtie \Theta \text{ and } \mathbf{(2)}\ \Theta \supseteq \Lambda\}$. Obviously, $\bigcup\{\Theta_n : n \in \mathbb{N}\} \in \mathcal{S}$. Hence, $\mathcal{S}$ is nonempty. Moreover, for all nonempty chains $(\Pi_i)_{i\in I}$ of elements of $\mathcal{S}$, $\bigcup\{\Pi_i : i \in I\}$ is an element of $\mathcal{S}$. Thus, by Zorn's Lemma, $\mathcal{S}$ possesses a maximal element Θ. Consequently, Θ is a theory such that $\Gamma \bowtie \Theta$ and $\Theta \supseteq \Lambda$. Hence, it only remains to be proved that Θ is proper and prime.

We claim that Θ is proper. If not, $\bot \in \Theta$. Since $\Gamma \bowtie \Theta$, $\Diamond\bot \in \Gamma$: a contradiction. Thus, Θ is proper.

We claim that Θ is prime. If not, there exists formulas C, D such that $C \vee D \in \Theta$, $C \notin \Theta$ and $D \notin \Theta$. Consequently, by the maximality of Θ in $\mathcal{S}$, $\Theta + C \notin \mathcal{S}$ and $\Theta + D \notin \mathcal{S}$. Hence, there exists a formula E such that $E \in \Theta + C$ and $\Diamond E \notin \Gamma$ and there exists a formula F such that $F \in \Theta + D$ and $\Diamond F \notin \Gamma$. Thus, $C \supset E \in \Theta$ and $D \supset F \in \Theta$. Consequently, $C \vee D \supset E \vee F \in \Theta$. Since $C \vee D \in \Theta$, $E \vee F \in \Theta$. Since $\Gamma \bowtie \Theta$, $\Diamond(E \vee F) \in \Gamma$. Hence, either $\Diamond E \in \Gamma$, or $\Diamond F \in \Gamma$: a contradiction. Thus, Θ is prime. ◀

Proof of Lemma 13. We only show the case of $\Box$ here. Suppose $\Box B \notin \Gamma$. Let $\mathcal{S} = \{\Delta : \Delta$ is a theory such that **(1)** $\Gamma \subseteq \Delta$ and **(2)** $\Box B \notin \Delta\}$.

Since $\Box B \notin \Gamma$, $\Gamma \in \mathcal{S}$. Hence, $\mathcal{S}$ is nonempty. Moreover, for all nonempty chains $(\Delta_i)_{i\in I}$ of elements of $\mathcal{S}$, $\bigcup\{\Delta_i : i \in I\}$ is an element of $\mathcal{S}$. Thus, by Zorn's Lemma, $\mathcal{S}$ possesses a maximal element Δ. Consequently, Δ is a theory such that $\Gamma \subseteq \Delta$ and $\Box B \notin \Delta$.

We claim that Δ is proper. If not, then $\Delta = \mathcal{L}$. Hence, $\Box B \in \Delta$: a contradiction. Thus, Δ is proper.

We claim that Δ is prime. If not, there exists formulas C, D such that $C \vee D \in \Delta$, $C \notin \Delta$ and $D \notin \Delta$. Consequently, by the maximality of Δ in $\mathcal{S}$, $\Delta + C \notin \mathcal{S}$ and $\Delta + D \notin \mathcal{S}$. Hence, $\Box B \in \Delta + C$ and $\Box B \in \Delta + D$. Thus, $C \supset \Box B \in \Delta$ and $D \supset \Box B \in \Delta$. Consequently, $C \vee D \supset \Box B \in \Delta$. Since $C \vee D \in \Delta$, $\Box B \in \Delta$: a contradiction. Hence, Δ is prime.

We claim that for all formulas C, if $C \vee B \in \Box\Delta$ then $\Diamond C \in \Delta$. If not, there exists a formula C such that $C \vee B \in \Box\Delta$ and $\Diamond C \notin \Delta$. Thus, by the maximality of Δ in $\mathcal{S}$, $\Delta + \Diamond C \notin \mathcal{S}$. Consequently, $\Box B \in \Delta + \Diamond C$. Hence, $\Diamond C \supset \Box B \in \Delta$. Since $C \vee B \in \Box\Delta$, $\Box(C \vee B) \in \Delta$. Since $\Diamond C \supset \Box B \in \Delta$, $\Box B \in \Delta$: a contradiction. Thus, for all formulas C, if $C \vee B \in \Box\Delta$ then $\Diamond C \in \Delta$.

Let $\mathcal{T} = \{\Lambda : \Lambda$ is a theory such that **(1)** $\Box\Delta \subseteq \Lambda$, **(2)** for all formulas C, if $C \vee B \in \Lambda$ then $\Diamond C \in \Delta$ and **(3)** $B \notin \Lambda\}$. Since $\Box B \notin \Delta$, $B \notin \Box\Delta$. Consequently, $\Box\Delta \in \mathcal{T}$. Hence, $\mathcal{T}$ is nonempty. Moreover, for all nonempty chains $(\Lambda_i)_{i\in I}$ of elements of $\mathcal{T}$, $\bigcup\{\Lambda_i : i \in I\}$ is an element of $\mathcal{T}$. Thus, by Zorn's Lemma, $\mathcal{T}$ possesses a maximal element Λ. Consequently, Λ is a theory such that $\Box\Delta \subseteq \Lambda$, for all formulas C, if $C \vee B \in \Lambda$ then $\Diamond C \in \Delta$ and $B \notin \Lambda$. Hence, it only remains to be proved that Λ is proper and prime and $\Delta \bowtie \Lambda$.

We claim that Λ is proper. If not, $\Lambda = \mathcal{L}$. Thus, $B \in \Lambda$: a contradiction. Consequently, Λ is proper.

We claim that Λ is prime. If not, there exists formulas C, D such that $C \vee D \in \Lambda$, $C \notin \Lambda$ and $D \notin \Lambda$. Hence, by the maximality of Λ in $\mathcal{T}$, $\Lambda + C \notin \mathcal{T}$ and $\Lambda + D \notin \mathcal{T}$. Thus, either there exists a formula E such that $E \vee B \in \Lambda + C$ and $\Diamond E \notin \Delta$, or $B \in \Lambda + C$ and either there exists a formula F such that $F \vee E \in \Lambda + D$ and $\Diamond F \notin \Delta$, or $B \in \Lambda + D$. Consequently, we have to consider the following four cases.

(1) Case "there exists a formula E such that $E \vee B \in \Lambda + C$ and $\Diamond E \notin \Delta$ and there exists a formula F such that $F \vee B \in \Lambda + D$ and $\Diamond F \notin \Delta$": Hence, $C \supset E \vee B \in \Lambda$ and $D \supset F \vee B \in \Lambda$. Thus, $C \vee D \supset E \vee F \vee B \in \Lambda$. Since $C \vee D \in \Lambda$, $E \vee F \vee B \in \Lambda$. Consequently, $\Diamond(E \vee F) \in \Delta$. Hence, either $\Diamond E \in \Delta$, or $\Diamond F \in \Delta$: a contradiction.

(2) Case "there exists a formula E such that $E \vee F \in \Lambda + C$ and $\Diamond E \notin \Delta$ and $B \in \Lambda + D$": Thus, $C \supset E \vee B \in \Lambda$ and $D \supset B \in \Lambda$. Consequently, $C \vee D \supset E \vee B \in \Lambda$. Since $C \vee D \in \Lambda$, $E \vee B \in \Lambda$. Hence, $\Diamond E \in \Delta$: a contradiction.

(3) Case "$B \in \Lambda + C$ and there exists a formula F such that $F \vee B \in \Lambda + D$ and $\Diamond F \notin \Delta$": Thus, $C \supset B \in \Lambda$ and $D \supset F \vee B \in \Lambda$. Consequently, $C \vee D \supset F \vee B \in \Lambda$. Since $C \vee D \in \Lambda$, $F \vee B \in \Lambda$. Hence, $\Diamond F \in \Delta$: a contradiction.

(4) Case "$B \in \Lambda + C$ and $B \in \Lambda + D$": Thus, $C \supset B \in \Lambda$ and $D \supset B \in \Lambda$. Consequently, $C \vee D \supset B \in \Lambda$. Since $C \vee D \in \Lambda$, $B \in \Lambda$: a contradiction.

Hence, Λ is prime.

Lastly, we claim that $\Delta \bowtie \Lambda$. If not, there exists a formula C such that $C \in \Lambda$ and $\Diamond C \notin \Delta$. Thus, $C \vee B \in \Lambda$. Consequently $\Diamond C \in \Delta$: a contradiction. Hence, $\Delta \bowtie \Lambda$. ◀

Proof of Lemma 28. By induction on the structure of Δ^*. If $\Delta^* = \emptyset$ it follows by definition. Otherwise $\Delta^* = [\Phi_1 \Rightarrow \Psi_1^*], \ldots, [\Phi_k \Rightarrow \Psi_k^*]$ where $\Delta = \Delta_0, [\Phi_1 \Rightarrow \Psi_1], \ldots, [\Phi_k \Rightarrow \Psi_k]$ and Δ_0 is $[\cdot]$-free. By hypothesis $x \nVdash \Delta$, thus $x \nVdash [\Phi_i \Rightarrow \Psi_i]$ for $i = 1, \ldots, k$. Therefore there

are $y_1, \ldots, y_k$ with Rxy_i for $i = 1, \ldots, k$ such that $y_i \Vdash \Phi_i \Rightarrow \Psi_i$. This means that (a) $y_i \Vdash C$ for every $C \in \Phi_i$ and (b) $y_i \nVdash \Psi_i$. By (FC) property there are $y'_1, \ldots, y'_k$ such that $Rx'y'_i$ and $y'_i \geq y_i$ for $i = 1, \ldots, k$. By (a) it follows that (c) $y'_i \Vdash C$ for every $C \in \Phi_i$; moreover by induction hypothesis it follows that (d) $y'_i \nVdash \Psi^*_i$. Thus from (c) and (d) we have $y'_i \nVdash \Phi_i \Rightarrow \Psi^*_i$, whence $x' \nVdash [\Phi_i \Rightarrow \Psi^*_i]$ for for $i = 1, \ldots, k$, which means that $x' \nVdash \Delta^*$. ◀

Proof of Theorem 49. (Sketch) We prove that PROCEDURE(A) terminates producing a finite derivation, in this case all leaves are axioms or global-saturated. A non-axiomatic leaf S is necessarily global-saturated, otherwise S would be further expanded in Step 8 of PROCEDURE(A) and it would not be a leaf. Thus it suffices to prove that the procedure produces a finite derivation. Let $\mathcal{D}$ built by PROCEDURE(A). First we claim that all branches of $\mathcal{D}$ are finite. Suppose for the sake of a contradiction that $\mathcal{D}$ contains an infinite branch $\mathcal{B} = S_0, \ldots, S_i, \ldots$, with $S_0 ==\!\Rightarrow A$. The branch is generated by applying repeatedly $\mathbf{EXP1}(\cdot), \mathbf{EXP2}(\cdot)$ and $\mathbf{EXP3}(\cdot)$ to each S_i (or more precisely to some $T_i \in^+ S_i$) . Since each one of these sub-procedures terminates, the three of them must infinitely alternate on the branch. By (invariant) Lemma, if $T_i \in^+ S_i$ satisfies a saturation condition for a rule (R) or is blocked in (S_i) it will remain so in all S_j with $j > i$. That is to say, further steps in the branch cannot "undo" a fulfilled saturation condition or "unblock" a blocked sequent. We can conclude that the branch must contain infinitely many phases of $\mathbf{EXP3}(\cdot)$ each time applied to an unblocked sequent in some S_i. This entails that $\mathcal{B}$ contains infinitely many sequents that are not $\simeq$-equivalent, but this contradicts previous lemma 48. Thus each branch of the derivation $\mathcal{D}$ built by PROCEDURE(A) is finite. To conclude the proof, just observe that $\mathcal{D}$ is a tree whose branches have a finite length and is finitely branching (namely each node/sequent has at most 2 successors, as the rules of $\mathbf{CC_{FIK}}$ are at most binary), therefore $\mathcal{D}$ is finite. ◀

Proof of Proposition 46. We only prove the claim for $\mathbf{EXP1}(\mathcal{D}, S, T)$, the other cases being obvious. To this purpose we show that any derivation $\mathcal{D}o$, with root $\Gamma \Rightarrow \Delta^\sharp$ and generated by R1-rules, is finite. Then the claim follows since $\mathbf{EXP1}(\mathcal{D}, S, T)$ is obtained simply by "appending" $\mathcal{D}o$ to $\mathcal{D}$, where we replace every sequent T' in $\mathcal{D}o$ by $G\{T'\}$, as $S = G\{T\}$. In order to prove that $\mathcal{D}o$ is finite, notice that (i) all R1-rules are at most binary, (ii) the length of a branch of $\mathcal{D}o$ is bounded by the size of the maximal sequent that can occur in it because of non-redundancy restriction. But by proposition 45, every sequent T' in $\mathcal{D}o$ has a bounded size (namely $O(|T|^{|T|+1})$), whence we get a bound on the length of any branch of $\mathcal{D}o$. In conclusion $\mathcal{D}o$ is a finitely-branching tree, whose branches have a finite length, whence it is finite. ◀

In the following proofs, we abbreviate $R_S, \leq_S$ as R and $\leq$ respectively for readability.

Proof of Proposition 51. For (HP), take arbitrary $x_{S_1}, x_{S_2} \in W_S$ with $x_{S_1} \leq x_{S_2}$. Suppose S_1, S_2 are of form $\Gamma_1 \Rightarrow \Delta_1$ and $\Gamma_2 \Rightarrow \Delta_2$ respectively, then $\Gamma_1 \Rightarrow \Delta_1 \subseteq^{\mathbf{S}} \Gamma_2 \Rightarrow \Delta_2$. By definition, it follows $\Gamma_1 \subseteq \Gamma_2$. As $V_S(x_{S_1}) = \{p \mid p \in \Gamma_1\}$ and $V_S(x_{S_2}) = \{p \mid p \in \Gamma_2\}$, we have $V_S(x_{S_1}) \subseteq V_S(x_{S_2})$.

For (FC), take arbitrary $x_{\Gamma\Rightarrow\Delta}, x_{\Sigma\Rightarrow\Pi}, x_{\Lambda\Rightarrow\Theta} \in W_S$ where $x_{\Gamma\Rightarrow\Delta} \leq x_{\Sigma\Rightarrow\Pi}$ as well as $Rx_{\Gamma\Rightarrow\Delta}x_{\Lambda\Rightarrow\Theta}$, our goal is to find some $x_0 \in W_S$ s.t. both $x_{\Lambda\Rightarrow\Theta} \leq x_0$ and $Rx_{\Sigma\Rightarrow\Pi}x_0$ hold. Since $Rx_{\Gamma\Rightarrow\Delta}x_{\Lambda\Rightarrow\Theta}$, by the definition of R, we see that $[\Lambda \Rightarrow \Theta] \in \Delta$ and hence $\Gamma \Rightarrow \Delta$ can be written explicitly as $\Gamma \Rightarrow \Delta', [\Lambda \Rightarrow \Theta]$. Meanwhile, since $x_{\Gamma\Rightarrow\Delta} \leq x_{\Sigma\Rightarrow\Pi}$, by the definition of $\leq$, we have $\Gamma \Rightarrow \Delta', [\Lambda \Rightarrow \Theta] \subseteq^{\mathbf{S}} \Sigma \Rightarrow \Pi$. By the definition of structural inclusion, there is a block $[\Phi \Rightarrow \Psi] \in \Pi$ s.t. $\Lambda \Rightarrow \Theta \subseteq^{\mathbf{S}} \Phi \Rightarrow \Psi$. Since $\Phi \Rightarrow \Psi \in^+ \Sigma \Rightarrow \Pi \in^+ S$ and $\in^+$ is transitive, we see that $x_{\Phi\Rightarrow\Psi} \in W_S$ as well. Take $x_{\Phi\Rightarrow\Psi}$ to be x_0, by the construction of $\mathcal{M}_S$, it follows directly $x_{\Lambda\Rightarrow\Theta} \leq x_0$ and $Rx_{\Sigma\Rightarrow\Pi}x_0$. ◀

Proof of Lemma 52. We prove the lemma by induction on the complexity of A. For convenience, we abbreviate $x_{\Phi \Rightarrow \Psi}$ as x.

We only show the case when A is of the form $\Box B$. For (a), let $\Box B \in \Phi$. $\Phi \Rightarrow \Psi$ satisfies the saturation condition associated with $(\Box_R)$ for $\Box B$ regardless of whether the sequent itself is blocked or not. Assume for the sake of a contradiction that $x \nVdash \Box B$. Then there exists $x_{\Sigma \Rightarrow \Pi}, x_{\Lambda \Rightarrow \Theta}$ denoted as x_1, x_2 s.t. $x \leq x_1, Rx_1x_2$ and $x_2 \nVdash B$. By IH, we see that $B \notin \Lambda$. Meanwhile, according to the model construction, we see that $\Phi \Rightarrow \Psi \subseteq^{\mathbf{S}} \Sigma \Rightarrow \Pi$ and $[\Lambda \Rightarrow \Theta] \in \Pi$. Moreover we have $\Phi \subseteq \Sigma$, thus $\Box B \in \Sigma$ as well. Also, since $\Sigma \Rightarrow \Pi$ is of form $\Sigma \Rightarrow \Pi', [\Lambda \Rightarrow \Theta]$, by the saturation condition associated with $(\Box_L)$, we have $B \in \Lambda$, which leads to a contradiction.

For (b), let $\Box B \in \Psi$. We distinguish whether $\Phi \Rightarrow \Psi$ is blocked or not. Assume that $\Phi \Rightarrow \Psi$ is not blocked, then it satisfies the one of the two saturation conditions associated with $(\Box_R)$ for $\Box B$:

(1) there is a block $[\Lambda \Rightarrow \Theta] \in \Psi$ with $B \in \Theta$. By IH, we have $x_{\Lambda \Rightarrow \Theta} \nVdash B$. By reflexivity $x \leq x$ and model construction $Rxx_{\Lambda \Rightarrow \Theta}$, so that $x \nVdash \Box B$.

(2) there is a block $\langle \Omega \Rightarrow [\Lambda \Rightarrow \Theta], \Xi \rangle \in \Psi$ with $B \in \Theta$. Denote the sequent $\Omega \Rightarrow [\Lambda \Rightarrow \Theta], \Xi$ by S_0. Since $\Phi \Rightarrow \Psi$ is saturated with (trans) and (inter), by Proposition 35, we have $\Phi \Rightarrow \Psi \subseteq^{\mathbf{S}} S_0$. According to the model construction, we see that $x \leq x_{S_0}$ and $Rx_{S_0}x_{\Lambda \Rightarrow \Theta}$. Since $B \in \Theta$, by IH we have $x_{\Lambda \Rightarrow \Theta} \nVdash B$ and we can conclude $x \nVdash \Box B$.

Assume that $\Phi \Rightarrow \Psi$ is blocked and does not satisfy condition (1) for $\Box B$, otherwise the proof proceeds as in case (1) above. Then there is an unblocked sequent $\Sigma \Rightarrow \Pi \in^+ S$ such that $\Phi \Rightarrow \Psi$ is blocked by it. Then $\Sigma \Rightarrow \Pi \simeq \Phi \Rightarrow \Psi$, which implies $\Pi^\sharp = \Psi^\sharp$, so $\Box B \in \Pi$ as well. Moreover, by definition, we have $\Phi \Rightarrow \Psi \subseteq^{\mathbf{S}} \Sigma \Rightarrow \Pi$, whence by model construction (**) $x \leq x_{\Sigma \Rightarrow \Pi}$. Given that $\Sigma \Rightarrow \Pi$ is R3-saturated, it satisfies the saturation condition associated with $(\Box_R)$ for $\Box B$, but since $\Sigma \Rightarrow \Pi \simeq \Phi \Rightarrow \Psi$, we have that $\Sigma \Rightarrow \Pi$ does not satisfy condition (1), thus it must satisfy condition (2). Therefore there is a block $\langle \Omega \Rightarrow [\Lambda \Rightarrow \Theta], \Xi \rangle \in \Pi$, such that $B \in \Theta$. Letting $S_0 = \Omega \Rightarrow [\Lambda \Rightarrow \Theta], \Xi$, we have $x_{\Sigma \Rightarrow \Pi} \leq x_{S_0}$ and $Rx_{S_0}x_{\Lambda \Rightarrow \Theta}$. By (**) we have also $x \leq x_{S_0}$ and we conclude as in case (2) above. ◀

Tropical Mathematics and the Lambda-Calculus I

Metric and Differential Analysis of Effectful Programs

Davide Barbarossa
Università di Bologna, Italy

Paolo Pistone
Università di Bologna, Italy

Abstract

We study the interpretation of the lambda-calculus in a framework based on tropical mathematics, and we show that it provides a unifying framework for two well-developed quantitative approaches to program semantics: on the one hand program metrics, based on the analysis of program sensitivity via Lipschitz conditions, on the other hand resource analysis, based on linear logic and higher-order program differentiation. To do that, we focus on the semantics arising from the relational model weighted over the tropical semiring, and we discuss its application to the study of "best case" program behavior for languages with probabilistic and non-deterministic effects. Finally, we show that a general foundation for this approach is provided by an abstract correspondence between tropical algebra and Lawvere's theory of generalized metric spaces.

2012 ACM Subject Classification Theory of computation → Lambda calculus; Theory of computation → Categorical semantics; Theory of computation → Linear logic

Keywords and phrases Relational semantics, Differential lambda-calculus, Tropical semiring, Program metrics, Lawvere quantale

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.14

Related Version *Extended Version*: `https://arxiv.org/abs/2311.15704`

Funding This work has been supported by the European Research Council through the project ERC CoG 818616 DIAPASoN.

1 Introduction

In recent years, more and more interest in the programming language community has been directed towards the study of *quantitative* properties of programs like, e.g., the number of computation steps or the probability of convergence, as opposed to purely *qualitative* properties like termination or program equivalence. Notably, a significant effort has been made to extend, or adapt, well-established qualitative methods, like type systems, relational logics or denotational semantics, to account for quantitative properties. We can mention, for example, intersection type systems aimed at capturing time or space resources [3, 28] or convergence probabilities [9, 19], relational logics to account for probabilistic properties like, e.g., differential privacy [13] or metric preservation [24, 68], as well as the study of denotational models for probabilistic [33, 46] or differential [36] extensions of the λ-calculus. The main reason to look for methods relying on (quantitative extensions of) type-theory or denotational semantics is that these approaches yield *modular* and *compositional* techniques, that is, allow one to deduce properties of complex programs from the properties of their constituent parts.

Two approaches to quantitative semantics. Among such quantitative approaches, two have received considerable attention.

On the one hand, there is the approach of *program metrics* [10, 11, 68] and *quantitative equational theories* [58]: when considering probabilistic or approximate computation, rather than asking whether two programs compute *the same* function, it makes more sense to ask whether they compute functions which do not differ *too much*. This has motivated the study of denotational frameworks in which types are endowed with a metric, measuring similarity of behavior; this approach has found applications in, e.g., differential privacy [68] and coinductive methods [12], and was recently extended to account for the full λ-calculus [25, 42, 66].

On the other hand, there is the approach based on *differential* [36] or *resource-aware* [18] extensions of the λ-calculus, which is well-connected to the so-called *relational semantics* [30, 50, 56] and has a syntactic counterpart in the study of *non-idempotent* intersection types [28, 59]. This family of approaches have been exploited to account for higher-order program differentiation [36], to establish reasonable *cost-models* for the λ-calculus [2], and have also been shown suitable to the probabilistic setting [9, 19, 50].

In both approaches the notion of *linearity*, in the sense of linear logic [43] (i.e. of using inputs exactly once), plays a crucial role. In metric semantics, linear programs correspond to *non-expansive* maps, i.e. to functions that do not increase distances, and the possibility of duplicating inputs leads to interpret programs with a fixed duplication bound as *Lipschitz-continuous* maps [10]. By contrast, in the standard semantics of the differential λ-calculus, linear programs correspond to linear maps, in the usual algebraic sense, while the possibility of duplicating inputs leads to consider functions defined as *power series*.

A natural question, at this point, is whether these two apparently unrelated ways of interpreting linearity and duplication can be somehow reconciled. At a first glance, there seems to be a "logarithmic" gap between the two approaches: in metric models a program duplicating an input n times yields a linear (hence *Lipschitz*) function nx, whereas in differential models it would lead to a polynomial function x^n, thus not Lipschitz. The fundamental idea behind this work is the observation that this gap is naturally overcome once we interpret these functions in the framework of tropical mathematics, where, as we will see, the monomial x^n precisely reads as the linear function nx.

Tropical mathematics and program semantics. Tropical mathematics was introduced in the seventies by the Brazilian mathematician Imre Simon [72] as an alternative approach to algebra and geometry where the usual ring structure of numbers based on addition and multiplication is replaced by the semiring structure given, respectively, by "min" and "+". For instance, the polynomial $p(x, y) = x^2 + xy^2 + y^3$, when interpreted over the tropical semiring, translates as the piecewise linear function $\varphi(x, y) = \min\{2x, x + 2y, 3y\}$. In the last decades, tropical geometry evolved into a vast and rich research domain, providing a combinatorial counterpart of usual algebraic geometry, with important connections with optimisation theory [55]. Computationally speaking, working with min and + is generally easier than working with standard addition and multiplication; for instance, the fundamental (and generally intractable) problem of finding the roots of a polynomial admits a *linear time* algorithm in the tropical case (and, moreover, the tropical roots can be used to approximate the actual roots [62]). The combinatorial nature of several methods in tropical mathematics explains why these are so widely applied in computer science, notably for convex analysis and machine learning (see [57] for a recent survey).

Coming back to our discussion on program semantics, tropical mathematics seems to be just what we look for, as it turns polynomial functions like x^n into Lipschitz maps like nx. At this point, it is worth mentioning that a tropical variant of the usual relational semantics of linear logic and the λ-calculus has already been considered [50], and shown

capable of capturing *best-case* quantitative properties, but has not yet been studied in detail. Furthermore, connections between tropical linear algebra and metric spaces have also been observed [38] within the abstract setting of *quantale-enriched* categories [47, 74]. However, a thorough investigation of the interpretation of the λ-calculus within tropical mathematics and of the potentialities of its applications has not yet been undertaken.

In this paper we make a first step in such direction, by demonstrating that the relational interpretation of the λ-calculus based on tropical mathematics does indeed provide the desired bridge between differential and metric semantics, and suggests new combinatorial methods to study probabilistic and non-deterministic programs.

Contributions and outline of the paper. Our contributions in this paper are the following:

- We first show that tropical polynomials naturally arise in the best-case analysis of probabilistic and non-deterministic programs, turning the study of quantitative program behavior into a purely combinatorial problem. This is in Sections 3 and 4.
- We study the relational model over the tropical semiring, which provides a semantics of effectful extensions of the simply typed λ-calculus (STLC in the following) and PCF [67]. Notably, we show that higher-order programs are interpreted by a generalizations of tropical power series [61], and we show that these functions are locally Lipschitz-continuous, thus yielding a full-scale metric semantics. This is in Sections 5 and 6.
- We exploit the differential structure of the relational model to study the *tropical Taylor expansion* of a λ-term, which can be seen as an approximation of the term by way of Lipschitz-continuous maps, and we show that it can be used to compute approximated Lipschitz-constants for higher-order programs. This is in Section 7.
- We conclude by framing the connection between the tropical, differential and metric viewpoints at a more abstract level. We recall a well-known correspondence between Lawvere's *generalized metric spaces* [51, 74] and modules over the tropical semi-ring [69] and we show that it yields a model of the differential λ-calculus which extends the tropical relational model. This is in Section 8.

2 A Bridge between Metric and Differential Aspects

In this section, we discuss in some more detail the two approaches to quantitative semantics we mentioned in the Introduction, at the same time providing an overview of how we aim at bridging them using tropical mathematics.

Metric Approach: Bounded λ-Calculus. In many situations (e.g. when dealing with computationally difficult problems) one does not look for algorithms to compute a function *exactly*, but rather to approximate it (in an efficient way) within some error bound. In other common situations (e.g. in differential privacy [8, 68]) one needs to verify that an algorithm is not *too sensitive* to errors, that is, that a small error in the input will produce a comparably small error in the output. In all these cases, it is common to consider forms of denotational semantics in which types are endowed with a *behavioral metric*, that is, a metric on programs which accounts for differences in behavior. A fundamental insight coming from this line of work is that, if one can somehow *bound* the number of times that a program may duplicate its input, the resulting program will be *Lipschitz-continuous*: if M may duplicate at most L times, then an error ϵ between two inputs will result in an error less or equal to $L \cdot \epsilon$ in the corresponding outputs [10, 68] (yet, this property may fail in a concurrent setting, see e.g. [41]). For instance, the higher-order program $M = \lambda f.\lambda x.f(f(x))$, which

duplicates the functional input f, yields a 2-Lipschitz map between the metric space $\mathbb{R} \multimap \mathbb{R}$ of non-expansive real functions and itself: if f, g are two non-expansive maps differing by at most ϵ (i.e. for which $|f(x) - g(x)| \leqslant \epsilon$ holds for all $x \in \mathbb{R}$), then the application of M to f and g will produce two maps differing by at most 2ϵ.

These observations have led to the study of λ-calculi with *graded* exponentials, like Fuzz [68], inspired from Girard's Bounded Linear Logic [44], which have been applied to the study of differential privacy [10, 39]. The types of such systems are defined by combining linear constructors with a *graded linear exponential comonad* $!_r(-)$ [49].

Yet, what about the good old, "unbounded", simply typed λ-calculus? Actually, by using unbounded duplications, one might lose the Lipschitz property. For instance, while the functions $M_k = \lambda x.k \cdot x : \mathbb{R} \to \mathbb{R}$ are all Lipschitz-continuous, with Lipschitz constant k, the function $M = \lambda x.x^2$ obtained by "duplicating" x is not Lipschitz anymore: M is, so to say, *too* sensitive to errors. More abstractly, it is well-known that the category Met of metric spaces and non-expansive maps, is *not* cartesian closed, so it is not a model of STLC (yet, several cartesian closed *sub*-categories of Met do exist, see e.g. [21, 25]). Still, one might observe that the program M above is actually Lipschitz-continuous, if not globally, at least *locally* (i.e. over any compact set). Indeed, some cartesian closed categories of locally Lipschitz maps have been produced in the literature [33, 66], and a new example will be exhibited in this paper.

Resource Approach: the Differential λ-Calculus. A different family of approaches to linearity and duplication arises from the study of the *differential λ-calculus* [36] (and differential linear logic [30]) and its categorical models. The key ingredient is a *differential constructor* $\mathsf{D}[\_,\_]$, added to the usual syntax of the λ-calculus. The intuition is that, given M of type $A \to B$ and N of type A, the program $\mathsf{D}[M, N]$, still of type $A \to B$, corresponds to the *linear application* of M to N: this means that N is passed to M so that the latter may use it exactly once. This is also why $\mathsf{D}[M, N]$ still has type $A \to B$, since M might need *other* copies of an input of type A. In particular, the application of $\mathsf{D}[M, N]$ to an "error term" 0 ensures that M will use N exactly once (we say *linearly*).

The reason why D is called a "differential", is twofold: semantically, its interpretation is a generalisation of the usual differential form analysis (see Section 5); syntactically, it allows to define the so-called *Taylor expansion* $\mathcal{T}$ of programs: the idea is that one can expand any application MN as an infinite formal sum of *linear* applications $\mathsf{D}^k[M, N^k]0$, i.e. where N is linearly passed exactly k times to M; doing this recursively gives rise to the suggestive Taylor formula $\mathcal{T}(MN) := \sum_{k=0}^{\infty} \frac{1}{!k} \cdot \mathsf{D}^k[\mathcal{T}(M), \mathcal{T}(N)^k]0$. In other words, unbounded duplications correspond to some sort of limit of bounded, but arbitrarily large, ones.

Tropical Mathematics: Lipschitz Meets Taylor. At this point, as the Taylor formula decomposes an unbounded application as a limit of bounded ones, one might well ask whether it could be possible to see this formula as interpreting a λ-term as a limit of Lipschitz maps, in some sense, thus bridging the metric and differential approaches. Here, a natural direction to look for is the *weighted relational semantics* [50], due to its strict relations with the Taylor expansion of programs. However, in this semantics, arbitrary terms correspond to power series, and terms with bounded applications correspond to polynomials, hence in any case to functions which are *not* Lipschitz.

Yet, what if such polynomials were tropical ones, i.e. piecewise linear functions? This way, the Taylor formula could really be interpreted as a decomposition of λ-terms via limits (indeed, infs) of Lipschitz maps. In other words, unbounded term application could be seen as a limit of *more and more sensitive* operations.

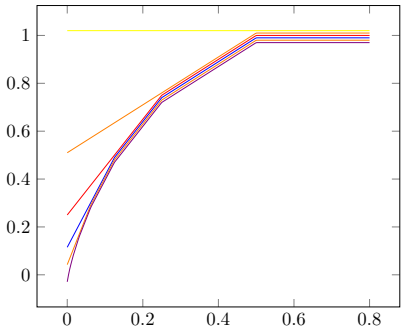


Figure 1 Tropical polynomials $\varphi_0, \dots, \varphi_4$ (top to bottom), and their limit tps φ (in violet). The points where the slope changes are the tropical roots of φ, i.e. the points $x = 2^{-(i+1)}$, satisfying $ix + 2^{-i} = (i+1)x + 2^{-(i+1)}$.

This viewpoint, that we develop in the following sections, leads to the somehow unexpected discovery of a bridge between the metric and differential study of higher-order programs. This connection not only suggests the application of optimization methods based on tropical mathematics to the study of the λ-calculus and its quantitative extensions, but it scales to a more abstract level, leading to introduce a differential operator for continuous functors between *generalized* metric spaces (in the sense of [51]), as shown in Section 8.

3 Tropical Polynomials and Power Series

At the basis of our approach is the observation that the *tropical semiring* $([0,\infty], \min, +)$, which is at the heart of tropical mathematics, coincides with the *Lawvere quantale* $\mathbb{L} = ([0,\infty], \geqslant, +)$ [47, 74], the structure at the heart of the categorical study of metric spaces initiated by Lawvere himself [51]. Let us recall that a quantale is a complete lattice endowed with a continuous monoid action. In the case of $\mathbb{L}$, the lattice is defined by the reverse order $\geqslant$ on $\mathbb{R}$, and the monoid action is provided by addition. Notice that the lattice join operation of $\mathbb{L}$ coincides with the idempotent semiring operation min.

Power series and polynomials over the tropical semiring are defined as follows:

▶ **Definition 1.** *A* tropical power series *(tps) in k variables is a function $f : \mathbb{L}^k \to \mathbb{L}$ of shape $f(x) = \inf_{i \in I}\{ix + \hat{f}(i)\}$, where $I \subseteq \mathbb{N}^k$, ix is the scalar product and $\hat{f} \in \mathbb{L}^{\mathbb{N}^k}$ is a vector of coefficients. When I is finite, f is called a* tropical polynomial.

Hence, a unary tps is a function $f : \mathbb{L} \to \mathbb{L}$ of the form $f(x) = \inf_{i \in I}\{ix + a_i\}$, with $I \subseteq \mathbb{N}$ and the $a_i \in \mathbb{L}$. In Section 5 we also consider tps in *infinitely many* variables.

A tropical polynomial is always a piece-wise linear function since, e.g. in one variable, it has shape $f(x) = \min_{0 \leqslant j \leqslant n}\{i_j x + c_{i_j}\}$. For example, the polynomials $\varphi_n(x) = \min_{0 \leqslant j \leqslant n}\{jx + 2^{-j}\}$ are illustrated in Fig. 1 for $0 \leqslant n \leqslant 4$.

A *tropical root* of a tps φ is a point $x \in \mathbb{L}$ where φ is not differentiable (i.e. where the slope of φ changes). When φ is a polynomial, the roots of φ coincide with the points where the minimum defining φ is attained at least twice (see Figure 1). Unlike in standard algebra, tropical roots of tropical polynomials can be computed in linear time [62].

While tropical polynomials are essentially combinatorial objects, this cannot be said for tps: since infs are not in general mins, a tps is a "limit" of tropical polynomials of higher and higher degree, and its behavior is in general way more difficult to study than that of tropical polynomials [61]. E.g., the tps $\varphi(x) := \inf_{n \in \mathbb{N}}\{nx + 2^{-n}\}$ (see Fig. 1) is the "limit" of the polynomials φ_n.

▶ Remark 2. There is a well-known relation between tropical polynomial/power series and usual polynomials/power series. For a power series $f(x) = \sum_{i\in I} a_i x^i$ (polynomials being the case for I finite) defined on $[0,1]$ and coefficients a_i in $[0,1]$, one can define the *tropicalisation* $\mathtt{t}f : \mathbb{L} \to \mathbb{L}$ of f as the tropical polynomial/power series function $\mathtt{t}f(\alpha) := \inf_{i\in I}\{-\log(a_i) + i\alpha\}$.

For instance, the tropicalisation sends the infinite power series $\sum_{n\geqslant 2} x^n$ to the tps $\varphi(x) = \inf_n (n+2)x$. Yet the latter always coincides with the tropical monomial $\varphi(x) = 2x$, since $nx \geqslant 0$ for all $n \in \mathbb{N}$, which is in turn the tropicalisation of the polynomial x^2. This shows that tropicalisation is *not* in general an injective operation and in fact, as we show in Theorem 3 below, tps (in finitely many variables) have a tendency to collapse, if not globally at least locally, onto tropical polynomials.

For instance, by looking at Fig. 1 it appears that, *far from* 0, φ behaves like some of the polynomials φ_n. In particular, φ coincides on $[\epsilon, \infty]$ with φ_n, for $\epsilon \geqslant 2^{-(n+1)}$ (the smallest tropical root of φ_n). However, at $x = 0$ we have that $\varphi(x = 0) = \inf_{n\in\mathbb{N}} 2^{-n} = 0$, and this is the only point where the inf is *not* a min. Also, while the derivative of f is bounded on all $(0, \infty)$, for $x \to 0^+$ it tends to ∞. In fact, this is a general phenomenon, as showed below:

▶ **Theorem 3.** *For all tps* $f(x) = \inf_{n\in\mathbb{N}^k}\{nx + \hat{f}(n)\}$*, for all* $0 < \epsilon < \infty$*, there is a* finite $\mathcal{F}_\epsilon \subseteq \mathbb{N}^k$ *such that* f *coincides on all* $[\epsilon, \infty]^k$ *with* $P_\epsilon(x) := \min_{n\in\mathcal{F}_\epsilon}\{nx + \hat{f}(n)\}$.

As we'll see, the potential of collapsing infinitary objects (i.e. tps) into combinatorial ones (i.e. tropical polynomials), is one of the most intriguing features of tropical semantics.

For the interested reader, we provide the proof of Theorem 3 below. Let us first set the following:

▶ **Definition 4.** *Let* $\preceq$ *be the product order on* $\mathbb{N}^k$ *(i.e. for all* $m, n \in \mathbb{N}^K$*,* $m \preceq n$ *iff* $m_i \leqslant n_i$ *for all* $1 \leqslant i \leqslant K$*). Of course* $m \prec n$ *holds exactly when* $m \preceq n$ *and* $m_i < n_i$ *for at least one* $1 \leqslant i \leqslant K$*. Finally, we set* $m \prec_1 n$ *iff* $m \prec n$ *and* $\sum_{i=1}^K n_i - m_i = 1$ *(i.e. they differ on exactly one coordinate).*

▶ Remark 5. If $U \subseteq \mathbb{N}^K$ is infinite, then U contains an infinite ascending chain $m_0 \prec m_1 \prec m_2 \prec \dots$. This is a consequence of König Lemma (KL): consider the directed acyclic graph $(U, \prec_1)$, indeed a K-branching tree; if there is no infinite ascending chain $m_0 \prec m_1 \prec m_2 \prec \dots$, then in particular there is no infinite ascending chain $m_0 \prec_1 m_1 \prec_1 m_2 \prec_1 \dots$ so the tree U has no infinite ascending chain; then by KL it is finite, contradicting the assumption.

Proof of Theorem 3. We will actually show the existence of $\mathcal{F}_\epsilon \subseteq_{\text{fin}} \mathbb{N}^k$ such that:

1. if $\mathcal{F}_\epsilon = \varnothing$ then $f(x) = +\infty$ for all $x \in \mathbb{L}^k$;
2. if $f(x_0) = +\infty$ for some $x_0 \in [\epsilon, \infty)^K$ then $\mathcal{F}_\epsilon = \varnothing$;
3. the restriction of f on $[\epsilon, \infty]^k$ coincides with $P_\epsilon(x) := \min\limits_{n\in\mathcal{F}_\epsilon}\{nx + \hat{f}(n)\}$.

Let $\mathcal{F}_\epsilon$ be the complementary in $\mathbb{N}$ of the set:

$$\{n \in \mathbb{N}^K \mid \textit{either } \hat{f}(n) = +\infty \textit{ or there is } m \prec n \textit{ s.t. } \hat{f}(m) \leqslant \hat{f}(n) + \epsilon\}.$$

In other words, $n \in \mathcal{F}_\epsilon$ iff $\hat{f}(n) < +\infty$ and for all $m \prec n$, one has $\hat{f}(m) > \hat{f}(n) + \epsilon$. Suppose that $\mathcal{F}_\epsilon$ is infinite; then, using Remark 5, it contains an infinite ascending chain $\{m_0 \prec m_1 \prec \cdots\}$. By definition of $\mathcal{F}_\epsilon$ we have then $+\infty > \hat{f}(m_0) > \hat{f}(m_1) + \epsilon > \hat{f}(m_2) + 2\epsilon > \cdots$, so that $+\infty > \hat{f}(m_0) > \hat{f}(m_i) + i\epsilon \geqslant i\epsilon$ for all $i \in \mathbb{N}$. This contradicts the Archimedean property of $\mathbb{R}$. Hence $\mathcal{F}_\epsilon$ is finite.

1. We show that if $\mathcal{F}_\epsilon = \varnothing$, then $\hat{f}(n) = +\infty$ for all $n \in \mathbb{N}^K$. This immediately entails the desired result. We go by induction on the well-founded order $\prec$ over $n \in \mathbb{N}^K$:
   - if $n = 0^K \notin \mathcal{F}_\epsilon$, then $\hat{f}(n) = +\infty$, because there is no $m \prec n$.
   - if $n \notin \mathcal{F}_\epsilon$, with $n \neq 0^K$ then either $\hat{f}(n) = +\infty$ and we are done, or there is $m \prec n$ s.t. $\hat{f}(m) \leqslant \hat{f}(n) + \epsilon$. By induction $\hat{f}(m) = +\infty$ and, since $\epsilon < +\infty$, this entails $\hat{f}(n) = +\infty$.
2. If $f(x_0) = +\infty$ with $x_0 \in [\epsilon, \infty)^K$, then necessarily $\hat{f}(n) = +\infty$ for all $n \in \mathbb{N}^K$. Therefore, no $n \in \mathbb{N}^K$ belongs to $\mathcal{F}_\epsilon$.
3. We have to show that $f(x) = P_\epsilon(x)$ for all $x \in [\epsilon, +\infty]^K$. By 1), it suffices to show that we can compute $f(x)$ by taking the inf, that is therefore a min, only in $\mathcal{F}_\epsilon$ (instead of all $\mathbb{N}^K$). If $\mathcal{F}_\epsilon = \varnothing$ then by 2) we are done (remember that $\min \varnothing := +\infty$). If $\mathcal{F}_\epsilon \neq \varnothing$, we show that for all $n \in \mathbb{N}^K$, if $n \notin \mathcal{F}_\epsilon$, then there is $m \in \mathcal{F}_\epsilon$ s.t. $\hat{f}(m) + mx \leqslant \hat{f}(n) + nx$. We do it again by induction on $\prec_1$:
   - if $n = 0^K$, then from $\mathbf{n} \notin \mathcal{F}_\epsilon$, by definition of $\mathcal{F}_\epsilon$, we have $\hat{f}(n) = +\infty$ (because there is no $n' \prec n$). So any element of $\mathcal{F}_\epsilon \neq \varnothing$ works.
   - if $n \neq 0^K$, then we have two cases: either $\hat{f}(n) = +\infty$, in which case we are done as before by taking any element of $\mathcal{F}_\epsilon \neq \varnothing$. Or $\hat{f}(n) < +\infty$, in which case (again by definition of $\mathcal{F}_\epsilon$) there is $n' \prec n$ such that $\hat{f}(n') \leqslant \hat{f}(n) + \epsilon$ ($\star$). Therefore we have (remark that the following inequalities hold also for the case $x = +\infty$):

$$\begin{array}{lcll} \hat{f}(n') + n'x & \leqslant & \hat{f}(n) + \epsilon + n'x & \text{by } (\star) \\ & \leqslant & \hat{f}(n) + (n - n')x + n'x & \text{because } \epsilon \leqslant \min x \text{ and } \min x \leqslant (n - n')x \\ & = & \hat{f}(n) + nx. & \end{array}$$

   Now, if $n' \in \mathcal{F}_\epsilon$ we are done. Otherwise $n' \notin \mathcal{F}_\epsilon$ and we can apply the induction hypothesis on it, obtaining an $m \in \mathcal{F}_\epsilon$ s.t. $\hat{f}(m) + mx \leqslant \hat{f}(n') + n'x$. Therefore this m works. ◀

4 Tropical Semantics and First Order Effectful Programs

Before discussing how full-scale higher-order programming languages can be interpreted in terms of tropical power series, we highlight how such functions may naturally arise in the study of effectful programming languages. We will see that, when considering probabilistic and non-deterministic programs, tropical tools can be used to describe the behavior of programs in *the best/worst case*, and may lead to collapse the description of infinitely many possible behaviors into a combinatorial account of the optimal ones.

Maximum Likelihood Estimators for Probabilistic Languages. Let us start with a very basic probabilistic language: the terms are $M ::= \texttt{True} \mid \texttt{False} \mid M \oplus_p M$, for $p \in [0, 1]$, and the operational semantics is $M \oplus_p N \to pM$ and $M \oplus_p N \to (1 - p)N$, so that $M \oplus_p N$ plays the role of a probabilistic coin toss of bias p. Consider the program $M := (\texttt{True} \oplus_p \texttt{False}) \oplus_p ((\texttt{True} \oplus_p \texttt{False}) \oplus_p (\texttt{False} \oplus_p \texttt{True}))$. Calling $q = 1 - p$, to each occurrence of $\texttt{True}$ or $\texttt{False}$ in M, univocally determined by an address $\omega \in \{l, r\}^*$, is associated a monomial $P_\omega(p, q)$ which determines the probability of the event "$M \twoheadrightarrow_\omega \texttt{True}/\texttt{False}$", that is, that M reduces to $\texttt{True}/\texttt{False}$ according to the choices in ω. Thinking of p, q as parameters, $P_\omega(p, q)$ can thus be read as the *likelihood function* of the event "$M \twoheadrightarrow_\omega \texttt{True}/\texttt{False}$". For instance, we have $P_{rll}(p, q) := qp^2$, $P_{rrr}(p, q) := q^3$, and $P_{rrl}(p, q) = P_{rlr}(p, q) := q^2p$. The polynomial function $Q_{\texttt{True}}(p, q) := P_{ll}(p, q) + P_{rll}(p, q) + P_{rrr}(p, q) = p^2 + p^2q + q^3$ gives instead the probability of the event "$M \twoheadrightarrow \texttt{True}$", and analogously for $Q_{\texttt{False}}(p, q) := P_{lr}(p, q) + P_{rrl}(p, q) + P_{rlr}(p, q) = pq + 2pq^2$.

This way, the probabilistic evaluation of M is presented as a *hidden Markov model* [14], a fundamental statistical model, and notably one to which tropical methods are generally applied [64]. Then, a natural question in this case, for a fixed ω_0, is the following: knowing that M reached a normal form, say $\mathtt{True}$, what is the *maximum likelihood estimator* for the event "$M \twoheadrightarrow_{\omega_0} \mathtt{True}$"? In other words, what is the choice of p, q that maximizes the conditional probability $\mathbb{P}(M \twoheadrightarrow_{\omega_0} \mathtt{True} \mid M \twoheadrightarrow \mathtt{True})$, i.e. that makes ω_0 the most likely path among those leading to $\mathtt{True}$?

Answering this question amounts at finding a solution to the following constrained maximization problem in the unknown $p \in [0,1]$:

$$P_{\omega_0}(p, 1-p) = \max_{\omega} P_{\omega}(p, 1-p)$$

which is related to the polynomial, say, $Q_{\mathtt{True}}(p,q)$. Since $-\log P_\omega(p,q) = (\mathtt{t}P_\omega)(-\log p, -\log q)$, this is equivalent to finding a solution of the following constrained *minimization* problem in the unknowns $(p,x,y) \in [0,1] \times [0,\infty] \times [0,\infty]$:

$$\mathtt{t}P_{\omega_0}(x,y) = \mathtt{t}Q_{\mathtt{True}}(x,y), \qquad x = -\log p, \qquad y = -\log(1-p). \tag{$\star$}$$

Since the first equation can be solved easily (i.e. in linear time) by computing the tropical roots of $\mathtt{t}Q_{\mathtt{True}}$, we can obtain an explicit relation between x and y that can be used to solve the whole system, finally finding our p, as the next example shows.

▶ **Example 6.** For our running example M, let us suppose that we observed the event "$M \twoheadrightarrow \mathtt{True}$", so that our probabilities are conditioned under this observation. We have

$$\mathtt{t}Q_{\mathtt{True}}(x,y) = \min\{\mathtt{t}P_{ll}(x,y), \mathtt{t}P_{rll}(x,y), \mathtt{t}P_{rrr}(x,y)\} = \min\{2x, y+2x, 3y\}.$$

The tropical roots of $\mathtt{t}Q_{\mathtt{True}}(x,y)$ are all the points of the form $(x, \frac{2}{3}x)$. Recall that these are the points where $(\star)$ is satisfied for *at least two* distinct values of ω_0 (indeed for $\omega_0 \in \{ll, rrr\}$). From this it follows that $\mathtt{t}Q_{\mathtt{True}}(x,y) = \mathtt{t}P_{rrr}(x,y) = 3y$ holds iff $y \leqslant \frac{2}{3}x$, and $\mathtt{t}Q_{\mathtt{True}}(x,y) = 2x = \mathtt{t}P_{ll}(x,y)$ otherwise. In this way can find the maximum likelihood estimator for $\omega_0 = rrr$: via the substitution $x := -\log p$, $y := -\log(1-p)$, the condition $y \leqslant \frac{2}{3}x$ is equivalent to $-\log(1-p) \leqslant -\frac{2}{3}\log p$, i.e. $1-p \geqslant p^{\frac{2}{3}}$. This means that, if $p \in [0,1]$ satisfies $1-p \geqslant p^{\frac{2}{3}}$ (for example, $p = \frac{1}{4}$), then $P_{rrr}(p, 1-p) = \max_{\omega=ll,rll,rrr} P_\omega(p, 1-p)$. In other words, knowing that M sampled $\mathtt{True}$ in its normal form, the most likely sampled occurrence of $\mathtt{True}$ is the one at the address rrr iff $1-p \geqslant p^{\frac{2}{3}}$.

As we'll see in Section 5, this analysis extends to PCF-style programs. For example, the program $M = \mathbf{Y}(\lambda x.\mathtt{True} \oplus_p x)$ yields the power series $Q_{\mathtt{True}}(p,q) = \sum_{n=0}^{\infty} pq^n = \frac{p}{1-q}$ that sums all *infinitely many* ways in which M may reduce to $\mathtt{True}$. Notice that the tropicalised series $\mathtt{t}Q_{\mathtt{True}}(-\log p, -\log q) = \inf_{n\in\mathbb{N}}\{-\log p - n\log q\} = -\log p$ collapses onto a single monomial describing the *unique* most likely reduction path of M leading to $\mathtt{True}$, namely the one that passes through a coin toss only once.

Best Case Analysis for Non-Deterministic Languages. This example is inspired from [50]. We consider now a basic non-deterministic language with terms $M ::= \mathtt{True} \mid \mathtt{Gen} \mid M + M$, with an operation semantics comprising a non-deterministic reduction rule $M_1 + M_2 \xrightarrow{\alpha} M_i$ and a generation rule $\mathtt{Gen} \xrightarrow{\beta} \mathtt{True} + \mathtt{Gen}$, where in each case the value $\alpha, \beta \in \mathbb{L}$ indicates a *cost* associated with the reduction (e.g. the estimated clock value for the simulation of each reduction on a given machine model). Then, any reduction $\omega : M \twoheadrightarrow N$ of a term to (one of its) normal form is associated with a tropical monomial $P_\omega(\alpha, \beta)$ consisting of

the sum of the costs of all reductions in ω. For a given normal form N, the reductions $\omega_i : M \twoheadrightarrow N$ give rise to a tps $\inf_{i\in I} P_{\omega_i}(\alpha, \beta)$. For example, consider the non-deterministic term $M := \texttt{Gen} + ((\texttt{True} + \texttt{True}) + \texttt{Gen})$. The (infinitely many) reduction paths leading to $\texttt{True}$ can be grouped as follows:

- left, then reduce $\texttt{Gen}$ $n+1$-times, then left;
- right, then left and then either left or right;
- right twice, then reduce $\texttt{Gen}$ $n+1$-times and then left.

This leads to the tps $\varphi_{M\twoheadrightarrow\texttt{True}}(\alpha) = \inf_{n\in\mathbb{N}} \{(n+2)\alpha + (n+1)\beta, 3\alpha, (n+3)\alpha + (n+1)\beta\} = \min\{2\alpha+\beta, 3\alpha\}$, which describes all possible behaviors of M. Notice that, since α and β are always positive, the power series $\varphi_{M\twoheadrightarrow\texttt{True}}(\alpha)$ is indeed equivalent to the tropical polynomial $\min\{2\alpha+\beta, 3\alpha\}$. In other words, of the infinitely many behaviors of M, only finitely many have chances to be *optimal*: either left + $\texttt{Gen}$ + left, or right + left + (left,right). Also in this case, reducing to best-case analysis leads to collapse the infinitary description of *all* behaviors to a purely combinatorial description of the finitely many optimal ones.

Once reduced $\varphi_{M\twoheadrightarrow\texttt{True}}$ to a polynomial, the best behavior among these will depend on the values of α and β, and by studying the tropical polynomial $\varphi_{M\twoheadrightarrow\texttt{True}}$ one can thus answer questions analogous to 2) above, that is, what are the best choices of costs α, β making a *chosen* reduction of M to $\texttt{True}$ the cheapest one?

5 Tropical Semantics of Higher-Order Programs

In this section we first recall a general and well-known construction that yields, for any *continuous* semiring Q, a model $Q\mathsf{Rel}_!$ of effectful extensions of the simply typed λ-calculus and PCF, and we show how, when $Q = \mathbb{L}$, it captures optimal program behavior; moreover, we discuss how this model adapts to *graded* and *differential* variants of STLC.

Linear/Non-Linear Algebra on Q-Modules. For any *continuous* semiring Q (i.e. a cpo equipped with an order-compatible semiring structure), one can define a category $Q\mathsf{Rel}$ ([50] calls it Q^{Π}) of "Q-valued matrices" as follows: $Q\mathsf{Rel}$ has sets as objects and set-indexed matrices with coefficients in Q as morphisms, i.e. $Q\mathsf{Rel}(X, Y) := Q^{X\times Y}$. The composition $st \in Q^{X\times Z}$ of $t \in Q^{X\times Y}$ and $s \in Q^{Y\times Z}$ is given by $(st)_{a,c} := \sum_{b\in Y} s_{b,c}t_{a,b}$ (observe that this series always converges because Q is continuous). For any set X, Q^X is a Q-semimodule and we can identify $Q\mathsf{Rel}(X, Y)$ with the set of linear maps from Q^X to Q^Y, which have shape $f(x)_b := \sum_{a\in X} \hat{f}_{a,b}x_a$, for some matrix $\hat{f} \in Q^{X\times Y}$. Notice that usual linear algebra conventions correspond to work in $Q\mathsf{Rel}^{\mathrm{op}}$, e.g. the usual matrix-vector product defines a map $Q^Y \to Q^X$. Following [29, 47, 50], we are instead working with transpose matrices.

$Q\mathsf{Rel}$ admits a comonad ! which acts on objects by taking the finite multisets. Remember that the coKleisli category $\mathcal{C}_!$ of a category $\mathcal{C}$ w.r.t. a comonad ! is the category whose objects are the same of $\mathcal{C}$, and $\mathcal{C}_!(X, Y) := \mathcal{C}(!X, Y)$, with composition $\circ_!$ defined via the co-multiplication of !. Now, although a matrix $t \in Q\mathsf{Rel}_!(X, Y)$ yields a linear map $\mathbb{L}^{!X} \to \mathbb{L}^Y$, by exploiting the coKleisli structure we can also "express it in the base X", which leads to the *non-linear* map $t^! : Q^X \to Q^Y$ defined by the power series $t^!(x) = t \circ_! x : b \mapsto \sum_{\mu\in!X} t_{\mu,b} \cdot x^\mu$, where $x^\mu = \prod_{a\in x} x_a^{\mu(a)}$.

When we instantiate $Q = \mathbb{L}$, we obtain the category $\mathbb{L}\mathsf{Rel}$ of $\mathbb{L}$-valued matrices. As one might expect, this category is tightly related to Lawvere's theory of (generalized) metric spaces. For the moment, let us just observe that a (possibly ∞) metric on a set X is nothing but a "$\mathbb{L}$-valued square matrix" $d : X \times X \to \mathbb{L}$ satisfying axioms like e.g. the triangular law. We will come back to this viewpoint in Section 8.

Composition in $\mathbb{L}\mathsf{Rel}$ reads as $(st)_{a,c} := \inf_{b\in Y}\{s_{b,c} + t_{a,b}\}$, and the non-linear maps $t^! : \mathbb{L}^X \to \mathbb{L}^Y$ have shape $t^!(x)_b = \inf_{\mu\in !X}\{\mu x + t_{\mu,b}\}$, where $\mu x := \sum_{a\in X} \mu(a)x_a$. These correspond to the generalisation of tps with *possibly infinitely* many variables (in fact, as many as the elements of X). By identifying $!\{*\} \simeq \mathbb{N}$ and $\mathbb{L}^{\{*\}} \simeq \mathbb{L}$, the tps generated by the morphisms in $\mathbb{L}\mathsf{Rel}_!(\{*\},\{*\})$ are exactly the usual tps's of one variable. For example, the φ of Figure 1 is indeed of shape $\varphi = t^!$, for $t \in \mathbb{L}^{!\{*\}\times\{*\}}$, $t_{\mu,*} := 2^{-\#\mu}$.

▶ Remark 7. The operation $f \mapsto f^!$ turning a matrix into a function is reminiscent of the well-known operation of taking the *convex conjugate* f^* of a function f defined over a vector space (itself a generalization of the Legendre transformation). Indeed, let X, Y be sets and let $\langle \_,\_\rangle : X \times Y \to \mathbb{R}$. For $f : X \to \mathbb{R}$, let $f^* : Y \to \mathbb{R}$ be defined by $f^*(y) := \sup_{x\in X}\{\langle x,y\rangle - f(x)\}$. Then for $X = !A$, $Y = \mathbb{L}^A$, where A is a set, and $\langle \mu, y\rangle := \mu y$, we have $f^!(y) = (-f)^*(-y)$ for all $f \in \mathbb{L}^{!A}$.

The $\mathbb{L}$-Weighted Relational Model. The categories $Q\mathsf{Rel}$ are well-known to yield a model of both probabilistic and non-deterministic versions of PCF (see e.g. [34, 50]), which are called *weighted relational models*. The interpretation of the simply typed λ-calculus STLC in $\mathbb{L}\mathsf{Rel}$ relies on the fact that all categories $Q\mathsf{Rel}_!$ are cartesian closed [50], with cartesian product and exponential objects acting on objects X, Y as, respectively, $X + Y$ and $!X \times Y$. Hence, any typable term $\Gamma \vdash M : A$ gives rise to a morphism $[\![\Gamma \vdash M : A]\!] \in \mathbb{L}\mathsf{Rel}_!([\![\Gamma]\!],[\![A]\!])$, and thus to a generalized tps $[\![\Gamma \vdash M : A]\!]^! : \mathbb{L}^{[\![\Gamma]\!]} \to \mathbb{L}^{[\![A]\!]}$.

▶ **Example 8.** The evaluation morphism $\mathsf{ev} \in \mathbb{L}\mathsf{Rel}_!((!X \times Y) + X, Y)$ yields the tps $\mathsf{ev}^! : \mathbb{L}^{!X\times Y} \times \mathbb{L}^X \to \mathbb{L}^Y$ given by $b \in Y \mapsto \mathsf{ev}^!(F,x)_b = \inf_\mu\{F_{\mu,b} + \mu x\}$. So, for instance, supposing the ground type o of STLC is interpreted as the singleton set $\{*\}$, and recalling the identification $!\{*\} \simeq \mathbb{N}$, the interpretation of the term $x : o, z : (o \to o \to o) \vdash zxx : o$, involving two consecutive evaluations, yields the tps $\varphi : \mathbb{L} \times \mathbb{L}^{\mathcal{M}_{\mathrm{fin}}(\mathbb{N}\times\mathbb{N})} \to \mathbb{L}$ given by $\varphi(x,z) = \inf_{n,n'}\{z_{[(n,n')]} + (n+n')x\}$.

This interpretation extends to PCF by interpreting the fixpoint combinator $\mathbf{Y}$ via the matrices $\mathrm{fix}^X = \inf_n \{\mathrm{fix}^X_n\} \in \mathbb{L}^{!(!X\times X)\times X}$, where $\mathrm{fix}^X_0 = 0$ and $\mathrm{fix}^X_{n+1} = \mathrm{ev} \circ_! \langle \mathrm{fix}^X_n, \mathrm{id}\rangle$.

One can easily check, by induction on a typing derivation, that for any program of STLC or PCF, the associated matrix is *discrete*, that is, its values are included in $\{0,\infty\}$. Indeed, as suggested in Section 3, the actual interest of tropical semantics lies in the interpretation of effectful programs. As the homsets $\mathbb{L}\mathsf{Rel}_!(X,Y)$ are $\mathbb{L}$-modules, it is possible to interpret in it extensions of STLC and PCF comprising $\mathbb{L}$-module operations $\alpha\cdot M$ and $M+N$ [50], by letting $[\![\Gamma \vdash \alpha\cdot M : A]\!] = [\![\Gamma \vdash M : A]\!] + \alpha$ and $[\![\Gamma \vdash M + N : A]\!] = \min\{[\![\Gamma \vdash M : A]\!],[\![\Gamma \vdash N : A]\!]\}$. More precisely, [50] considers a language PCF^Q corresponding to PCF extended with Q-module operations, with an operational semantics given by rules $M \xrightarrow{1} M'$ for each rule $M \to M'$ of PCF (here 1 is the monoidal unit of Q) as well as $M_1 + M_2 \xrightarrow{1} M_i$ and $\alpha \cdot M \xrightarrow{\alpha} M$. Hence, any reduction $\omega = \rho_1 \dots \rho_k : M \twoheadrightarrow N$ is naturally associated with a weight $\mathsf{w}(\omega) = \sum_{i=1}^k \mathsf{w}(\rho_i) \in Q$. In particular, from [Theorem V.6] [50] we deduce the following adequation result:

▶ **Proposition 9.** $[\![\vdash_{\mathrm{PCF}^{\mathbb{L}}} M : \mathrm{Nat}]\!]_n = \inf\{\mathsf{w}(\omega) \mid \omega : M \to \underline{n}\}$ *for all* $n \in \mathbb{N}$.

The previous result allows to relate the tropical semantics of a program with its best-case operational behavior. Observe that the two examples shown in Section 3 can easily be rephrased in the language $\mathrm{PCF}^{\mathbb{L}}$. For instance, for the probabilistic example, one can use the translation $(M \oplus_p N)^\circ = \min\{M^\circ + p, N^\circ + (1-p)\}$, so that the reductions $M \oplus_p N \xrightarrow{p} M$ and $M \oplus_p N \xrightarrow{1-p} N$ translate into a sequence of two reductions $(M \oplus_p N)^\circ \xrightarrow{0} M^\circ \xrightarrow{p} M$ and

$$\frac{}{x :_1 A \vdash x : A}$$

$$\frac{\Gamma \vdash M : A}{\Gamma, x :_0 B \vdash M : A} \qquad \frac{\Gamma, x :_n B, y :_m B \vdash M : A}{\Gamma, x :_{n+m} B \vdash M\{x/y\} : A}$$

$$\frac{\Gamma, x :_n A \vdash M : B}{\Gamma \vdash \lambda x.M :!_n A \multimap B} \qquad \frac{\Gamma \vdash M :!_n A \multimap B \quad \Delta \vdash N : A}{\Gamma + n\Delta \vdash MN : B}$$

Figure 2 Typing rules for bSTLC.

$(M \oplus_p N)^\circ \xrightarrow{0} N^\circ \xrightarrow{1-p} N^\circ$. Let PPCF (for *probabilistic* PCF [34]) be standard PCF extended with the constructor $M \oplus_p N$ ($p \in [0,1]$) and its associated reduction rules. From the above discussion we deduce the following:

▶ **Corollary 10.** *Let* $\vdash_{\mathrm{PPCF}} M : \mathrm{Nat}$ *and* $n \in \mathbb{N}$. *Considering its interpretation as a function of* $p, 1-p$, *we have that* $[\![\vdash_{\mathrm{PPCF}} M^\circ : \mathrm{Nat}]\!]_n(-\log(p), -\log(1-p))$ *is the minimum negative log-probability of any reduction from* M *to* $\underline{n}$, *i.e. the negative log-probability of (any of) the (equiprobable) most likely reduction path from* M *to* n.

Remark that this implies that all solution $p \in [0,1]$ to the equation $-\log \mathsf{w}(\omega) = [\![\vdash_{\mathrm{PPCF}^{\mathbb{L}}} M^\circ : \mathrm{Nat}]\!]_n(-\log(p), -\log(1-p))$ are the values of the probabilistic parameter which make the reduction ω the most likely.

▶ Remark 11. The function $[\![\vdash_{\mathrm{PPCF}} M^\circ : \mathrm{Nat}]\!]_n(\alpha, \beta)$ is a tps, and Theorem 3 ensures that this function coincides *locally* with a tropical polynomial. This means that, for any choice of $p, 1-p$, the most likely reduction path of M can be searched for within a *finite* space.

Finally, [50] obtained a similar result for a non-deterministic version of PCF, by translating each term into $\mathrm{PCF}^{\mathbb{L}}$ via $(\lambda x.M)^\circ = \lambda x.M^\circ + 1$ and $(\mathbf{Y}M)^\circ = \mathbf{Y}(M^\circ + 1)$ ([50] considers the discrete tropical semiring $\mathbb{N} \cup \{\infty\}$, but the result obviously transports to $\mathbb{L}$), and in that case [50, Corollary VI.10] gives that $[\![\vdash M^\circ : \mathrm{Nat}]\!]_n$ computes the minimum number of β- and fix- redexes reduced in a reduction sequence from M to $\underline{n}$.

$\mathbb{N}$-Graded Types. We now show how to interpret in $\mathbb{L}\mathsf{Rel}_!$ a graded version of STLC, that we call bSTLC, indeed a simplified version of the well-studied language Fuzz [68]. This language is based on a graded exponential $!_n A$, corresponding to the possibility of using an element of type A *at most* n times. In particular, if a function $\lambda x.M$ of type $!_n A \multimap B$, then, for any N of type A, x is duplicated *at most* n times in any reduction of $(\lambda x.M)N$ to the normal form.

Graded simple types are defined by $A ::= o \mid !_n A \multimap A$; the contexts of the typing judgements are sets of declarations of the form $x :_n A$, with $n \in \mathbb{N}$; given two contexts Γ, Δ, we define $\Gamma + \Delta$ recursively as follows: if Γ and Δ have no variable in common, then $\Gamma + \Delta = \Gamma \cup \Delta$; otherwise, we let $(\Gamma, x :_m A) + (\Delta, x :_n A) = (\Gamma + \Delta), x :_{m+n} A$. Moreover, for any context Γ and $m \in \mathbb{N}$, we let $m\Gamma$ be made all $x :_{mn} A$ for $(x :_n A) \in \Gamma$. The typing rules of bSTLC are illustrated in Fig. 2,

On the side of $\mathbb{L}\mathsf{Rel}$, one can check that the comonad ! of $\mathbb{L}\mathsf{Rel}$ can be "decomposed" into a family of "graded exponentials functors" $!_n : \mathbb{L}\mathsf{Rel} \to \mathbb{L}\mathsf{Rel}$ ($n \in \mathbb{N}$), where $!_n X$ is the set of multisets on X of cardinality *at most* n. The sequence $(!_n)_{n\in\mathbb{N}}$ gives rise to a so-called $\mathbb{N}$-*graded linear exponential comonad* on (the SMC) $\mathbb{L}\mathsf{Rel}$ [49]. As such, $(\mathbb{L}\mathsf{Rel}, (!_n)_{n\in\mathbb{N}})$ yields then a model of bSTLC. More details can be found in the extended version of this article.

$$\frac{}{\Gamma \vdash 0 : A} \qquad \frac{}{\Gamma, x : A \vdash x : A}$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \to B} \qquad \frac{\Gamma \vdash M : A \to B \quad \Gamma \vdash \mathbb{T} : A}{\Gamma \vdash M\mathbb{T} : B}$$

$$\frac{\Gamma \vdash M : A \to B \quad \Gamma \vdash N : A}{\Gamma \vdash \mathsf{D}[M, N] : A \to B} \qquad \frac{\Gamma \vdash M_1 : A \cdots \Gamma \vdash M_n : A}{\Gamma \vdash M_1 + \cdots + M_n : A}\ (n \geqslant 2)$$

Figure 3 Typing rules of ST∂LC.

Notice that arrow types are interpreted via $[\![!_n A \multimap B]\!] := !_n[\![A]\!] \times [\![B]\!]$ and that, whenever $[\![*]\!]$ is finite, the set $[\![A]\!]$ is finite for any type A of bSTLC.

The Differential λ-Calculus. We recall the interpretation in $\mathbb{L}\mathsf{Rel}_!$ of the simply typed *differential* λ-calculus ST∂LC, an extension of STLC ensuring exact control of duplications. The syntax of ST∂LC (see [20, Section 3]) is made of *terms* M and *sums* $\mathbb{T}$, mutually generated by: $M ::= x \mid \lambda x.M \mid M\mathbb{T} \mid \mathsf{D}[M, M]$ and $\mathbb{T} ::= 0 \mid M \mid M + \mathbb{T}$, quotiented by equations that make $+, 0$ form a commutative monoid on the set of sums, by linearity of $\lambda x.(\_)$, $\mathsf{D}[\_, \_]$ and $(\_)\mathbb{T}$ (but *not* of $M(\_)$) and by irrelevance of the order of consecutive $\mathsf{D}[\_, \_]$. We follow the tradition of quotienting also for the idempotency of $+$. The typing rules are illustrated in Figure 3, where a context Γ is a list of typed variable declarations. The main feature of this language is that $\mathsf{D}^n[\lambda x.M, N^n]0$ has a non-zero normal form iff x is duplicated exactly n times during reduction.

The categorical models of ST∂LC are called *cartesian closed differential λ-categories* (CC∂λC) [16, 17, 20]. These are CCCs enriched over commutative monoids (i.e. morphisms are summable and there is a 0 morphism), with the cartesian closed structure compatible with the additive structure, and equipped with a certain *differential operator* D, turning a morphism $f : A \to B$ into a morphism $Df : A \times A \to B$, and generalising the usual notion of differential, see e.g. [15]. An example is the CC∂λC of convenient vector spaces with smooth maps, where D is the "real" differential of smooth maps.

Applying [52, Theorem 6.1] one can check that $\mathbb{L}\mathsf{Rel}_!$ is a CC∂λC when equipped with $D : \mathbb{L}\mathsf{Rel}(!X, Y) \to \mathbb{L}\mathsf{Rel}(!(X \& X), Y)$ defined as $(Dt)_{\mu\oplus\rho,b} = t_{\rho+\mu,b}$ if $\#\mu = 1$ and as ∞ otherwise (using the isomorphism $(\mu, \rho) \in !Z \times !Z' \mapsto \mu \oplus \rho \in !(Z + Z')$). The differential operator D of $\mathbb{L}\mathsf{Rel}_!$ translates into a differential operator $D_!$ turing a tps $f : \mathbb{L}^X \to \mathbb{L}^Y$ into a tps $D_!f : \mathbb{L}^X \times \mathbb{L}^X \to \mathbb{L}^Y$, linear in its first variable, and given by $D_!f(x, y)_b = \inf_{a\in X, \mu\in !X} \left\{ \hat{f}_{\mu+a,b} + x_a + \mu y \right\}$. One can check that, when f is a tropical polynomial, $D_!f$ coincides with the standard tropical derivative (see e.g. [45]).

6 On Tropical Power Series

As seen in Section 3, tropical polynomials are piecewise-linear functions, hence concave and Lipschitz-continuous. Moreover, tps in finitely many variables are locally equivalent to tropical polynomials (except at some singular points), and are thus also concave and locally Lipschitz-continuous. In this section we show that much of these properties extend also to tps with infinitely many variables, as those arising from the tropical relational model. The literature on tropical power series is often recent (e.g. [61]), and several results we prove in this section are, to our knowledge, new.

Notice that, as a set, $\mathbb{L}^X = [0,\infty]^X$, and with the usual $+$ and $\cdot$ it is a $\mathbb{R}_{\geqslant 0}$-semimodule, let us call it $\overline{\mathbb{R}}^X_{\geqslant 0}$. Together with the usual sup-norm $\|x\|_\infty := \sup_{a\in X} x_a$, it can be showed to be a Scott-complete *normed cone* (see [71] or the extended version). Suitable categories of cones have been recently investigated as models of probabilistic computation ([23, 31, 35]). The cone structure of $\overline{\mathbb{R}}^X_{\geqslant 0}$ also induces a partial order on it, its *cone partial-order*: $x \leqslant y$ iff $y = x + z$ for some (unique) $z \in \overline{\mathbb{R}}^X_{\geqslant 0}$. It actually coincides with the pointwise order on $\overline{\mathbb{R}}_{\geqslant 0}$ (and makes it a Scott-continuous dcpo). In this section we consider tps w.r.t. this structure.

Continuity of tps. Looking at Fig. 1, we see that φ, just like the polynomials φ_n, is non-decreasing and concave. This is indeed always the case:

▶ **Proposition 12.** *All tps are non-decreasing and concave, w.r.t. the pointwise order on* $\overline{\mathbb{R}}^X_{\geqslant 0}$.

The tps φ is continuous on $\mathbb{R}_{\geqslant 0}$ (w.r.t. the usual norm of real numbers). We can generalise this property, dropping the case of x having some 0 coordinate. But we have to be careful, because while in the finite dimensional $\mathbb{R}^n$, every real convex function is continuous because it is necessarily locally bounded from above (the sup-norm and the euclidean one are equivalent) [22, Proposition 4.7], in infinite dimensions the former condition is no longer true [22, Example 4.8]. However, [22, Proposition 4.4.(3)] shows that it is the only requirement to ask: if a real-valued convex function with domain a convex open subset of a locally convex topological $\mathbb{R}$-vector space (LCTVS) is, locally around any point, bounded from above by a finite non-zero constant, then it is continuous on all its domain.

▶ **Theorem 13.** *All tps* $f : \overline{\mathbb{R}}^X_{\geqslant 0} \to \overline{\mathbb{R}}_{\geqslant 0}$ *are continuous on* $(0,\infty)^X$*, w.r.t. to the norm* $\|\cdot\|_\infty$.

Proof. By Proposition 12, $-f$ is convex. Since $f \geqslant 0$ on all $\overline{\mathbb{R}}^X_{\geqslant 0}$, we have e.g. $-f \leqslant 1$ on $\overline{\mathbb{R}}^X_{\geqslant 0}$. Now $(0,\infty)^X \subseteq \mathbb{R}^X$ is open and convex, so [22, Proposition 4.4.(3)] entails the continuity of $-f$ on $(0,\infty)^X$, hence that of f on it. ◀

In analogy with [27, Proposition 17], we also have:

▶ **Theorem 14.** *All monotone (w.r.t. pointwise order) and* $\|\cdot\|_\infty$*-continuous functions* $f : (0,\infty)^X \to (0,\infty)$ *are Scott-continuous. In particular, all tps* $f : \overline{\mathbb{R}}^X_{\geqslant 0} \to \overline{\mathbb{R}}_{\geqslant 0}$ *are Scott-continuous on* $(0,\infty)^X$ *w.r.t. the pointwise orders.*

Lipschitz-continuity of tps. Let us first look at what happens with those tps which are either *linear* or obtained via bounded exponentials. The result below is in analogy with what happens in the usual metric semantics of Fuzz, where linear functions are non-expansive and n-bounded functions are n-Lipschitz [68].

▶ **Proposition 15.**

1. *If a tps* $f : \overline{\mathbb{R}}^X_{\geqslant 0} \to \overline{\mathbb{R}}^Y_{\geqslant 0}$ *arises from a matrix* $\hat{f} : X \times Y \to \overline{\mathbb{R}}_{\geqslant 0}$ *(i.e. it is linear), then* f *is non-expansive (i.e.* 1*-Lipschitz).*
2. *If* $f : \overline{\mathbb{R}}^X_{\geqslant 0} \to \overline{\mathbb{R}}^Y_{\geqslant 0}$ *arises from a matrix* $\hat{f} : !_n X \times Y \to \overline{\mathbb{R}}_{\geqslant 0}$*, then* f *is* n*-Lipschitz-continuous.*

Proof sketch.

1). Using the fact that $f(x)_b = \inf_{a\in X}\{\hat{f}_{a,b} + x_a\}$, the problem reduces to: $|(\hat{f}_{a,b} - x_a) - (\hat{f}_{a,b} - y_a)| = |x_a - y_a| \leqslant \|x - y\|_\infty$.

2). Follows from 1. and the remark that, for all $x \in \mathbb{L}^X$, $\|!_n x - !_n y\|_\infty \leqslant n \cdot \|x - y\|_\infty$, where $!_n x$ is the restriction of $!x$ to $\mathcal{M}_{\leqslant n}(X)$. ◀

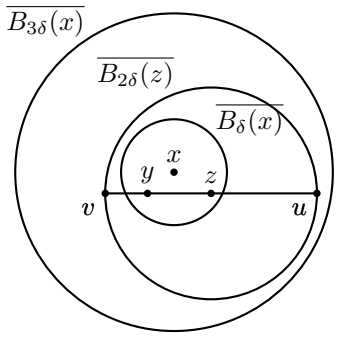


Figure 4 Drawing of the proof of Theorem 18.

Observe that on the hom-sets $\mathbb{L}\mathsf{Rel}_!(X,Y)$ there are two natural notions of distance: the metric $\|f-g\|_\infty$ arising from the norm and the one arising from the usual sup-metric $d_\infty(f,g) := \sup_{x\in\mathbb{L}^X} \|f^!(x) - g^!(x)\|_\infty$. In general one has $\|f-g\|_\infty \geqslant d_\infty(f,g)$, the equality holding when $f^!, g^!$ are linear (i.e. when they arise from morphisms of $\mathbb{L}\mathsf{Rel}(X,Y)$).

When a tps is expressed as an inf of finitely many monomials we simply call it a tropical polynomial. For any such tropical polynomial $\varphi : \overline{\mathbb{R}}_{\geqslant 0}^X \to \overline{\mathbb{R}}_{\geqslant 0}^Y$, the associated matrix has shape $!_{\deg(\varphi)}(X) \times Y \to \overline{\mathbb{R}}_{\geqslant 0}$ (as a monomial $\mu_i x + c_i$ yields a matrix entry on $!_{\#\mu_i} X \times Y$). Hence, using Proposition 15 2., we have:

▶ **Corollary 16.** *Any tropical polynomial $\varphi : \overline{\mathbb{R}}_{\geqslant 0}^X \to \overline{\mathbb{R}}_{\geqslant 0}$ is $\deg(\varphi)$-Lipschitz continuous.*

We now show that, if we consider the *full* exponential !, i.e. arbitrary tps, we can still prove that a local Lipschitz condition holds. In [22, Theorem 6.4] a locally Lipschitz property is obtained for locally convex topological vector spaces, under the hypothesis of continuity. [22, Proposition 6.5] shows that continuity is used in order to have a locally bounded condition, the crucial ingredient of the proof. Instead of showing how our case fits into such theorems, we prefer to state the following theorem, basically a particular case of [22, Theorem 6.9, Lemma 6.10]:

▶ **Theorem 17.** *All tps $f : \overline{\mathbb{R}}_{\geqslant 0}^X \to \overline{\mathbb{R}}_{\geqslant 0}$ are locally Lipschitz on $(0,\infty)^X$. Moreover, the Lipschitz constant of f on $\overline{B_\delta(x)}$ can be chosen to be $\frac{1}{\delta}\max_{\overline{B_{3\delta}(x)}} f$.*

The fundamental ingredient of the proof of this result is the following Theorem 18, from which Theorem 17 immediately follows, since $(0,\infty)^X$ is open and convex in $(\mathbb{R}^X, \|\cdot\|)$ and all tps are non-negative.

▶ **Theorem 18.** *Let $f : V \subseteq (\mathbb{R}^X, \|\cdot\|) \to (\mathbb{R}, |\cdot|)$, with V open and convex and $\|\cdot\|$ any norm. If f is concave and locally bounded, then f is locally Lipschitz. Moreover, the Lipschitz constant of f on $\overline{B_\delta(x)}$ can be chosen to be $\frac{1}{\delta}\max_{\overline{B_{3\delta}(x)}} |f|$.*

Proof. Call $\overline{B_\delta(x)} := B_1$, $\overline{B_{3\delta}(x)} := B_3$. It suffices to show that for all $x \in V$, there is $\delta > 0$ s.t. $B_3 \subseteq \text{interior}(V)$, $K := \max_{B_3} |f|$ exists and f is $\frac{K}{\delta}$-Lipschitz on B_1. A δ satisfying the first two conditions exists since V is open and because f is locally bounded and B_3 is compact. We show that the third condition holds for all such δ. For that, fix $y, z \in B_1$ and call $r := \frac{d(y,z)}{2\delta} \in [0,1]$. We want to show that $|f(y) - f(z)| \leqslant \frac{K}{\delta} d(y,z) = 2Kr$. Wlog $y \neq z$, otherwise there is nothing to prove.

So $r \neq 0$ and we can consider $u := \frac{1+r}{r} z - \frac{1}{r} y$, $v := \frac{1}{r} y - \frac{r-1}{r} z$. We have $u, v \in \overline{B_{2\delta}(z)} =: B_2$. Indeed, $d(u,z) = \|u - z\| = \|\frac{z}{r} + z - \frac{y}{r} - z\| = \frac{\|z-y\|}{r} = 2\delta$ and similarly $d(v,z) = 2\delta$. Geometrically, those are actually the intersections between B_2 and the line generated by y and

z, see Figure 4. Now we have the convex combinations $z = \frac{1}{1+r}y + \frac{r}{1+r}u$ and $y = (1-r)z + rv$, so the concavity of f entails on one hand: $f(z) \geqslant \frac{1}{1+r}f(y) + \frac{r}{1+r}f(u) \geqslant \frac{f(y)}{1+r} - \frac{rK}{1+r}$, i.e. $f(y) - f(z) \leqslant r(K + f(z)) \leqslant 2rK$, and on the other hand: $f(y) \geqslant (1-r)f(z) + rf(v) \geqslant f(z) - r(f(z) + K)$, i.e. $f(z) - f(y) \leqslant r(f(z) + K) \leqslant 2rK$. In the previous inequalities we have used that $f(u), f(v) \geqslant -K$. This follows from the fact that $u, v \in B_2 \subseteq B_3$, as it can be immediately checked, thus $|f(u)|, |f(v)| \leqslant K$. Putting the final inequalities together, we have $|f(y) - f(z)| \leqslant 2rK$, i.e. the thesis. ◀

7 Lipschitz Meets Taylor

In this section we finally relate the metric and differential analysis of higher-order programs in the tropical relational model.

The key ingredient is the notion of Taylor expansion $\mathcal{T}(M)$ of a λ-term M. This is a set of terms of the differential λ-calculus defined inductively as: $\mathcal{T}(x) = \{x\}$, $\mathcal{T}(\lambda x.M) = \{\lambda x.t \mid t \in \mathcal{T}(M)\}$ and $\mathcal{T}(MN) = \{t \cdot \langle u_1, \dots, u_k\rangle \mid k \in \mathbb{N}, t \in \mathcal{T}(M), u_i \in \mathcal{T}(N)\}$, where $t \cdot \langle u_1, \dots, u_k\rangle$ is an abbreviation for $\mathsf{D}^k[t, u_1, \dots, u_k]0$. Observe that in the terms appearing in $\mathcal{T}(M)$ all applications are *bounded*: they may use an exact number of copies of their input. Such terms are usually called *resource λ-terms* [56, 65]. One can easily check that for all terms $\Gamma \vdash_{\mathrm{STLC}} M : A$ and $t \in \mathcal{T}(M)$, also $\Gamma \vdash_{\mathrm{ST}\partial\mathrm{LC}} t : A$ holds.

▶ **Example 19.** Considering the term $M = zxx$ from Example 8, all terms $t_{n,m} = z\langle x^n\rangle\langle x^m\rangle$, for $n, m \in \mathbb{N}$, are in $\mathcal{T}(M)$. Notice that the interpretation of $t_{n,m}$ yields a tropical polynomial $[\![t_{n,m}]\!]^!(x)(z) = y_{[n,m]} + (n+m)x$, rather than a tps. However, this is not a general fact: consider $y : (o \to o) \to (o \to o), x : (o \to o) \vdash t : (o \to o)$ with $t = y \cdot \langle y \cdot \langle x\rangle\rangle \in \mathcal{T}(y(yx))$. Then $[\![t]\!]^! : \mathbb{L}^{!\mathbb{N}\times\mathbb{N}} \times \mathbb{L}^{\mathbb{N}} \to \mathbb{L}^{\mathbb{N}}$ is given by $[\![t]\!]^!(y, x)_i = \inf_{m,n\in\mathbb{N}} \{y_{[m],i} + y_{[n],m} + x_n\}$, which is not a polynomial. Yet, $[\![t]\!]^!$ is Lipschitz, more precisely, 1-Lipschitz in x and 2-Lipschitz in y. This is a general fact, as shown by Theorem 22 below.

We have already shown that the tropical differential makes $\mathbb{L}\mathsf{Rel}_!$ a model of the differential λ-calculus. We now show that it also models the Taylor expansion (this needs not be true for *any* CC$\partial\lambda$C). First, it can be patiently checked that (see [56, Definition 4.22]):

▶ **Theorem 20.** *Morphisms in $(\mathbb{L}\mathsf{Rel}_!, D)$ can be Taylor-expanded: for all $t \in \mathbb{L}\mathsf{Rel}_!(Z, !X \multimap Y)$, $s \in \mathbb{L}\mathsf{Rel}_!(Z, X)$ we have* $\mathrm{ev} \circ_! \langle t, s\rangle = \inf_{n\in\mathbb{N}} \{((\dots((\Lambda^- t) \star s) \star \dots) \star s) \circ_! \langle \mathrm{id}, \infty\rangle\}$.

The equation above is a tropical reformulation of the Taylor formula from the Introduction: $u \star s = (Du) \circ_! \langle\langle \infty, s \circ_! \pi_1\rangle, \mathrm{id}\rangle$ corresponds to the application of the derivative of u on s, and Λ^- is the uncurry operator. Hence the right-hand term corresponds to the inf of the n-th derivative of $\Lambda^- t$ applied to "n copies" of s.

Second, since $\mathbb{L}\mathsf{Rel}_!$ has countable sums (all countable infs converge), an immediate adaptation of the proof of [56, Theorem 4.23] shows:

▶ **Corollary 21.** $[\![\Gamma \vdash_{\mathrm{STLC}} M : A]\!] = \inf_{t\in\mathcal{T}(M)} [\![\Gamma \vdash_{\mathrm{ST}\partial\mathrm{LC}} t : A]\!]$.

Using the results of the previous section, as well as the results above, we now deduce the following properties:

▶ **Theorem 22.** *Let $\mathcal{S}$ be one of $\mathrm{PCF}^{\mathbb{L}}, \mathrm{STLC}, \mathsf{bSTLC}, \mathrm{ST}\partial\mathrm{LC}$. Let $\Gamma \vdash_{\mathcal{S}} M : A$ and $a \in [\![A]\!]$.*

1. *For $\mathcal{S} = \mathsf{bSTLC}$, $[\![\Gamma \vdash_{\mathcal{S}} M : A]\!]^!_a$ is a tropical polynomial, and thus Lipschitz;*
2. *For $\mathcal{S} = \mathrm{ST}\partial\mathrm{LC}$, if $t \in \mathcal{T}(M)$, then $[\![\Gamma \vdash_{\mathcal{S}} t : A]\!]^!_a$ is Lipschitz;*
3. *For $\mathcal{S} = \mathrm{STLC}, \mathrm{PCF}^{\mathbb{L}}$, then $[\![\Gamma \vdash_{\mathcal{S}} M : A]\!]^!_a$ is locally Lipschitz;*
4. *For $\mathcal{S} = \mathrm{STLC}$, $\mathcal{T}(M)$ decomposes $[\![\Gamma \vdash_{\mathrm{STLC}} M : A]\!]^!_a$ as an $\inf_{t\in\mathcal{T}(M)} [\![\Gamma \vdash_{\mathrm{ST}\partial\mathrm{LC}} t : A]\!]^!_a$ of Lipschitz functions.*

Proof.
1). From Proposition 15 2. and the remark that for any type of bSTLC, $[\![A]\!]$ is finite.
2). From Proposition 15 2. observing that a resource term $t(x)$ may use a variable x a fixed number n times, so that its matrix lies in $\mathbb{L}^{!_n X \times Y}$.
3). From Theorem 17.
4). It follows from Corollary 21 plus the fact that, for $(f_n)_{n\in\mathbb{N}} \subseteq \mathbb{L}^{!X\times Y}$, we have $(\inf_{n\in\mathbb{N}} f_n)^! = \inf_{n\in\mathbb{N}} f_n^!$. ◀

We conclude our discussion with an application of the Taylor expansion in $\mathbb{L}\mathsf{Rel}_!$: as proved in the previous section, all tps are locally Lipschitz; now, Theorem 22 can be used to compute approximations of the Lipschitz constants of an actual higher-order program.

▶ **Corollary 23.** *Suppose $x : A \vdash_{\mathrm{STLC}} M : B$ and $\vdash_{\mathrm{STLC}} N : A$. Then for all $t \in \mathcal{T}(M)$ such that $[\![t]\!]^!([\![N]\!]) \neq \infty$, and $\delta > 0$, the tps $[\![x : A \vdash_{\mathrm{STLC}} M : B]\!]^!$ is $\frac{[\![t]\!]^!([\![N]\!]+3\delta)}{\delta}$-Lipschitz over the open ball $B_\delta([\![N]\!])$.*

Proof. Thm. 17 yields the estimate $\max_{\overline{B_{3\delta}([\![N]\!])}} [\![M]\!]^!$. As from Thm. 22 4. it follows that $[\![t]\!]^! \geqslant [\![M]\!]^!$, we deduce that $K = \max_{\overline{B_{3\delta}([\![N]\!])}} [\![t]\!]^! \geqslant \max_{\overline{B_{3\delta}([\![N]\!])}} [\![M]\!]^!$ is also a local Lipschitz constant for $[\![M]\!]^!$. Moreover, since $[\![t]\!]^!$ is concave and non-decreasing, the max of $[\![t]\!]^!$ is attained at the maximum point of $\overline{B_{3\delta}([\![N]\!])}$, that is, $K = [\![t]\!]^!(x + 3\delta)$. Finally, from $[\![t]\!]^!([\![N]\!]) < \infty$ and the continuity of $[\![t]\!]^!$ we deduce $K < \infty$. ◀

▶ **Example 24.** Consider again the term $M = zxx$ from Example 8. The (generalized) tps $[\![M]\!]^!(x)(y) = \inf_{n,n'\in\mathbb{N}}\{y_{[(n,n')]} + (n+n')x\}$ is not (globally) Lipschitz: for any $L > 0$, choose a natural number $N > L$, let $Y \in \mathbb{L}^{\mathcal{M}_{\mathrm{fin}}(\mathbb{N}\times\mathbb{N})}$ be such that $Y_\mu < \infty$ only if $\mu = [(n, n')]$ with $n + n' \geqslant N$; then $|[\![M]\!]^!(x)(Y) - [\![M]\!]^!(x + \epsilon)(Y)| \geqslant N\epsilon > L\epsilon$. Now take the approximant $t = z\langle x^{N-1}\rangle\langle x\rangle \in \mathcal{T}(M)$ (chosen so that $[\![t]\!]^!(x)(Y) < \infty$). Its interpretation is the monomial $[\![t]\!]^!(x)(Y) = Y_{[(N-1,1)]} + Nx$. We can then compute a Lipschitz-constant for $[\![M]\!]^!$ around $\langle x, Y\rangle$ as $\frac{1}{\delta}[\![t]\!]^!(\langle x, Y\rangle + \delta) = 3N + 3 + \frac{Y_{[(N-1,1)]}+Nx}{\delta}$.

8 Generalized Metric Spaces and $\mathbb{L}$-Modules

As we have seen, the morphisms of $\mathbb{L}\mathsf{Rel}$ can be seen as continuous functions between the $\mathbb{L}$-modules $\mathbb{L}^X$, when the latter are taken with the metric induced by the ∞-norm. This viewpoint gives a metric flavor to $\mathbb{L}\mathsf{Rel}$, and allowed us to relate differential and metric structure. Yet, how far can this correspondence be pushed? In particular, is this correspondence restricted to $\mathbb{L}$-modules of the form $\mathbb{L}^X$ (i.e. with a fixed base), or does it hold in some sense for arbitrary $\mathbb{L}$-modules? Is this correspondence restricted to the ∞-norm metric, or does it hold for other metrics too?

8.1 $\mathbb{L}$-Modules and Cocomplete $\mathbb{L}$-Categories

An answer to the questions above comes from an elegant categorical correspondence between tropical linear algebra and the theory of *generalized metric spaces*, initiated by Lawvere's pioneering work [51], and at the heart of the emergent field of *monoidal topology* [47, 74].

On the one hand we have $\mathbb{L}$-modules: these are triples $(M, \leq, \star)$ where $(M, \leq)$ is a sup-lattice, and $\star : \mathbb{L} \times M \to M$ is a continuous (left-)action of $\mathbb{L}$ on it, where continuous means that $\star$ commutes with both joins in $\mathbb{L}$ and in M. A $\mathbb{L}$-module homomorphism is a map $f : M \to N$ commuting with both joins and the $\mathbb{L}$-action. We let $\mathbb{L}\mathsf{Mod}$ indicate the category of $\mathbb{L}$-modules and their homomorphisms.

On the other hand we have Lawvere's *generalized metric spaces* [47, 51, 74]: Lawvere was the first to observe that a metric space can be described as a $\mathbb{L}$-*enriched* category. Indeed, spelling out the definition, a $\mathbb{L}$-enriched category (in short, a $\mathbb{L}$-category) is given by a set X together with a "hom-set" $X(-,-) : X \times X \to \mathbb{L}$ satisfying $0 \geqslant X(x,x)$ and $X(y,z) + X(x,y) \geqslant X(x,z)$. This structure clearly generalizes the usual definition of metric spaces, which are indeed precisely the $\mathbb{L}$-categories which are *skeletal* (i.e. $X(x,y) = 0$ implies $x = y$) and *symmetric* (i.e. $X(x,y) = X^{\mathrm{op}}(x,y)$, where $X^{\mathrm{op}}(x,y) = X(y,x)$). A basic example of $\mathbb{L}$-category is $\mathbb{L}$ itself, with the distance $\mathbb{L}(x,y) = y \dot{-} x$.

Moreover, a $\mathbb{L}$-enriched *functor* between $\mathbb{L}$-categories is nothing but a non-expansive map $f : X \to Y$, since functoriality reads as $Y(f(x), f(y)) \leqslant X(x,y)$. Functors of shape $\Phi : Y^{\mathrm{op}} \times X \to \mathbb{L}$ are called *distributors* and noted $\Phi : X \nrightarrow Y$. Two distributors $\Phi : Y \nrightarrow X$ and $\Psi : Z \nrightarrow Y$ can be composed just like ordinary matrices in $\mathbb{L}\mathsf{Rel}$: $\Psi \diamond \Phi : Z \nrightarrow X$ is given by $(\Psi \diamond \Phi)_{x,z} = \inf_{y \in Y} \{\Psi(y,z) + \Phi(x,y)\}$.

Lawvere also observed that the usual notion of Cauchy-completeness can be formulated, in this framework, as the existence of suitable colimits [51]. Let us recall the notion of weighted colimit in this context:

▶ **Definition 25** (weighted colimits). *Let X, Y, Z be $\mathbb{L}$-categories, $\Phi : Z \nrightarrow Y$ be a distributor and $f : Y \to X$ be a functor. A functor $g : Z \to X$ is the* Φ-weighted colimit of f over X, *noted* $\mathrm{colim}(\Phi, f)$, *if for all $z \in Z$ and $x \in X$*

$$X(g(z), x) = \sup_{y \in Y} \{X(f(y), x) \dot{-} \Phi(y, z)\}$$

A functor $f : X \to Y$ is said *cocontinuous* if it commutes with all existing weighted colimits in X, i.e. $f(\mathrm{colim}(\Phi, g)) = \mathrm{colim}(\Phi, f \circ g)$. A $\mathbb{L}$-enriched category. A $\mathbb{L}$-category X is said *cocomplete* if all weighted colimits over X exist. We let $\mathbb{L}\mathsf{CCat}$ indicate the category of cocomplete $\mathbb{L}$-categories and cocontinuous $\mathbb{L}$-enriched functors as morphisms.

Observe that cocompleteness for a symmetric $\mathbb{L}$-category X implies the usual Cauchy completeness. Indeed, a Cauchy sequence $(x_n)_{n \in \mathbb{N}}$ in X yields two adjoint distributors $\phi^* : 1 \nrightarrow X$ and $\phi_* : X \nrightarrow 1$, where $\phi^*(x') = \lim_{n \to \infty} X(x', x_n,)$ and $\phi_*(x') = \lim_{n \to \infty} X(x_n, x')$. Hence, $\mathrm{colim}(\phi^*, 1_X) : 1 \to X$ must be a point x satisfying $0 = X(x,x) = \sup_{y \in X} \lim_{n \to \infty} (X(y,x) \dot{-} X(y, x_n))$, which implies $\lim_{n \to \infty} X(x_n, x) = 0$.

It turns out that the notions of $\mathbb{L}$-module and cocomplete $\mathbb{L}$-category are indeed equivalent. More precisely, the categories $\mathbb{L}\mathsf{Mod}$ and $\mathbb{L}\mathsf{CCat}$ are isomorphic [73]. First, any $\mathbb{L}$-module $(M, \leq, \star)$ can be endowed with the structure of a $\mathbb{L}$-category by letting $M(x,y) = \inf \{\epsilon \mid \epsilon \star x \geqslant y\}$. Moreover, a homomorphism of $\mathbb{L}$-modules induces a cocontinuous functor of the associated $\mathbb{L}$-categories. Conversely, in cocomplete $\mathbb{L}$-categories it is possible to introduce a continuous $\mathbb{L}$-action via suitable weighted colimits called *tensors*:

▶ **Definition 26** (tensors, cf. [74]). *Let X be a $\mathbb{L}$-category, $x \in X$ and $\epsilon \in \mathbb{L}$. The* tensor of x and ϵ, *if it exists, is the colimit $\epsilon \otimes x := \mathrm{colim}([\epsilon], \Delta x)$, where $[\epsilon] : 1 \nrightarrow 1$ is the constantly ϵ distributor and $\Delta x : 1 \to X$ is the constant functor.*

A cocomplete $\mathbb{L}$-category can thus be endowed with a $\mathbb{L}$-module structure with order given by $x \leq_X y$ iff $X(y,x) = 0$, and action given by tensors $\epsilon \otimes x$. Moreover, a cocontinuous functor between cocomplete $\mathbb{L}$-categories is the same as a homomorphism of the associated $\mathbb{L}$-modules.

8.2 Exponential and Differential Structure of $\mathbb{L}\mathsf{Mod} \simeq \mathbb{L}\mathsf{CCat}$

We now show that the correspondence between $\mathbb{L}$-modules and cocomplete $\mathbb{L}$-categories lifts to a model of the differential λ-calculus, generalizing the co-Kleisli category $\mathbb{L}\mathsf{Rel}_!$.

In order to define a Lafont exponential ! over $\mathbb{L}\mathsf{Mod}$, we exploit a well-known recipe from [50, 60]. The first step is to define a symmetric algebra $\mathrm{Sym}_n(M)$ as the equalizer of all permutative actions on n-tensors $M \otimes \cdots \otimes M$. Notice that each element of $!_n M$ can be described as a join of "multisets" $[x_1, \ldots, x_n]$, where the latter is the equivalence class of the tensors $x_1 \otimes \cdots \otimes x_n \in M^{\otimes_n}$ under the action of permutations $\sigma \in \mathfrak{S}_n$. The $\mathbb{L}$-module $!_n M$ is a cocomplete $\mathbb{L}$-category with distance function defined on basic "multisets" as follows:

$$(!_n M)(\alpha, \beta) = \sup_{\sigma \in \mathfrak{S}_n} \inf_{\tau \in \mathfrak{S}_n} \sum_{i=1}^{n} X(x_{\sigma(i)}, y_{\tau(i)}) \tag{1}$$

where $\alpha = [x_1, \ldots, x_n]$ and $\beta = [y_1, \ldots, y_n]$, and extended to arbitrary elements $\alpha = \bigvee_i \alpha_i$ and $\beta = \bigvee_j \beta_j$ by $(!_n M)(\alpha, \beta) = \sup_i \inf_j (!_n M)(\alpha_i, \beta_j)$.

Next, we define $!M$ as the infinite biproduct $\prod_n !_n M$, yielding the cofree commutative comonoid over M (cf. [60, Proposition 1]). Using the fact that biproducts commute with tensors in $\mathbb{L}\mathsf{Rel}$, by standard results [60], we obtain that the coKleisli category $\mathbb{L}\mathsf{Mod}_!$ is a CCC. Moreover, the constructions for $\mathbb{L}\mathsf{Mod}$ generalize those of $\mathbb{L}\mathsf{Rel}$, in the sense that $!(\mathbb{L}^X) \simeq \mathbb{L}^{\mathcal{M}_{\mathrm{fin}}(X)}$ and that $\mathbb{L}\mathsf{Mod}_!(\mathbb{L}^X, \mathbb{L}^Y) \simeq \mathbb{L}\mathsf{Rel}_!(X, Y)$.

Finally, since the coKleisli category of a Lafont category with biproducts is always a CC∂C [53, Theorem 21], we can endow $\mathbb{L}\mathsf{Mod}_!$ with a differential operator E, generalizing $D^!$, given by

$$Ef(\alpha) = \bigvee \left\{ f(\beta \cup [x]) \;\middle|\; \iota_n(\beta) \otimes \iota_1(x) \leqslant S(\alpha) \right\} \tag{2}$$

where $\iota_k : M_k \to \prod_{i \in I} M_i$ is the injection morphism given by $\iota_k(x)(k) = x$ and $\iota_k(x)(i \neq k) = \infty$, and $S : !(M \times N) \to !M \otimes !N$ is the Seely isomorphism [60], and conclude that:

▶ **Theorem 27.** *($\mathbb{L}\mathsf{Mod}_!/\mathbb{L}\mathsf{CCat}_!, E$) is a CC$\partial$C.*

9 Related Work

The applications of tropical mathematics in computer science abound, e.g. in automata theory [48, 72], machine learning [57, 64, 77], optimization [4, 5], and convex analysis [54].

As we said, the relational semantics over the tropical semiring was quickly explored in [50], to provide a "best case" resource analysis of a PCF-like language with non-deterministic choice. The connections between differential λ-calculus (and differential linear logic), relational semantics, and non-idempotent intersection types are very well-studied (see [28], and more recently, [59] for a more abstract perspective, and [40, 63] for a 2-categorical, or proof-relevant, extension). *Probabilistic coherent spaces* [33], a variant of the relational semantics, provide an interpretation of higher-order probabilistic programs as analytic functions. In [32] it was observed that such functions satisfy a local Lipschitz condition somehow reminiscent of our examples in Section 3.

The study of linear or bounded type systems for sensitivity analysis was initiated in [44] and later developed [26, 68, 70]. Related approaches, although not based on metrics, are provided by *differential logical relations* [24] and *change action* models [7]. More generally, the literature on program metrics in denotational semantics is vast. Since at least [75] metric spaces, also in Lawvere's generalized sense [51], have been exploited as an alternative framework to standard, domain-theoretic, denotational semantics; notably models of STLC and PCF based on *ultra*-metrics and *partial* metrics have been proposed [25, 37, 66].

Motivated by connections with computer science and fuzzy set-theory, the abstract study of generalized metric spaces in the framework of *quantale*- or even *quantaloid*-enriched categories has led to a significant literature in recent years (e.g. [47, 74]), and connections with tropical mathematics also have been explored e.g. in [38, 76]. Moreover, applications of quantale-modules to both logic and computer science have also been studied [1, 69].

Finally, connections between program metrics and the differential λ-calculus have been already suggested in [66]; moreover, *cartesian difference categories* [6] have been proposed as a way to relate derivatives in differential categories with those found in change action models.

10 Conclusion and Future Work

The main goals of this paper are two. Firstly, to demonstrate the existence of a conceptual bridge between two well-studied quantitative approaches to higher-order programs, and to highlight the possibility of transferring results and techniques from one approach to the other. Secondly, to suggest that tropical mathematics, a field which has been largely and successfully applied in computer science, could be used for the quantitative analysis of functional programming languages. While the first goal was here developed in detail, and at different levels of abstraction, for the second goal we only sketched a few interesting directions, and we leave their development to a second paper of this series.

While the main ideas of this article only use basic concepts from the toolbox of tropical mathematics, an exciting direction is that of looking at potential applications of more advanced tools from tropical algebraic and differential geometry (e.g. Newton polytopes, tropical varieties, tropical differential equations). Another interesting question is how much of our results on tps and their tropical Taylor expansion can be extended to the abstract setting of generalized metric spaces and continuous functors.

References

1 Samson Abramsky and Steven Vickers. Quantales, observational logic and process semantics. *Mathematical Structures in Computer Science*, 3(2):161–227, 1993. doi:10.1017/S0960129500000189.

2 Beniamino Accattoli, Ugo Dal Lago, and Gabriele Vanoni. The (in)efficiency of interaction. In *Proceedings POPL 2021*, volume 5, New York, NY, USA, 2021. Association for Computing Machinery.

3 Beniamino Accattoli, Ugo Dal Lago, and Gabriele Vanoni. Multi types and reasonable space. *Proceedings ICFP 2022*, 6, 2022.

4 Marianne Akian, Stéphane Gaubert, and Alexander Guterman. Tropical polyhedra are equivalent to mean payoff games. *International Journal of Algebra and Computation*, 22(01):1250001, 2023/01/16 2012. doi:10.1142/S0218196711006674.

5 Marianne Akian, Stéphane Gaubert, Viorel Niţică, and Ivan Singer. Best approximation in max-plus semimodules. *Linear Algebra and its Applications*, 435(12):3261–3296, 2011. doi:10.1016/j.laa.2011.06.009.

6 Mario Alvarez-Picallo and Jean-Simon Pacaud Lemay. Cartesian difference categories. In Jean Goubault-Larrecq and Barbara König, editors, *Proceedings FoSSaCS 2020*, pages 57–76, Cham, 2020. Springer International Publishing.

7 Mario Alvarez-Picallo and C.-H. Luke Ong. Change actions: Models of generalised differentiation. In Mikołaj Bojańczyk and Alex Simpson, editors, *Proceedings FoSSaCS 2019*, pages 45–61, Cham, 2019. Springer International Publishing.

8 Mário S. Alvim, Miguel E. Andrés, Konstantinos Chatzikokolakis, Pierpaolo Degano, and Catuscia Palamidessi. Differential privacy: On the trade-off between utility and information leakage. In *Proceedings FAST 2011*, FAST–11, pages 39–54, Berlin, Heidelberg, 2011. Springer-Verlag. doi:10.1007/978-3-642-29420-4_3.

9 Melissa Antonelli, Ugo Dal Lago, and Paolo Pistone. Curry and Howard Meet Borel. In *Proceedings LICS 2022*, pages 1–13,. IEEE Computer Society, 2022.

10 Arthur Azevedo de Amorim, Marco Gaboardi, Justin Hsu, Shin-ya Katsumata, and Ikram Cherigui. A semantic account of metric preservation. In *Proceedings POPL 2017*, pages 545–556, New York, NY, USA, 2017. Association for Computing Machinery. doi:10.1145/3009837.3009890.

11 Marco Azevedo de Amorim, Gaboardi, Arthur, Justin Hsu, and Shin-ya Katsumata. Probabilistic relational reasoning via metrics. In *Proceedings LICS 2019*. IEEE Computer Society, 2019.

12 Paolo Baldan, Filippo Bonchi, Henning Kerstan, and Barbara König. Coalgebraic behavioral metrics. *Logical Methods in Computer Science*, 14(3), 2018. doi:10.23638/LMCS-14(3:20)2018.

13 Gilles Barthe, Boris Köpf, Federico Olmedo, and Santiago Zanella Béguelin. Probabilistic relational reasoning for differential privacy. In *Proceedings POPL 2012*. ACM Press, 2012. doi:10.1145/2103656.2103670.

14 Leonard E. Baum and Ted Petrie. Statistical inference for probabilistic functions of finite state markov chains. *The Annals of Mathematical Statistics*, 37(6):1554–1563, 2023/01/17/1966. URL: http://www.jstor.org/stable/2238772.

15 Richard Blute, Thomas Ehrhard, and Christine Tasson. A convenient differential category. *Cahiers de Topologie et Géométrie DIfférentielle Catégorique*, LIII:211–232, 2012. arXiv:1006.3140.

16 Richard F. Blute, Robin Cockett, J.S.P. Lemay, and R.A.G. Seely. Differential categories revisited. *Applied Categorical Structures*, 28:171–235, 2020.

17 Richard F. Blute, Robin Cockett, and R.A.G. Seely. Cartesian Differential Categories. *Theory and Applications of Categories*, 22(23):622–672, 2009.

18 Gérard Boudol. The lambda-calculus with multiplicities. In Eike Best, editor, *Proceedings CONCUR'93*, pages 1–6, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

19 Flavien Breuvart and Ugo Dal Lago. On intersection types and probabilistic lambda calculi. In *Proceedings PPDP 2018*, PPDP '18, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3236950.3236968.

20 Antonio Bucciarelli, Thomas Ehrhard, and Giulio Manzonetto. Categorical models for simply typed resource calculi. In *Proceedings MFPS 2010*, volume 265 of *Electronic Notes in Theoretical Computer Science*, pages 213–230, 2010. doi:10.1016/j.entcs.2010.08.013.

21 Maria Manuel Clementino and Dirk Hofmann. Exponentiation in V-categories. *Topology and its Applications*, 153(16):3113–3128, 2006. doi:10.1016/j.topol.2005.01.038.

22 Stefan Cobzaş. Lipschitz properties of convex functions. *Advances in Operator Theory*, 2(1):21–49, 2017. doi:10.22034/aot.1610.1022.

23 Raphaëlle Crubillé. Probabilistic stable functions on discrete cones are power series. In *Proceedings LICS 2018*, pages 275–284. ACM, 2018. doi:10.1145/3209108.3209198.

24 Ugo Dal Lago, Francesco Gavazzo, and Akira Yoshimizu. Differential logical relations, part I: the simply-typed case. In *Proceedings ICALP 2019*, pages 111:1–111:14, 2019. doi:10.4230/LIPIcs.ICALP.2019.111.

25 Ugo Dal Lago, Furio Honsell, Marina Lenisa, and Paolo Pistone. On quantitative algebraic higher-order theories. In *Proceedings FSCD 2022*, volume 228 of *LIPIcs*, pages 4:1–4:18, 2022.

26 Ugo Dal Lago and Ulrich Schöpp. Computation by interaction for space-bounded functional programming. *Information and Computation*, 248:150–194, June 2016. doi:10.1016/j.ic.2015.04.006.

27 Vincent Danos and Thomas Ehrhard. Probabilistic coherence spaces as a model of higher-order probabilistic computation. *Information and Computation*, 209(6):966–991, 2011. doi:10.1016/j.ic.2011.02.001.

28 Daniel de Carvalho. Execution time of λ-terms via denotational semantics and intersection types. *Mathematical Structures in Computer Science*, 28(7):1169–1203, 2018.

29 Thomas Ehrhard. Finiteness spaces. *Mathematical Structures in Computer Science*, 15(4):615–646, 2005. URL: `https://www.cambridge.org/core/article/finiteness-spaces/E5E9CE1FA4050A56EF25CFB6F6A5754F`.

30 Thomas Ehrhard. An introduction to differential linear logic: proof-nets, models and antiderivatives. *Mathematical Structures in Computer Science*, pages 1–66, February 2017. `doi:10.1017/S0960129516000372`.

31 Thomas Ehrhard. Cones as a model of intuitionistic linear logic. In *Proceedings LICS 2020*, pages 370–383. IEEE Computer Society, 2020. `doi:10.1145/3373718.3394758`.

32 Thomas Ehrhard. Differentials and distances in probabilistic coherence spaces. *Logical Methods in Computer Science*, 18(3):2:1–2:33, 2022.

33 Thomas Ehrhard, Michele Pagani, and Christine Tasson. The computational meaning of probabilistic coherence spaces. In *Proceedings LICS 2011*, pages 87–96. IEEE Computer Society, 2011. `doi:10.1109/LICS.2011.29`.

34 Thomas Ehrhard, Michele Pagani, and Christine Tasson. Full Abstraction for Probabilistic PCF. *Journal of the ACM*, 65(4), 2018.

35 Thomas Ehrhard, Michele Pagani, and Christine Tasson. Measurable cones and stable, measurable functions: a model for probabilistic higher-order programming. In *Proceedings POPL 2018*, volume 2, pages 59:1–59:28, 2018. `doi:10.1145/3158147`.

36 Thomas Ehrhard and Laurent Regnier. The differential lambda-calculus. *Theoretical Computer Science*, 309(1):1–41, December 2003. `doi:10.1016/S0304-3975(03)00392-X`.

37 Martín Hötzen Escardó. A metric model of PCF, 1999. Unpublished note presented at the Workshop on Realizability Semantics and Applications, June 1999. Available at the author's webpage.

38 Soichiro Fuji. Enriched categories and tropical mathematics, 2019. `arXiv:1909.07620`.

39 Marco Gaboardi, Andreas Haeberlen, Justin Hsu, Arjun Narayan, and Benjamin C. Pierce. Linear dependent types for differential privacy. *SIGPLAN Not.*, 48(1):357–370, January 2013. `doi:10.1145/2480359.2429113`.

40 Zeinab Galal. A bicategorical model for finite nondeterminism. In *Proceedings FSCD 2021*, volume 195 of *LIPIcs*, pages 10:1–10:17, 2021.

41 D. Gebler and S. Tini. SOS specifications for uniformly continuous operators. *Journal of Computer and System Science*, 92:113–151, 2018.

42 Guillaume Geoffroy and Paolo Pistone. A partial metric semantics of higher-order types and approximate program transformations. In *Proceedings CSL 2021*, volume 183 of *LIPIcs*, pages 35:1–35:18, 2021.

43 Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–101, 1987. `doi:10.1016/0304-3975(87)90045-4`.

44 Jean-Yves Girard, Andre Scedrov, and Philip J. Scott. Bounded linear logic: A modular approach to polynomial time computability. *Theoretical Computer Science*, 97:1–66, 1992. Extended abstract in *Feasible Mathematics*, S. R. Buss and P. J. Scott editors, Proceedings of the MCI Workshop, Ithaca, NY, June 1989, Birkhauser, Boston, pp. 195–209.

45 Dima Grigoriev. Tropical differential equations. *Advances in Applied Mathematics*, 82:120–128, 2017. `doi:10.1016/j.aam.2016.08.002`.

46 Chris Heunen, Ohad Kammar, Sam Staton, and Hongseok Yang. A convenient category for higher-order probability theory. In *Proceedings LICS 2017*. IEEE Computer Society, 2017.

47 Dirk Hofmann, Gavin J Seal, and W Tholen. *Monoidal Topology: a Categorical Approach to Order, Metric and Topology*. Cambridge University Press, New York, 2014.

48 C. Kahlert and L.O. Chua. The complete canonical piecewise-linear representation. I. the geometry of the domain space. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 39(3):222–236, 1992. `doi:10.1109/81.128016`.

49 Shin-ya Katsumata. A double category-theoretic analysis of graded linear exponential comonads. In *Proceedings FoSSaCS 2018*, pages 110–127. Springer International Publishing, 2018.

50 Jim Laird, Giulio Manzonetto, Guy McCusker, and Michele Pagani. Weighted relational models of typed lambda-calculi. In *Proceedings LICS 2013*, pages 301–310. IEEE Computer Society, 2013. doi:10.1109/LICS.2013.36.

51 F. William Lawvere. Metric spaces, generalized logic, and closed categories. *Rendiconti del Seminario Matematico e Fisico di Milano*, 43(1):135–166, December 1973. doi:10.1007/BF02924844.

52 Jean-Simon Pacaud Lemay. Convenient antiderivatives for differential linear categories. *Mathematical Structures in Computer Science*, 30(5):545–569, 2020.

53 Jean-Simon Pacaud Lemay. Coderelictions for Free Exponential Modalities. In *Proceedings CALCO 2021*, volume 211 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 19:1–19:21, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.CALCO.2021.19.

54 Yves Lucet. What shape is your conjugate? a survey of computational convex analysis and its applications. *SIAM Journal on Optimization*, 20(1):216–250, 2009. doi:10.1137/080719613.

55 Diane Maclagan and Bernd Sturmfels. *Introduction to tropical geometry*, volume 161 of *Graduate Studies in Mathematics*. American Mathematical Society, 2015.

56 Giulio Manzonetto. What is a categorical model of the differential and the resource λ-calculi? *Mathematical Structures in Computer Science*, 22(3):451–520, 2012. doi:10.1017/S0960129511000594.

57 Petros Maragos, Vasileios Charisopoulos, and Emmanouil Theodosis. Tropical geometry and machine learning. *Proceedings of the IEEE*, 109(5):728–755, 2021. doi:10.1109/JPROC.2021.3065238.

58 Radu Mardare, Prakash Panangaden, and Gordon Plotkin. Quantitative algebraic reasoning. In *Proceedings LICS 2016*. IEEE Computer Society, 2016.

59 Damiano Mazza, Luc Pellissier, and Pierre Vial. Polyadic approximations, fibrations and intersection types. In *Proceedings POPL 2018*. ACM, 2018.

60 Paul-André Melliès, Nicolas Tabareau, and Christine Tasson. An explicit formula for the free exponential modality of linear logic. *Mathematical Structures in Computer Science*, 28(7):1253–1286, 2018. doi:10.1017/S0960129516000426.

61 Gian Maria Negri Porzio, Vanni Noferini, and Leonardo Robol. Tropical Laurent series, their tropical roots, and localization results for the eigenvalues of nonlinear matrix functions, 2021. arXiv:2107.07982.

62 Vanni Noferini, Meisam Sharify, and Françoise Tisseur. Tropical roots as approximations to eigenvalues of matrix polynomials. *SIAM J. Matrix Anal. Appl.*, 36(1):138–157, January 2015. doi:10.1137/14096637X.

63 Federico Olimpieri. Intersection type distributors. In *Proceedings LICS 2021*. IEEE Computer Society, 2021. doi:10.1109/LICS52264.2021.9470617.

64 Lior Pachter and Bernd Sturmfels. Tropical geometry of statistical models. *Proceedings of the National Academy of Sciences*, 101(46):16132–16137, 2023/01/16 2004. doi:10.1073/pnas.0406010101.

65 Michele Pagani and Paolo Tranquilli. Parallel reduction in resource λ-calculus. In *Proceedings APLAS 2009*, pages 226–242, 2009.

66 Paolo Pistone. On generalized metric spaces for the simply typed λ-calculus. In *Proceedings LICS 2021*, pages 1–14. IEEE Computer Society, 2021.

67 Gordon Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5(3):223–255, 1977.

68 Jason Reed and Benjamin C. Pierce. Distance makes the types grow stronger. *Proceedings ICFP 2010*, pages 157–168, 2010.

69 Ciro Russo. *Quantale Modules, with Applications to Logic and Image Processing*. PhD thesis, Università degli Studi di Salerno, available at arXiv:0909.4493, 2007.

70 Ulrich Schöpp. Stratified Bounded Affine Logic for Logarithmic Space. In *22nd Annual IEEE Symposium on Logic in Computer Science (LICS 2007)*, pages 411–420, July 2007. doi:10.1109/LICS.2007.45.

71 Peter Selinger. Towards a semantics for higher-order quantum computation. In *Proceedings QPL 2004*, TUCS General Publication No 33, pages 127–143, 2004.

72 Imre Simon. On semigroups of matrices over the tropical semiring. *Informatique Théorique et Applications*, 28:277–294, 1994.

73 Isar Stubbe. Categorical structures enriched in a quantaloid: Tensored and cotensored categories. *Theory and Applications of Categories*, 16(14):283–306, 2006.

74 Isar Stubbe. An introduction to quantaloid-enriched categories. *Fuzzy Sets and Systems*, 256:95–116, 2014. Special Issue on Enriched Category Theory and Related Topics (Selected papers from the 33rd Linz Seminar on Fuzzy Set Theory, 2012). doi:10.1016/j.fss.2013.08.009.

75 Franck van Breugel. An introduction to metric semantics: operational and denotational models for programming and specification languages. *Theoretical Computer Science*, 258(1):1–98, 2001. doi:10.1016/S0304-3975(00)00403-5.

76 Simon Willerton. Tight spans, Isbell completions and semi-tropical modules. *Theory and Applications of Categories*, 28(22):696–732, 2013.

77 Liwen Zhang, Gregory Naitzat, and Lek-Heng Lim. Tropical geometry of deep neural networks. In *Proceedings ICML 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 5819–5827. PMLR, 2018. URL: http://proceedings.mlr.press/v80/zhang18i.html.

Expressivity Landscape for Logics with Probabilistic Interventionist Counterfactuals

Fausto Barbero
University of Helsinki, Finland

Jonni Virtema
University of Sheffield, UK

Abstract

Causal multiteam semantics is a framework where probabilistic dependencies arising from data and causation between variables can be together formalized and studied logically. We discover complete characterizations of expressivity for several logics that can express probabilistic statements, conditioning and interventionist counterfactuals. The results characterize the languages in terms of families of linear equations and closure conditions that define the corresponding classes of causal multiteams. The characterizations yield a strict hierarchy of expressive power. Finally, we present some undefinability results based on the characterizations.

2012 ACM Subject Classification Computing methodologies → Probabilistic reasoning; Mathematics of computing → Causal networks

Keywords and phrases Interventionist counterfactuals, Multiteam semantics, Causation, Probability logic, Linear inequalities, Expressive power

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.15

Related Version *Full Version*: https://doi.org/10.48550/arXiv.2303.11993

Funding *Fausto Barbero*: DFG grant VI 1045-1/1; Academy of Finland grants 316460 and 349803
Jonni Virtema: DFG grant VI 1045-1/1; Academy of Finland grant 345634

1 Introduction

The main approach to the study of empirical data in the 20th century has been that of statistics, which makes use of probabilistic notions such as *correlation* and *conditional (in)dependence* between variables. We follow here another line of study – going back at least to Sewall Wright [40] – insisting that the analysis should not stop at correlations, but instead should yield information about causation among variables (conditional on appropriate scientific assumptions). The methods involved in the analysis of causes and effects have gained in popularity in the last decades, and their mathematics has been vastly developed under the label of *causal inference* (see, e.g., [30, 35]). Today the methods of causal inference are heavily utilized, e.g., in epidemiology [23], econometrics [22], social sciences [28] and machine learning [32, 33].

One of the next crucial steps in the development of artificial intelligence will be the capability of AI systems to represent and reason about causal knowledge (see, e.g., [31]). For the development of AI applications of causal inference, the clarification of the related formal logical theory is vital. It turns out that many concepts involved in the analysis of causes can be reduced to the study of *interventionist counterfactuals* in *causal models*. Causal models represent causation between variables using so-called *structural equations*, which describe deterministic causal laws that relate the variables to each other. In their simplest form, interventionist counterfactuals are expressions of the form

"if variables $X_1, \ldots, X_n$ were set to values $x_1, \ldots, x_n$, then Y would take value y".

Such conditionals are counterfactual (contrary to fact) in that their evaluation forces us to consider an alternative scenario in which the variables $X_1, \ldots, X_n$ are subtracted to the laws that currently determine their behaviour, and the (possibly new) values taken by such variables are fixed by some external intervention. The causal laws encoded in the model then allow us to find out, computationally, how all the variables in the system are affected in this alternative scenario. Research on logics encompassing interventionist counterfactuals has been active in the past two decades. For example, [13, 17, 7, 9] provided complete axiomatizations for languages of increasing generality. The papers [18, 41] drew precise connections with the earlier Stalnaker-Lewis theory of counterfactuals [36, 27]. In [1] logics for causal reasoning were studied via translations to first-order logic, and the articles [17, 16, 29] discuss the complexity of causal and probabilistic languages.

The classical literature on causal inference does not neatly separate the methods of probability and of causal analysis; many standard concepts in causal inference are expressed by mixing probabilistic and causal statements. In other words, causal inference uses an array of new notational devices that are not entirely reducible to classical probabilistic reasoning; two significant examples (from [30]) of these new notations are the conditional *do* expressions ($Pr(y \mid do(x), z)$) and Pearl's "counterfactuals" ($Pr(Y_{X=x} \mid Z = z)$). We refer the reader to [8] for a detailed discussion of the meaning and use of these expressions. Roughly speaking, they both describe the probability that the variable Y takes value y after intervening to set X to x, conditional upon the observation that Z takes value z; but the two expressions differ subtly in that in the former the conditioning is performed in the system modified by the intervention that sets X to x, while in the latter expression conditioning is relative to the pre-intervention system. To this regard, we follow the proposal of Barbero and Sandu [2, 4] to decompose these complex causal-probabilistic expressions in terms of a minimal set of logical primitives. In particular, probabilistic conditioning and causal interventions will correspond to two distinct logical conditionals, $\supset$ and $\boxdot\!\!\rightarrow$.

In order to make this decomposition possible, one needs to move from causal models to the more general *causal multiteam semantics*, where all the needed logical operators are available. Team semantics is the semantical framework of modern logics of dependence and independence. Introduced by Hodges [24] and adapted to dependence logic by Väänänen [38], team semantics defines truth in reference to collections of assignments, called *teams*. Team semantics is particularly suitable for the formal analysis of dependencies and independencies in data. Recent developments in the area have broadened the scope of team semantics to cover probabilistic and quantitative notions of dependence and independence. Durand et al. [11, 10] introduced multiset and probabilistic variants of team semantics as frameworks for studying probabilistic dependency notions such as conditional independence logically. Further analysis has revealed that definability and complexity of logics in these frameworks are intimately connected to definability and complexity of Presburger ([14, 39]) and real arithmetic ([21, 20]).

Causal teams, proposed by Barbero and Sandu [3], fuse together teams and causal models, and model inferences encompassing both functional dependencies arising from data and causal dependencies arising from structural equations. The logics considered by Barbero and Sandu use atomic expressions of the form $X = x$ and $=(X; Y)$ to state that the variable X takes the value x and that (in the data) the value of the variable Y is functionally determined by the values of the variable X, respectively. *Interventionist counterfactuals* ($X = x \boxdot\!\!\rightarrow \psi$) and *selective implications* ($\alpha \supset \psi$) then describe consequences of actions and consequences of learning from observations. For example, the intended reading of the formula "Pressure = 300 $\boxdot\!\!\rightarrow$ Volume = 4" is: *If we raise the pressure to* 300 *kPa, the*

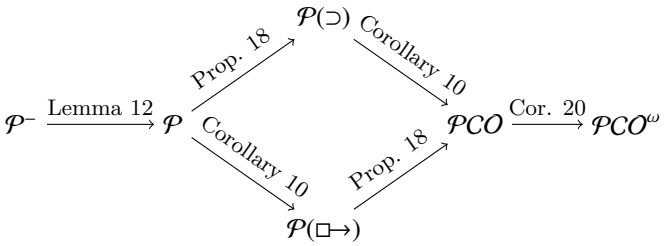


Figure 1 Arrows denote strict inclusion of expressivity; $\mathcal{P}(\square\!\!\rightarrow)$ and $\mathcal{P}(\supset)$ are incomparable.

volume of the gas will be 4 m^3. On the other hand, the intended reading of the formula "Pressure = 20 $\supset$ 10 < Altitude < 30" is: *If we read* 20 *kPa from the barometer, the current altitude is between 10 and 30 km.*

Finally, the *causal multiteam semantics* coined by Barbero and Sandu [4] fuses together multiteams and causal models. The shift from teams to multiteams makes it now possible to study probabilistic conditioning and causal interventions in a unified framework. Barbero and Sandu study a language called $\mathcal{PCO}$ (for Probabilities, Causes and Observations) which they claim to capture a fair portion of the probabilistic causal reasoning that appears in the field of causal inference. It does indeed suffice to capture many forms of probabilistic conditioning, and it suffices to express conditional *do* expressions, the "Pearl counterfactuals" mentioned above and more general kinds of statements. For example, the statement "the probability that a sick untreated patient would be healed when treated is at least $\frac{2}{3}$" can be formalised as (Sick = 1 $\wedge$ Treated = 0) $\supset$ (Treated = 1 $\square\!\!\rightarrow$ Pr(Sick = 0) $\geq \frac{2}{3}$). The paper [4] raises however the doubt whether $\mathcal{PCO}$ can express, in general, the comparison of conditional probabilities (e.g., statements of the form $\mathrm{Pr}(\alpha \mid \beta) \geq \mathrm{Pr}(\gamma \mid \delta)$). We show here that it fails to do so; thus, $\mathcal{PCO}$ cannot be used, for instance, to compare the expected efficacy of two distinct (non-enforced) medical treatments.

The cornerstone of this inexpressibility result is an abstract characterization of the expressive power of $\mathcal{PCO}$, which in particular shows that the classes of probability distributions that are consistent with a given $\mathcal{PCO}$ formula can be described in terms of a certain class of linear inequalities. On the other hand, by a geometrical argument we see that there are statements of comparison of conditional probabilities which unavoidably involve inequalities of second degree. The quest for an understanding of language $\mathcal{PCO}$ naturally proceeds via an understanding of the expressivity of its key resources: evaluation atoms ($\mathrm{Pr}(\alpha) \geq \epsilon$), comparison atoms ($\mathrm{Pr}(\alpha) \geq \mathrm{Pr}(\beta)$), observations ($\supset$) and interventions ($\square\!\!\rightarrow$). This leads us to the study of four fragments $\mathcal{P}^-$, $\mathcal{P}$, $\mathcal{P}(\supset)$, and $\mathcal{P}(\square\!\!\rightarrow)$. We characterize the expressive power of each of these sublogics, as well as the expressivity of $\mathcal{PCO}$, in terms of closure properties and of an appropriate class of linear inequalities; these results are schematized in Table 1. Together with geometrical reasoning, these characterizations yield a strict hierarchy of expressive power, as summarized in Figure 1. The table and the figure also include a language $\mathcal{PCO}^\omega$ that extends $\mathcal{PCO}$ with (countably) infinite disjunctions. The manuscript [4] already shows that this language is more expressive than $\mathcal{PCO}$; our results yield an alternative proof.

The characterization and hierarchy results can be found in Section 3, after a presentation of the semantics and syntax of the languages (Section 2). Section 4 presents the inexpressibility result for conditional comparison atoms, and briefly discusses the related issue of definability of dependencies and independencies.

Table 1 Characterizations of expressivity of logics. E.g., a class $\mathcal{K}$ of causal multiteams is definable by a $\mathcal{P}(\supset)$-formula iff $\mathcal{K}$ is signed binary, closed under change of laws and rescaling, and has the empty multiteam property. $\mathcal{K}$ is a union of signed binary, when $\mathcal{K} = \bigcup_{\mathcal{F}\in\mathbb{F}_\sigma} \mathcal{K}^{\mathcal{F}}$, for signed binary sets of causal multiteams $\mathcal{K}^{\mathcal{F}}$ of function component $\mathcal{F}$.

Logic	Type of inequalities	Closure properties: change of laws	Closure properties: rescaling & empty multiteam	References
$\mathcal{P}^-$	monic	X	X	Thm. 11
$\mathcal{P}$	signed monic	X	X	Thm. 11
$\mathcal{P}(\supset)$	signed binary	X	X	Thm. 16
$\mathcal{P}(\square\!\!\rightarrow)$	union of signed monics		X	Thm. 14
$\mathcal{PCO}$	union of signed binary		X	Thm. 19
$\mathcal{PCO}^\omega$	(unrestricted)		X	[4]

2 Logics with causal multiteam semantics

Capital letters such as $X, Y, \ldots$ denote **variables** (standing for specific magnitudes such as "temperature" and "volume") which take **values** denoted by small letters. The values of the variable X will be often denoted by $x, x', \ldots$. Sets (and tuples, depending on the context) of variables and values are denoted by boldface letters such as $\mathbf{X}$ and $\mathbf{x}$. We consider probabilities that arise from the counting measures of finite (multi)sets. For finite sets $S \subseteq T$, we define $P_T(S) := \frac{|S|}{|T|}$.

A **signature** is a pair $(\mathsf{Dom}, \mathsf{Ran})$, where Dom is a finite set of variables and Ran a function mapping each $X \in \mathsf{Dom}$ to a finite set $\mathsf{Ran}(X)$ of values (the **range** of X). We stipulate a fixed ordering on Dom, and write $\mathbf{W}$ for the tuple of all the variables of Dom listed in that order. We write $\mathbf{W}_X$ for the variables of $\mathsf{Dom} \setminus \{X\}$ listed according to the fixed order. For a tuple $\mathbf{X} = (X_1, \ldots, X_n)$ of variables, $\mathsf{Ran}(\mathbf{X})$ denotes the Cartesian product $\mathsf{Ran}(X_1)\times\cdots\times\mathsf{Ran}(X_n)$. An **assignment** of signature σ is a mapping $s : \mathsf{Dom} \to \bigcup_{X\in\mathsf{Dom}} \mathsf{Ran}(X)$ such that $s(X) \in \mathsf{Ran}(X)$ for each $X \in \mathsf{Dom}$. The set of all assignments of signature σ is denoted by $\mathbb{B}_\sigma$. For an assignment s having the variables of $\mathbf{X}$ in its domain, $s(\mathbf{X})$ denotes the tuple $(s(X_1), \ldots, s(X_n))$. For $\mathbf{X} \subseteq \mathsf{Dom}$, $s_{\restriction\mathbf{X}}$ is the restriction of s to the variables in $\mathbf{X}$.

A **team** T of signature σ is a subset of $\mathbb{B}_\sigma$. Intuitively, a multiteam is just a multiset analogue of a team. We represent **multiteams** as (finite) teams with an extra variable *Key* (not belonging to the signature) ranging over $\mathbb{N}$, which takes different values over different assignments of the team, and which is never mentioned in the formal languages. A multiteam can be then presented as a table; e.g., the following

T :

Key	X	Y
0	0	0
1	0	0
2	0	1

describes a multiteam containing two "copies" of the assignment $s(X, Y) = (0, 0)$ (first two rows) plus another assignment $t(X, Y) = (0, 1)$. We will say that the variable domain of this multiteam T is $\mathsf{Dom} = \{X, Y\}$, and omit mentioning the *Key* variable. Multiteams will be used to encode probability distributions over the underlying team (in this case, the distribution that assigns probability $\frac{2}{3}$ to assignment s, and probability $\frac{1}{3}$ to t). The "underlying team" (i.e., support of a multiteam) is characterized formally later in Definition 6.

Multiteams by themselves do not encode any solid notion of causation; they do not tell us how a system would be affected by an intervention. We therefore need to enrich multiteams with additional structure. In particular, we will associate to some of the variables a deterministic causal law. The law for variable V takes the form of a function, which describes the way the value of V is generated from the values of other variables in the system. These laws will be used crucially in order to compute how the model is affected by an intervention. Furthermore, we will require that each assignment in the multiteam agrees with these laws.

▶ **Definition 1.** *A* ***causal multiteam*** *of signature* $(\mathrm{Dom}, \mathrm{Ran})$ *with* ***endogenous variables*** $\mathrm{End}(T) \subseteq \mathrm{Dom}$ *is a pair* $T = (T^-, \mathcal{F})$ *such that:*

1. T^- *is a multiteam of domain* Dom*,*
2. $\mathcal{F}$ *is a function* $\{(V, \mathcal{F}_V) \mid V \in \mathrm{End}(T)\}$ *that assigns to each endogenous variable* V *a non-constant* $|\mathbf{W}_V|$*-ary function* $\mathcal{F}_V : \mathrm{Ran}(\mathbf{W}_V) \to \mathrm{Ran}(V)$*,*
3. $(T^-, \mathcal{F})$ *satisfies the* ***compatibility constraint****:* $\mathcal{F}_V(s(\mathbf{W}_V)) = s(V)$*, for all* $s \in T^-$ *and* $V \in \mathrm{End}(T)$*.*

T^- *and* $\mathcal{F}$ *will be called, respectively, the* ***multiteam component*** *and the* ***function component*** *of* T*. We write* $(\mathrm{Dom}(T), \mathrm{Ran}(T))$ *to denote the signature of the causal multiteam* T*.*

Notice that, due to the compatibility constraint, not all instances for $End(T)$ and T^- give rise to causal multiteams. The function component $\mathcal{F}$ induces a system of structural equations; an equation $V := \mathcal{F}_V(\mathbf{W}_V)$ for each variable $V \in \mathrm{End}(T)$. Note that some of the variables in $\mathbf{W}_V$ may not be necessary for evaluating V. For example, if V is given by the structural equation $V := X + 1$, all the variables in $\mathbf{W}_V \setminus \{X\}$ are irrelevant (we call them **dummy arguments** of $\mathcal{F}_V$). The set of non-dummy arguments of $\mathcal{F}_V$ is denoted as PA_V (the set of **parents** of V).

We associate to each causal multiteam T a **causal graph** G_T, whose vertices are the variables in Dom and where an arrow is drawn from each variable in PA_V to V, whenever $V \in \mathrm{End}(T)$ (see Example 3 and picture 2 for a depiction). The variables in $\mathrm{Dom}(T) \setminus \mathrm{End}(T)$ are called **exogenous** (written $\mathrm{Exo}(T)$).

In the present paper we restrict attention to systems of variables that are connected by causal laws that do not form cycles (e.g., we exclude the possibility that X causally affects Y, Y causally affects Z, and in turn Z affects X); such systems are usually called *recursive*. Concretely, we enforce the following convention:

Throughout the paper we will implicitly assume that causal multiteams have an acyclic causal graph.

While the study of cyclic systems is far from absent from the literature (e.g. [37, 34, 17, 1]), in a probabilistic context it introduces a number of complications that go well beyond the scope of the framework considered in this paper.

▶ **Definition 2.** *A causal multiteam* $S = (S^-, \mathcal{F}_S)$ *is a* ***causal sub-multiteam*** *of* $T = (T^-, \mathcal{F}_T)$*, if they have the same signature,* $S^- \subseteq T^-$*, and* $\mathcal{F}_S = \mathcal{F}_T$*. We then write* $S \leq T$*.*

We consider causal multiteams as dynamic models, that can be affected by observations and interventions. Given a causal multiteam $T = (T^-, \mathcal{F})$ and a formula α of some formal language (evaluated over causal multiteams according to some semantic relation $\models$), "observing α" produces the causal sub-multiteam $T^\alpha = ((T^\alpha)^-, \mathcal{F})$ of T, where $(T^\alpha)^- := \{s \in T^- \mid (\{s\}, \mathcal{F}) \models \alpha\}$.[1] An intervention on T will *not*, in general, produce a sub-multiteam of T. It will instead

[1] Throughout the paper, the semantic relation in terms of which T^α is defined will be the semantic relation for language CO, which shall be defined below.

T^-:

Key	X	Y	Z
0	0	1	1
1	1	2	3
2	1	2	3
3	2	3	5
4	2	3	5
5	2	3	5

$\rightsquigarrow$

Key	X	Y	Z
0	0	**1**	...
1	1	**1**	...
2	1	**1**	...
3	2	**1**	...
4	2	**1**	...
5	2	**1**	...

$\rightsquigarrow$ $T^-_{Y=1}$:

Key	X	Y	Z
0	0	1	**1**
1	1	1	**2**
2	1	1	**2**
3	2	1	**3**
4	2	1	**3**
5	2	1	**3**

Figure 2 Causal multiteams for Example 3, showing how the multiteam component $T^-_{Y=1}$ of a causal multiteam is computed from T^- given an intervention $do(Y = 1)$. The figure also describes the associated causal graphs.

modify the values that appear in some of the columns of T. We consider interventions that are described by conjunctions of the form $X_1 = x_1 \wedge \cdots \wedge X_n = x_n$ (or, shortly, $\mathbf{X} = \mathbf{x}$). Such a formula is **inconsistent** if there are two indexes i, j such that X_i and X_j denote the same variable, while x_i and x_j denote distinct values; it is **consistent** otherwise. Applying an intervention $do(\mathbf{X} = \mathbf{x})$, where $\mathbf{X} = \mathbf{x}$ is consistent, to a causal multiteam $T = (T^-, \mathcal{F})$ of endogenous variables $\mathbf{V}$ will produce a causal multiteam $T_{\mathbf{X}=\mathbf{x}} = (T^-_{\mathbf{X}=\mathbf{x}}, \mathcal{F}_{\mathbf{X}=\mathbf{x}})$, where the function component is $\mathcal{F}_{\mathbf{X}=\mathbf{x}} := \mathcal{F}_{\restriction(\mathbf{V}\setminus\mathbf{X})}$ (the restriction of $\mathcal{F}$ to the set of variables $\mathbf{V} \setminus \mathbf{X}$) and the multiteam component is $T^-_{\mathbf{X}=\mathbf{x}} := \{s^{\mathcal{F}}_{\mathbf{X}=\mathbf{x}} \mid s \in T^-\}$, where each $s^{\mathcal{F}}_{\mathbf{X}=\mathbf{x}}$ is the unique assignment compatible with $\mathcal{F}_{\mathbf{X}=\mathbf{x}}$ defined (recursively) as

$$s^{\mathcal{F}}_{\mathbf{X}=\mathbf{x}}(V) = \begin{cases} x_i & \text{if } V = X_i \in \mathbf{X} \\ s(V) & \text{if } V \in \mathrm{Exo}(T) \setminus \mathbf{X} \\ \mathcal{F}_V(s^{\mathcal{F}}_{\mathbf{X}=\mathbf{x}}(\mathbf{W}_V)) & \text{if } V \in \mathrm{End}(T) \setminus \mathbf{X}. \end{cases}$$

We emphasize that the uniqueness of $s^{\mathcal{F}}_{\mathbf{X}=\mathbf{x}}$, and thus the correctness of this definition, hinges on our assumption that the causal graphs are acyclic. For an explanation of how interventions may be defined in the cyclic (non-probabilistic) case, see [1].

▶ **Example 3.** Consider the causal multiteam $T = (T^-, \mathcal{F})$ depicted in Figure 2, where each row of the leftmost table depicts an assignment of T^- (e.g., the third row represents an assignment s with $s(Key) = 2$, $s(X) = 1$, $s(Y) = 2$, $s(Z) = 3$). The rows of the table are compatible with the laws $\mathcal{F}_Z(X, Y) = X + Y$ and $\mathcal{F}_Y(X) = X + 1$, while X is exogenous. T encodes probabilities for formulas that discuss variables X, Y, Z and their possible values; for example, $P_T(Z = 3) = \frac{1}{3}$.

Suppose we can enforce the variable Y to take the value 1. The effect of such an intervention, depicted in the right-hand side of Figure 2, is to first set the value of Y to 1 (in all rows) and then to recompute the values of Z using the function $\mathcal{F}_Z$. The probability distribution has changed: now $P_{T_{Y=1}}(Z = 3) = \frac{1}{2}$. Furthermore, the function $\mathcal{F}_Y$ is omitted from $T_{Y=1}$, and thus the arrow from X to Y has been omitted from the causal graph.

Given two languages $\mathcal{L}, \mathcal{L}'$ of signature σ, whose semantics is defined over causal multiteams, and formulae $\varphi \in \mathcal{L}$ and $\varphi' \in \mathcal{L}'$, we write $\varphi \equiv_\sigma \varphi'$ if $T \models \varphi \Leftrightarrow T \models \varphi'$ holds for all causal multiteams T of signature σ. We omit the index σ if it is clear from the context. Similarly, we may write $\mathcal{L}_\sigma$ to emphasise that the signature of $\mathcal{L}$ is σ.

We write $\mathcal{L} \leq \mathcal{L}'$ if for every $\varphi \in \mathcal{L}$ there is $\varphi' \in \mathcal{L}'$ with $\varphi \equiv \varphi'$. We write $\mathcal{L} < \mathcal{L}'$ if $\mathcal{L} \leq \mathcal{L}'$ but $\mathcal{L}' \not\leq \mathcal{L}$. Finally, we write $\mathcal{L} \equiv \mathcal{L}'$ if $\mathcal{L} \leq \mathcal{L}'$ and $\mathcal{L}' \leq \mathcal{L}$. $\mathcal{K}^\sigma_\varphi$ is the set of all causal multiteams of signature σ that satisfy φ. $\mathcal{K}^\sigma_\varphi$ will be (with the exception of contradictory formulae) a countably infinite set.

A class $\mathcal{K}$ of causal multiteams is **definable** in $\mathcal{L}_\sigma$ if $\mathcal{K} = \mathcal{K}_\varphi^\sigma$ for some $\varphi \in \mathcal{L}_\sigma$.

A class $\mathcal{K}$ is **flat** if $(T^-, \mathcal{F}) \in \mathcal{K}$ iff $(\{s\}, \mathcal{F}) \in \mathcal{K}$ for every $s \in T^-$. A class $\mathcal{K}$ of causal multiteams of signature σ has the **empty multiteam property**, if $\mathcal{K}$ includes all empty causal multiteams of signature σ (we say that a causal multiteam $(T^-, \mathcal{F})$ is **empty** if the multiteam T^- is). A σ-formula φ has one of the above (or to be defined) properties, if $\mathcal{K}_\varphi^\sigma$ has it. A language $\mathcal{L}$ is flat (resp. has the empty team property), if every $\varphi \in \mathcal{L}$ is flat (resp. has the empty team property). In general, we say that $\mathcal{L}$ has a certain property if and only if each $\varphi \in \mathcal{L}$ has it.

The language $\mathcal{CO}$, introduced in [3], is defined by the following BNF grammar:

$$\alpha ::= Y = y \mid Y \neq y \mid \alpha \wedge \alpha \mid \alpha \vee \alpha \mid \alpha \supset \alpha \mid \mathbf{X} = \mathbf{x} \,\square\!\!\rightarrow\, \alpha,$$

where $\mathbf{X} \cup \{Y\} \subseteq \mathrm{Dom}$, $y \in \mathrm{Ran}(Y)$, and $\mathbf{x} \in \mathrm{Ran}(\mathbf{X})$. It is a language for the description of facts. We will later introduce extensions that allow us to talk about the probabilities of the facts that are expressible in $\mathcal{CO}$. Formulae of the forms $Y = y$ and $Y \neq y$ are **literals**. Semantics for $\mathcal{CO}$ is given by the following clauses:

$T \models Y = y$	iff	$s(Y) = y$ for all $s \in T^-$.
$T \models Y \neq y$	iff	$s(Y) \neq y$ for all $s \in T^-$.
$T \models \alpha \wedge \beta$	iff	$T \models \alpha$ and $T \models \beta$.
$T \models \alpha \vee \beta$	iff	there are $T_1, T_2 \leq T$ s.t. $T_1^- \cup T_2^- = T^-$, $T_1^- \cap T_2^- = \emptyset, T_1 \models \alpha$ and $T_2 \models \beta$.
$T \models \alpha \supset \beta$	iff	$T^\alpha \models \beta$.
$T \models \mathbf{X} = \mathbf{x} \,\square\!\!\rightarrow\, \beta$	iff	$T_{\mathbf{X}=\mathbf{x}} \models \beta$ or $\mathbf{X} = \mathbf{x}$ is inconsistent.

where T^α is defined simultaneously with the clauses, as previously explained.

The intuitive readings of the conditional formulas $\alpha \supset \beta$ and $\mathbf{X} = \mathbf{x} \,\square\!\!\rightarrow\, \beta$ are, respectively, "After observing (or learning) α, we know that β holds" and "After setting $\mathbf{X}$ to $\mathbf{x}$, we know that β holds". Some of the semantic clauses for the other connectives may look unusual to a reader unaccustomed to team semantics, but they are natural lifts of the usual Tarskian clauses from a setting in which formulas are evaluated on single assignments to a setting where they are evaluated on a multiplicity of assignments (for an overview of team semantics, the reader may consult e.g. [12]). As an example, the clause for a disjunction $\alpha \vee \beta$ is just stating that each assignment in T satisfies either α or β. It says so by saying that T can be split into two parts, one containing assignments that satisfy α and one containing assignments that satisfy β. This reading of the clauses is made possible by the fact that language $\mathcal{CO}$ is flat. The proof of the following result is similar to that of the analogous result for causal teams [3, Thm. 2.10].

▶ **Theorem 4.** *$\mathcal{CO}_\sigma$ is flat and therefore has the empty multiteam property.*

In a sense, flatness tells us that $\mathcal{CO}$ behaves as a classical language. The probabilistic languages that we shall consider later will not be flat; probabilistic statements are meaningful at the level of teams but not at the level of the single assignments.

We also remark that in [3] the operator $\vee$ was defined without insisting that $T_1^- \cap T_2^- = \emptyset$. This was done since the paper considered set-based semantics. As our semantics is based on multisets, the appropriate definition of $\vee$ uses a union that is sensitive to multiplicities (i.e. disjoint union). Theorem 4 entails that this distinction is irrelevant for $\mathcal{CO}$, but it will have an impact in forthcoming works that apply $\vee$ to formulae φ that do not have the following property called *downward closure*: if $T \models \varphi$ and $S \leq T$, then $S \models \varphi$.

If we pick a variable X in the signature and a value $x \in Ran(X)$, we can abbreviate the formulae $X = x \vee X \neq x$ and $X = x \wedge X \neq x$ as $\top$, resp. $\bot$ (the former is a valid formula because it just says that the multiteam can be split in two parts, the assignments where X takes value x and those where it does not). The so-called **dual negation** of a formula α, $T \models \alpha^d$ iff $(\{s\}, \mathcal{F}) \not\models \alpha$ for all $s \in T^-$, is then definable in $\mathcal{CO}$ as $\alpha \supset \bot$.

Next, we introduce a language with **probabilistic atoms** $\Pr(\alpha) \geq \epsilon$, $\Pr(\alpha) > \epsilon$, $\Pr(\alpha) \geq \Pr(\beta)$, $\Pr(\alpha) > \Pr(\beta)$, where $\alpha, \beta \in \mathcal{CO}$ and $\epsilon \in [0,1] \cap \mathbb{Q}$. The first two are called **evaluation atoms**, and the latter two **comparison atoms**. Probabilistic atoms together with literals of $\mathcal{CO}$ are called **atomic formulae**. The probabilistic language $\mathcal{PCO}$ is then given by the following grammar:

$$\varphi ::= \eta \mid \varphi \wedge \varphi \mid \varphi \sqcup \varphi \mid \alpha \supset \varphi \mid \mathbf{X} = \mathbf{x} \boxdot\!\!\rightarrow \varphi,$$

where $\mathbf{X} \subseteq \mathrm{Dom}$, $\mathbf{x} \in \mathrm{Ran}(\mathbf{X})$, η is an atomic formula, and α is a $\mathcal{CO}$ formula. Note that the antecedents of $\supset$ and the arguments of probability operators are $\mathcal{CO}$ formulae. The semantic clauses for the additional operators are given below:

$T \models \psi \sqcup \chi$	iff	$T \models \psi$ or $T \models \chi$
$T \models \Pr(\alpha) \geq \epsilon$	iff	$T^- = \emptyset$ or $P_T(\alpha) \geq \epsilon$
$T \models \Pr(\alpha) > \epsilon$	iff	$T^- = \emptyset$ or $P_T(\alpha) > \epsilon$
$T \models \Pr(\alpha) \geq \Pr(\beta)$	iff	$T^- = \emptyset$ or $P_T(\alpha) \geq P_T(\beta)$
$T \models \Pr(\alpha) > \Pr(\beta)$	iff	$T^- = \emptyset$ or $P_T(\alpha) > P_T(\beta)$,

where $P_T(\alpha)$ is a shorthand for $P_{T^-}((T^\alpha)^-)$.[2] The language $\mathcal{PCO}$ still has the empty team property but it is not flat. The definability of the dual negation in $\mathcal{CO}$ allows us to introduce many useful abbreviations:

$$\Pr(\alpha) \leq \epsilon := \Pr(\alpha^d) \geq 1 - \epsilon \qquad \Pr(\alpha) = \epsilon := \Pr(\alpha) \geq \epsilon \wedge \Pr(\alpha) \leq \epsilon$$
$$\Pr(\alpha) < \epsilon := \Pr(\alpha^d) > 1 - \epsilon \qquad \Pr(\alpha) \neq \epsilon := \Pr(\alpha) > \epsilon \sqcup \Pr(\alpha) < \epsilon$$

We will see in Section 4 that the $\supset$ operator enables us to express some statements involving conditional probabilities.

We consider the following syntactic fragments of $\mathcal{PCO}$, which preserve the syntactic restrictions yielded by its two level syntax – that the antecedents of $\supset$ and the arguments of Pr are always $\mathcal{CO}$ formulae. $\mathcal{P}$ is the fragment without $\supset$ and $\boxdot\!\!\rightarrow$. $\mathcal{P}^-$ is the fragment of $\mathcal{P}$ without comparison atoms. $\mathcal{P}(\boxdot\!\!\rightarrow)$ and $\mathcal{P}(\supset)$ are fragments of $\mathcal{PCO}$ without $\supset$ and $\boxdot\!\!\rightarrow$, respectively. Finally, $\mathcal{PCO}^\omega$ is the extension of $\mathcal{PCO}$ with countable disjunctions of the form $\bigsqcup_{i \in I} \psi_i$, where the ψ_i are $\mathcal{PCO}$ formulae.

▶ **Example 5.** Let $T = (T^-, \mathcal{F})$ be a causal multiteam over variables GroundSpeed, DescentAngle, StructuralIntegrity, SafeLanding depicting data related to landing an Airbus A350-900 aircraft. The first three variables are numerical, while the last is Boolean. The structural equation $F_{\mathrm{SL}}(\mathrm{GS}, \mathrm{DA}, \mathrm{SI})$ outputs a Boolean value "true" when a plane of given structural integrity is expected to make a safe landing at a given speed and angle. The formula "$\mathrm{SI} \neq 0 \supset [(\mathrm{GS} = 300 \wedge \mathrm{DA} = 4) \boxdot\!\!\rightarrow \Pr(\mathrm{SL} = \mathrm{false}) < 0.01]$" expresses that the

[2] We remark that in $\mathcal{PCO}$ (but not in $\mathcal{CO}$!) is it also possible to define, inductively, an operator that behaves as classical negation on nonempty causal multiteams (*weak contradictory negation*). Details can be found in [6]; we will not use it here.

probability of landing failure is less than 1% when setting a landing speed of $300km/h$ and descent angle of 4 degrees, conditional on the plane not being grounded due to structural condition ($\mathrm{SI} = 0$).

Since we can assume that SI is exogenous (the assessment of structural integrity is not affected by the speed and angle set during the flight), this statement can be equivalently written as "$(\mathrm{GS} = 300 \wedge \mathrm{DA} = 4) \boxdot\!\!\rightarrow (\mathrm{SI} \neq 0 \supset \Pr(\mathrm{SL} = \mathrm{false}) < 0.01)$". This would not be legitimate if SI was causally affected by GS or DA; the operators $\boxdot\!\!\rightarrow$ and $\supset$ do not in general commute with each other.

3 Expressive power of fragments of $\mathcal{PCO}$

We start by rephrasing the known characterizations from the literature. A number of results appear in the literature (e.g. in [7]) that characterize causal languages in the context of causal *team* semantics. A causal team (of signature σ) is, essentially, a pair $(T^-, \mathcal{F})$, where T^- is a team instead of a multiteam (i.e., a set of assignments on Dom instead of $Dom \cup \{Key\}$), satisfying the conditions given in Definition 1. Each causal multiteam can be seen as a causal team enriched with a probability distribution. This correspondence is expressed precisely as follows:

▶ **Definition 6.** *The* ***support*** *of a causal multiteam $T = (T^-, \mathcal{F})$ is the causal team* $\mathrm{Team}(T) = (\mathrm{Team}(T^-), \mathcal{F})$*, where* $\mathrm{Team}(T^-) := \{s_{\restriction Dom} \mid s \in T^-\}$.

It is immediate to see that a language without probabilistic features (such as $\mathcal{CO}$) cannot tell apart two causal multiteams that have the same support. From this, it is straightforward but tedious (see the extended version of the paper, [5]) to show that the characterization of $\mathcal{CO}$ given in [7, Theorem 4.4] in terms of causal teams holds unchanged over causal multiteams:

▶ **Theorem 7** (Characterization of $\mathcal{CO}$)**.** *Let σ be a finite signature, and $\mathcal{K}$ a class of causal multiteams of signature σ. Then $\mathcal{K}$ is definable by a $\mathcal{CO}_\sigma$ formula (resp. a set of $\mathcal{CO}_\sigma$ formulae) if and only if $\mathcal{K}$ is flat.*

$\mathcal{PCO}$ is a purely probabilistic language; it cannot tell apart multiteams representing the same distribution. Given an assignment t and a causal team $T = (T^-, \mathcal{F})$, we write $\#(t, T)$ for the number of copies of t in T^- and (provided T is nonempty) $\epsilon_t^T := \frac{\#(t,T)}{|T^-|}$ for the probability of t in T. Two causal teams $S = (S^-, \mathcal{F})$ and $T = (T^-, \mathcal{G})$ are **rescalings** of each other ($S \sim T$) if $\mathcal{F} = \mathcal{G}$ and either $S^- = T^- = \emptyset$ or $\epsilon_t^T = \epsilon_t^S$ for each assignment t. A class $\mathcal{K}$ of causal multiteams of signature σ is **closed under rescaling** if, whenever $S \in \mathcal{K}$ and $S \sim T$, also $T \in \mathcal{K}$. An ideal language for purely probabilistic reasoning should be characterized just by this condition. It turns out that $\mathcal{PCO}$ is not expressive enough for the task, however its extension with countable global disjunctions $\mathcal{PCO}^\omega$ is.

▶ **Theorem 8** ([4])**.** *A* nonempty *class $\mathcal{K}$ of multiteams of signature σ is definable in $\mathcal{PCO}^\omega_\sigma$ iff $\mathcal{K}$ has the empty multiteam property and is closed under rescaling.*

The key to the proof is the fact that for any causal multiteam $(T^-, \mathcal{F})$ one can write $\mathcal{PCO}$-formulae Θ_{T^-} and $\Phi^{\mathcal{F}}$ that characterize the properties of having team component T^- (up to rescaling) and function component $\mathcal{F}$, respectively. A set $\mathcal{K}$ of causal multiteams is then defined by the formula $\bigsqcup_{(T^-,\mathcal{F})\in\mathcal{K}}(\Theta_{T^-} \wedge \Phi^{\mathcal{F}})$. Since $\mathcal{K}$ can be countably infinite, the proof crucially depends on the use of infinitary disjunctions and gives us no hints on how to obtain a finitary logic with such expressivity. Actually, a counting argument given in [4] shows that such a language must be uncountable, and thus that $\mathcal{PCO} < \mathcal{PCO}^\omega$. Our characterization of the expressivity of $\mathcal{PCO}$ will provide an alternative proof for the strict inclusion.

In order to characterize the expressivity of $\mathcal{PCO}$ and its fragments, we need to introduce some classes of linear equations and closure properties of classes of causal multiteams. For the latter, we have already seen closure under rescaling and the empty multiteam property. A class $\mathcal{K}$ of causal multiteams of signature σ is **closed under change of laws** if, whenever $(T^-, \mathcal{F}) \in \mathcal{K}$ and $\mathcal{G}$ is a system of functions of signature σ such that $(T^-, \mathcal{G})$ satisfies the compatibility constraint (point 3. of definition 1), then $(T^-, \mathcal{G}) \in \mathcal{K}$.

It is self-evident that the logics without $\Box\!\!\rightarrow$ are closed under change of laws, while the logics with $\Box\!\!\rightarrow$ are not. Thus, the following hold.

▶ **Lemma 9.** *$\mathcal{P}^-$, $\mathcal{P}$, and $\mathcal{P}(\supset)$ are closed under change of laws. $\mathcal{PCO}$, $\mathcal{P}(\Box\!\!\rightarrow)$, and $\mathcal{CO}$ are not closed under change of laws.*

▶ **Corollary 10.** *$\mathcal{P} < \mathcal{P}(\Box\!\!\rightarrow)$, $\mathcal{P}(\supset) < \mathcal{PCO}$, and $\mathcal{P}(\Box\!\!\rightarrow) \not\leq \mathcal{P}(\supset)$.*

3.1 Monic and signed monic probability sets: $\mathcal{P}^-$, $\mathcal{P}$, and $\mathcal{P}(\Box\!\!\rightarrow)$

We characterize the expressivity of fragments of $\mathcal{PCO}$ by investigating the families of subsets of $\mathbb{Q}^n$ that are definable in the logics. For a given signature σ, we fix an enumeration $s_1, \ldots, s_n$ of the assignments of $\mathbb{B}_\sigma$; every *nonempty* causal multiteam T can then be associated with a **probability vector** $\overline{\mathrm{p}}_T = (\epsilon^T_{s_1}, \ldots, \epsilon^T_{s_n}) \in \mathbb{Q}^n$. Similarly, a class $\mathcal{K}$ of causal multiteams of signature σ has an associated **probability set** $\overline{\mathrm{P}}_\mathcal{K} = \{\overline{\mathrm{p}}_T \mid T \in \mathcal{K}, T \text{ nonempty }\}$. Note that $\overline{\mathrm{p}}_T$ and $\overline{\mathrm{P}}_\mathcal{K}$ are, respectively, a point and a subset of the **standard $n-1$-simplex** Δ^{n-1} (i.e. the set of points of $[0,1]^n \cap \mathbb{Q}^n$ that satisfy the equation $\epsilon_{s_1} + \cdots + \epsilon_{s_n} = 1$), respectively. To each formula φ, we can associate a probability set $\overline{\mathrm{P}}_\varphi := \overline{\mathrm{P}}_{\mathcal{K}_\varphi}$. Note that if S, T are causal multiteams of the same signature and same function component, such that $\overline{\mathrm{p}}_S = \overline{\mathrm{p}}_T$, then S is a rescaling of T. Similarly, a class $\mathcal{K}$ of causal multiteams of signature σ that is closed under change of laws and rescaling is the largest class of causal multiteams of signature σ having probability set $\overline{\mathrm{P}}_\mathcal{K}$.

A **linear inequality** is an expression of the form $a_1\epsilon_1 + \cdots + a_n\epsilon_n \rhd b$, where $\rhd \in \{\geq, \leq, >, <\}$, $a_1, \ldots, a_n, b \in \mathbb{Q}$, and $\epsilon_1, \ldots \epsilon_n$ are variables (in the usual algebraic sense). A linear inequality is **signed monic** if each of the a_i is in $\{0, 1, -1\}$. It is **monic** if each of the a_i is in $\{0, 1\}$. A probability set $\overline{\mathrm{P}}$ is **(signed) monic** if it is a finite union of subsets of Δ^{n-1} defined by finite systems of (signed) monic inequalities. A class $\mathcal{K}$ of causal multiteams of a fixed signature is **(signed) monic** if $\overline{\mathrm{P}}_\mathcal{K}$ is a (signed) monic probability set.

We will show that being monic and closed under change of laws and rescaling characterizes expressibility in $\mathcal{P}^-$, whereas being signed monic and closed under change of laws and rescaling characterizes expressibility in $\mathcal{P}$. The full proofs of the following theorem and the subsequent lemma can be found in the extended version of the paper ([5]). A crucial role in the proofs is played by the fact that there are only finitely many assignments of signature σ (say $s_1, \ldots, s_n$) and that we can describe each such assignment s_i with a formula $\hat{\alpha}_i := \mathbf{W} = s_i(\mathbf{W})$, where $\mathbf{W}$ lists all the variables in *Dom*.

▶ **Theorem 11.** *A class $\mathcal{K}$ of multiteams of signature σ is definable in $\mathcal{P}^-$ if and only if $\mathcal{K}$ is monic, has the empty multiteam property, and is closed under change of laws and rescaling. $\mathcal{K}$ is definable in $\mathcal{P}$ if and only if $\mathcal{K}$ is signed monic, has the empty multiteam property, and closed under change of laws and rescaling.*

Proof (sketch). The fact that $\mathcal{P}^-$ and $\mathcal{P}$ have the empty multiteam property and are closed under rescaling follows from Theorem 8. Since $T \models \Pr(\alpha) \rhd \epsilon$ (resp. $T \models \Pr(\alpha) \rhd \Pr(\beta)$) iff the monic inequality $\sum_{s \in \mathrm{Team}((T^\alpha)^-)} \epsilon^T_s \rhd \epsilon$ (resp. the signed monic inequality $\sum_{s \in \mathrm{Team}((T^\alpha)^-)} \epsilon^T_s + \sum_{s \in \mathrm{Team}((T^\beta)^-)} (-1) \cdot \epsilon^T_s \rhd 0$) holds, we obtain that $\mathcal{P}^-$ (resp. $\mathcal{P}$) is monic (resp. signed monic) by induction on the syntax of formulae.

For $\mathcal{P}^{-}$, the right-to-left entailment is proved via a direct translation from finite unions of finite systems of signed monic inequalities into $\mathcal{P}^{-}$ formulae. The union of systems, which defines the probability set of $\mathcal{K}$, is expressed via a formula of the form $\varphi := \bigsqcup_{1\leq j\leq m} \bigwedge_{i\in I_j} \psi_i$, where each $\psi_i := \Pr(\bigvee_{s_k\in\mathbb{B}_\sigma | a^i_k = 1} \hat{\alpha}_k) \lhd b^i$ expresses an inequality of the form $a^i_1\epsilon_1 + \cdots + a^i_n\epsilon_n \lhd b^i$ (it is easy to see that b^i can always be assumed to be in $[0,1]\cap\mathbb{Q}$). The fact that $\mathcal{K}$ has the empty team property and closure under rescaling guarantees that $\mathcal{K}$ is "maximal", i.e. it contains *all* the causal multiteams whose probability set is defined by this system; thus $\mathcal{K} = \mathcal{K}^\sigma_\varphi$.

In contrast, for $\mathcal{P}$, we do not construct any general direct translations of signed monic inequalities into $\mathcal{P}$ formulas. However, the signed monic inequalities with constant coefficient 0, say $\sum_{i\in I}\epsilon_i - \sum_{j\in J}\epsilon_j \lhd 0$ with $I\cap J = \emptyset$, are easily translated as $\Pr(\bigwedge_{i\in I}\hat{\alpha}_i) \lhd \Pr(\bigwedge_{j\in J}\hat{\alpha}_j)$. In order to extend the argument to inequalities with nonzero constant coefficient, we first use the simplex equality $\epsilon_1 + \cdots + \epsilon_n = 1$ in order to show that we can assume that such inequality e has at least one null variable coefficient – say, it is of the form $a_1\epsilon_1 + \cdots + a_{n-1}\epsilon_{n-1} \lhd b$ (one must be careful to ensure that in this simplified inequality we still have $a_i \in \{0,1,-1\}$ and $b\in[0,1]\cap\mathbb{Q}$). But now e is equivalent to a system of three inequalities:

$$\begin{cases} a_1\epsilon_1 + \cdots + a_{n-1}\epsilon_{n-1} - \epsilon_n \lhd 0 \\ \epsilon_n \leq b \\ \epsilon_n \geq b \end{cases}$$

the first of which is expressible in $\mathcal{P}$ (since its constant coefficient is zero), while the second and third are even expressible in $\mathcal{P}^{-}$. ◀

It is not immediate to see whether $\mathcal{P}^{-} \leq \mathcal{P}$ is strict. However, by analyzing the geometry of Δ^{n-1} we are able show that there are signed monic classes of causal multiteams that are not monic. The following lemma establishes that not all signed monic probability sets can be captured by monic inequalities (more specifically, that this happens for sets defined by a single signed monic inequality). Together with the previous theorem this implies that $\mathcal{P}^{-} < \mathcal{P}$.

▶ **Lemma 12.** *Consider a nonempty probability set $\overline{\mathrm{P}} \subset \Delta^{n-1}$ which is defined by an inequality $a_1\epsilon_1 + \cdots + a_n\epsilon_n \leq b$, where there are indexes i, j such that a_i is 1 and a_j is -1, and b is a rational number in $[0,1]$. Then $\overline{\mathrm{P}}$ is not a monic probability set.*

Proof (sketch). The projection of the set described in the statement on the (i,j)-plane has as its frontier a line that is perpendicular to the segment of extremes $(0,1),(1,0)$. On the other hand, monic equalities describe, in this projection, only lines that are either parallel to this segment or parallel to one of the axis. ◀

Next we turn to characterize the expressivity of $\mathcal{P}(\boxdot\!\!\rightarrow)$. First note that while $\mathcal{P}(\boxdot\!\!\rightarrow)$ is in general more expressive than $\mathcal{P}$ (Corollary 10), if we restrict attention to causal multiteams with a fixed function component, all occurrences of $\boxdot\!\!\rightarrow$ can be eliminated from $\mathcal{P}(\boxdot\!\!\rightarrow)$ formulae (or even $\mathcal{PCO}$ formulae). The following result is proven in the extended version of the paper ([5]).

▶ **Proposition 13.** *Let $\varphi \in \mathcal{P}(\boxdot\!\!\rightarrow)_\sigma$ (resp. $\mathcal{PCO}_\sigma$), and $\mathcal{F}$ a function component of signature σ. Then there is a formula $\varphi^{\mathcal{F}} \in \mathcal{P}_\sigma$ (resp. $\mathcal{P}(\supset)_\sigma$) such that, for every causal multiteam T of signature σ and function component $\mathcal{F}$, $T \models \varphi \Leftrightarrow T \models \varphi^{\mathcal{F}}$.*

Proof (sketch). Write α_s for the formula $\mathbf{W} = s(\mathbf{W})$. First, for every subformulae of φ of the form $\beta \supset \psi$, replace β with $\bigvee_{(\{s\},\mathcal{F})\models\beta} \alpha_s$ (this removes occurrences of $\boxdot\!\!\rightarrow$ from antecedents of $\supset$). Next, we use the fact that $\boxdot\!\!\rightarrow$ distributes over $\wedge, \sqcup, \supset$ to guarantee that the consequents

of $\boxdot\!\!\rightarrow$ are atoms. The atoms can be assumed to be probabilistic (since $X = x \equiv \Pr(X = x) \geq 1$, and similarly for $X \neq x$). Then, we use the equivalences

$$\mathbf{X} = \mathbf{x} \boxdot\!\!\rightarrow \Pr(\alpha) \lhd \epsilon \equiv \Pr(\mathbf{X} = \mathbf{x} \boxdot\!\!\rightarrow \alpha) \lhd \epsilon$$

$$\mathbf{X} = \mathbf{x} \boxdot\!\!\rightarrow \Pr(\alpha) \lhd \Pr(\beta) \equiv \Pr(\mathbf{X} = \mathbf{x} \boxdot\!\!\rightarrow \alpha) \lhd \Pr(\mathbf{X} = \mathbf{x} \boxdot\!\!\rightarrow \beta)$$

to ensure that all the occurences of $\boxdot\!\!\rightarrow$ are inside arguments of Pr. Finally, we replace each subformula of the form $\Pr(\alpha) \lhd \epsilon$ with $\Pr(\bigvee_{(\{s\},\mathcal{F})\models\alpha} \alpha_s) \lhd \epsilon$, and similarly for comparison atoms. ◂

Notice that, for any fixed finite signature σ, there is only a finite number of distinct function components. We denote the set they form as $\mathbb{F}_\sigma$.

▸ **Theorem 14.** *Let $\mathcal{K}$ be a class of causal multiteams of signature σ. $\mathcal{K}$ is definable by a $\mathcal{P}(\boxdot\!\!\rightarrow)_\sigma$ formula if and only if 1) $\mathcal{K}$ has the empty multiteam property, 2) $\mathcal{K}$ is closed under rescaling, and 3) $\mathcal{K} = \bigcup_{\mathcal{F}\in\mathbb{F}_\sigma} \mathcal{K}^{\mathcal{F}}$, where each $\mathcal{K}^{\mathcal{F}}$ is a signed monic set of causal multiteams of function component $\mathcal{F}$.*

Proof. We have already mentioned that there is a $\mathcal{PCO}$ formula $\Phi^{\mathcal{F}}$ characterizing the property of having function component $\mathcal{F}$. We can obtain an equivalent formula (call it $\Psi^{\mathcal{F}}$) in $\mathcal{P}(\boxdot\!\!\rightarrow)$ by replacing each subformula of $\Phi^{\mathcal{F}}$ of the form $\alpha \supset \beta$ with $\Pr(\alpha^d \vee \beta) = 1$ (the trick works because no consequent of $\supset$ in $\Phi^{\mathcal{F}}$ contains probabilistic atoms).

$\Rightarrow$) Suppose $\mathcal{K} = \mathcal{K}_\varphi$, where $\varphi \in \mathcal{P}(\boxdot\!\!\rightarrow)_\sigma$. Now define, for each $\mathcal{F} \in \mathbb{F}_\sigma$, $\mathcal{K}^{\mathcal{F}} := \mathcal{K}_{\varphi\wedge\Psi^{\mathcal{F}}}$, where $\Psi^{\mathcal{F}}$ is as described above. Clearly $\varphi \equiv \bigsqcup_{\mathcal{F}\in\mathbb{F}_\sigma}(\varphi \wedge \Psi^{\mathcal{F}})$, so $\mathcal{K}_\varphi = \bigcup_{\mathcal{F}\in\mathbb{F}_\sigma} \mathcal{K}^{\mathcal{F}}$.

Now, by Theorem 8, $\mathcal{K}_\varphi$ is closed under rescaling and has the empty multiteam property. Next, observe that, by Proposition 13, for every $\mathcal{F} \in \mathbb{F}_\sigma$ there is a formula of $\mathcal{P}_\sigma$, call it $\varphi^{\mathcal{F}}$, which is satisfied by the same causal multiteams of function component $\mathcal{F}$ as $\varphi \wedge \Psi^{\mathcal{F}}$ is. In other words, $\mathcal{K}^{\mathcal{F}}$ is the restriction of $\mathcal{K}_{\varphi^{\mathcal{F}}}$ to causal multiteams of function component $\mathcal{F}$. Thus, since $\mathcal{K}_{\varphi^{\mathcal{F}}}$ is closed under change of laws (Lemma 9), we have $\overline{\mathrm{P}}_{\mathcal{K}^{\mathcal{F}}} = \overline{\mathrm{P}}_{\mathcal{K}_{\varphi^{\mathcal{F}}}}$. Now $\mathcal{K}_{\varphi^{\mathcal{F}}}$ is signed monic (Theorem 11), and thus by $\overline{\mathrm{P}}_{\mathcal{K}^{\mathcal{F}}} = \overline{\mathrm{P}}_{\mathcal{K}_{\varphi^{\mathcal{F}}}}$ we conclude that also $\mathcal{K}^{\mathcal{F}}$ is signed monic.

$\Leftarrow$) Suppose $\mathcal{K}$ is closed under rescaling, has the empty multiteam property and $\mathcal{K} = \bigcup_{\mathcal{F}\in\mathbb{F}_\sigma} \mathcal{K}^{\mathcal{F}}$ for some sets $\mathcal{K}^{\mathcal{F}}$ as in the statement. Write $\hat{\mathcal{K}}^{\mathcal{F}}$ for the set of all causal multiteams of signature σ whose team component appears in $\mathcal{K}^{\mathcal{F}}$. It is straightforward then that also $\hat{\mathcal{K}}^{\mathcal{F}}$ is closed under rescaling, has the empty multiteam property and is signed monic; however, $\hat{\mathcal{K}}^{\mathcal{F}}$ is also, by definition, closed under change of laws. Thus, by Theorem 11, there is a $\mathcal{P}$ formula $\varphi^{\mathcal{F}}$ such that $\hat{\mathcal{K}}^{\mathcal{F}} = \mathcal{K}_{\varphi^{\mathcal{F}}}$. Note that, $\mathcal{K}^{\mathcal{F}}$ is the set of all causal multiteams of $\mathcal{K}_{\varphi^{\mathcal{F}}}$ that have function component $\mathcal{F}$. Thus $\mathcal{K}^{\mathcal{F}} = \mathcal{K}_{\varphi^{\mathcal{F}}\wedge\Psi^{\mathcal{F}}}$. Thus $\mathcal{K}$ is defined by the $\mathcal{P}(\boxdot\!\!\rightarrow)_\sigma$ formula $\bigsqcup_{\mathcal{F}\in\mathbb{F}_\sigma}(\varphi^{\mathcal{F}} \wedge \Psi^{\mathcal{F}})$. ◂

Note that the sets $\mathcal{K}^{\mathcal{F}}$ in the statement of the theorem are themselves closed under rescaling if $\mathcal{K}$ is. This immediately follows from the fact that any two causal multiteams $(T, \mathcal{F}), (S, \mathcal{G})$ with $\mathcal{F} \neq \mathcal{G}$ are *not* rescalings of each other.

3.2 Signed binary probability sets: $\mathcal{P}(\supset)$ and $\mathcal{PCO}$

A subset $\overline{\mathrm{P}}$ of Δ^{n-1} is **signed binary** if it is a finite union of sets defined by finite systems of inequalities of the form

$$c^- \sum_{i\in I} \epsilon_i + c^+ \sum_{j\in J} \epsilon_j \lhd b$$

where $I \cap J = \emptyset$, $c^-, c^+ \in \mathbb{Z}$, $c^- \leq 0$, $c^+ \geq 0$, $b \in \mathbb{Q}$. Likewise, a class $\mathcal{K}$ of causal multiteams of signature σ is signed binary if $\overline{\mathrm{P}}_{\mathcal{K}}$ is.

▶ **Lemma 15.** *Every formula $\varphi \in \mathcal{P}(\supset)$ is signed binary.*

Proof. The proof proceeds by induction on φ. We only discuss the most difficult case, when φ is of the form $\alpha \supset \psi$. Write $\lhd$ for any symbol in $\{\leq, \geq, <, >\}$. Using the distributivity of $\supset$ over $\wedge$ and $\vee$, and the equivalences $X = x \equiv \Pr(X = x) = 1, X \neq x \equiv \Pr(X \neq x) = 1, \mathbf{X} = \mathbf{x} \sqsubset\!\!\rightarrow \Pr(\alpha) \lhd \epsilon \equiv \Pr(\mathbf{X} = \mathbf{x} \sqsubset\!\!\rightarrow \alpha) \lhd \epsilon$ and $\mathbf{X} = \mathbf{x} \sqsubset\!\!\rightarrow \Pr(\alpha) \lhd \Pr(\beta) \equiv \Pr(\mathbf{X} = \mathbf{x} \sqsubset\!\!\rightarrow \alpha) \lhd \Pr(\mathbf{X} = \mathbf{x} \sqsubset\!\!\rightarrow \beta)$, we can assume ψ to be a probabilistic atom. Hence we have two cases.

1) Assume ψ is $\Pr(\beta) \lhd b$. Now $T = (T^-, \mathcal{F}) \in \mathcal{K}_\varphi$ iff either $P_T(\alpha) \leq 0$ or $P_T(\beta \mid \alpha) \lhd b$. The latter is equivalent to $P_T(\beta \wedge \alpha) \lhd b \cdot P_T(\alpha)$, which can be rewritten as

$$\sum_{\substack{s \in \mathbb{B}_\sigma \\ \{s\} \models \beta \wedge \alpha}} \epsilon_s^T \lhd b \cdot \sum_{\substack{s \in \mathbb{B}_\sigma \\ \{s\} \models \alpha}} \epsilon_s^T$$

where we write e.g. $\{s\} \models \alpha$ as a shorthand for $(\{s\}, \mathcal{F}) \models \alpha$.

The above can be rewritten as

$$\sum_{\substack{s \in \mathbb{B}_\sigma \\ \{s\} \models \beta \wedge \alpha}} \epsilon_s^T \lhd b \cdot \Big(\sum_{\substack{s \in \mathbb{B}_\sigma \\ \{s\} \models \beta \wedge \alpha}} \epsilon_s^T + \sum_{\substack{s \in \mathbb{B}_\sigma \\ \{s\} \models \neg\beta \wedge \alpha}} \epsilon_s^T \Big)$$

which again is equivalent to

$$(1 - b) \cdot \sum_{\substack{s \in \mathbb{B}_\sigma \\ \{s\} \models \beta \wedge \alpha}} \epsilon_s^T + (-b) \cdot \sum_{\substack{s \in \mathbb{B}_\sigma \\ \{s\} \models \neg\beta \wedge \alpha}} \epsilon_s^T \lhd 0. \tag{1}$$

Now, since $b \in [0, 1]$, we have $1 - b \geq 0$ and $-b \leq 0$. Then, by multiplying both sides of (1) by a common denominator of $1 - b$ and $-b$, we obtain a signed binary inequality.

On the other hand, the inequality $P_T(\alpha) \leq 0$ can be rewitten as $\sum_{\{s\} \models \alpha} \epsilon_s \leq 0$. Thus $\overline{\mathrm{P}}_\varphi$ is the union of two sets defined by signed binary inequalities.

2) Assume ψ is $\Pr(\beta) \lhd \Pr(\gamma)$. Now $T \in \mathcal{K}_\varphi$ iff either $P_T(\alpha) \leq 0$ or $P_T(\beta \mid \alpha) \lhd P_T(\gamma \mid \alpha)$. The proof then proceeds as in the previous case. ◀

▶ **Theorem 16.** *A class $\mathcal{K}$ of multiteams of signature σ is definable by a formula of $\mathcal{P}(\supset)$ if and only if $\mathcal{K}$ is signed binary, has the empty multiteam property and is closed under change of laws and rescaling.*

Proof (sketch). $\Rightarrow$) By Theorem 8, $\mathcal{K}$ is closed under rescaling. Closure under change of laws follows from Lemma 9. Lemma 15 shows that $\overline{\mathrm{P}}_\mathcal{K}$ is signed binary. The empty multiteam property is given by Theorem 4.

$\Leftarrow$) The proof strategy is analogous to that used for the characterization of $\mathcal{P}$ (in Theorem 11), although it involves more difficult calculations. We need to show that every constraint of the form

$$c^- \sum_{i \in I} \epsilon_i + c^+ \sum_{j \in J} \epsilon_j \lhd b$$

where $I \cap J = \emptyset$, $c^-, c^+ \in \mathbb{Z}$, $c^- \leq 0$, $c^+ \geq 0$, $b \in \mathbb{Q}$, can be expressed in $\mathcal{P}(\supset)$.

First of all, let us prove it in the special case when b is 0. Write d for $c^+ - c^-$. Notice that $-d \leq c^- \leq 0 \leq c^+ \leq d$. We can also assume that $d > 0$ (the case when $d = 0$ is covered by Theorem 11). Then $-\frac{c^-}{d}$ is a rational number in $[0, 1]$, and thus the following is a $\mathcal{P}(\supset)$ formula (where, as before, $\hat{\alpha}_j$ stands for $\mathbf{W} = s_j(\mathbf{W})$):

$$\Big(\bigvee_{k \in I \cup J} \hat{\alpha}_k \Big) \supset \Pr\Big(\bigvee_{j \in J} \hat{\alpha}_j \Big) \lhd -\frac{c^-}{d}.$$

Now we have

$$
\begin{aligned}
&T \models \Big(\bigvee_{k\in I\cup J} \hat{\alpha}_k \Big) \supset \Pr(\bigvee_{j\in J} \hat{\alpha}_j) \lhd -\frac{c^-}{d} \\
&\iff P_T(\bigvee_{j\in J} \hat{\alpha}_j \mid \bigvee_{k\in I\cup J} \hat{\alpha}_k) \lhd -\frac{c^-}{d} \\
&\iff d \cdot P_T(\bigvee_{j\in J} \hat{\alpha}_j \wedge \bigvee_{k\in I\cup J} \hat{\alpha}_k) \lhd -c^- \cdot P_T(\bigvee_{k\in I\cup J} \hat{\alpha}_k) \\
&\iff d \cdot P_T(\bigvee_{j\in J} \hat{\alpha}_j) \lhd -c^- \cdot P_T(\bigvee_{k\in I\cup J} \hat{\alpha}_k) \\
&\iff d \sum_{j\in J} \epsilon_j^T \lhd -c^- \sum_{k\in I\cup J} \epsilon_k^T \\
&\iff c^- \sum_{i\in I} \epsilon_i^T + (d + c^-) \sum_{j\in J} \epsilon_j^T \lhd 0 \\
&\iff c^- \sum_{i\in I} \epsilon_i^T + c^+ \sum_{j\in J} \epsilon_j^T \lhd 0,
\end{aligned}
$$

as required.

Now let us consider the case when $b \neq 0$. Suppose, first, that we have an inequality of the form $c^- \sum_{i\in I} \epsilon_i + c^+ \sum_{j\in J} \epsilon_j \lhd b$ that satisfies the additional constraint that $I \cup J = \{1, \ldots, n\}$, i.e. it contains all variables. We show that then it is equivalent to an inequality of the same form, but with coefficient 0 for at least one variable. Assuming that I is nonempty, let us pick a variable in I (that we may assume wlog to be ϵ_n). Thus the inequality can be rewritten as:

$$c^- \sum_{i\in I\setminus\{n\}} \epsilon_i + c^+ \sum_{j\in J} \epsilon_j + c^- \epsilon_n \lhd b.$$

Using the fact that, in Δ^{n-1}, $\epsilon_1 + \cdots + \epsilon_n = 1$, we can rewrite the inequality as

$$c^- \sum_{i\in I\setminus\{n\}} \epsilon_i + c^+ \sum_{j\in J} \epsilon_j + c^- - c^- \epsilon_1 - \cdots - c^- \epsilon_{n-1} \lhd b$$

i.e.,

$$\sum_{j\in J} (c^+ - c^-)\epsilon_j \lhd b - c^-,$$

which is of the correct form. In case I is empty, we can perform analogous transformations to eliminate a variable indexed in J.

Thus we can always assume that an inequality $c^- \sum_{i\in I} \epsilon_i + c^+ \sum_{j\in J} \epsilon_j \lhd b$ (as above) has coefficient 0 for ϵ_n. Let k be a positive integer such that $kb \in \mathbb{Z}$. Then, it is easy to see that our inequality is equivalent to the following system:

$$
\begin{cases}
(kbc^-) \sum_{i\in I} \epsilon_i + (kbc^+) \sum_{j\in J} \epsilon_j + (kbc^-)\epsilon_n \lhd 0 \\
\epsilon_n \leq -\frac{b}{c^-} \\
\epsilon_n \geq -\frac{b}{c^-}
\end{cases}
$$

The first of these inequalities is expressible with a $\mathcal{P}(\supset)$ formula by the discussion above. By theorem 11 the other two inequalities are expressed by $\mathcal{P}^-$ formulae. ◂

In order to prove that $\mathcal{P}(\supset)$ is strictly more expressive than $\mathcal{P}$, we can follow a similar strategy as for separating $\mathcal{P}$ and $\mathcal{P}^-$. In other words, we use Theorem 16 together with the fact that there are signed binary probability sets that are not signed monic, as established by the following lemma.

▶ **Lemma 17.** *Let* $\overline{\mathrm{P}} \subset \Delta^{n-1}$ *be a probability set defined by a single inequality* $a_1\epsilon_1 + \cdots + a_n\epsilon_n \leq b$, *where* $0 \neq a_i, a_j \in \mathbb{Z}$ *and* $|a_i| \neq |a_j|$, *for some indices* i, j. *Then* $\overline{\mathrm{P}}$ *is not signed monic.*

Proof (sketch). The proof is analogous to that of Lemma 12, using the fact that the projection on the (i, j)-plane of the figure described in the statement is neither parallel to any axis, nor parallel or orthogonal to the segment of extremes $(0, 1), (1, 0)$. ◀

Actually, the lemma immediately yields multiple separation results.

▶ **Proposition 18.** *1)* $\mathcal{P} < \mathcal{P}(\supset)$, *2)* $\mathcal{P}(\supset) \not\leq \mathcal{P}(\square\!\!\rightarrow)$, *3)* $\mathcal{P}(\square\!\!\rightarrow) < \mathcal{PCO}$.

We are finally ready to characterize the expressive power of $\mathcal{PCO}$; the proof is analogous to that of Theorem 14.

▶ **Theorem 19.** *Let* $\mathcal{K}$ *be a class of causal multiteams of signature* σ. $\mathcal{K}$ *is definable by a* $\mathcal{PCO}_\sigma$ *formula if and only if 1) it has the empty multiteam property, 2) it is closed under rescaling, and 3)* $\mathcal{K} = \bigcup_{\mathcal{F} \in \mathbb{F}_\sigma} \mathcal{K}^{\mathcal{F}}$, *where each* $\mathcal{K}^{\mathcal{F}}$ *is a signed binary set of causal multiteams of function component* $\mathcal{F}$.

By Theorem 8, $\mathcal{PCO}^\omega$ formulae may characterize arbitrary probability sets. By Theorem 19, instead, we know that the probability sets of $\mathcal{PCO}$ formulae are all definable in terms of linear inequalities. A strict inclusion of languages immediately follows. An alternative proof for this using a counting argument was given in [4].

▶ **Corollary 20.** $\mathcal{PCO} < \mathcal{PCO}^\omega$.

4 Definability of probabilistic and dependence atoms

Next we briefly explore the relationships of our logics and the probabilistic atoms studied in probabilistic and multiteam semantics. We consider the dependence atom by Väänänen [38], and marginal distribution identity and probabilistic independence atoms by Durand et al. [10].

The dependence atom $=(\mathbf{X}; \mathbf{Y})$ expresses that the values of $\mathbf{X}$ functionally determine the values of $\mathbf{Y}$. Dependence atoms can be expressed already in $\mathcal{P}(\supset)$:

$$=(\mathbf{X}; \mathbf{Y}) := \bigwedge_{\mathbf{x} \in \mathrm{Ran}(\mathbf{X})} \bigsqcup_{\mathbf{y} \in \mathrm{Ran}(\mathbf{Y})} \mathbf{X} = \mathbf{x} \supset \mathbf{Y} = \mathbf{y}$$

The marginal distribution identity atom $\mathbf{X} \approx \mathbf{Y}$ states that the marginal distributions induced by $\mathbf{X}$ and $\mathbf{Y}$ are identical. This can be defined in $\mathcal{P}$ by

$$\mathbf{X} \approx \mathbf{Y} := \bigwedge_{\mathbf{x} \in \mathrm{Ran}(\mathbf{X}) \cap \mathrm{Ran}(\mathbf{Y})} \Pr(\mathbf{X} = \mathbf{x}) = \Pr(\mathbf{Y} = \mathbf{x}) \wedge \bigwedge_{\mathbf{x} \in \mathrm{Ran}(\mathbf{X}) \setminus \mathrm{Ran}(\mathbf{Y})} \Pr(\mathbf{X} = \mathbf{x}) = 0 \wedge \bigwedge_{\mathbf{y} \in \mathrm{Ran}(\mathbf{Y}) \setminus \mathrm{Ran}(\mathbf{X})} \Pr(\mathbf{Y} = \mathbf{y}) = 0.$$

The conditional probabilistic atoms inherit their semantics from probability theory:

$$T \models \Pr(\alpha \mid \beta) \rhd \epsilon \quad \text{iff} \quad (T^\beta)^- = \emptyset \text{ or } P_{T^\beta}(\alpha) \rhd \epsilon.$$
$$T \models \Pr(\alpha \mid \beta) \rhd \Pr(\gamma \mid \delta) \quad \text{iff} \quad (T^\beta)^- = \emptyset \text{ or } (T^\delta)^- = \emptyset \text{ or } P_{T^\beta}(\alpha) \rhd P_{T^\delta}(\beta),$$

and we may also write e.g. $\Pr(\alpha \mid \beta) \rhd \Pr(\gamma)$ as an abbreviation for $\Pr(\alpha \mid \beta) \rhd \Pr(\gamma \mid \top)$. Related to these, the atom $\mathbf{X} \perp\!\!\!\perp_{\mathbf{Z}} \mathbf{Y}$ (*conditional independence atom*) states that for any given value for the variables in $\mathbf{Z}$ the variable sets $\mathbf{X}$ and $\mathbf{Y}$ are probabilistically independent. Its special case with $\mathbf{Z} = \emptyset$ is called *marginal independence atom*. We can define these atoms in terms of conditional comparison atoms:

$$\mathbf{X} \perp\!\!\!\perp \mathbf{Y} := \bigwedge_{\substack{\mathbf{x}\in \mathrm{Ran}(\mathbf{X})\\ \mathbf{y}\in \mathrm{Ran}(\mathbf{Y})}} \Pr(\mathbf{X}=\mathbf{x}) = \Pr(\mathbf{X}=\mathbf{x} \mid \mathbf{Y}=\mathbf{y})$$

$$\mathbf{X} \perp\!\!\!\perp_{\mathbf{Z}} \mathbf{Y} := \bigwedge_{\substack{\mathbf{x}\in \mathrm{Ran}(\mathbf{X})\\ \mathbf{y}\in \mathrm{Ran}(\mathbf{Y})\\ \mathbf{z}\in \mathrm{Ran}(\mathbf{Z})}} \Pr(\mathbf{X}=\mathbf{x} \mid \mathbf{Z}=\mathbf{z}) = \Pr(\mathbf{X}=\mathbf{x} \mid \mathbf{YZ}=\mathbf{yz})$$

Hence the atoms (and the dependence atom expressed as $\mathbf{Y} \perp\!\!\!\perp_{\mathbf{X}} \mathbf{Y}$) are expressible in $\mathcal{P}$ extended with the conditional probability comparison atoms. It is an open question whether the probabilistic independence atoms are already expressible in $\mathcal{PCO}$.

The above definitions of atoms imply that our languages, if enriched with conditional probability atoms and arbitrary applications of the disjunction $\vee$, are strong enough to the express properties of multiteams that are expressible in the quantifier free fragments of the logics FO($\perp\!\!\!\perp$) (*probabilistic independence logic*) and FO($\approx$) (*probabilistic inclusion logic*), over any fixed finite structure. The expressivity and complexity of these logics have been thoroughly studied in the probabilistic and multiteam semantics literature (see [10, 11, 14, 19, 20, 21, 39]).

It was observed in [4] that $\Pr(\alpha \mid \gamma) \rhd \epsilon$ and $\Pr(\alpha \mid \gamma) \rhd \Pr(\beta \mid \gamma)$ can be defined by $\gamma \supset \Pr(\alpha) \rhd \epsilon$ and $\gamma \supset \Pr(\alpha) \rhd \Pr(\beta)$, respectively. The latter result concerns comparison atoms in which both probabilities are conditioned over the same formula, γ. We establish that this restriction is necessary, and that $\Pr(\alpha \mid \gamma) \geq \Pr(\beta \mid \delta)$ is not, in general, expressible in $\mathcal{PCO}$. The full proof of the theorem is given in the extended version of the paper ([5]).

▶ **Theorem 21.** *The comparison atoms* $\Pr(\alpha \mid \beta) \lhd \Pr(\gamma \mid \delta)$ *and* $\Pr(\alpha \mid \beta) \lhd \Pr(\gamma)$*, (where* $\lhd \in \{\leq, \geq, <, >, =\}$*) are not, in general, expressible in* $\mathcal{PCO}$*.*

Proof (sketch). Due to the equivalence $\Pr(\alpha \mid \beta) \lhd \Pr(\gamma \mid \top) \equiv \Pr(\alpha \mid \beta) \lhd \Pr(\gamma)$, it suffices to prove the theorem for $\Pr(\alpha \mid \beta) \lhd \Pr(\gamma)$.

Fix a signature σ, $\delta \in [0,1] \cap \mathbb{Q}$ and take four distinct assignments $s_i, s_j, s_k, s_l \in \mathbb{B}_\sigma$. The proof proceeds by showing that the conjunction

$$\Xi := \Pr(\hat{\alpha}_k \vee \hat{\alpha}_i \mid \hat{\alpha}_l \vee \hat{\alpha}_i) \lhd \Pr(\hat{\alpha}_l \vee \hat{\alpha}_j) \ \wedge \ \Pr(\hat{\alpha}_i) = \delta \ \wedge \ \Pr(\hat{\alpha}_j \vee \hat{\alpha}_k \vee \hat{\alpha}_l) = 1 - \delta$$

has a probability set that cannot be characterized in terms of systems of linear inequalities, and thus is not expressible in $\mathcal{P}(\supset)$; extending the result to the whole $\mathcal{PCO}$ is then straightforward. Calculation shows that any T satisfies Ξ if and only if the two-variable inequality

$$2\epsilon_k\epsilon_l + 2\delta\epsilon_k + (2\delta - 2)\epsilon_l + 2\delta^2 \rhd 0$$

holds. Standard geometric techniques (analysis of the homogeneous discriminant) tell us that, in the intersection of the (k,l)-plane with Δ^3, the frontier of the set defined by this inequality is a segment of a nondegenerate conic (a hyperbola). But, clearly, no linear set can have a segment of hyperbola as a subset of its frontier. ◀

5 Conclusion

We embarked for a comprehensive study of the expressive power of logics of probabilistic reasoning and causal inference in the unified setting of causal multiteam semantics. We focused on the logic $\mathcal{PCO}$ that can express probability comparisons in a dataset, and encompasses interventionist counterfactuals and selective implications for describing consequences of actions and consequences of learning from observations, respectively. In addition, we considered

the syntactic fragments $\mathcal{P}^-$, $\mathcal{P}$, $\mathcal{P}(\supset)$, and $\mathcal{P}(\boxminus\!\!\rightarrow)$ of $\mathcal{PCO}$ and proved that they form a strict expressivity hierarchy (see Figure 1 on page 3). Moreover, we discovered natural complete characterizations, for each of the aforementioned logics, based on the families of linear equations needed to define the corresponding classes of causal multiteams (satisfying some invariances); these results are summarized in Table 1 (on page 4). Finally, we established that conditional probability statements of the forms $\Pr(\alpha \mid \beta) \leq \Pr(\gamma \mid \delta)$ and $\Pr(\alpha \mid \beta) \leq \Pr(\gamma)$ are not in general expressible in $\mathcal{PCO}$, and separated $\mathcal{PCO}$ from its extension $\mathcal{PCO}^\omega$ with infinitary disjunctions.

Analogous to the folklore result that the logic $L_{\infty\omega}$ can define all classes of finite structures, it was shown in [4] that the same holds for $\mathcal{PCO}^\omega$ with respect to all classes of causal multiteams that are closed under *rescaling* and have the *empty multiteam property*. While any logic that is expressively complete in this sense is uncountable, it is an interesting task to identify more expressive finitary languages. We describe some future directions of research:

- In the languages we considered, the usage of the *strict tensor* $\vee$ was restricted to $\mathcal{CO}$ formulae. What impact would removing this restriction have on the expressivity of the languages? We conjecture that liberalizing this operator would allow to capture probability sets described by any linear inequality.
- Can (conditional) probabilistic independence atoms be expressed in $\mathcal{PCO}$? We conjecture the negative in line with [21, Proposition 26], which establishes that it is not expressible in FO(≈), the *probabilistic inclusion logic* of [19] (although the proof in [21] relies on the use of quantifiers).
- How can our results be extended to cover infinite signatures? Here one might need to extend the languages with quantifiers ranging over data values.
- Our characterizations cover only logics that express linear properties of data. Can we generalize our results if some natural source of multiplication, such as conditional probabilistic independence or the conditional comparison atoms, are added to the logics? It was shown by Hannula et al. [20] that the so-called *probabilistic independence logic* is equiexpressive with a variant of existential second-order logic that has access to addition and multiplication of reals.
- Finally, a promising direction for future work would be to study temporal aspects of causal inference (see e.g., [25]) via (probabilistic) temporal logics by generalising *temporal team semantics* introduced by Krebs et al. [26] and further developed by Gutsfeld et al. [15].

We conclude by pointing out the formal similarity of our work with some results obtained for first-order logics with probabilistic dependencies, such as the aforementioned language FO(≈). Such languages do not formalize causation, and yet we can conjecture that $\mathcal{PCO}$ might be embeddable in FO(≈) (similarly as the language $\mathcal{CO}$ is embedded into first-order logic in [1]). This idea is supported by a result of Hannula and Virtema ([21]) that establishes that definability in FO(≈) can be reformulated in linear programming. It is however unknown which exact fragment of linear programming corresponds (in the sense of our Table 1) to the language FO(≈); such a characterization would give precise limits to the possibility of embedding results.

References

1 Fausto Barbero and Pietro Galliani. Embedding causal team languages into predicate logic. *Annals of Pure and Applied Logic*, pages 103–159, 2022. doi:10.1016/j.apal.2022.103159.

2 Fausto Barbero and Gabriel Sandu. Interventionist counterfactuals on causal teams. In *CREST 2018 Proceedings – Electronic Proceedings in Theoretical Computer Science*, volume 286, pages 16–30. Open Publishing Association, January 2019. doi:10.4204/eptcs.286.2.

3 Fausto Barbero and Gabriel Sandu. Team semantics for interventionist counterfactuals: observations vs. interventions. *Journal of Philosophical Logic*, 50:471–521, 2021.

4 Fausto Barbero and Gabriel Sandu. Multiteam semantics for interventionist counterfactuals: probabilities and causation. pre-print, 2023. arXiv:2305.02613.

5 Fausto Barbero and Jonni Virtema. Expressivity landscape for logics with probabilistic interventionist counterfactuals. *CoRR*, abs/2303.11993, 2023. doi:10.48550/arXiv.2303.11993.

6 Fausto Barbero and Jonni Virtema. Strongly complete axiomatization for a logic with probabilistic interventionist counterfactuals. In Sarah Gaggl, Maria Vanina Martinez, and Magdalena Ortiz, editors, *Logics in Artificial Intelligence*, pages 649–664, Cham, 2023. Springer Nature Switzerland.

7 Fausto Barbero and Fan Yang. Characterizing counterfactuals and dependencies over (generalized) causal teams. *Notre Dame Journal of Formal Logic*, 63(3), 2022. doi:10.1215/00294527-2022-0017.

8 Elias Bareinboim, Juan Correa, Duligur Ibeling, and Thomas Icard. On Pearl's hierarchy and the foundations of causal inference (1st edition). In Hector Geffner, Rina Dechter, and Joseph Y. Halpern, editors, *Probabilistic and Causal Inference: the Works of Judea Pearl*, pages 507–556. ACM Books, 2022.

9 Rachael Briggs. Interventionist counterfactuals. *Philosophical Studies: An International Journal for Philosophy in the Analytic Tradition*, 160(1):139–166, 2012.

10 Arnaud Durand, Miika Hannula, Juha Kontinen, Arne Meier, and Jonni Virtema. Approximation and dependence via multiteam semantics. *Ann. Math. Artif. Intell.*, 83(3-4):297–320, 2018. doi:10.1007/s10472-017-9568-4.

11 Arnaud Durand, Miika Hannula, Juha Kontinen, Arne Meier, and Jonni Virtema. Probabilistic team semantics. In Flavio Ferrarotti and Stefan Woltran, editors, *Foundations of Information and Knowledge Systems*, pages 186–206, Cham, 2018. Springer International Publishing.

12 Arnaud Durand, Juha Kontinen, and Heribert Vollmer. Expressivity and complexity of dependence logic. *Dependence Logic: Theory and Applications*, pages 5–32, 2016.

13 David Galles and Judea Pearl. An axiomatic characterization of causal counterfactuals. *Foundations of Science*, 3(1):151–182, January 1998.

14 Erich Grädel and Richard Wilke. Logics with multiteam semantics. *ACM Trans. Comput. Log.*, 23(2):13:1–13:30, 2022. doi:10.1145/3487579.

15 Jens Oliver Gutsfeld, Arne Meier, Christoph Ohrem, and Jonni Virtema. Temporal team semantics revisited. In Christel Baier and Dana Fisman, editors, *LICS '22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2 - 5, 2022*, pages 44:1–44:13. ACM, 2022. doi:10.1145/3531130.3533360.

16 Joseph Halpern. *Actual causality*. MIT Press, 2016.

17 Joseph Y. Halpern. Axiomatizing causal reasoning. *J. Artif. Int. Res.*, 12(1):317–337, May 2000.

18 Joseph Y. Halpern. From causal models to counterfactual structures. *Review of Symbolic Logic*, 6(2):305–322, 2013. doi:10.1017/s1755020312000305.

19 Miika Hannula, Åsa Hirvonen, Juha Kontinen, Vadim Kulikov, and Jonni Virtema. Facets of distribution identities in probabilistic team semantics. In *European Conference on Logics in Artificial Intelligence*, pages 304–320. Springer, 2019.

20 Miika Hannula, Juha Kontinen, Jan Van den Bussche, and Jonni Virtema. Descriptive complexity of real computation and probabilistic independence logic. In Holger Hermanns, Lijun Zhang, Naoki Kobayashi, and Dale Miller, editors, *LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020*, pages 550–563. ACM, 2020. doi:10.1145/3373718.3394773.

21 Miika Hannula and Jonni Virtema. Tractability frontiers in probabilistic team semantics and existential second-order logic over the reals. *Ann. Pure Appl. Log.*, 173(10):103108, 2022. doi:10.1016/j.apal.2022.103108.

22 James J Heckman and Edward J Vytlacil. Econometric evaluation of social programs, part i: Causal models, structural models and econometric policy evaluation. *Handbook of econometrics*, 6:4779–4874, 2007.

23 MA Hernan and J Robins. *Causal Inference: What if*. Boca Raton: Chapman & Hill/CRC, forthcoming.

24 Wilfrid Hodges. Compositional semantics for a language of imperfect information. *Logic Journal of the IGPL*, 5:539–563, 1997.

25 Samantha Kleinberg. A logic for causal inference in time series with discrete and continuous variables. In Toby Walsh, editor, *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 943–950. IJCAI/AAAI, 2011. doi:10.5591/978-1-57735-516-8/IJCAI11-163.

26 Andreas Krebs, Arne Meier, Jonni Virtema, and Martin Zimmermann. Team semantics for the specification and verification of hyperproperties. In Igor Potapov, Paul G. Spirakis, and James Worrell, editors, *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018, August 27-31, 2018, Liverpool, UK*, volume 117 of *LIPIcs*, pages 10:1–10:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.MFCS.2018.10.

27 David Lewis. *Counterfactuals*. Oxford: Blackwell Publishers, 1973.

28 Stephen L Morgan and Christopher Winship. *Counterfactuals and causal inference*. Cambridge University Press, 2015.

29 Milan Mossé, Duligur Ibeling, and Thomas Icard. Is causal reasoning harder than probabilistic reasoning? *The Review of Symbolic Logic*, pages 1–26, 2022.

30 Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, New York, NY, USA, 2000.

31 Judea Pearl and Dana Mackenzie. *The book of why: the new science of cause and effect*. Basic books, 2018.

32 Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of causal inference: foundations and learning algorithms*. MIT Press, 2017.

33 Bernhard Schölkopf. Causality for machine learning. In *Probabilistic and Causal Inference: The Works of Judea Pearl*, pages 765–804. Association for Computing Machinery, 2022.

34 Peter Spirtes. Directed cyclic graphical representations of feedback models. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 491–498, 1995.

35 Peter Spirtes, Clark Glymour, and Richard N. Scheines. *Causation, Prediction, and Search*, volume 81 of *Lecture Notes in Statistics*. Springer New York, 1993.

36 Robert C. Stalnaker. A theory of conditionals. *Americal Philosophical Quarterly*, pages 98–112, 1968.

37 Robert H Strotz and Herman OA Wold. Recursive vs. nonrecursive systems: An attempt at synthesis (part i of a triptych on causal chain systems). *Econometrica: Journal of the Econometric Society*, pages 417–427, 1960.

38 Jouko Väänänen. *Dependence Logic: A New Approach to Independence Friendly Logic*, volume 70 of *London Mathematical Society Student Texts*. Cambridge University Press, 2007.

39 Richard Wilke. On the Presburger fragment of logics with multiteam semantics. *Ann. Pure Appl. Log.*, 173(10):103120, 2022. doi:10.1016/j.apal.2022.103120.

40 Sewall Wright. Correlation and causation. *Journal of agricultural research*, 20:557–585, 1921.

41 Jiji Zhang. A Lewisian logic of causal counterfactuals. *Minds and Machines*, 23(1):77–93, 2013.

A General Constructive Form of Higman's Lemma

Stefano Berardi
Dipartimento di Informatica, Università di Torino, Italy

Gabriele Buriola
Dipartimento di Informatica, Università di Verona, Italy

Peter Schuster
Dipartimento di Informatica, Università di Verona, Italy

Abstract

In logic and computer science one often needs to constructivize a theorem $\forall f \exists g.P(f,g)$, stating that for every infinite sequence f there is an infinite sequence g such that $P(f,g)$. Here P is a computable predicate but g is not necessarily computable from f. In this paper we propose the following constructive version of $\forall f \exists g.P(f,g)$: for every f there is a "long enough" finite prefix g_0 of g such that $P(f,g_0)$, where "long enough" is expressed by membership to a bar which is a free parameter of the constructive version.

Our approach with bars generalises the approaches to Higman's lemma undertaken by Coquand–Fridlender, Murthy–Russell and Schwichtenberg–Seisenberger–Wiesnet. As a first test for our bar technique, we sketch a constructive theory of well-quasi orders. This includes yet another constructive version of Higman's lemma: that every infinite sequence of words has an infinite ascending subsequence. As compared with the previous constructive versions of Higman's lemma, our constructive proofs are closer to the original classical proofs.

2012 ACM Subject Classification Theory of computation → Constructive mathematics; Theory of computation → Proof theory; Mathematics of computing → Discrete mathematics

Keywords and phrases intuitionistic logic, constructive mathematics, formal proof, inductive predicate, bar induction, well quasi-order, Higman's lemma

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.16

Acknowledgements The present study was started while the third author worked within the project "Reducing complexity in algebra, logic, combinatorics – REDCOM" belonging to the programme "Ricerca Scientifica di Eccellenza 2018" of the Fondazione Cariverona. The second and the third author are members of the "Gruppo Nazionale per le Strutture Algebriche, Geometriche e le loro Applicazioni" (GNSAGA) of the Istituto Nazionale di Alta Matematica (INdAM). The authors wish to thank Daniel Fridlender for his interest and suggestions, and to express their gratitude to Thierry Coquand, whose original idea set the basis for this work. Last but not least, the anonymous reviewers' meticulous reading and constructive critique have been extremely helpful.

1 Introduction: Higman's Lemma in Constructive Mathematics

By "Higman's lemma for sequences" we understand the following statement: "If Σ is a finite alphabet, then for every infinite sequence $\sigma = a_0, a_1, a_2, \ldots$ of words over Σ, there exists an infinite subsequence$\tau = a_{i_0}, a_{i_1}, a_{i_2} \ldots$ of σ such that $a_{i_0} \sqsubseteq a_{i_1} \sqsubseteq a_{i_2}, \ldots$, where $\sqsubseteq$ denotes the embedding order on words" (see later for more details). Given our focus on finite alphabet, Σ will always denote such a set. Although Higman's lemma for sequences has a well-known classical proof, i.e. with classical logic, one can easily check that it has no constructive proofs; the selection of the weakly increasing, **w.i.** for short, subsequence cannot be made recursively in general.

By simply taking the first two elements of any infinite w.i. subsequence, Higman's lemma for sequences entails that every infinite sequence $a_0, a_1, a_2, \ldots$ of words over a finite alphabet has a w.i. subsequence of length 2, i.e. there are $i_0 < i_1$ for which $a_{i_0} \sqsubseteq a_{i_1}$. This is

one of the consequences of Higman's lemma for sequences that are provable already with intuitionistic logic: Murthy and Russell [15] applied combinatorial techniques over finite sequences; Richman and Stolzenberg [22] proved it for decidable well quasi-orders using hereditary (inductive) well-foundedness; Coquand and Fridlender [7] resorted to inductive definitions for the binary case and Seisenberger [24] extended the same approach to the general case; Schwichtenberg, Seisenberger and Wiesnet [23] extracted the computational content of Higman's lemma; and Powell [19] applied Gödel's Dialectica interpretation.

There are more examples along the same lines. We can ask for a w.i. subsequence of length k for any $k \in \mathbb{N}$ whatsoever, for which we take the first k elements of any w.i. infinite subsequence. But we do not have to stop here; for example, we can construct a non-empty w.i. subsequence of length $\mathtt{len}(a_{i_0})+1$. In this case we take the first element of the w.i. infinite subsequence, compute its length $n = \mathtt{len}(a_{i_0})$ as a word, and then take n more elements from the w.i. infinite subsequence. More generally, for every functional F from infinite sequences to $\mathbb{N}$ there is an infinite sequence σ having an infinite subsequence τ the first $F(\tau)$ elements of which are in weakly increasing order. We will deduce all these statements from an even stronger theorem, the Higman lemma for bars, which we will state and prove with intuitionistic logic.

The constructive history of Higman's lemma

The theory of well quasi-orders has found applications in many different fields, and so has Higman's lemma, one of this theory's milestones. Given the concrete character of Higman's lemma especially in the case of a finite alphabet, and its applicability in computer science, the search for a constructive and more perspicuous proof has started very early: not only to make possible program extraction from proofs, but also for a better understanding both of the original non-constructive proof of Higman's lemma and the short and elegant but still non-constructive proof by Nash-William [16]. To position the results of this paper in the literature, we now briefly survey the existing constructive approaches to Higman's lemma and related results such as Kruskal's theorem. For an historical survey of well quasi-order broadly understood we refer to [13].

The presumably first constructive proof of Higman's lemma was obtained by Murthy and Russell [15] using a smart manipulation of finite strings. Richman and Stolzenberg [22] then proved Higman's lemma by induction on subsets. Coquand and Fridlender [7] instead used structural induction over inductive definitions; their results were extended by Seisenberger [24]. Fridlender [9] gave a type-theoretic version of Higman's lemma, and Veldman [28] an inductive intuitionistic proof. Worthy of mention is Berger's constructive proof [3] of the equivalence between Dickson's lemma and Higman's lemma for a two-element alphabet.

The connection between Higman's lemma and programs has been addressed several times. Schwichtenberg, Seisenberger, and Wiesnet [23] analyzed the computational content of Higman's lemma. Powell has successfully applied Gödel's Dialectica interpretation to well quasi-orders [20] and Higman's lemma [19]. Concerning computer-assisted theorem proving, Berghofer [4], has formalized a constructive proof of Higman's lemma in Isabelle, starting from the article by Coquand and Fridlender; more recently, Sternagel [26] used open induction to obtain a proof in Isabelle/HOL.

Finally, also Kruskal's theorem [12], the natural extension of Higman's lemma from strings to finite trees, has been put under constructive scrutiny by Veldman [28] and Seisenberger [25], whereas Goubault-Larrecq [10] gave a topological constructive version of Kruskal's theorem.

A Constructive Form of Higman's Lemma – Now for Bars

Classically, a bar B for lists is a set of finite lists such that every infinite chain of one-step extensions meets B, i.e. has an element in B. Within intuitionistic logic we need and have a more perspicuous definition of bar (see later). Higman's lemma for bars now says: "for every bar B, every infinite sequence σ of words on a finite alphabet Σ has an infinite subsequence τ with a weakly increasing prefix τ_0 in B". Here we interpret $\tau_0 \in B$ as that τ_0 is "long enough for our purposes", with "our purposes" expressed by the choice of B.

E.g. if B is the set of lists of length 2 or more, then Higman's lemma for bars entails that every infinite sequence σ of words on Σ has an infinite subsequence τ with a weakly increasing prefix τ_0 in B, i.e. of two or more elements. This is the first of the desired consequences of Higman's lemma, all of which can be deduced from Higman's lemma for bars.

We will prove Higman's lemma for bars with intuitionistic logic. In fact, we prove a stronger version in which the requirement "σ is infinite" is replaced by one about the bar B. During our proof, we are able to constructively interpret several non-constructive classical theorems of the following form: for every sequence f there is an infinite sequence g such that $P(f, g)$. Higman's lemma for sequences is of course a typical case.

For instance, we rephrase the notion of wqo (well quasi-order), the main ingredient of the classical proof of Higman's lemma, by quantification on bars; and call "wqo(bar)" this novel notion of wqo. We have short proofs with intuitionistic logic that the concept wqo(bar) is closed by unions (provided that the union is a preorder), by products and by right-invertible morphisms; these are all properties of wqo typically occurring in a classical proof of Higman's lemma. With the notion of wqo(bar) at hand, we consider possible to develop a constructive version of the theory of wqo close to the classical one.

The structure of the article is as follows. In Section §2 we briefly recall the words and sequences terminology; Section §3 is devoted to prove several properties of bars used in the rest of the paper; in Section §4 we state Higman's lemma for bars and prove the desired corollaries; in Section §5, after some closure properties of the concept wqo(bar), we prove Higman's lemma for bars.

2 Lists, Words and Sequences

We start by recalling some well-known terminology about lists, sublists and labels (§2.1), as well as notions related to alphabets and words (§2.2). This part is straighforward, and the reader may want to jump to the subsection about the anticone of a word (§2.3).

2.1 Lists and Operations on Lists

Let $\mathbb{N}$ be the set of natural numbers and I a given set. We call a *list* l on I any map l such that $l : [0, n[\to I$ for some $n \in \mathbb{N}$ or $l : \mathbb{N} \to I$. We set $\mathtt{dom}(l) = [0, n[$, $\mathtt{range}(l) = l([0, n[)$ in the first case and $\mathtt{dom}(l) = \mathbb{N}$, $\mathtt{range}(l) = l(\mathbb{N})$ in the second case; moreover, we call $\mathtt{dom}(l)$ the set of indexes of l and $\mathtt{range}(l)$ the set of elements of l, abbreviating $i \in \mathtt{range}(l)$ by $i \in l$. The length of l, denoted $\mathtt{len}(l)$, is $n \in \mathbb{N}$ in the first case and is ∞ (infinite) in the second case; in the first case we say that the list l is finite and in the second case that l is infinite. We call each $x \in \mathtt{range}(l)$ an element of l and write $\mathtt{Fin}(I)$ for the set of finite lists on I, $\mathtt{Inf}(I)$ for the set of infinite lists, and $\mathtt{List}(I) = \mathtt{Fin}(I) \cup \mathtt{Inf}(I)$ for the set of all lists on I.

We write a finite list $l \in \mathtt{Fin}(I)$ of length $n = \mathtt{len}(l)$ as $\langle l(0), \dots, l(n-1) \rangle$, denoting by $\mathtt{Nil} = \langle\rangle$ the empty list, the unique list of length 0. For $l, m \in \mathtt{List}(I)$, we write $l \sqsubseteq m$, or "l is a sublist of m" for: there is a finite increasing list $f : [0, \mathtt{len}(l)[\to [0, \mathtt{len}(m)[$ of natural

numbers such that $l(i) = m(f(i))$ for all $i \in [0, \mathtt{len}(a)[$; we call such an f an *embedding* of l in m. For instance, if I is the English alphabet, if $l = \langle w, o, r, d\rangle$, representing the word "word", and $m = \langle w, o, r, l, d\rangle$, representing the word "world", then $l \sqsubseteq m$. An embedding of l in m is $f : [0, 4[\to [0, 5[$ defined by $f(0) = 0$,$f(1) = 1$, $f(2) = 2$ and $f(3) = 4$. $\mathtt{range}(f)$ does not include 3, the index of the symbol "l" in "world". Another example: $l : \mathbb{N} \to \mathbb{N}$ defined by $a(i) = 2i$ is the list of all even numbers, $m : \mathbb{N} \to \mathbb{N}$ defined by $m(i) = i$ is the list of all natural numbers, and $f(i) = 2i$ is an embedding from l to m. Roughly speaking, we have $l \sqsubseteq m$ if and only if we can obtain l by skipping zero or more elements from m, without changing the order of the elements of b.

If I, J are sets, then $I \times J = \{\langle x, y\rangle \mid x \in I, y \in J\}$ denotes their Cartesian product. A *binary relation* R on I, J is given by a subset of $I \times J$ and we write R^{-1} for the inverse binary relation $\{\langle y, x\rangle \in J \times I \mid \langle x, y\rangle \in R\}$, using the notation $R(x, y)$ or xRy for $\langle x, y\rangle \in R$. If X is a set, then the *R-upward cone* of X, denoted $R(X)$, is the set of $y \in J$ such that $\exists x \in X.R(x, y)$; often abbreviating "upward cone" by "cone". If $x \in I$, we write $R(x)$ for $R(\{x\})$, and we call $R(x)$ the R-cone of x in J; moreover, we call $J \setminus R(x)$ the *anticone* of x in J: it is the cone of x with respect to the complement in $I \times J$ of the relation R.

We write $\sqsubseteq_{I,J}$ for the binary relation $\{\langle l, m\rangle \in \mathtt{Fin}(I) \times \mathtt{Fin}(J) \mid l \sqsubseteq m\}$ defined by the sublist predicate restricted to $\mathtt{Fin}(I) \times \mathtt{Fin}(J)$ and we write $\sqsubseteq_I$ for $\sqsubseteq_{I,I}$ and $\sqsupseteq_{I,J}$ for the inverse binary relation $\{\langle l, m\rangle \in \mathtt{Fin}(I) \times \mathtt{Fin}(J) \mid l \sqsupseteq m\}$

For $i \in \mathbb{N}$, we define the restriction $l\lceil i \in \mathtt{Fin}(I)$ of l to $[0, i[$ by $(l\lceil i)(j) = l(j)$ for all $j < \mathtt{len}(l), j < i$ and $\mathtt{len}(l\lceil i) = \min(\mathtt{len}(l), i)$. When l is a restriction of some (possibly infinite) list m, then we say that l is a *prefix* of m and we write $l \leqslant m$.

We define the concatenation $m = l{\star}l'$ of two lists l, l' with l finite by: $m(i) = l(i)$ for all $i < \mathtt{len}(l)$ and $m(\mathtt{len}(l) + j) = l'(j)$ for all $j < \mathtt{len}(l')$. By definition we have $\langle l(0), \ldots, l(n-1)\rangle{\star}\langle l'(0), \ldots, l'(m-1), \ldots\rangle = \langle l(0), \ldots, l(n-1), l'(0), \ldots, l'(n'-1), \ldots\rangle$. The length of m is $\mathtt{len}(l) + \mathtt{len}(l')$ if l' is finite and ∞ if l' is infinite. If $m = l{\star}l'$ for some $l, l' \in \mathtt{Fin}(I)$, then we say that l' is a *suffix* of m.

Given $l, m \in \mathtt{Fin}(I)$, we write $l <_1 m$ if $m = l{\star}\langle i\rangle$ for some $i \in I$, i.e. m is obtained by adding the element i to the end of l. We write $<$ for the transitive closure of the relation $<_1$. Remark that the prefix relation $\leqslant$ is the reflexive closure of $<$.

2.2 Alphabet, Words and Sequences

A finite alphabet Σ is any finite set Σ in bijection with $[0, n[$ for some n and through some map f. Equality on Σ is provably decidable with intuitionistic logic, because $i = j$ in Σ if and only if $f(i) = f(j)$ in $\mathbb{N}$. We call the elements of Σ "symbols" of the alphabet, and we denote them with the letters a, b, c and their variants, $a', a_1, \ldots$ the basic example is $\Sigma = \{0, 1\}$. A *word* on Σ is given by a finite list on Σ and we write $\Sigma^* = \mathtt{Fin}(\Sigma)$ for the set of words on Σ. We use $\mathtt{nil}$ for the empty word in Σ^*, this is just another name for $\mathtt{Nil} = \langle\rangle$, and we denote words with the letters v, w, z and their variants, $v', v_1, \ldots$; moreover, with a slight and harmless abuse of notation, we use the expression $c{\in}w$ and $c{\notin}w$ to denote respectively that c is, or is not, one of the letters of w. A "symbol" could be anything, therefore we could use "finite set" for "alphabet" and "finite list on a finite set" for word, but it is customary to use "alphabet" and "word" in the context of Higman's lemma, because the intended application of the Lemma are the words of an alphabet. If $v, w \in \Sigma^*$, when $v \sqsubseteq w$ we say that v is a subword of w and w a superword of v.

We introduce abbreviations used only for words. If $c_1, \ldots, c_n \in \Sigma$, we abbreviate the word $w = \langle c_1, \ldots, c_n\rangle$ with $w = c_1 \ldots c_n$, written without spaces inside. If $v, w \in \Sigma^*$, we abbreviate $v{\star}w$ with the juxtaposition vw. If $c \in \Sigma$ and $w \in \Sigma^*$, we abbreviate $\langle c\rangle{\star}w$ by cw, and $w{\star}\langle c\rangle$ by wc.

We call a "sequence of words" on Σ, just a "sequence" for short, any list on Σ^*. A sequence is finite if it is a finite list and it is infinite if it is an infinite list. Again, we could use "list of words" as well, but it is customary to say "sequence". Within this terminology, $\texttt{Fin}(\Sigma^*)$ and $\texttt{Inf}(\Sigma^*)$ are the set of finite and infinite sequences on Σ^*. Finally, we adopt the following notation rule: finite sequences are denoted by Latin letters, whereas infinite sequences by Greek letters.

2.3 Anticone and Slice of a Word

In this subsection we characterize the words which are superwords of a given word and those which are not. Let us fix $v \in \Sigma^*$. We recall that $\not\sqsubseteq_\Sigma(v)$ denotes the anticone of v, which is the set of all $w \in \Sigma^*$ such that $v \not\sqsubseteq w$. The first step in our proof of Higman's lemma is to characterize the words in the anticone of v. To this aim, we need one preliminary step. We introduce a smaller set of words $\texttt{Slice}_\Sigma(v) \subseteq \Sigma^*$, dubbed the *slice* of v, consisting of all words $w \in \Sigma^*$ for which v is minimal among the words not embeddable in w.

▶ **Definition 1** (Slice of v). *For each word $v \in \Sigma^*$ we define $\texttt{Slice}_\Sigma(v)$ as the set of words in Σ^* which are superwords of all $v' < v$, but are not superwords of v.*

We characterize the words in $\texttt{Slice}_\Sigma(v)$. We have $\texttt{Slice}_\Sigma(\texttt{nil}) = \emptyset$, because all words are superlists of $\texttt{nil}$. Assume that $v = c_0 \dots c_{k-1}$ is not empty, that is, that $k \geqslant 1$. Then by definition unfolding we have:

$$\texttt{Slice}_\Sigma(v) = \{w \in \Sigma^* \mid (c_0 \dots c_{k-2} \sqsubseteq w) \wedge (c_0 \dots c_{k-1} \not\sqsubseteq w)\}$$

To say otherwise, $\texttt{Slice}_\Sigma(v) = \sqsubseteq_\Sigma(c_0 \dots c_{k-2}) \cap \not\sqsubseteq_\Sigma(c_0 \dots c_{k-1})$, which is the set of words in Σ^* which are superwords of $c_0 \dots c_{k-2}$ but not of $c_0 \dots c_{k-1}$. We provide a detailed description of words in $\texttt{Slice}_\Sigma(v)$. Let us abbreviate $\Sigma_i = \Sigma \setminus \{c_i\}$: then Σ_i^* is the set of $w \in \Sigma^*$ such that $c_i \notin w$. We will prove that the words in $\texttt{Slice}_\Sigma(v)$ are exactly all words of the form $w = w_0 c_0 w_1 c_1 \dots c_{k-2} w_{k-1}$, such that $c_i \notin w_i$, that is, such that $w_i \in \Sigma_i^*$, for all $i < k$. Such a decomposition will be unique and, for all $i < k$, it will define a map α_i such that $w_i = \alpha_i(w)$. We first prove that we have a slightly different decomposition for the words of the cone of v, then we prove the required decomposition for the words of $\texttt{Slice}_\Sigma(v)$.

▶ **Lemma 2** (Characterization of cone and of slice). *Let $v = c_0 \dots c_{k-1}$, $w \in \Sigma^*$.*

1. Cone. *If v is embedded in w through f, then there is a unique decomposition $w = w_0 c_0 w_1 c_1 \dots w_{k-1} c_{k-1} w_k$, such that $c_i \notin w_i$, for all $i < k$. We have no requirement for w_k. Furthermore, if*

   $$g(i) = \texttt{len}(w_0 c_0 w_1 c_1 \dots c_{i-1} w_i)$$

   for all $i < k$, then g is the minimum embedding of v in w in the point-wise ordering: $g(i) \leqslant f(i)$ for all $i < k$.
2. Slice. *If $k \geqslant 1$, then $\texttt{Slice}_\Sigma(v)$ is the set of all words w such that $w = w_0 c_0 \dots w_{k-2} c_{k-2} w_{k-1}$ and $c_i \notin w_i$ for all $i < k$. The decomposition of w if it exists it is unique.*

From the uniqueness of the decomposition of $w \in \texttt{Slice}_\Sigma(v)$ we define the maps $\alpha_i(w)$ for $i < \texttt{len}(v)$.

▶ **Definition 3** (The maps α_i). *Assume that $v = c_0 \dots c_{k-1}$, $k \geqslant 1$ and $i \in \mathbb{N}$, $i < k$. Let us abbreviate $\Sigma_i = \Sigma \setminus \{c_i\}$. Assume that $w = w_0 c_0 \dots w_{k-2} c_{k-2} w_{k-1}$ and $c_i \notin w_i$ for all $i < k$. We define $\alpha_i : \texttt{Slice}_\Sigma(v) \to \Sigma_i^*$ by $\alpha_i(w) = w_i$.*

If X and Y are sets with binary relations R and S, respectively, then by a *morphism* $f : (X, R) \to (Y, S)$ we understand a map $f : X \to Y$ such that if xRx', then $f(x)Sf(x')$.

The "product" α of the α_i in Def. 3 defines a bijection, whose inverse is a morphism for $\sqsubseteq$; the map α plays a crucial role in the proof of Higman's lemma.

▶ **Lemma 4** (Product map and Slices). *The product map* $\alpha = \alpha_1 \times \ldots \times \alpha_k : \mathtt{Slice}_\Sigma(v) \to \Sigma_0^* \times \ldots \times \Sigma_{k-1}^*$, *defined as* $\alpha(w) = (\alpha_0(w), \ldots, \alpha_{k-1}(w))$, *is a bijection. Its inverse* α^{-1} *is a morphism from* $(\Sigma_0^* \times \ldots \times \Sigma_{k-1}^*, \sqsubseteq \times \ldots \times \sqsubseteq)$ *to* $(\mathtt{Slice}_\Sigma(v), \sqsubseteq)$.

Now we can characterize the anticone $\not\sqsubseteq_\Sigma(v)$ as a finite union of slices $\mathtt{Slice}_\Sigma(v')$.

▶ **Lemma 5** (Anticone). $\not\sqsubseteq_\Sigma(v)$ *is the union of all* $\mathtt{Slice}_\Sigma(v')$ *for* $v' \leqslant v$.

These are all the properties we need about words, for what concerns bars we refer to the next section.

3 Bars: Definition and Properties

In this section we define bars and their related notions, proving with intuitionistic logic the properties required in the rest of the paper. The strongest property says that the Cartesian product of barred sets is barred by the union of the inverse image of the two projections. It is worth noticing that if we consider the empty bar, then from each result in this section about bars (except for "monotonicity", which only makes sense for bars) we obtain some well-known result about well-founded sets.

3.1 Quasi-orders, Labels, Well-founded Relations and Bars

A *quasi-order* $(P, \leqslant)$ is a set P with a transitive and reflexive relation $\leqslant$; a quasi-order $(P, \leqslant)$ is a *partial order* if $\leqslant$ is antisymmetric. A sequence $(p_k)_k$, finite or infinite, over $(P, \leqslant)$ is *weakly increasing*, for short w.i., if, for every indices $i \leqslant j$, we have $p_i \leqslant p_j$.

A *labelling* of I on P is a map $\varphi : I \to P$. A length n list $l = \langle l(0), \ldots, l(n-1)\rangle \in \mathtt{Fin}(I)$ can be turned into a list $\varphi l = \langle \varphi l(0), \ldots, \varphi l(n-1)\rangle \in \mathtt{Fin}(P)$ on P, by composing with the labelling φ of I. When $I = P$, we also consider the identical label $\varphi = \mathtt{id}$, in which the list of labels of a list is the list itself. We write $\mathtt{Incr}(\leqslant, \varphi, I)$ for the set of finite lists $l \in \mathtt{Fin}(I)$ such that φl is a weakly increasing list in P with respect to $\leqslant$.

We say that $B \subseteq \mathtt{Fin}(I)$ is $<_1$*-closed*, or closed by one-step extension, if for all $l \in B$, $la <_1 m$ we have $m \in B$. Being closed by one-step extension is the same than being closed by $\leqslant$ (by extension).

We define now the notions of a (hereditarily) well-founded set (see for instance [14,17,18]) and of a barred set, both given with respect to a given binary relation R. Our definitions are classically equivalent to the definition "all R-decreasing sequences intersect the bar" but in intuitionistic logic they allow to derive more results. Our bars generalize Troelstra's definition of bar ([27], page 77, Def. 1.9.20).

We notice that the word *inductive* is often used as a synonimous of *hereditary*.

▶ **Definition 6** (Well-founded and Barred Sets). *Let* P, X, B *be sets and* R *be a binary relation.*

1. P *is* X, R-hereditary *whenever, for all* $x \in X$, *if for all* $x' \in X$ *with* $x'Rx$ *we have* $x' \in P$, *then* $x \in P$.
2. X *is* R-well-founded *if for all* P X, R*-hereditary such that* $P \subseteq X$ *we have* $P = X$.
3. B bars X, R *if for all* P X, R*-hereditary such that* $B \cap X \subseteq P \subseteq X$ *we have* $P = X$.
4. B bars x *in* X, R *if for all* P X, R*-hereditary such that* $B \cap X \subseteq P \subseteq X$ *we have* $x \in P$.

Some comments on these definitions are in order. We already stressed that "X, R-hereditary" is exactly "X, R-inductive". This is the word chosen for instance in [2]. Next, we remark that B bars X, R if and only if B bars x in X, R for every $x \in X$.

In general, the subset consisting of the $x \in X$ such that B bars x in X, R is defined as the intersection of all X, R-hereditary $P \subseteq X$ such that $B \cap X \subseteq P$; and one can easily check that this intersection itself is X, R-hereditary. Hence "B bars x in X, R" coincides with the predicate $B \cap X \mid x$ from [7, 8]: that is, the least X, R-hereditary predicate on X which contains $B \cap X$. B is often called the *inductively defined predicate* from X, R.

So "B bars x in X, R" can be interpreted as "x is accessible from B in X, R": for $B = \emptyset$ this is nothing but the *accessibility predicate* from [5]. Accordingly, X is R-well-founded if and only if X, R is barred by $B = \emptyset$, or barred by any B such that $B \cap X = \emptyset$.

In Troelstra ([27], page 77, Def. 1.9.20) the definition of bar is given with $X =$ the set of all lists of natural numbers and $R =$ the one-step extension; it is also assumed that B is either decidable or closed by extensions. But the main difference is that the definition of bar is given as in classical mathematics, B is a bar if all infinite lists of natural numbers have some prefix in B. Instead, we defined B as the intersection of all X, R-hereditary properties, since we find this version more suitable for constructive proofs; this is the typical definition in the context of generalized inductive definitions [1, 21].

In the case we do not mention it, by R we mean $>_1$, the reverse of the one-step extension relation. In this case we say that B bars l in X, respectively that B bars X, meaning that B bars l in $X, >_1$, respectively that B bars $X, >_1$.

A subset B of X is said to be *R-downward-closed* if, for all $x \in B$, if yRx, then $y \in B$. We have a puzzling point to stress here, if $R =>_1$, then R-downward-closed in fact means that for all $x \in B$ if $y >_1 x$, then $y \in B$. That is, "R-downward-closed" in this case means "closed by one-step *extensions*". The reason is that in the literature, set of lists are often used to represent trees, and in the case of trees, it is customary to consider "smaller" a one-step-extension of a node of a tree, i.e. downward trees. We will still use the word "downward-closed" in this case, because it is a well-established terminology for inductive reasoning, but we will point out that "downward-closed" in this case means "closed by one-step extensions".

A last warning. In our definition, bars for set of lists *do not have to be closed by extensions*. For instance, the set B of all finite lists on I having odd length is a bar for the set of all lists on I and $>_1$, because each list is either odd and barred by B, or has all one-step extensions odd and barred B, and in this case is barred because being barred is an hereditary predicate. Yet, each one-step extension of a list in B is some even length list, which is not in B. Closure of a bar for a set of lists by list extension is an useful feature in some proofs, nevertheless it is not strictly required in most cases.

3.2 Basic Properties of Bars

In this subsection, we derive some basic properties for bars, requiring little more than definition unfolding.

An *R-descending chain* in X is a finite or infinite list $x_0 R^{-1} x_1 R^{-1} x_2 R^{-1} \dots$ of elements of X. For instance, a $<$-descending chain in $\mathbb{N}$ is any (necessarily finite) list $x_0 > x_1 > x_2 > \dots$ of natural numbers. We will prove that if B bars X, R, then every infinite R-descending chain in X intersects B. Using classical Logic, and some choice, the two properties are equivalent; but within intuitionistic logic, we only have the implication from the former to the latter.[1]

[1] We sketch a folk-lore proof. There is a model of Intuitionistic Logic in which all chain are recursive, while some order $<$ on some X has all infinite recursive $<$-descending chain finite and some non-recursive

▶ **Proposition 7** (Infinite R-descending chains). *Let X, B be sets and R be a binary relation.*
1. *X is X, R-hereditary.*
2. *The intersection $\cap\mathcal{F}$ of any inhabited family $\mathcal{F}$ of X, R-hereditary sets is X, R-hereditary.*
3. *the predicate "B bars x in X, R" on $x \in X$ is between $B \cap X$ and X and it is itself X, R-hereditary.*
4. *If B bars X, R, then every infinite R-descending chain in X intersects B in an infinite set of indexes.*

If B bars X, R, then we can prove that a property $P \subseteq X$ holds for all $x \in X$ by bar-induction on B, X, R. Bar-induction is the following principle. Assume that $P \subseteq X$ and: *(i. base case)* for all $x \in B \cap X$ we have $x \in P$; *(ii. inductive case)* if for all yRx, $y \in X$ we have $y \in P$, then $x \in P$. Then we conclude that $P = X$. As an example, Proposition 7.4 is proved by bar-induction on B, X, R.

We give an interpretation of a proof by bar-induction of some property P on X. We have to think of $B \cap X$ as the set of elements for which we can prove the property P directly. The one-step extension yRx of a sequence x are all elements "smaller" than x and in the inductive step of bar-induction, we have proved that if all elements "smaller" than an element x are in P, then x is in P. Eventually, if B bars X, R, then we conclude that $P = X$.

A tool for proving that B bars X, R is the notion of "simulation". We say that x' is an R-predecessor of x if $x'Rx$. Roughly speaking, $V \subseteq X \times Y$ is a simulation between X, R and Y, S if whenever two elements are related by V, then any R-predecessor of the first element is V-related with some S-predecessor of the second element.

▶ **Definition 8.** *We say that $V \subseteq X \times Y$* simulates *X, R in Y, S if for all $x, x' \in X$, $y \in Y$, if $x'Rx$ and xVy, then there is some $y' \in Y$, $y'Sy$ such that $x'Vy'$.*

We will prove that a simulation V, when V is everywhere defined, moves bars backwards from Y to X. By this we mean: if B bars Y, S, then $V^{-1}(B)$ bars X, R. In particular, simulation moves well-foundedness backwards: if we take $B = \emptyset$, we obtain that if Y is S-well-founded then X is R-well-founded. We will prove the same result for *morphisms*; that is, if $f : X \to Y$ maps pairs related by R into pairs related by S, then f^{-1} maps bars for Y, S into bars for X, R.

▶ **Lemma 9** (Simulation Lemma). *Let X, Y, B, C be sets and R, S be binary relations.*
1. (simulation) *Assume that $V \subseteq X \times Y$ simulates X, R in Y, S, that V is everywhere defined, i.e., for every $x \in X$ there exists $y \in Y$ such that xVy, and that C bars Y, S; then $B = V^{-1}(C)$ bars X.*
2. (morphism) *Assume that $f : X, R \to Y, S$ is a morphism and C bars Y; then $f^{-1}(C)$ bars X.*

Now we prove that, if we extend a bar and we reduce the barred set and the relation, then the fact of being a bar is preserved. Choosing the empty bar, we obtain a well-known result for well-founded relations, namely well-foundedness is preserved by moving to a subrelation. To say otherwise: if X is R-well-founded, with $Y \subseteq X$ and $S \subseteq R$, then Y is S-well-founded.

▶ **Lemma 10** (Monotonicity and Antimonotonicity). *Let X, Y, B, C be sets, and R, S binary relations.*
1. (monotonicity) *If B bars X, R and $B \cap X \subseteq C \cap X$, then C bars X, R.*
2. (antimonotonicity) *If B bars X, R and $Y \subseteq X$, $S \subseteq R$, then B bars Y, S.*

---

infinite $<$-descending chain infinite, with set of elements C. In this model all infinite $<$-descending chain in X intersects $\emptyset$, because no infinite $<$-descending chain exists. Yet, the set $P = X \setminus C$ is $X, <$-hereditary while $P \neq X$. Thus, it is not true that $\emptyset$ bars X, R.

For every family of sets Y_x indexed by $x \in X$ we write $\Sigma_{x\in X} Y_x$ for the set of pairs (x, y) such that $x \in X$ and $y \in Y_x$.

Now let R be a binary relation, and $S = \{S_x\}_{x\in X}$ an indexed family of binary relations on Y. We can think of S as a ternary relation such that $S(x, y', y) \Leftrightarrow S_x(y', y)$ for all $x \in X$ and $y', y \in Y$. The *lexicographic product* $R \times S$ is the relation comparing (x', y') with (x, y) according to xRx', or, if $x = x'$, according to yS_xy'. Formally:

$$(x', y')(R \times S)(x, y) \quad \Leftrightarrow \quad x'Rx \vee (x' = x \wedge y'S_xy).$$

$R \times S$ is a partial order if R and all S_x are partial orders, in this case $R \times S$ is called the lexicographic order on pairs.

Assume that the dependency on $x \in X$ is trivial, that is, for some Z, T and for all $x \in X$ we have $Y_x = Z$, $S_x = T$. In this case we write $R \times T$ for $R \times S$. By definition unfolding, $R \times T$ is a relation on $\Sigma_{x\in X} Y_x = X \times Z$ defined by $(x', y')(R \times T)(x, y) \Leftrightarrow x'Rx \vee (x' = x \wedge y'Ty)$.

With the next lemma we define a bar D for $\Sigma_{x\in X} Y_x$, $R \times S$. When the dependency on $x \in X$ is trivial, D is a bar for $X \times Z$, $R \times T$. Our result generalises [14, Chapter I, Theorem 6.3], which is, in our terminology, the special case when D is the empty bar.

▶ **Lemma 11** (Lexicographic Product). *Let X, Y, B and C_x for $x \in X$ be sets, R a binary relation and S a ternary relation. Suppose that B bars X, R, and that C_x bars Y_x, S_x for all $x \in X$. Let D be the set of all pairs $(x, y) \in \Sigma_{x\in X} Y_x$ such that $x \in B$ or $y \in C_x$.*

1. *D bars $\Sigma_{x\in X} Y_x$ with $R \times S$, the lexicographic product of R, S.*
2. *If for some Z, T and for all $x \in X$ we have $Y_x = Z$, $S_x = T$, then D bars $X \times Z$, $R \times T$.*

4 Higman's Lemma for Bars

In this section, we state Higman's lemma for bars, which is a constructive version of Higman's lemma for subsequences, and we argue why this version is stronger with intuitionistic logic than the versions proposed until now.

Let $(P, \leqslant)$ be a partial order. For a given labelling $\varphi : I \to P$, we recall that we write $\mathtt{Fin}(I)$ for the set of finite lists in I and $\mathtt{Incr}(\varphi, I)$ for the subset $\mathtt{Incr}(\leqslant, \varphi, I)$ of $\mathtt{Fin}(I)$ consisting of the finite lists ℓ in I such that $\varphi\ell$ is a weakly increasing list on P for the order $\leqslant$. We can now introduce the constructive version wqo(bar) of the notion of wqo.[2] A quasi-order $(P, \leqslant)$ is wqo(bar) if for every set $X \subseteq \mathtt{Fin}(I)$, a bar B for the subset of X consisting of all w.i. lists (those in $\mathtt{Incr}(\varphi, I)$) is a bar for the whole of X, provided that X is closed by sublists and B by superlists.

▶ **Definition 12** (Well quasi-order with bars). *A quasi-order $(P, \leqslant)$ is called* wqo(bar) *if*

$$B \text{ bars } X \cap \mathtt{Incr}(\varphi, I) \implies B \text{ bars } X \tag{1}$$

for all labellings $\varphi\colon I \to P$ of I by P, for every subset $X \subseteq \mathtt{Fin}(I)$ closed by I-sublists and for every subset $B \subseteq \mathtt{Fin}(I)$ closed by I-superlists.

As before, "B bars ..." is meant for the converse $>_1$ of the one-step extension order $<_1$ on $\mathtt{Fin}(I)$.

Classically, (1) means that every infinite $<_1$-increasing chain $\sigma\colon \mathbb{N} \to X$ meets B if this is the case already for any such σ for which in addition we have $\varphi\sigma(0) \sqsubseteq \varphi\sigma(1) \sqsubseteq \dots$ Within classical logic, condition (1) is equivalent to the more commonly used notion of wqo(set): for every infinite list $\sigma : \mathbb{N} \to \Sigma^*$ there is a an infinite $\sqsubseteq$-weakly increasing sublist $\tau : \mathbb{N} \to \Sigma^*$.

[2] For a constructive comparison of the customary concepts of wqo we refer to [6].

We focus on partial orders $P = \Sigma^*$, given by the set of words for a finite alphabet Σ, with the subword order $\sqsubseteq$ as $\leqslant$, and we prove that:

▶ **Theorem 13** (Higman's lemma for bars). *If Σ is a finite alphabet, then Σ^* is a wqo(bar).*

We postpone the proof of Theorem 13 to §5. In the rest of this section we derive with intuitionistic logic some corollaries of Theorem 13, in order to show the interest from an constructive viewpoint of stating the result in this form.

Our corollaries are about functionals. We add a bottom element $\perp$ to $\mathbb{N}$, then we consider partial and total continuous functional $F\colon \mathtt{Inf}(\Sigma^*) \to \mathbb{N} \cup \{\perp\}$ on infinite sequences of words. We take the canonical topology on $\mathtt{Inf}(\Sigma^*) \to \mathbb{N} \cup \{\perp\}$.[3] F maps infinite sequences of words in $\mathbb{N} \cup \{\perp\}$. F continuous means that F, when convergent, uses only a finite part of its input. Informally, a partial functional F explores larger and larger finite prefixes of an infinite sequence of words, until F finds a prefix long enough to compute some $n \in \mathbb{N}$. Formally, we define F as a map on *finite* lists, which can return the bottom element $\perp$, and if it returns $n \in \mathbb{N}$ on a finite list l then returns the same n on all extensions of l. If σ is infinite, then $F(\sigma) = n$ if and only if $F(l) = n$ for some finite prefix l of σ. Classically, F is called "total" if F returns some $n \in \mathbb{N}$ on all infinite lists. In order to make possible proofs with intuitionistic logic, we define totality through a bar instead.

▶ **Definition 14.**

1. *The strict order $\prec$ on $\mathbb{N} \cup \{\perp\}$ is defined by $\perp \prec n$ for all $n \in \mathbb{N}$ and no comparison between two natural numbers. $\preceq$ is the associated weak order.*
2. *A* partial continuous functional *is a map $F\colon \mathtt{Fin}(\Sigma^*) \to \mathbb{N} \cup \{\perp\}$ which is monotone with respect to the prefix order $\leqslant$ and $\preceq$.*
3. *A partial continuous functional F is* (bar-)total *if $F^{-1}(\mathbb{N})$ bars $\mathtt{Fin}(\Sigma^*)$.*
4. *If F is a total continuous functional, then its* canonical extension *to all $\sigma \in \mathtt{Inf}(\Sigma)$ is given by $F(\sigma) = n$ if for some finite prefix l of σ, we have $F(l) = n$.*[4]

▶ **Proposition 15.** *If F is bar-total and $\sigma \in \mathtt{Inf}(\Sigma)$, then $F(\sigma)$ exists, it is in $\mathbb{N}$ and it is unique.*

Proof. . From $F^{-1}(\mathbb{N})$ bar of $\mathtt{Fin}(\Sigma^*)$ and Lemma 7.4, every infinite list σ has some finite prefix l in the bar $F^{-1}(\mathbb{N})$, therefore $F(\sigma) = F(l) = n \in \mathbb{N}$ for some $n \in \mathbb{N}$. The value n is unique: if $F(\sigma) = F(l') = n' \in \mathbb{N}$ for another finite prefix of σ, then either $l \leqslant l'$ or $l' \leqslant l$, therefore $F(l) \preceq F(l')$ or $F(l') \preceq F(l)$, that is, $n \preceq n'$ or $n' \preceq n$. In both cases we conclude $n = n'$. ◀

Thus, if F is bar-total, then F is "total" with the usual classical definition: F returns some $n \in \mathbb{N}$ on all infinite lists. Classically, the reverse implication holds, but with intuitionistic logic bar-total is a stronger property.[5] From now on, by "total" we will always mean bar-total.

---

[3] For any $l \in \mathtt{Inf}(\Sigma^*)$, we define $O_l = \{m \in \mathtt{Inf}(\Sigma^*) \mid l \leqslant m\}$; we then take on $\mathbb{N}$ the discrete topology, on $\mathtt{Inf}(\Sigma^*)$ the topology generated by the sets O_l with $l \in \mathtt{Inf}(\Sigma^*)$ and the function topology on $\mathtt{Inf}(\Sigma^*) \to \mathbb{N} \cup \{\perp\}$.

[4] The idea is that we can approximate an element of $\mathtt{Inf}(\Sigma)$ considering all its initial segments which are elements of $\mathtt{Fin}(\Sigma^*)$.

[5] We claim that there is some recursive functional F which is defined on all recursive sequences, but returning $\perp$ on some non-recursive sequence. The proof uses the folk-lore result there is some recursive tree, whose recursive branches are all finite, but having some infinite non-recursive branch.

Let us fix a total functional F and a finite alphabet Σ. Higman's lemma for subsequences implies that for every infinite list σ over Σ^*, there is an infinite sublist $\tau \sqsubseteq \sigma$ whose first $F(\tau)$ elements are in w.i. order. Classically, it is enough to take any infinite w.i. sub-list τ of σ and then a finite prefix l of $F(\tau)$ elements. We call "F-long" the prefix of τ with $F(\tau)$-elements.

Informally speaking, this result means that we can provide infinite sublists τ having a w.i. prefix of any given length, with the length $F(\tau)$ we require described by some bar-total continuous functional F applied to the very sublist τ we are defining. We can provide a proof with intuitionistic logic of this result as an immediate corollary of Higman's lemma for bars.

▶ **Corollary 16** (sublists with an F-long w.i. prefix)**.** *Let Σ be a finite alphabet and $F : \mathtt{Fin}(\Sigma^*) \to \mathbb{N} \cup \{\bot\}$ a bar-total continuous functional. Then every infinite sequence of words $\sigma \in \mathtt{Inf}(\Sigma^*)$ has an infinite subsequence τ with the first $F(\tau)$ elements in w.i. order, i.e. such that τ has an F-long w.i. prefix.*

Proof. Let $\varphi = \mathrm{id}_I$ where $I = \Sigma^*$. Set $X_0 = \mathtt{Incr}(\varphi, I)$ and $X = \mathtt{Fin}(I)$. Let $B_0 = \{\rho \in X \mid F(\rho) \in \mathbb{N}\}$. By the hypotheses on F, this B_0 bars X, and is upwards closed in X for the prefix order $\leqslant$. By the antimonotonicity of bars (Lemma 10.2), B_0 also bars $X_0 \subseteq X$.

Let $B_1 = \{\rho \in B_0 \cap X_0 \mid \mathtt{len}(\rho) \geqslant F(\rho)\}$. *Claim:* B_1 bars X_0. To prove this, set $P = \{\rho \in X_0 \mid B_1 \text{ bars } \rho\}$. Then the Claim means $P = X_0$, which we show by bar induction with the bar B_0 for X_0. Since P is hereditary (Proposition 7), which is the induction step, we only need to verify the base case $B_0 \cap X_0 \subseteq P$. To this end we show $\rho \in P$ for all $\rho \in B_0 \cap X_0$ by induction on $f(\rho) = \max(0, F(\rho) - \mathtt{len}(\rho))$.

Case $f(\rho) = 0$: Then $F(\rho) \leqslant \mathtt{len}(\rho)$ and thus $\rho \in B_1 \subseteq P$.

Case $f(\rho) = n + 1$: For every $\rho' \in X_0$ with $\rho <_1 \rho'$ we have $\mathtt{len}(\rho') = \mathtt{len}(\rho) + 1$, and $F(\rho) = F(\rho')$ by continuity, so $f(\rho') = n$. In addition, $\rho' \in B_0$ (because $\rho \in B_0$ and B_0 is upwards closed for $\leqslant \supseteq <_1$); whence $\rho' \in P$ by induction. As P is hereditary, $\rho \in P$ follows.

This ends the proof of the Claim.

Now let $B = \{\rho \in X \mid \exists \eta \sqsubseteq \rho\, (\eta \in B_1)\}$. Then B is upwards closed for $\sqsubseteq$, i.e. closed by superlists; trivially, X is closed by sublists; and B bars $X_0 = X \cap \mathtt{Incr}(\varphi, I)$. The latter holds by the monotonicity of bars (Lemma 10.1); in fact B_1 bars X_0 by the Claim, and $B_1 \subseteq B$. In all, Higman's lemma for bars (Theorem 13) applies, and yields that B bars X.

Now let $\sigma \in \mathtt{Inf}(I)$. Since B bars X, the infinite list σ has a finite prefix $\sigma_0 \in B$. By definition of B, there is $\tau_0 \sqsubseteq \sigma_0$ such that $\tau_0 \in B_1$, which is to say that $\tau_0 \in X_0 = \mathtt{Incr}(\varphi, I)$, $F(\tau_0) \in \mathbb{N}$ and $\mathtt{len}(\tau_0) \geqslant F(\tau_0)$. We extend τ_0 to an infinite sublist τ of σ. From $F(\tau_0) \in \mathbb{N}$ we get $F(\tau) = F(\tau_0) \leqslant \mathtt{len}(\tau_0)$. Hence the first $F(\tau)$ entries of τ form a prefix of τ_0 and thus are in w.i. order. ◀

▶ **Example 17.** Let $\sigma \in \mathtt{Inf}(\Sigma^*)$ be an infinite sequence of words over a finite alphabet Σ.

1. For all $k \in \mathbb{N}$ there is some w.i. length k subsequence of σ.
2. There are w.i. subsequences τ_1, τ_2, τ_3 of σ which have length $\mathtt{len}(\tau_1(0))+1$, $\mathtt{len}(\tau_2(0))^2+1$ and $2^{\mathtt{len}(\tau_3(0))}$.

Proof. Apply Corollary 16 to the functionals defined by $F_0(\rho) = k$, $F_1(\rho) = \mathtt{len}(\rho(0)) + 1$, $F_2(\rho) = \mathtt{len}(\rho(0))^2 + 1$ and $F_3(\rho) = 2^{\mathtt{len}(\rho(0))}$ where $\rho \in \mathtt{Fin}(\Sigma^*)$, which are bar-total continuous. In fact, $F_0^{-1}(\mathbb{N}) = \mathtt{Fin}(\Sigma^*)$ and $F_\nu^{-1}(\mathbb{N}) = \mathtt{Fin}(\Sigma^*) \setminus \{\mathtt{nil}\}$ for $\nu \in \{1, 2, 3\}$; whence $F_\nu^{-1}(\mathbb{N})$ bars $\mathtt{Fin}(\Sigma^*)$ in all cases. ◀

The particular case $k = 2$ of Example 17 means that there are $i < j$ for which $\sigma(i) \sqsubseteq \sigma(j)$. This is Higman's lemma in its usual form.

5 A Constructive Proof of Higman's Lemma for Bars

In this section we first prove some basic properties of wqo(bar): closure under finite product, finite union and right-invertible morphism. All these properties are classically true for the classically equivalent notion of wqo, see for example the original article by Higman [11]. Subsequently, we prove Higman's lemma for bars by induction on the finite alphabet Σ. We assume that all Δ^* are wqo(bar), for all Δ smaller than Σ, in order to prove that Σ^* is a wqo(bar). The crucial step will be proving that the anticone of every $v \in \Sigma^*$ is a wqo(bar).

5.1 Essential Properties of Wqo (bar)

We start by giving two immediate examples, of a quasi-order which is wqo(bar) and a quasi-order which is no wqo(bar). For every set I, $(I, =)$ is a quasi-order. Assume that Σ is a finite set, we can prove with intuitionistic logic that $(\Sigma, =)$ is a wqo(bar); whereas $(\mathbb{N}, =)$ is not.

▶ **Proposition 18** (wqo(bar)). *Assume that Σ is a finite set. Then*
1. *$(\Sigma, =)$ is wqo(bar).*
2. *$(\mathbb{N}, =)$ is not wqo(bar).*

Proof.
1. We assume that $X \subseteq \mathtt{Fin}(I)$ is closed by I-sublist, that B is closed by I-superlists, and that B bars $X \cap \mathtt{Incr}(=, \varphi, I)$, in order to prove that B bars X. We argue by induction on Σ. Assume that $\Sigma = \emptyset, \{x\}$. Then all labelling (if any) are constantly equal to x, therefore are weakly increasing. We deduce that $X \cap \mathtt{Incr}(=, \varphi, I) = X$ and we conclude that B bars X. Assume that Σ has two or more elements. Then $\Sigma = \Sigma_1 \cup \Sigma_2$ for some $\Sigma_1, \Sigma_2 \subset \Sigma$. Let $I_1 = \varphi^{-1}(\Sigma_1)$ and $I_2 = \varphi^{-1}(\Sigma_2)$. Then $I = I_1 \cup I_2$, and by antimonotonicity B bars $X \cap \mathtt{Incr}(=, \varphi, I_1)$ and B bars $X \cap \mathtt{Incr}(=, \varphi, I_2)$. By $X \subseteq \mathtt{Fin}(I)$ closed by I-sublist, B is closed by I-superlists and Lemma 19 we conclude that B bars X.
2. $\mathbb{N}, =$ is a partial order. In order to prove that it is not a wqo(bar), we will provide some $X \subseteq \mathtt{Fin}(I)$ closed by I-sublist, some B is closed by I-superlists, such that B bars $X \cap \mathtt{Incr}(=, \varphi, I)$ and B does not bars X. We choose $X =$ the set of non-repeating lists of length 1 words. X is closed by I-sublists and all its length $\geqslant 2$ sublists are not increasing, because if $i \neq j$, then $\langle i \rangle \not\sqsubseteq \langle j \rangle$. Then $X \cap \mathtt{Incr}(=, \varphi, I)$ consists of all lists with 1 word of length 1. These lists are not comparable by $>_1$, therefore this set is trivially well-founded by $>_1$, and it is barred by $B = \emptyset$. However, B does not bar X, because the infinite list $\sigma = \langle 0 \rangle, \langle 1 \rangle, \langle 2 \rangle, \ldots$ in $\mathbb{N}$ does not intersect $\emptyset$. ◀

For comparison, if we use the notion of wqo(set), then point 1 above say that all infinite lists on a finite set Σ have an infinite constant sublist, while point 2 says there is an infinite list on $\mathbb{N}$ with no infinite constant sublist. Point 1 requires classical logic (this is why we avoid using the notion of wqo(set)). Point 2 follows by taking the infinite list $0, 1, 2, 3, \ldots$.

In order to derive more basic properties of wqo(bar), we have first to find a constructive counterpart of the following classical property. In classical logic, given an infinite list σ in $\mathtt{List}(I_1 \cup I_2)$, if σ_1 is the sublist obtained by restricting σ to the elements in I_1, and σ_2 is the sublist obtained by restricting σ to the elements in I_2, then either σ_1 is infinite or σ_2 is infinite. We propose to call this property the *Riffling Property for infinite lists*, because if I_1, I_2 are disjoint, then σ can obtained from σ_1, σ_2 as when we riffle two decks of card in order to obtain a single deck of cards, while preserving the order we have in each deck. Riffling is not provable with intuitionistic logic, because we cannot decide whether we have

an infinite sublist in $\texttt{Fin}(I_1)$ or in $\texttt{Fin}(I_2)$. In order to constructivise riffling, we prove a kind of contrapositive: if X is a set of lists and we bar with B the infinite I_1-lists in X and the infinite I_2-lists in X, then we bar with B the infinite $I_1 \cup I_2$-lists in X. When we state Riffling, we move from lists in X to sublists in X, and from sublists in the bar B to lists in the same B. Therefore Riffling requires two new assumptions, that B is closed by I-superlists and that X is closed by I-sublist. These are the same assumptions we have in the definition of wqo(bar).

▶ **Lemma 19** (Riffling for Bars). *Assume that the set X is closed by $I_1 \cup I_2$-sublists and the set B is closed by $I_1 \cup I_2$-superlists, then:*

$$B \text{ bars } X \cap \texttt{Fin}(I_1) \ \wedge \ B \text{ bars } X \cap \texttt{Fin}(I_2) \implies B \text{ bars } X \cap \texttt{Fin}(I_1 \cup I_2)$$

From the Riffling Property for Bars we deduce with intuitionistic logic that wqo(bar) are closed under binary compatible union.

▶ **Lemma 20** (Compatible union of wqo's). *If $(P, \leqslant_P)$ and $(Q, \leqslant_Q)$ are wqo(bar), $(P \cup Q, \leqslant)$ is a quasi-order and $\leqslant_P, \leqslant_Q \subseteq \leqslant$, then $(P \cup Q, \leqslant)$ is a wqo(bar).*

The next step is to prove with intuitionistic logic that wqo(bar) are closed by componentwise product.

▶ **Lemma 21** (Componentwise product of wqo(bar)). *Assume that $(P, \leqslant_P)$ and $(Q, \leqslant_Q)$ are wqo(bar). Then $(P \times Q, \leqslant_P \times \leqslant_Q)$ with the componentwise order is a wqo(bar).*

The last preliminary step is to prove with intuitionistic logic that wqo(bar)'s are closed by right-invertible morphisms. Again, this property is easily proved for the classical definition of wqo. Assume that we have a morphism $f : P \to Q$ with right inverse $g : Q \to P$ (i.e., $fg = \texttt{id}_Q$) and $(P, \leqslant_P)$ is a wqo, then every infinite list $\sigma : \mathbb{N} \to Q$ is mapped by g into an infinite list $g\sigma : \mathbb{N} \to P$, which has an infinite w.i. sublist $\tau : \mathbb{N} \to P$, which is mapped by f into an infinite w.i. list $f\tau : \mathbb{N} \to Q$. From τ sublist of $g\sigma$ we deduce that $f\tau$ is a sublist of $fg\sigma$. From $fg\sigma = \sigma$ we conclude that $f\tau$ is an infinite w.i. sublist of σ. If we use the notion of wqo (bar), we can provide a proof with intuitionistic logic for the same result.

▶ **Lemma 22** (right-invertible morphism on a wqo(bar)). *Assume that $(P, \leqslant_P)$ is a wqo(bar), $(Q, \leqslant_Q)$ is a quasi-order and $f : P \to Q$ is a morphism with right inverse g.*[6] *Then $(Q, \leqslant_Q)$ is a wqo(bar).*

5.2 The Anticone of a Word is a Wqo (bar)

In this subsection, we fix a labelling $\varphi : I \to \Sigma^*$, and we assume that Δ^* is a wqo(bar) for all $\Delta \subset \Sigma$; then we prove that the anticone of every $v \in \Sigma^*$ is a wqo(bar). This is a crucial step in the proof of Higman's lemma for bars.

▶ **Lemma 23** (Slices and Anticones of a Word). *Assume that Σ is a finite alphabet and for all $\Delta \subset \Sigma$ the partial order Δ^* a wqo(bar). Let $v = c_1 \dots c_k \in \Sigma^*$, then:*

1. $\texttt{Slice}_\Sigma(v)$*, the slice of v, is a wqo(bar).*
2. $\not\sqsubseteq_\Sigma(v)$*, the anticone of v, is a wqo(bar).*

[6] Observe that g does not need to be a morphism.

5.3 A Decomposition of Finite Lists of Words over a Finite Language

In this subsection, where $\varphi : I \to P = \Sigma^*$ labels an arbitrary set I with words over a finite alphabet Σ, we introduce the last ingredient needed in the proof of the Higman lemma for bars. We extract from each finite list l with labels $\langle w_0, \dots, w_{p-1}\rangle$ two disjoint sublists:

1. some φ-w.i.sub-list $\mathtt{Lex}(l,\varphi)$ of l, with labels $w_{i_0} \sqsubseteq \dots \sqsubseteq w_{i_{n-1}}$. $\mathtt{Lex}(l,\varphi)$ is the sub-list obtained by selecting each time as next element *the first element making the sub-list φ-w.i.*;
2. the suffix $\mathtt{Suff}(l,\varphi)$ of l, with labels $\langle w_m, \dots, w_{p-1}\rangle$ of l, such that $m = i_{n-1} + 1$, and that $w_{i_{n-1}} \not\sqsubseteq w_m, \dots, w_{p-1}$. If this is not possible, then $\mathtt{Suff}(l,\varphi)$ is the empty list.

In our terminology, the elements of $\mathtt{Suff}(l,\varphi)$ are in the anticone of $w_{i_{n-1}}$, where $w_{i_{n-1}}$ is the last element of $\mathtt{Lex}(l,\varphi)$. We will prove the Higman lemma for bars by bar induction on such pair of lists. The formal definition of the two sub-lists $\mathtt{Lex}, \mathtt{Suff}$ runs as follows. We have to define first two integer lists $\mathtt{lex}, \mathtt{suff}$, with low case $\mathtt{l}, \mathtt{s}$, consisting of the list of indexes of $\mathtt{Lex}, \mathtt{Suff}$ in φl.

▶ **Definition 24** (Decomposition of a list). *Assume l is any finite list on I, labeled by a map $\varphi : I \to P = \Sigma^*$. Suppose $\varphi l = \langle w_0, \dots, w_{p-1}\rangle$ is the list of labels of l. By induction on l, we define $\mathtt{lex}(l,\varphi)$, $\mathtt{suff}(l,\varphi)$.*

1. *We define* $\mathtt{lex}(\mathtt{Nil},\varphi) = \mathtt{suff}(\mathtt{Nil},\varphi) = \mathtt{Nil}$ *and* $\mathtt{lex}(\langle i\rangle,\varphi) = \langle 0\rangle$, $\mathtt{suff}(\langle i\rangle,\varphi) = \mathtt{Nil}$.
2. *Suppose* $\mathtt{len}(l) = p \geqslant 1$, $\mathtt{lex}(l,\varphi) = \langle i_0 \dots, i_{n-1}\rangle$ *(an integer list) and $x \in I$. We define the clause for $l{\star}\langle x\rangle$ by cases on the condition: "$w_{i_{n-1}} \sqsubseteq \varphi(x)$".*
   a. Assume $w_{i_{n-1}} \sqsubseteq \varphi(x)$. *Then we set* $\mathtt{lex}(l{\star}\langle x\rangle,\varphi) = \mathtt{lex}(l,\varphi){\star}\langle p\rangle$ *(we add the index p of x to* lex*) and* $\mathtt{suff}(l{\star}\langle x\rangle,\varphi) = \mathtt{Nil}$ *(we reset* suff *to* nil*).*
   b. Assume $w_{i_{n-1}} \not\sqsubseteq \varphi(x)$. *Then we set* $\mathtt{lex}(l{\star}\langle x\rangle,\varphi) = \mathtt{lex}(l,\varphi)$ *(*lex *stays the same) and* $\mathtt{suff}(l{\star}\langle x\rangle,\varphi) = \mathtt{suff}(l,\varphi){\star}\langle p\rangle$ *(we add the index p of x to* suff*).*

Finally, we define $\mathtt{Lex}, \mathtt{Suff}$*, the maps with capital* $\mathtt{L}, \mathtt{S}$*, by:* $\mathtt{Lex}(l,\varphi) = l\mathtt{lex}(l,\varphi)$ *and* $\mathtt{Suff}(l,\varphi) = l\mathtt{suff}(l,\varphi)$.

A (crucial) example: let $I = \Sigma^*$, $\varphi = \mathtt{id}$ (labelling φl and list l coincide), and $l = \langle w_0, w_1, w_2, w_3, w_4\rangle$, with

$$w_0 = a, \quad w_1 = ab, \quad w_2 = abb, \quad \text{and } w_3 = bb, \ w_4 = bbb$$

According to Def.24 we obtain:

1. for $m = \mathtt{nil}$: $\mathtt{lex}(m,\varphi) = \mathtt{nil}$.
2. for $m = \langle w_0\rangle$: $\mathtt{lex}(m,\varphi) =$ the index 0 of w_0
3. for $m = \langle w_0, w_1\rangle$: $\mathtt{lex}(m,\varphi) =$ the indexes $0, 1$ of w_0, w_1
4. for $m = \langle w_0, w_1, w_2\rangle$: $\mathtt{lex}(m,\varphi) =$ the indexes $0, 1, 2$ of w_0, w_1, w_2

When m increases to $m = \langle w_0, w_1, w_2, w_3\rangle$, the new word w_3 added to m is discarded in $\mathtt{lex}(m,\varphi)$. Indeed, we have $w_2 \not\sqsubseteq w_3, w_4$, therefore if $m = \langle w_0, w_1, w_2, w_3\rangle$ then $\mathtt{lex}(m,\varphi)$ is again equal to the indexes $0, 1, 2$ of w_0, w_1, w_2. The same when $m = \langle w_0, w_1, w_2, w_3, w_4\rangle$: the new word w_4 added to m is again discarded, and we still have $\mathtt{lex}(m,\varphi) =$ the integer list $0, 1, 2$.

The indexes of the discarded words are piled up in suff. The first three values of $\mathtt{suff}(m,\varphi)$ are: nil, nil, nil. From $w_2 \not\sqsubseteq w_3, w_4$, we deduce the following values for suff: $\mathtt{suff}(m,\varphi) =$ the integer list whose only element is 3, and $\mathtt{suff}(m,\varphi) =$ the integer list $3, 4$.

The outputs of $\texttt{Lex}(m,\varphi)$ and $\texttt{Suff}(m,\varphi)$ (with capital $\texttt{L},\texttt{S}$) are the same, except that we take words instead of indexes of words. For the same values of m we obtain for $\texttt{Lex}(m,\varphi)$: $\texttt{nil}$, $\langle w_0\rangle$, $\langle w_0,w_1\rangle$, $\langle w_0,w_1,w_2\rangle$, then again $\langle w_0,w_1,w_2\rangle$ and again $\langle w_0,w_1,w_2\rangle$.

The words w_3,w_4 discarded from $\texttt{Lex}(m,\varphi)$ are piled up in $\texttt{Suff}(m,\varphi)$. Indeed, according to Def.24 we obtain for $\texttt{Suff}(m,\varphi)$: $\texttt{nil}$, $\texttt{nil}$, $\texttt{nil}$, then $\langle w_3\rangle$ and $\langle w_3,w_4\rangle$.

A last example. Suppose we add $w_5 = abb$ to m. In this case $w_2 \sqsubseteq w_5$, then w_5 is added to $\texttt{Lex}(l,\varphi)$ and we obtain $\texttt{Lex}(l{\star}\langle w_5\rangle,\varphi) = w_0w_1w_2w_5$. Instead, $\texttt{Suff}$ is reset to $\texttt{nil}$: according to Def. 24, we obtain $\texttt{Suff}(l{\star}\langle w_5\rangle,\varphi) = \texttt{nil}$.

The name we choose for the map $\texttt{lex}$ comes from the fact that $f = \texttt{lex}(l,\varphi)$ is the minimum in the lexicographic ordering of all integer lists such that lf is a φ-w.i. sublist of l. In this paper, however, we do not need a proof of this feature of f and we do not include further details.

The following properties of $f = \texttt{lex}(l,\varphi)$ and $g = \texttt{suff}(l,\varphi)$ are immediate from the definition. First, that $lf \in \texttt{Incr}(\varphi,I)$ for all $l \in \texttt{Fin}(I)$. Second, if $l > \texttt{Nil}$, $g = \texttt{suff}(l,\varphi)$, then lg is equal to the suffix of l after the last element of lf, and that lg is in the anticone of the last element of lf. Both properties can be proved by induction on l.

5.4 Proof of the Main Theorem

▶ **Theorem 25** (Higman's lemma for bars). *If Σ is a finite alphabet, then Σ^* is a wqo(bar).*

Proof. We argue by principal induction on Σ. If $\Sigma = \emptyset$ then $\Sigma^* = \texttt{Nil}$ and Σ^* is a wqo by Lemma 18. Assume Σ has some element. We assume that for all $\Delta \subset \Sigma$ the partial order Δ^* is a wqo(bar), in order to prove that Σ^* is a wqo(bar). We assume that I is a set, $\varphi : I \to \Sigma^*$ any labelling of elements of I by words, $X \subseteq \texttt{Fin}(I)$ is a set of finite I-lists closed by I-sublists, B is a set of finite I-lists closed by I-superlists, and B bars $X \cap \texttt{Incr}(\varphi,I)$; our goal is to prove that B bars X.

Let $\texttt{Lex},\texttt{Suff}$ as in Def. 24, and $\sigma,l \in X$. We define a map $f(\sigma) = \texttt{Lex}(\sigma) \times \texttt{Suff}(\sigma)$ proving that $f : X \to Y$ is a morphism, where $Y := \Sigma_{l\in X\cap\texttt{Incr}(\varphi,I)}Y_l$, for a family of sets $\{Y_l \mid l \in X \cap \texttt{Incr}(\varphi,I)\}$ we are going to define. We will prove that Y is barred by some D such that $f^{-1}(D) \subseteq B$; "B bars X" follows from the Simulation Lemma (9) and monotonicity.

If $l = \texttt{Nil}$, we set $Y_{\texttt{Nil}} = \{\texttt{Nil}\}$. If $l \neq \texttt{Nil}$, we define each set Y_l as the set of all I-lists in X φ-labeled by words which are *not* super-words of (which are in the anticone of) the last word of φl. We formally define Y_l as follows. Let v be *the last element of* φl: then we set $Y_l := \texttt{Fin}(\varphi^{-1}(\not\sqsubseteq_\Sigma(v))) \cap X$.

By definition of $\texttt{Lex}$, $\texttt{Suff}$ and the closure of X by I-sublists, we have $\texttt{Lex}(\sigma) \in X \cap \texttt{Incr}(\varphi,I)$ and $\texttt{Suff}(\sigma) \in Y_l$. Moreover, by definition of $\texttt{Lex}$ and $\texttt{Suff}$, whenever we add one element i to l, either we add the same i to $\texttt{Lex}(\sigma,\varphi)$, or $\texttt{Lex}(\sigma,\varphi)$ stays the same and we add i to $\texttt{Suff}(\sigma,\varphi)$. Thus, f is a morphism from $(X,>_1)$ to $\Sigma_{l\in X\cap\texttt{Incr}(\varphi,I)}Y_l$ with relation the lexicographic product $>_1 \times >_1$. By Lemma 23.2 (Slices and Anticones), $\not\sqsubseteq_\Sigma(v)$ is a wqo(bar). B bars $X \cap \texttt{Incr}(\varphi,I)$ by assumption. Then B bars the subset $\texttt{Fin}(\varphi^{-1}(\not\sqsubseteq_\Sigma(v))) \cap X \cap \texttt{Incr}(\varphi,I)$ by antimonotonicity. $\not\sqsubseteq_\Sigma(v)$ is a wqo(bar), therefore B bars $\texttt{Fin}(\varphi^{-1}(\not\sqsubseteq_\Sigma(v))) \cap X$, which is Y_l. Let D be the set of pairs (l,m) such that $l \in B$ or $m \in B$. By Lemma 11 (Lexicographic Product), D bars $Y = \Sigma_{l\in X\cap\texttt{Incr}(\varphi,I)}Y_l$, $>_1 \times >_1$. By Simulation Lemma (9) we deduce that $f^{-1}(D)$ bars X. In order to prove that B bars X, by monotonicity it is enough to prove that $f^{-1}(D) \subseteq B$.

Assume that $\sigma \in f^{-1}(D)$, then $f(\sigma) = \texttt{Lex}(\sigma) \times \texttt{Suff}(\sigma) \in D$, and by definition of D, we deduce that $\texttt{Lex}(\sigma) \in B$ or $\texttt{Suff}(\sigma) \in B$. From $\texttt{Lex}(\sigma),\texttt{Suff}(\sigma) \sqsubseteq \sigma$ and closure of B by I-superlists, we conclude that $\sigma \in B$, as wished. ◀

Conclusion

Higman's lemma for sequences says that over a finite alphabet every infinite sequence of words has an infinite weakly increasing subsequence, and is inherently nonconstructive. As a constructive alternative we now have put forward what we call Higman's lemma for bars: over a finite alphabet, every bar for the weakly increasing finite lists of words which is closed by super-lists is already a bar for *all* finite lists. In particular, for every total continuous functional, every infinite sequence of such words has a weakly increasing finite sublist of length bounded below by the functional. We also proved the common form of Higman's lemma: the words over a finite alphabet form a well quasi-order, for our notion of well quasi-order. As we work as much as possible in settings more abstract than the one of words over a finite alphabet, we prepare for a constructive theory of well (quasi-)order, and, more in general, for a constructive version of classical theories dealing with Π^1_2-statements.

References

1 P. Aczel. *An introduction to inductive definitions*, volume 90 of *Stud. Logic Found. Math.*, pages 739–782. North-Holland, 1977.

2 S. Berardi and S. Steila. Ramsey's Theorem for Pairs and k Colors as a sub-Classical Principle of Arithmetic. *J. Symbolic Logic*, 82(2):737–753, 2017.

3 J. Berger. Dickson's Lemma and Higman's Lemma are Equivalent. *South American Journal of Logic*, 2(1):35–39, 2016.

4 S. Berghofer. A Constructive Proof of Higman's Lemma in Isabelle. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 3085 of *Lecture Notes in Computer Science*, pages 66–82, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

5 Yves Bertot and Ekaterina Komendantskaya. Inductive and coinductive components of corecursive functions in coq. *Electronic Notes in Theoretical Computer Science*, 203(5):25–47, 2008. Proceedings of the Ninth Workshop on Coalgebraic Methods in Computer Science (CMCS 2008). doi:10.1016/j.entcs.2008.05.018.

6 G. Buriola, P. Schuster, and I. Blechschmidt. A Constructive Picture of Noetherian Conditions and Well Quasi-orders. In Gianluca Della Vedova, Besik Dundua, Steffen Lempp, and Florin Manea, editors, *Unity of Logic and Computation*, pages 50–62, Cham, 2023. Springer Nature Switzerland.

7 T. Coquand and D. Fridlender. A proof of Higman's lemma by structural induction. Unpublished Manuscript. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.47.486, November 1993.

8 Thierry Coquand and Henrik Persson. Gröbner bases in type theory. In T. Altenkirch, W. Naraschewski, and B. Reus, editors, *Types for Proofs and Programs (Irsee, 1998)*, volume 1657 of *Lecture Notes in Comput. Sci.*, pages 33–46. Springer, Berlin, 1999.

9 D. Fridlender. *Higman's Lemma in Type Theory*. PhD thesis, Chalmers University of Technology, Göteborg, 1997.

10 J. Goubault-Larrecq. A Constructive Proof of the Topological Kruskal Theorem. In Krishnendu Chatterjee and Jirí Sgall, editors, *Mathematical Foundations of Computer Science 2013*, pages 22–41, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

11 G. Higman. Ordering by Divisibility in Abstract Algebras. *Proceedings of the London Mathematical Society*, s3-2(1):326–336, 1952.

12 J.B. Kruskal. Well-quasi-ordering, the tree theorem, and Vazsonyi's conjecture. *Trans. Amer. Math. Soc.*, 95:210–225, 1960.

13 J.B. Kruskal. The theory of well-quasi-ordering: A frequently discovered concept. *J. Comb. Theory A*, 13:297–305, 1972.

14 Ray Mines, Fred Richman, and Wim Ruitenburg. *A Course in Constructive Algebra*. Springer, New York, 1988. Universitext.

15 C.R. Murthy and J.R. Russell. A constructive proof of Higman's lemma. *5th Annual Symposium on Logic in Computer Science, Philadelphia PA*, pages 257–267, 1992.

16 C.St.J.A. Nash-William. On well-quasi-ordering finite trees. *Proc. Cambridge Phil. Soc.*, 59:833–835, 1963.

17 Hervé Perdry. Strongly Noetherian rings and constructive ideal theory. *J. Symb. Comput.*, 37(4):511–535, 2004.

18 Hervé Perdry and Peter Schuster. Noetherian orders. *Math. Structures Comput. Sci.*, 21:111–124, 2011.

19 T. Powell. Applying Gödel's Dialectica interpretation to obtain a constructive proof of Higman's lemma. *Proceedings of Classical Logic and Computation (CLAC'12)*, volume 97 of *EPTCS*:49–62, 2012.

20 T. Powell. Well Quasi-orders and the Functional Interpretation. In Peter M. Schuster, Monika Seisenberger, and Andreas Weiermann, editors, *Well-Quasi Orders in Computation, Logic, Language and Reasoning: A Unifying Concept of Proof Theory, Automata Theory, Formal Languages and Descriptive Set Theory*, pages 221–269, Cham, 2020. Springer International Publishing. doi:10.1007/978-3-030-30229-0_9.

21 Michael Rathjen. Generalized inductive definitions in constructive set theory. In Laura Crosilla and Peter Schuster, editors, *From Sets and Types to Topology and Analysis: Towards Practicable Foundations for Constructive Mathematics*, volume 48 of *Oxford Logic Guides*, chapter 16. Clarendon Press, Oxford, 2005.

22 F. Richman and G. Stolzenberg. Well quasi-ordered sets. *Advances in Mathematics*, 97:145–153, 1993.

23 H. Schwichtenberg, M. Seisenberger, and F. Wiesnet. Higman's Lemma and Its Computational Content. In R. Kahle, T. Strahm, and T. Studer, editors, *Advances in Proof Theory*, pages 353–375, Cham, 2016. Springer International Publishing. doi:10.1007/978-3-319-29198-7_11.

24 M. Seisenberger. An Inductive Version of Nash-Williams' Minimal-Bad-Sequence Argument for Higman's Lemma. *Types for Proofs and Programs*, *LNCS* Vol. 2277, 2001.

25 Monika Seisenberger. Kruskal's tree theorem in a constructive theory of inductive definitions. In P. Schuster, U. Berger, and H. Osswald, editors, *Reuniting the antipodes—constructive and nonstandard views of the continuum (Venice, 1999)*, volume 306 of *Synthese Library*, pages 241–255. Kluwer, Dordrecht, 2001.

26 C. Sternagel. A Mechanized Proof of Higman's Lemma by Open Induction. In Peter M. Schuster, Monika Seisenberger, and Andreas Weiermann, editors, *Well-Quasi Orders in Computation, Logic, Language and Reasoning: A Unifying Concept of Proof Theory, Automata Theory, Formal Languages and Descriptive Set Theory*, pages 339–350, Cham, 2020. Springer International Publishing. doi:10.1007/978-3-030-30229-0_12.

27 A.S. Troelstra. Metamathematica Investigation of Intuitionistic Arithmetic and Analysis. *ILLC pre-publication series*, pages X–93–05, 1993.

28 W. Veldman. An Intuitionistic Proof of Kruskal's Theorem. *Archive for Mathematical Logic*, 43(2):215–264, 2004.

Quantifiying the Robustness of Dynamical Systems. Relating Time and Space to Length and Precision

Manon Blanc
Institut Polytechnique de Paris, Ecole Polytechnique, LIX, Palaiseau, France
Université Paris-Saclay, LISN, Orsay, France

Olivier Bournez
Institut Polytechnique de Paris, Ecole Polytechnique, LIX, Palaiseau, France

Abstract

Reasoning about dynamical systems evolving over the reals is well-known to lead to undecidability. In particular, it is known that there cannot be reachability decision procedures for first-order theories over the reals extended with even very basic functions, or for logical theories that reason about real-valued functions, or decision procedures for state reachability. This mostly comes from the fact that reachability for dynamical systems over the reals is fundamentally undecidable, as Turing machines can be embedded into (even very simple) dynamical systems.

However, various results in the literature have shown that decision procedures exist when restricting to robust systems, with a suitably-chosen notion of robustness. In particular, it has been established in the field of verification that if the state reachability is not sensitive to infinitesimal perturbations, then decision procedures for state reachability exist. In the context of logical theories over the reals, it has been established that decision procedures exist if we focus on properties not sensitive to arbitrarily small perturbations. For example by considering properties that are either true or δ-far from being true, for some $\delta > 0$.

In this article, we first propose a unified theory explaining in a uniform framework these statements, that were established in different contexts.

More fundamentally, while all these statements are only about computability issues, we also consider complexity theory aspects. We prove that robustness to some precision is inherently related to the complexity of the decision procedure. When a system is robust, it makes sense to quantify at which level of perturbation it is. We prove that assuming robustness to a polynomial perturbation on precision leads to a characterisation of PSPACE. We prove that assuming robustness to polynomial perturbation on time or length leads to similar statements for PTIME.

In other words, precision on computations is inherently related to space complexity, while length or time of trajectories, is intrinsically related to time complexity. These statements can also be interpreted in relation to several recent results about the computational power of analogue models of computation.

2012 ACM Subject Classification Theory of computation → Models of computation; Theory of computation → Computability; Theory of computation → Complexity classes; Theory of computation → Complexity theory and logic; Computer systems organization → Analog computers

Keywords and phrases Computability, Complexity theory, Computable analysis, Verification, Decision, Robustness, Dynamical Systems, Models of computation, Analogue Computations

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.17

Funding This work was partially supported by Labex Digicosme and ANR δIFFERENCE.

1 Introduction

The relations between dynamical systems over the reals and computations have been the source of many works, with sometimes very different motivations [23, 22, 2, 4, 11]. Let us focus on some of them.

The context of the verification of continuous and hybrid systems. As many systems in our world are naturally modelled by dynamical systems over the reals (or by so-called hybrid systems, mixing continuous and discrete aspects), verification of safety properties on these systems is inherently related to state reachability for dynamical systems. Roughly speaking, a system is safe if the subset of "*bad states*" (i.e. those not satisfying some property) cannot be reached from the subset of the initial states of the system. Unfortunately, it is well-known that such questions are undecidable, even for very simple dynamical systems. For example, even for piecewise affine functions [22, 23] over the compact domain $[0, 1]^2$.

Such a statement is proved by showing that a Turing machine, which is a particular discrete-time dynamical system, evolving with time over configurations, can be embedded into such systems. This requires mapping the infinite set of possible configurations of a Turing machine into a real domain. Hence, it requires infinite precision encoding when the system is defined over a compact domain. This establishes that verification is undecidable for systems with infinite precisions. However this does not seem to prove that undecidability holds for systems that would not be based on infinite precision computations.

In that spirit, while several undecidability results were stated for hybrid systems, such as Linear Hybrid Automata [18], Piecewise Constant Derivative systems [2], an informal conjecture (that we will call the *"robustness conjecture"*) appeared in the field of verification of hybrid and continuous systems by various authors. It states that *undecidability is due to non-stability, non-robustness and sensitivity to the initial values of the systems*. There were several attempts to formalise and prove this, including [14, 1].

▶ Remark 1. Actually, the *"robustness conjecture"* is known to be an informal conjecture, as it is related to the considered mathematical notion of robustness. It holds for some mathematical concepts of robustness, such as the ones considered in this article. It is provably false for some other mathematical concepts of robustness: see e.g. [19].

▶ Remark 2. The mathematical formalisation of robustness, or the question of the various "natural" concepts of robustness, is related to philosophical questions about the limits of mathematical models, or of computability theory. A Turing machine is an ideal model, as well as dynamical systems over the reals are often idealisations of models. We do not aim at going to these kinds of discussions[1].

▶ Remark 3. Our point in this article is to understand how various notions of robustness lead to decidability, namely the ones of [1, 15, 28]. We argue that the *"robustness conjecture"* can be unified in a general theory, in these various approaches. Furthermore, we prove that this holds at the decidability level and also at the complexity level: quantifying the accepted level of robustness corresponds naturally to the complexity of the associated questions.

Here, we are starting from the approach of [1], where classes of dynamical systems are considered. A notion of perturbed dynamics by a small ϵ is associated with each of them. A perturbed reachability relation is defined as the intersection of all reachability relations obtained by ϵ-perturbations. The authors of [1] showed that, for many models, the perturbed reachability relation is co-computably enumerable (co-c.e., Π_1) and any co-c.e. relation can be defined as the perturbed reachability relation of such models. Consequently, it follows from basic computability arguments, namely that a computably enumerable and co-computably enumerable set is decidable that *if robustness is defined as the stability of the reachability relation under infinitesimal perturbation*, then robust systems have a decidable reachability relation and hence a decidable verification (i.e. the *robusness conjecture* holds).

[1] Even if our results shed some light on these questions, such as the fact that complexity theory is essentially quantifying the accepted level of robustness.

The context of the decision procedures for logical theories over the reals. In the context of decision procedures for logic over the reals, the authors of [15] observed that it is well-known that some logics, such as real arithmetic, are decidable. However, decidability does not hold for simple extensions of real arithmetic. Indeed, even the set of Σ_1-sentences in a language extending real arithmetic with the sine function is already undecidable. But if a relaxed and more "robust" notion of correctness is considered (one asks to answer true when a given formula ϕ is true and to return false when it is δ-robustly wrong) the truth of a formula becomes algorithmically solvable. In other words, undecidability intrinsically comes from the fact that the truth of a sentence might depend on infinitesimally small variations of its interpretation.

Recently, the author of [28] proposed a first-order predicate language for reasoning about multi-dimensional smooth real-valued functions. They proved the specification of an algorithm solving formulas robustly satisfiable with respect to some metrics. The proof of decidability can also be interpreted using an argument similar to [1, 15].

Our contributions

We extend and relate these approaches to very general settings and provide a general framework for explaining these observations.

Namely, using various arguments from computability and computable analysis, we establish the following:

- We consider various classes of dynamical systems: in turn, Turing machines, then discrete-time dynamical systems preserving the rationals, then general discrete-time dynamical systems and eventually continuous-time dynamical systems. For all of them, we define robustness as non-sensitivity to infinitesimal perturbations of the associated reachability relation, in the spirit of [1].
  - We prove that for this natural concept of robustness, the *"robustness conjecture"* holds: verification or reachability relation (and hence safety verification) is decidable (Corollary 14, Corollary 25, Corollary 42, Corollary 49, Corollary 54) .
  - We characterise and relate robustness to the question of decidability by proving that the converse holds if some property is added (Corollary 32). This means that there is a form of completeness of the above statement: when decidability holds, establishing the robustness of the corresponding system can be done, for a suitable perturbation or metric.
- Furthermore, we relate this approach, inspired by [1], in the context of verification, to the concept of δ-decision of [15], in the context of decision procedures in logic[2]. A system is robust iff its reachability relation is either true or ϵ-far from being true (Proposition 27).
- We also prove that robustness can be seen as having a reachability relation that can be represented as a pixelated image. It is a simple and elegant geometric property (Corollary 51 and 52).
- More fundamentally, while the above results are about decidability, we also discuss *complexity* issues. Indeed, when a system is robust, it is natural to quantify the allowed level of perturbation.

[2] We mix the notation δ and ϵ when talking about precision. They are indeed the same. Our problem is that the framework considered in [15] uses the terminology δ-decidability, whereas [1] is in a context of real-analysis and uses ϵ to quantify error bounds. We decided to keep both δ and ϵ. Otherwise, this would conflict with their usual meaning in the two contexts.

- We show that considering a perturbation polynomially small relates to a very intuitive way to the complexity of the associated verification or decision problem (Theorem 18, Theorem 36).
- More precisely, polynomial space computability is related to precision (Theorem 18, Theorem 36) while polynomial time computability is related to the time or length of the trajectory (Theorem 64).

More on related work. The approach of considering dynamical systems with respect to infinitesimal perturbations of dynamics is an old idea, sometimes with concepts reinvented later with other names. We can mention the concept of "chain reachability" studied by Conley in the 1970's. In the field of verification, the idea of infinitely perturbed dynamics has been considered to provide alternative semantics of some models: see e.g. [27] for timed automata. The approaches considered in [1] and [15] belong to the line of investigation considering general dynamical systems and aiming at studying the frontier between decidability and undecidability.

Up to our knowledge, such a unifying framework has never been established. For the computability aspects, with respect to some of the existing works: Compared to [1], we allow more general discrete-time and continuous-time dynamical systems, such as those with unbounded domains. Some generalisations have also been obtained in [9], but focusing on dynamical systems as language recognisers and mainly focusing on generalisations of [1, Theorem 4]. The logic considered in [15] allows to talk about finite-time reachability properties, but not reachable sets. As far as we know, complexity aspects have never been discussed.

Motivation and interpretation related to models of computation. An orthogonal field of research is about understanding how analogue (possibly continuous-time) models of computation behave compared to more classical discrete models such as Turing machines. This includes models based on ordinary differential equations like the GPAC [29], or algebraic models based on ordinary differential equations inspired from computability theory [10], or from computer algebra [6]. A long-standing open problem was how to measure time complexity in continuous time models. It was recently proved [8] that the length of the solution curves provides a measure equivalent to time for digital models. The question of a natural measure for space complexity remains open, despite some very recent characterisations of $(F)PSPACE$ using ODEs [7].

▶ Remark 4. The theory developed in the current article comes from an attempt to get to a simpler characterisation of $(F)PSPACE$, with continuous ODEs. We obtained this theory initially with the idea that getting to $(F)PSPACE$ requires a way to forbid undecidability. This led us to develop this theory based on these notions of robustness, guaranteeing computability.

The theory developed here provides arguments to state that, over a compact domain, space corresponds to the precision of the computations, while it corresponds to the logarithm of the size of some graphs for systems over more general domains. Meanwhile, this idea has been used to provide a simple characterisation of $FPSPACE$ with discrete ordinary differential equations in [3]. The question of whether a simple characterisation with continuous ODEs can be obtained remains open.

Preliminaries. $d(\cdot,\cdot)$ is norm-sup (also called uniform) distance. An (open) (resp. close) *rational ball* is a subset of real numbers of the form $B(\mathbf{x},\delta) = \{\mathbf{y} \in \mathbb{R}^d : d(\mathbf{x},\mathbf{y}) < \delta\}$ (resp. $\overline{B}(\mathbf{x},\delta) = \{\mathbf{y} \in \mathbb{R}^d : d(\mathbf{x},\mathbf{y}) \leq \delta\}$) for some rational $\mathbf{x}$ and δ, and some integer d. We could

use the Euclidean distance, but this distance has the advantage that its balls correspond directly to rounding at a precision. A set of reals of the form $\prod_{i=1}^{d}[a_i, b_i]$, for rational (a_i), (b_i), will be called a *rational closed box*. An open rational box is obtained by considering open intervals in the previous definition. The least closed set containing X is denoted by $cls(X)$. We write $\ell(\cdot)$ for the function that measures the binary size of its argument. We say that a function $\mathbf{f} : \mathbb{Q}^d \to \mathbb{Q}^d$ or $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^d$ is Lipschitz when there exists some constant K such that $d(\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{y})) \leq Kd(\mathbf{x}, \mathbf{y})$. We basically have in mind in all this article, dynamical systems over $\mathbb{R}^d$, even if in some of the subsections we consider that they might preserve rationals.

2 On graphs reachability and perturbated TMs

Our theory relies on some well-known observations from complexity theory. We start by recalling some facts and a few basic concepts.

2.1 Some considerations from complexity theory

First, we recall some complexity results about the following decision problem PATH(G, u, v): Given a directed graph $G = (V, \to)$ and some vertices $u, v \in V$, determine whether there is some path between u and v in G, denoted by $u \xrightarrow{*} v$.

▶ **Lemma 5** (Reachability for graphs, [30]). PATH$(G, u, v) \in$ NLOGSPACE.

▶ **Lemma 6** (Immerman–Szelepcsényi's theorem [20, 31]). NLOGSPACE = coNLOGSPACE.

We mainly focus on its complement:

▶ **Corollary 7.** *Consider the following decision problem* NOPATH(G, u, v)*: given a directed graph* $G = (V, \to)$ *and some vertices* $u, v \in V$*, determine whether there is no path between* u *and* v *in* G*.*

Then NOPATH$(G, u, v) \in$ NLOGSPACE*.*

▶ **Theorem 8** (Savitch's theorem, [30]). *For any function* $f : \mathbb{N} \to \mathbb{N}$ *with* $f(n) \geq \log n$*, we have* NSPACE$(f(n)) \subseteq$ SPACE$(f^2(n))$.

▶ **Corollary 9.** PATH$(G, u, v) \in$ SPACE$(log^2(n))$ *and* NOPATH$(G, u, v) \in$ SPACE$(log^2(n))$.

▶ Remark 10. Notice that detecting whether there is no path between u and v is equivalent to determining whether all paths starting from u "loop", i.e. remain disjoint from v. The above statement is established using a more subtle method than a simple depth-of-width search of the graph. One uses the trick of the proof of Savitch's theorem, i.e. a recursive procedure (expressing reachability in less than 2^t steps, called CANYIELD$(\mathrm{C}_1, \mathrm{C}_2, \mathrm{t})$ in [30]) guaranteeing the wanted space complexity.

A (general) *discrete-time dynamical system* $\mathcal{P}$ is given by a set X, called *domain* and some (possibly partial) function f from X to X. A trajectory of $\mathcal{P}$ is a sequence (x_t) evolving according to f: that is $\forall t, x_{t+1} = f(x_t)$. We say x^* (or a set X^*) is reachable from x if there is a trajectory with $x_0 = x$ and $x_t = x^*$ (respectively $x_t \in X^*$) for some t.

▶ Remark 11. In other words, any discrete-time dynamical system $\mathcal{P}$ can be seen as a particular (deterministic) directed graph $G = (V, \to)$, where V is not necessarily finite: G corresponds to $V = X$ and $\to$ to the graph of the function f.

In particular, the following is a classical result (not following from the most obvious algorithm, but from Savitch theorem, i.e. from sometimes so-called arithmetisation techniques).

▶ **Lemma 12** (Reachability for finite graphs). *Let $s(n) \geq \log(n)$. Assume the vertices of $G = (V, \to)$ can be encoded in binary using words of length $s(n)$. Assume the relation $\to$ is decidable using a space polynomial in $s(n)$. Then, given the encoding of $u \in V$ and of $v \in V$, we can decide whether there is some path from u to v, in a space polynomial in $s(n)$.*

Our theory covers various dynamical systems. In particular, as a Turing machine is a particular type of discrete-time dynamical system, we think this helps, for pedagogical reasons, to discuss first the case of Turing machines. We follow, on this aspect, what was done in [1].

2.2 The case of Turing machines

We focus on the framework of Turing Machines (TMs). Let Σ be a finite alphabet and let $B \notin \Sigma$ be the blank symbol. A TM over Σ is a tuple (Q, q_0, F, R, Γ) where Q is a finite set of control states, $q_0 \in Q$ is the initial control state, $F \subseteq Q$ (respectively $R \subseteq Q$) is a set of accepting (respectively rejecting) states, with $F \cap R = \emptyset$ and Γ is a set of transitions of the form $(q, a) \to (q', b, \delta)$ where $q, q' \in Q$, $a, b \in \Sigma \cup \{B\}$ and $\delta \in \{-1, 0, 1\}$. When the machine has accepted or rejected, the decision remains unchanged: when $q \in F$, then $q' \in F$ and when $q \in R$ then $q' \in R$.

We write $\mathcal{C}_{\mathcal{M}}$ for the set of the configurations of a TM and write a configuration as a triple $(q, \cdots a_{-2}a_{-1}, a_0a_1a_2\cdots)$: q gives the internal state and a_0 the position of the head.

Given a transition $(q, a) \to (q', b, \delta)$ in Γ, if the control state is q and the symbol pointed by the head of the machine is equal to a, then the machine can change its configuration C to the configuration C' in the following manner: the control state is now q', the symbol pointed by the head is replaced by b and then the head is moved to the left or the right, or it stays at the same position according to whether δ is $-1, 1$, or 0, respectively. We write $C \vdash C'$ when this holds, i.e. C' is the one-step next configuration of the configuration C. Then $(\mathcal{C}_{\mathcal{M}}, \vdash)$ corresponds to a particular dynamical system.

Word $w = a_1 \cdots a_n \in \Sigma^*$ is accepted by $\mathcal{M}$ if, starting from the initial configuration $C_0 = C_0[w] = (q_0, \cdots BBB, a_1a_2 \cdots a_nBBB \cdots)$ the machine eventually stops in an accepting control state: that is, if we write $\mathcal{F}$ for the configurations where $q \in F$, iff $C_0 \vdash^* C^*$ for some $C^* \in \mathcal{F}$. Let $L(\mathcal{M})$ denote the set of such words, i.e., the computably enumerable (c.e) language semi-recognised by $\mathcal{M}$. We say that w is rejected by $\mathcal{M}$ if, starting from the configuration C_0 the machine $\mathcal{M}$ eventually stops in a rejecting state. $\mathcal{M}$ is said to always halt if for all w, either w is accepted or w is rejected.

Article [1] introduces the concept of space-perturbed TM: given $n > 0$, the idea is that the n-perturbed version of the machine $\mathcal{M}$ is unable to remain correct at a distance more than n from the head of the machine. Formally, the n-perturbed version $\mathcal{M}_n$ of $\mathcal{M}$ is defined exactly as $\mathcal{M}$ except before any transition, all the symbols at a distance n or more from the head can be altered at every step. Hence $\mathcal{M}_n$ is nondeterministic. A word w is accepted by $\mathcal{M}_n$ iff there exists a run of this machine which stops in an accepting state. Let $L_n(\mathcal{M})$ be the n-perturbed language of $\mathcal{M}$. From definitions, if a word is accepted by $\mathcal{M}$, then it is also recognised by all the $\mathcal{M}_n$'s: perturbed machines have more behaviours. Moreover, $L_{n+1}(\mathcal{M}) \subseteq L_n(\mathcal{M})$. Let $L_\omega(\mathcal{M}) = \bigcap_n L_n(\mathcal{M})$: this is the set of words accepted by $\mathcal{M}$ when subject to arbitrarily "small" perturbations. We have $L(\mathcal{M}) \subseteq L_\omega(\mathcal{M}) \subseteq \cdots \subseteq L_2(\mathcal{M}) \subseteq L_1(\mathcal{M})$.

Here is a key observation: The ω-perturbed language of a TM is co-computably enumerable:

▶ **Theorem 13** (Perturbed reachability is co-c.e. [1]). *$L_\omega(\mathcal{M}) \in \Pi_1^0$.*

Since a set that is c.e. and co-c.e. is decidable, following [1], if we define robustness as $L_\omega(\mathcal{M}) = L(\mathcal{M})$, then robust languages are necessarily decidable (i.e. the *"robustness conjecture"* holds).

▶ **Corollary 14** (Robust $\approx$ decidable [1]). *If $L_\omega(\mathcal{M}) = L(\mathcal{M})$ then $L(\mathcal{M})$ is decidable. If $\mathcal{M}$ always halts, then $L(\mathcal{M})$ is decidable and $L_\omega(\mathcal{M}) = L(\mathcal{M})$.*

A simple point, but a key observation for coming discussions, is the following: one can talk about complexity and not only computability. Indeed, when a language is robust, it makes sense to measure what level of perturbation s can be tolerated. This is the purpose of Definition 16.

▶ Remark 15. Assume $L = L(M)$ is robust. By definition, $L = L(M) = L_\omega(\mathcal{M})$. This means that for any word w, there must exist some n (depending possibly on w) such that $w \in L(M)$ and $w \in L_n(w)$ have the same truth value. This n can be read as the associated tolerated level of perturbation. It quantifies the tolerated level of robustness. Now, we can always consider that this function that associates n to w depends only on its length (as there are finitely many words of a given length, and as we can always replace n by a bigger n).

Formally: assuming a language $L = L(M)$ is robust means $L = L(M) = L_\omega(\mathcal{M})$. Let us consider some length ℓ, and reason about words of length ℓ (i.e. about words of Σ^ℓ where Σ is the alphabet of the Turing machine). We must have $L \cap \Sigma^\ell = L(\mathcal{M}) \cap \Sigma^\ell = L_\omega(\mathcal{M}) \cap \Sigma^\ell$. Now, by definition of $L_\omega(M)$, $L_\omega(M) \cap \Sigma^\ell$ is necessarily $L_n(M) \cap \Sigma^\ell$ for some n. Consequently, we must have $L(\mathcal{M}) \cap \Sigma^\ell = L_n(M) \cap \Sigma^\ell$ for some $n = s(\ell)$ for a robust language.

In other words, for a robust language, we have necessarily

$$L = L(\mathcal{M}) = L_{\{s\}}(\mathcal{M})$$

for some function s, for the coming definition. This function $n = s(\ell)$ quantifies the tolerated level of robustness. If one prefers, a robust language is necessarily s-robust for some s, according to Definition 17.

▶ **Definition 16** (Level of robustness n given by s). *Given a function $s : \mathbb{N} \to \mathbb{N}$, we write $L_{\{s\}}(\mathcal{M})$ for the set of words accepted by $\mathcal{M}$ with space perturbation s: $L_{\{s\}}(\mathcal{M}) = \{w |\ w \in L_{s(\ell(w))}(\mathcal{M})\}$.*

▶ **Definition 17** (s-robust language). *We say that a robust language is s-robust, when $L = L(\mathcal{M}) = L_{\{s\}}(\mathcal{M})$.*

It is natural to consider the case where the function s is a polynomial. It turns out that this corresponds to (and is even a characterisation of) PSPACE:

▶ **Theorem 18** (Polynomial robustness $\Leftrightarrow$ PSPACE). *$L \in$ PSPACE iff for some $\mathcal{M}$ and some polynomial p, $L = L(\mathcal{M}) = L_{\{p\}}(\mathcal{M})$.*

Proof. ($\Rightarrow$) If M always terminates and works in polynomial space, then there exists a polynomial $q(\cdot)$ that bounds the size of the used part of the tape of M. Considering a polynomial $p \geq q + 2$, we have for $n \in N$ $L_{p(\ell(w))}(\mathcal{M}) \subseteq L(M)$. We always have the other inclusion.

($\Leftarrow$) $L_{p(n)}(\mathcal{M}) \in PSPACE$ is direct from the definitions. ◀

3 Embedding TMs into dynamical systems

Many authors have embedded TMs in various classes of dynamical systems to get undecidability or various hardness results. We present in this section how this is usually done, to point out where (intuitive) (non-)robustness issues appear.

Generally speaking, the trick is the following. If we forget about blanks, assuming alphabet $\Sigma = \{0, 1\}$, we can consider that $\mathcal{C}_\mathcal{M} \subseteq \mathcal{C} = \mathbb{N} \times \Sigma^* \times \Sigma^*$. Fix some encoding function of configurations into a vector of real numbers: $\Upsilon : \mathcal{C} \to \mathbb{R}^d$, with $d \in \mathbb{N}$. For example, $\Upsilon(q, w_1, w_2) = (q, \gamma(w_1), \gamma(w_2))$ with $\gamma : \Sigma^* \to \mathbb{R}$ taken as:

- the encoding $\gamma_\mathbb{N}$ mapping the word $w = a_1 \dots a_n$ to the integer whose binary expression is w,
- or $\gamma_{[0,1]}$ mapping w to the real number of $[0, 1]$ whose binary expansion is w,
- or more generally, $\gamma^k_{[0,1]}$ or $\gamma^k_\mathbb{N}$, using base k instead of base 2 for $k \geq 2$,
- or $\gamma'^k_{[0,1]}$ mapping w to $(\gamma^k_{[0,1]}(w), \ell(w))$.

Consider a function $\mathbf{f} : X \subseteq \mathbb{R}^d \to X$ such that for any configuration C, denoting by C' the next configuration, $\mathbf{f}(\Upsilon(C)) = \Upsilon(C')$: one step of the TM corresponds to one step in the dynamical system $(X, \mathbf{f})$ with respect to γ. Then, the diagram of the execution commutes for any number of steps:

$$\begin{array}{ccccccc}
C & \xrightarrow{\vdash} & C' & \xrightarrow{\vdash} & C'' & \xrightarrow{\vdash} & C''' \dashrightarrow \\
\Upsilon\downarrow & & \Upsilon\downarrow & & \Upsilon\downarrow & & \Upsilon\downarrow \\
\Upsilon(C) & \xrightarrow{\mathbf{f}} & \Upsilon(C') & \xrightarrow{\mathbf{f}} & \Upsilon(C'') & \xrightarrow{\mathbf{f}} & \Upsilon(C''') \dashrightarrow
\end{array}$$

The questions related to the existence of trajectories in the dynamical system $(X, \mathbf{f})$ associated with the TM correspond then to the questions about the existence of trajectories over $(X, \mathbf{f})$. Specifically, it provides c.e.-hardness of reachability for various classes of dynamical systems, as it is for TM. Call such a situation a *step-by-step* emulation. However, encodings such as $\gamma_{[0,1]}$, whose image is compact, map intrinsically distinct configurations to points arbitrarily close to each other (a sequence over a compact must have some accumulation point). Encodings like $\gamma_\mathbb{N}$ do not have a compact image but involve emulations with arbitrarily large integers, which is another issue. These observations led to the already mentioned (informal) conjecture that undecidabilty/hardness may hold only for non-robustness systems. This leads to discussing now formally robustness issues for general dynamical systems over $\mathbb{R}^d$ for some $d \in \mathbb{N}$.

4 Discrete-Time Dynamical Systems

We now study the case of general discrete-time systems. We aim at focusing on discrete-time systems of type $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^d$.

4.1 The case of rational systems

For clarity, as this general case requires to talk about computability issues on the reals (we do so later in Section 4.2) we first focus on the case of systems of type $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^d$ such that $\mathbf{f}(\mathbb{Q}) \subseteq \mathbb{Q}$. In other words, we first focus on the case of rational systems, i.e. $\mathbf{f} : \mathbb{Q}^d \to \mathbb{Q}^d$ (possibly obtained as the restriction to the rationals of a function over the reals).

A rational discrete-time dynamical system will be called $\mathbb{Q}$-computable when the function (hence from the rationals to the rationals) is. A rational discrete-time dynamical system will be called Lipschitz when the function is: there exists some constant K such that $d(\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{y})) \leq Kd(\mathbf{x}, \mathbf{y})$, for all $\mathbf{x}$, $\mathbf{y}$. It will be called locally Lipschitz when for any $\mathbf{z}$ and $\epsilon > 0$ there exists some constant K such that $d(\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{y})) \leq Kd(\mathbf{x}, \mathbf{y})$, for all $\mathbf{x}$, $\mathbf{y}$ in $B(\mathbf{z}, \epsilon)$.

With each rational discrete-time dynamical system $\mathcal{P}$ is associated its reachability relation $R^{\mathcal{P}}(\cdot, \cdot)$ on $\mathbb{Q}^d \times \mathbb{Q}^d$. Namely, for two rational points $\mathbf{x}$ and $\mathbf{y}$, $R^{\mathcal{P}}(\mathbf{x}, \mathbf{y})$ holds iff there exists a trajectory of $\mathcal{P}$ from $\mathbf{x}$ to $\mathbf{y}$. The reachability relation of a $\mathbb{Q}$-computable system is computably enumerable: to enumerate, a Turing machine can just simulate the dynamics.

▶ Remark 19. Article [1] considers only the special case of Piecewise affine (PAM) maps, as representative of discrete-time systems, which are particular $\mathbb{Q}$-computable Lipschitz systems.

▶ Remark 20. Here is an example of $\mathbb{Q}$-computable Lipschitz systems. Take some recurrent neural network, with d neurons, with the ReLU activation function, defined as $\mathrm{ReLU}(x) = \max(0, x)$. Its dynamic can be written as $\mathbf{x}_{t+1} = \mathrm{ReLU}(\mathbf{A}\mathbf{x} + \mathbf{B})$ where $\mathbf{A}$ is some $d \times d$ matrix with rational entries and $\mathbf{B}$ is some vector of dimension d with rational entries, where the ReLU function is applied componentwise.

Reachability for $\mathbb{Q}$-computable systems is undecidable and c.e.-complete:

▶ **Theorem 21** (Computational power of PAMs [23, 22, 1]). *Any c.e. language is reducible to the reachability relation of a PAM.*

Let us discuss whether undecidability still holds for "robust systems".

We apply the paradigm of small perturbations: consider a discrete-time dynamical system $\mathcal{P}$ with a function $\mathbf{f}$. For any $\epsilon > 0$ we consider the ϵ-perturbed system $\mathcal{P}_\epsilon$. Its trajectories are defined as sequences $\mathbf{x}_t$ satisfying $d(\mathbf{x}_{t+1}, \mathbf{f}(\mathbf{x}_t)) < \epsilon$ for all t. This non-deterministic system is considered as $\mathcal{P}$ submitted to a noise of magnitude ϵ. For convenience, we write $\mathbf{y} \in \mathbf{f}_\epsilon(\mathbf{x})$ as a synonym for $d(\mathbf{f}(\mathbf{x}), \mathbf{y}) < \epsilon$. We denote reachability in the system $\mathcal{P}_\epsilon$ by $R^{\mathcal{P}}_\epsilon(\cdot, \cdot)$.

All trajectories of a non-perturbed system $\mathcal{P}$ are also trajectories of the ϵ-perturbed system $\mathcal{P}_\epsilon$. If $\epsilon_1 < \epsilon_2$ then any trajectory of the ϵ_1-perturbed system is also a trajectory of the ϵ_2-perturbed PAM. Define $R^{\mathcal{P}}_\omega(\mathbf{x}, \mathbf{y})$ iff $\forall \epsilon > 0$ $R^{\mathcal{P}}_\epsilon(\mathbf{x}, \mathbf{y})$: this relation encodes reachability with arbitrarily small perturbing noise. From definitions:

▶ **Lemma 22** ([1]). *For any $0 < \epsilon_2 < \epsilon_1$ and any $\mathbf{x}$ and $\mathbf{y}$ the following implications hold: $R^{\mathcal{P}}(\mathbf{x}, \mathbf{y}) \Rightarrow R^{\mathcal{P}}_\omega(\mathbf{x}, \mathbf{y}) \Rightarrow R^{\mathcal{P}}_{\epsilon_2}(\mathbf{x}, \mathbf{y}) \Rightarrow R^{\mathcal{P}}_{\epsilon_1}(\mathbf{x}, \mathbf{y})$.*

▶ **Theorem 23** (Perturbed reachability is co-c.e.). *Consider a locally Lipschitz $\mathbb{Q}$-computable system whose domain $X \subseteq \mathbb{R}^d$ is a closed rational box[3]. Then the relation $R^{\mathcal{P}}_\omega(\mathbf{x}, \mathbf{y}) \subseteq \mathbb{Q}^d \times \mathbb{Q}^d$ is in the class Π_1.*

Proof. This extends [1, Theorem 5], using an alternative proof. As $\mathbf{f}$ is locally Lipschitz and X is compact, we know that $\mathbf{f}$ is Lipschitz: there exists some $L > 0$ so that $d(\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{y})) \leq L \cdot d(\mathbf{x}, \mathbf{y})$. For every $\delta = 2^{-m}$, $m \in \mathbb{N}$, we associate some graph $G_m = (V_\delta, \to_\delta)$: its vertices, denoted by $(\mathcal{V}_i)_i$, correspond to some finite discretisation and covering of compact X by rational open balls $\mathcal{V}_i = B(\mathbf{x}_i, \delta_i)$ of radius $\delta_i < \delta$. There is an edge from $\mathcal{V}_i$ to $\mathcal{V}_j$ in this graph, that is to say $\mathcal{V}_i \to_\delta \mathcal{V}_j$, iff $B(\mathbf{f}(\mathbf{x}_i), (L+1)\delta) \cap \mathcal{V}_j \neq \emptyset$. With our hypothesis on the domain, such a graph can be effectively obtained from m, considering a suitable discretisation of the rational box X.

[3] Recall that a rational box X is a subset of real numbers, not of rational numbers. Consequently, such an X is compact.

▷ Claim 1. assume $R^{\mathcal{P}}_{\epsilon}(\mathbf{x},\mathbf{y})$ with $\mathbf{x} \in \mathcal{V}_i$ for $\epsilon = 2^{-n}$. Then $\mathcal{V}_i \overset{*}{\to}_{\epsilon} \mathcal{V}_j$ for all $\mathcal{V}_j$ with $\mathbf{y} \in \mathcal{V}_j$. This holds as the graph for $\delta = \epsilon$ is made to always have more trajectories/behaviours than $R^{\mathcal{P}}_{\epsilon}$.

▷ Claim 2. for any $\epsilon = 2^{-n}$, there is some $\delta = 2^{-m}$ so that if we have $\mathcal{V}_i \overset{*}{\to}_{\delta} \mathcal{V}_j$ then $R^{\mathcal{P}}_{\epsilon}(\mathbf{x},\mathbf{y})$ whenever $\mathbf{x} \in \mathcal{V}_i$, $\mathbf{y} \in \mathcal{V}_j$.

Claim 2 says that $\neg R^{\mathcal{P}}_{\epsilon}(\mathbf{x},\mathbf{y})$ implies $\neg(\mathcal{V}_i \overset{*}{\to}_{\delta} \mathcal{V}_j)$ whenever $\mathbf{x} \in \mathcal{V}_i$, $\mathbf{y} \in \mathcal{V}_j$, for the corresponding δ.

From the two above items, $\neg R^{\mathcal{P}}_{\omega}(\mathbf{x},\mathbf{y})$ holds iff for some $\delta = 2^{-m}$, $\neg(\mathcal{V}_i \overset{*}{\to}_{\delta} \mathcal{V}_j)$ for some $\mathcal{V}_i$, $\mathcal{V}_j$ with $\mathbf{x} \in \mathcal{V}_i$, $\mathbf{y} \in \mathcal{V}_j$. This holds iff for some integer m, NOPATH$(G_m, \mathcal{V}_i, \mathcal{V}_j)$ for some $\mathcal{V}_i$, $\mathcal{V}_j$ with $\mathbf{x} \in \mathcal{V}_i$, $\mathbf{y} \in \mathcal{V}_j$. The latter property is c.e., as it is a union of decidable sets (uniform in m), as NOPATH$(G_m, \mathcal{V}_i, \mathcal{V}_j)$ is a decidable property over finite graph G_m. ◀

▶ **Definition 24** (Robust reachability relation)**.** *We say that the reachability relation is* robust *when* $R^{\mathcal{P}} = R^{\mathcal{P}}_{\omega}$.

We get the *"robustness conjecture"*:

▶ **Corollary 25** (Robust ⇒ decidable, [1, Corollary 5])**.** *Assume the hypotheses of Theorem 23. If the relation* $R^{\mathcal{P}}$ *is robust then it is decidable.*

4.1.1 Robustness versus decidability and δ-decidability

We now discuss how far the above statement is to a characterisation of decidability.

There is indeed a converse property if some condition is added. Before stating this in Corollary 32, we relate robustness to the concept of δ-decidability in [15] and also the existence of some witness of non-reachability.

Given $\mathbf{x}$, $R^{\mathcal{P}}(\mathbf{x})$ denotes the set of the points $\mathbf{y}$ reachable from $\mathbf{x}$: $R^{\mathcal{P}}(\mathbf{x}) = \{\mathbf{y} | R^{\mathcal{P}}(\mathbf{x},\mathbf{y})\}$. This is also the smallest set such that $\mathbf{x} \in R^{\mathcal{P}}(\mathbf{x})$ and $\mathbf{f}(R^{\mathcal{P}}(\mathbf{x})) \subseteq R^{\mathcal{P}}(\mathbf{x})$.

▶ **Definition 26.** *$R^{\mathcal{P}}(\mathbf{x},\mathbf{y})$ is said to be ϵ-far from being true if there is $\mathcal{R}^* \subseteq X$ so that*
1. $\mathbf{x} \in \mathcal{R}^*$,
2. $\mathbf{f}_{\epsilon}(\mathcal{R}^*) \subseteq \mathcal{R}^*$,
3. $\mathbf{y} \notin \mathcal{R}^*$.

When this holds, we have $\neg R^{\mathcal{P}}(\mathbf{x},\mathbf{y})$. Indeed, for all $\epsilon > 0$, $R^{\mathcal{P}}_{\epsilon}(\mathbf{x}) = \{\mathbf{y} | R^{\mathcal{P}}_{\epsilon}(\mathbf{x},\mathbf{y})\}$ is the smallest set satisfying $\mathbf{x} \in R^{\mathcal{P}}_{\epsilon}(\mathbf{x})$ and $\mathbf{f}_{\epsilon}(R^{\mathcal{P}}_{\epsilon}(\mathbf{x})) \subseteq R^{\mathcal{P}}_{\epsilon}(\mathbf{x})$. Thus, as $\mathcal{R}^*$ also satisfies these properties by the first two conditions, $R^{\mathcal{P}}_{\epsilon}(\mathbf{x}) \subseteq \mathcal{R}^*$ and hence $\mathbf{y} \notin R^{\mathcal{P}}(\mathbf{x})$ as $R^{\mathcal{P}}(\mathbf{x}) \subseteq R^{\mathcal{P}}_{\epsilon}(\mathbf{x}) \subseteq \mathcal{R}^*$. Then $\mathbf{y} \notin \mathcal{R}^*$ from the third condition.

In other words, $\mathcal{R}^*$ is a *witness* of the non-reachability of $\mathbf{y}$ from $\mathbf{x}$. We will say that it is *at level* ϵ. This provides a relation to δ-decidability considered in [15]:

▶ **Proposition 27** (Robust ⇔ Reachability relation is true or ϵ-far from being true)**.** *We have* $R^{\mathcal{P}}_{\omega} = R^{\mathcal{P}}$ *if and only if for all* $\mathbf{x},\mathbf{y} \in \mathbb{Q}^d$*, either*
1. $R^{\mathcal{P}}(\mathbf{x},\mathbf{y})$ *is true*
2. *or* $R^{\mathcal{P}}(\mathbf{x},\mathbf{y})$ *is false and there exists* $\epsilon > 0$ *such that it is ϵ-far from being true.*

Proof.

(⇒): For all $\epsilon > 0$, $R^{\mathcal{P}}_{\epsilon}(\mathbf{x})$ satisfies $\mathbf{x} \in R^{\mathcal{P}}_{\epsilon}(\mathbf{x})$ and $\mathbf{f}_{\epsilon}(R^{\mathcal{P}}_{\epsilon}(\mathbf{x})) \subseteq R^{\mathcal{P}}_{\epsilon}(\mathbf{x})$ (this is even the smallest set such that this holds). Let $\mathbf{y} \in \mathbb{Q}^d$, let us assume that $R^{\mathcal{P}}(\mathbf{x},\mathbf{y}) = R^{\mathcal{P}}_{\omega}(\mathbf{x},\mathbf{y})$ is not true. Then, there exists ϵ such that $R^{\mathcal{P}}_{\epsilon}(\mathbf{x},\mathbf{y})$ is false, i.e. $\mathbf{y} \notin R^{\mathcal{P}}_{\epsilon}(\mathbf{x})$. Consider $\mathcal{R}^* = R^{\mathcal{P}}_{\epsilon}(\mathbf{x})$. Then, $\mathbf{x} \in \mathcal{R}^*$ and from the first paragraph $\mathbf{f}_{\epsilon}(\mathcal{R}^*) \subseteq \mathcal{R}^*$ and $\mathbf{y} \notin \mathcal{R}^*$.

($\Leftarrow$): When $R^{\mathcal{P}}(\mathbf{x},\mathbf{y})$ is true, for all $\epsilon > 0$, $R^{\mathcal{P}}_{\epsilon}(\mathbf{x},\mathbf{y})$ is true, so $R^{\mathcal{P}}_{\omega}(\mathbf{x},\mathbf{y})$ is. When $R^{\mathcal{P}}(\mathbf{x},\mathbf{y})$ is false, by hypothesis, $R^{\mathcal{P}}(\mathbf{x},\mathbf{y})$ is ϵ-far from being true for some $\epsilon > 0$: there exists a set $\mathcal{R}^*$ satisfying $\mathbf{x} \in \mathcal{R}^*$ and $\mathbf{f}_{\epsilon}(\mathcal{R}^*) \subseteq \mathcal{R}^*$. As $R^{\mathcal{P}}_{\epsilon}(\mathbf{x})$ is the smallest such set, $R^{\mathcal{P}}_{\epsilon}(\mathbf{x}) \subseteq \mathcal{R}^*$. As $\mathbf{y} \notin \mathcal{R}^*$, $\mathbf{y} \notin R^{\mathcal{P}}_{\epsilon}(\mathbf{x})$. Hence $R^{\mathcal{P}}_{\omega}(\mathbf{x},\mathbf{y})$ is false. ◀

We say that a subset R^* of X is ϵ-rejecting (with respect to $\mathbf{y}$) if it satisfies 2. and 3. of Definition 26: that is to say, $\mathbf{f}_{\epsilon}(R^*) \subseteq R^*$, and $\mathbf{y} \notin R^*$. A trajectory reaching such a R^* will never leave it.

▶ **Definition 28.** *A system is* eventually decisional *if for all* $\mathbf{x}$, $\mathbf{y}$, *there is some* R^* *ϵ-rejecting (with respect to* $\mathbf{y}$*) so that either the trajectory starting from* $\mathbf{x}$ *reaches* $\mathbf{y}$ *or, when not, it reaches* R^**.*

We come back to the converse of Corollary 25: from Proposition 27, a robust dynamical system (i.e. $R^{\mathcal{P}}_{\omega} = R^{\mathcal{P}}$) is eventually decisional, by considering $R^* = \mathcal{R}^*$ for the $\mathcal{R}^*$ given by item 2) there. Conversely:

▶ **Lemma 29.** *Take* $\mathbf{x}$ *and* $\mathbf{y}$ *with* $R^{\mathcal{P}}_{\omega}(\mathbf{x},\mathbf{y})$ *but not* $R^{\mathcal{P}}(\mathbf{x},\mathbf{y})$*. For* $\mathbf{f}$ *Lipschitz, the trajectory starting from* $\mathbf{x} \in \mathbb{Q}^d$ *can not reach any ϵ-rejecting subset.*

Proof. By contradiction, assume the trajectory starting from $\mathbf{x}$ reaches an ϵ-rejecting R^*. By considering one more step, we can assume that it reaches the interior of R^* for the first time at t, since, if it reaches the frontier at $\mathbf{x}^*$, $B(\mathbf{f}(\mathbf{x}^*),\epsilon) \subseteq R^*$ and $\mathbf{f}(\mathbf{x}^*)$ is in the interior of that ball. From $\mathbf{x}$ the position at time t remains at a positive distance of $\mathbf{y}$. As $\mathbf{f}$ is Lipschitz, the t-th iteration of $\mathbf{f}$ is. So, there exists $0 < \epsilon' < \epsilon$ taken sufficiently small so that $R^{\mathcal{P}}_{\epsilon'}$ intersects the interior of R^* and remains at a positive distance of $\mathbf{y}$. Once in R^*, ϵ'-perturbed trajectories stay in it ($\epsilon' < \epsilon$). We get $\mathbf{y} \notin R^{\mathcal{P}}_{\epsilon'}$. Thus $\neg R^{\mathcal{P}}_{\omega}(\mathbf{x},\mathbf{y})$: contradiction. ◀

▶ **Corollary 30.** *Consider a Lipschitz rational dynamical system. It is robust iff it is eventually decisional.*

We can even compute the witnesses under the hypotheses of Theorem 23. A dynamical system is *effectively eventually decisional* when there is an algorithm such that, given $\mathbf{x}$ and $\mathbf{y}$, it outputs an R^* in the form of the union of rational balls. We can reinforce Corollary 25:

▶ **Proposition 31.** *Assume the hypotheses of Theorem 23. If* $R^{\mathcal{P}}_{\omega} = R^{\mathcal{P}}$ *then* $R^{\mathcal{P}}$ *is computable and the system is effectively eventually decisional.*

Proof. The proof of Theorem 23 shows that when $R^{\mathcal{P}}_{\omega}(\mathbf{x},\mathbf{y})$ is false, then $R^{\mathcal{P}}_{\epsilon}(\mathbf{x},\mathbf{y})$ is false for some $\epsilon = 2^{-n}$ and there is a $\delta = 2^{-m}$ and some graph G_m with vertices $\mathcal{V}_i$ and $\mathcal{V}_j$, $\mathbf{x} \in \mathcal{V}_i$, $\mathbf{y} \in \mathcal{V}_j$ and $\neg(\mathcal{V}_i \xrightarrow{*}_{\delta} \mathcal{V}_j)$. Denote by R^{G_m} the union of the vertices $\mathcal{V}_k$ such that $\mathcal{V}_i \xrightarrow{*}_{\delta} \mathcal{V}_k$, $\mathbf{x} \in \mathcal{V}_i$ in G_m. Consider $\mathcal{R}^* = R^{G_m}$: this is a witness at level $\delta = 2^{-m}$ from the properties of the construction. Then m can be found by testing increasing m until a proper graph is found. The corresponding $\mathcal{R}^* = R^{G_m}$ of the first graph found will be a witness at level $\delta = 2^{-m}$. ◀

The reachability relation of an effectively eventually decisional system is necessarily decidable (given $\mathbf{x}$ and $\mathbf{y}$, compute the path until it reaches $\mathbf{y}$ (then accept), or R^* (then reject)):

▶ **Corollary 32** (Decidable $\Leftrightarrow$ Robust, for eventually decisional systems)**.** *Under the hypotheses of Theorem 23,* $R^{\mathcal{P}}$ *is robust iff* $R^{\mathcal{P}}$ *is decidable and* $R^{\mathcal{P}}$ *is effectively eventually decisional iff* $R^{\mathcal{P}}_{\omega}$ *is effectively eventually decisional.*

4.1.2 Complexity issues

Assume the dynamical system is robust. Hence, for all $\mathbf{x}, \mathbf{y} \in \mathbb{Q}$, there exists ϵ (depending on $\mathbf{x}$, $\mathbf{y}$) such that $R^{\mathcal{P}}(\mathbf{x}, \mathbf{y})$ and $R^{\mathcal{P}}_{\epsilon}(\mathbf{x}, \mathbf{y})$ have the same truth value (unchanged by smaller ϵ). It is then natural to quantify the level of required robustness according to $\mathbf{x}$ and $\mathbf{y}$, i.e. on the value ϵ. As we may always assume $\epsilon = 2^{-n}$ for some $n \in \mathbb{N}$, we write $R^{\mathcal{P}}_n$ for $R^{\mathcal{P}}_{\epsilon=2^{-n}}$ and we introduce:

▶ **Definition 33** (Level of robustness ϵ given by s)**.** *Given a function $s : \mathbb{N} \to \mathbb{N}$, we write $R^{\mathcal{P}}{}_{\{s\}}$ for the relation defined as: for any rational points $\mathbf{x}$ and $\mathbf{y}$ the relation holds iff $R^{\mathcal{P}}_{s(\ell(\mathbf{x})+\ell(\mathbf{y}))}(\mathbf{x}, \mathbf{y})$.*

A robust dynamical system is necessarily s-robust for some function s, according to the next definition: this follows from exactly the same arguments as the ones we used for the related concepts for Turing machines. This function s quantifies the tolerated level of robustness.

▶ **Definition 34** (s-robust language)**.** *We say that a dynamical system is s-robust, when $R^{\mathcal{P}} = R^{\mathcal{P}}{}_{\{s\}}$.*

We can then naturally consider the case where s is a polynomial: considering robustness to polynomial perturbations corresponds to PSPACE:

▶ **Theorem 35.** *Consider a locally Lipschitz $\mathbb{Q}$-computable system, with $\mathbf{f} : \mathbb{Q} \to \mathbb{Q}$ computable in polynomial time, whose domain X is a closed rational box. Given some polynomial p, $R^{\mathcal{P}}{}_{\{p\}} \in$ PSPACE.*

Proof. From Theorem 23, for all n there exists some m (depending on n), such that $R^{\mathcal{P}}_n(\mathbf{x}, \mathbf{y})$ and $R^{G_m}(\mathbf{x}, \mathbf{y})$ have the same truth value, where R^{G_m} denotes reachability in the graph G_m. With the hypotheses, given $\mathbf{x}$ and $\mathbf{y}$, we can determine whether $R^{\mathcal{P}}{}_{\{p\}}(\mathbf{x}, \mathbf{y})$, by determining the truth value of $R^{\mathcal{P}}_n(\mathbf{x}, \mathbf{y})$, taking n polynomial in $\ell(\mathbf{x}) + \ell(\mathbf{y})$. From Theorem 23, the corresponding m is linearly related to n. The analysis of Corollary 12 shows that the truth value of $R^{G_m}(\mathbf{x}, \mathbf{y})$ can be determined in space polynomial in m. ◀

▶ **Theorem 36** (Polynomially robust to precision ⇒ PSPACE)**.** *With the same hypotheses, if $R^{\mathcal{P}} = R^{\mathcal{P}}{}_{\{p\}}$ for some polynomial p, then $R^{\mathcal{P}} \in$ PSPACE.*

This is even a characterisation of PSPACE:

▶ **Theorem 37** (Polynomially robust to precision ⇔ PSPACE)**.** *Any* PSPACE *language is reducible to PAM's reachability relation: $R^{\mathcal{P}} = R^{\mathcal{P}}{}_{\{p\}}$, for some polynomial p.*

Assuming the hypotheses of Theorem 36, when $R^{\mathcal{P}} = R^{\mathcal{P}}{}_{\{p\}}$ for some polynomial p, we also see that we can determine a witness of $\neg R^{\mathcal{P}}(\mathbf{x}, \mathbf{y})$ in polynomial space (using a suitable representation of it).

4.2 The case of computable systems

We now consider the case of general discrete-time dynamical systems. Then $\mathbf{f}$ may take some non-rational values and we need the notion of computability of functions over the reals: this requires the model of computable analysis: see e.g. [32] or [12] for full presentations.

We review the most basic ideas of computable analysis in the next subsection.

4.2.1 Some basics of computable analysis

The idea behind classical computability and complexity is to fix some representations of objects (such as graphs, integers, etc, ...) using finite words over some finite alphabet, say $\Sigma = \{0, 1\}$ and to say that such an object is computable when such a representation can be produced using a Turing machine. The computable analysis is designed to be able to also talk about objects such as real numbers, functions over the reals, closed subsets, compacts subsets, ..., which cannot be represented by finite words over Σ (a clear reason for it is that such words are countable while the set $\mathbb{R}$, for example, is not). However, they can be represented by some infinite words over Σ and the idea is to fix such representations for these various objects, called *names*, with suitable computable properties. In particular, in all the following proposed representations, it can be proved that an object is computable iff it has some computable representation.

▶ Remark 38. Here the notion of computability involved is one of Type 2 Turing machines, that is to say, computability over possibly infinite words: the idea is that such a machine has some read-only input tape(s), that contains the input(s), which can correspond to either a finite or infinite word(s), a read-write working tape and one (or several) write-only output tape(s). It evolves as a classical Turing machine, the only difference being that we consider it outputs an infinite word when it writes forever the symbols of that word on its (or one of its) write-only infinite output tape(s): see [32] for details.

A name for a point $\mathbf{x} \in \mathbb{R}^d$ is a sequence (I_n) of nested open rational balls with $I_{n+1} \subseteq I_n$ for all $n \in \mathbb{N}$ and $\{x\} = \bigcap_{n\in\mathbb{N}} I_n$. Such a name can be encoded as an infinite sequence of symbols.

We call a real function $f :\subseteq \mathbb{R} \to \mathbb{R}$ computable, iff some (Type 2 Turing) machine maps any name of any $x \in \mathrm{dom}(f)$ to a name of $f(x)$. For real functions $\mathbf{f} :\subseteq \mathbb{R}^n \to \mathbb{R}$ we consider machines reading n names in parallel.

It can be proved that a computable function is necessarily continuous. A name for a function $\mathbf{f}$ is a list of all pairs of open rational balls (I, J) such that $\mathbf{f}(cls(I)) \subseteq J$. Following the above remark, one can prove that a real function is computable iff it has some computable name.

A name for a closed set F is a sequence (I_n) of all open rational balls such that $cls(I_n) \cap F = \emptyset$ and a sequence (J_n) of all open rational balls such that $J_n \cap F \neq \emptyset$.

Given some closed set F, the distance function $d_F : \mathbb{R}^n \to \mathbb{R}$ is defined by $d_F(x) := \inf_{y\in F} d(x, y)$. Closed subset $F \subseteq \mathbb{R}^n$ is computable iff its distance function $d_A : \mathbb{R}^n \to \mathbb{R}$ is ([32, Corollary 5.1.8]). A name for a compact K is a name of F as a closed set and an integer L such that $K \subseteq B(0, L)$.

A closed set is called computably-enumerable closed if one can effectively enumerate the rational open balls intersecting it: $\{(q, \epsilon) \in \mathbb{Q}^n \times \mathbb{Q}_+ \mid \mathrm{B}(q, \epsilon) \cap A \neq \emptyset\}$ is computably enumerable ([12, Definition 5.13],[32, Definition 5.1.1]). A closed set is called co-computably-enumerable closed if one can effectively enumerate the rational closed balls in its complement: the set $\left\{(q, \epsilon) \in \mathbb{Q}^n \times \mathbb{Q}_+ \mid \overline{\mathrm{B}}(q, \epsilon) \subseteq U\right\}$ is computably enumerable ([12, Definition 5.10],[32, Definition 5.1.1]).

We need also the concept of polynomial time computable function in computable analysis: see [21]. In short, a quickly converging name of $\mathbf{x} \in \mathbb{R}^d$ is a name of $\mathbf{x}$, with I_n of radius $< 2^{-n}$. A function $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^{d'}$ is said to be computable in polynomial time, if there is some oracle TM M, such that, for all $\mathbf{x}$, given any fast converging name of $\mathbf{x}$ as an oracle, given n, M produces some open rational ball of radius $< 2^{-n}$ containing $\mathbf{f}(\mathbf{x})$, in a time polynomial in n.

4.2.2 Computable systems

A system is said computable if the function $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^d$ is. From the model of computable analysis, given the name of $\mathbf{f}$, $\mathbf{x}, \mathbf{y} \in \mathbb{Q}$, it is impossible in general to tell effectively if $\mathbf{f}(\mathbf{x}) = \mathbf{y}$. Thus, given some rational ball $B(\mathbf{y}, \delta)$, we have to forbid "frontier reachability": $B(\mathbf{y}, \delta)$ would not be reachable, but its frontier $\overline{B}(\mathbf{y}, \delta) - B(\mathbf{y}, \delta)$ would. A natural question arises: given some rational ball with the promise that either $B(\mathbf{y}, \delta)$ is reachable (that case implies that $\overline{B}(\mathbf{y}, \delta)$ is), or that $\overline{B}(\mathbf{y}, \delta)$ is not, decide which possibility holds. We call this the *ball (decision) problem*. From definitions from CA, when $R^{\mathcal{P}}(\mathbf{x})$ is a closed set, $R^{\mathcal{P}}(\mathbf{x})$ is a computable closed set iff its associated ball problem is algorithmically solvable.

For computable systems, the ball decision problem is c.e: we mean, there is a Turing machine whose halting set intersected with the rational balls satisfying the promise is the set of positive instances. Indeed, just simulate the system's evolution, starting from $\mathbf{x}$ until step T, with increasing precision and T, until one finds the guarantee that $\mathbf{x}_T$ at time T remains in $B(\mathbf{y}, \delta')$ for some $\delta' < \delta$. If the ball is reachable, it will terminate by computing a sufficient approximation of the corresponding $\mathbf{x}_T$. It cannot terminate without guaranteeing reachability. It is not co-c.e. in general.

▶ Remark 39. Our framework for discussing the computability of sets is similar to the concept of a maximally partially decidable set, formalised in [24, 25]. Similar ideas have also been implicitly used in many other articles considering various real problems, using computable analysis. A similar formalisation is also considered in [5].

To a discrete-time system, we can also associate its reachability relation $R^{\mathcal{P}}(\cdot, \cdot, \cdot)$ over $\mathbb{Q}^d \times \mathbb{Q}^d \times \mathbb{N}$. For two points $\mathbf{x}, \mathbf{y} \in \mathbb{Q}$, $\eta = 2^{-p}$, encoded by $p \in \mathbb{N}$, $R^{\mathcal{P}}(\mathbf{x}, \mathbf{y}, p)$ iff there exists a trajectory of $\mathcal{P}$ from $\mathbf{x}$ to $\overline{B}(\mathbf{y}, \eta)$. We define $R^{\mathcal{P}}_{\epsilon}$ similarly and $R^{\mathcal{P}}_{\omega} = \bigcap_{\epsilon} R^{\mathcal{P}}_{\epsilon}$. This relation encodes reachability with arbitrarily small perturbing noise to some closed ball.

▶ **Lemma 40.** *For any $0 < \epsilon_2 < \epsilon_1$ and any $\mathbf{x}$ and $\mathbf{y}$, η, the following implications hold:* $R^{\mathcal{P}}(\mathbf{x}, \mathbf{y}, p) \Rightarrow R^{\mathcal{P}}_{\omega}(\mathbf{x}, \mathbf{y}, p) \Rightarrow R^{\mathcal{P}}_{\epsilon_2}(\mathbf{x}, \mathbf{y}, p) \Rightarrow R^{\mathcal{P}}_{\epsilon_1}(\mathbf{x}, \mathbf{y}, p)$.

Given $\mathbf{x}$ and $0 < \epsilon_2 < \epsilon_1$, $R^{\mathcal{P}}(\mathbf{x}) \subseteq R^{\mathcal{P}}_{\epsilon_2}(\mathbf{x}) \subseteq cls(R^{\mathcal{P}}_{\epsilon_2}(\mathbf{x})) \subseteq R^{\mathcal{P}}_{\epsilon_1}(\mathbf{x}) \subseteq cls(R^{\mathcal{P}}_{\epsilon_1}(\mathbf{x}))$. Hence, $R^{\mathcal{P}}_{\omega}(\mathbf{x}) = \bigcap_{\epsilon>0} R^{\mathcal{P}}_{\epsilon}(\mathbf{x}) = \bigcap_{\epsilon>0} cls(R^{\mathcal{P}}_{\epsilon}(\mathbf{x}))$ is a closed set.

▶ **Theorem 41** (Perturbed reachability is co-r.e.)**.** *Consider a locally Lipschitz computable system whose domain X is a computable compact. $R^{\mathcal{P}}_{\omega}(\mathbf{x}, \mathbf{y}, p) \subseteq \mathbb{Q}^d \times \mathbb{Q}^d \times \mathbb{N}$ is in Π_1.*

This can be considered as extending [9, Theorem 13], established in a very simpler framework.

▶ **Corollary 42** (Robust ⇒ decidable)**.** *Assume Theorem 41's hypotheses and that for all rational $\mathbf{x}$, $R^{\mathcal{P}}(\mathbf{x})$ is closed and $R^{\mathcal{P}}(\mathbf{x}) = R^{\mathcal{P}}_{\omega}(\mathbf{x})$. Then, the ball decision problem is decidable.*

Proof. Given some instance $B(\mathbf{y}, \delta)$ of the ball problem, run in parallel the c.e. algorithm for it (and when its termination is detected, accepts) and the c.e. algorithm for $\left(R^{\mathcal{P}}(\mathbf{x})\right)^c = \left(R^{\mathcal{P}}_{\omega}(\mathbf{x})\right)^c$ (and when its termination is detected, rejects). ◀

4.2.3 Complexity issues

▶ **Definition 43** (Level of robustness ϵ given by s)**.** *Given some function $s : \mathbb{N} \to \mathbb{N}$, we write $R^{\mathcal{P}}{}_{\{s\}}$ as: for two rational points $\mathbf{x}$ and $\mathbf{y}$ and p, the relation holds iff $R^{\mathcal{P}}_{s(\ell(\mathbf{x})+\ell(\mathbf{y})+p)}(\mathbf{x}, \mathbf{y}, p)$.*

As before, a robust dynamical system is necessarily s-robust for some function s, according to the next definition. The function s quantifies the tolerated level of robustness.

▶ **Definition 44** (s-robust language). *We say that a dynamical system is s-robust, when $R^{\mathcal{P}} = R^{\mathcal{P}}{}_{\{s\}}$.*

▶ **Theorem 45.** *Take a locally Lipschitz system, with $\mathbf{f}$ polynomial time computable, whose domain X is a closed rational box. Then $R^{\mathcal{P}}{}_{\{p\}} \subseteq \mathbb{Q}^d \times \mathbb{Q}^d \times \mathbb{N} \in$ PSPACE, when p is a polynomial.*

Proof. The proof of Theorem 41 (like for Theorem 23) shows that when $R^{\mathcal{P}}_{\omega}(\mathbf{x}, \mathbf{y}, p)$ is false, then $R^{\mathcal{P}}_{\epsilon}(\mathbf{x}, \mathbf{y}, p)$ is false for some $\epsilon = 2^{-n}$. With the hypotheses, given $\mathbf{x}$, $\mathbf{y}$ and p, we take n polynomial in $\ell(\mathbf{x}) + \ell(\mathbf{y}) + p$. The corresponding m is polynomially related to n (linear in n). An analysis similar to Theorem 35, shows the truth value of $R^{G_m}(\mathbf{x}, \mathbf{y}, p)$ can be determined in space polynomial in m. ◀

Then, once again:

▶ **Theorem 46** (Polynomially robust to precision ⇒ PSPACE). *Assuming Theorem 45's hypotheses, and that for all rational $\mathbf{x}$, $R^{\mathcal{P}}(\mathbf{x})$ is closed and $R^{\mathcal{P}}(\mathbf{x}) = R^{\mathcal{P}}{}_{\{p\}}$ for a polynomial p. Then the ball decision problem is in* PSPACE.

5 Relating robustness to drawability

We can go further and prove geometric properties: in the previous sections, we associated with every discrete-time dynamical system a reachability relation over the rationals. But we could also see it as a relation over the reals and use the framework of computable analysis, regarding subsets of $\mathbb{R}^d \times \mathbb{R}^d$. From the statements of [32], the following holds:

▶ **Theorem 47.** *Consider a computable discrete-time system $\mathcal{P}$ whose domain is a computable compact. For all computable $\mathbf{x}$, $cls(R^{\mathcal{P}}(\mathbf{x})) \subseteq \mathbb{R}^d$ is a c.e. closed subset.*

A closed set is called *co-c.e. closed* if we can effectively enumerate the rational closed balls in its complement. Using proofs similar to Theorems 41 and 23:

▶ **Theorem 48.** *Consider a computable locally Lipschitz discrete-time system whose domain X is a computable compact. For all computable $\mathbf{x}$, $cls(R^{\mathcal{P}}_{\omega}(\mathbf{x})) \subseteq \mathbb{R}^d$ is a co-c.e. closed subset.*

▶ **Corollary 49** (Robust ⇒ computable)). *Assume the Theorem 48's hypotheses. If $R^{\mathcal{P}}$ is robust then for all computable $\mathbf{x}$, $cls(R^{\mathcal{P}}(\mathbf{x})) \subseteq \mathbb{R}^d$ is computable.*

For closed sets, the notion of computability can be interpreted as the possibility of being plotted with an arbitrarily chosen precision: $\mathbf{z}/2^n$ corresponds to a pixel at precision 2^{-n}, 1 is black (the pixel is plotted black), 0 is white (the pixel is plotted white).

▶ **Theorem 50** ([12]). *For a closed set $A \subseteq \mathbb{R}^k$, A is computable iff it can be plotted: there exists a computable function $f : \mathbb{N} \times \mathbb{Z}^k \to \mathbb{N}$ with $range(f) \subseteq \{0, 1\}$ and such that for all $n \in \mathbb{N}$ and $\mathbf{z} \in \mathbb{Z}^k$*

$$f(n, \mathbf{z}) = \begin{cases} 1 & \text{if } B(\frac{\mathbf{z}}{2^n}, 2^{-n}) \cap A \neq \emptyset, \\ 0 & \text{if } B(\frac{\mathbf{z}}{2^n}, 2.2^{-n}) \cap A = \emptyset, \\ 0 \text{ or } 1 & \text{otherwise.} \end{cases}$$

▶ **Corollary 51** (Robust ⇒ drawable)). *Assume Theorem 48's hypotheses. If $R^{\mathcal{P}}$ is robust then for all computable $\mathbf{x}$, $cls(R^{\mathcal{P}}(\mathbf{x})) \subseteq \mathbb{R}^d$ can be plotted.*

This is even effective in the name of $\mathbf{x}$ and $\mathbf{f}$. The converse holds with additional topological properties.

▶ **Theorem 52.** *Assume $R^{\mathcal{P}}$ is closed and can be plotted effectively in the name of $\mathbf{x}$ and $\mathbf{f}$. Then the system is robust, i.e. $R^{\mathcal{P}}_{\omega} = R^{\mathcal{P}}$.*

We prove a stronger statement: if $cls(R^{\mathcal{P}})$ can be plotted effectively in a name of $\mathbf{x}$ and $\mathbf{f}$, then $R^{\mathcal{P}}_{\omega}(\mathbf{x},\mathbf{y}) = R^{\mathcal{P}}(\mathbf{x},\mathbf{y})$ except maybe for $(\mathbf{x},\mathbf{y}) \in cls(R^{\mathcal{P}}) - R^{\mathcal{P}}$.

Proof. By Theorem 50, $cls(R^{\mathcal{P}})$ is computable which is equivalent to the computability of the distance function $d(\cdot, cls(R^{\mathcal{P}}))$ [32, Corollary 5.1.8]. It means that given a rational ball, a name for $\mathbf{x}$ and $\mathbf{y}$, with $\neg R^{\mathcal{P}}(\mathbf{x},\mathbf{y})$, the following procedure terminates when $(\mathbf{x},\mathbf{y}) \notin cls(R^{\mathcal{P}}) - R^{\mathcal{P}}$: compute a name of $d((\mathbf{x},\mathbf{y}), cls(R^{\mathcal{P}}(\mathbf{x})))$ until a strictly positive proof is found: $d((\mathbf{x},\mathbf{y}), cls(R^{\mathcal{P}}(\mathbf{x}))) = 0$ would mean $(\mathbf{x},\mathbf{y}) \in cls(R^{\mathcal{P}})$, but not in $R^{\mathcal{P}}$.

It answers by reading $m \in \mathbb{N}$ cells of the names of $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{f}$. It returns the same if the names are altered after m symbols. Thus, there exists a precision ϵ (related to m, usually 2^{-m} for exponentially fast convergence) so $\neg R^{\mathcal{P}}(\mathbf{x},\mathbf{y})$ remains true for an ϵ-neighborhood of $\mathbf{x}$ and $\mathbf{y}$ and unchanged by a small variation of $\mathbf{f}$. Hence, for all $\mathbf{x}$, $\mathbf{y}$, when $\neg R^{\mathcal{P}}(\mathbf{x},\mathbf{y})$, there exists some ϵ such that $\neg R^{\mathcal{P}}_{\epsilon}(\mathbf{x},\mathbf{y})$ ($\neg R^{\mathcal{P}}_{\omega}(\mathbf{x},\mathbf{y})$). When $R^{\mathcal{P}}(\mathbf{x},\mathbf{y})$ holds, $R^{\mathcal{P}}_{\omega}(\mathbf{x},\mathbf{y})$ holds. ◀

6 Continuous-time systems

The previous ideas can be extended to continuous-time or hybrid systems.

▶ **Definition 53.** *A* continuous-time dynamical system $\mathcal{P}$ *is given by a set $X \subseteq \mathbb{R}^d$ and some Ordinary Differential Equation of the form $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ on X.*

The maximal interval of existence of solutions can be non-computable, even for computable Ordinary Differential Equations (ODEs) [16]. To simplify, we assume the ODEs have solutions[4] defined over all $\mathbb{R}$. A trajectory of $\mathcal{P}$ starting at $\mathbf{x}_0 \in X$ is a solution of the differential equation with initial condition $\mathbf{x}(0) = \mathbf{x}_0$, defined as a continuous right-derivable function $\xi : \mathbb{R}^+ \to X$ such that $\xi(0) = \mathbf{f}(\mathbf{x}_0)$ and for every t, $\mathbf{f}(\xi(t))$ is equal to the right-derivative of $\xi(t)$. To each continuous-time dynamical system $\mathcal{P}$ we associate its reachability relation $R^{\mathcal{P}}$ as before.

For any $\epsilon > 0$, the ϵ-perturbed system $\mathcal{P}_\epsilon$ is described by the differential inclusion $d(\dot{\mathbf{x}}, \mathbf{f}(\mathbf{x})) < \epsilon$. This non-deterministic system can be seen as $\mathcal{P}$ submitted to a noise of magnitude ϵ. We denote reachability in the system $\mathcal{P}_\epsilon$ by $R^{\mathcal{P}}_{\epsilon}$. The limit reachability relation $R^{\mathcal{P}}_{\omega}$ is introduced as before.

▶ **Theorem 54** (Perturbed reachability is co-r.e.)**.** *Consider a continuous-time dynamical system, with $\mathbf{f}$ locally Lipschitz, computable, whose domain is a computable compact, then, for all computable $\mathbf{x}$, $cls(R^{\mathcal{P}}_{\omega}(\mathbf{x})) \subseteq \mathbb{R}^d$ is a co-c.e. closed subset.*

Proof. Its proof can be considered as the main technical result established in [26]. An alternative proof is similar to Theorems 41 and 23: adapt the construction of the involved graph G_m to cover the flow of the trajectory. With our hypotheses, the solutions are defined over all $\mathbb{R}$. It is proved in [16] that Lipschitz (and even effectively locally Lipschitz) homogeneous computable ODEs have computable solutions over their maximal domain. ◀

[4] A non-total solution must necessarily leave any compact, see e.g. [17], so X is compact is not a restriction.

▶ **Corollary 55** (Robust ⇒ decidable). *Assume the hypotheses of Theorem 54. If $R^{\mathcal{P}}$ is robust then for all computable* $\mathbf{x}$*, $cls(R^{\mathcal{P}}(\mathbf{x})) \subseteq \mathbb{R}^d$ is computable.*

7 Other perturbations

Inspired by analogue computations [8], when time has been related to the length of trajectories, we can also consider time or length-perturbations.

Time-perturbation. We can start by considering time-perturbed TM . The idea is that given $n > 0$, the n-perturbed version of $\mathcal{M}$ is unable to remain correct after a time n. Given $n > 0$, the n-perturbed version of $\mathcal{M}$, is defined exactly likewise, except after a time greater than n, its internal state q can change in a non-deterministic manner. The associated language is $L^n(\mathcal{M})$. From definitions: $L(\mathcal{M}) \subseteq L^\omega(\mathcal{M}) \subseteq \cdots \subseteq L^2(\mathcal{M}) \subseteq L^1(\mathcal{M})$.

▶ **Theorem 56** (Length robust ⇒ decidable). *$L^\omega(\mathcal{M})$ is in the class Π_1. Consequently, whenever $L^\omega(\mathcal{M}) = L(\mathcal{M})$, $L(\mathcal{M})$ is decidable.*

▶ **Theorem 57.** *When M always stops, $L^\omega(\mathcal{M}) = L(\mathcal{M})$.*

▶ **Definition 58** (Level of robustness n given by t). *Given $t : \mathbb{N} \to \mathbb{N}$, we write $L^{\{t\}}(\mathcal{M})$ for the set of words accepted by $\mathcal{M}$ with time perturbation t: $L^{\{t\}}(\mathcal{M}) = \{w |\ w \in L^{t(\ell(w))}(\mathcal{M})\}$.*

A robust dynamical system is necessarily t-robust for some function t, according to the next definition:

▶ **Definition 59** (t-robust to time language). *We say that a robust language is t-robust to time, when $L = L(\mathcal{M}) = L^{\{t\}}(\mathcal{M})$.*

▶ **Theorem 60** (Polynomially robust to time ⇔ PTIME). *A language L is in* PTIME *iff for some $\mathcal{M}$ and some polynomial p, $L = L(\mathcal{M}) = L^{\{p\}}(\mathcal{M})$.*

Furthermore, any PTIME *language is reducible to PAM's reachability: $R^{\mathcal{P}} = L^{\{p\}}(\mathcal{P})$ for some polynomial p.*

Length-perturbation. As we said, inspired by analogue computations [8], we can also consider length perturbations: Fix a distance $\delta(\cdot,\cdot)$ over the domain X. A finite trajectory of a discrete-time dynamical system $\mathcal{P}$ is a finite sequence $(x_t)_{t\in 0\ldots T}$ such that $x_{t+1} = f(x_t)$ for all $0 \leq t < T$. Its associated *length* is defined as $\mathcal{L} = \sum_{i=0}^{T-1} \delta(x_i, x_{i+1})$. We consider a *length-perturbed* discrete-time dynamical system: given $L > 0$, the L-perturbed version of the system is unable to remain correct after a length L. We define $R^{\mathcal{P},L}(\mathbf{x},\mathbf{y})$ as there exists a finite trajectory of $\mathcal{P}$ from $\mathbf{x}$ to $\mathbf{y}$ of length $\mathcal{L} \leq L$. When considering TMs as dynamical systems, $\delta(\cdot,\cdot)$ is a distance over configurations of TMs. Word w is said to be accepted in length d if the trajectory starting from $C_0[w]$ to the accepting configuration has length $\leq d$.

▶ **Definition 61.** *Distance $\delta(C, C')$ is called* time-metric *iff whenever C' is the configuration following configuration C, we have $\delta(C, C') \leq p(\ell(C))$, and $\delta(C, C') \geq \frac{1}{p(\ell(C))}$ for some polynomial p.*

Write $\mathcal{L}(M, t)$ for the set of words accepted by M in length less than t.

▶ **Definition 62** (Tolerating some level of robustness L given by f). *Given $f : \mathbb{N} \to \mathbb{N}$, we write $L^{(f)}(\mathcal{M})$ for $L^{(f)}(\mathcal{M}) = \{w |\ w \in \mathcal{L}(\mathcal{M}, f(\ell(w))\}$.*

▶ **Theorem 63** (Length robust for some time-metric distance ⇔ PTIME)**.** *Assume $\delta(\cdot,\cdot)$ is time metric. Then, a language L is in $PTIME$ iff for some TM $\mathcal{M}$ and some polynomial $p(n)$, $L = L(\mathcal{M}) = L^{(f)}(\mathcal{M})$.*

One way to obtain a distance $\delta(C, C')$ is to take the Euclidean distance between $\Upsilon(C)$ and $\Upsilon(C')$ for $\gamma = \gamma_{[0,1]}$, where $\gamma_{[0,1]}$ and Υ are the functions considered in Section 3. The obtained distance is time metric. Given $f : \mathbb{N} \to \mathbb{N}$, we write $R^{\mathcal{P},(f)}$ for the set of words accepted by $\mathcal{M}$ with length perturbation f: $R^{\mathcal{M},(f)} = \{w|\ w \in R^{\mathcal{M},f(\ell(w)}\}$.

▶ **Theorem 64** (Polynomially length robust ⇔ PTIME)**.** *Assume distance d is time metric and $R^{\mathcal{P}} = R^{\mathcal{P},(p)}$ for some polynomial p. Then $R^{\mathcal{P}} \in$ PTIME.*

8 Conclusion and future work

In this article, we have proposed a unified theory explaining in a uniform framework various statements relating robustness, defined as being non-sensitive to infinitesimal perturbations, to decidability. Most of the statements in the spirit of the *"robustness conjecture"* have been established using arguments from computability over the rationals or the reals, playing with variations on the statement that a semi-computable and co-semi-computable set is decidable.

More importantly, while existing statements of this form were only at the level of decidability, we showed that it is possible to also talk about complexity: robustness to polynomial perturbations on precision corresponds to PSPACE, robustness to polynomial perturbations on time or length corresponds to PTIME.

We also related the approach of [1] to the concept of δ-decidability of [15], as well as the drawability of the associated dynamics.

Notice that the proposed approach can also cover the so-called hybrid systems without difficulties. Various models have been considered in the literature for such systems, but one common point is that they all correspond to continuous-time dynamical systems, where the dynamics might be discontinuous, so not computable. In a very general view, a *hybrid system* $\mathcal{P}$ is given by a set $X \subseteq \mathbb{R}^d$, a semi-group T and a flow function $\phi : X \times T \to X$ satisfying $\phi(\mathbf{x}, 0) = \mathbf{x}$ and $\phi(\phi(\mathbf{x}, t), t') = \phi(\mathbf{x}, t + t')$. Previous proofs use the fact that reachability $R^{\mathcal{P}}$ is c.e. and perturbed reachability is co-c.e. The former is usually obvious in any of the considered models, as we expect to be able to simulate the model. The latter is usually less trivial. If we look at our proof methods, we only need to construct some computable abstraction satisfying Claims 1 and 2. One key remark is that we need these properties not about the function $\mathbf{f}$ but its graph. Assuming a function such that the closure of its graph is computable, is more general than assuming computability. For example, the characteristic function $\chi_{[0,\infty)}$ is not computable, as it is not continuous. But its graph, as well as its closure, is easy to draw: see discussions e.g. in [13]. In particular, this allowed us to talk about discontinuous functions in the current article.

Regarding analogue models of computation, a variation on our concept of robustness has already been used to provide a characterisation of PSPACE for discrete-time ordinary differential equations in [3].

We believe that the theory developed here might be used to prove formally that space complexity corresponds to precision for continuous-time models of computation, over some compact domains, providing a more natural measure than the conditions considered in [7].

References

1 Eugene Asarin and Ahmed Bouajjani. Perturbed Turing machines and hybrid systems. In *Proceedings of the 16th Annual IEEE Symposium on Logic in Computer Science (LICS-01)*, pages 269–278, Los Alamitos, CA, June 16–19 2001. IEEE Computer Society Press.

2 Eugene Asarin, Oded Maler, and Amir Pnueli. Reachability analysis of dynamical systems having piecewise-constant derivatives. *Theoretical Computer Science*, 138(1):35–65, February 1995.

3 Manon Blanc and Olivier Bournez. A characterisation of functions computable in polynomial time and space over the reals with discrete ordinary differential equations: Simulation of Turing machines with analytic discrete odes. In *Mathematical Foundations of Computer Science (MFCS'2023)*, 2023.

4 Vincent Blondel and John Tsitsiklis. A survey of computational complexity results in systems and control. *Automatica*, 36(9):1249–1274, 2000.

5 Olivier Bournez, Johanne Cohen, and Valentin Dardilhac. On the δ-decidability of decision problems for neural network questions. In *Computability, Continuity, Constructivity - from Logic to Algorithms CCC'23*, 2023.

6 Olivier Bournez, Felipe Cucker, Paulin Jacobé de Naurois, and Jean-Yves Marion. Computability over an arbitrary structure. sequential and parallel polynomial time. In Andrew D. Gordon, editor, *Foundations of Software Science and Computational Structures, 6th International Conference (FOSSACS'2003)*, volume 2620 of *Lecture Notes in Computer Science*, pages 185–199, Warsaw, 2003. Springer.

7 Olivier Bournez, Riccardo Gozzi, Daniel S Graça, and Amaury Pouly. A continuous characterization of PSPACE using polynomial ordinary differential equations. *Journal of Complexity*, 77:101755, August 2023. URL: `https://www.sciencedirect.com/science/article/pii/S0885064X23000249?dgcid)=author`.

8 Olivier Bournez, Daniel S. Graça, and Amaury Pouly. Polynomial Time corresponds to Solutions of Polynomial Ordinary Differential Equations of Polynomial Length. *Journal of the ACM*, 64(6):38:1–38:76, 2017. `doi:10.1145/3127496`.

9 Olivier Bournez, Daniel S. Graça, and Emmanuel Hainry. Robust computations with dynamical systems. In *Mathematical Foundations of Computer Science, MFCS'2010*, volume 6281 of *Lecture Notes in Computer Science*, pages 198–208. Springer, 2010. `doi:10.1007/978-3-642-15155-2_19`.

10 Olivier Bournez and Emmanuel Hainry. An analog characterization of elementary computable functions over the real numbers. In Josep Díaz, Juhani Karhumäki, Arto Lepistö, and Donald Sannella, editors, *International Colloquium on Automata, Languages and Programming (ICALP 2004)*, volume 3142 of *Lecture Notes in Computer Science*, pages 269–280, 2004.

11 Olivier Bournez and Amaury Pouly. A survey on analog models of computation. In *Handbook of Computability and Complexity in Analysis*, pages 173–226. Springer, 2021.

12 Vasco Brattka, Peter Hertling, and Klaus Weihrauch. A tutorial on computable analysis. In *New computational paradigms*, pages 425–491. Springer, 2008.

13 Mark Braverman. Computational complexity of Euclidean sets: Hyperbolic Julia sets are poly-time computable. Master's thesis, University of Toronto, 2004.

14 Martin Fränzle. Analysis of hybrid systems: An ounce of realism can save an infinity of states. In Jörg Flum and Mario Rodríguez-Artalejo, editors, *Computer Science Logic, 13th International Workshop, CSL '99, 8th Annual Conference of the EACSL, Madrid, Spain, September 20-25, 1999, Proceedings*, volume 1683 of *Lecture Notes in Computer Science*, pages 126–140. Springer, 1999.

15 Sicun Gao, Jeremy Avigad, and Edmund M Clarke. Delta-decidability over the reals. In *Logic in Computer Science (LICS), 2012 27th Annual IEEE Symposium on*, pages 305–314. IEEE, 2012.

16 Daniel S. Graça, N. Zhong, and J. Buescu. Computability, noncomputability and undecidability of maximal intervals of IVPs. *Transactions of the American Mathematical Society*, 2006. To appear.

17 Philip Hartman. *Ordinary Differential Equations*. John Wiley and Sons, 1964.

18 Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. What's decidable about hybrid automata? *Journal of Computer and System Sciences*, 57(1):94–124, August 1998.

19 Thomas A. Henzinger and Jean-François Raskin. Robust undecidability of timed and hybrid systems. In Nancy A. Lynch and Bruce H. Krogh, editors, *Hybrid Systems: Computation and Control, Third International Workshop, HSCC 2000, Pittsburgh, PA, USA, March 23-25, 2000, Proceedings*, volume 1790 of *Lecture Notes in Computer Science*, pages 145–159. Springer, 2000.

20 N. Immerman. Nondeterministic space is closed under complementation. In *Structure in Complexity Theory Conference, 1988. Proceedings., Third Annual*, pages 112–115. IEEE, 1988.

21 Ker-I Ko. *Complexity Theory of Real Functions*. Progress in Theoretical Computer Science. Birkhaüser, Boston, 1991.

22 Pascal Koiran, Michel Cosnard, and Max Garzon. Computability with low-dimensional dynamical systems. *Theoretical Computer Science*, 132(1-2):113–128, September 1994.

23 Cristopher Moore. Generalized shifts: unpredictability and undecidability in dynamical systems. *Nonlinearity*, 4(3):199–230, 1991.

24 Eike Neumann. Decision problems for linear recurrences involving arbitrary real numbers. *Logical Methods in Computer Science*, 17, 2021.

25 Eike Neumann. On the complexity of robust eventual inequality testing for C-finite functions. In *International Conference on Reachability Problems*, pages 98–112. Springer, 2023.

26 A. Puri, V. Borkar, and P. Varaiya. Epsilon-approximation of differential inclusions. In *Proceedings of the 34th IEEE Conference on Decision and Control*, pages 2892–2897, 1995.

27 Anuj Puri. Dynamical properties of timed automata. *Discrete Event Dynamic Systems*, 10:87–113, 2000.

28 Stefan Ratschan. Deciding predicate logical theories of real-valued functions. In *Symposium on Mathematical Foundations of Computer Science (MFCS'2023)*, 2023.

29 Claude E. Shannon. Mathematical theory of the differential analyser. *Journal of Mathematics and Physics MIT*, 20:337–354, 1941.

30 Michael Sipser. *Introduction to the Theory of Computation*. PWS Publishing Company, 1997.

31 R. Szelepcsényi. The method of forced enumeration for nondeterministic automata. *Acta informatica*, 26(3):279–284, November 1988. doi:10.1007/BF00299636.

32 Klaus Weihrauch. *Computable Analysis: an Introduction*. Springer, 2000.

From Local to Global Optimality in Concurrent Parity Games

Benjamin Bordais
Université Paris-Saclay, CNRS, ENS Paris-Saclay, LMF, 91190 Gif-sur-Yvette, France
TU Dortmund University, Germany
Center for Trustworthy Data Science and Security, University Alliance Ruhr, Germany

Patricia Bouyer
Université Paris-Saclay, CNRS, ENS Paris-Saclay, LMF, 91190 Gif-sur-Yvette, France

Stéphane Le Roux
Université Paris-Saclay, CNRS, ENS Paris-Saclay, LMF, 91190 Gif-sur-Yvette, France

Abstract

We study two-player games on finite graphs. Turn-based games have many nice properties, but concurrent games are harder to tame: e.g. turn-based stochastic parity games have positional optimal strategies, whereas even basic concurrent reachability games may fail to have optimal strategies. We study concurrent stochastic parity games, and identify a local structural condition that, when satisfied at each state, guarantees existence of positional optimal strategies for both players.

2012 ACM Subject Classification Theory of computation → Solution concepts in game theory

Keywords and phrases Game forms, stochastic games, parity games, Blackwell/Martin values

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.18

Related Version *Full Version*: `https://arxiv.org/pdf/2311.14373.pdf`

Funding This paper was partially funded by ANR-22-CE48-0012 (BisoUS).

1 Introduction

Two-player games played on finite graphs have been a helpful model in areas of computer science. In such games, states/vertices of the graph are colored; the actions of the players induce an infinite path in the graph, thus inducing an infinite sequence of colors. Who wins depends on the color sequence. These games can be turn-based, i.e. at each state a unique player chooses an outgoing edge leading to a probability distribution over successor states, or concurrent, i.e. at each state, the combination of one action per player determines the probability distribution over successor states. In such stochastic settings, Player A wants to maximize her probability to win, and Player B to minimize the very same probability.

We study the above games in the case of parity objective: colors are natural numbers, and a sequence is winning for Player A iff the maximal color seen infinitely often is even. This objective has been well-studied in connection with model-checking. The turn-based version of these games has nice properties involving deterministic positional strategies, where " positional" means that the played action depends only on the current state: with only deterministic probability distributions over successor states, either of the players has a winning such strategy [15]; with arbitrary probability distributions over successor states, both players have optimal such strategies [16, 7]. Note that in concurrent parity games, positional optimal strategies for distinct starting states yield one positional strategy optimal for all states uniformly. So we omit the word " uniform" in this paper.

The above properties and others break in basic concurrent games except in safety games, as recorded in Table 1, where safety and reachability are special cases of (co-)Büchi, and parity subsumes (co-)Büchi. Büchi games may not have optimal strategies but when they

Table 1 Memory status of (almost-)optimal strategies in concurrent games. The second column says for each objective whether optimal strategies always exist or not; the third column gives the nature (positional or infinite memory) of optimal strategies when such optimal strategies exist; the fourth column gives the nature of ε-optimal strategies; the last column gives the nature of subgame-optimal strategies, a refinement of optimal strategies. This shows the diversity of memory requirements for the various objectives listed on the left.

Objectives	$\exists$ opt strat?	opt	ε-opt	SubG-opt
Safety	always [9]	pos. [9]	pos. [9]	pos. [9]
Reach.	not always [11]	pos. [2]	pos. [10]	pos. [2]
Büchi	not always	pos. [3]	∞ [8]	pos. [3]
Co-B.	not always	∞ [3]	pos. [6]	pos. [5]
Parity	not always	∞ [8]	∞ [8]	∞ [8]

do, they have positional ones; co-Büchi games may not have optimal strategies, and they may have only infinite-memory ones; and the other way around for ε-optimal strategies, hence incomparable difficulty between Büchi and co-Büchi. This shows that concurrent parity is strictly harder than (co-)Büchi, since parity games may have only infinite-memory (ε-)optimal strategies. (See also the column on subgame optimal strategies.)

Nevertheless, concurrent games' poor behavior in general should not deter us from studying them, as many complex systems are inherently concurrent. See [12] for further arguments. Continuing a recent line of research [1, 2, 3] we seek local structural good behaviors that scale up to the whole game.

The following key property still holds in parity concurrent games, by determinacy of Blackwell games [13]: each state has a value $u \in [0, 1]$, i.e. for all $\varepsilon > 0$, Player A has an ε-optimal strategy for plays starting in the state, i.e. guaranteeing winning probability at least $u - \varepsilon$ to win, regardless of Player B's strategy; and likewise Player B can guarantee that Player A wins with probability at most $u + \varepsilon$. Here we are interested in strategies that are optimal, i.e. that realize exactly the value u, and that are positional. They do not exist in general (see [2, Figure 2]), but our main result is a transfer property from local to global positional optimality, a weak version being the next theorem; the terminology that it uses is explained afterwards.

▶ **Theorem 1.** *If at every state the induced game form is positionally optimizable, both players have positional optimal strategies in the game.*

A game form is a map from pairs of actions to probability distributions of successor states, see Fig. 1 to the left, or [1, p. 5] for more examples. So each state induces a game form. Given a game form $\mathcal{F}$, an $\mathcal{F}$-game has a unique non-trivial state which induces $\mathcal{F}$ and the other player states are of two kinds: first kind, they loop back to the non-trivial state; second kind, they are not colored, but they have an explicit value in $[0, 1]$, and the game stops there. Moreover Player A prefers higher explicit values. See Fig. 1 or Fig. 5. Finally, $\mathcal{F}$ is called *positionally optimizable* if both players have positional optimal strategies in all $\mathcal{F}$-games, which we show to be decidable.

The proof of Theorem 1 involves an extraction of an *environment function*, which gives for each state a summary of some local information sufficient to globally play optimally in the game; the summary of some state q is a (small) $\mathcal{F}$-game, where $\mathcal{F}$ is the original game form of state q. The extraction of this environment function is made by analyzing in an appropriate order the immediate neighbors of the states, and propagating gathered information further away.

As a corollary of Thm. 1, we among other things recover the known result that positional strategies are sufficient to play optimally in turn-based stochastic games [16, 7], since turn-based game forms are positionally optimizable. Let us rephrase and strengthen Thm. 1: if the induced game forms of a game behave well in all (small) $\mathcal{F}$-games, they behave well in all larger games; conversely, by definition an arena inducing a poorly-behaved game form yields at least one poorly-behaved small game. Let us highlight two benefits of Thm. 1: First, designing a game using only well-behaved components, i.e. game forms, ensures the existence of positional optimal strategies, which was our main purpose. Second, it provides a local, structural criterion for games to have positional optimal strategies.

An extended version of this paper with all the technical details can be found in [4].

2 Preliminaries and game forms

If Q is a non-empty set, we denote by Q^* (resp. Q^+, Q^ω) the set of finite (resp. non-empty finite, infinite) sequences of Q. A *(discrete probability) distribution* over a set Q is a function $\mu : Q \to [0,1]$ with a finite support $\mathsf{Sp}(\mu) := \{x \in Q \mid \mu(x) > 0\}$, such that $\sum_{x\in\mathsf{Sp}(\mu)} \mu(x) = 1$. The distribution μ is *deterministic* if its support is a singleton. For all $S \subseteq Q$, we let $\mu[S] := \sum_{x\in S} \mu(x)$. The set of distributions over the set Q is denoted $\mathcal{D}(Q)$.

For all $i \leq j \in \mathbb{N}$, we write $[\![i, j]\!]$ for the set $\{k \in \mathbb{N} \mid i \leq k \leq j\}$. This set is typed in the sense that these are seen as integers and not reals numbers, so that we will be able to consider the disjoint union of $[0,1]$ with such a set of integers which may include 0 or 1. For all finite sets $S \subseteq \mathbb{N}$, we let $\mathsf{Even}(S)$ (resp. $\mathsf{Odd}(S)$) be the smallest even (resp. odd) integer that is greater than or equal to all elements in S.

We recall the definition of game forms and of games in normal forms.

▶ **Definition 2** (Game form and game in normal form). *Let* O *be a non-empty set of outcomes. A* game form *(*GF *for short) on* D *is a tuple* $\mathcal{F} = \langle \mathsf{Act_A}, \mathsf{Act_B}, \mathsf{O}, \varrho\rangle$ *where* $\mathsf{Act_A}$ *(resp.* $\mathsf{Act_B}$*) is the non-empty finite set of actions available to Player* A *(resp.* B*) and* $\varrho : \mathsf{Act_A} \times \mathsf{Act_B} \to \mathcal{D}(\mathsf{O})$ *maps each pair of actions to a distribution over the outcomes. We denote by* $\mathsf{Form}(\mathsf{O})$ *the set of game forms on* O*. A Player-*A *(resp. Player-*B*) game form* $\mathcal{F}$ *is such that* $|\mathsf{Act_B}| = 1$ *(resp.* $|\mathsf{Act_A}| = 1$*). A game form is trivial if* $|\mathsf{Act_B}| = 1$ *and* $|\mathsf{Act_A}| = 1$*.*

When $\mathsf{O} = [0,1]$*, we say that* $\mathcal{F}$ *is a* game in normal form*. For a valuation* $v : \mathsf{D} \to [0,1]$*,* $\langle \mathcal{F}, v\rangle$ *denotes the game in normal form* $\langle \mathsf{Act_A}, \mathsf{Act_B}, [0,1], v \circ \varrho\rangle$ *induced from* $\mathcal{F}$ *by* v*.*

An example of a game form is depicted on the left of Fig. 1 (page 7) where the actions available to Player A are the rows and the actions available to Player B are the columns. Strategies available to Player-A (resp. B) are then the probability distributions over their respective sets of actions. In a game in normal form, Player A tries to maximize the outcome, whereas Player B tries to minimize it.

▶ **Definition 3** (Outcome and value of a game in normal form). *Let* $\mathcal{F} = \langle \mathsf{Act_A}, \mathsf{Act_B}, [0,1], \varrho\rangle$ *be a game in normal form. For* $\mathsf{C} \in \{\mathsf{A}, \mathsf{B}\}$*, the set of strategies of Player* C *is* $\mathcal{D}(\mathsf{Act_C})$ *and is thereafter denoted* $\Sigma_\mathsf{C}(\mathcal{F})$*. For a pair of strategies* $(\sigma_\mathsf{A}, \sigma_\mathsf{B}) \in \Sigma_\mathsf{A}(\mathcal{F}) \times \Sigma_\mathsf{B}(\mathcal{F})$*, their outcome in* $\mathcal{F}$ *is* $\mathsf{out}_\mathcal{F}(\sigma_\mathsf{A}, \sigma_\mathsf{B}) := \sum_{a\in\mathsf{Act_A}} \sum_{b\in\mathsf{Act_B}} \sigma_\mathsf{A}(a) \cdot \sigma_\mathsf{B}(b) \cdot \varrho(a, b) \in [0,1]$*.*

Let $\sigma_\mathsf{A} \in \Sigma_\mathsf{A}(\mathcal{F})$ *be a Player-*A *strategy. Its* value *is* $\mathsf{val}_\mathcal{F}(\sigma_\mathsf{A}) := \inf_{\sigma_\mathsf{B}\in\Sigma_\mathsf{B}(\mathcal{F})} \mathsf{out}_\mathcal{F}(\sigma_\mathsf{A}, \sigma_\mathsf{B})$*; and dually for Player* B*. When* $\sup_{\sigma_\mathsf{A}\in\Sigma_\mathsf{A}(\mathcal{F})} \mathsf{val}_\mathcal{F}(\sigma_\mathsf{A}) = \inf_{\sigma_\mathsf{B}\in\Sigma_\mathsf{B}(\mathcal{F})} \mathsf{val}_\mathcal{F}(\sigma_\mathsf{B})$*, it defines the* value *of the game* $\mathcal{F}$*, denoted* $\mathsf{val}_\mathcal{F}$*. If* $\mathsf{val}_\mathcal{F}(\sigma_\mathsf{A}) = \mathsf{val}_\mathcal{F}$*, the strategy* σ_A *is said to be* optimal *for Player* A*. This is defined analogously for Player* B*.*

Since the sets of actions are finite, Von Neumann's minimax theorem [14] ensures the existence of a value and of optimal strategies for both players in any game in normal form.

In the following, strategies in game forms will be called GF-strategies in order not to confuse them with strategies in concurrent games (on graphs).

3 Concurrent games

3.1 Concurrent arenas and games

▶ **Definition 4** (Finite stochastic concurrent arena). *A finite* concurrent arena $\mathcal{C}$ *is a tuple* $\langle Q, \mathsf{F}\rangle$ *where* Q *is a non-empty finite set of states and* $\mathsf{F} : Q \to \mathsf{Form}(\mathsf{D})$ *maps each state to its induced game form, which describes the interaction of the players at this state.*

In the following, the arena $\mathcal{C}$ will refer to a tuple $\langle Q, \mathsf{F}\rangle$, unless otherwise stated. In this paper, we focus on (max) parity objectives: given a coloring function, the goal of Player A is that the maximum of the colors visited infinitely often is even.

▶ **Definition 5** (Parity game). *Let* $\mathsf{col} : Q \to \mathbb{N}$ *be a coloring function. It induces the* parity objective $W(\mathsf{col}) \subseteq Q^\omega$ *defined by* $W(\mathsf{col}) := \{\rho \in Q^\omega \mid \max(\mathsf{col}(\rho)_\infty)$ *is even*$\}$ *where* $\mathsf{col}(\rho)_\infty := \{k \in \mathbb{N} \mid \forall i \in \mathbb{N},\ \exists j \geq i,\ \mathsf{col}(\rho)_j = k\} \neq \emptyset$ *denotes the set of colors seen infinitely often along* ρ*. A* parity game $\mathcal{G} = \langle \mathcal{C}, \mathsf{col}\rangle$ *is a pair formed of a concurrent arena* $\mathcal{C}$ *and a coloring function* $\mathsf{col} : Q \to \mathbb{N}$*.*

We fix a parity game $\mathcal{G} = \langle \mathcal{C}, \mathsf{col}\rangle$ for the rest of this section. In such a game, strategies map the history of the game (i.e. the finite sequence of states visited so far) to a GF-strategy in the game form corresponding to the current state of the game.

▶ **Definition 6** (Strategies). *A* strategy *for Player* A *is a function* $\mathsf{s}_\mathsf{A} : \bigcup_{q\in Q}(Q^* \cdot q \to \Sigma_\mathsf{A}(\mathsf{F}(q)))$*. It is* positional *if for all* $q \in Q$*, there is a* GF*-strategy* $\sigma^q_\mathsf{A} \in \Sigma_\mathsf{A}(\mathsf{F}(q))$ *such that, for all* $\pi = \rho \cdot q \in Q^+$*:* $\mathsf{s}_\mathsf{A}(\pi) = \sigma^q_A$*. In that case, the strategy* s_A *is said to be* defined *by* $(\sigma^q_\mathsf{A})_{q\in Q}$*. We denote by* $\mathsf{S}^\mathsf{A}_\mathcal{C}$ *and* $\mathsf{PS}^\mathsf{A}_\mathcal{C}$ *the set of all strategies and positional strategies respectively in arena* $\mathcal{C}$ *for Player* A*. A strategy* s_A *is* deterministic *if for all* $\rho \in Q^+$*,* $\mathsf{s}_\mathsf{A}(\rho)$ *is deterministic. The definitions are analogous for Player* B*.*

Unlike deterministic games with deterministic strategies, the outcome of a game, given two strategies (one for each Player), is not a single play but rather a distribution over plays. To formalize this, we first define the probability to go from a state q to a state q' given two GF-strategies in a game form $\mathsf{F}(q)$.

▶ **Definition 7** (Probability transition). *Given states* $q, q' \in Q$ *and two strategies* $(\sigma_\mathsf{A}, \sigma_\mathsf{B}) \in \Sigma_\mathsf{A}(\mathsf{F}(q)) \times \Sigma_\mathsf{B}(\mathsf{F}(q))$ *the probability to go from* q *to* q' *if the players play, in* q*,* σ_A *and* σ_B*, is:* $\mathbb{P}^{\sigma_\mathsf{A},\sigma_\mathsf{B}}(q, q') := \mathsf{out}_{\langle \mathsf{F}(q), \mathbb{1}_{q'}\rangle}(\sigma_\mathsf{A}, \sigma_\mathsf{B})$*, where* $\mathbb{1}_{q'} : Q \to [0, 1]$ *is the indicator function such that, for all* $q'' \in Q$*, we have* $\mathbb{1}_{q'}(q'') = 1$ *if and only if* $q'' = q'$*.*

We now define the probability of occurrence of finite paths, and consequently of any Borel set, given a strategy per player.

▶ **Definition 8** (Probability distribution given two strategies). *Let* $(\mathsf{s}_\mathsf{A}, \mathsf{s}_\mathsf{B}) \in \mathsf{S}^\mathsf{A}_\mathcal{C} \times \mathsf{S}^\mathsf{B}_\mathcal{C}$ *be two arbitrary strategies. We denote by* $\mathbb{P}^{\mathsf{s}_\mathsf{A},\mathsf{s}_\mathsf{B}} : Q^+ \to \mathcal{D}(Q)$ *the function giving the probability distribution over the next state of the arena given the sequence of states already seen. That is, for all finite path* $\pi = \pi_0 \dots \pi_n \in Q^+$ *and* $q \in Q$*, we have:* $\mathbb{P}^{\mathsf{s}_\mathsf{A},\mathsf{s}_\mathsf{B}}(\pi)[q] := \mathbb{P}^{\mathsf{s}_\mathsf{A}(\pi),\mathsf{s}_\mathsf{B}(\pi)}(\pi_n, q)$*. The probability of a finite path* $\pi = \pi_0 \cdots \pi_n \in Q^+$ *from a state* $q_0 \in Q$ *with the pair of strategies* $(\mathsf{s}_\mathsf{A}, \mathsf{s}_\mathsf{B})$ *is then equal to* $\mathbb{P}^{\mathcal{C},q_0}_{\mathsf{s}_\mathsf{A},\mathsf{s}_\mathsf{B}}(\pi) := \Pi_{i=0}^{n-1}\mathbb{P}^{\mathsf{s}_\mathsf{A},\mathsf{s}_\mathsf{B}}(\pi_{\leq i})[\pi_{i+1}]$ *if* $\pi_0 = q_0$ *and* 0 *otherwise. The probability of a cylinder set* $\mathsf{Cyl}(\pi) := \{\pi \cdot \rho \mid \rho \in Q^\omega\}$ *is* $\mathbb{P}^{\mathcal{C},q_0}_{\mathsf{s}_\mathsf{A},\mathsf{s}_\mathsf{B}}[\mathsf{Cyl}(\pi)] := \mathbb{P}^{\mathsf{s}_\mathsf{A},\mathsf{s}_\mathsf{B}}(\pi)$ *for any finite path* $\pi \in Q^*$*. This induces the probability measure over Borel sets in the usual way. We denote by* $\mathbb{P}^{\mathcal{C},q_0}_{\mathsf{s}_\mathsf{A},\mathsf{s}_\mathsf{B}}$ *this probability measure, mapping each Borel set to a value in* $[0, 1]$*.*

The values of strategies and of the game follow.

▶ **Definition 9** (Value of strategies and of the game). *Let* $\mathsf{s}_\mathsf{A} \in \mathsf{S}^\mathsf{A}_\mathcal{C}$ *be a Player-*A *strategy. The vector* $\chi_\mathcal{G}[\mathsf{s}_\mathsf{A}] : Q \to [0,1]$ *giving the value of the strategy* s_A *is such that, for all* $q_0 \in Q$*, we have* $\chi_\mathcal{G}[\mathsf{s}_\mathsf{A}](q_0) := \inf_{\mathsf{s}_\mathsf{B} \in \mathsf{S}^\mathsf{B}_\mathcal{C}} \mathbb{P}^{\mathcal{C},q_0}_{\mathsf{s}_\mathsf{A},\mathsf{s}_\mathsf{B}}[W(\mathsf{col})]$*. The vector* $\chi_\mathcal{G}[\mathsf{A}] : Q \to [0,1]$ *giving the value for Player* A *is such that, for all* $q_0 \in Q$*, we have* $\chi_\mathcal{G}[\mathsf{A}](q_0) := \sup_{\mathsf{s}_\mathsf{A} \in \mathsf{S}^\mathsf{A}_\mathcal{C}} \chi_\mathcal{G}[\mathsf{s}_\mathsf{A}](q_0)$*. The vector* $\chi_\mathcal{G}[\mathsf{B}] : Q \to [0,1]$ *giving the value of the game for Player* B *is defined symmetrically.*

By Martin's result on the determinacy of Blackwell games [13]: $\chi_\mathcal{G}[\mathsf{A}] = \chi_\mathcal{G}[\mathsf{B}]$*, which defines the* value *of the game:* $\chi_\mathcal{G} := \chi_\mathcal{G}[\mathsf{A}] = \chi_\mathcal{G}[\mathsf{B}]$*. A Player-*$\mathsf{A}$ *strategy* s_A *such that* $\chi_\mathcal{G} := \chi_\mathcal{G}[\mathsf{s}_\mathsf{A}]$ *is* optimal *(and similarly for Player* B*).*

Note that optimal strategies may not exist in general; when they exist they can be arbitrarily complex; see the table page 2.

Finally, for convenience, we extend our formalism by considering stopping states with output values, i.e. states that, when visited, immediately stop the game and induce a specific value in $[0,1]$. The fact that the value of a stopping state q is set to be u is denoted $\mathsf{val}(q) \leftarrow u$. Stopping states can be encoded by simple gadgets in our formalism, they will be depicted as dashed states.

3.2 Markov chains and sufficient condition for optimality

In this subsection, we give a condition on positional strategies to be optimal in a parity game. First, we introduce the notions of Markov chain and bottom strongly connected component.

▶ **Definition 10** (Markov chain, BSCC). *A* Markov chain $\mathcal{M}$ *is a pair* $\mathcal{M} = \langle Q, \mathbb{P} \rangle$ *where* Q *is a finite set of states and* $\mathbb{P} : Q \to \mathcal{D}(Q)$*. A* bottom strongly connected component *(BSCC for short)* $H \subseteq Q$ *is a subset of states such that the underlying graph of* H *is strongly connected (w.r.t. edges given by positive probability transitions from states to states) and* H *cannot be exited: for all* $q \in H$ *and* $q' \in Q$*,* $\mathbb{P}(q)(q') > 0$ *implies* $q' \in H$*.*

Two positional strategies (one per player) in a concurrent arena not only induce a probability measure on infinite sequences of states, but also a Markov chain, whose graph is a subgraph of the arena. If we only fix a positional strategy for one of the players, we will consider the set of BSCCs that are compatible with that strategy in the following sense.

▶ **Definition 11** (Induced Markov chains and BSCCs compatible with a strategy). *Let* s *be a positional strategy for one of the players. For every positional and deterministic strategy* s' *for the other player, we denote by* $\mathcal{M}_{\mathsf{s},\mathsf{s}'} = \langle Q, \mathbb{P}^{\mathsf{s},\mathsf{s}'} \rangle$ *the Markov chain* induced *by* s *and* s'*, and by* H_s *the set of BSCCs* compatible *with* s*, i.e. the BSCCs of some Markov chain* $\mathcal{M}_{\mathsf{s},\mathsf{s}'}$*. A BSCC* $H \in H_\mathsf{s}$ *is* even-colored *if* $\max \mathsf{col}[H]$ *is even. Otherwise, it is* odd-colored*.*

We define three properties relating positional strategies and valuations of the states. A Player-A strategy dominates a valuation v if, regardless of what the other player plays, the value of every state is at most the expected value of its successors. Further, a Player-A strategy parity dominates the valuation v if in addition all the BSCCs compatible with it are even-colored. Finally, a Player-A strategy guarantees the valuation v if, from every state, the value of the strategy is at least the value of the states w.r.t. v. In particular, if a strategy guarantees the valuation $\chi_\mathcal{G} : Q \to [0,1]$, then it is optimal (by definition).

▶ **Definition 12** ((Parity) Domination, Guarantees). *Let* $v : Q \to [0,1]$ *be a valuation of the states. Let* $\mathsf{s}_\mathsf{A} \in \mathsf{PS}^\mathsf{A}_\mathcal{C}$ *be a positional Player-*A *strategy. This strategy* s_A*:*

- dominates v *if for all* $q \in Q$*,* $v(q) \leq \mathsf{val}_{\langle \mathsf{F}(q), v \rangle}(\mathsf{s}_\mathsf{A}(q))$*;*

- parity dominates v *if it dominates* v *and all BSCCs* H *compatible with* s_A *s.t.* $\min v[H] > 0$*;*
- guarantees v *if for all* $q \in Q$, $v(q) \leq \chi_\mathcal{G}[\mathsf{s}_\mathsf{A}](q)$.

The definitions are symmetrical for a Player-B *positional strategy* $\mathsf{s}_\mathsf{B} \in \mathsf{PS}^\mathsf{B}_\mathcal{C}$.

As stated in Proposition 13 below, if a strategy parity dominates a valuation, then it also guarantees it.

▶ **Proposition 13.** *Let* $\mathsf{s}_\mathsf{A} \in \mathsf{PS}^\mathsf{A}_\mathcal{C}$ *be a positional Player-*A *strategy and* $v : Q \to [0,1]$ *be a valuation. If* s_A *dominates* v*, then for all BSCC* $H \in H_{\mathsf{s}_\mathsf{A}}$*, there is some* $v_H \in [0,1]$ *such that* $v[H] = \{v_H\}$*. If in addition* s_A *parity dominates* v*, it also guarantees* v*.*

In the remainder of this paper, we will be interested in showing that a strategy is optimal. We will do so by establishing that it parity dominates the valuation $\chi_\mathcal{G} : Q \to [0,1]$. The benefit of parity domination is that, compared to optimality, it specifies more explicitly how the strategy behaves in a game. It is for instance used in Proposition 28 where optimal Player-A strategies are obtained by gluing together pieces of strategies that parity dominates some valuations.

4 Local environment and local game

The goal of this section is to define small parity games with a single non-trivial local interaction, which will enlighten game forms that should be used in parity games if we require positional optimal strategies. We first consider what (parity) environments on a given set of outcomes are. Informally, these environments tell a game form how it should view its outcomes: either as stopping states, or as colored states. The formal definition is given below.

▶ **Definition 14** (Parity environment and its induced parity game)**.** *Let* O *be a non-empty finite set of outcomes. An* environment E *on* O *is a tuple* $E := \langle c, e, p\rangle$ *where* $c, e \in \mathbb{N}$ *with* $c \leq e$ *and* $p : \mathsf{O} \to \{q_\mathsf{init}\} \uplus [\![0, e]\!] \uplus [0,1]$ *maps each outcome to what will be states in small* $\mathcal{F}$*-games.*

The size *of* E *w.r.t. Player* A *(resp.* B*) is* $\mathsf{Sz}_\mathsf{A}(E) := \mathsf{Even}(e) - c$ *(resp.* $\mathsf{Sz}_\mathsf{B}(E) := \mathsf{Odd}(e) - c$*). We denote by* $\mathsf{Env}(\mathsf{O})$ *the set of all environments on* D*.*

We can then consider the games induced by such an environment (along with a game form). Informally, given a game form $\mathcal{F} \in \mathsf{Form}(\mathsf{O})$ and a parity environment $E = \langle c, e, p\rangle \in \mathsf{Env}(\mathsf{O})$, we consider the small parity arena $\mathcal{C}_Y$ induced by $Y := (\mathsf{O}, \mathcal{F}, E)$ defined as follows: there is a single central state q_init whose local interaction is given by $\mathcal{F}$. The outcomes of $\mathcal{F}$ lead in $\mathcal{C}_Y$ to states in $\{q_\mathsf{init}\} \uplus \{k_i \mid i \in [\![0, e]\!]\} \uplus [0,1]$, as prescribed by p. All states in $[0,1]$ are stopping states and all states in $\{k_i \mid i \in [\![0, e]\!]\}$ are trivial and loop back to q_init. The small parity game $\mathcal{G}_Y$ that we consider is then obtained from the arena $\mathcal{C}_Y$ by considering a coloring function col that maps q_init to c and every state k_i to i. These small games correspond to the $\mathcal{F}$-games in the introduction. This is formally defined below.

▶ **Definition 15** (Parity game induced by an environment)**.** *Consider a non-empty finite set of outcomes* O*, a game form* $\mathcal{F} \in \mathsf{Form}(\mathsf{O})$ *and an environment* $E = \langle c, e, p\rangle \in \mathsf{Env}(\mathsf{O})$*. Let* $Y := (\mathsf{O}, \mathcal{F}, E)$*. The local arena* $\mathcal{C}_Y = \langle Q, \mathsf{F}\rangle$ *induced by* Y *is such that:*

- $Q := \{q_\mathsf{init}\} \cup K_e \cup p_{[0,1]}$*, where* $K_e := \{k_i \mid i \in [\![0, e]\!]\}$ *and* $p_{[0,1]} = p[\mathsf{O}] \cap [0,1]$*;*
- *for all* $x \in p_{[0,1]}$*, we set the value of the stopping state* x *to be* x *itself:* $\mathsf{val}(u) \leftarrow u$*;*
- $\mathsf{F}(q_\mathsf{init}) := \mathcal{F}$ *(up to identifying integers in* $[\![0, e]\!]$ *and states in* $\{k_i \mid i \in [\![0, e]\!]\}$*), and for all* $i \in [\![0, e]\!]$*, we set* $\mathsf{F}(k_i)$ *to be a trivial game form with* q_init *as only possible outcome.*

$$\mathcal{F} := \begin{bmatrix} y & x \\ x & y \end{bmatrix}$$

- $E := \langle 2, 4, p \rangle$;
- $p(x) := 1/2 \in [0, 1]$;
- $p(y) := 3 \in [\![0, 4]\!]$.

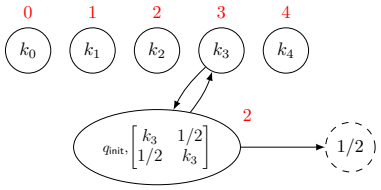


Figure 1 On the left, a game form with set of outcomes $\mathsf{O} := \{x, y\}$. In the middle, the description of an environment on O. On the right, the parity game $\mathcal{G}_{(\{x,y\},\mathcal{F},E)}$. Recall that the dashed state is a stopping state with value 1/2.

For all $u \in [0, 1]$, we denote by $v^u_Y : Q \to [0, 1]$ the valuation such that: $v^u_Y(q_{\mathsf{init}}) = v^u_Y(k_i) := u$ for all $i \in [\![0, e]\!]$ and $v^u_Y(x) := x$ for all $x \in p_{[0,1]}$.

*Furthermore, for all Player-*A GF*-strategies $\sigma_{\mathsf{A}} \in \Sigma_{\mathsf{A}}(\mathcal{F})$, we denote by $\mathsf{s}^Y_{\mathsf{A}}(\sigma_{\mathsf{A}})$ the Player-*A *positional strategy defined by σ_{A} in the arena $\mathcal{C}_Y$.*

The game $\mathcal{G}_Y$ is then equal to $\mathcal{G}_Y := \langle \mathcal{C}_Y, \mathsf{col} \rangle$ where $\mathsf{col}(q_{\mathsf{init}}) := c$ and for all $i \in [\![0, e]\!]$, we have $\mathsf{col}(k_i) := i$.

▶ **Example 16.** This definition is illustrated in Fig. 1 to the right. The colors of the non-stopping states are depicted in red next to the states. Furthermore, the edges from all k_i, for $i \neq 3$, leading back to q_{init} are not represented. ⌟

What we are interested in is the existence of positional optimal strategies for both players. In such games, these strategies are entirely defined by a GF-strategy in a game form $\mathcal{F}$.

▶ **Definition 17** (Optimal GF-strategies). *Given $E = \langle c, e, p \rangle \in \mathsf{Env}(\mathsf{O})$, and $Y := (\mathsf{O}, \mathcal{F}, E)$, a Player-*A GF*-strategy $\sigma_{\mathsf{A}} \in \Sigma_{\mathsf{A}}(\mathcal{F})$ is said to be* optimal *w.r.t. Y if the Player-*A *positional strategy $\mathsf{s}^Y_{\mathsf{A}}(\sigma_{\mathsf{A}})$ is optimal in $\mathcal{G}_Y$. The definition is analogous for Player* B*.*

Given a finite set of outcomes O, we can now define the game forms on O ensuring the existence of optimal strategies w.r.t. all environments.

▶ **Definition 18** (Optimizable game forms). *Given $\mathcal{F} \in \mathsf{Form}(\mathsf{O})$, $n \in \mathbb{N}$, and a player $\mathsf{C} \in \{\mathsf{A}, \mathsf{B}\}$, the game form $\mathcal{F}$ is said to be* positionally maximizable up to n *w.r.t. Player* C *if, for each environment $E \in \mathsf{Env}(\mathsf{O})$ with $\mathsf{Sz}_{\mathsf{C}}(E) \leq n$, there is an optimal* GF*-strategy for Player* C *w.r.t. $(\mathsf{O}, \mathcal{F}, E)$. When this holds for both players, $\mathcal{F}$ is said to be* positionally optimizable up to n*. If this holds for all $n \in \mathbb{N}$, $\mathcal{F}$ is simply said to be* positionally optimizable*.*

▶ Remark 19. Note first that there are some game forms that are not positionally maximizable w.r.t. any player up to 1. This is e.g. the case of the game form appearing in [2, Fig. 2].

Moreover, by definition, from a game form $\mathcal{F} \in \mathsf{Form}(\mathsf{D})$ that is not positionally optimizable up to some $n \in \mathbb{N}$, there exists an environment $E \in \mathsf{Env}(\mathsf{D})$ such that one player has no positional optimal strategy in the parity game $\mathcal{G}_{(\mathsf{D},\mathcal{F},E)}$, where the difference between $\mathsf{col}(q_{\mathsf{init}})$ and the maximum of the colors appearing in $\mathcal{G}_{(\mathsf{D},\mathcal{F},E)}$ is at most n.

▶ **Example 20.** In the game $\mathcal{G}_{(\mathsf{O},\mathcal{F},E)}$ on the right of Fig. 1, Player A has positional optimal strategies: it suffices to play both rows with positive probability. (This is similar for Player B.) As a side remark, the game form on the left of Fig. 1 is positionally optimizable. ⌟

In Lemma 21 below, we formulate more explicitly (using the notion of parity domination from Definition 12) what optimal GF-strategies are.

▶ **Lemma 21.** *Let $E = \langle c, e, p\rangle \in \mathsf{Env}(\mathsf{O})$ and $Y = (\mathsf{O}, \mathcal{F}, E)$. A Player-*$\mathsf{A}$ GF*-strategy $\sigma_\mathsf{A} \in \Sigma_\mathsf{A}(\mathcal{F})$ is* optimal *w.r.t. Y if and only if, letting $u := \chi_{\mathcal{G}_Y}(q_{\mathsf{init}})$, either (i) $u = 0$, or (ii) the positional Player-*A *strategy $\mathsf{s}^Y_\mathsf{A}(\sigma_\mathsf{A})$ parity dominates the valuation v^u_Y.*

Furthermore (ii) is equivalent to:

- *(1) the Player-*A *positional strategy $\mathsf{s}^Y_\mathsf{A}(\sigma_\mathsf{A})$ dominates the valuation v^u_Y; and*
- *(2) for all $b \in \mathsf{Act}_\mathsf{B}$, if the probability under (σ_A, b) to reach a stopping state is null, then $\max(\mathsf{Color}(\mathcal{F}, p, \sigma_\mathsf{A}, b) \cup \{c\})$ is even, where $\mathsf{Color}(\mathcal{F}, p, \sigma_\mathsf{A}, b) := \{i \in [\![0, e]\!] \mid \varrho(\sigma_\mathsf{A}, b)[p^{-1}[\{i\}]] > 0\}$ is the set of colors that can be seen with positive probability under (σ_A, b).*

This is symmetrical for Player B.

▶ Remark 22. This proposition states that for a Player-A GF-strategy σ_A to be optimal in a local game $\mathcal{G}_Y$ with positive value, it must be the case that for every Player-B action b: either there is a positive probability (w.r.t. (σ_A, b)) to exit q_{init} and the expected value of the stopping states visited is at least u; or the game loops on q_{init} with probability 1, and the maximum of the colors that can be seen with positive probability (w.r.t. (σ_A, b)) is even. In particular, if $c \leq \max \mathsf{Color}(\mathcal{F}, p, \sigma_\mathsf{A}, b)$ or if c is odd, then $\max \mathsf{Color}(\mathcal{F}, p, \sigma_\mathsf{A}, b)$ is even.

5 Local environment and global strategy

The goal of this section is to state and prove Theorem 25 below: the main theorem of this paper. This theorem states that it is possible to extract, for every state of a game and for each player, a local environment which summarizes the context of the state to the player, and tells her how to positionally play optimally.

For the remainder of this section, we fix a parity game $\mathcal{G} = \langle \mathcal{C}, \mathsf{col}\rangle$. In particular, the set of states Q is fixed. Before going any further, we give useful notations below.

▶ **Definition 23** (Value slice)**.** *For all subsets of states $S \subseteq Q$, we denote by $V_S := \{u \in [0, 1] \mid \exists q \in S,\ \chi_\mathcal{G}(q) = u\}$ the finite set of values of states in S. Furthermore, for all $u \in V_Q$, we let $Q_u := \{q \in Q \mid \chi_\mathcal{G}(q) = u\}$ be the set of states whose value is u: it is the* u-slice *of $\mathcal{G}$. Finally, for all $u \in V_Q$, we let $e_u := \mathsf{Even}(\mathsf{col}[Q_u])$ and $o_u := \mathsf{Odd}(\mathsf{col}[Q_u])$.*

We also introduce the notion of positional strategies generated by an environment function before stating Theorem 25: these are the positional strategies that play GF-strategies that are optimal in the (local) parity games induced by the environment function.

▶ **Definition 24** (Strategy generated by environment functions)**.** *For all environment functions $\mathsf{Ev} : Q \to \mathsf{Env}(Q)$, a Player-*$\mathsf{A}$ *positional strategy s_A is* generated *by Ev if for all $q \in Q$, the* GF*-strategy $\mathsf{s}_\mathsf{A}(q) \in \Sigma_\mathsf{A}(\mathsf{F}(q))$ is optimal w.r.t. $(\mathsf{O}, \mathsf{F}(q), \mathsf{Ev}(q))$ (and similarly for Player* B*).*

▶ **Theorem 25.** *Let $\mathcal{G} = \langle \mathcal{C}, \mathsf{col}\rangle$ be a parity game. Assume that for all states $q \in Q$, the game form $\mathsf{F}(q)$ is:*

- *positionally maximizable up to $e_{\chi_\mathcal{G}(q)} - \mathsf{col}(q)$ w.r.t. Player* A*; and*
- *positionally maximizable up to $o_{\chi_\mathcal{G}(q)} - \mathsf{col}(q)$ w.r.t. Player* B

Then, there is a function $\mathsf{Ev}_\mathsf{A} : Q \to \mathsf{Env}(Q)$ (resp. $\mathsf{Ev}_\mathsf{B} : Q \to \mathsf{Env}(Q)$) such that all Player-$\mathsf{A}$ *(resp. Player-*B*) positional strategies s_A (resp. s_B) generated by Ev_A (resp. Ev_B) are optimal in $\mathcal{G}$; and such Player-*A *(resp.* B*) positional strategies exist.*

▶ Remark 26. Given some $u \in V_Q$, one can realize that the requirement at states $q, q' \in Q_u$ changes depending on the color of q and q'. More specifically, if $\mathsf{col}(q) < \mathsf{col}(q')$, then the requirement at state q is at least as strong as the requirement at state q' since the game form

$\mathsf{F}(q)$ should behave well for environments of larger size than the game form $\mathsf{F}(q')$. As we shall see, by Proposition 50, the requirement at state q is actually strictly stronger than the requirement at state q'.

The remainder of this section is devoted to an explanation of the construction of the environment function Ev_A (the construction being similar for Player B). We first argue that we can restrict ourselves to a specific u-slice Q_u for some $u \in V_Q$.

▶ **Definition 27** (Game restricted to a u-slice). *For all $u \in V_Q$, let $\mathcal{G}^u$ be the concurrent game obtained from $\mathcal{G}$ by making all states outside of Q_u stopping: for every $q \in Q \setminus Q_u$, we set $\mathsf{val}(q) \leftarrow \chi_\mathcal{G}(q)$. The states, game forms and coloring function on Q_u are left unchanged.*

Interestingly, a Player-A positional strategy optimal in $\mathcal{G}$ can be obtained by merging appropriate positional strategies s^u_A in the games $\mathcal{G}^u$ for all $u \in V_Q \setminus \{0\}$.

▶ **Proposition 28.** *For all $u \in V_Q \setminus \{0\}$, let s^u_A be a Player-A strategy that parity dominates the valuation $\chi_\mathcal{G}$ in $\mathcal{G}^u$. Then, the Player-A positional strategy s_A s.t. $\mathsf{s}_\mathsf{A}(q) := \mathsf{s}^u_\mathsf{A}(q)$ for all $u \in V_Q \setminus \{0\}$ and $q \in Q_u$ guarantees the valuation $\chi_\mathcal{G}$ in $\mathcal{G}$ (i.e. it is optimal).*

This justifies that, for the remainder of this section, we focus on a given u-slice Q_u for some positive $u \in (0, 1]$. We also let $e := e_u$ and $o := o_u$ for the remainder of this section.

5.1 Overview of the proof

In order to give an idea of the steps that we take to prove Theorem 25, let us first consider the very simple case of finite turn-based deterministic (i.e. where all probability distribution over successors states are deterministic) reachability games. In this setting, computing the area L_A from which Player A has a winning strategy can be done inductively. That is, initially we set $L_\mathsf{A} := T$ where T denotes the target that Player A wants to reach. Then, the inductive step is handled with a (deterministic) attractor: we add to L_A any Player-A state with a successor in L_A and any Player-B state with all successors in L_A. After finitely many steps, there is no more state to add in L_A: this exactly corresponds to the states from which Player A has a winning strategy.

Computing a single attractor is not merely enough to take into account the intricate behavior of parity objectives and the complexity of concurrent (and stochastic)interactions, which is what Theorem 25 deals with. Therefore, we are going to iteratively compute several layers of (virtual) colors, with a local update to change the (virtual) color (and therefore the layer it belongs to) of a state. This local update can be seen as an attractor except in a concurrent stochastic setting. Hence, when we update the (virtual) color of a state, we take into account the concurrent interaction of the players at each state along with the probability to see stopping states or states with different (virtual) colors. We define this local update in Subsection 5.3. Let us describe below the steps that we take to capture the behavior of the parity objective.

We compute layers of successive probabilistic attractors with leaks towards the stopping states. Although we compute a strategy, e.g., for Player A, we alternate players to build layers, then move the last non-empty layer into the closest layer with same parity, then backtrack the attractor computation from this layer downwards, and start over again the full attractor computation on the new layer structure. In a more concrete way, let us assume below that the highest color in the u-slice is 6. We proceed as follows:

1. Add the states colored with 4 to layer $\mathcal{L}^4$.
2. Recursively add to $\mathcal{L}^4$ (and give them virtual color 4) the states where Player A can guarantee that with positive probability (pp) either a leak towards stopping states occurs now with expected explicit value at least u ($\mathsf{Leak}_{\geq u}$), or with pp the next state is in $\mathcal{L}^4$.
3. Add the remaining states colored with 3 to layer $\mathcal{L}^3$.
4. Recursively add to $\mathcal{L}^3$ (and give them virtual color 3) the states where Player B can guarantee that either $\mathsf{Leak}_{<u}$ occurs now with pp, or the next state is surely not in $\mathcal{L}^4$ and with pp in $\mathcal{L}^3$.
5. Add the remaining states colored with 2 to layer $\mathcal{L}^2$.
6. Recursively add to $\mathcal{L}^2$ (and give them virtual color 2) the states where Player A can guarantee that either $\mathsf{Leak}_{\geq u}$ will occur with pp, or the maximal layer index of the next states seen with pp is 2 or 4.
7. And so on, for colors 1 and 0. The layers so far only give information about what can happen at finite horizon. For instance, from $\mathcal{L}^2$, Player A can guarantee that either $\mathsf{Leak}_{\geq u}$ will occur with pp, or the maximal color that will be seen with pp is in $\{2, 4\}$.
8. Now, if e.g. $\mathcal{L}^0 \neq \emptyset$, we merge $\mathcal{L}^0$ into $\mathcal{L}^2$ and we reset the states that are in layer $\mathcal{L}^1$. Similarly, if e.g. $\mathcal{L}^0 = \emptyset$ and $\mathcal{L}^1 \neq \emptyset$, we merge $\mathcal{L}^1$ into $\mathcal{L}^3$ and we reset the states that are in layer $\mathcal{L}^2$. This is, arguably, the most surprising step, we justify it in Ex. 42.
9. We then repeat the above attractor alternation from step 1. all over again, until all the states are eventually in $\mathcal{L}^4$, which is bound to happen as we shall prove.

The key property (namely *faithfulness*, defined below in Def. 38) that is growing throughout the above computation and will hold in the final layer $\mathcal{L}^4$ involves layer games: the $\mathcal{L}^n$-game is derived from the u-slice by abstracting each $\mathcal{L}^i$ with $i \neq n$ via one state k_i^n from which the player who dislikes the parity of n chooses any next state in $\mathcal{L}^n$, making it harder for the other to win. If $i > n$ then k_i^n is i-colored, else $(n-1)$-colored, also making it harder for the other to win. And states in $\mathcal{L}^n$ bear their true colors. See for instance Fig. 4. The $\mathcal{L}^n$-game is only seemingly harder to win: it is actually equivalently hard, but its useful properties are easier to prove.

The key growing property is as follows: between two merges, the attractor computation from the top layer down to $\mathcal{L}^n$ ensures that Player A has a positional strategy of value at least u in each $\mathcal{L}^i$-game for even $i \geq n$, and Player B less than u for odd $i \geq n$. In the very end, there is only one even layer with all states bearing their true colors, and no abstract states: the layer game equals the u-slice game. We have thus computed a positional optimal strategy.

Let us hint at how to show positional optimality in the $\mathcal{L}^n$-games when it holds: we break $\mathcal{L}^n$ each into one simple parity game built on $\mathsf{F}(q)$ per state q in $\mathcal{L}^n$, abstracting the other states in $\mathcal{L}^n$ into one. Our theorem assumption yields an optimal GF-strategy for Player A or B in the simple parity game. Gluing them does the job.

5.2 Extracting an environment function from a parity game

For the remainder of the section, we illustrate the definitions and lemmas on the game depicted in Fig. 2 and 3. We give the notations that we use to describe these examples below.

▶ **Example 29.** We explain the notations used to depict this game (it is in fact the same arena in both Fig. 2 and 3, with different coloring functions – real or virtual). On the sides in green are the slices $Q_0, Q_{1/4}, Q_{3/4}$ and Q_1 from left to right. We focus on the central slice $Q_{1/2}$. In $Q_{1/2}$, there are seven states, five of which (the square-shaped ones) are

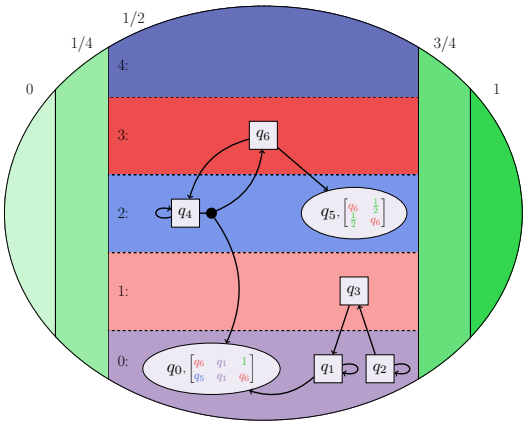


Figure 2 The depiction of a game restricted to the 1/2-slice $Q_{1/2}$ with the initial coloring function col.

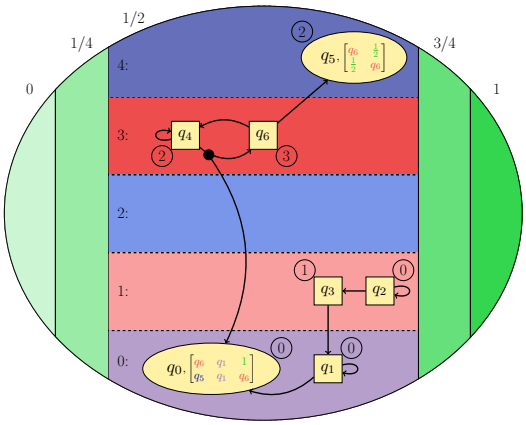


Figure 3 The same game restricted to $Q_{1/2}$ with a different coloring function vcol.

turn-based for Player B, that is, Player A has only one available action. On the other hand, the two circled-shaped states q_0 and q_5 are 'truly" concurrent in the sense that both players have several actions available. Furthermore, note that there is only one non-deterministic distribution function: from q_4, Player B may either loop on q_4 or go to with equal probability to q_0 and q_6. The other arrows lead to a single state and the outcomes of the game forms in q_0 or q_5 is a single state or a value: 1 or 1/2. These formally refer to a (distribution over) stopping states outside of the 1/2-slice $Q_{1/2}$. The horizontal layers depict the colors of the states. In Fig. 2, the coloring function considered is the initial one col whereas in Fig. 3 we have depicted a virtual coloring function vcol. For instance, $\mathsf{col}(q_6) = 3$ whereas $\mathsf{col}(q_5) = 2$. Similarly, $\mathsf{vcol}(q_6) = 3$ whereas state $\mathsf{vcol}(q_5) = 4$. Note that, in Fig. 3, the real colors (given by col) are reminded next to some states with circled numbers. Finally, note that $e := e_{1/2} = 4$. ⌟

Given a virtual coloring function (defining layers), we need to extract local environments from the parity game $\mathcal{G}$, which summarize how the Players see their neighboring states via the virtual coloring function. This is (partly) done in Def. 30 bia a successor function p.

▶ **Definition 30** (Successor function extracted from an arena and a virtual coloring function). *Given $S \subseteq Q_u$ and a virtual coloring function $\mathsf{vcol} : Q_u \to [\![0, e]\!]$. The function $p_{S,\mathsf{vcol}} : S \to S \uplus [\![0, e]\!] \uplus V_{Q \setminus Q_u}$ is such that, for all $d \in \mathsf{D}$:*

- *for all $q \in S$, $p_{S,\mathsf{vcol}}(q) := q \in Q$;*
- *for all $q \in Q \setminus Q_u$, $p_{S,\mathsf{vcol}}(q) := \chi_{\mathcal{G}}(q) \in [0, 1]$;*
- *for all $q \in S_u \setminus S$, $p_{S,\mathsf{vcol}}(q) := \mathsf{vcol}(q)$.*

Given a virtual coloring function $\mathsf{vcol} : Q_u \to [\![0, e]\!]$ and a color $n \in [\![0, e]\!]$, we can now extract a small parity game (the layer-games from Subsection 5.1) from $\mathcal{G}$ where the states with truly concurrent interactions are all in $\mathsf{vcol}^{-1}[n]$ (the interactions at these states is the same as in $\mathcal{G}$), the states in $Q \setminus Q_u$ are stopping states and the arena loops back to $\mathsf{vcol}^{-1}[n]$ when a state in $Q_u \setminus \mathsf{vcol}^{-1}[n]$ is seen. This is done in the next definition.

▶ **Definition 31** (Parity game extracted from the u-slice). *Consider a virtual coloring function $\mathsf{vcol} : Q_u \to [\![0, e]\!]$ and a color $n \in [\![0, e]\!]$. Let $\mathsf{C} \in \{\mathsf{A}, \mathsf{B}\}$ be a Player: A if n is odd and B if n is even. The arena $\mathcal{C}^n_{\mathsf{vcol}} = \langle Q', \mathsf{F}' \rangle$ along with the coloring function $\mathsf{vcol}_n : Q' \to \mathbb{N}$ are such that, denoting $Q_n := \mathsf{vcol}^{-1}[n]$:*

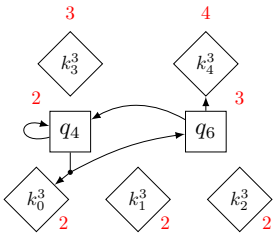


Figure 4 The game $\mathcal{L}^3_{\mathsf{vcol}}$. Exiting arrows from $k_0^3, k_1^3, k_2^3, k_3^3$ and k_4^3 are not depicted: they would all loop back to both q_4 and q_6.

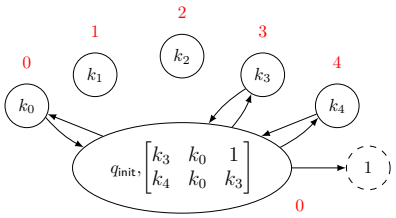


Figure 5 The game $\mathcal{G}^0_{q_0,\mathsf{vcol}}$ with vcol the coloring function depicted in Fig. 3.

- $Q' := Q_n \cup K^n \cup V_{Q\setminus Q_u}$ *where all* $x \in V_{Q\setminus Q_u}$ *are stopping states with* $\mathsf{val}(x) \leftarrow x$*;*
- *for all* $q \in Q_n$, $\mathsf{F}'(q) := \langle \mathsf{Act}^q_{\mathsf{A}}, \mathsf{Act}^q_{\mathsf{B}}, Q', \varrho_{p_{n,\mathsf{vcol}}} \rangle$ *where, for all* $\sigma_{\mathsf{A}} \in \mathsf{Act}^q_{\mathsf{A}}$, $\sigma_{\mathsf{B}} \in \mathsf{Act}^q_{\mathsf{B}}$, *and* $q \in Q$, *we have* $\varrho_{p_{n,\mathsf{vcol}}}(\sigma_{\mathsf{A}}, \sigma_{\mathsf{B}})(q) := \varrho(\sigma_{\mathsf{A}}, \sigma_{\mathsf{B}})[p^{-1}_{n,\mathsf{vcol}}[q]]$*;*
- *for all* $k \in K^n$, *we set* $\mathsf{F}'(k)$ *as a Player-*C *state whose outcomes are all the states in* Q_n*;*
- *for all* $q \in Q_n$, *we let* $\mathsf{vcol}_n(q) := \mathsf{col}(q)$ *and for all* $i \in [\![0, e]\!]$, *we have* $\mathsf{vcol}_n(k^n_i) := \max(i, n-1)$.

For $t \in [0,1]$, *we define the valuation* $v^t_{n,\mathsf{vcol}} : Q' \to [0,1]$*:* $v^t_{n,\mathsf{vcol}}[Q_n \cup K^n] := \{t\}$ *and for all* $x \in V_{Q\setminus Q_u}$, $v^t_{n,\mathsf{vcol}}(x) := x$.

The game $\mathcal{L}^n_{\mathsf{vcol}}$ *is then equal to* $\mathcal{L}^n_{\mathsf{vcol}} = \langle \mathcal{C}^n_{\mathsf{vcol}}, \mathsf{vcol}_n \rangle$.

The notation $\mathcal{L}^n_{\mathsf{vcol}}$ comes from the fact that the game is extracted for the n-colored layer w.r.t. the coloring function vcol. The idea behind Def. 31 is the following: the states of interest are those of Q_n, that is, those for which the virtual color given by vcol is n. Note however that the colors of these states in $\mathcal{L}^n_{\mathsf{vcol}}$ are given by the real coloring function col. On the other hand, for all $i \in [\![0, e]\!]$, the state k^n_i in $\mathcal{L}^n_{\mathsf{vcol}}$ correspond to the states in $\mathcal{G}^u$ colored with i w.r.t. vcol. In the case where n is even, as formally defined later in Def. 37, we will require that any Player-A positional strategy generated by a given environment has value at least u, in the game $\mathcal{L}^n_{\mathsf{vcol}}$, from all states in Q_n. However, all states k^n_i for $i \in [\![0, e]\!]$ are Player-B's, who can then choose to loop back to any state in Q_n. Therefore, given a Player-A positional strategy s_{A}, if the game cannot exit to any stopping state, for the strategy s_{A} not to have value 0, the game may loop on some k^n_i only at the condition that the highest color seen with positive probability is even. In addition, note that the color of the state k^n_i for $i \in [\![0, n-1]\!]$ is $n-1$ (which is odd). Hence, all other things being equal, the game is harder for Player A when $n = 4$ than when $n = 2$ or 0.

▶ **Example 32.** The game $\mathcal{L}^3_{\mathsf{vcol}}$ is partly depicted on the left of Fig. 4 (the virtual coloring function vcol being the one depicted in Fig. 3). The colors of the states in $\mathcal{L}^3_{\mathsf{vcol}}$ are depicted in red (for the central states q_4 and q_6, it is their real color in the original game). Although the arrows are not depicted, from all states $k^3_0, k^3_1, k^3_2, k^3_3$ and k^3_4, Player A can decide to which state among $\{q_4, q_6\}$ to loop back (since $n = 3$ is odd). In an even-colored layer, it would have been Player B to decide. The color of states k^3_i is i if $i \geq 3$ or 2 if $i \leq 2$. ⌟

Given a virtual coloring function, we also associate a local environment to each state. This is crucial since this will allow us to properly define (in Definition 35 below) the probabilistic attractor with leaks towards the stopping states mentioned in Subsection 5.1.

▶ **Definition 33** (Induced local environment). *Given $q \in Q_u$, a virtual coloring function* $\mathsf{vcol} : Q_u \to [\![0, e]\!]$ *and* $n \in [\![0, e]\!]$*, the environment associated to state q w.r.t.* vcol *and n is* $E^n_{q,\mathsf{vcol}} := \langle \max(c_n, \mathsf{vcol}(q)), e, p_{\{q\},\mathsf{vcol}} \rangle$ *where* $c_n = n + 1$ *if n is odd and* $c_n := n - 1$ *if n is even. The corresponding (local) game* $\mathcal{G}_{(\mathsf{O},\mathsf{F}(q),E^n_{q,\mathsf{vcol}})}$ *is denoted* $\mathcal{G}^n_{q,\mathsf{vcol}}$*, and for all* $x \in [0, 1]$*, we set* $v^x_{q,\mathsf{vcol}} := v^x_{(\mathsf{O},\mathsf{F}(q),E^0_{q,\mathsf{vcol}})}$ *(see Def. 14).*

▶ **Example 34.** The game $\mathcal{G}^n_{q_5,\mathsf{col}}$ is depicted in Fig. 1 (right) for $n = 0, 1, 2$. However, if $n = 3$, the color of q_{init} would be 4, and if $n = 4$, it would be 3. The game $\mathcal{G}^n_{q_0,\mathsf{vcol}}$ is depicted in Fig. 5 for $n = 0$. However, if $n = 1$, the color of q_{init} would be 2, if $n = 2$, the color of would be 1, if $n = 3$, the color would be 4 and if $n = 4$ the color would be 3. ⌟

5.3 Local operator

We want to define a way to update a virtual coloring function vcol. This amounts to computing the probabilistic attractor with leaks towards the stopping states mentioned in Subsection 5.1. This is done via a local operator mapping a given state q to the best color k for which Player A can achieve the value u in the local parity game $\mathcal{G}^k_{q,\mathsf{vcol}}$. Note that "best" is to be understood considering an ordering compatible with the parity objective. Specifically, taking the point-of-view of Player A, any even number is better than any odd number, and when they increase, odd numbers get worse whereas even numbers get better. This induces a new total strict order relation $\prec_{\mathsf{par}}$ on $\mathbb{N}$ such that, for all $m, n \in \mathbb{N}$, we have $m \prec_{\mathsf{par}} n$ if: m is odd and n is even; or $m > n$ and m and n are odd; or $m < n$ and m and n are even.

▶ **Definition 35** (Local operator). *Let $q \in Q_u$, and* $\mathsf{vcol} : Q_u \to [\![0, e]\!]$ *a (possibly virtual) coloring function. The color* $\mathsf{NewCol}(q, \mathsf{vcol}) \in \mathbb{N}$ *induced by* vcol *at q is defined by:*

$$\mathsf{NewCol}(q, \mathsf{vcol}) := \max_{\prec_{\mathsf{par}}} \left\{ n \in [\![0, e]\!] \mid \chi_{\mathcal{G}^n_{q,\mathsf{vcol}}}(q_{\mathsf{init}}) \geq u \right\}$$

The meaning of a new virtual color n assigned to a state q via NewCol is the following: in the game $\langle \mathcal{C}, \mathsf{vcol} \rangle$, from state q and in at most one step, the highest color w.r.t. vcol seen with positive probability when both players play optimally is n (and no stopping state is seen).

Let us explain the choice of c_n in Def. 33. In a local environment parameterized by n, the integer n induces a shifted parity objective for Player A: her objective is that the maximal color seen infinitely often is at least n w.r.t. $\prec_{\mathsf{par}}$; in particular $n = 0$ induces the standard parity objective. The value c_n encodes that winning condition. For instance, if $n = 2$, assuming $\mathsf{vcol}(q) = 0$ for simplicity, then $c_n = 1$, which implies that seeing 0 infinitely often is not enough, but seeing 2 infinitely often is enough to win.

▶ **Example 36.** We first consider Fig. 1 and we compute $\mathsf{NewCol}(q_5, \mathsf{col})$ on the game on the right. We realize that, regardless of the color of state q_{init}, Player A can (positionally) play both rows with positive probability and ensure reaching (almost-surely) the stopping state 1/2: for all $n \in [\![0, 4]\!]$, $\chi_{\mathcal{G}^n_{q_5,\mathsf{col}}}(q_{\mathsf{init}}) = 1/2$. Hence, $\mathsf{NewCol}(q_5, \mathsf{col}) = 4$.

We consider Fig. 5 and we compute $\mathsf{NewCol}(q_0, \mathsf{vcol})$. As mentioned in Ex. 34, the game $\mathcal{G}^4_{q_0,\mathsf{vcol}}$ corresponds to the game depicted in Fig. 5 except that q_{init} is colored with 3. One can realize that, with this choice (of coloring of the state q_{init}), if the highest color $i \in [\![0, 4]\!]$ such that k_i is seen infinitely often is such that $i \prec_{\mathsf{par}} 4$, then Player A loses. The value of this game is 0 as Player B can ensure looping on k_0 and q_{init} (by playing, positionally and deterministically, the middle column) thus ensuring that the highest color seen infinitely often is 3. Thus, $\mathsf{NewCol}(q_0, \mathsf{vcol}) \prec_{\mathsf{par}} 4$. In the game $\mathcal{G}^2_{q_0,\mathsf{vcol}}$, q_{init} is colored with 1. Again, with this choice (of coloring of the state q_{init}), if the highest color $i \in [\![0, 4]\!]$ such that k_i is seen infinitely is such that $i \prec_{\mathsf{par}} 2$, then Player A loses. The value of this game is also 0

as Player B can still play the middle column ensuring that the highest color seen infinitely often is 1. Thus, $\mathsf{NewCol}(q_0, \mathsf{vcol}) \prec_{\mathsf{par}} 2$. Consider now the game $\mathcal{G}^0_{q_0,\mathsf{vcol}}$, the one depicted in Fig. 5. The value of the state q_{init} is now 1. Indeed, if Player A plays the two rows with equal probability, one can see that this strategy parity dominates (see Def. 12) the valuation $v^1_{q_0,\mathsf{vcol}}$ (recall Def. 33). Indeed, the BSCCs compatible with this strategy are $\{q_{\mathsf{init}}, k_3, k_4\}$ and $\{q_{\mathsf{init}}, k_0\}$ and they are even-colored. Hence, by Proposition 13, $\chi_{\mathcal{G}^0_{q_1,\mathsf{col}}}(q_{\mathsf{init}}) = 1 \geq 1/2$ and $\mathsf{NewCol}(q_0, \mathsf{vcol}) \succeq_{\mathsf{par}} 0$. That is, $\mathsf{NewCol}(q_0, \mathsf{vcol}) = 0$. ⌟

5.4 Faithful coloring function

To prove Theorem 25, we iteratively build a virtual coloring function and a local environment function. We want to define the desirable property that the pair of coloring and environment functions should satisfy that will be preserved step by step. First, we need to define the notion of an environment function witnessing a color.

▶ **Definition 37** (Environment witnessing a color). *Let* $\mathsf{vcol} : Q_u \to [\![0, e]\!]$ *be a virtual coloring function,* $n \in [\![0, e]\!]$*, and* $\mathsf{Ev} : \mathsf{vcol}^{-1}[n] \to \mathsf{Env}(\mathsf{D})$ *be an environment function. Assuming that* n *is even (resp. odd), we say that the pair* $(\mathsf{vcol}, \mathsf{Ev})$ witnesses *the color* n *if for all* $q \in \mathsf{vcol}^{-1}[n]$, $\mathsf{Sz}_{\mathsf{A}}(\mathsf{Ev}(q)) \leq e - \mathsf{col}(q)$ *(resp.* $\mathsf{Sz}_{\mathsf{B}}(\mathsf{Ev}(q)) \leq o - \mathsf{col}(q)$*) and all positional strategies* s_{A} *(resp.* s_{B}*) generated by* Ev *(recall Def. 24) in the game* $\mathcal{L}^n_{\mathsf{vcol}}$ *parity dominate the valuation* $v^u_{n,\mathsf{vcol}}$ *(resp.* $v^{u'}_{n,\mathsf{vcol}}$ *for some* $u' < u$*), recall Def. 31.*

It means that, in the virtual games given by vcol, in the even layers, Player A can achieve at least what she should be able to achieve in this u-slice (i.e. the value of the states is at least u). Whereas, in the odd-colored layers, Player B can prevent Player A from achieving this.

We can now define the central notion of interest: for a pair of coloring function and environment function to be faithful (to what really happens in the parity game). We only give a definition of faitfhfulness that we can use in this paper, but note that in [4], we require additional properties for faithfulness.

▶ **Definition 38** (Faithful pair of coloring and environment functions). *Let* $\mathsf{vcol} : Q_u \to [\![0, e]\!]$ *be a virtual coloring function,* $n \in [\![0, e+1]\!]$*, and* $\mathsf{Ev} : Q_u \to \mathsf{Env}(\mathsf{D})$ *a partial environment function defined on* $\mathsf{vcol}^{-1}[[\![n, e]\!]]$*. We say that* $(\mathsf{vcol}, \mathsf{Ev})$ *is* faithful down to n *if:*

- *for all* $k \in [\![n, e]\!]$*, the pair* $(\mathsf{vcol}, \mathsf{Ev})$ *witnesses color* k*;*
- *for all* $q \in Q_u$*, if* $\mathsf{vcol}(q) < n$*, then* $\mathsf{col}(q) = \mathsf{vcol}(q)$ *and* $\mathsf{NewCol}(q, \mathsf{vcol}) < n$*;*

If $n = 0$*, we say that the pair* $(\mathsf{vcol}, \mathsf{Ev})$ *is* completely faithful.

A benefit of faithful environments and coloring functions lies in the proposition below: if all states are mapped w.r.t. the coloring function to e, then the environment function guarantees the value u in the whole u-slice Q_u.

▶ **Proposition 39.** *For a coloring function* $\mathsf{vcol} : Q_u \to [\![0, e]\!]$ *and an environment function* $\mathsf{Ev} : Q_u \to \mathsf{Env}(\mathsf{D})$*, assume that* $(\mathsf{vcol}, \mathsf{Ev})$ *is completely faithful and that* $\mathsf{vcol}[Q_u] = \{e\}$*. Then, all Player-*A *positional strategies generated by the environment function* Ev *parity dominate the valuation* $\chi_{\mathcal{G}}$ *in the game* $\mathcal{G}^u$*.*

Proof. This is direct from the definitions. Indeed, as $(\mathsf{vcol}, \mathsf{Ev})$ is completely faithful, it witnesses the color e: all Player-A positional strategies s_{A} generated by Ev in the game $\mathcal{L}^e_{\mathsf{vcol}}$ parity dominate the valuation $v^u_{e,\mathsf{vcol}}$. Since $\mathsf{vcol}[Q_u] = \{e\}$, the games $\mathcal{L}^e_{\mathsf{vcol}}$ and $\mathcal{G}^u$ are identical. Similarly, the valuation $v^u_{e,\mathsf{vcol}}$ is equal to the valuation $\chi_{\mathcal{G}}$ in the game $\mathcal{G}^u$. ◀

5.5 Computing a completely faithful pair

Given Prop. 39, our goal is to come up with a pair of an environment function and a coloring function completely faithful such that all states are colored with e. We first consider how to obtain a completely faithful pair from the initial coloring function and the empty environment function (which is faithful down to $e+1$): we proceed by building a new pair that is faithful down to $n-1$, given a pair $(\mathsf{vcol}, \mathsf{Ev})$ faithful down to some $n \in [\![1, e+1]\!]$.

To do so, let us be guided by the second property of faithfulness: to be faithful down to $n-1$, no state $q \in Q_u$ such that $\mathsf{vcol}(q) \leq n-2$ should be such that $\mathsf{NewCol}(q, \mathsf{vcol}) = n-1$. Hence, we adopt the following procedure $\mathsf{UpdateColEnv}$: we first associate an environment to all states whose color is already $n-1$. Then, for all states $q \in Q_u$ such that $\mathsf{NewCol}(q, \mathsf{vcol}) = n-1$, we change their colors to $n-1$ until no state $q \in Q_u$ with $\mathsf{vcol}(q) \leq n-2$ satisfies $\mathsf{NewCol}(q, \mathsf{vcol}) = n-1$. The environment associated to each such state q newly colored by $n-1$ is given by the coloring function vcol for which $\mathsf{NewCol}(q, \mathsf{vcol}) = n-1$ for the first time (crucially, this is done before the color of q is updated to $n-1$). We state as an informal lemma the property satisfied by the procedure described above.

▶ **Lemma 40.** *Let* $\mathsf{vcol} : Q_u \to [\![0, e]\!]$, $n \in [\![1, e+1]\!]$ *be a coloring function, and* $\mathsf{Ev} : Q_u \to \mathsf{Env}(\mathsf{D})$ *be a partial environment function defined on* $\mathsf{vcol}^{-1}[[\![n, e]\!]]$. *Assume that* $(\mathsf{vcol}, \mathsf{Ev})$ *is faithful down to* n. *Let* $(\mathsf{vcol}', \mathsf{Ev}') \leftarrow \mathsf{UpdateColEnv}(n-1, \mathsf{vcol}, \mathsf{Ev})$ *be the pair computed by the procedure described above. Then* $(\mathsf{vcol}', \mathsf{Ev}')$ *is faithful down to* $n-1$.

We illustrate below this lemma on examples.

▶ **Example 41.** We consider the example depicted in Fig. 2. The first step is to build a pair that is faithful down to $e = 4$. As mentioned in Ex. 36, we have $\mathsf{NewCol}(q_5, \mathsf{col}) = 4$. Hence, the color of this state is changed to 4 (we obtain a virtual coloring function vcol^{q_5}) and we set $\mathsf{Ev}(q_5) := E^4_{q_5,\mathsf{col}}$. Note that a Player-A GF-strategy σ_A is optimal in this environment if and only if it plays both rows with positive probability. Furthermore, note that, in the extracted game $\mathcal{L}^4_{\mathsf{vcol}^{q_5}}$, a Player-A positional strategy playing such a GF-strategy σ_A in q_5 parity dominates the valuation $v^{1/2}_{Q_4,\mathsf{col}^{q_5}}$. Hence, the pair $(\mathsf{vcol}^{q_5}, \mathsf{Ev})$ is faithful down to 4.

Consider now the layer 3. First, the state q_6 already has color 3, so it only remains to set its environment: $\mathsf{Ev}(q_6) := E^3_{q_6,\mathsf{vcol}^{q_5}}$. We then realize that $\mathsf{NewCol}(q_4, \mathsf{vcol}^{q_5}) = 3$. Indeed, q_4 is colored with 2 and may go with equal probability to a state colored with 0 and to a state colored with 3. The color of this state is therefore changed, thus obtaining a new virtual coloring function $\mathsf{vcol}^{q_5,q_6,q_4}$. We set its environment: $\mathsf{Ev}(q_4) := E^3_{q_4,\mathsf{vcol}^{q_5}}$. One can realize that the pair $(\mathsf{vcol}^{q_5,q_6,q_4}, \mathsf{Ev})$ witnesses the color 3: a positional Player-B strategy generated by this environment would be so that (i) from q_6, it goes to q_4 with probability 1 (to avoid k_4 that is colored with 4) and (ii) from q_5, it goes to q_6 with positive probability (to see the color 3 with positive probability). Such a strategy has value 0 in the game $\mathcal{L}^3_{\mathsf{vcol}^{q_5}} = \mathcal{L}^3_{\mathsf{vcol}}$ from Fig. 4, hence the pair $(\mathsf{vcol}^{q_5,q_6,q_4}, \mathsf{Ev})$ witnesses the color 3.

We illustrate on this step why the environment needs to be set before setting the new color and not after. That is, we explain why it would not be correct to set $\mathsf{Ev}(q_4) := E^3_{q_4,\mathsf{vcol}^{q_5,q_6,q_4}}$ instead of what we do above. In this environment, the state q_4 has color 3. Hence, looping with probability 1 on q_4 is an optimal GF-strategy for Player B w.r.t. $(\mathsf{D}, \mathsf{F}(q_4), \mathsf{Ev}(q_4))$. Then, the corresponding pair of coloring and environment functions would not witness the color 3. Indeed, a Player B strategy that loops with probability 1 on q_4 is generated by this environment, and it has value $1 \geq u$ (because the real color of this state is 2, and not 3).

This process is repeated down to 0. In Fig. 3, the depicted coloring function (with an appropriate environment function, not shown in Fig. 3) are in fact completely faithful (which is what the procedure $\mathsf{UpdateColEnv}$ would output on the coloring function of Fig. 2). ⌟

Proof sketch. We want to prove that the pair $(\mathsf{vcol}', \mathsf{Ev}')$ witnesses the color $n-1$ (the other condition for faithfulness is ensured by the construction). We consider the case where $n-1$ is even, the other case is similar (but one needs to take the point-of-view of Player B). Consider a Player-A positional strategy $\mathsf{s_A}$ generated by the environment function Ev' in the game $\mathcal{L}^{n-1}_{\mathsf{vcol}'}$. Let $Q_{n-1} := \mathsf{vcol}'^{-1}[n-1]$ and let $v := v^u_{n-1,\mathsf{vcol}'}$. For every $q \in Q_{n-1}$, let $Y_q := (\mathsf{D}, \mathsf{F}(q), \mathsf{Ev}'(q))$ be the local environment at state q and let $\mathsf{Ev}'(q) = \langle c_q, e, p_q \rangle$. From the characterization of Lemma 21 (item (ii.1)), by carefully analyzing the links between the local games $\mathcal{G}_{Y_q}$ for all $q \in Q_{n-1}$ and the game $\mathcal{L}^{n-1}_{\mathsf{vcol}'}$, we can show that the strategy $\mathsf{s_A}$ dominates the valuation v.

It remains to show that all BSCCs (that are not reduced to a stopping state and are) compatible with s_A are even-colored. Consider such a BSCC H and a Player-B deterministic positional strategy $\mathsf{s_B}$ which induces H. For every state $q \in H$, since no stopping state appears in H, it must be that the probability to reach a stopping state in $\mathcal{G}_{Y_q}$ w.r.t. (σ_A, b) is 0. For every state $q \in Q_{n-1}$, the coloring function vcol_q associated with environment $\mathsf{Ev}'(q)$ is such that $\mathsf{vcol}_q(q) \leq n-1$.[1] Hence, the color c_q is such that $c_q = \max(n-2, \mathsf{vcol}_q(q)) \leq n-1$. Now, assume that some state k_i is in H for some $i > n-1 \geq c_q$. In that case, as explained in Remark 22, the highest i such that k_i is in H must be even. Hence, H is even-colored. Assume now that no state k_i in H is such that $i > n-1$. In that case, if a state in H has color $n-1$ (like the state q_6 in Fig. 3 in the case where $n-1=3$), then $n-1$ is the highest color in H and H is even-colored. Consider the first state q whose color is now $n-1$ (w.r.t. vcol') but whose previous color was not $n-1$. In that case, we have $c_q = \max(n-2, \mathsf{vcol}_q(q)) = n-2$ is odd. Furthermore, the state q has changed its color because $\mathsf{NewCol}(q, \mathsf{vcol}_q) = n-1$. With Remark 19, since $\mathsf{s_A}(q)$ is optimal w.r.t. Y_q, it follows that there is a positive probability to reach, in the game $\mathcal{G}_{Y_q}$ the state k_{n-1}. In the game $\mathcal{L}^{n-1}_{\mathsf{vcol}'}$, this corresponds to a positive probability to reach a state $q' \in H$ colored with $n-1$ w.r.t. vcol_q (recall Def. 30). Since q is the first state to have changed its color, we can deduce that q' already had color $n-1$ w.r.t. vcol. Furthermore, one can show that q' is colored with $n-1$ w.r.t. the real coloring function col. Overall, in the game $\mathcal{L}^{n-1}_{\mathsf{vcol}'}$, with the GF-strategy $\mathsf{s_A}(q)$, there is a positive probability to reach in one step a state q' colored with $n-1$. Iteratively, we obtain that, considering the k-th state whose color is now $n-1$ (i.e. w.r.t. vcol') but whose initial color was not $n-1$, there is a positive probability to reach (in at most k steps) a state colored with $n-1$. Hence, the highest color appearing in H is $n-1$, which is even. We obtain that $\mathsf{s_A}$ parity dominates the valuation v. ◀

Applying iteratively this algorithm on all colors from e down to 0 starting with the initial coloring function induces a completely faithful pair $(\mathsf{vcol}, \mathsf{Ev})$. However, it may be the case that some states are not mapped to e, which does not allow us to apply Prop. 39. The question is then: from such a completely faithful configuration, how can one make some progress towards a situation where Prop. 39 can be applied?

▶ **Example 42.** Consider the coloring function of Fig. 3. As mentioned in Ex. 41, with an appropriate environment function (not shown in Fig. 3), we have a completely faithful pair. To gain some intuition on what should be done next, let us focus only on the states q_1, q_2, q_3. A simplified version is presented in Fig. 6 (with a slight modification: instead of going to q_0, q_1 loops on itself): the initial (and true) colors of the states are in circles next to them and their color w.r.t. the current virtual coloring function is written in red. In this game, it is

[1] This is because all states $q \in Q_{n-1}$ satisfy $\mathsf{col}(q) \leq n-1$. This is one of the additional conditions for faithfulness that we did mention, but that is used in the definition of faithfulness in [4].

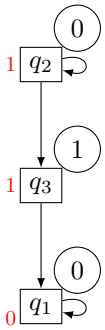


Figure 6 A (deterministic turn-based) game with only three states.

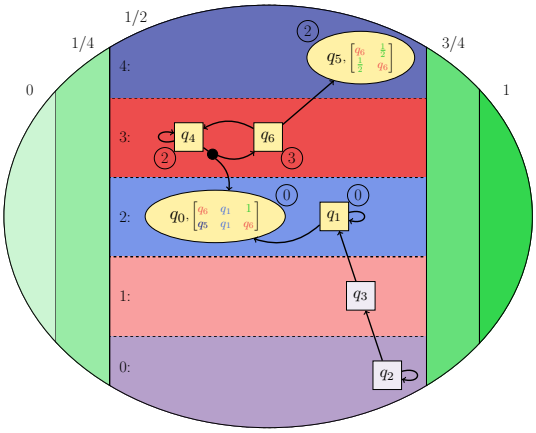


Figure 7 The same arena as in Fig. 2,3 but with a different coloring function.

obvious that Player A wins surely from q_2: indeed, either the game stays indefinitely in q_2, or it eventually reaches and settles in q_1.

The current virtual color 1 assigned to both q_2 and q_3 does not properly reflect the fact that if the game reaches q_3, even though Player B plays optimally according to the local game associated to q_2, it will end up looping in q_1, which will be losing for Player B. In a way, we would like to propagate the information that reaching q_1 is bad for Player B. Since 0 is the smallest color, there is no harm in increasing it to 2, the game from q_1 will be the same: it will be won by Player A by looping. Player B will now be able to know that going to q_1 is dangerous for her, which will be obtained by applying the previous iterative process.

In a more general concurrent game, the next step of the process when we have a completely faithful configuration not satisfying the assumptions of Proposition 39 consists in changing all the states with the least virtual color n to the color $n+2$. However, note that there is a (very important) second step: the colors of all states virtually colored with $n+1$ should be reset to their initial colors. The reason why can be seen again in Fig. 6. After the color of q_1 becomes 2, the color of q_3 will also become 2. However, if the color of the state q_2 is not reset, then it is not going to change since Player B can choose to loop to q_2 and see the color 1 for ever (in game $\mathcal{G}^0_{q_2,\mathsf{vcol}}$). That is, from Player B's perspective, looping on q_2 is winning, which is not what happens in the real game: the coloring function does not faithfully describes what happens in the game. The changes made to the coloring function vcol from Fig. 3 can be seen in Fig. 7. Note that the process of increasing the colors of some states by 2 can only be done with the *least* color (otherwise faithfulness will not be preserved). ⌟

The process IncLeast described in Ex. 42 can be summed up as follows: we increase the least virtually-colored layer n by 2 and we reset the environment and colors of the last but least virtually-colored layer. It ensures faithfulness down to $n+2$ if the initial pair is completely faithful, as informally stated below.

▶ **Lemma 43.** *Let* $\mathsf{vcol} : Q_u \to [\![0, e]\!]$, $\mathsf{Ev} : Q_u \to \mathsf{Env}(\mathsf{D})$ *be a coloring and an environment function. Let* $n := \min \mathsf{vcol}[Q]$. *Assume that* $n \leq e-2$ *and the pair* $(\mathsf{vcol}, \mathsf{Ev})$ *is completely faithful. If* $(\mathsf{vcol}', \mathsf{Ev}') \leftarrow \mathsf{IncLeast}(\mathsf{vcol}, \mathsf{Ev})$ *is the result of increasing the least-colored layer by 2 and resetting the environment of the last but least-colored layer as described above, then* $(\mathsf{vcol}', \mathsf{Ev}')$ *is faithful down to* $n+2$.

Proof sketch. Let $Q_n := \mathsf{vcol}^{-1}[n]$ and $Q_{n+2} := \mathsf{vcol}^{-1}[n+2]$. Let us argue that the pair $(\mathsf{vcol}'', \mathsf{Ev})$ obtained after increasing the least color by 2, before resetting the color and environment of the last but least-colored layer, witnesses the color $n+2$.

Consider a Player-A positional strategy $\mathsf{s_A}$ generated by the environment Ev in the game $\mathcal{L}^{n+2}_{\mathsf{vcol}''}$. Let $v := v^u_{n+1,\mathsf{vcol}''}$. Similarly to the case of Lemma 40, $\mathsf{s_A}$ dominates the valuation v. Consider a BSCC H compatible with $\mathsf{s_A}$. If $H \cap Q_{n+2} = \emptyset$, then H is even-colored. Indeed, $(\mathsf{vcol}, \mathsf{Ev})$ witnesses the color n and, in addition, the probability to go an $(n+1)$-colored state k_i^{n+2} in the game $\mathcal{L}^{n+2}_{\mathsf{vcol}''}$ is exactly the probability to go to an $(n+1)$-colored state k_i^n in the game $\mathcal{L}^n_{\mathsf{vcol}}$ (since n is the least color). Furthermore, H is also even-colored as soon as $H \cap Q_n = \emptyset$ since $(\mathsf{vcol}, \mathsf{Ev})$ witnesses the color $n+2$. Now, assume that none of these cases occur. Then, one can show that: either a state k_i is seen for some $i \geq n+2$, and H is even-colored; or, from some states in Q_{n+2}, there is a positive probability to exit Q_{n+2} and no state k_i is seen for $i \geq n+2$. Now, looking at what happens in game $\mathcal{L}^{n+2}_{\mathsf{vcol}}$, some states k_i are seen for $i \leq n+1$, and such states are colored with $n+1$. Hence, since $(\mathsf{vcol}, \mathsf{Ev})$ witnesses the color $n+2$, it must be that the highest color in H is $n+2$, which is even. Therefore it is also the case in the game $\mathcal{L}^{n+2}_{\mathsf{vcol}''}$. In all the cases, H is even-colored. ◀

As stated in Lemma 43, the update of colors described in Ex. 42 can be done only if, for a completely faithful pair, the least virtual color n appearing is at most $e-2$. If $n = e$, we are actually in the scope of Lemma 39 since in that case all states have virtual color e. However, there remains the case where we have $n = e-1$. In fact, this case cannot happen.

▶ **Lemma 44.** *Consider a coloring function* $\mathsf{vcol} : Q_u \to [\![0, e]\!]$*, an environment function* $\mathsf{Ev} : Q_u \to \mathsf{Env}(\mathsf{D})$*. Assume that* $(\mathsf{vcol}, \mathsf{Ev})$ *is completely faithful. Then, for* $C := \mathsf{vcol}[Q]$*, we have* $\min C \neq e-1$.

Proof sketch. Let $Q_{e-1} := \mathsf{vcol}^{-1}[e-1]$. Towards a contradiction, let $\mathsf{s_B}$ be a Player-B positional strategy generated by Ev in the game $\mathcal{L}^{e-1}_{\mathsf{vcol}}$. It parity dominates the valuation $v^{u'}_{e-1,\mathsf{vcol}}$ for some $u' < u$. Hence, all BSCCs compatible with $\mathsf{s_B}$ are odd-colored: they all stay in the layer Q_{e-1}. Indeed, since $e-1 = \min C$, exiting Q_{e-1} while staying in Q_u mean seeing $Q_e := \mathsf{vcol}^{-1}[e]$ with e even and the highest color in the game. Hence, either the game stays indefinitely in Q_{e-1} and Player B wins almost surely, or there is some positive probability to visit stopping states, and in that case their expected values is at most u'. Hence, in the game $\mathcal{G}^u$, the strategy $\mathsf{s_B}$ has values less than u from the states $Q_{e-1} \subseteq Q_u$, which is a contradiction. ◀

Finally, all these pieces are put together by iteratively applying UpdateColEnv until we obtain a completely faithful pair and applying IncLeast to a completely faithful pair to make some progress towards the completely faithful pair where all states are colored with e. The only remaining step is to prove the termination of this procedure. Consider the virtual coloring functions as vectors in $\mathbb{N}^{e+1}$ indicating the number of states mapped to each color. Then, one can realize that applying UpdateColEnv does not decrease these vectors for a lexicographic order (i.e. we first compare the number of states mapped to e, then the number of states mapped to $e-1$, etc). Furthermore, applying IncLeast increases these vectors for a lexicographic order. In addition, the maximum for this order is achieved when all states are colored with e. Hence, the procedure described above terminates in finitely many steps. We can now finalize the argument for proving Theorem 25.

Proof sketch of Theorem 25 for Player A. Pick $u \in V_{\mathcal{G}} \setminus \{0\}$. According to the previous discussion, there is a completely faithful pair of environment and coloring functions $(\mathsf{vcol}_u, \mathsf{Ev}^u_{\mathsf{A}})$ mapping each state in Q_u to e_u. Hence, by Proposition 39, all Player-A positional strategies

generated by the environment function $\mathsf{Ev}_{\mathsf{A}}^u$ parity dominate the valuation $\chi_{\mathcal{G}}$ in the game $\mathcal{G}^u$. Since we assume that all game forms appearing in Q_u are positionally maximizable up to $e_u - \mathsf{col}(q)$ w.r.t. Player A, such positional strategies generated by $\mathsf{Ev}_{\mathsf{A}}^u$ do exist. Considering the environment function $\mathsf{Ev}_{\mathsf{A}} : Q \to \mathsf{Ev}(\mathsf{D})$ that merges all the environment functions $(\mathsf{Ev}_{\mathsf{A}}^u)_{u \in V_{\mathcal{G}} \setminus \{0\}}$ together (and that is defined arbitrarily on Q_0), it follows by Proposition 28, that all Player-A positional strategies generated by that environment function Ev are optimal. ◀

6 Discussion on positionally optimizable game forms

As mentioned in the introduction, this work extends previous lines of research [1, 2, 3]. We discuss the more closely related work [3]. The goal in [3] was to characterize the game forms ensuring the existence of almost-optimal positional strategies for Büchi objectives (resp. optimal positional strategies for co-Büchi objectives). In both cases, there is a lift from local properties to global properties, similarly to what is done in this work. However, the techniques are quite different: in [3], the proofs involve the use of nested fixed points, as is done for computing values of graph games [9]. This establishes a link between local and global behaviors. However, this comes at the cost of having to handle local (i.e. at game form level) and global (i.e. at graph game level) fixed-points. With Büchi and co-Büchi objectives, there are only two nested fixed points. (Recall that they can be expressed as two-color parity objectives.) For general parity objectives, the number of fixed points would be linear in the number of involved colors. That is why, in this work, we decided to consider good local behaviors in a more abstract way, without considering how the values are effectively computed. That way, we handle arbitrarily many colors instead of only two without prohibitive complexification.

We conclude with a discussion on positionally optimizable game forms. First, we would like to emphasize that Theorem 25 along with Remark 19 give exactly the game forms that should be used in parity games to ensure the existence of positional optimal strategies for both players. Indeed:

- By Remark 19, given any game form that is not positionally optimizable, one can build a small parity game from it, like in Definition 14, where a player has no optimal strategy;
- By Theorem 25, if all the local interactions occurring in a concurrent parity game are positionally optimizable game forms, then both players have positional optimal strategies.

However, it has to be noted that there are concurrent parity games with non-positionally optimizable local interactions where both players have positional optimal strategies. This is e.g. the case of parity games without stopping states where all states have the same color.

Let us now give some properties that are ensured by positionally optimizable game forms. We first give some notations for positionally optimizable game forms (and the corresponding decision problems).

▶ **Definition 45.** *For $n \in \mathbb{N}$, we let $\mathsf{ParO}(n)$ be the set of all game forms positionally optimizable up to n and we let $\mathsf{IsO}(n)$ be the problem of deciding whether a game form is in $\mathsf{ParO}(n)$. Furthermore, we let $\mathsf{ParO} := \bigcap_{n \in \mathbb{N}} \mathsf{ParO}(n)$ and we denote by IsO the problem of deciding whether a game form is in ParO.*

First, let us introduce the notion of relevant environment, i.e. environment $E = \langle c, e, p \rangle$ such that $c \in \{0, 1\}$ and p takes all values in $[\![c, e]\!]$. They are fomally defined below.

▶ **Definition 46.** *For a set of outcomes* O*, an environment* $E = \langle c, e, p\rangle \in \mathsf{Env}$ *is* relevant *if* $p^{-1}[\{c-1\}] = p^{-1}[\{q_{\mathsf{init}}\}] = \emptyset$ *and for all* $i \in [\![c, e]\!]$*, there is some* $o \in \mathsf{O}$ *such that* $p(o) = i$*. The* size *of a relevant environment* E *is equal to* $\mathsf{Sz}(E) := e - c$*.*

The benefit of relevant environments appears below: a game form is positionally optimizable if and only if there are optimal GF-strategies w.r.t. all *relevant* environments.

▶ **Proposition 47.** *Consider a set of outcomes* O *and a game form* $\mathcal{F} \in \mathsf{Form}(\mathsf{O})$*. For all* $n \in \mathbb{N}$*, the game form* $\mathcal{F}$ *is* $\mathsf{ParO}(n)$ *if and only if, for all relevant environments* E *with* $\mathsf{Sz}(E) \leq n-1$*, for both players, there is an optimal* GF*-strategy w.r.t.* $(\mathsf{O}, \mathcal{F}, E)$*.*

Some positionally optimizable game forms. As mentioned above, given the distance to the highest color in the game, Theorem 25 and Remark 19 give exactly the game forms that should be used in parity games to ensure the existence of positional optimal strategies for both players. The game forms in ParO are the ones that can be used in all parity games, regardless of the number of colors involved, while ensuring the existence of positional optimal strategies (for both players). Such game forms do exist, e.g. turn-based game forms are positionally optimizable (straightforwardly). In fact, all *determined* [1] game forms are positionally optimizable. Furthermore, all game forms with at most two outcomes are in ParO.

▶ **Proposition 48.** *Consider a set of outcomes* O *and a game form* $\mathcal{F} = \langle \mathsf{Act}_\mathsf{A}, \mathsf{Act}_\mathsf{B}, \mathsf{O}, \varrho\rangle \in \mathsf{Form}(\mathsf{O})$*. Assume that* $|\mathsf{O}| \leq 2$ *or that* $\mathcal{F}$ *is turn-based or that* $\mathcal{F}$ *is determined, i.e. such that:*

- *for all* $(a, b) \in \mathsf{Act}_\mathsf{A} \times \mathsf{Act}_\mathsf{B}$*, we have* $\varrho(a, b)$ *deterministic;*
- *for all* $v : \mathsf{O} \to \{0, 1\}$*, there is either some* $a \in \mathsf{Act}_\mathsf{A}$ *such that* $\varrho(a, \mathsf{Act}_\mathsf{B}) \subseteq \{1\}$ *or there is either some* $b \in \mathsf{Act}_\mathsf{B}$ *such that* $\varrho(\mathsf{Act}_\mathsf{A}, b) \subseteq \{0\}$*.*

Then, $\mathcal{F}$ *is positionally optimizable.*

Decidability. Similarly to "maximizable" game forms designed for reachability games [2] and to "maximizable" game forms designed (co-)Büchi games [3], it is rather easy to get convinced that positionally optimizable game forms used in this paper can be defined in the first-order theory of the reals (FO-$\mathbb{R}$): the characterizations of Lemma 21 and the fact that it is sufficient to only consider relevant environment, as stated in Proposition 47, even allow us to place the $\mathsf{ParO}(n)$ (for all n) and the ParO problems in the $\forall\exists$-fragment of FO-$\mathbb{R}$.

▶ **Proposition 49.** *For all* $n \in \mathbb{N}$*, the problem* $\mathsf{IsO}(n)$ *is decidable. And so is* IsO*.*

Hierarchy. For all $n \in \mathbb{N}$, the game forms in $\mathsf{ParO}(n)$ are the ones to be used – to ensure the existence of positional optimal strategies for both players – in parity games at states where the gap between the color of the state and the maximum color in the game is at most n. Straightforwardly, $\mathsf{ParO}(n) \subseteq \mathsf{ParO}(n+1)$. In fact, this inclusion is strict for all $n \in \mathbb{N}$. This defines an infinite hierarchy of game forms.

▶ **Proposition 50.** *For all* $n \in \mathbb{N}$*, we have* $\mathsf{ParO}(n) \subsetneq \mathsf{ParO}(n+1)$*.*

References

1 Benjamin Bordais, Patricia Bouyer, and Stéphane Le Roux. From local to global determinacy in concurrent graph games. In Mikolaj Bojanczyk and Chandra Chekuri, editors, *41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2021, December 15-17, 2021, Virtual Conference*, volume 213 of *LIPIcs*, pages 41:1–41:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.FSTTCS.2021.41.

2 Benjamin Bordais, Patricia Bouyer, and Stéphane Le Roux. Optimal strategies in concurrent reachability games. In Florin Manea and Alex Simpson, editors, *30th EACSL Annual Conference on Computer Science Logic, CSL 2022, February 14-19, 2022, Göttingen, Germany (Virtual Conference)*, volume 216 of *LIPIcs*, pages 7:1–7:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.CSL.2022.7.

3 Benjamin Bordais, Patricia Bouyer, and Stéphane Le Roux. Playing (almost-)optimally in concurrent Büchi and co-Büchi games. In Anuj Dawar and Venkatesan Guruswami, editors, *42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2022, December 18-20, 2022, IIT Madras, Chennai, India*, volume 250 of *LIPIcs*, pages 33:1–33:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.FSTTCS.2022.33.

4 Benjamin Bordais, Patricia Bouyer, and Stéphane Le Roux. From local to global optimality in concurrent parity games. *CoRR*, abs/2311.14373, 2023. doi:10.48550/arXiv.2311.14373.

5 Benjamin Bordais, Patricia Bouyer, and Stéphane Le Roux. Subgame optimal strategies in finite concurrent games with prefix-independent objectives. In Orna Kupferman and Pawel Sobocinski, editors, *Foundations of Software Science and Computation Structures – 26th International Conference, FoSSaCS 2023, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2023, Paris, France, April 22-27, 2023, Proceedings*, volume 13992 of *Lecture Notes in Computer Science*, pages 541–560. Springer, 2023. doi:10.1007/978-3-031-30829-1_26.

6 Krishnendu Chatterjee, Luca de Alfaro, and Thomas A. Henzinger. The complexity of quantitative concurrent parity games. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006*, pages 678–687. ACM Press, 2006. URL: http://dl.acm.org/citation.cfm?id=1109557.1109631.

7 Krishnendu Chatterjee, Marcin Jurdzinski, and Thomas A. Henzinger. Quantitative stochastic parity games. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004, New Orleans, Louisiana, USA, January 11-14, 2004*, pages 121–130. SIAM, 2004. URL: http://dl.acm.org/citation.cfm?id=982792.982808.

8 Luca de Alfaro and Thomas A. Henzinger. Concurrent omega-regular games. In *15th Annual IEEE Symposium on Logic in Computer Science, Santa Barbara, California, USA, June 26-29, 2000*, pages 141–154. IEEE Computer Society, 2000. doi:10.1109/LICS.2000.855763.

9 Luca de Alfaro and Rupak Majumdar. Quantitative solution of omega-regular games. *Journal of Computer and System Sciences*, 68:374–397, 2004.

10 Hugh Everett. Recursive games. *Annals of Mathematics Studies – Contributions to the Theory of Games*, 3:67–78, 1957.

11 Panganamala R Kumar and Tzong-Huei Shiau. Existence of value and randomized strategies in zero-sum discrete-time stochastic dynamic games. *SIAM Journal on Control and Optimization*, 19(5):617–634, 1981.

12 Marta Kwiatkowska, Gethin Norman, David Parker, Gabriel Santos, and Rui Yan. Probabilistic model checking for strategic equilibria-based decision making. In *Proc. 47th International Symposium on Mathematical Foundations of Computer Science (MFCS'22)*, LIPIcs. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.MFCS.2022.4.

13 Donald A. Martin. The determinacy of blackwell games. *The Journal of Symbolic Logic*, 63(4):1565–1581, 1998.

14 John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton Univ. Press, Princeton, 1944.

15 Wieslaw Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1–2):135–183, 1998.

16 Wiesław Zielonka. Perfect-information stochastic parity games. In *Proc. 7th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'04)*, volume 2987 of *Lecture Notes in Computer Science*, pages 499–513. Springer, 2004.

Ehrenfeucht–Fraïssé Games in Semiring Semantics

Sophie Brinke
RWTH Aachen University, Germany

Erich Grädel
RWTH Aachen University, Germany

Lovro Mrkonjić
RWTH Aachen University, Germany

Abstract

Ehrenfeucht–Fraïssé games provide a fundamental method for proving elementary equivalence (and equivalence up to a certain quantifier rank) of relational structures. We investigate the soundness and completeness of this method in the more general context of semiring semantics. Motivated originally by provenance analysis of database queries, semiring semantics evaluates logical statements not just by true or false, but by values in some commutative semiring; this can provide much more detailed information, for instance concerning the combinations of atomic facts that imply the truth of a statement, or practical information about evaluation costs, confidence scores, access levels or the number of successful evaluation strategies. There is a wide variety of different semirings that are relevant for provenance analysis, and the applicability of classical logical methods in semiring semantics may strongly depend on the algebraic properties of the underlying semiring.

While Ehrenfeucht–Fraïssé games are sound and complete for logical equivalences in classical semantics, and thus on the Boolean semiring, this is in general not the case for other semirings. We provide a detailed analysis of the soundness and completeness of model comparison games on specific semirings, not just for classical Ehrenfeucht–Fraïssé games but also for other variants based on bijections or counting.

Finally we propose a new kind of games, called *homomorphism games*, based on the fact that there exist locally very different semiring interpretations that can be proved to be elementarily equivalent via separating sets of homomorphisms. We prove that these homomorphism games provide a sound and complete method for logical equivalences on finite lattice semirings.

2012 ACM Subject Classification Theory of Computation → Finite Model Theory

Keywords and phrases Semiring semantics, elementary equivalence, Ehrenfeucht–Fraïssé games

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.19

Related Version *Full Version*: `https://arxiv.org/abs/2308.04910` [4]

1 Introduction

Semiring provenance was proposed in 2007 in a seminal paper by Green, Karvounarakis, and Tannen [15]. It is based on the idea to annotate the atomic facts in a database by values in some commutative semiring, and to propagate these values through a database query, keeping track whether information is used alternatively (as in disjunctions or existential quantifications) or jointly (as in conjunctions or universal quantifications). Depending on the chosen semiring, the provenance valuation then gives practical information about a query, beyond its truth or falsity, for instance concerning the *confidence* that we may have in its truth, the *cost* of its evaluation, the number of successful evaluation strategies, and so on. Beyond such provenance evaluations in specific *application semirings*, more precise information is obtained by evaluations in *provenance semirings* of polynomials, which permit us to *track* which atomic facts are used (and how often) to compute the answer to the query.

In databases, semiring provenance has been successfully applied to a number of different scenarios, such as conjunctive queries, positive relational algebra, datalog, nested relations, XML, SQL-aggregates, graph databases (see, e.g., the surveys [9, 16]), but for a long time, it had essentially been restricted to negation-free query languages; there has been no systematic tracking of *negative or absent information*, and for quite some time, this has been an obstacle for extending semiring provenance to other branches of logic in computer science.

A new approach to provenance analysis for languages with negation has been proposed in 2017 by Grädel and Tannen [13], based on transformations into negation normal form, quotient semirings of polynomials with dual indeterminates, and a close relationship to semiring valuations of games [14]. Since then, semiring provenance has been extended to a systematic investigation of *semiring semantics* for many logical systems, including first-order logic, modal logic, description logics, guarded logic and fixed-point logic [3, 5, 6, 7, 13] and also to a general method for strategy analysis in games [11, 14].

In classical semantics, a model $\mathfrak{A}$ of a formula assigns a Boolean value to each literal. $\mathcal{S}$-interpretations π, for a suitable semiring $\mathcal{S}$, generalise this by assigning a semiring value from $\mathcal{S}$ to each literal. We interpret a value of 0 as *false* and all other semiring values as *nuances of true*, or *true with additional information*. In this context, classical semantics corresponds to semiring semantics on the Boolean semiring $\mathbb{B} = (\{0,1\}, \vee, \wedge, 0, 1)$, the Viterbi semiring $\mathbb{V} = ([0,1], \max, \cdot, 0, 1)$ can model *confidence* scores, the tropical semiring $\mathbb{T} = (\mathbb{R}_+^\infty, \min, +, \infty, 0)$ is used for cost analysis, and min-max-semirings $(A, \max, \min, a, b)$ for a totally ordered set $(A, <)$ can model different access levels. Other interesting semirings are the Łukasiewicz semiring $\mathbb{L}$, used in many-valued logic, and its dual $\mathbb{D}$, which we call the semiring of doubt. Provenance semirings of polynomials, such as $\mathbb{N}[X]$, *track* certain literals by mapping them to different indeterminates. The overall value of a formula is then a polynomial that describes precisely what combinations of literals imply the truth of the formula. There are other provenance semirings, obtained from $\mathbb{N}[X]$ by dropping coefficients and/or exponents or by absorption, to get semirings $\mathbb{B}[X], \mathrm{Trio}[X], \mathbb{W}[X], \mathbb{S}[X]$ and $\mathrm{PosBool}[X]$. They are less informative than $\mathbb{N}[X]$ (which is the free semiring generated by X), but have specific algebraic properties and admit simpler evaluation procedures. For applications to infinite universes, and for stronger logics than first-order logic, provenance semirings with more general objects than polynomials are needed, such as $\mathbb{N}^\infty[\![X]\!]$, the semirings of formal power series, and $\mathbb{S}^\infty[X|$, the semirings of generalised absorptive polynomials with potentially infinite exponents, which are fundamental for semiring semantics of fixed-point logics [7, 14].

The development of semiring semantics raises the question to what extent classical techniques and results of logic extend to semiring semantics, and how this depends on the algebraic properties of the underlying semiring, and this paper is part of a general research programme that explores such questions. In previous investigations, we have studied, for instance, the relationship between elementary equivalence and isomorphism for finite semiring interpretations and their definability up to isomorphism [12], 0-1 laws [10], and locality properties as given by the theorems of Gaifman and Hanf [2]. In all these studies, it has turned out that classical methods of mathematical logic can be extended to semiring semantics for certain semirings, but that they fail for others. Further, these questions are often surprisingly difficult: even quite simple facts of logic in the standard Boolean semantics become interesting research problems for semirings, and they often require completely new methods.

The objective of this paper is to study the applicability of Ehrenfeucht–Fraïssé games – and related model comparison games – as a method for proving elementary equivalence (i.e. indistinguishability by first-order sentences, denoted $\equiv$) and m-equivalence (i.e. indistinguishability by sentences of quantifier rank up to m, denoted $\equiv_m$) in semiring semantics.

Let us recall the classical Ehrenfeucht–Fraïssé Theorem[1] (see e.g. [8]).

▶ **Theorem 1** (Ehrenfeucht–Fraïssé). *Let τ be a finite relational vocabulary. For any two τ-structures $\mathfrak{A}$ and $\mathfrak{B}$, and for all $m \in \mathbb{N}$, the following statements are equivalent:*

(1) $\mathfrak{A} \equiv_m \mathfrak{B}$*;*

(2) *Player II (Duplicator) has a winning strategy for the game $G_m(\mathfrak{A}, \mathfrak{B})$;*

(3) *There exists an m-back-and-forth system $(I_j)_{j \leq m}$ for $\mathfrak{A}$ and $\mathfrak{B}$;*

(4) $\mathfrak{B} \models \chi^m_{\mathfrak{A}}$*, where $\chi^m_{\mathfrak{A}}$ is the characteristic sentence of quantifier rank m for $\mathfrak{A}$.*

In semiring semantics, the structures $\mathfrak{A}$ and $\mathfrak{B}$ are generalised to (model-defining) semiring interpretations π_A and π_B mapping instantiated τ-literals into a semiring $\mathcal{S}$. The notions of m-equivalence, local isomorphisms, Ehrenfeucht–Fraïssé games, and back-and-forth systems all generalise in a straightforward way to $\mathcal{S}$-interpretations, for any semiring $\mathcal{S}$ (see Section 2). Also the observation that m-back-and-forth systems can be viewed as algebraic descriptions of winning strategies of Player II in m-turn Ehrenfeucht–Fraïssé games holds for arbitrary semiring interpretations, i.e. the equivalence (2) $\Leftrightarrow$ (3) holds for any semiring. The notion of characteristic sentences will be discussed later in Section 5. Our main concern is the relationship between (1) and (2), or equivalently (1) and (3). We shall have to consider both directions separately.

▶ **Definition 2.** Let $\mathcal{S}$ be an arbitrary commutative semiring. We say that

(1) *G_m is sound for $\equiv_m$ on $\mathcal{S}$* if for any pair π_A, π_B of model-defining $\mathcal{S}$-interpretations, the existence of a winning strategy of Player II for $G_m(\pi_A, \pi_B)$ implies that $\pi_A \equiv_m \pi_B$;

(2) *G_m is complete for $\equiv_m$ on $\mathcal{S}$* if for any pair π_A, π_B of model-defining $\mathcal{S}$-interpretations such that $\pi_A \equiv_m \pi_B$, Player II has a winning strategy for $G_m(\pi_A, \pi_B)$.

In this terminology, the Ehrenfeucht–Fraïssé Theorem says that for every m, G_m is both sound and complete for $\equiv_m$ on the Boolean semiring. However, we shall prove that the Boolean semiring is the only semiring with this property, and for general semirings, the games G_m need be neither sound nor complete. But there are also positive results, and the detailed study of soundness and completeness of Ehrenfeucht–Fraïssé games on semirings is quite interesting and diverse. For instance, we shall prove that G_m is sound for $\equiv_m$ precisely on *fully idempotent* semirings (where both semiring operations are idempotent). Examples of fully idempotent semirings include all min-max semirings, lattice semirings, and the semirings PosBool$[X]$ of irredundant positive Boolean DNF-formulae. We shall then turn to more powerful games, which are more difficult to win for Duplicator, but if she wins, stronger results follow. In particular, we study the general Ehrenfeucht–Fraïssé game $G(\pi_A, \pi_B)$ where Spoiler can choose a number m, and then the game $G_m(\pi_A, \pi_B)$ is played. If, on a semiring $\mathcal{S}$, G_m is sound for $\equiv_m$ for all m, then a winning strategy for Duplicator for $G(\pi_A, \pi_B)$ implies that $\pi_A \equiv_m \pi_B$ for all m, and hence $\pi_A \equiv \pi_B$. Thus, soundness of all games G_m implies soundness of G. The converse is not true; there are semirings on which G is sound for $\equiv$, although the games G_m are unsound for $\equiv_m$. Trivially, G is sound on semirings that do not admit interpretations with infinite universes due to the impossibility of infinite sums or products, such as $\mathbb{N}$ or the provenance semirings $\mathbb{B}[X], \mathbb{S}[X]$ and $\mathbb{N}[X]$. More interesting cases include semirings that are not idempotent, but where adding or multiplying any element repeatedly with itself stabilises after at most n steps, or the semiring $\mathbb{N}^\infty = \mathbb{N} \cup \{\infty\}$. But there also exist a number of semirings on which the unrestricted

[1] Detailed definitions of all notions will be given in Section 2.

Ehrenfeucht–Fraïssé game G is unsound for elementary equivalence, including the semirings $\mathbb{T}, \mathbb{V}, \mathbb{L}$ and $\mathbb{D}$. Further we shall consider *bijection* and *counting games*, which are variants of pebble games for bounded-variable logics with counting from [17, 18]. Actually the m-move bijection games BG_m and counting games CG_m are equivalent, and they turn out to be sound for $\equiv_m$ on *every* semiring. However, with few exceptions, such as the semirings $\mathbb{N}$ and $\mathbb{N}[X]$, they are not complete. We also study parametrised versions CG^n_m of counting games.

On many semirings $\mathcal{S}$, the methods established in [12] permit us to construct elementarily equivalent $\mathcal{S}$-interpretations $\pi_A \equiv \pi_B$, although locally some elements of π_A look different from all elements of π_B, so that Spoiler wins $G_m(\pi_A, \pi_B)$ for some small m. The game G_m is then incomplete for $\equiv_m$, and the game G is incomplete for $\equiv$. Since the games CG^n_m and BG_m are more difficult to win for Player II than G_m, they are incomplete as well. This approach successfully works for the semirings $\mathbb{V}, \mathbb{T}, \mathbb{L}, \mathbb{D}, \mathbb{N}^\infty, \mathbb{W}[X], \mathbb{S}[X], \mathbb{B}[X]$, and $\mathbb{S}^\infty[X]$.

The soundness and completeness results of these games are summarised in Figure 1.

Application semirings:		$\mathcal{S} \not\cong \mathbb{B}$ fully idempotent	$\mathbb{T} \cong \mathbb{V}$	$\mathbb{L} \cong \mathbb{D}$	$\mathbb{N}$	$\mathbb{N}^\infty$
Soundness	G_m for $\equiv_m$	✓	✗	✗	✗	✗
	CG^n_m for $\equiv_m$	✓	✗	✗	✗	✗
	BG_m for $\equiv_m$	✓	✓	✓	✓	✓
	G for $\equiv$	✓	✗	✗	✓	✓
Completeness	G_m for $\equiv_m$	✗	✗	✗	✓	✗
	CG^n_m for $\equiv_m$	✗	✗	✗	✓	✗
	BG_m for $\equiv_m$	✗	✗	✗	✓	✗
	G for $\equiv$	✗	✗	✗	✓	✗

Provenance semirings:		PosBool$[X]$	$\mathbb{W}[X]$	$\mathbb{S}[X], \mathbb{B}[X]$	$\mathbb{N}[X]$	$\mathbb{S}^\infty[X]$
Soundness	G_m for $\equiv_m$	✓	✗	✗	✗	✗
	CG^n_m for $\equiv_m$	✓	✓	✗	✗	✗
	BG_m for $\equiv_m$	✓	✓	✓	✓	✓
	G for $\equiv$	✓	✓	✓	✓	✓
Completeness	G_m for $\equiv_m$	✗	✗	✗	✓	✗
	CG^n_m for $\equiv_m$	✗	✗	✗	✓	✗
	BG_m for $\equiv_m$	✗	✗	✗	✓	✗
	G for $\equiv$	✗	✗	✗	✓	✗

Figure 1 Due to full idempotence. Due to n-idempotence. Holds for any semiring. Follows from the finiteness of the universes. Cannot hold since elementary equivalence of finite interpretations does not imply isomorphism.

The proof that locally different $\mathcal{S}$-interpretations are nevertheless elementarily equivalent often proceeds via separating sets of homomorphisms. We use this method to propose a new kind of games, called *homomorphism games*, involving the selection of a homomorphism into the Boolean semiring, and a one-sided winning condition, due to the property that homomorphisms may transfer model-defining $\mathcal{S}$-interpretations into $\mathbb{B}$-interpretations that are no longer model-defining. We prove that these homomorphism games provide a sound and complete method for proving logical equivalences on finite lattice semirings.

2 Semiring semantics

We briefly summarise semiring semantics for first-order logic, as introduced in [13], and the resulting generalised notions of isomorphism and equivalence.

▶ **Definition 3** (Semiring). A *commutative semiring* is an algebraic structure $\mathcal{S} = (S, +, \cdot, 0, 1)$ with $0 \neq 1$, such that $(S, +, 0)$ and $(S, \cdot, 1)$ are commutative monoids, $\cdot$ distributes over $+$, and $0 \cdot s = s \cdot 0 = 0$.

A commutative semiring is *naturally ordered* (by addition) if $s \leq t :\Leftrightarrow \exists r(s + r = t)$ defines a partial order. In particular, this excludes rings. We only consider commutative and naturally ordered semirings and simply refer to them as *semirings*. A semiring $\mathcal{S}$ is *idempotent* if $s + s = s$ for each $s \in S$ and *multiplicatively idempotent* if $s \cdot s = s$ for all $s \in S$. If both properties are satisfied, we say that $\mathcal{S}$ is *fully idempotent*. Finally, $\mathcal{S}$ is *absorptive* if $s + st = s$ for all $s, t \in S$ or, equivalently, if multiplication is decreasing in $\mathcal{S}$, i.e. $st \leq s$ for $s, t \in S$ (equivalence is shown in [7]). Every absorptive semiring is idempotent.

Application semirings. There are several applications which can be modelled by semirings and provide useful practical information about the evaluation of a formula.

- A totally ordered set $(S, \leq)$ with least element s and greatest element t induces the *min-max semiring* $(S, \max, \min, s, t)$. It can be used to reason about access levels.
- The *tropical semiring* $\mathbb{T} = (\mathbb{R}^{\infty}_{+}, \min, +, \infty, 0)$ provides the opportunity to annotate basic facts with a cost which has to be paid for accessing them and realise a cost analysis.
- The *Viterbi semiring* $\mathbb{V} = ([0, 1]_{\mathbb{R}}, \max, \cdot, 0, 1)$, which is in fact isomorphic to $\mathbb{T}$ via $y \mapsto -\ln y$, can be used for reasoning about confidence.
- An alternative semiring for this is the *Łukasiewicz semiring* $\mathbb{L} = ([0, 1]_{\mathbb{R}}, \max, \odot, 0, 1)$, where multiplication is given by $s \odot t = \max(s + t - 1, 0)$. It is isomorphic to the semiring of doubt $\mathbb{D} = ([0, 1]_{\mathbb{R}}, \min, \oplus, 1, 0)$ with $s \oplus t = \min(s + t, 1)$.
- The *natural semiring* $\mathbb{N} = (\mathbb{N}, +, \cdot, 0, 1)$ is used to count the number of evaluation strategies proving that a sentence is satisfied. It is also important for bag semantics in databases.

Provenance semirings. Provenance semirings of polynomials provide information on which combinations of literals imply the truth of a formula. The universal provenance semiring is the semiring $\mathbb{N}[X]$ of multivariate polynomials with indeterminates from X and coefficients from $\mathbb{N}$. Other provenance semirings are obtained as quotient semirings of $\mathbb{N}[X]$ induced by congruences for (full) idempotence and absorption. The resulting provenance values are less informative, but their computation is more efficient.

- By dropping coefficients from $\mathbb{N}[X]$, we get the free idempotent semiring $\mathbb{B}[X]$ whose elements are finite sets of monomials. It is the quotient induced by $x + x \sim x$.
- If, in addition, exponents are dropped, we obtain the Why-semiring $\mathbb{W}[X]$ of finite sums of monomials that are linear in each argument.
- The free absorptive semiring $\mathbb{S}[X]$ consists of $0, 1$ and all antichains of monomials with respect to the absorption order $\succcurlyeq$. A monomial m_1 absorbs m_2, denoted $m_1 \succcurlyeq m_2$, if it has smaller exponents, i.e. $m_2 = m \cdot m_1$ for some monomial m.
- Finally, the lattice semiring PosBool$[X]$ freely generated by the set X arises from $\mathbb{S}[X]$ by collapsing exponents.

For a given finite relational vocabulary τ, we denote by $\mathrm{Lit}_n(\tau)$ the set of literals $R\bar{x}$ and $\neg R\bar{x}$ where $R \in \tau$ and $\bar{x}$ is a tuple of variables from $\{x_1, \ldots, x_n\}$. The set $\mathrm{Lit}_A(\tau)$ refers to literals $R\bar{a}$ and $\neg R\bar{a}$ that are instantiated with elements from a universe A.

▶ **Definition 4** ($\mathcal{S}$-interpretation). Given a semiring $\mathcal{S}$, a mapping $\pi\colon \mathrm{Lit}_A(\tau) \to \mathcal{S}$ is an *$\mathcal{S}$-interpretation (of vocabulary τ and universe A)*. We say that $\mathcal{S}$ is *model-defining* if exactly one of the values $\pi(L)$ and $\pi(\overline{L})$ is zero for any pair of complementary literals $L, \overline{L} \in \mathrm{Lit}_A(\tau)$.

An $\mathcal{S}$-interpretation $\pi\colon \mathrm{Lit}_A(\tau) \to \mathcal{S}$ inductively extends to valuations $\pi[\![\varphi(\bar{a})]\!]$ of instantiated first-order formulae $\varphi(\bar{x})$ in negation normal form. Equalities are interpreted by their truth value, that is $\pi[\![a = a]\!] \coloneqq 1$ and $\pi[\![a = b]\!] \coloneqq 0$ for $a \neq b$ (and analogously for inequalities). Based on that, the semantics of disjunctions and existential quantifiers is defined via sums, while conjunctions and universal quantifiers are interpreted as products.

$$\pi[\![\psi(\bar{a}) \vee \vartheta(\bar{a})]\!] \coloneqq \pi[\![\psi(\bar{a})]\!] + \pi[\![\vartheta(\bar{a})]\!] \qquad \pi[\![\psi(\bar{a}) \wedge \vartheta(\bar{a})]\!] \coloneqq \pi[\![\psi(\bar{a})]\!] \cdot \pi[\![\vartheta(\bar{a})]\!]$$
$$\pi[\![\exists x \psi(\bar{a}, x)]\!] \coloneqq \sum_{a \in A} \pi[\![\psi(\bar{a}, a)]\!] \qquad \pi[\![\forall x \psi(\bar{a}, x)]\!] \coloneqq \prod_{a \in A} \pi[\![\psi(\bar{a}, a)]\!]$$

▶ **Lemma 5** (Fundamental Property). *Let $\pi\colon \mathrm{Lit}_A(\tau) \to \mathcal{S}$ be an $\mathcal{S}$-interpretation and $h\colon \mathcal{S} \to \mathcal{T}$ be a semiring homomorphism. Then, $(h \circ \pi)$ is a $\mathcal{T}$-interpretation and it holds that $h(\pi[\![\varphi(\bar{a})]\!]) = (h \circ \pi)[\![\varphi(\bar{a})]\!]$ for all $\varphi(\bar{x}) \in \mathrm{FO}(\tau)$ and instantiations $\bar{a} \subseteq A$.*

Basic model theoretic concepts such as equivalence and isomorphism naturally generalise to semiring semantics and yield more fine-grained notions. Given a mapping $\sigma\colon A \to B$ and some $L \in \mathrm{Lit}_A(\tau)$, we denote by $\sigma(L)$ the τ-literal over B which arises from L by replacing each occurrence of $a \in A$ with $\sigma(a) \in B$.

▶ **Definition 6** (Isomorphism). $\mathcal{S}$-interpretations $\pi_A\colon \mathrm{Lit}_A(\tau) \to \mathcal{S}$ and $\pi_B\colon \mathrm{Lit}_B(\tau) \to \mathcal{S}$ are *isomorphic*, denoted as $\pi_A \cong \pi_B$, if there is a bijection $\sigma\colon A \to B$ such that $\pi_A(L) = \pi_B(\sigma(L))$ for all $L \in \mathrm{Lit}_A(\tau)$. A mapping $\sigma\colon \bar{a} \mapsto \bar{b}$ is a *local isomorphism* between π_A and π_B if it is an isomorphism between the subinterpretations $\pi_A|_{\mathrm{Lit}_{\bar{a}}(\tau)}$ and $\pi_B|_{\mathrm{Lit}_{\bar{b}}(\tau)}$.

▶ **Definition 7** (Elementary equivalence). Two $\mathcal{S}$-interpretations $\pi_A\colon \mathrm{Lit}_A(\tau) \to \mathcal{S}$ and $\pi_B\colon \mathrm{Lit}_B(\tau) \to \mathcal{S}$ with elements $\bar{a} \in A^n$ and $\bar{b} \in B^n$ are *elementarily equivalent*, denoted $(\pi_A, \bar{a}) \equiv (\pi_B, \bar{b})$, if $\pi_A[\![\varphi(\bar{a})]\!] = \pi_B[\![\varphi(\bar{b})]\!]$ for all $\varphi(\bar{x}) \in \mathrm{FO}(\tau)$. They are *$m$-equivalent*, denoted $(\pi_A, \bar{a}) \equiv_m (\pi_B, \bar{b})$, if the above holds for all $\varphi(\bar{x})$ with $\mathrm{qr}(\varphi(\bar{x})) \leq m$ where $\mathrm{qr}(\varphi(\bar{x}))$ refers to the quantifier rank of $\varphi(\bar{x})$.

As in classical semantics, isomorphic $\mathcal{S}$-interpretations are elementarily equivalent. The converse concerning finite $\mathcal{S}$-interpretations, however, marks an important difference to Boolean semantics; it fails for a number of semirings, including all min-max semirings with at least three elements, while it still holds on other semirings such as $\mathbb{T}, \mathbb{V}, \mathbb{N}$ and $\mathbb{N}[X]$ (see [12]).

3 m-turn Ehrenfeucht–Fraïssé games

Given that the notion of local isomorphisms extends in a straightforward way from structures to semiring interpretations, we also obtain Ehrenfeucht–Fraïssé games $G_m(\pi_A, \pi_B)$ played on $\mathcal{S}$-interpretations π_A, π_B: In the i-th turn, Spoiler chooses some element $a_i \in A$ or $b_i \in B$, and Duplicator answers with an element in the other $\mathcal{S}$-interpretation; the play then continues with the subgame $G_{m-i}(\pi_A, a_1, \ldots, a_i, \pi_B, b_1, \ldots, b_i)$. After m moves, tuples $\bar{a} = (a_1, \ldots, a_m)$ in A and $\bar{b} = (b_1, \ldots, b_m)$ in B have been selected, and Duplicator wins the play if $\sigma\colon \bar{a} \mapsto \bar{b}$ is a local isomorphism.

However, while classical structures $\mathfrak{A}$ and $\mathfrak{B}$ are separated by a formula $\exists x\psi(x)$ or $\forall x\psi(x)$ if, and only if, there is some $a \in A$ (or $b \in B$) such that for all $b \in B$ (or $a \in A$, respectively) the formula $\psi(x)$ separates $(\mathfrak{A}, a)$ from $(\mathfrak{B}, b)$, neither of the implications translates to semiring semantics. This is illustrated by very simple semiring interpretations[2] and causes both soundness and completeness of $G_m(\pi_A, \pi_B)$ for $\equiv_m$ to fail in general.

$(\mathbb{N}, +, \cdot, 0, 1)$

π_A :

A	R	$\neg R$
a_1	1	0
a_2	1	0
a_3	2	0

π_B :

B	R	$\neg R$
b_1	1	0
b_2	2	0
b_3	2	0

$\pi_A[\![\exists x Rx]\!] = 4 \neq 5 = \pi_B[\![\exists x Rx]\!]$

$(\{0, 1, 2, 3, 4\}, \max, \min, 0, 4)$

π_A :

A	R	$\neg R$
a_1	1	0
a_2	2	0
a_3	4	0

π_B :

B	R	$\neg R$
b_1	1	0
b_2	3	0
b_3	4	0

$\pi_A[\![\exists x Rx]\!] = 4 = \pi_B[\![\exists x Rx]\!]$
$\pi_A[\![\forall x Rx]\!] = 1 = \pi_B[\![\forall x Rx]\!]$

This suggests that the direct adaptation of the game rules poses problems and raises the question on which semirings the game G_m is sound, and on which it is complete for $\equiv_m$. In particular, we aim to relate this to the algebraic properties of the underlying semiring.

3.1 Soundness of the games and counting in semirings

The fact that quantifiers in classical semantics do not capture counting is one of the central limitations of the expressive power of first-order logic. However, in semiring semantics, this is more complicated: Given a formula $\psi(x)$ and some $s \in \mathcal{S}$, the number of $a \in A$ such that $\pi[\![\psi(a)]\!] = s$ may affect both $\pi[\![\exists x\psi(x)]\!]$ and $\pi[\![\forall x\psi(x)]\!]$. Only in fully idempotent semirings unequal sums or products can be attributed to differing sets of summands or factors, which causes full idempotence to be a necessary and sufficient condition for the soundness of G_m.

▶ **Theorem 8.** *The games G_m are sound for $\equiv_m$ on a semiring $\mathcal{S}$ and all $m \in \mathbb{N}$ if, and only if, $\mathcal{S}$ is fully idempotent.*

Proof. ($\Leftarrow$): Suppose that $\mathcal{S}$ is fully idempotent. Based on a separating formula $\varphi(\bar{x}) \in \mathrm{FO}(\tau)$ with $\pi_A[\![\varphi(\bar{a})]\!] \neq \pi_B[\![\varphi(\bar{b})]\!]$ and $\mathrm{qr}(\varphi(\bar{x})) \leq m$ where $\bar{a} \in A^n$ and $\bar{b} \in B^n$, we construct a winning strategy for Spoiler in the game $G_m(\pi_A, \bar{a}, \pi_B, \bar{b})$ by induction. We only consider the cases $\varphi(\bar{x}) = Qx\psi(\bar{x}, x)$ with $Q \in \{\exists, \forall\}$ where $\mathrm{qr}(\varphi(\bar{x})) \leq m$. It holds that

$$\pi_A[\![\exists x\psi(\bar{a}, x)]\!] = \sum_{a\in A} \pi_A[\![\psi(\bar{a}, a)]\!] \neq \sum_{b\in B} \pi_B[\![\psi(\bar{b}, b)]\!] = \pi_B[\![\exists x\psi(\bar{b}, x)]\!] \text{ or}$$
$$\pi_A[\![\forall x\psi(\bar{a}, x)]\!] = \prod_{a\in A} \pi_A[\![\psi(\bar{a}, a)]\!] \neq \prod_{b\in B} \pi_B[\![\psi(\bar{b}, b)]\!] = \pi_B[\![\forall x\psi(\bar{b}, x)]\!].$$

Both cases imply $\{\pi_A[\![\psi(\bar{a}, a)]\!] \colon a \in A\} \neq \{\pi_B[\![\psi(\bar{b}, b)]\!] \colon b \in B\}$ due to full idempotence. Spoiler wins the game $G_m(\pi_A, \bar{a}, \pi_B, \bar{b})$ by choosing some element $a \in A$ or $b \in B$ witnessing this inequality. For all possible answers $b \in B$ or $a \in A$, respectively, it holds that $\pi_A[\![\psi(\bar{a}, a)]\!] \neq \pi_B[\![\psi(\bar{b}, b)]\!]$. Applying the induction hypothesis yields that Spoiler has a winning strategy for the remaining game $G_{m-1}(\pi_A, \bar{a}, a, \pi_B, \bar{b}, b)$ as $\mathrm{qr}(\psi(\bar{x}, x)) \leq m - 1$.

[2] We describe semiring interpretations over a monadic vocabulary by tables, whose rows are indexed by elements of the universe, and columns by the predicate symbols and their negations, such that the entry for row a and column P has the semiring value of the literal Pa.

(⇒): If $\mathcal{S}$ is not fully idempotent, there is some $s \in \mathcal{S}$ such that $s + s \neq s$ or $s \cdot s \neq s$. Clearly, Duplicator wins $G_1(\pi_A, \pi_B)$ on the following $\mathcal{S}$-interpretations, while $\pi_A \not\equiv_1 \pi_B$ due to $\pi_A[\![\exists x Rx]\!] = s + s \neq s = \pi_B[\![\exists x Rx]\!]$ or $\pi_A[\![\forall x Rx]\!] = s \cdot s \neq s = \pi_B[\![\forall x Rx]\!]$.

π_A :

A	R	$\neg R$
a_1	s	0
a_2	s	0

π_B :

B	R	$\neg R$
b	s	0

◀

This result motivates the consideration of more powerful games such as m-turn bijection games, a variant of the pebble games which, on finite classical structures, characterise m-equivalence in FO with counting quantifiers [17].

▶ **Definition 9.** The game $BG_m(\pi_A, \bar{a}, \pi_B, \bar{b})$ uses the same positions and winning condition as $G_m(\pi_A, \bar{a}, \pi_B, \bar{b})$, but in each round Duplicator has to provide a bijection $h\colon A \to B$. If such a bijection does not exist, i.e. $|A| \neq |B|$, Spoiler wins immediately. Otherwise, Spoiler chooses some $a \in A$ and the pair $(a, h(a))$ is added to the current position.

In contrast to the classical Ehrenfeucht–Fraïssé game, this modification ensures soundness without requiring full idempotence of the underlying semiring.

▶ **Theorem 10.** *For every $m \in \mathbb{N}$, the game BG_m is sound for $\equiv_m$ on every semiring $\mathcal{S}$.*

Proof. Suppose that $\varphi(\bar{x}) = Qx\psi(\bar{x}, x)$ with $Q \in \{\exists, \forall\}$ and $\mathrm{qr}(\varphi(\bar{x})) = m$ separates $(\pi_A, \bar{a})$ from $(\pi_B, \bar{b})$. For any bijection $h\colon A \to B$ Duplicator may choose in the game $BG_m(\pi_A, \bar{a}, \pi_B, \bar{b})$, there must be some $a_h \in A$ such that $\pi_A[\![\psi(\bar{a}, a_h)]\!] \neq \pi_B[\![\psi(\bar{b}, h(a_h))]\!]$ since otherwise $\sum_{a\in A} \pi_A[\![\psi(\bar{a}, a)]\!] = \sum_{a\in A} \pi_B[\![\psi(\bar{b}, h(a))]\!] = \sum_{b\in B} \pi_B[\![\psi(\bar{b}, b)]\!]$ and analogously for products. By choosing a_h, Spoiler wins the game by induction. ◀

While demanding a bijection from Duplicator does ensure the soundness of BG_m, it is often at the expense of completeness. This is due to the fact that different multiplicities of a semiring value in two interpretations do not necessarily imply separability by a first-order sentence. In particular, this is the case for fully idempotent semirings, on which the games G_m are already sound, but the resulting issues concern other semirings as well. We illustrate this on the semiring $\mathbb{W}[x, y]$, where the precise numbers of occurrences of single semiring values may differ in their effect on the separability of the resulting interpretations, as shown below.

A	R	$\neg R$
a_1	$x + y$	0

$\not\equiv_1$

B	R	$\neg R$
b_1	$x + y$	0
b_2	$x + y$	0

$\equiv_1$

C	R	$\neg R$
c_1	$x + y$	0
c_2	$x + y$	0
c_3	$x + y$	0

We observe that the semirings $\mathbb{W}[X]$, while not being fully idempotent, for instance due to $(x + y)(x + y) = x + xy + y$, satisfy a weaker idempotence condition.

▶ **Definition 11.** Let $n \in \mathbb{N}$. A semiring $\mathcal{S}$ is *n-idempotent* if $\sum_{i\in I} s = \sum_{j\in J} s$ and $\prod_{i\in I} s = \prod_{j\in J} s$ for all $s \in \mathcal{S}$ and all index sets I, J such that $|I| \geq n$ and $|J| \geq n$.

It can easily be verified that $\mathbb{W}[X]$ is $|X|$-idempotent as monomials can be seen as sets of variables, and multiplication corresponds to their union. For such semirings, we want to replace the requirement for Duplicator to provide a bijection by a weaker requirement that still maintains soundness. For this, we use counting games, introduced by Immermann and Lander [18], which are equivalent to bijection games, but admit a parametrisation by the size of the sets that are picked in each turn.

▶ **Definition 12.** Let $n \in \mathbb{N}$. In each turn of the game $CG^n_m(\pi_A, \bar{a}, \pi_B, \bar{b})$, Spoiler chooses a set $X \subseteq A$ or $X \subseteq B$ with $|X| \leq n$ and Duplicator has to react with a subset Y of the other universe such that $|X| = |Y|$. Afterwards, Spoiler picks some $y \in Y$, Duplicator must respond with some element $x \in X$ and the pair (x, y), or (y, x), is added to the current position. As before, the winning condition is given by local isomorphism.

Note that the game CG^1_m corresponds to the classical Ehrenfeucht–Fraïssé game G_m and 1-idempotence coincides with full idempotence. Theorem 8 can be generalised as follows.

▶ **Theorem 13.** *The game CG^n_m is sound for $\equiv_m$ exactly on n-idempotent semirings $\mathcal{S}$.*

3.2 Completeness and incompleteness

As opposed to a Boolean quantifier or a move in an Ehrenfeucht–Fraïssé game, a quantifier in semiring semantics does not pick out a specific element of the universe. Instead, it induces a sum or product over all elements. As a consequence, completeness of the m-turn Ehrenfeucht–Fraïssé game, and thus also completeness of other variants of model comparison games, fail in general. In particular, this applies to semirings on which elementary equivalence of finite interpretations does not imply isomorphism. Indeed, on any pair of finite non-isomorphic semiring interpretations, Spoiler wins G_m for sufficiently large m by picking all elements in the larger universe, or in any universe if both have the same cardinality. A particular example, presented in [12], of non-isomorphic but elementarily equivalent $\mathcal{S}$-interpretations $\pi^{s,t}_A$ and $\pi^{s,t}_B$ for arbitrary elements s, t of a fully idempotent semiring $\mathcal{S}$ is the following:

$\pi^{s,t}_A$:

A	R_1	R_2	$\neg R_1$	$\neg R_2$
a_1	0	t	s	0
a_2	s	0	0	t
a_3	t	s	0	0
a_4	0	0	t	s

$\pi^{s,t}_B$:

B	R_1	R_2	$\neg R_1$	$\neg R_2$
b_1	t	0	0	s
b_2	0	s	t	0
b_3	s	t	0	0
b_4	0	0	s	t

For any $s, t \in \mathcal{S}$, we have that $\pi^{s,t}_A \equiv \pi^{s,t}_B$ [12, Theorem 13], but obviously, Spoiler even wins the game $G_1(\pi^{s,t}_A, \pi^{s,t}_B)$ for distinct and non-zero values $s, t \in \mathcal{S}$. Thus, completeness of G_m for $\equiv_m$ and full idempotence are mutually exclusive on semirings with at least three elements, while soundness requires full idempotence, which entails the following result.

▶ **Theorem 14.** *If, for all $m \in \mathbb{N}$, the game G_m is sound and complete for $\equiv_m$ on $\mathcal{S}$, then $\mathcal{S}$ is isomorphic to $\mathbb{B}$.*

Several further semirings, such as $\mathbb{L}, \mathbb{W}[X], \mathbb{S}[X]$ or $\mathbb{B}[X]$, admit pairs of finite interpretations that are non-isomorphic but elementarily equivalent, which immediately disproves completeness of G_m for $\equiv_m$ on those semirings (see Figure 1). Moreover, even on semirings such as $\mathbb{T}$, $\mathbb{N}$ and $\mathbb{N}[X]$, where it is known that elementary equivalence does coincide with isomorphism on finite interpretations [12], G_m is not necessarily complete. As a counterexample on the tropical semiring $\mathbb{T} = (\mathbb{R}^\infty_+, \min, +, \infty, 0)$, consider the following $\mathbb{T}$-interpretations.

π_A :

A	R	$\neg R$
a_0	0	∞
a_1	1	∞
a_2	1	∞

π_B :

B	R	$\neg R$
b_0	0	∞
b_1	2	∞

Clearly, Spoiler already wins $G_1(\pi_A, \pi_B)$, but we can show that $\pi_A \equiv_1 \pi_B$, thus the game G_1 is incomplete for $\equiv_1$ on $\mathbb{T}$. The 1-equivalence immediately follows from the following criterion.

▶ **Proposition 15.** *Two $\mathbb{T}$-interpretations π_A, π_B over vocabulary $\tau = \{R\}$ consisting of a single unary relation symbol are 1-equivalent if*
(1) $\pi_A(\neg Ra) = \pi_B(\neg Rb) = \infty$ *for all $a \in A$ and $b \in B$,*
(2) $\min_{a\in A} \pi_A(Ra) = \min_{b\in B} \pi_B(Rb)$ *and*
(3) $\sum_{a\in A} \pi_A(Ra) = \sum_{b\in B} \pi_B(Rb)$.

On the other side, in contrast to the classical m-turn Ehrenfeucht–Fraïssé game, there are semirings other than $\mathbb{B}$ on which the m-turn bijection game is both sound and complete.

▶ **Theorem 16.** *For every $m \in \mathbb{N}$, the bijection game BG_m is sound and complete for $\equiv_m$ on $\mathbb{N}$ and $\mathbb{N}[X]$.*

4 Characterising elementary equivalence

The Ehrenfeucht–Fraïssé theorem also provides a game-theoretic characterisation of elementary equivalence via the game $G(\mathfrak{A}, \mathfrak{B})$, where Spoiler chooses the number of turns at the beginning of each play. We now discuss soundness and completeness of G for $\equiv$ on semirings. For classical structures, soundness and completeness of G for $\equiv$ is equivalent to soundness and completeness of G_m for $\equiv_m$, for all m, but this is in general not the case on semirings.

For the study of the game G, interpretations on infinite universes are of particular interest. This especially applies to soundness, which is trivial in the finite case since a winning strategy for Duplicator already implies isomorphism on finite interpretations. Semiring semantics for infinite interpretations requires sum and product operators on infinite families $(s_i)_{i\in I} \subseteq S$ of semiring elements. There are certain semirings such as $\mathbb{N}, \mathbb{N}[X], \mathbb{B}[X]$ and $\mathbb{S}[X]$ which do not admit a reasonable definition of such infinitary operations, and we thus have to restrict ourselves to finite universes. Otherwise, we make use of the natural order and interpret infinite sums according to $\sum_{i\in I} s_i \coloneqq \sup\{\sum_{i\in I'} s_i | I' \subseteq I \text{ finite}\}$. For infinitary products we distinguish the case of absorptive semirings, where multiplication is decreasing and we thus interpret the product as the *infimum* of the finite subproducts, and the cases, such as $\mathbb{N}^\infty$ or $\mathbb{W}[X]$, where multiplication is increasing and we replace infima by suprema. Previous results such as the soundness of G_m on fully idempotent semirings straightforwardly extend to infinite interpretations by transferring semiring properties such as full idempotence to the infinitary operations.

Soundness of G for $\equiv$ holds whenever Spoiler wins $G(\pi_A, \pi_B)$ for all first-order separable interpretations π_A and π_B. Thus, the following question is essential: Given π_A, π_B and a separating sentence ψ, is the required number of turns for Spoiler to win $G(\pi_A, \pi_B)$ bounded in advance? On fully idempotent semirings, $m \coloneqq \mathrm{qr}(\psi)$ turns suffice for Spoiler to win $G(\pi_A, \pi_B)$ since G_m is sound for $\equiv_m$, which immediately implies soundness of G on all fully idempotent semirings. However, full idempotence is not a necessary condition, soundness of G is still preserved on many semirings that admit a weaker bound than m. For instance, on any n-idempotent semiring for some $n \in \mathbb{N}$, $n \cdot m$ turns suffice to ensure Spoiler's victory.

▶ **Proposition 17.** *Let $\mathcal{S}$ be n-idempotent for some $n \in \mathbb{N}$. For any $\mathcal{S}$-interpretations π_A and π_B it holds that $\pi_A \equiv_m \pi_B$ if Duplicator wins the game $G_{nm}(\pi_A, \pi_B)$. In particular, the game G is sound for $\equiv$ on $\mathcal{S}$.*

This follows from soundness of n-counting games as stated in Theorem 13. If $\pi_A \not\equiv_m \pi_B$, Spoiler wins $G_{nm}(\pi_A, \pi_B)$ by adapting his winning strategy for $CG^n_m(\pi_A, \pi_B)$: Instead of drawing n-element sets, he draws n elements one by one. Note that the bound $n \cdot m$ does not depend on π_A and π_B at all but only on the quantifier rank m and the semiring.

However, other semirings, such as $\mathbb{N}^\infty$, may not admit an inherent bound $t(m) \in \mathbb{N}$ such that a winning strategy of Duplicator for $G_{t(m)}(\pi_A, \pi_B)$ always implies $\pi_A \equiv_m \pi_B$. To demonstrate this, consider a pair of sets ($\mathbb{N}^\infty$-interpretations with empty vocabulary) with $t(m)$ and $t(m)+1$ elements, respectively. Clearly, Duplicator wins on those sets for up to $t(m)$ turns, but the sentence $\exists x(x = x)$ with quantifier rank 1 suffices to separate them.

In order to prove that the game G is still sound for $\equiv$ on $\mathbb{N}^\infty$, it is crucial to observe that two separable interpretations π_A, π_B with $\pi_A[\![\psi]\!] \neq \pi_B[\![\psi]\!]$ admit a parameter k that induces an upper bound on the number of moves required by Spoiler to win $G(\pi_A, \pi_B)$. On $\mathbb{N}^\infty$, this parameter is easily obtained by observing that $\pi_A[\![\psi]\!]$ or $\pi_B[\![\psi]\!]$ is finite (see [4] for a proof).

▶ **Theorem 18.** *Let π_A and π_B be $\mathbb{N}^\infty$-interpretations with elements $\bar{a} \in A^n$, $\bar{b} \in B^n$ and $k \geq 1$. If there is a separating formula $\varphi(\bar{x})$ with $\mathrm{qr}(\varphi(\bar{x})) \leq m$ such that $\pi_A[\![\varphi(\bar{a})]\!] < k$ or $\pi_B[\![\varphi(\bar{b})]\!] < k$, then Spoiler wins $G_{km}(\pi_A, \bar{a}, \pi_B, \bar{b})$.*

It turns out that a similar approach is applicable to the semiring $\mathbb{S}^\infty[X]$, which extends the semiring $\mathbb{S}[X]$ of absorptive polynomials to allow infinite exponents (and thus infinite products), albeit the derivation of a suitable parameter is more involved. Recall that a monomial m *absorbs* a monomial m' if the exponents for all $x \in X$, denoted by $m(x)$ and $m'(x)$ respectively, satisfy $m(x) \leq m'(x)$, and that absorptive polynomials only retain absorption-dominant monomials. We say that a monomial m *separates* polynomials p and q if $m \in p$ and m is not absorbed by any monomial from q.

These concepts can be extended to any subset $Y \subseteq X$: m *Y-absorbs* m' iff $m(x) \leq m'(x)$ for $x \in Y$, and it is *Y-separating* for p and q if it is contained in one of the polynomials but not Y-absorbed by any of the monomials from the other polynomial. Finally, we can parametrise monomials m by adding their exponents $e_Y(m) \coloneqq \sum_{x \in Y} m(x)$ for all the variables $x \in Y$. Now, we can extract a finite parameter from any pair of distinct polynomials p, q as follows.

▶ **Lemma 19.** *For any two distinct polynomials $p, q \in \mathbb{S}^\infty[X]$, there is a set $Y \subseteq X$ and a Y-separating monomial m such that the parameter $e_Y(m)$ is finite.*

Proof. It is known that $p \leq q$ holds if, and only if, every monomial in p is absorbed by some monomial from q (see [7]). Thus, there is a monomial m in either p or q that is not absorbed by any monomial from the other polynomial. Otherwise, p and q would absorb each other, which would imply $p = q$. Pick $Y \coloneqq \{x \in X \mid m(x) \neq \infty\}$. It follows that $e_Y(m)$ is finite and that m is not Y-absorbed by any monomial from the other polynomial since any m' that Y-absorbs m would also absorb m entirely. ◀

For example, $x^n y^\infty$ and $x^\infty y^\infty$ are $\{x\}$-separated by $m \coloneqq x^n y^\infty$ with $e_{\{x\}}(x^n y^\infty) = n$. This property can be exploited to limit the number of turns required by Spoiler to win $G(\pi_A, \pi_B)$ on separable $\mathbb{S}^\infty[X]$-interpretations.

▶ **Theorem 20.** *Let $k \geq 1$ and π_A, π_B be $\mathbb{S}^\infty[X]$-interpretations with elements $\bar{a} = (a_1, \ldots, a_n)$ and $\bar{b} = (b_1, \ldots, b_n)$. If there is a separating formula $\varphi(\bar{x})$ with $\mathrm{qr}(\varphi(\bar{x})) \leq m$, a set $Y \subseteq X$ and a Y-separating monomial m for $\pi_A[\![\varphi(\bar{a})]\!]$ and $\pi_B[\![\varphi(\bar{b})]\!]$ such that $e_Y(m) < k$, then Spoiler wins $G_{km}(\pi_A, \bar{a}, \pi_B, \bar{b})$.*

Proof. We show the claim by structural induction on the separating formula $\varphi(\bar{x})$. Since π_A and π_B are interchangeable, we may assume w.l.o.g. that the Y-separating monomial m is part of $\pi_A[\![\varphi(\bar{a})]\!]$. If $\varphi(\bar{x})$ is a literal, Spoiler wins immediately.

- If $\varphi(\bar{x}) = \varphi_1(\bar{x}) \vee \varphi_2(\bar{x})$, the Y-separating monomial m must be part of $\pi_A[\![\varphi_i(\bar{a})]\!]$ for some $i \in \{1,2\}$, but by definition, it cannot be Y-absorbed by any monomial in $\pi_B[\![\varphi_i(\bar{b})]\!]$. Thus, m Y-separates $\pi_A[\![\varphi_i(\bar{a})]\!]$ from $\pi_B[\![\varphi_i(\bar{b})]\!]$ and the claim follows by induction.
- If $\varphi(\bar{x}) = \exists x\psi(\bar{x}, x)$, then $\pi_A[\![\varphi(\bar{a})]\!] = \sum_{a\in A} \pi_A[\![\psi(\bar{a}, a)]\!]$, and analogous to the previous case, we observe that m is part of $\pi_A[\![\psi(\bar{a}, a)]\!]$ for some $a \in A$, but not Y-absorbed by any $\pi_B[\![\psi(\bar{b}, b)]\!]$ for $b \in B$. Thus, Spoiler can pick such an element $a \in A$ and win the remaining subgame by induction hypothesis.
- If $\varphi(\bar{x}) = \varphi_1(\bar{x}) \wedge \varphi_2(\bar{x})$, the Y-separating monomial $m = m_1 \cdot m_2$ is obtained by multiplying two monomials with $m_i \in \pi_A[\![\varphi_i(\bar{a})]\!]$ for $i \in \{1,2\}$. There is at least one $i \in \{1,2\}$ such that m_i Y-separates $\pi_A[\![\varphi_i(\bar{a})]\!]$ from $\pi_B[\![\varphi_i(\bar{b})]\!]$. Otherwise, each m_i would be Y-absorbed by some $m_i' \in \pi_B[\![\varphi_i(\bar{b})]\!]$, which would yield a contradiction since $m' = m_1' \cdot m_2' \in \pi_B[\![\varphi(\bar{b})]\!]$ would Y-absorb m. Clearly, $e_Y(m_i) \leq e_Y(m) < k$, hence Spoiler wins by invoking the induction hypothesis on the suitable subformula.
- If $\varphi(\bar{x}) = \forall x\psi(\bar{x}, x)$, then $\pi_A[\![\varphi(\bar{a})]\!] = \prod_{a\in A} \pi_A[\![\psi(\bar{a}, a)]\!]$. Decompose the monomial m into $m = \prod_{a\in A} m_a$ such that $m_a \in \pi_A[\![\psi(\bar{a}, a)]\!]$ holds for all $a \in A$. It follows that $e_Y(m) = \sum_{a\in A} e_Y(m_a) < k$, thus $e_Y(m_a)$ is nonzero for $\ell < k$ elements $a_1, \ldots, a_\ell \in A$ and zero otherwise. Spoiler picks those elements and Duplicator replies with $b_1, \ldots, b_\ell$.
  - If there is any $1 \leq i \leq \ell$ such that m_{a_i} is not Y-absorbed by any monomial in $\pi_B[\![\psi(\bar{b}, b_i)]\!]$, then m_{a_i} Y-separates $\pi_A[\![\psi(\bar{a}, a_i)]\!]$ from $\pi_B[\![\psi(\bar{b}, b_i)]\!]$, and together with $e_Y(m_{a_i}) \leq e_Y(m) < k$, we can apply the induction hypothesis.
  - Otherwise, each m_{a_i} is Y-absorbed by some $m_{b_i} \in \pi_B[\![\psi(\bar{b}, b_i)]\!]$. Since $\prod_{i=1}^{\ell} m_{b_i}$ Y-absorbs m, it is impossible that each $\pi_B[\![\psi(\bar{b}, b)]\!]$ for $b \in B \setminus \{b_1, \ldots, b_\ell\}$ contains some monomial m' with $e_Y(m') = 0$. Otherwise, those monomials would not contribute anything to the exponents of variables $x \in Y$, and their product together with $m_{b_1}, \ldots, m_{b_\ell}$ would result in a monomial $m'' \in \pi_B[\![\varphi(\bar{b})]\!]$ that Y-absorbs m, contradicting the definition of m. Now, it only remains for Spoiler to pick some $b \in B \setminus \{b_1, \ldots, b_\ell\}$ such that $\pi_B[\![\psi(\bar{b}, b)]\!]$ only contains monomials m' with $e_Y(m') > 0$. Duplicator must answer $a \in A \setminus \{a_1, \ldots, a_\ell\}$, but then $e_Y(m_a) = 0$, hence m_a Y-separates $\pi_A[\![\psi(\bar{a}, a)]\!]$ from $\pi_B[\![\psi(\bar{b}, b)]\!]$ and we can apply the induction hypothesis. ◀

▶ **Corollary 21.** *The game G is sound for $\equiv$ on the semirings $\mathbb{W}[X], \mathbb{N}^\infty$ and $\mathbb{S}^\infty[X]$.*

However, G is unsound for some important semirings. We construct a counterexample for the soundness of G on $\equiv$ in the tropical semiring (which is isomorphic to the Viterbi semiring $\mathbb{V}$) and transfer it to the isomorphic variant $\mathbb{D}$ of $\mathbb{L}$ by making sure that the valuations are in the interval $[0, 1]$, and that the separating formula does not evaluate to a semiring element greater than 1 in both interpretations. The main idea behind the construction is that, given a sequence $(s_i)_{\geq 1}$ of edge labels, Spoiler cannot distinguish an infinite star with exactly i edges labelled with $s_i \in \mathbb{T}$ for each $i \in \mathbb{N}$ from an infinite star where $\min(i, m)$ edges are labelled with s_i (see Figure 2). However, for an appropriate sequence of edge labels such star graphs with distinguished centre nodes can be separated in FO by summing up all edge labels using the formula $\psi(x) = \forall y(x = y \vee Exy)$.

▶ **Lemma 22.** *There is a sequence $(s_i)_{i\geq 1}$ of real numbers in $[0, 1]$ such that for each $m \in \mathbb{N}_{>0}$*

$$1 > \sum_{i\geq 1} i \cdot s_i > \sum_{i\geq 1} \min(i, m) \cdot s_i.$$

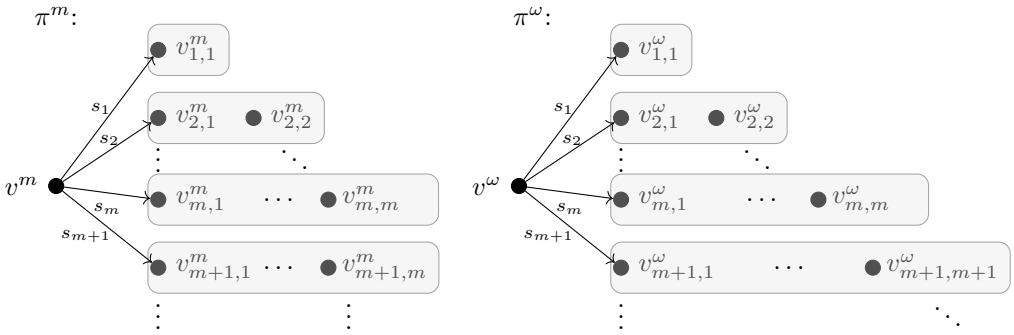


Figure 2 Infinite star graphs used to construct a counterexample against the soundness of the game G with respect to $\mathbb{T}$- and $\mathbb{D}$-interpretations. The grey boxes are meant to indicate $\pi^m[\![Ev^m v^m_{i,j}]\!] = \pi^\omega[\![Ev^m v^m_{i,j}]\!] = s_i$ for each j. Non-edges are assigned their Boolean truth value.

Proof. We prove the claim for $(s_i)_{i\geq 1}$ where $s_i := \frac{1}{i \cdot 2^{i+1}}$. Due to convergence of the geometrical series we obtain that $\sum_{i\geq 1} i \cdot s_i = 0.5$. Further,

$$\sum_{i\geq 1} i \cdot s_i = \sum_{i\geq 1} \min(i,m) \cdot s_i + \underbrace{\sum_{i>m} (m-i) \cdot s_i}_{>0} > \sum_{i\geq 1} \min(i,m) \cdot s_i,$$

which implies the claim. ◀

In order to ensure that Duplicator wins the game G_m for each $m \in \mathbb{N}$ on single semiring interpretations π and π', we combine the star graphs π^m for arbitrarily large m. The idea is to include in both π and π' the star graphs π^m for each $m \in \mathbb{N}$ as disjoint subgraphs, and to add an additional copy of π^ω to π' only. Using the sequence of edge labels satisfying $\sum_{i\geq 1} i \cdot s_i > \sum_{i\geq 1} \min(i,m) \cdot s_i$ for each $m \in \mathbb{N}_{>0}$ yields $\pi^\omega[\![\psi(v^\omega)]\!] > \pi^m[\![\psi(v^m)]\!]$, so the additional subgraph π^ω in π' would not contribute to the valuation of the sentence $\exists x \psi(x)$. Hence, we add additional vertices to the star graphs π^m in both π and π' which increase the sum over all outgoing edges and cause $\exists x \psi(x)$ to separate the resulting semiring interpretations.

▶ **Theorem 23.** *The game G is not sound for $\equiv$ on $\mathbb{T}, \mathbb{D}, \mathbb{V}$ and $\mathbb{L}$.*

Proof. Let $\mathcal{S} \in \{\mathbb{T}, \mathbb{D}\}$ and $(s_i)_{i\geq 1}$ be defined by $s_i := \frac{1}{i \cdot 2^{i+1}}$. Further, let s^m_∞ denote $\sum_{i\geq 1} \min(i,m) \cdot s_i$ for each $m \geq 1$. We inductively define a function $f\colon \mathbb{N} \setminus \{0\} \to \mathbb{N} \setminus \{0\}$ which determines the number of additional nodes that are added to the star graphs. Let $f(1)$ be the smallest number such that $s^1_\infty + f(1) \cdot s_1 > 0.5$. For $m > 1$, we define $f(m)$ as the minimum number yielding $s^m_\infty + f(m) \cdot s_m \geq s^{m-1}_\infty + f(m-1) \cdot s_{m-1}$. Since $0 < s_i < 1$ for all $i \geq 1$, f is well-defined. Hence, we obtain a chain $s^1_\infty + f(1) \cdot s_1 \leq s^2_\infty + f(2) \cdot s_2 \leq \ldots$ which is strictly upper bounded by 0.5. Based on f and $(s_i)_{i\geq 1}$, we construct $\mathcal{S}$-interpretations π and π' over the vocabulary $\tau = \{E\}$ consisting of a binary relation symbol. The universes V and V' are composed as follows.

$$V = \{v^m\colon m \geq 1\} \cup \{v^m_{i,j}\colon j \leq \min(i,m)\} \cup \{v^m_{m,m+j}\colon j \leq f(m)\}$$
$$V' = V \cup \{v^\omega\} \cup \{v^\omega_{i,j}\colon j \leq i\}$$

The valuations in π and π' are defined according to the following rules, which apply to all $m, n, i, j \in \mathbb{N}_{>0}$ with $m \neq n$ such that the respective nodes are contained in V or V'.

- $\pi(Ev^m v^m_{i,j}) = \pi'(Ev^m v^m_{i,j}) = \pi'(Ev^\omega v^\omega_{i,j}) = s_i$
- $\pi(Ev^m v^n_{i,j}) = \pi'(Ev^m v^n_{i,j}) = \pi'(Ev^\omega v^m_{i,j}) = \pi'(Ev^m v^\omega_{i,j}) = 1$
- $\pi(Ev^m v^n) = \pi'(Ev^m v^n) = \pi'(Ev^\omega v^m) = \pi'(Ev^m v^\omega) = 1$

Further, the negations of the instantiated τ-literals defined above are valuated with 0. All remaining unnegated τ-literals over V and V' are valuated with 0 and their negations with 1. In both $\mathbb{T}$ and $\mathbb{D}$, we obtain the following valuations of the formula $\psi(x) = \forall y(x = y \vee Exy)$.

- $\pi[\![\psi(v^m_{i,j})]\!] = \pi'[\![\psi(v^m_{i,j})]\!] = \pi'[\![\psi(v^\omega_{i,j})]\!] = 0$
- $\pi[\![\psi(v^m)]\!] = \pi'[\![\psi(v^m)]\!] = s^m_\infty + f(m) \cdot s_m$
- $\pi'[\![\psi(v^\omega)]\!] = 0.5$

By construction of f, this implies $\pi_A[\![\exists x\psi(x)]\!] = s^1_\infty + f(1) \cdot s_1 > 0.5 = \pi_B[\![\exists x\psi(x)]\!]$, hence $\pi_A \not\equiv_2 \pi_B$. In order to construct a winning strategy for Duplicator in the game $G(\pi, \pi')$, let $V^n_0 = \{v^n\}$ and V^n_i for $i \geq 1$ contain all elements $v^n_{i,j}$ in V. We consider the partition $\mathcal{P} \coloneqq \{V^n_i \colon n \geq 1, i \geq 0\}$ of V and $\mathcal{P}' \coloneqq \mathcal{P} \cup \{V^\omega_i \colon i \geq 0\}$ of V'. Based on the number of turns m Spoiler chooses in the game $G(\pi_A, \pi_B)$, we define a bijection $g_m \colon \mathcal{P} \to \mathcal{P}'$ as follows.

$$g_m(V^n_i) \coloneqq \begin{cases} V^n_i, & n < m \\ V^\omega_i, & n = m \\ V^{n-1}_i, & n > m \end{cases}$$

Duplicator wins the game $G_m(\pi, \pi')$ by responding to any element in $V^n_i \subseteq V$ with an arbitrary element in $g_m(V^n_i)$ and every element in $V^n_i \subseteq V'$ with any element in $g^{-1}_m(V^n_i)$, merely making sure that (in)equalities with regard to the elements that have already been chosen are respected. This is possible because for each V^n_i we have that $|V^n_i| = |g_m(V^n_i)|$ or that $|V^n_i| \geq m$ and $|g_m(V^n_i)| \geq m$. ◀

We now turn to the study of completeness. Analogous to m-turn Ehrenfeucht–Fraïssé games, the game G cannot be complete for semirings where elementary equivalence and isomorphism of finite interpretations do not coincide since Duplicator clearly loses G on non-isomorphic finite interpretations. In the remaining cases, G must be complete with respect to finite interpretations because Spoiler winning the game implies non-isomorphism, but on finite interpretations, this already implies separability by a first-order formula.

▶ **Proposition 24.** *Let $\mathcal{S} \in \{\mathbb{T}, \mathbb{V}, \mathbb{N}, \mathbb{N}[X]\}$. If Spoiler wins $G(\pi_A, \pi_B)$ and π_A, π_B are finite $\mathcal{S}$-interpretations, then $\pi_A \not\equiv \pi_B$. Thus, G is complete for $\equiv$ on finite $\mathcal{S}$-interpretations.*

The question arises whether this completeness result can be lifted to infinite semiring interpretations. For the tropical semiring $\mathbb{T}$ we describe a counterexample which proves that G is incomplete for $\equiv$ on $\mathbb{T}$ (and hence also on $\mathbb{V}$ due to $\mathbb{V} \cong \mathbb{T}$).

▶ **Theorem 25.** *There are $\mathbb{T}$-interpretations π_A, π_B such that Spoiler wins $G_1(\pi_A, \pi_B)$ although $\pi_A \equiv \pi_B$. In particular, G is incomplete for $\equiv$ on $\mathbb{T}$.*

Proof. Let π_A and π_B be $\mathbb{T}$-interpretations with just one unary predicate R and universes $A \coloneqq \{a_i : i \in \mathbb{N}\}$ and $B \coloneqq \{b_i : i \in \mathbb{N}\}$, whose valuations are $\pi_A(Ra_i) = \pi_B(Rb_i) = 0$ if i is even, while $\pi_A(Ra_i) = 1$ and $\pi_B(Rb_i) = 2$ for all odd i; since the interpretations are assumed to be model-defining this implies that $\pi_A(\neg Ra_i) = \pi_B(\neg Rb_i) = \infty$ for all $i \in \mathbb{N}$.

Clearly, Spoiler wins $G_1(\pi_A, \pi_B)$. To prove that $\pi_A \equiv \pi_B$, we first show that for each formula $\varphi(\bar{x})$ the valuations $\pi_A[\![\varphi(\bar{a})]\!]$ and $\pi_B[\![\varphi(\bar{b})]\!]$ can only take the values 0 and ∞ if the tuples $\bar{a}$ and $\bar{b}$ only consist of even elements $a_{2\ell}$ and $b_{2\ell}$. The reasoning is identical for both interpretations, so we just consider π_A, and proceed by induction on $\varphi(\bar{x})$. For literals the claim holds by definition and for conjunctions and disjunctions it follows since $\{0, \infty\}$ is closed under the operations min and $+$.

Consider $\varphi(\bar{x}) = \exists y \psi(\bar{x}, y)$. For all $a \in A$ with $\pi_A(Ra) = 0$, it follows by the induction hypothesis that $\pi_A[\![\psi(\bar{a}, a)]\!] \in \{0, \infty\}$. If there is some $a \in A$ such that $\pi_A[\![\psi(\bar{a}, a)]\!] = 0$, it immediately follows that $\pi_A[\![\varphi(\bar{a})]\!] = \inf_{a \in A} \pi_A[\![\psi(\bar{a}, a)]\!] = 0$. Hence, it remains to show the claim for the case $\pi_A[\![\psi(\bar{a}, a)]\!] = \infty$ for all $a \in A$ with $\pi_A(Ra) = 0$. Fix some $c \in A$ that is not contained in $\bar{a}$ such that $\pi_A(Rc) = 0$. For each $a \in A$ with $\pi_A(Ra) = 1$ it holds, by monotonicity of the semiring operations, that $\pi_A[\![\psi(\bar{a}, a)]\!] \geq \pi_A[\![\psi(\bar{a}, c)]\!]$ with respect to the usual order on $\mathbb{R}_+^\infty$ (which is the inverse of the natural order on $\mathbb{T}$) and since $\pi_A[\![\psi(\bar{a}, c)]\!] = \infty$, we have that $\pi_A[\![\varphi(\bar{a})]\!] = \inf_{a \in A} \pi_A[\![\psi(\bar{a}, a)]\!] = \infty$.

Finally, let $\varphi(\bar{x}) = \forall y \psi(\bar{x}, y)$. Again, for all $a \in A$ with $\pi_A(Ra) = 0$ it holds that $\pi_A[\![\psi(\bar{a}, a)]\!] \in \{0, \infty\}$ by induction hypothesis. If there is an $a \in A$ such that $\pi_A[\![\psi(\bar{a}, a)]\!] = \infty$, it immediately follows that $\pi_A[\![\varphi(\bar{a})]\!] = \sum_{a \in A} \pi_A[\![\psi(\bar{a}, a)]\!] = \infty$. Therefore it remains to show the claim for the case that $\pi_A[\![\psi(\bar{a}, a)]\!] = 0$ for all $a \in A$ with $\pi_A(Ra) = 0$. We observe that for all $a, a' \in A$ that do not occur in $\bar{a}$ with $\pi_A(Ra) = \pi_A(Ra')$, it holds that $(\pi_A, \bar{a}, a) \cong (\pi_A, \bar{a}, a')$. Hence, if there was some $a \in A$ with $\pi_A(Ra) = 1$ such that $\pi_A[\![\psi(\bar{a}, a)]\!] = s$ for some $s > 0$, then $\pi_A[\![\psi(\bar{a}, a)]\!] = s$ would hold for all $a \in A$ with $\pi_A(Ra) = 1$, which implies $\pi_A[\![\varphi(\bar{a})]\!] = \sum_{a \in A} \pi_A[\![\psi(\bar{a}, a)]\!] = \infty$. Otherwise, we have that $\pi_A[\![\psi(\bar{a}, a)]\!] = 0$ for all $a \in A$, thus $\pi_A[\![\varphi(\bar{a})]\!] = 0$, which completes the induction.

In particular we have for every sentence $\varphi \in \mathrm{FO}(\{R\})$ that $\pi_A[\![\varphi]\!], \pi_B[\![\varphi]\!] \in \{0, \infty\}$. We claim that $\pi_A[\![\varphi]\!] = \pi_B[\![\varphi]\!]$. The function $h \colon \mathbb{T} \to \mathbb{T}$ defined by $s \mapsto 2s$ is an endomorphism on $\mathbb{T}$ that is compatible with the infinitary operations, and obviously, $(h \circ \pi_A) \cong \pi_B$. If $\pi_A[\![\varphi]\!] = 0$, then $\pi_B[\![\varphi]\!] = 2 \cdot 0 = 0$ due to the fundamental property. Otherwise, we have that $\pi_A[\![\varphi]\!] = \infty = 2 \cdot \infty = \pi_B[\![\varphi]\!]$. Hence $\pi_A \equiv \pi_B$. ◀

The natural semiring does not admit infinitary operations, so we consider its extension $\mathbb{N}^\infty$ instead. But on $\mathbb{N}^\infty$, counterexamples disproving completeness also exist, see [4].

▶ **Theorem 26.** *There are $\mathbb{N}^\infty$-interpretations π_A and π_B such that Spoiler wins $G_1(\pi_A, \pi_B)$ although $\pi_A \equiv \pi_B$. In particular, the game G is incomplete for $\equiv$ on $\mathbb{N}^\infty$.*

Consequently, completeness of G for $\equiv$ also fails on any semiring which extends $\mathbb{N}[X]$ and admits infinitary operations if it contains $\mathbb{N}^\infty$ as a subsemiring.

5 The homomorphism game

Finally, we propose a new kind of model comparison games referred to as *homomorphism games*. The idea is to reduce a given pair of $\mathcal{S}$-interpretations to $\mathbb{B}$-interpretations via homomorphisms. In general, the resulting $\mathbb{B}$-interpretations are no longer model-defining, which is why their m-equivalence is not captured by G_m. While soundness of G_m for $\equiv_m$ on fully idempotent semirings $\mathcal{S}$ does not rely on the assumption that the $\mathcal{S}$-interpretations are model-defining, completeness for $\equiv_m$ even fails on $\mathbb{B}$ because a priori there is no connection between literals and their negations. This is illustrated by the $\mathbb{B}$-interpretations π_A and π_B.

π_A :

A	R_1	R_2	$\neg R_1$	$\neg R_2$
a_0	1	0	0	0
a_1	0	0	0	0
a_2	1	1	0	0
a_3	0	0	0	0
a_4	1	1	0	0
⋮	⋮	⋮	⋮	⋮

π_B :

B	R_1	R_2	$\neg R_1$	$\neg R_2$
b_0	0	0	0	0
b_1	1	1	0	0
b_2	0	0	0	0
b_3	1	1	0	0
b_4	0	0	0	0
⋮	⋮	⋮	⋮	⋮

▶ **Proposition 27.** *On $\mathbb{B}$-interpretations that are not model-defining, the game G is incomplete for $\equiv$ and the m-turn game G_m is incomplete for $\equiv_m$ for any $m > 0$.*

Proof. Consider the interpretations π_A and π_B. We can construct a bijection $\sigma_{\leq} \colon A \to B$ such that $\pi_B(\sigma_{\leq}(L)) \leq \pi_A(L)$ for all $L \in \mathrm{Lit}_A(\tau)$ by mapping a_0 to some b_i with even i. Similarly, we can construct a bijection $\sigma_{\geq} \colon A \to B$ with $\pi_A(L) \leq \pi_B(\sigma_{\geq}(L))$ for all L by mapping a_0 to some b_i with odd i.

By structural induction, it follows that $\pi_B[\![\varphi(\sigma_{\leq}(\bar{a}))]\!] \leq \pi_A[\![\varphi(\bar{a})]\!] \leq \pi_B[\![\varphi(\sigma_{\geq}(\bar{a}))]\!]$ holds for all first-order formulae $\varphi(\bar{x})$ with k free variables $\bar{x}$ and $\bar{a} \in A^k$. For sentences ψ, this yields $\pi_A[\![\psi]\!] = \pi_B[\![\psi]\!]$, hence we have $\pi_A \equiv \pi_B$. However, Spoiler already wins $G_1(\pi_A, \pi_B)$ by picking a_0, which proves the claim. ◀

Additionally, finite counterexamples showing the unsoundness of G_m exist as well and can be constructed based on π_A and π_B by considering suitable subinterpretations of size $2m$ or $2m + 1$, respectively (see [4]). Due to Proposition 27, we consider a one-sided variant of the Ehrenfeucht–Fraïssé game which yields a characterisation of m-equivalence for $\mathbb{B}$-interpretations without requiring them to be model-defining.

Let $\mathcal{S}$ be naturally ordered by $\leq$ and π_A, π_B be two $\mathcal{S}$-interpretations. We say that $(\pi_A, \bar{a}) \leq (\pi_B, \bar{b})$ if for every literal $L(\bar{x})$ we have that $\pi_A(L(\bar{a})) \leq \pi_B(L(\bar{b}))$. Further, we say that $(\pi_A, \bar{a}) \preceq_m (\pi_B, \bar{b})$ if it holds that $\pi_A[\![\varphi(\bar{a})]\!] \leq \pi_B[\![\varphi(\bar{b})]\!]$ for any formula $\varphi(\bar{x})$ of quantifier rank at most m.

▶ **Definition 28.** The one-sided game $G^{\leq}_m(\pi_A, \pi_B)$ is played in the same way as $G_m(\pi_A, \pi_B)$, but the winning condition for Duplicator, assuming that the tuples $\bar{a}, \bar{b}$ were chosen after m moves, is $(\pi_A, \bar{a}) \leq (\pi_B, \bar{b})$ instead of $(\pi_A, \bar{a}) \equiv_0 (\pi_B, \bar{b})$.

Using monotonicity of both semiring operations with respect to the natural order, we obtain the following soundness result, which can be proved analogously to Theorem 8.

▶ **Proposition 29.** *Let $\mathcal{S}$ be any fully idempotent semiring. Then $G^{\leq}_m$ is sound for $\preceq_m$ on $\mathcal{S}$.*

On $\mathbb{B}$, the one-sided game $G^{\leq}_m$ is also complete for $\preceq_m$ even for $\mathbb{B}$-interpretations that are not model-defining. To prove this, we inductively construct characteristic formulae $\chi^m_{\pi_A,\bar{a}}(\bar{x})$ analogous to the classical Ehrenfeucht–Fraïssé theorem, but we omit literals $\neg R\bar{x}$ in $\chi^0_{\pi_A,\bar{a}}(\bar{x})$ if $\pi_A(R\bar{a}) = 0$. Let $\varphi^{=}_{\bar{a}}(\bar{x})$ define the equalities and inequalities of the elements in $\bar{a}$.

$$\chi^0_{\pi_A,\bar{a}}(\bar{x}) \coloneqq \varphi^{=}_{\bar{a}}(\bar{x}) \wedge \bigwedge \{L(\bar{x}) \in \mathrm{Lit}_n(\tau) : \pi_A(L(\bar{a})) = 1\}$$

$$\chi^{m+1}_{\pi_A,\bar{a}}(\bar{x}) \coloneqq \bigwedge_{a \in A} \exists x\, \chi^m_{\pi_A,\bar{a},a}(\bar{x}, x) \wedge \forall x \bigvee_{a \in A} \chi^m_{\pi_A,\bar{a},a}(\bar{x}, x)$$

▶ **Theorem 30.** *For any two $\mathbb{B}$-interpretations π_A and π_B with elements $\bar{a} \in A^n$ and $\bar{b} \in B^n$ and any $m \in \mathbb{N}$, the following are equivalent:*

(1) *Duplicator wins $G^{\leq}_m(\pi_A, \bar{a}, \pi_B, \bar{b})$;*

(2) *$\pi_B[\![\chi^m_{\pi_A,\bar{a}}(\bar{b})]\!] = 1$;*

(3) *$(\pi_A, \bar{a}) \preceq_m (\pi_B, \bar{b})$.*

To construct homomorphism games based on the one-sided games $G^{\leq}_m$ on $\mathbb{B}$-interpretations, we make use of separating sets of homomorphisms, which were introduced in [12].

▶ **Definition 31.** Given semirings $\mathcal{S}$ and $\mathcal{S}'$, a set H of homomorphisms from $\mathcal{S}$ to $\mathcal{S}'$ is called *separating* if for all $s, t \in S$ with $s \neq t$ there is some $h \in H$ with $h(s) \neq h(t)$.

For two given $\mathcal{S}$-interpretations π_A and π_B which are separable by some sentence ψ, we can think of the valuations $s \neq t$ of ψ in π_A and π_B, respectively, as witnesses for the separability of π_A and π_B. Further, whenever there is a homomorphism h such that $h(s) \neq h(t)$ and $(h \circ \pi_A) \equiv_m (h \circ \pi_B)$, we can exclude the pair (s,t) as a candidate for witnessing $\pi_A \not\equiv_m \pi_B$ due to the fundamental property. Thus, separating sets of homomorphisms yield the following reduction technique.

▶ **Lemma 32.** *Let $\mathcal{S}$ and $\mathcal{S}'$ be semirings and H a separating set of homomorphisms from $\mathcal{S}$ to $\mathcal{S}'$. Moreover let π_A, π_B be $\mathcal{S}$-interpretations, $\bar{a} \in A^n$ and $\bar{b} \in B^n$. It holds that $(h \circ \pi_A, \bar{a}) \equiv_m (h \circ \pi_B, \bar{b})$ for all $h \in H$ if, and only if, $(\pi_A, \bar{a}) \equiv_m (\pi_B, \bar{b})$.*

Based on a separating set H of homomorphisms $h \colon \mathcal{S} \to \mathbb{B}$, the *homomorphism game* $HG_m(H, \pi_A, \pi_B)$ can be defined as follows. Spoiler first chooses some $h \in H$ and puts either $\pi_0 = h \circ \pi_A$ and $\pi_1 = h \circ \pi_B$, or the other way around, i.e. $\pi_0 = h \circ \pi_B$ and $\pi_1 = h \circ \pi_A$. Then the game $G^{\leq}_m(\pi_0, \pi_1)$ is played. Using the fact that $G^{\leq}_m$ is sound and complete for $\preceq_m$ even on $\mathbb{B}$-interpretations which are not model-defining, soundness and completeness of HG_m for $\equiv_m$ can be stated as follows.

▶ **Theorem 33.** *Let $\mathcal{S}$ be a semiring with a separating set H of homomorphisms into $\mathbb{B}$. Given $\mathcal{S}$-interpretations π_A, π_B and $\bar{a} \in A^n, \bar{b} \in B^n$, the following are equivalent for $m \in \mathbb{N}$:*

(1) *Duplicator wins $HG_m(H, \pi_A, \bar{a}, \pi_B, \bar{b})$;*

(2) *$h(\pi_B[\![\chi^m_{h\circ\pi_A,\bar{a}}(\bar{b})]\!]) = h(\pi_A[\![\chi^m_{h\circ\pi_B,\bar{b}}(\bar{a})]\!]) = 1$ for each $h \in H$;*

(3) *$(\pi_A, \bar{a}) \equiv_m (\pi_B, \bar{b})$.*

Motivated by Birkhoff's representation theorem [1], we can explicitly construct a separating set of homomorphisms from any finite lattice semiring (i.e. fully idempotent and absorptive semiring) into $\mathbb{B}$, and embed it into the rules of the homomorphism game. Indeed, every semiring for which there is a separating set of homomorphisms to $\mathbb{B}$ must be a lattice semiring since for every homomorphism $h \colon \mathcal{S} \to \mathbb{B}$ and $s, t \in S$, we have $h(s \cdot s) = h(s) \wedge h(s) = h(s)$ and $h(s + st) = h(s) \vee (h(s) \wedge h(t)) = h(s)$. Due to absorption, we assume that the infinitary operations of a lattice semiring are given by $\sum_{i \in I} s_i := \sup\{\sum_{i \in I'} s_i | I' \subseteq I \text{ finite}\}$ and $\prod_{i \in I} s_i := \inf\{\prod_{i \in I'} s_i | I' \subseteq I \text{ finite}\}$.

▶ **Definition 34.** Let $\mathcal{S}$ be a finite lattice semiring. A non-zero element $s \in \mathcal{S}$ is said to be *+-indecomposable* if for all $r, t \in \mathcal{S}$ with $r \neq s$ and $t \neq s$ it holds that $r + t \neq s$. We denote the set of non-zero +-indecomposable of elements in $\mathcal{S}$ as $idc(\mathcal{S})$.

In a min-max semiring, for instance, every non-zero element is +-indecomposable. By contrast, the +-indecomposable elements in PosBool$[X]$ correspond to the monomials.

▶ **Lemma 35.** *For each $s \in idc(\mathcal{S})$ the mapping $h_s \colon \mathcal{S} \to \mathbb{B}$ defined by*

$$h_s(t) = \begin{cases} 1, & t + s = t \\ 0, & \text{otherwise} \end{cases}$$

is a homomorphism from $\mathcal{S}$ into $\mathbb{B}$.

Proof. Let $s \in idc(\mathcal{S})$ be non-zero and +-indecomposable.

(1) Since $0 + s = s \neq 0$, it holds that $h_s(0) = 0$. Further, we have that $1 + s = 1 + 1 \cdot s = 1$ due to absorption, hence $h_s(1) = 1$.

(2) In order to prove that $h_s(r+t) = h_s(r) + h_s(t)$ for all $r, t \in \mathcal{S}$, it remains to show that $s + (r+t) = r + t$ is equivalent to $s + r = r$ or $s + t = t$. If $s + (r+t) = r + t$, then with absorption and distributivity $sr + st = s(r+t) = s(s+r+t) = s + s(r+t) = s$. Since s is $+$-indecomposable by assumption, this implies $sr = s$ or $st = s$. Suppose w.l.o.g. that $sr = s$ which yields $r = r + sr = r + s$. For the converse implication, assume that $r + s = r$ or $t + s = t$. Clearly, both implications immediately yield $s + (r+t) = r + t$.

(3) To prove $h_s(r \cdot t) = h_s(r) \cdot h_s(t)$, we show that $s + rt = rt$ is equivalent to $s + r = r$ and $s + t = t$. If $s + rt = rt$, we can infer that $s + r = s + (r + rt) = (s + rt) + r = rt + r = r$ and an analogous result for t. Conversely, suppose that $s + r = r$ and $s + t = t$. Then $rt = (s + r)(s + t) = s + (r \cdot t)$ follows by distributivity.

(4) Pertaining to the compatibility of h_s with infinitary operations in $\mathcal{S}$, note that any infinite sum or product can be transformed into a finite sum or product due to full idempotence and the assumption that $\mathcal{S}$ is finite. Thus, the proof is already complete. ◀

Although we only consider the mappings h_s for $+$-indecomposable s to ensure that h_s is a homomorphism, any two elements in $\mathcal{S}$ can be separated by some h_s.

▶ **Lemma 36.** *The set $\{h_s \colon s \in idc(\mathcal{S})\}$ is a separating set of homomorphisms from $\mathcal{S}$ to $\mathbb{B}$.*

Proof. For $t \in \mathcal{S}$ let $S_t = \{s \in idc(\mathcal{S}) \colon s + t = t\}$. Due to idempotence, we have that $t + \sum_{s \in S_t} s = t$. Since $\mathcal{S}$ is assumed to be finite, there must be a tuple $t_1, \dots, t_n \in idc(\mathcal{S})$ with $t_1 + \dots + t_n = t$. With idempotence, this implies $t + t_i = t$, which yields $t_i \in S_t$ for each $1 \leq i \leq n$. Hence, we have that $t + \sum_{s \in S_t} s = \sum_{1 \leq i \leq n} t_i + \sum_{s \in S_t} s = \sum_{s \in S_t} s$. Overall, we obtain $t = t + \sum_{s \in S_t} s = \sum_{s \in S_t} s$.

Let $r, t \in \mathcal{S}$ with $r \neq t$. Since $r = \sum_{s \in S_r} s$ and $t = \sum_{s \in S_t} s$, it must hold that $S_r \neq S_t$. Let s be a witness for the inequality and assume w.l.o.g that $s \in S_r$. By definition of S_r, it holds that $s + r = r$, hence $h_s(r) = 1$. By contrast, $s \notin S_t$ yields $s + t \neq t$ and thus $h_s(t) = 0$. ◀

Now that we have an explicit construction a separating set of homomorphisms to $\mathbb{B}$ which applies to any finite lattice semiring, we can reformulate the homomorphism game as $HG_m(\pi_A, \pi_B)$ corresponding to $HG_m(H_{idc}, \pi_A, \pi_B)$ for finite lattice semirings as follows.

▶ **Definition 37.** At the beginning of each play in $HG_m(\pi_A, \pi_B)$, Spoiler chooses either $\pi_0 = \pi_A$ and $\pi_1 = \pi_B$ or vice versa, and some $s \in idc(\mathcal{S})$. In the i-th of m rounds, Spoiler chooses some $a_i \in A$ or $b_i \in B$ and Duplicator has to respond with an element a_i or b_i in the other structure. Duplicator wins the play if for the chosen tuples $\bar{c}, \bar{d}$ and each $L(\bar{x}) \in \mathrm{Lit}_m(\tau)$ $\pi_0(L(\bar{c})) + s = \pi_0(L(\bar{c}))$ implies $\pi_1(L(\bar{d})) + s = \pi_1(L(\bar{d}))$.

The direct construction of the separating set of homomorphisms also allows an explicit formulation of characteristic formulae $\chi^{m,s}_{\pi_A,\bar{a}}(\bar{x})$ for each $s \in idc(\mathcal{S})$ corresponding to the $\mathbb{B}$-interpretations $h_s \circ \pi_A$. Again $\varphi^{=}_{\bar{a}}(\bar{x})$ characterises the equalities and inequalities of the elements in $\bar{a}$.

$$\chi^{0,s}_{\pi_A,\bar{a}}(x_1, \dots, x_n) \coloneqq \varphi^{=}_{\bar{a}}(\bar{x}) \wedge \bigwedge \{L(\bar{x}) \in \mathrm{Lit}_n(\tau) \mid \pi_A(\bar{a}) + s = \pi_A(L(\bar{a}))\}$$

$$\chi^{m+1,s}_{\pi_A,\bar{a}}(x_1, \dots, x_n) \coloneqq \bigwedge_{a \in A} \exists x \, \chi^{m,s}_{\pi_A,\bar{a},a}(\bar{x}, x) \wedge \forall x \bigvee_{a \in A} \chi^{m,s}_{\pi_A,\bar{a},a}(\bar{x}, x)$$

In terms of the set $H_{idc} = \{h_s \colon s \in idc(\mathcal{S})\}$, the correctness of the game HG_m for finite lattice semirings can be stated as follows.

▶ **Theorem 38.** *The game HG_m is sound and complete for $\equiv_m$ on every finite lattice semiring $\mathcal{S}$. More precisely, given any $\mathcal{S}$-interpretations π_A, π_B and $\bar{a} \in A^n, \bar{b} \in B^n$ the following are equivalent for each $m \in \mathbb{N}$:*
(1) *Duplicator wins $HG_m(\pi_A, \bar{a}, \pi_B, \bar{b})$;*
(2) *For each $s \in idc(\mathcal{S})$, it holds that*

$$\pi_B[\![\chi^{m,s}_{\pi_A,\bar{a}}(\bar{b})]\!] + s = \pi_B[\![\chi^{m,s}_{\pi_A,\bar{a}}(\bar{b})]\!] \quad \text{and} \quad \pi_A[\![\chi^{m,s}_{\pi_B,\bar{b}}(\bar{a})]\!] + s = \pi_A[\![\chi^{m,s}_{\pi_B,\bar{b}}(\bar{a})]\!];$$

(3) $(\pi_A, \bar{a}) \equiv_m (\pi_B, \bar{b})$.

While applying to arbitrary finite lattice semirings, the set H_{idc} of homomorphisms is in general not sufficient to separate any two elements of an infinite lattice semiring. As an example, consider $\mathcal{S} = (\mathbb{Z}, +^{\mathcal{S}}, \cdot^{\mathcal{S}}, 0, 1)$ with $s +^{\mathcal{S}} t = \gcd(s,t)$ if $s \neq 0$ or $t \neq 0$, while $0 +^{\mathcal{S}} 0 = 0$ and $s \cdot^{\mathcal{S}} t = \mathrm{lcm}(s,t)$ for $s, t \in \mathbb{Z}$. For each $s \in \mathbb{Z}$, it holds that $\gcd(2s, 3s) = s$, so for $s \neq 0$ there are distinct r and t such that $s = r +^{\mathcal{S}} t$. By contrast, $\gcd(s,t) \neq 0$ for all $s, t \in \mathbb{Z} \setminus \{0\}$, hence $idc(\mathcal{S}) = \{0\}$, but $\{h_0\}$ is not a separating set of homomorphisms. Nevertheless, separating sets of homomorphisms into $\mathbb{B}$ also exist for infinite lattice semirings and can be constructed based on the prime ideals in $\mathcal{S}$. But in general, there does not have to be a separating set of *continuous* homomorphisms, which respect infinitary summation and multiplication in $\mathcal{S}$. Thus, the prime ideals in $\mathcal{S}$ yield a homomorphism game on finite $\mathcal{S}$-interpretations, while $\mathcal{S}$ itself might be infinite (see [4] for details).

▶ **Example 39.** We can use the homomorphism game to show that first-order logic with semiring semantics cannot express the following property on min-max-semirings with the monadic signature $\{Q, R\}$: *"For the majority of elements e in the universe, Qe has a greater value than Re."* To prove this, we use the following two $\mathcal{S}_4$-interpretations on the min-max-semiring $\mathcal{S}_4$ with four elements $\{0, 1, 2, 3\}$.

π_A :

A	Q	R	$\neg Q$	$\neg R$
a_1	1	3	0	0
a_2	2	1	0	0
a_3	3	2	0	0

π_B :

B	Q	R	$\neg Q$	$\neg R$
b_1	3	1	0	0
b_2	1	2	0	0
b_3	2	3	0	0

Clearly, π_A has the desired property while π_B does not. However, we can show with the homomorphism game $HG_m(\pi_A, \pi_B)$ that $\pi_A \equiv \pi_B$. First, we observe that every non-zero $s \in \mathcal{S}_4$ is $+$-indecomposable. The set $idc(\mathcal{S}_4)$ induces homomorphisms $h_{\geq i} \colon \mathcal{S}_4 \to \mathbb{B}$ for $i \in \{1, 2, 3\}$ such that $h_{\geq i}(j) = 1$ iff $j \geq i$. Hence, we essentially play the homomorphism game $HG_m(H, \pi_A, \pi_B)$ with the separating set of homomorphisms $H = \{h_{\geq 1}, h_{\geq 2}, h_{\geq 3}\}$. Now, it only remains to observe that applying any of these homomorphisms to π_A and π_B makes them isomorphic to each other, thus, Duplicator clearly has a winning strategy. This demonstrates the viability of homomorphism games as a proof method for inexpressibility results in semiring semantics.

6 Conclusion

We have provided a rather detailed study of soundness and completeness of Ehrenfeucht–Fraïssé games, and related model comparison games, for proving elementary equivalence and m-equivalence in semiring semantics. The general picture that emerges is quite diverse. While the m-move games G_m are sound and complete for $\equiv_m$ only on the Boolean semiring, the games still provide a sound method on fully idempotent semirings, such as min-max

semirings, lattice semirings, and the provenance semirings PosBool[X]. This permits to generalise certain classical results in logic, proved via Ehrenfeucht–Fraïssé games or back-and-forth systems, from Boolean structures to semiring interpretations in fully idempotent semirings. A particular example is the proof of a Hanf locality theorem for such semirings in [2]. For proving elementary equivalence, without restriction of the quantifier rank, Ehrenfeucht–Fraïssé games without a fixed number of moves provide a more powerful method, in the sense that it is sound on more semirings, including not only $\mathbb{N}$ and $\mathbb{N}^\infty$ but also the provenance semirings $\mathbb{W}[X]$, $\mathbb{B}[X], \mathbb{S}[X]$, $\mathbb{N}[X]$, and $\mathbb{S}^\infty[X]$. While in classical semantics, a separating sentence of quantifier rank m leads to a winning strategy of Spoiler in at most m moves, the situation in semirings may be more complicated, in the sense that a winning strategy of Spoiler which "simulates" a separating sentence may still exist, but may require a larger number of moves than given by the quantifier rank; as a consequence the unrestricted game G may still provide a sound method for proving elementary equivalence, although the m-move games are unsound for $\equiv_m$.

The most straightforward application of Ehrenfeucht–Fraïssé games and other model comparison games are inexpressibility results, showing that a property P is not expressible in a logic L. Classically, this is accomplished by constructing two structures, precisely one of which satisfies the property P, and then providing a winning strategy for Duplicator in an appropriate model comparison game on the two structures. This method only relies on the soundness of the model comparison game without requiring completeness. Hence, our soundness results enable us to lift inexpressibility results to semiring semantics for a significant class of semirings. Consider, for instance, a min-max-semiring $\mathcal{S}$ modelling access levels and $\mathcal{S}$-interpretations π that annotate every edge of a graph with a required access level. Then there is no first-order formula $\varphi(x, y)$ such that $\pi[\![\varphi(v, w)]\!]$ evaluates to the minimal access level required to go from v to w.

We have also studied bijection and counting games, and we have shown in particular, that m-move bijection games are sound for $\equiv_m$ on *all* semirings. We remark that these games have originally been invented in the form of k-pebble games for logics with counting. This means that rather than just selecting, in m turns, two m-tuples, the games proceed by moving a fixed number of k pairs of pebbles through the two structures in an a priori unrestricted number of moves. These games capture equivalences for formulae that may use at most k variables which can, however, be quantified again and again. We have chosen here the simplified variants of m-move games rather than k-pebble games, to study the relationship with the classical Ehrenfeucht–Fraïssé games for $\equiv_m$. However, also the definition of k-pebble bijection and counting games extends in a straightforward way from classical structures to semiring interpretations and their soundness properties for k-variable equivalences are analogous to those of the m-move variants for m-equivalence. But clearly, the k-pebble variants of these games deserve further study, and this will be part of our future work on the subject. We conjecture that by lifting the well-known CFI-construction to semirings one can show that there is no semiring where first-order logic, and even fixed point logic, is strong enough to express all properties that are decidable in PTIME.

On the other side, it has turned out that all these model comparison games are incomplete for elementary equivalence and m-equivalence on most semirings, with the exceptions of $\mathbb{N}$ and $\mathbb{N}[X]$. Most of these incompleteness results rely on the construction of logically equivalent semiring interpretations on which, however, Spoiler wins the games in few moves. The proof of elementarily equivalence for such interpretations in general relies on separating sets of homomorphisms. Based on this technique, we have proposed a new kind of model comparison games, homomorphism games, which in fact are sound and complete for m-equivalence

on finite lattice semirings. This also raises the question whether it is possible to develop further games that are sound and complete for more, or even all, semirings. An essential part of the homomorphism game is a one-sided version of the classical Ehrenfeucht–Fraïssé game, with a winning condition that is based on (weak) local homomorphisms rather than local isomorphisms, and which capture the notion that one interpretation never evaluates to strictly larger values than the other. This game itself is interesting also in many other contexts and will be further studied in future work.

References

1 Garrett Birkhoff. *Lattice Theory*. American Mathematical Society, Providence, 3rd edition, 1967.

2 Clotilde Bizière, Erich Grädel, and Matthias Naaf. Locality theorems in semiring semantics. In *48th International Symposium on Mathematical Foundations of Computer Science (MFCS 2023)*, volume 272 of *LIPIcs*, pages 20:1–20:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.MFCS.2023.20.

3 Camille Bourgaux, Ana Ozaki, Rafael Peñaloza, and Livia Predoiu. Provenance for the description logic ELHr. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI 2020)*, pages 1862–1869. ijcai.org, 2020. doi:10.24963/IJCAI.2020/258.

4 Sophie Brinke, Erich Grädel, and Lovro Mrkonjić. Ehrenfeucht–Fraïssé games in semiring semantics, 2023. Full version of this paper. arXiv:2308.04910.

5 Katrin Dannert and Erich Grädel. Provenance analysis: A perspective for description logics? In *Description Logic, Theory Combination, and All That*, volume 11560 of *Lecture Notes in Computer Science*, pages 266–285. Springer, 2019. doi:10.1007/978-3-030-22102-7_12.

6 Katrin Dannert and Erich Grädel. Semiring provenance for guarded logics. In *Hajnal Andréka and István Németi on Unity of Science*, volume 19 of *Outstanding Contributions to Logic*, pages 53–79. Springer, Cham, 2021. doi:10.1007/978-3-030-64187-0_3.

7 Katrin Dannert, Erich Grädel, Matthias Naaf, and Val Tannen. Semiring provenance for fixed-point logic. In *29th EACSL Annual Conference on Computer Science Logic (CSL 2021)*, volume 183 of *LIPIcs*, pages 17:1–17:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.CSL.2021.17.

8 Heinz-Dieter Ebbinghaus and Jörg Flum. *Finite Model Theory*. Springer, 2nd edition, 1995. doi:10.1007/3-540-28788-4.

9 Boris Glavic. Data provenance. *Foundations and Trends in Databases*, 9(3-4):209–441, 2021. doi:10.1561/1900000068.

10 Erich Grädel, Hayyan Helal, Matthias Naaf, and Richard Wilke. Zero-one laws and almost sure valuations of first-order logic in semiring semantics. In *Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2022)*, pages 41:1–41:12. ACM, 2022. doi:10.1145/3531130.3533358.

11 Erich Grädel, Niels Lücking, and Matthias Naaf. Semiring provenance for Büchi games: Strategy analysis with absorptive polynomials. In *Proceedings 12th International Symposium on Games, Automata, Logics, and Formal Verification (GandALF 2021)*, volume 346 of *EPTCS*, pages 67–82, 2021. doi:10.4204/EPTCS.346.5.

12 Erich Grädel and Lovro Mrkonjić. Elementary equivalence versus isomorphism in semiring semantics. In *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*, volume 198 of *LIPIcs*, pages 133:1–133:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ICALP.2021.133.

13 Erich Grädel and Val Tannen. Semiring provenance for first-order model checking, 2017. arXiv:1712.01980.

14 Erich Grädel and Val Tannen. Provenance analysis for logic and games. *Moscow Journal of Combinatorics and Number Theory*, 9(3):203–228, 2020. doi:10.2140/moscow.2020.9.203.

15 Todd J. Green, Grigoris Karvounarakis, and Val Tannen. Provenance semirings. In *Proceedings of the Twenty-Sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 2007)*, pages 31–40. ACM, 2007. doi:10.1145/1265530.1265535.

16 Todd J. Green and Val Tannen. The semiring framework for database provenance. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS 2017)*, pages 93–99. ACM, 2017. doi:10.1145/3034786.3056125.

17 Lauri Hella. Logical hierarchies in PTIME. In *Proceedings of the Seventh Annual Symposium on Logic in Computer Science (LICS 1992)*, pages 360–368. IEEE Computer Society, 1992. doi:10.1109/LICS.1992.185548.

18 Neil Immerman and Eric Lander. Describing graphs: A first-order approach to graph canonization. In *Complexity Theory Retrospective*, pages 59–81. Springer, New York, 1990. doi:10.1007/978-1-4612-4478-3_5.

Quantum Circuit Completeness: Extensions and Simplifications

Alexandre Clément
Université Paris-Saclay, ENS Paris-Saclay, CNRS, Inria, LMF, 91190, Gif-sur-Yvette, France

Noé Delorme
Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France

Simon Perdrix
Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France

Renaud Vilmart
Université Paris-Saclay, ENS Paris-Saclay, CNRS, Inria, LMF, 91190, Gif-sur-Yvette, France

Abstract

Although quantum circuits have been ubiquitous for decades in quantum computing, the first complete equational theory for quantum circuits has only recently been introduced. Completeness guarantees that any true equation on quantum circuits can be derived from the equational theory.

We improve this completeness result in two ways: (i) We simplify the equational theory by proving that several rules can be derived from the remaining ones. In particular, two out of the three most intricate rules are removed, the third one being slightly simplified. (ii) The complete equational theory can be extended to quantum circuits with ancillae or qubit discarding, to represent respectively quantum computations using an additional workspace, and hybrid quantum computations. We show that the remaining intricate rule can be greatly simplified in these more expressive settings, leading to equational theories where all equations act on a bounded number of qubits.

The development of simple and complete equational theories for expressive quantum circuit models opens new avenues for reasoning about quantum circuits. It provides strong formal foundations for various compiling tasks such as circuit optimisation, hardware constraint satisfaction and verification.

2012 ACM Subject Classification Theory of computation → Quantum computation theory; Theory of computation → Equational logic and rewriting

Keywords and phrases Quantum Circuits, Completeness, Graphical Language

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.20

Related Version *Full Version*: `https://arxiv.org/abs/2303.03117` [13]

Funding This work is supported by the Plan France 2030 through the PEPR integrated project EPiQ ANR-22-PETQ-0007 and the HQI initiative ANR-22-PNCQ-0002; it is also supported by the ANR project SoftQPro ANR-17-CE25-0009-02, by the STIC-AmSud project Qapla' 21-STIC-10, and by the European projects NEASQC and HPCQS.

1 Introduction

Introduced in the 80's by Deutsch [19], the quantum circuit[1] model is ubiquitous in quantum computing. Various quantum computing tasks – circuit optimisation, fault tolerant quantum computing, hardware constraint satisfaction, and verification – involve quantum circuit transformations [24, 34, 35, 36, 38]. It is therefore convenient to equip the quantum circuit formalism with an *equational theory* providing a way to transform a quantum circuit while

[1] Originally called *Quantum Computational Networks*, the term *quantum circuits* is nowadays unanimously used.

preserving the represented unitary map. When the equational theory is powerful enough to guarantee that any true property can be derived, it is said to be *complete*, in other words, any two circuits representing the same unitary map can be transformed into one another using the rules of the equational theory.

The first complete equational theory (denoted $\mathrm{QC}_{\mathrm{old}}$ in the following) for quantum circuits has been introduced recently [12]. This equational theory has been derived from the LOv-calculus [11], a language for optical quantum computing. Before that, complete equational theories were only known for non-universal fragments of quantum circuits, such as Clifford+T circuits acting on two qubits [6, 16], Clifford+CS circuits acting on three qubits [7], the stabiliser fragment [33, 42], the CNot-dihedral fragment [1], or fragments of reversible circuits [25, 15, 14].

The quantum circuit model can naturally be extended to encompass ancillary qubits, measurements, or qubit discarding, in order to express more general evolutions like isometries and completely positive trace preserving maps. In a model of quantum circuits with ancillae, one can use an additional work space by adding fresh qubits, as well as releasing qubits when they are in a specific state. Even if the vanilla quantum circuits form a universal model of quantum computation,[2] this additional space is useful in many cases. It is for instance commonly used for the construction of quantum oracles.[3] Another important example is the parallelisation of quantum circuits: ancillae enable a better parallelisation of quantum gates, leading generally to a tradeoff between space (number of ancillae) and depth (parallel time) [37]. Notice that ancillae should be carefully used as the computation should leave a clean work space: one can only get rid of a qubit at the end of the computation if this qubit is in the $|0\rangle$-state.

We also consider another extension of quantum circuits where arbitrary qubits can be discarded (or traced out), whatever their states are. This extension allows for the representation of: (*i*) quantum measurements and more generally classically controlled computations; and (*ii*) arbitrary general quantum computations (CPTP maps[4]). Such quantum circuits can be used to deal with fault-tolerant quantum computing and error correcting codes which, by construction, require an additional workspace, measurements and corrections. One can also represent measurement-based quantum computation [43, 17] with this class of circuits. The study of hybrid quantum-classical models is also a subject of interest in algorithmic and complexity theory [22, 2].

Contributions. We address here the problem of simplifying the complete equational theory $\mathrm{QC}_{\mathrm{old}}$. Obtained through a non-trivial translation from the LOv-calculus, $\mathrm{QC}_{\mathrm{old}}$ involves non-trivial equations (see Figure 3), in particular Equation ($\mathrm{K}^*_{\mathrm{old}}$) depicts a family of equations acting on an unbounded number of qubits, witness of the non-functoriality of the back and forth translations between quantum circuits and optical circuits, due to the fundamentally different interpretations of the parallel composition in the two circuit languages.

We show that several rules, including two of the three most intricate ones (Equations (12) and (13)), can actually be derived from the other rules, the third one (Equation ($\mathrm{K}^*_{\mathrm{old}}$)) being slightly simplified. This leads to a simpler, more compact and easier to use complete equational theory, which however still involves a family of equations acting on an unbounded number of qubits.

---

[2] Any n-qubit unitary transformation can be implemented by a n-qubit vanilla quantum circuit.

[3] Implementation of the n-qubit unitary transformation $U_f : |x, y\rangle \mapsto |x, y \oplus f(x)\rangle$ given a classical circuit implementing the boolean function f [39].

[4] Completely positive trace-preserving maps.

We consider the more expressive frameworks of quantum circuits with ancilla and/or discards. Several constructions for discarding [23, 9], measurements and quantum operations [45], allow one to turn the complete equational theory for vanilla quantum circuits into complete equational theories for quantum circuits with ancilla and/or discards, by adding a few extra equations. We then mainly show that in these more expressive setting, the unbounded family of equations (K^*_{old}) can be derived from bounded ones, leading to complete equational theories acting on a most three qubits.

Related work. The first complete equational theory for a universal quantum computing model has been introduced in 2017 for the ZX-calculus [26]. Since then, complete equational theories have been introduced for other universal fragments of the ZX-calculus [27, 21, 28, 47, 29] and its variants ZH-, ZW-calculi [3, 20]. ZX-like languages differ from quantum circuits mainly in two ways: they are more expressive, allowing the representation of any matrix[5] so in particular those representing post-selected evolutions for instance; the second major difference – and the most important in our context – is that not all the generators are unitary, thus even if a ZX-diagram represents an overall unitary evolution, it does not provide in general a (deterministic) implementation by means of elementary gates contrary to the quantum circuit model. To circumvent this problem one can consider the so-called subclass of circuit-like ZX-diagrams which is in one-to-one correspondence with quantum circuits, however this class is not closed under the known complete equational theories of the ZX-calculus. In particular, the problem of transforming a ZX-diagram representing a unitary evolution into a circuit-like one has been studied in the context of circuit optimisation [30], leading to various heuristics [31, 4, 18]. However, this approach fails so far to lead to a complete equational theory for quantum circuits.

The paper is structured as follows. In Section 2, we consider vanilla quantum circuits together with a new equational theory QC. We prove the completeness of QC first for the fragment of 1-CNot circuits,[6] that we then use to derive the remaining equations of the already known complete equational theory QC_{old} introduced in [12]. In Section 3, we introduce an extension of vanilla quantum circuits with $|0\rangle$-state initialisation. Universal for isometries, such quantum circuits with initialisation are introduced as an intermediate step towards circuits with ancillae and/or discard. We add to the equational theory QC two basic equations involving qubit-initialisation, and provide a proof of completeness of the augmented equational theory QC_{iso} using a particular circuit decomposition based on the so-called cosine-sine decomposition of unitary maps. The completeness of QC_{iso} is extended to provide complete equational theories for quantum circuits with ancillae ($QC_{ancilla}$ in Section 4) – which additionally allow for the release of qubits when they are in a specific state – and for quantum circuits with qubit discarding ($QC_{discard}$ in Section 5) – which allows the tracing out of any qubits. Both extensions provide alternative representations of multi-controlled gates, allowing the simplification of the remaining intricate rule – which acts on an unbounded number of qubits – into its 2-qubit version.

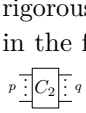
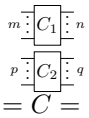
Due to space constraints, we only sketch the proofs in the present paper. Please refer to the full version [13] for the detailed proofs together with all the required derivations.

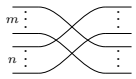
[5] the only constraint is on the dimension of the matrices which must be a power of two for the qubit case, qudit versions also exist [8, 41]

[6] The sub-class of quantum circuits made of at most one CNot gate.

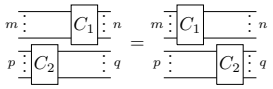
2 Vanilla quantum circuits

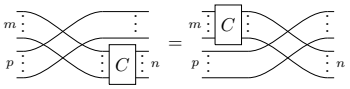
2.1 Graphical languages

We define quantum circuits using the formalism of props [32], which are, in category-theoretic terms, strict symmetric monoidal categories whose objects are generated by a single object, or equivalently with $(\mathbb{N}, +)$ as a monoid of objects. The prop formalism provides a formal and rigorous framework to describe graphical languages. The main features of props are recalled in the following. Circuits $C_1 : m \to n$ and $C_2 : p \to q$ in a prop, depicted as  and can be composed: (1) "in sequence" $C_2 \circ C_1 : m \to q$ if $n = p$, graphically  ; (2) "in parallel" $C_1 \otimes C_2 : m + p \to n + q$, graphically  . The *unit* for tensor product $\otimes$ is the *empty circuit*: $\boxdot : 0 \to 0$. This means $\boxdot \otimes C = C = C \otimes \boxdot$ for any circuit C. The circuit $— : 1 \to 1$ depicts the identity,  $: 2 \to 2$ is the identity on two wires and more generally $—^{\otimes m} := — \otimes (—)^{\otimes m-1} : m \to m$ (with $(—)^{\otimes 0} := \boxdot$) is the identity on m wires. Graphically, we obviously have $—^{\otimes n} \circ C = C = C \circ —^{\otimes m}$ for any $C : m \to n$. Finally, a prop is also endowed with a particular circuit  $: 2 \to 2$ which satisfies  $=$  . Graphically (and semantically in what follows)  swaps places. By compositions, we may build the following family of circuits

 $: m + n \to n + m$

which exchanges m-sized and n-sized registers. In a prop, circuits satisfy a set of identities, that graphically translate as "being able to deform the circuit". For instance, the following identities are valid transformations:





In the following, all the considered theories will be props, and hence will have the empty, identity and swap circuits as basic generators.

2.2 Vanilla quantum circuits and their equational theory

We first consider the vanilla model of quantum circuits generated by the very standard gateset: Hadamard, Phase gates, and CNot, together with global phases:

▶ **Definition 1.** *Let* **QC** *be the prop generated by*  $: 1 \to 1$,  $: 1 \to 1$,  $: 2 \to 2$ *and* $: 0 \to 0$ *for any* $\varphi \in \mathbb{R}$.

We associate with any quantum circuit its standard interpretation as a unitary map:

▶ **Definition 2** (Semantics)**.** *For any* n*-qubit* **QC***-circuit* C*, let* $[\![C]\!] : \mathbb{C}^{\{0,1\}^n} \to \mathbb{C}^{\{0,1\}^n}$ *be the* semantics *of* C *inductively defined as the linear map satisfying* $[\![C_2 \circ C_1]\!] = [\![C_2]\!] \circ [\![C_1]\!]$*;* $[\![C_1 \otimes C_2]\!] = [\![C_1]\!] \otimes [\![C_2]\!]$*; and*

$$[\![\boxdot]\!] = 1 \mapsto 1 \qquad [\![\text{}]\!] = 1 \mapsto e^{i\varphi} \qquad [\![\text{}]\!] = |x\rangle \mapsto \frac{|0\rangle + (-1)^x |1\rangle}{\sqrt{2}} \qquad [\![\text{}]\!] = |x\rangle \mapsto e^{ix\varphi} |x\rangle$$

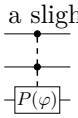
$$[\![\text{  }]\!] = |x, y\rangle \mapsto |x, x \oplus y\rangle \qquad [\![—]\!] = |x\rangle \mapsto |x\rangle \qquad [\![\text{  }]\!] = |x, y\rangle \mapsto |y, x\rangle$$

Note that for any **QC**-circuit C, $[\![C]\!]$ is unitary. Conversely, it is well known that any unitary map acting on a finite number of qubits can be represented by a **QC**-circuit:

▶ **Proposition 3** (Universality). **QC** *is universal, i.e. for any unitary* $U : \mathbb{C}^{\{0,1\}^n} \to \mathbb{C}^{\{0,1\}^n}$ *there exists a* **QC***-circuit* C *such that* $[\![C]\!] = U$.

Quantum circuits, as defined above, only have four different kinds of generators, however, it is often convenient to use other gates that can be defined by combining them. For instance, following [5, 12], Pauli gates, Toffoli, X-rotations, and multi-controlled gates are defined in Figure 1. Note that while the phase gate $-[P(\varphi)]-$ is 2π-periodic, the X-rotation $-[R_X(\theta)]-$ is 4π-periodic.

We use the standard bullet-based notation for multi-controlled gates. For instance 

denotes the application of a phase gate $-[P(\varphi)]-$ on the third qubit controlled by the first two qubits. With a slight abuse of notations, we use dashed lines for arbitrary number of control qubits, e.g. 

$: n+1 \to n+1$ or simply 

$: n+1 \to n+1$ have $n \geq 0$ control qubits (possibly zero), whereas 

$: n+2 \to n+2$ and 

$: 1+n+1 \to 1+n+1$ have at least one control qubit.

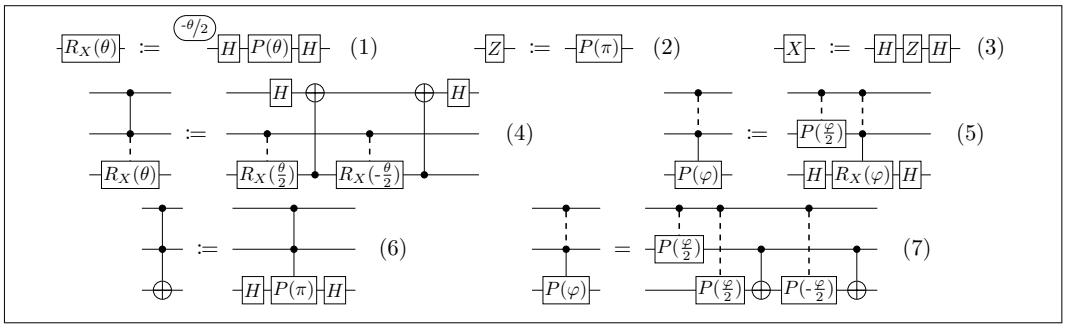


Figure 1 Shortcut notations for usual gates defined for any $\varphi, \theta \in \mathbb{R}$. Equation (1) defines X-rotations while Equations (2) and (3) define Pauli gates. Equations (4) and (5) are inductive definitions of multi-controlled gates. Equation (6) is the definition of the well known Toffoli gate. Equation (7) is a provably equivalent definition of the multi-controlled phase gate.

We equip the vanilla quantum circuits with the equational theory QC defined in Figure 2. We write $\mathrm{QC} \vdash C_1 = C_2$ when C_1 can be transformed into C_2 using the equations of QC. More formally, $\mathrm{QC} \vdash \cdot = \cdot$ is the smallest congruence which satisfies the equations of Figure 2 together with the deformation rules that come with the prop formalism. QC is sound, i.e. for any **QC**-circuits C_1, C_2 if $\mathrm{QC} \vdash C_1 = C_2$ then $[\![C_1]\!] = [\![C_2]\!]$. This can be proved by observing that all equations of QC are sound.

Figure 3 depicts the complete equational theory $\mathrm{QC}_{\mathrm{old}}$ for vanilla quantum circuits introduced in [12]. Compared to $\mathrm{QC}_{\mathrm{old}}$, Equations (8) and (9) are now subsumed by Equation (G) in QC, Equation (K^*) is a slight simplification of Equation ($\mathrm{K}^*_{\mathrm{old}}$) with one less parameter in the RHS circuit, whereas Equations (10) and (11) together with Equations (12) and (13) have been removed, as we prove in the following that they can be derived in QC.

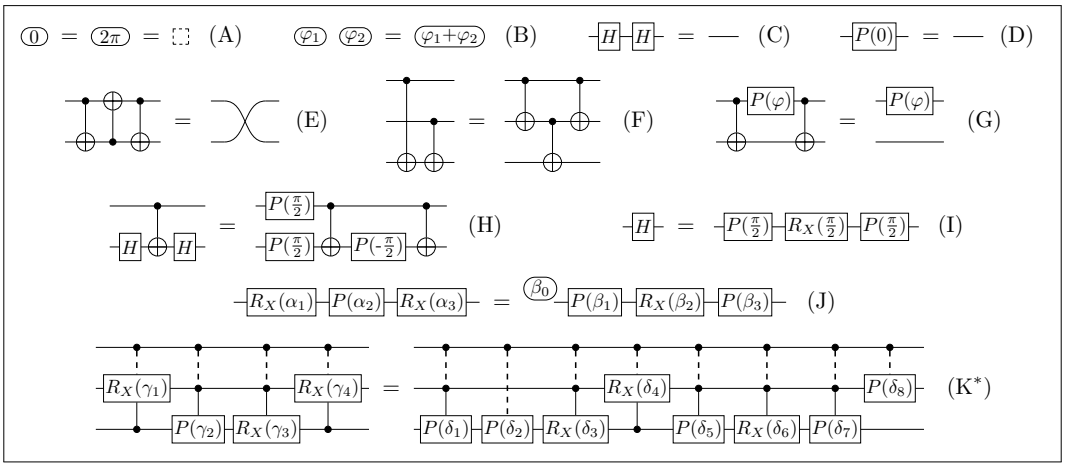


Figure 2 Equational theory QC. Equations (B) and (G) are defined for any $\varphi, \varphi_1, \varphi_2 \in \mathbb{R}$. In Equations (J) and (K*) the LHS circuit has arbitrary parameters which uniquely determine the parameters of the RHS circuit. Equation (J) follows from the well-known Euler-decomposition which states that any unitary can be decomposed, up to a global phase, into basic X- and Z-rotations. Thus for any $\alpha_i \in \mathbb{R}$, there exist $\beta_j \in \mathbb{R}$ such that Equation (J) is sound. We make the angles β_j unique by assuming that $\beta_1 \in [0, \pi)$, $\beta_0, \beta_2, \beta_3 \in [0, 2\pi)$ and if $\beta_2 \in \{0, \pi\}$ then $\beta_1 = 0$. Equation (K*) reads as follows: the equation is defined for any $n \geq 2$ input qubits, in such a way that all gates are controlled by the first $n-2$ qubits. Similarly to Equation (J), for any $\gamma_i \in \mathbb{R}$, there exist $\delta_j \in \mathbb{R}$ such that Equation (K*) is sound. We ensure that the angles δ_j are uniquely determined by assuming that $\delta_1, \delta_2, \delta_5 \in [0, \pi)$, $\delta_3, \delta_6, \delta_7, \delta_8 \in [0, 2\pi)$, $\delta_4 \in [0, 4\pi)$, if $\delta_3 = 0$ and $\delta_6 \neq 0$ then $\delta_2 = 0$, if $\delta_3 = \pi$ then $\delta_1 = 0$, if $\delta_4 \in \{0, 2\pi\}$ then $\delta_1 = \delta_3 = 0$, if $\delta_4 \in \{\pi, 3\pi\}$ then $\delta_2 = 0$, if $\delta_4 \in \{\pi, 3\pi\}$ and $\delta_3 = 0$ then $\delta_1 = 0$, and if $\delta_6 \in \{0, \pi\}$ then $\delta_5 = 0$.

2.3 Reasoning on quantum circuits

To derive an equation $C_1 = C_2$ over quantum circuits, one can apply some rules of the equational theory to transform step by step C_1 into C_2. In the context of vanilla quantum circuits, we can take advantage of the reversibility of generators to *simplify* equations. Indeed, intuitively, proving $C_1 \circ -\boxed{H}- = C_2 \circ -\boxed{H}-$ is equivalent to proving $C_1 = C_2$ as $-\boxed{H}-$ is (provably) reversible. Similarly, proving $C_1 = C_2$ should be equivalent to proving $C_1 \circ C_2^\dagger = —$, where the adjoint of a circuit is defined as follows:

▶ **Definition 4.** *For any* **QC***-circuit C, let $C^\dagger$ be the* adjoint *of C inductively defined as $(C_2 \circ C_1)^\dagger := C_1^\dagger \circ C_2^\dagger$; $(C_1 \otimes C_2)^\dagger := C_1^\dagger \otimes C_2^\dagger$; and for any $\varphi \in \mathbb{R}$, $(\textcircled{\varphi})^\dagger := \textcircled{-\varphi}$, $(-\boxed{P(\varphi)}-)^\dagger := -\boxed{P(-\varphi)}-$, and $g^\dagger := g$ for any other generator g.*

▶ **Proposition 5.** $[\![C^\dagger]\!] = [\![C]\!]^\dagger$ *for any* **QC***-circuit C, where $[\![C]\!]^\dagger$ is the usual linear algebra adjoint of $[\![C]\!]$.*

Proof. By induction on C. ◀

▶ **Proposition 6** (Simplification principle)**.** *For any n-qubit* **QC***-circuits C, C_1, C_2*

$$\mathrm{QC} \vdash C \circ C_1 = C_2 \quad \Leftrightarrow \quad \mathrm{QC} \vdash C_1 = C^\dagger \circ C_2$$

and

$$\mathrm{QC} \vdash C_1 \circ C = C_2 \quad \Leftrightarrow \quad \mathrm{QC} \vdash C_1 = C_2 \circ C^\dagger$$

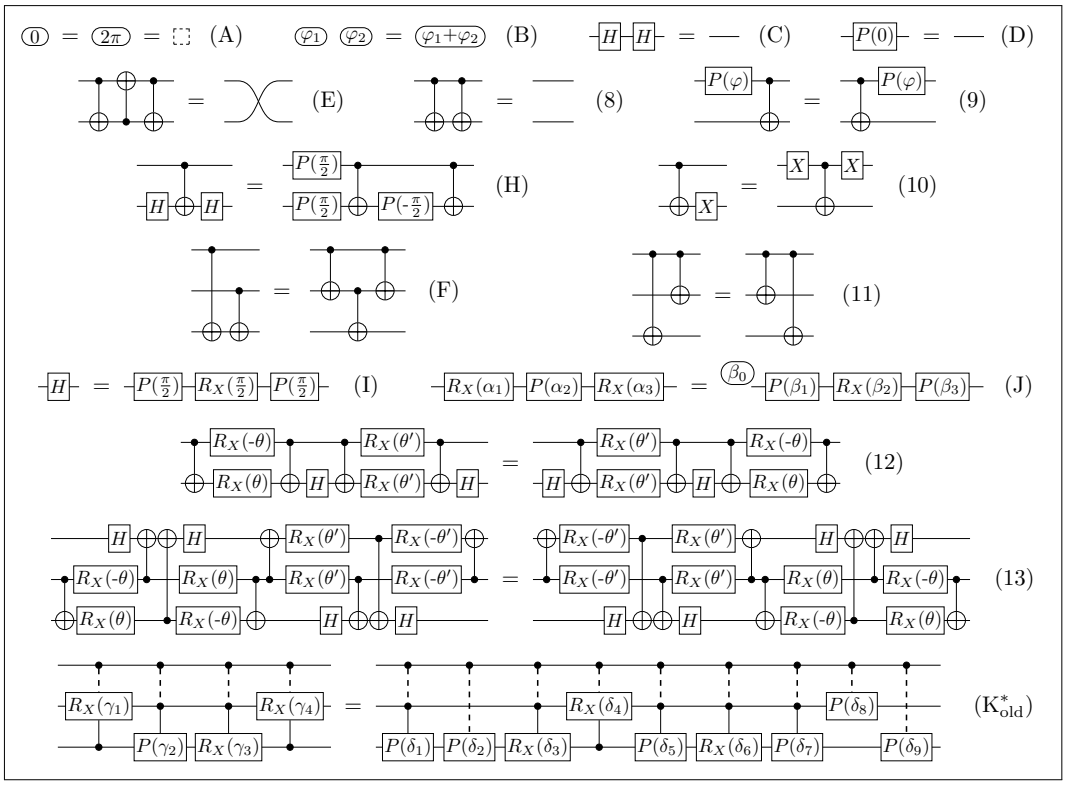


Figure 3 Equational theory $\mathrm{QC}_{\mathrm{old}}$ introduced in [12]. Equations (A),(B),(C),(D),(E),(H),(F),(I) are (J) are aslo in the equational theory QC. Equation ($\mathrm{K}^*_{\mathrm{old}}$) is the old version of Equation (K^*) with one more parameter, and where the uniqueness of the parameters δ_j is given by the conditions: $\delta_1, \delta_2, \delta_5 \in [0, \pi)$, $\delta_3, \delta_4, \delta_6, \delta_7, \delta_8, \delta_9 \in [0, 2\pi)$, if $\delta_3 = 0$ then $\delta_2 = 0$, if $\delta_3 = \pi$ then $\delta_1 = 0$, if $\delta_4 = 0$ then $\delta_1 = \delta_3$ $(= \delta_2) = 0$, if $\delta_4 = \pi$ then $\delta_2 = 0$, if $\delta_4 = \pi$ and $\delta_3 = 0$ then $\delta_1 = 0$, and if $\delta_6 \in \{0, \pi\}$ then $\delta_5 = 0$. Note that these conditions on the δ_j for $1 \leq j \leq 8$ are the same as in Equation (K^*) except for δ_4, which is restricted to be in $[0, 2\pi)$ instead of $[0, 4\pi)$, and for δ_2, which has to be 0 when $\delta_3 = 0$ even if $\delta_6 = 0$.

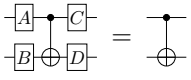
Proof. First we show by induction that $\mathrm{QC} \vdash C \circ C^\dagger = —^{\otimes n}$ and $\mathrm{QC} \vdash C^\dagger \circ C = —^{\otimes n}$ for any C. Then, w.l.o.g. we show that $(\mathrm{QC} \vdash C \circ C_1 = C_2) \Rightarrow (\mathrm{QC} \vdash C_1 = C^\dagger \circ C_2)$: assuming $\mathrm{QC} \vdash C \circ C_1 = C_2$, we have $\mathrm{QC} \vdash C_1 = C^\dagger \circ C \circ C_1 = C^\dagger \circ C_2$. ◀

2.4 Completeness

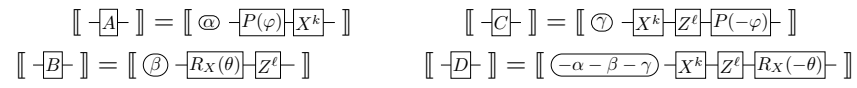
We prove the completeness of QC by showing that every equation of the original complete equational theory $\mathrm{QC}_{\mathrm{old}}$ introduced in [12] can be derived in QC. To this end we first show the completeness of QC for the (modest) fragment of quantum circuits containing at most one CNot gate.

▶ **Lemma 7** (1-CNot completeness). QC *is complete for circuits containing at most one* , *i.e. for any* **QC***-circuits* C_1, C_2 *with at most one*  , *if* $[\![C_1]\!] = [\![C_2]\!]$ *then* $\mathrm{QC} \vdash C_1 = C_2$.

Proof. First we can show that, for semantic reasons, it is enough to prove the statement for 2-qubit circuits containing no swap gate and exactly one CNot gate. Then, by the simplification principle (Proposition 6), it is sufficient to prove



whenever the equation is sound. By semantic analysis, we can show that there exist $\alpha, \beta, \gamma, \varphi, \theta \in \mathbb{R}$ and $k, \ell \in \{0, 1\}$ such that



where $-\boxed{X^k}-$ (resp. $-\boxed{Z^\ell}-$) denotes $-\boxed{X}-$ (resp. $-\boxed{Z}-$) if $k = 1$ (resp. $\ell = 1$) and — if $k = 0$ (resp. $\ell = 0$). Then, using the completeness of QC for one-qubit circuits (which is a direct consequence of the fact that all equations acting on at most one qubit of $\mathrm{QC}_{\mathrm{old}}$ are also in QC), it is straightforward to verify that Equations (A),(B),(23),(22),(32),(10),(9),(30),(18),(19),(D), and (25) capture all the possible cases. ◀

▶ **Proposition 8.** *Equation* (12) *can be derived in* QC.

Proof. Using the simplification principle (Proposition 6), one can turn Equation (12) into an equivalent equation whose circuits contain only one  . We conclude the proof using the completeness of QC for circuits containing at most one CNot (Lemma 7). The details are given in Appendix A. ◀

▶ **Proposition 9.** *Equation* (13) *can be derived in* QC.

Proof. It turns out that we can use Equation (12) to derive Equation (13) in QC. The derivation is given in Appendix A. ◀

▶ **Proposition 10.** *Equation* ($\mathrm{K}^*_{\mathrm{old}}$) *can be derived in* QC.

Proof. We show that for semantic reasons, we have either the angle δ_9 in ($\mathrm{K}^*_{\mathrm{old}}$) in $\{0, \pi\}$, or $\delta_2 = \delta_3 = \delta_5 = \delta_6 = 0$. When $\delta_9 = 0$, Equation ($\mathrm{K}^*_{\mathrm{old}}$) can be trivially derived from Equation (K^*). Otherwise, Equations (K^*) and ($\mathrm{K}^*_{\mathrm{old}}$) can be transformed into each other using elementary properties of multi-controlled gates. Moreover, these transformations induce a bijection between the 8-tuples of angles δ_j corresponding to the RHS of the instances of Equation (K^*) and the 9-tuples corresponding to the RHS of the instances of Equation ($\mathrm{K}^*_{\mathrm{old}}$), so that the uniqueness of the δ_j in Equation (K^*) follows from the uniqueness in Equation ($\mathrm{K}^*_{\mathrm{old}}$). ◀

▶ **Theorem 11** (Completeness). *The equational theory* QC*, defined in Figure 2, is complete for* $\mathbf{QC}$*-circuits, i.e. for any* $\mathbf{QC}$*-circuits* C_1, C_2*, if* $[\![C_1]\!] = [\![C_2]\!]$ *then* $\mathrm{QC} \vdash C_1 = C_2$.

Proof. All the rules of the complete equational theory introduced in [12] that are not in QC are provable in QC: Equations (8), (9), (10), (11) are proved in Appendix A, Equations (12), (13) and ($\mathrm{K}^*_{\mathrm{old}}$) are proved in Propositions 8, 9 and 10 respectively. ◀

3 Quantum circuits for isometries

In this section we consider a first standard extension of the vanilla quantum circuits which consists in allowing qubit initialisation in a specific state, namely in the $|0\rangle$-state.

▶ **Definition 12.** *Let* $\mathbf{QC}_{\mathrm{iso}}$ *be the prop generated by* $\varphi : 0 \to 0$, $-\boxed{H}- : 1 \to 1$, $-\boxed{P(\varphi)}- : 1 \to 1$,  $: 2 \to 2$ *and* $\vdash : 0 \to 1$ *for any* $\varphi \in \mathbb{R}$.

▶ **Definition 13** (Semantics). *We extend the semantics* $[\![\cdot]\!]$ *of vanilla quantum circuits (Definition 2) with* $[\![\vdash]\!] = |0\rangle$.

▶ **Proposition 14** (Universality). *Any isometry*[7] $V : \mathbb{C}^{\{0,1\}^n} \to \mathbb{C}^{\{0,1\}^m}$ *can be realised by a* $\mathbf{QC}_{\text{iso}}$*-circuit* $C : n \to m$ *s.t.* $[\![C]\!] = V$.

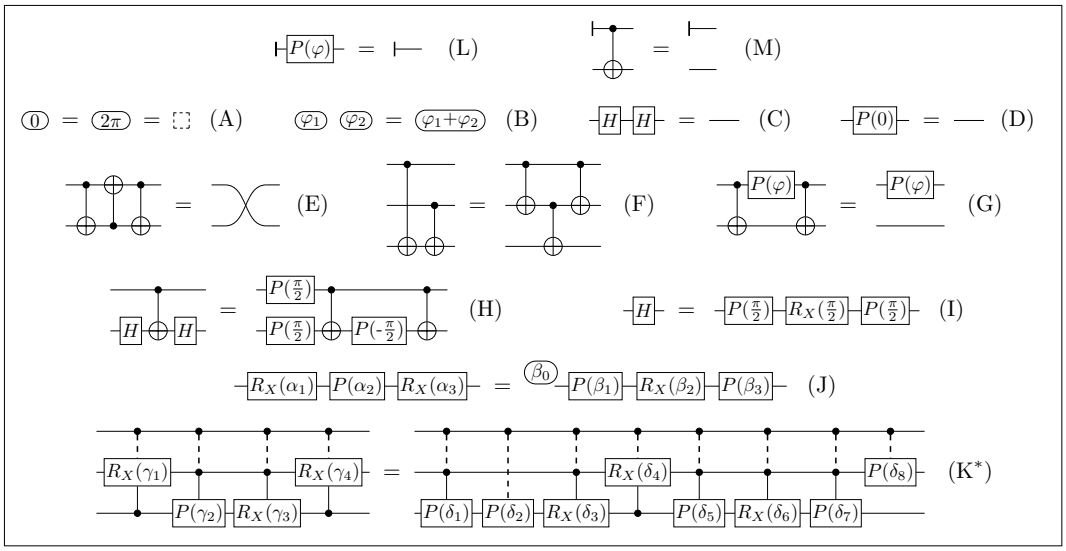


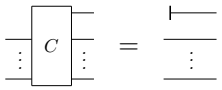
Figure 4 Equational theory QC_{iso}. It contains all the equations of QC together with Equation (L) (defined for any $\varphi \in \mathbb{R}$) and Equation (M), which are new equations governing the behaviour of the new generator $\vdash$.

For instance, the so-called copies in the standard basis ($|x\rangle \mapsto |xx\rangle$) and in the diagonal basis can be respectively represented as follows:

We consider the equational theory QC_{iso}, given in Figure 4, which is nothing but the equational theory QC augmented with the following two sound equations:

Viewing $P(\varphi)$ as a control-global-phase gate, Equations (L), (M) can be interpreted as instances of the following property: a control gate can be removed when one of its control qubit is initialised in the $|0\rangle$-state. This kind of properties can actually be derived within QC_{iso}.

▶ **Lemma 15.** *Let* C *be a* $\mathbf{QC}_{\text{iso}}$*-circuit such that* $\forall |\varphi\rangle \in \mathbb{C}^{2^n}$, $[\![C]\!]\,|\varphi\rangle = |0\rangle \otimes |\varphi\rangle$. *Then:*



Proof. We prove that the above circuit necessarily is a **QC**-circuit together with a single qubit initialisation. The semantics of the **QC**-circuit forces it to be equivalent to a controlled circuit, which can be shown to be deletable by the qubit initialisation, thanks to Equations (L) and (M). ◀

[7] An isometry is a linear map V s.t. $V^\dagger \circ V$ is the identity.

A direct corollary of Lemma 15 is the completeness of $\mathrm{QC}_{\mathrm{iso}}$ for quantum circuits with at most one initialisation. Notice that one can then use Lemma 17 of [45] to essentially prove the completeness of $\mathrm{QC}_{\mathrm{iso}}$. However, as the semantics in [45] is based on CPTP maps rather than isometries (so global phases should be treated carefully), and moreover the proof of this Lemma 17 is not described, we provide a direct completeness proof of $\mathrm{QC}_{\mathrm{iso}}$ in the following.

To do so, we may want to generalise Lemma 15 to any number of qubit initialisations. However, the proof does not generalise. Indeed, it relies on the fact that, semantically, the vanilla circuit of which we initialize a single qubit is necessarily of the form $\mathrm{diag}(I, U)$, with I and U of the same dimension, so we can start with a circuit implementing U and control each of its gates to get a circuit implementing $\mathrm{diag}(I, U)$ with only controls and phases on the control wire. To generalise this notion to more than one qubit initialisation, where semantically we would need to implement $\mathrm{diag}(I, U)$ with U of dimensions larger than I's, we need a finer-grain decomposition of said matrix. We hence resort to the following unitary decomposition:

▶ **Lemma 16.** *Let* $U = \left(\begin{array}{c|c|c} I & 0 & 0 \\ \hline 0 & U_{00} & U_{01} \\ \hline 0 & U_{10} & U_{11} \end{array}\right) \begin{array}{l} \}k \\ \}n-k \\ \}n \end{array}$ *be unitary with* U_{00} *and* U_{11} *square. Then, there exist:*

- A_0, A_1, B_0, B_1 *unitary,*
- $C = \mathrm{diag}(c_1, ..., c_d)$ *and* $S = \mathrm{diag}(s_1, ..., s_d)$ *($c_i, s_i \geq 0$ and $d \leq n-k$).*

such that:

- $C^2 + S^2 = I$
- $U = \left(\begin{array}{c|c|c} I & 0 & 0 \\ \hline 0 & A_0 & 0 \\ \hline 0 & 0 & A_1 \end{array}\right) \left(\begin{array}{c|c|c|c} I & 0 & 0 & 0 \\ \hline 0 & C & 0 & -S \\ \hline 0 & 0 & I & 0 \\ \hline 0 & S & 0 & C \end{array}\right) \left(\begin{array}{c|c|c} I & 0 & 0 \\ \hline 0 & B_0 & 0 \\ \hline 0 & 0 & B_1 \end{array}\right)$

The above decomposition is a variation on the *Cosine-Sine Decomposition* (CSD) [40], which has already appeared to be useful in quantum circuit synthesis [44].

Proof. The proof itself is a variation of the proof for the usual CSD. It specifically involves the so-called RQ and SVD decompositions. ◀

It is then possible to show the completeness of $\mathrm{QC}_{\mathrm{iso}}$:

▶ **Theorem 17** (Completeness). *The equational theory* $\mathrm{QC}_{\mathrm{iso}}$*, defined in Figure 4, is complete for* $\mathbf{QC}_{\mathrm{iso}}$*-circuits.*

Proof. The proof goes by showing that deriving equality between two $\mathbf{QC}_{\mathrm{iso}}$-circuits amounts to generalising Lemma 15 to any number of qubit initialisations, which is shown inductively using the above variation of the CSD. ◀

4 Quantum circuits with ancillae

In this section, we consider quantum circuits which are implementing unitary maps (or isometries) using ancillary qubits, a.k.a. ancillae, as additional work space. To represent quantum circuits with ancillae, we not only need to be able to initialise fresh qubits, but also to release qubits when they become useless. Note that to guarantee that the overall evolution is an isometry, one can only release a qubit in the $|0\rangle$-state.

To encompass the notion of ancillary qubits we extend $\mathbf{QC}_{\text{iso}}$-circuits (already equipped with qubit initialisation $\vdash$) with a qubit removal generator denoted $\dashv$. Because of the constraint that removed qubits must be in the $|0\rangle$-state, we define the language of quantum circuits with ancillae in two steps.

▶ **Definition 18.** *Let* $\mathbf{QC}_{\text{pre-ancilla}}$ *be the prop generated by* $\textcircled{\varphi} : 0 \to 0$, $\boxed{H} : 1 \to 1$, $\boxed{P(\varphi)} : 1 \to 1$, $\oplus : 2 \to 2$, $\vdash : 0 \to 1$ *and* $\dashv : 1 \to 0$ *for any* $\varphi \in \mathbb{R}$.

▶ **Definition 19** (Semantics). *We extend the semantics* $[\![\cdot]\!]$ *of quantum circuits for isometries (Definition 13) with* $[\![\dashv]\!] = \langle 0|$.

Notice that the semantics of a $\mathbf{QC}_{\text{pre-ancilla}}$-circuit is not necessarily an isometry as $[\![\dashv]\!]$ is not isometric.[8] As a consequence, we define $\mathbf{QC}_{\text{ancilla}}$ as the subclass of $\mathbf{QC}_{\text{pre-ancilla}}$-circuits with an isometric semantics:

▶ **Definition 20.** *Let* $\mathbf{QC}_{\text{ancilla}}$ *be the sub-prop of* $\mathbf{QC}_{\text{pre-ancilla}}$*-circuit* C *such that* $[\![C]\!]$ *is an isometry.*

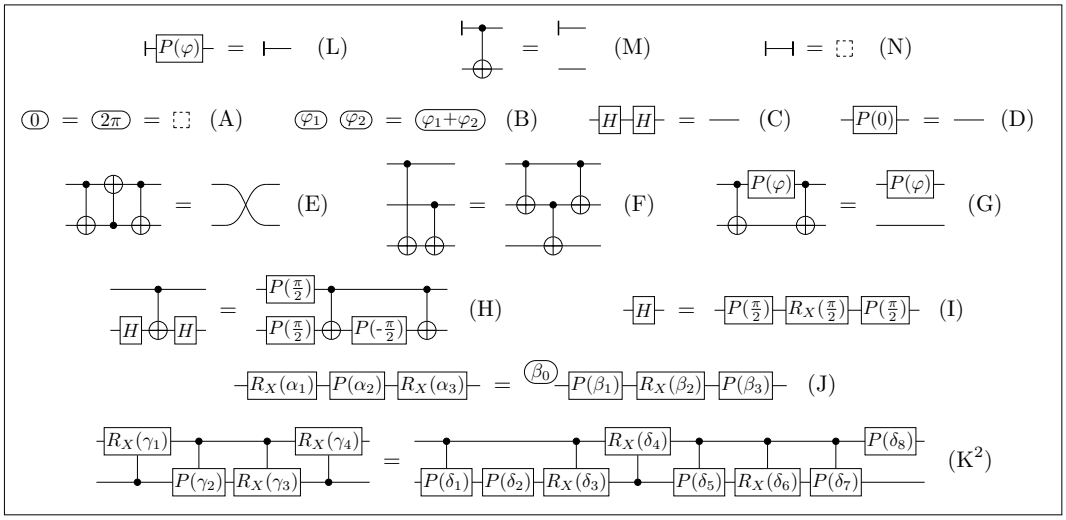
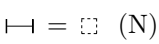


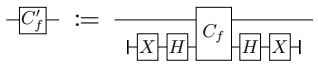
Figure 5 Equational theory $\text{QC}_{\text{ancilla}}$. It contains all the equations of QC_{iso} where Equation (K*) has been replaced by Equation (K²), together with Equation (N), which is a new equation that allows one to create ancillae.

Notice in particular that any $\mathbf{QC}_{\text{iso}}$-circuit is in $\mathbf{QC}_{\text{ancilla}}$, which implies the universality of $\mathbf{QC}_{\text{ancilla}}$ for isometries. We equip $\mathbf{QC}_{\text{ancilla}}$-circuits with the equational theory $\text{QC}_{\text{ancilla}}$ given in Figure 5, which is nothing but the equational theory QC_{iso} where Equation (K*) is replaced by its 2-qubit version Equation (K²), together with a new elementary equation (N) governing the behaviour of the qubit removal generator $\dashv$.

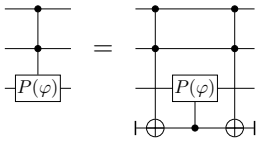


[8] Actually any linear map L s.t. $L^\dagger L \sqsubseteq I$ can be implemented by a $\mathbf{QC}_{\text{pre-ancilla}}$-circuit, where $\sqsubseteq$ is the Löwner partial order. Thus $\mathbf{QC}_{\text{pre-ancilla}}$ can be seen as a language for postselected quantum computations.

Quantum circuits with ancillae form a standard model of quantum computing. They are for instance used in the context of quantum oracles: given a circuit $C_f : n+1 \to n+1$ whose semantics is $|x, y\rangle \mapsto |x, y \oplus f(x)\rangle$ for some boolean function f, one can implement the corresponding phase oracle C'_f whose semantics is $|x\rangle \mapsto (-1)^{f(x)} |x\rangle$ as follows:



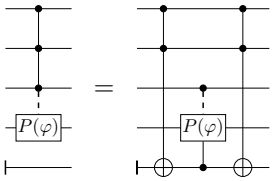
Quantum circuits with ancillae are also extensively used in the context of circuit parallelisation, as one can decrease the depth of a quantum circuit by adding ancillary qubits [37]. Finally, ancillary qubits can be used to provide an alternative realisation of multi-controlled gates, for instance a 3-qubit controlled gate can be implemented using an ancillary qubit, Toffoli gates, and the 2-qubit version of the gate:



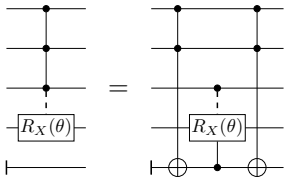
(14)

This can be generalised to any multi-controlled gates with at least two control qubits:

▶ **Proposition 21.** *The following two equations can be derived in* $\mathrm{QC}_{\mathrm{ancilla}}$.



(15)



(16)

Proof. By induction on the number of qubits. ◀

Notice that Equations (15) and (16) are actually derivable in $\mathrm{QC}_{\mathrm{iso}}$. However, in order to provide an alternative inductive definition of multi-control gates (like in Equation (14)), it requires the presence of at least one fresh qubit which can always be created in the context of quantum circuits with ancillae thanks to Equation (N).

Thanks to the alternative representation of multi-controlled gates, one can derive, in $\mathrm{QC}_{\mathrm{ancilla}}$, the equation (K*) for any arbitrary number of controlled qubits:

▶ **Proposition 22.** *Equation* (K*) *can be derived in* $\mathrm{QC}_{\mathrm{ancilla}}$.

Proof. Let (K^n) be Equation (K*) acting on n qubits for any $n \geq 2$. Equation (K^2) is in $\mathrm{QC}_{\mathrm{ancilla}}$. We first prove that (K^3) can be derived from (K^2) by defining the Fredkin gate (or controlled-swap gate) and by pushing the two last wires of the LHS circuit of (K^3) into two fresh ancillae, which allow us to apply (K^2) and reverse the construction to get the RHS circuit of (K^3). This technique is not applicable in the general case for any circuit because if the Fredkin gates are not triggered, it could be the case that the gates pushed into the ancillae do not release the ancillae into the $|0\rangle$-state. The key observation is that this is possible for (K^3) as every involved gates are either phase gate or uniquely controlled gate (which both act as identity on the $|0\rangle$-state). Then, we prove that (K^n) is derivable in $\mathrm{QC}_{\mathrm{ancilla}}$ for any $n \geq 4$ by induction on n using the alternative definition of multi-controlled gates (Proposition 21), which allows us to construct an instance of the LHS circuit of (K^{n-1}) from the LHS circuit of (K^n). ◀

We are now ready to prove the completeness of $\mathrm{QC}_{\mathrm{ancilla}}$:

▶ **Theorem 23** (Completeness). *The equational theory* $\mathrm{QC}_{\mathrm{ancilla}}$*, defined in Figure 5, is complete for* $\mathbf{QC}_{\mathrm{ancilla}}$*-circuits.*

Proof. Proposition 22 implies that for any $\dashv$-free circuits C_1, C_2, if $\mathrm{QC}_{\mathrm{iso}} \vdash C_1 = C_2$ then $\mathrm{QC}_{\mathrm{ancilla}} \vdash C_1 = C_2$. Using deformation of circuits, any $\mathbf{QC}_{\mathrm{ancilla}}$-circuit $C : n \to m$ can be written  , where $C' : n \to m + k$ is a $\mathbf{QC}_{\mathrm{iso}}$-circuit. Since both $[\![C]\!]$ and $[\![C']\!]$ are isometries and $[\![C]\!] = (Id \otimes \langle 0^k|)\,[\![C']\!]$, we have $[\![C']\!] = [\![C]\!] \otimes |0^k\rangle$. Given two $\mathbf{QC}_{\mathrm{ancilla}}$-circuits C_1, C_2 s.t. $[\![C_1]\!] = [\![C_2]\!]$, let $C_1' : n \to m + k$, and $C_2' : n \to m + \ell$ be the corresponding $\mathbf{QC}_{\mathrm{iso}}$-circuits. W.l.o.g. assume $k \leq \ell$, and pad C_1' with $\ell - k$ qubit initialisations: $C_1'' := C_1' \otimes (\vdash)^{\otimes \ell - k}$. We have $[\![C_1'']\!] = [\![C_2']\!]$, so by completeness of $\mathrm{QC}_{\mathrm{iso}}$, $\mathrm{QC}_{\mathrm{ancilla}} \vdash C_1'' = C_2'$, so $\mathrm{QC}_{\mathrm{ancilla}} \vdash C_1 \otimes (\ \mapsto\!\dashv)^{\otimes \ell - k} = C_2$. It suffices to apply Equation (N) to obtain $\mathrm{QC}_{\mathrm{ancilla}} \vdash C_1 = C_2$. ◀

5 Quantum circuits with discard for completely positive map

The last extension considered in this paper is the addition of a discard operator which consists in tracing out qubits. Contrary to quantum circuits with ancillae, any qubit can be discarded whatever its state is. Discarding a qubit is depicted by $\dashv$.

▶ **Definition 24.** *Let* $\mathbf{QC}_{\mathrm{discard}}$ *be the prop generated by* $-\boxed{H}- : 1 \to 1$, $-\boxed{P(\varphi)}- : 1 \to 1$, $: 2 \to 2$, $\vdash : 0 \to 1$ *and* $\dashv : 1 \to 0$ *for any* $\varphi \in \mathbb{R}$.

The ability to discard qubits implies that the evolution represented by such a circuit is not pure anymore. As a consequence the semantics is a completely positive trace-preserving (CPTP) map acting on density matrices (trace 1 positive semi-definite Hermitian matrices). Formally the new semantics is defined as follows:

▶ **Definition 25** (Semantics). *For any quantum* $\mathbf{QC}_{\mathrm{discard}}$*-circuit* $C : n \to m$*, let* $(\!|C|\!) : \mathcal{M}_{2^n,2^n}(\mathbb{C}) \to \mathcal{M}_{2^m,2^m}(\mathbb{C})$ *be the* semantics *of* C *inductively defined as the linear map* $(\!|C_2 \circ C_1|\!) = (\!|C_2|\!) \circ (\!|C_1|\!)$*;* $(\!|C_1 \otimes C_2|\!) = (\!|C_1|\!) \otimes (\!|C_2|\!)$*;* $(\!|\dashv|\!) = \rho \mapsto tr(\rho)$ *and for any other generator* g*,* $(\!|g|\!) = \rho \mapsto [\![g]\!]\,\rho\,[\![g]\!]^\dagger$*, where* $tr(M)$ *is the trace of the matrix* M *and* $M^\dagger$ *its adjoint.*

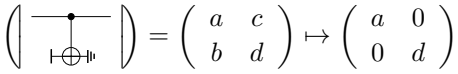
Notice that the global phase generator $\circledcirc$ is not part of the prop anymore. If it were, its interpretation would be $(\!|\circledcirc|\!) = \rho \mapsto [\![\circledcirc]\!]\,\rho\,[\![\circledcirc]\!]^\dagger = e^{i\varphi}\rho e^{-i\varphi} = \rho$, which is the same as that of the empty circuit. Thus, for this model the X-rotation can simply be defined as $-\boxed{R_X(\theta)}- \ := \ -\boxed{H}\boxed{P(\theta)}\boxed{H}-$ (the same definition as Figure 1 but without the global phase).

▶ **Proposition 26** (Universality). $\mathbf{QC}_{\mathrm{discard}}$ *is universal for CPTP maps.*

Proof. According to the Stinespring dilation lemma [46], any CPTP map $F : \mathcal{M}_{2^n,2^n}(\mathbb{C}) \to \mathcal{M}_{2^m,2^m}(\mathbb{C})$ can be purified as an isometry $V : \mathbb{C}^{2^n} \to \mathbb{C}^{2^{m+k}}$ such that for any ρ, $F(\rho) = tr_k(V\rho V^\dagger)$, where $tr_k(.)$ is the partial trace of the last k qubits. By universality of $\mathbf{QC}_{\mathrm{iso}}$ there exists a circuit C such that $[\![C]\!] = V$. Let C' be the global-phase-free version of C, thus $[\![C']\!] = e^{i\theta}V$. Seen as a $\mathbf{QC}_{\mathrm{discard}}$-circuit, C' has the semantics $(\!|C'|\!) = \rho \mapsto (e^{i\theta}V)\rho(e^{i\theta}V)^\dagger = V\rho V^\dagger$. Discarding the last k qubits of C' leads to a $\mathbf{QC}_{\mathrm{discard}}$-circuit implementing F. ◀

The new generator and new semantics allow us to model measurements. For instance, the standard basis measurement can be obtained via:

Indeed we recover the semantics of the standard basis measurement:



The output wire can be interpreted as a classical bit (encoded in a quantum bit), a (resp. d) being the probability to be 0 (resp. 1).

One can also encode classical gates, for instance the AND gate using Toffoli:

With the promise that the input is classical, i.e. the input density matrix is $\mathrm{diag}(p_{00}, p_{01}, p_{10}, p_{11})$ (where p_{xy} is the probability for the input to be in the state $xy \in \{0,1\}^2$), the output state is $\mathrm{diag}(p_{00}+p_{01}+p_{10}, p_{11})$ which corresponds to the behaviour of the AND gate.

More generally, one can represent classically controlled computation using the $\mathbf{QC}_{\mathrm{discard}}$-circuits, allowing to reason on fault-tolerant computations, error correcting codes and measurement-based quantum computation for instance.

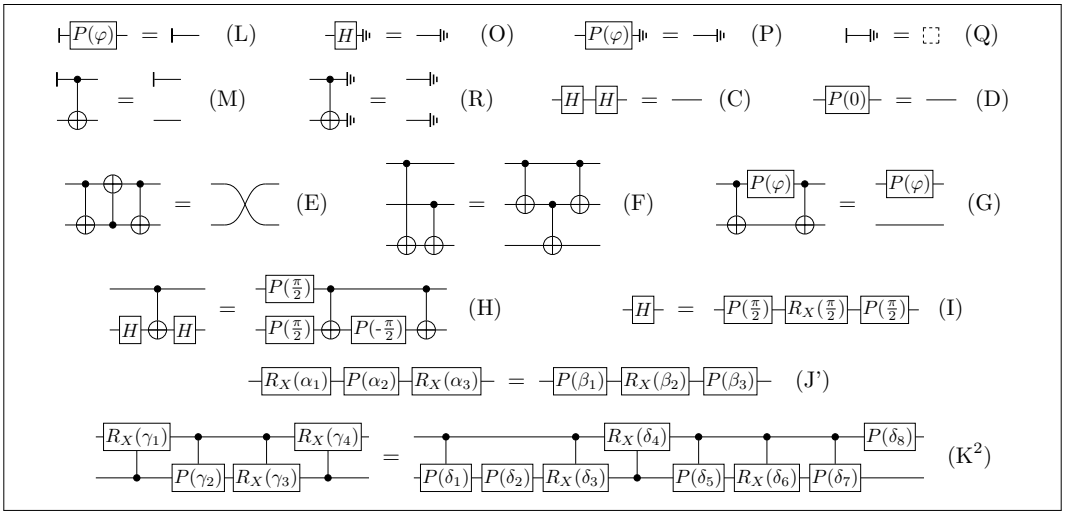
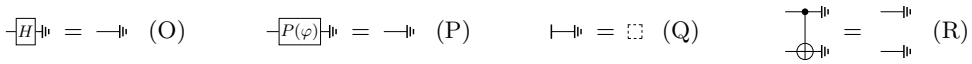


Figure 6 Equational theory $\mathrm{QC}_{\mathrm{discard}}$. It contains all the equations of $\mathrm{QC}_{\mathrm{ancilla}}$ except Equations (A), (B), (N) and where Equation (J) has been replaced by its global-phase free-version Equation (J'), together with Equations (O), (P) (defined for any $\varphi \in \mathbb{R}$), (Q) and (R), which are new equations governing the behaviour of the new generator ⊣⊩.

While [45] provides a way to get completeness for quantum circuits with measurements from a complete one for isometries, we instead use [9] which provides a similar result but for isometries with discard, as the latter is a little bit more atomic than measurements. This leads us to equip $\mathbf{QC}_{\mathrm{discard}}$-circuits with the equational theory $\mathrm{QC}_{\mathrm{discard}}$ defined in Figure 6, which is a global-phase-free version of $\mathrm{QC}_{\mathrm{ancilla}}$ where ⊣⊩ replaces ⊣, and with the addition of:



This observation allows us in particular to transport all the proofs using $\mathrm{QC}_{\mathrm{ancilla}}$ into the present theory, the only two differences being that ⊣ plays the role of ⊣ and that the $\mathrm{QC}_{\mathrm{discard}}$ version of the proofs have no global phase Ⓟ.

▶ **Theorem 27** (Completeness). *The equational theory* $\mathrm{QC}_{\mathrm{discard}}$*, defined in Figure 6, is complete for* $\mathbf{QC}_{\mathrm{discard}}$*-circuits.*

Proof. We can use the *discard construction* [9] to build $\mathrm{QC}_{\mathrm{iso}}^{\dashv}$ from $\mathrm{QC}_{\mathrm{iso}}$, by adding equation:

$$\dashv^{\otimes m} \circ U = \dashv^{\otimes n} \tag{17}$$

for any $\mathbf{QC}_{\mathrm{iso}}$-circuit $U : n \to m$. The discard construction guarantees that $\mathrm{QC}_{\mathrm{iso}}^{\dashv}$ is complete for CPTP maps (Proposition 2 in [9]). It remains to prove that all equations in $\mathrm{QC}_{\mathrm{iso}}^{\dashv}$ derive from those of $\mathrm{QC}_{\mathrm{discard}}$. All equations of the former except Equations (K*) and (17) appear in $\mathrm{QC}_{\mathrm{discard}}$. Those are trivially derivable. As mentioned above, it is possible to prove (K*) from $\mathrm{QC}_{\mathrm{discard}}$ exactly as in the case of $\mathrm{QC}_{\mathrm{ancilla}}$ by replacing each occurrence of ⊢⊣ = ⬚ by ⊢⊣ = ⬚. This means all the equations of $\mathrm{QC}_{\mathrm{iso}}$ are derivable. Finally, all the equations $\dashv^{\otimes m} \circ U = \dashv^{\otimes n}$ for different isometries U can be derived from Equations (O), (P), (Q), and (R). ◀

6 Concluding remarks

We have simplified the complete equational theory for quantum circuits, and provided ones for standard extensions of quantum circuits, including qubit initialisation, ancillae, and/or qubit discarding. The equational theory can be simplified in these more general settings, leading in particular to equations acting on a bounded number of qubits, avoiding the use of controlled gates on arbitrary number of qubits. It is interesting to notice that increasing the expressive power of the model makes the equational theory simpler.

This simplification of the equational theory is a step towards a minimal equational theory (i.e. an equational theory where each equation provably cannot be derived from the other ones). Notice that based on the present work, a minimal complete equational theory for vanilla quantum circuits has been introduced recently [10], showing in particular that equations acting on an unbounded number of qubits are necessary for vanilla quantum circuits. The question of the minimality in the context of quantum circuits with qubit initialisation, ancillae, and/or qubit discarding remains open.

Getting rid of Equation (K*) eases also the practical implementation of the rewriting rules as it avoids to consider a family of rules acting on an unbounded of qubits. Notice that regarding practical considerations, various equations presented in this paper have parameters, e.g. $-R_X(\alpha_1)-P(\alpha_2)-R_X(\alpha_3)- = -P(\beta_1)-R_X(\beta_2)-P(\beta_3)-$ that should be read as follows: for any angle α_i on the LHS, there exist β_j on the RHS so that the equation holds. β_j can be computed using fairly simple trigonometric operations. Notice that even if the equation looks non-symmetric, one can show conversely that for any β_j there exist α_i angles such that the equation holds (see Equation (21)).

References

1 Matthew Amy, Jianxin Chen, and Neil J. Ross. A finite presentation of CNOT-dihedral operators. *Electronic Proceedings in Theoretical Computer Science*, 266:84–97, February 2018. doi:10.4204/eptcs.266.5.

2 Atul Singh Arora, Andrea Coladangelo, Matthew Coudron, Alexandru Gheorghiu, Uttam Singh, and Hendrik Waldner. Quantum depth in the random oracle model. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 1111–1124. ACM, 2023. doi:10.1145/3564246.3585153.

3 Miriam Backens, Aleks Kissinger, Hector Miller-Bakewell, John van de Wetering, and Sal Wolffs. Completeness of the ZH-calculus. *Compositionality*, 5, July 2023. doi:10.32408/compositionality-5-5.

4 Miriam Backens, Hector Miller-Bakewell, Giovanni de Felice, Leo Lobski, and John van de Wetering. There and back again: A circuit extraction tale. *Quantum*, 5:421, March 2021. doi:10.22331/q-2021-03-25-421.

5 Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A. Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Physical Review A*, 52(5):3457–3467, November 1995. doi:10.1103/physreva.52.3457.

6 Xiaoning Bian and Peter Selinger. Generators and relations for 2-qubit Clifford+T operators. *Proceedings of QPL'22, Electronic Proceedings in Theoretical Computer Science*, 394:13–28, November 2023. doi:10.4204/eptcs.394.2.

7 Xiaoning Bian and Peter Selinger. Generators and relations for 3-qubit Clifford+CS operators. *Proceedings QPL'23, Electronic Proceedings in Theoretical Computer Science*, 384:114–126, August 2023. doi:10.4204/eptcs.384.7.

8 Robert I. Booth and Titouan Carette. Complete ZX-Calculi for the Stabiliser Fragment in Odd Prime Dimensions. In Stefan Szeider, Robert Ganian, and Alexandra Silva, editors, *47th International Symposium on Mathematical Foundations of Computer Science (MFCS 2022)*, volume 241 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 24:1–24:15, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.MFCS.2022.24.

9 Titouan Carette, Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. Completeness of graphical languages for mixed state quantum mechanics. *ACM Transactions on Quantum Computing*, 2(4), December 2021. doi:10.1145/3464693.

10 Alexandre Clément, Noé Delorme, and Simon Perdrix. Minimal equational theories for quantum circuits, 2023. arXiv:2311.07476.

11 Alexandre Clément, Nicolas Heurtel, Shane Mansfield, Simon Perdrix, and Benoît Valiron. LO_v-Calculus: A Graphical Language for Linear Optical Quantum Circuits. In Stefan Szeider, Robert Ganian, and Alexandra Silva, editors, *47th International Symposium on Mathematical Foundations of Computer Science (MFCS 2022)*, volume 241 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 35:1–35:16, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.MFCS.2022.35.

12 Alexandre Clément, Nicolas Heurtel, Shane Mansfield, Simon Perdrix, and Benoît Valiron. A complete equational theory for quantum circuits. In *Logic in Computer Science (LICS)*, 2023.

13 Alexandre Clément, Noé Delorme, Simon Perdrix, and Renaud Vilmart. Quantum circuit completeness: Extensions and simplifications, 2023. arXiv:2303.03117.

14 Robin Cockett and Cole Comfort. The category TOF. In Peter Selinger and Giulio Chiribella, editors, *Proceedings 15th International Conference on Quantum Physics and Logic, QPL 2018*, volume 287 of *EPTCS*, pages 67–84, 2019.

15 Robin Cockett, Cole Comfort, and Priyaa Srinivasan. The category CNOT. In Peter Selinger and Giulio Chiribella, editors, *Proceedings 15th International Conference on Quantum Physics and Logic, QPL 2018*, volume 287 of *EPTCS*, pages 258–293, 2019. doi:10.4204/EPTCS.266.18.

16 Bob Coecke and Quanlong Wang. ZX-rules for 2-qubit Clifford+T quantum circuits. In *International Conference on Reversible Computation*, pages 144–161. Springer, 2018.

17 Vincent Danos, Elham Kashefi, Prakash Panangaden, and Simon Perdrix. Extended measurement calculus. *Semantic techniques in quantum computation*, pages 235–310, 2009.

18 Niel de Beaudrap, Xiaoning Bian, and Quanlong Wang. Fast and Effective Techniques for T-Count Reduction via Spider Nest Identities. In Steven T. Flammia, editor, *15th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2020)*, volume 158 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 11:1–11:23, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.TQC.2020.11.

19 D. Deutsch. Quantum computational networks. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 425(1868):73–90, 1989.

20 Amar Hadzihasanovic. *The algebra of entanglement and the geometry of composition*. PhD thesis, University of Oxford, 2017.

21 Amar Hadzihasanovic, Kang Feng Ng, and Quanlong Wang. Two complete axiomatisations of pure-state qubit quantum computing. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '18, pages 502–511, New York, NY, USA, 2018. ACM. doi:10.1145/3209108.3209128.

22 Atsuya Hasegawa and François Le Gall. An optimal oracle separation of classical and quantum hybrid schemes. In Sang Won Bae and Heejin Park, editors, *33rd International Symposium on Algorithms and Computation, ISAAC 2022, December 19-21, 2022, Seoul, Korea*, volume 248 of *LIPIcs*, pages 6:1–6:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ISAAC.2022.6.

23 Mathieu Huot and Sam Staton. Universal properties in quantum theory. In Peter Selinger and Giulio Chiribella, editors, *Proceedings of the 15th International Conference on Quantum Physics and Logic, Halifax, Canada, 3-7th June 2018*, volume 287 of *Electronic Proceedings in Theoretical Computer Science*, pages 213–223, 2019. doi:10.4204/EPTCS.287.12.

24 Toshinari Itoko, Rudy Raymond, Takashi Imamichi, and Atsushi Matsuo. Optimization of quantum circuit mapping using gate transformation and commutation. *Integration*, 70:43–50, 2020.

25 Kazuo Iwama, Yahiko Kambayashi, and Shigeru Yamashita. Transformation rules for designing CNOT-based quantum circuits. In *Proceedings of the 39th annual Design Automation Conference*, pages 419–424, 2002.

26 Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. A complete axiomatisation of the ZX-calculus for Clifford+T quantum mechanics. In Anuj Dawar and Erich Grädel, editors, *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 559–568. ACM, 2018. doi:10.1145/3209108.3209131.

27 Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. Diagrammatic reasoning beyond Clifford+T quantum mechanics. In Anuj Dawar and Erich Grädel, editors, *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 569–578. ACM, 2018. doi:10.1145/3209108.3209139.

28 Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. A generic normal form for ZX-diagrams and application to the rational angle completeness. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*, pages 1–10. IEEE, 2019. doi:10.1109/LICS.2019.8785754.

29 Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. Completeness of the ZX-Calculus. *Logical Methods in Computer Science*, Volume 16, Issue 2, June 2020. doi:10.23638/LMCS-16(2:11)2020.

30 Aleks Kissinger and John van de Wetering. PyZX, 2018. URL: https://github.com/Quantomatic/pyzx.

31 Aleks Kissinger and John van de Wetering. Reducing the number of non-Clifford gates in quantum circuits. *Phys. Rev. A*, 102:022406, August 2020. doi:10.1103/PhysRevA.102.022406.

32 Stephen Lack. Composing PROPs. In *Theory and Applications of Categories*, volume 13(9), pages 147–163, 2004. URL: http://www.tac.mta.ca/tac/volumes/13/9/13-09abs.html.

33 Justin Makary, Neil J. Ross, and Peter Selinger. Generators and relations for real stabilizer operators. In Chris Heunen and Miriam Backens, editors, *Proceedings of the 18th International Conference on Quantum Physics and Logic, QPL 2021*, volume 343 of *EPTCS*, pages 14–36, 2021. doi:10.4204/EPTCS.343.2.

34 Dmitri Maslov, Gerhard W Dueck, D Michael Miller, and Camille Negrevergne. Quantum circuit simplification and level compaction. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(3):436–444, 2008.

35 Dmitri Maslov, Christina Young, D Michael Miller, and Gerhard W Dueck. Quantum circuit simplification using templates. In *Design, Automation and Test in Europe*, pages 1208–1213. IEEE, 2005.

36 D Michael Miller, Dmitri Maslov, and Gerhard W Dueck. A transformation based algorithm for reversible logic synthesis. In *Proceedings of the 40th annual Design Automation Conference*, pages 318–323, 2003.

37 Cristopher Moore and Martin Nilsson. Parallel quantum computation and quantum codes. *SIAM journal on computing*, 31(3):799–815, 2001.

38 Yunseong Nam, Neil J Ross, Yuan Su, Andrew M Childs, and Dmitri Maslov. Automated optimization of large quantum circuits with continuous parameters. *npj Quantum Information*, 4(1):1–12, 2018.

39 Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2002.

40 C.C. Paige and M. Wei. History and generality of the CS decomposition. *Linear Algebra and its Applications*, 208-209:303–326, 1994. doi:10.1016/0024-3795(94)90446-4.

41 Boldizsár Poór, Quanlong Wang, Razin A. Shaikh, Lia Yeh, Richie Yeung, and Bob Coecke. Completeness for arbitrary finite dimensions of ZXW-calculus, a unifying calculus. In *LICS*, pages 1–14, 2023. doi:10.1109/LICS56636.2023.10175672.

42 André Ranchin and Bob Coecke. Complete set of circuit equations for stabilizer quantum mechanics. *Physical Review A*, 90(1):012109, 2014.

43 Robert Raussendorf and Hans J Briegel. A one-way quantum computer. *Physical review letters*, 86(22):5188, 2001.

44 Vivek Shende, Stephen Bullock, and Igor Markov. Synthesis of quantum logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25, 2006-01-31 00:01:00 2006. URL: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=150894.

45 Sam Staton. Algebraic effects, linearity, and quantum programming languages. In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '15, pages 395–406, New York, NY, USA, 2015. Association for Computing Machinery. doi:10.1145/2676726.2676999.

46 W Forrest Stinespring. Positive functions on c^*-algebras. *Proceedings of the American Mathematical Society*, 6(2):211–216, 1955.

47 Renaud Vilmart. A near-minimal axiomatisation of ZX-calculus for pure qubit quantum mechanics. In *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–10, 2019. doi:10.1109/LICS.2019.8785765.

A Derivations of the equations of $\mathrm{QC}_{\mathrm{old}}$

In this appendix, we derive the equations of $\mathrm{QC}_{\mathrm{old}}$ that are not in QC, namely Equations (8),(9),(10),(11),(12) and (13) (the proof of Equation ($\mathrm{K}^*_{\mathrm{old}}$) can be found in the full version of the present paper [13]). In addition, we prove some other useful equations (depicted

in Figure 7) that are used as intermediate steps. Note that QC contains all equations of $\mathrm{QC}_{\mathrm{old}}$ acting on at most one qubit, thus QC is complete for one-qubit circuits and all equations of Figure 7 on one qubit are derivable. The proofs of the remaining equations are given below.

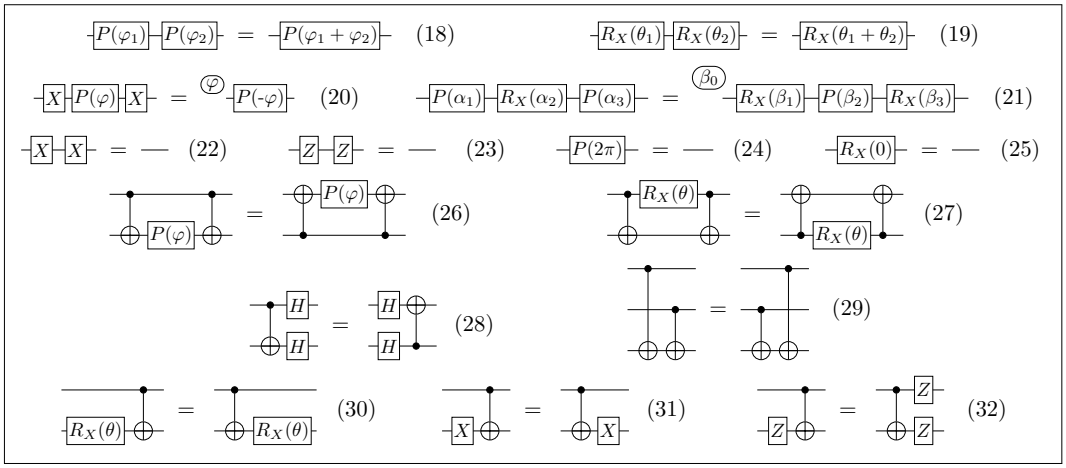
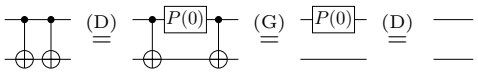


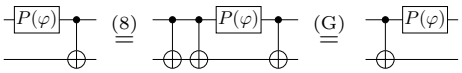
Figure 7 Useful intermediate equations.

Proof of Equation (8).



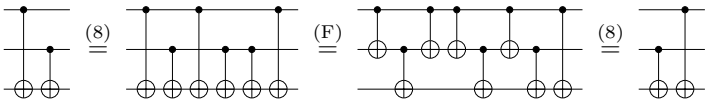
◀

Proof of Equation (9).



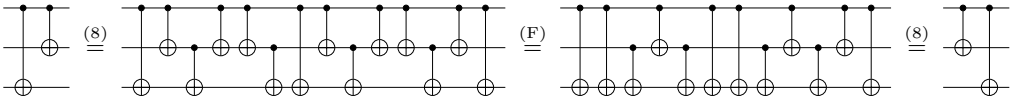
◀

Proof of Equation (29).



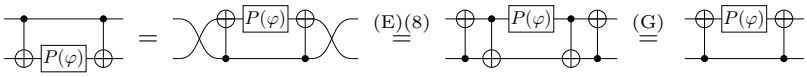
◀

Proof of Equation (11).



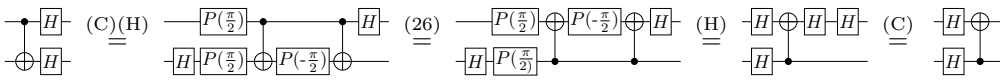
◀

Proof of Equation (26).



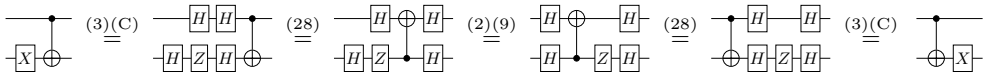
◀

Proof of Equation (28).



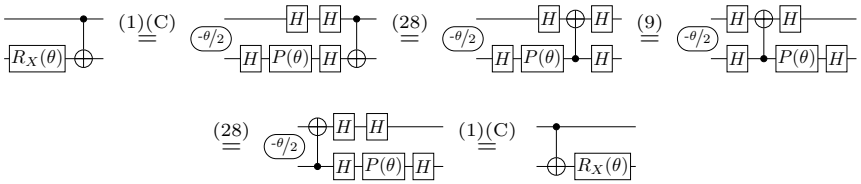
◀

Proof of Equation (31).



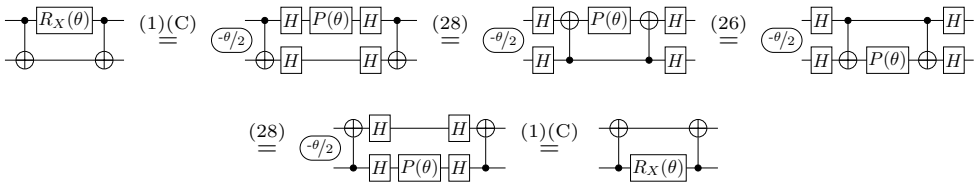
◀

Proof of Equation (30).



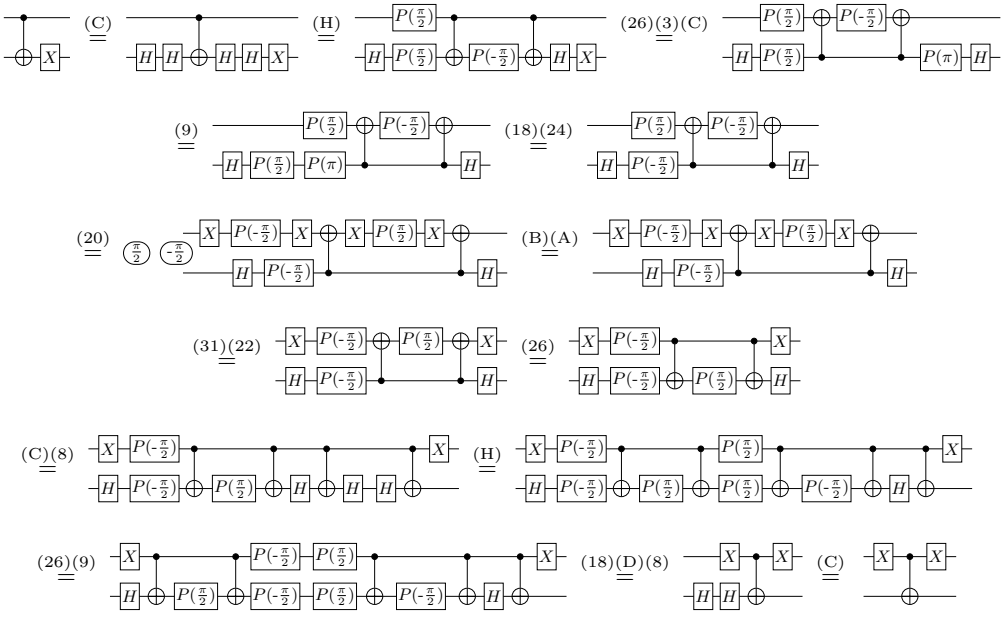
◀

Proof of Equation (27).



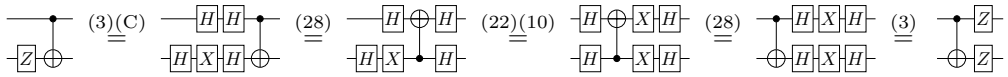
◀

Proof of Equation (10).

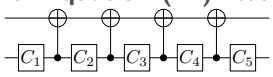
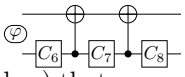
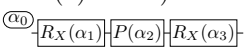
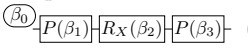


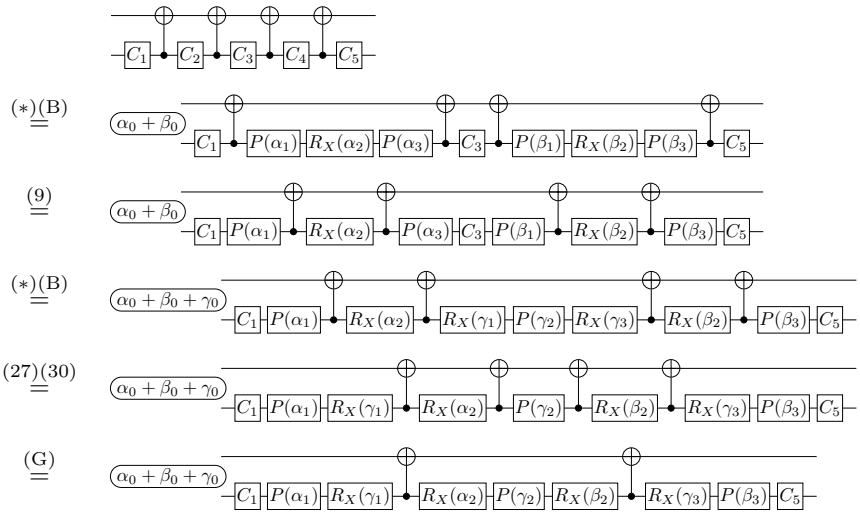
◀

Proof of Equation (32).

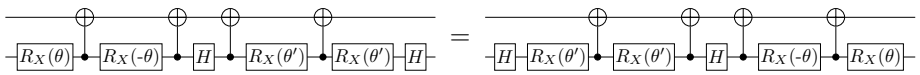


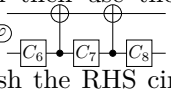
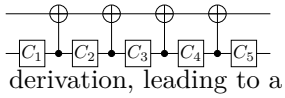
◀

Proof of Equation (12). We first show that any circuits (containing four CNot gates) of the form  can always be transformed in QC into a circuit (containing only two CNot gates) of the form  where denotes a one-qubit circuit. This uses the fact (referenced $(*)$ below) that any one-qubit circuit can be transformed in QC into a circuit of the form  or  (by the completeness of QC for one-qubit circuits and the well-know Euler-decomposition). The derivation goes as follows.

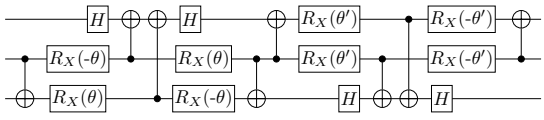


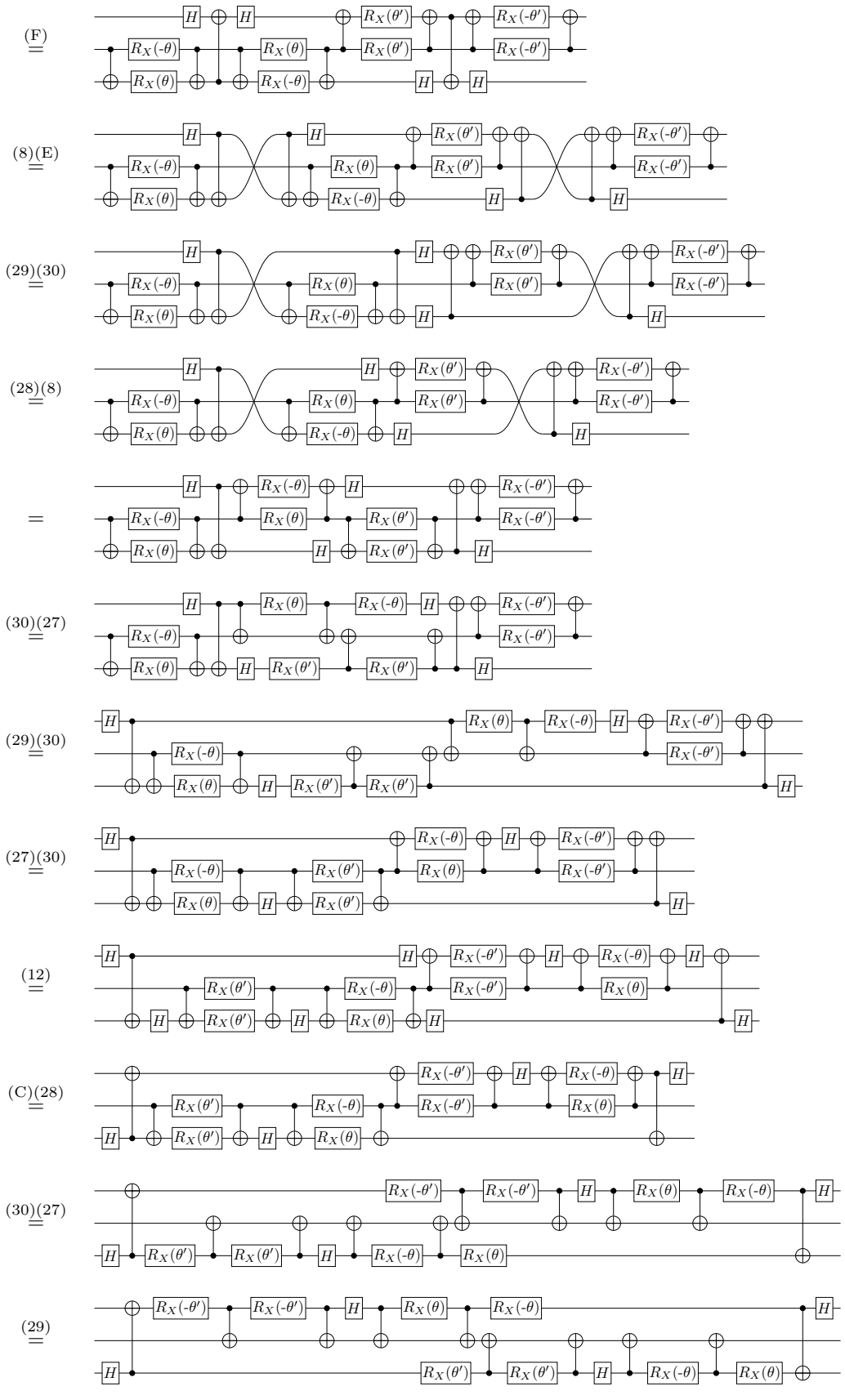
Then, using Equations (30) and (27), Equation (12) becomes



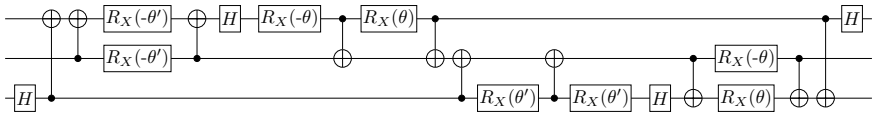
We can then use the above derivation to transform both circuits into circuits of the form  . Then, by using the simplification principle (Proposition 6), we can push the RHS circuit to the end of the LHS circuit, which leads to a circuit of the form  (up to a global phase) from which we can apply again the above derivation, leading to a circuit containing only two CNot gates. Then, by using the simplification principle again, we can turn the equation into an equivalent equation with only one CNot on both sides. Finally, the completeness of QC for circuits containing at most one CNot gate (Lemma 7) concludes the proof. ◀

Proof of Equation (13).

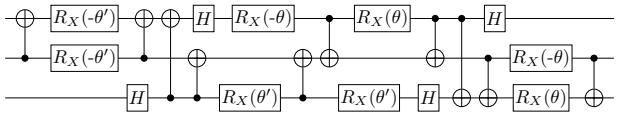



(F) =
(8)(E) =
(29)(30) =
(28)(8) =
=
(30)(27) =
(29)(30) =
(27)(30) =
(12) =
(C)(28) =
(30)(27) =
(29) =

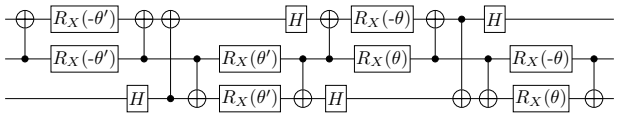
(30)(27) =



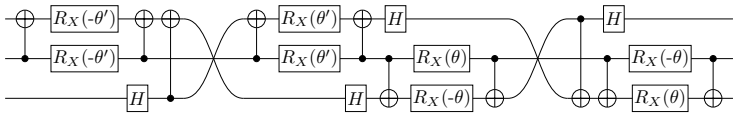
(29)(30) =



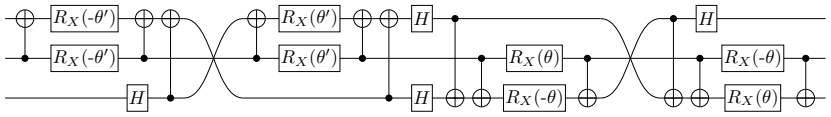
(30)(27) =



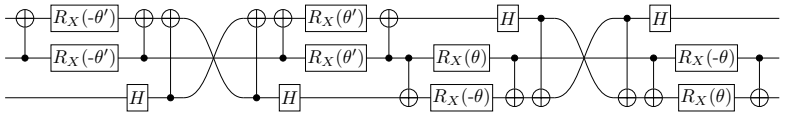
=



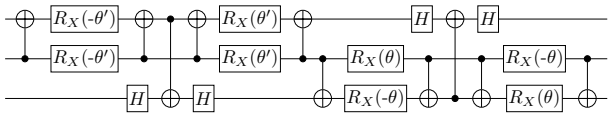
(8)(28) =



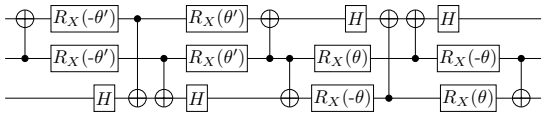
(29)(30) =



(8)(E) =



(F) =



◀

Reverse Tangent Categories

Geoffrey Cruttwell
Mount Allison University, Sackville, Canada

Jean-Simon Pacaud Lemay[1]
Macquarie University, Sydney, Australia

Abstract

Previous work has shown that reverse differential categories give an abstract setting for gradient-based learning of functions between Euclidean spaces. However, reverse differential categories are not suited to handle gradient-based learning for functions between more general spaces such as smooth manifolds. In this paper, we propose a setting to handle this, which we call *reverse tangent categories*: tangent categories with an involution operation for their differential bundles.

2012 ACM Subject Classification Theory of computation → Categorical semantics

Keywords and phrases Tangent Categories, Reverse Tangent Categories, Reverse Differential Categories, Categorical Machine Learning

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.21

Funding *Geoffrey Cruttwell*: Partially funded by an NSERC Discovery Grant.
Jean-Simon Pacaud Lemay: For this research, this author was funded by an NSERC PDF (456414649), an ARC DECRA (DE230100303), & an AFOSR Research Grant (FA9550-24-1-0008).

Acknowledgements The authors would like to thank Bryce Clark for pointing out [13], which ties in nicely with the story of this paper, as well as Geoff Vooys for useful discussions, answering questions, and helping find references regarding Example 31.

1 Introduction

This paper is a direct follow-up to the paper "Reverse Differential Categories", published in CSL 2020 [5], and continues a tradition of developing categorical structures to help understand and work with ideas from differential calculus in computer science, and specifically here in relation to machine learning and automatic differentiation [2, 3, 5, 7, 8, 10, 22].

Initial work on categorical formulations of differential structures in computer science focused on the so-called "forward" derivative. Given a map $f : A \to B$, the forward derivative is an operation that sends tangent vectors in A to tangent vectors in B. While there are several different (but related) categorical formulations for the forward derivative, the relevant ones for this paper are *Cartesian differential categories* [2] and *tangent categories* [3].

Cartesian differential categories formalize differential calculus over Euclidean spaces. A Cartesian differential category (Ex 3) comes equipped with a *differential combinator* D, which is an operation that for any map $f : A \to B$, produces a map $\mathsf{D}[f] : A \times A \to B$, called the derivative of f. Various axioms are then demanded of D which enforce the properties of ordinary differentiation, such as the chain rule, symmetry of mixed partial derivatives, etc. Intuitively, one considers an input $(a, v) \in A \times A$ as a point a and a tangent vector v to a, and so the derivative $\mathsf{D}[f]$ produces a tangent vector to $f(a)$ in B.

However, the definition of a Cartesian differential category assumes that a tangent vector to a point is of the same type as the point. While this is true for Euclidean spaces, this is certainly not true for more general spaces such as arbitrary smooth manifolds. To capture

[1] Corresponding Author

the operation of the forward derivative for such spaces, one can instead work with tangent categories, which formalize differential calculus over smooth manifolds. A tangent category (Def 1) in particular comes equipped with an endofunctor T, where we think of $\mathsf{T}(A)$ as the collection of all tangent vectors to points of an object A. As such, we may interpret $\mathsf{T}(A)$ as the abstract tangent bundle of A. For a map $f : A \to B$, the associated map $\mathsf{T}(f) : \mathsf{T}(A) \to \mathsf{T}(A)$ is interpreted as an operation which takes a tangent vector in A to a tangent vector in B. This tangent bundle functor also comes equipped with various additional structure that captures ordinary properties of differentiation – for example, functoriality of T corresponds to the chain rule. Tangent categories are a direct generalization of Cartesian differential categories which allow one to work with the forward derivative in more general settings.

However, many areas of computer science such as automatic differentiation and gradient-based learning make much more extensive use of the "reverse" derivative. Given a map $f : A \to B$, the reverse derivative is an operation which sends tangent vectors in B to tangent vectors in A. The reverse derivative is much more computationally efficient for maps between Euclidean spaces in which the domain space is much larger than the codomain space – which is the typical case in machine learning scenarios. The paper "Reverse Derivative Categories" [5] introduced *Cartesian reverse differential categories*, which provide a categorical abstraction of the reverse derivative operation over Euclidean spaces. This time, a Cartesian reverse differential category (Ex 28) comes equipped with a *reverse differential combinator* R which now takes a map $f : A \to B$ and produces a map of type $\mathsf{R}[f] : A \times B \to A$, called the reverse derivative of f. Intuitively, one thinks of $\mathsf{R}[f]$ as taking a point of the domain and a tangent vector of the *codomain* and returning a tangent vector of the domain at the point.

However, Cartesian reverse differential categories suffer from the same problem as their forward counterpart: they assume that the tangent vectors of a point in the space are in 1-1 correspondence with points of the space itself. The objective of this paper is to do what tangent categories did for Cartesian differential categories: introduce a setting where one has a reverse derivative operation, but in such a way that tangent vectors are not assumed to be the same as points of the space itself. As such, the main contribution of this paper is the introduction of *reverse tangent categories* (Sec 3), the reverse differential counterpart of tangent categories.

	Euclidean Spaces	Manifolds
Forward derivative	Cartesian differential category	Tangent category
Reverse derivative	Cartesian reverse differential category	**Reverse tangent category**

How does one go about defining a reverse tangent category? Our first attempt at defining "reverse tangent categories" was similar to a Cartesian reverse differential category, that is, trying to give a direct description of what this structure should look like in terms of a reverse differentiation operation. However, this has proven difficult (see Remark 26). So, instead, here we take a different approach. Usefully, Cartesian reverse differential categories have an alternative characterization. A Cartesian reverse differential category is precisely a Cartesian differential category equipped with a "linear dagger", which is an involution operation on linear maps. This linear dagger $\dagger$ is an operation which takes a map of type $f : C \times A \to B$ which is "linear in A" and transposes the linear argument to produce a map $f^\dagger : C \times B \to A$ which is now "linear in B". From this point of view, the reverse differential combinator is the transpose of the forward differential combinator, that is, $\mathsf{R}[f] := \mathsf{D}[f]^\dagger$. Using this approach as a guide, we define a reverse tangent category as a tangent category with a suitable notion of involution.

What form should such an involution for a reverse tangent category take? One way to look at the dagger operation of a Cartesian reverse differential category is via fibrations. Indeed, the dagger operation can be viewed as an operation which goes from the canonical "linear fibration" [5, 7] of a Cartesian differential category to its dual fibration. The notion of a dual fibration [14] is an operation which takes a fibration and returns another fibration in which the fibre over A is the opposite category of the fibre over A from the original fibration. The analog of the linear fibration in a tangent category is a fibration of differential bundles [4] – these abstract the notion of smooth vector bundles from ordinary differential geometry. Thus, we define a reverse tangent category (Def 24) to consist of a tangent category equipped with an involution operation which goes from a fibration of differential bundles (Def 16) to its dual fibration (Prop 21).

With this definition in hand, we then (i) give examples of such structure including smooth manifolds (Ex 27), but also examples from algebra (Ex 30) and algebraic geometry (Ex 31), (ii) show precisely how this definition relates to Cartesian reverse differential categories (Ex 28 & Prop 42), and (iii) provide some theoretical results about reverse tangent categories (Sec 4). Thus, the purpose of this paper is to properly introduce reverse tangent categories; we do not here consider applications of reverse tangent categories to gradient-based learning on (smooth) manifolds, or the relationship of these ideas to differential programming languages (for reverse cartesian differential categories, however, see [8]). That said, just as the original paper on Cartesian reverse differential categories inspired work on the use of such structures in gradient-based learning for Euclidean spaces [8, 10, 22], reverse tangent categories should provide a suitable setting to do the same for smooth manifolds (or any other "differential" setting).

Conventions. In an arbitrary category, we write objects as capital letters A, B, etc. and maps with lower letters as $f : A \to B$. We denote identity maps as $1_A : A \to A$, and, following the conventions of previous differential/tangent category papers, we write composition diagrammatically, that is, the composite of $f : A \to B$ and $g : B \to C$ is denoted $fg : A \to C$.

2 Forward Tangent Categories and Differential Bundles

In this section, we review the basics of tangent categories including the definition, some key examples, and differential bundles, which will play an important role in this paper.

▶ **Definition 1** ([3, Def 2.3]). *A **tangent structure** on a category $\mathbb{X}$ is a sextuple $\mathbb{T} := (\mathsf{T}, \mathsf{p}, \mathsf{s}, \mathsf{z}, \ell, \mathsf{c})$ consisting of:*

- **(i)** *An endofunctor $\mathsf{T} : \mathbb{X} \to \mathbb{X}$, called the **tangent bundle functor**;*
- **(ii)** *A natural transformation $\mathsf{p}_A : \mathsf{T}(A) \to A$, called the **projection**, such that for each $n \in \mathbb{N}$, the n-fold pullback[2] of p_A exists, denoted as $\mathsf{T}_n(A)$ with projections $\rho_j : \mathsf{T}_n(A) \to \mathsf{T}(A)$, and such that for all $m \in \mathbb{N}$, T^m preserves these pullbacks, that is, $\mathsf{T}^m(\mathsf{T}_n(A))$ is the n-fold pullback of $\mathsf{T}^m(\mathsf{p}_A)$ with projections $\mathsf{T}^m(\rho_j)$;*
- **(iii)** *A natural transformation[3] $\mathsf{s}_A : \mathsf{T}_2(A) \to \mathsf{T}(A)$, called the **sum**;*
- **(iv)** *A natural transformation $\mathsf{z}_A : A \to \mathsf{T}(A)$, called the **zero map**;*
- **(v)** *A natural transformation $\ell_A : \mathsf{T}(A) \to \mathsf{T}^2(A)$, called the **vertical lift**;*
- **(vi)** *A natural transformation $\mathsf{c}_A : \mathsf{T}^2(A) \to \mathsf{T}^2(A)$, called the **canonical flip**;*

*such that the equalities and universal property in [3, Def 2.3] are satisfied. A **tangent category** is a pair $(\mathbb{X}, \mathbb{T})$ consisting of a category $\mathbb{X}$ equipped with a tangent structure $\mathbb{T}$.*

[2] By convention, $\mathsf{T}_0(A) = A$ and $\mathsf{T}_1(A) = \mathsf{T}(A)$

[3] Note that by the universal property of the pullback, we can define functors $\mathsf{T}_n : \mathbb{X} \to \mathbb{X}$.

Let us briefly provide some intuition for the definition of a tangent category. Tangent categories formalize the properties of the tangent bundle on smooth manifolds from classical differential geometry. As such, an object A in a tangent category can be interpreted as a base space, and $\mathsf{T}(A)$ as its abstract tangent bundle. For maps, $\mathsf{T}(f)$ is interpreted as the differential of f, and the functoriality of T represents the chain rule. The projection p_A is the analogue of the natural projection from the tangent bundle to its base space, making $\mathsf{T}(A)$ an abstract fibre bundle over A. The sum s_A and the zero z_A make $\mathsf{T}(A)$ into an additive bundle over A; that is, a commutative monoid in the slice category[4] over A. To explain the vertical lift, recall that in differential geometry, the double tangent bundle (i.e. the tangent bundle of the tangent bundle) admits a canonical sub-bundle called the vertical bundle which is isomorphic to the tangent bundle. The vertical lift ℓ_A is an analogue of the embedding of the tangent bundle into the double tangent bundle via the vertical bundle. The vertical lift also satisfies a universal property, which is essential to generalize important properties of the tangent bundle from differential geometry. Lastly, the canonical flip c_A is an analogue of the smooth involution of the same name on the double tangent bundle, and its naturality captures the symmetry of mixed partial derivatives ($\frac{\partial f}{\partial x \partial y} = \frac{\partial f}{\partial y \partial x}$). For more details and intuition on tangent categories, see [3, Sec 2.5].

We now recall the main examples of tangent categories we will use throughout the paper. In particular, Ex 2 (which is arguably the canonical example of a tangent category) and Ex 4 directly relate tangent categories to differential geometry and differential calculus, Ex 5 (one of the main examples in Rosický's original paper [19, Ex 2]) provides a link to commutative algebra, and Ex 6 relates tangent categories to algebraic geometry. For other examples of tangent categories, see [4, Ex 2.2].

▶ **Example 2.** Let SMAN be the category whose objects are smooth manifolds and whose maps are smooth functions. For a smooth manifold M and a point $x \in M$, let $\mathsf{T}_x(M)$ be the tangent space to M at x, and recall that the tangent bundle of M is the smooth manifold $\mathsf{T}(M)$ which is the (disjoint) union of each tangent space:

$$\mathsf{T}(M) := \bigsqcup_{x \in M} \mathsf{T}_x(M)$$

So in local coordinates, elements of $\mathsf{T}(M)$ can be described as pair $(x, \vec{v}) \in \mathsf{T}(M)$ consisting of a point $x \in M$ and a tangent vector $\vec{v}$ at x. Now for a smooth function $f : M \to N$ and a point $x \in M$, there is an induced linear map $\mathsf{T}_x(f) : \mathsf{T}_x(M) \to \mathsf{T}_{f(x)}(N)$ called the derivative of f at x. This induces a functor $\mathsf{T} : \mathsf{SMAN} \to \mathsf{SMAN}$ which sends a smooth manifold M to its tangent bundle $\mathsf{T}(M)$, and a map $f : M \to N$ to the map $\mathsf{T}(f) : \mathsf{T}(M) \to \mathsf{T}(N)$, locally defined as:

$$\mathsf{T}(f)(x, \vec{v}) = (f(x), \mathsf{T}_x(f)(\vec{v}))$$

To describe the rest of the tangent structure, in local coordinates, elements of $\mathsf{T}_2(M)$ are triples $(x, \vec{v}, \vec{w}) \in \mathsf{T}_2(M)$ where $x \in M$ and $\vec{v}, \vec{w} \in \mathsf{T}_x(M)$, while elements of $\mathsf{T}^2(M)$ can be represented as quadruples $(x, \vec{v}, \vec{w}, \vec{u}) \in \mathsf{T}^2(M)$. Thus the natural transformations are defined as follows in local coordinates:

$$\mathsf{p}_M(x, \vec{v}) = x \qquad \mathsf{s}_M(x, \vec{v}, \vec{w}) = (x, \vec{v} + \vec{w}) \qquad \mathsf{z}_M(x) = (x, \vec{0})$$
$$\ell_M(x, \vec{v}) = (x, \vec{0}, \vec{0}, \vec{v}) \qquad \mathsf{c}_M(x, \vec{v}, \vec{w}, \vec{u}) = (x, \vec{w}, \vec{v}, \vec{u})$$

So $(\mathsf{SMAN}, \mathbb{T})$ is a tangent category [4, Ex 2.2.i].

[4] Commutative monoids in the slice category are also called additive bundles [3, Sec 2.1].

▶ **Example 3.** Every Cartesian differential category is a tangent category [3, Prop 4.7]. Where tangent categories formalize differential calculus over smooth manifolds, Cartesian differential categories instead formalize differential calculus over Euclidean spaces. Briefly, a Cartesian differential category [2, Def 2.1.1] is a category $\mathbb{X}$ with finite products (where we denote the binary product by $\times$, projections π_j, and pairing operator $\langle -, - \rangle$) which comes equipped with a **differential combinator** D which associates to every map $f : A \to B$ a map $\mathsf{D}[f] : A \times A \to B$, called the derivative of f. The differential combinator D satisfies seven axioms which are analogues of fundamental identities of the derivative, such as the chain rule, linearity of the derivative, etc. The canonical tangent functor of a Cartesian differential category is defined as follows:

$$\mathsf{T}(A) := A \times A \qquad\qquad \mathsf{T}(f) := \langle \pi_0 f, \mathsf{D}[f] \rangle$$

and where the rest of the tangent structure is defined in [3, Prop 4.7]. For more details on Cartesian differential categories, as well as examples, see [2, 5].

▶ **Example 4.** As an explicit example of a Cartesian differential category, let SMOOTH be the category whose objects are the Euclidean real vector spaces $\mathbb{R}^n$, and whose maps are smooth functions $F : \mathbb{R}^n \to \mathbb{R}^m$ between them. SMOOTH is a Cartesian differential category, where for a smooth function $F = \langle f_1, \ldots, f_m \rangle : \mathbb{R}^n \to \mathbb{R}^m$, its derivative $\mathsf{D}[F] : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^m$ is defined as the m-tuple of sums of the partial derivatives:

$$\mathsf{D}[F](\vec{x}, \vec{y}) := \left\langle \sum_{i=1}^{n} \frac{\partial f_1}{\partial x_i}(\vec{x}) y_i, \ldots, \sum_{i=1}^{n} \frac{\partial f_n}{\partial x_i}(\vec{x}) y_i \right\rangle$$

Then by the previous example, SMOOTH is a tangent category; its tangent structure is precisely the one for smooth manifolds in Ex 2, but restricted to the Euclidean spaces $\mathbb{R}^n$.

▶ **Example 5.** Let k be a commutative ring, and k-CALG be the category of commutative k-algebras. For a commutative k-algebra A, we denote its algebra of dual numbers as:

$$A[\epsilon] = \{a + b\epsilon |\ a, b \in A, \epsilon^2 = 0\}$$

Then, borrowing notation from [9], we have a functor $\mathbb{L} : k\text{-}\mathsf{CALG} \to k\text{-}\mathsf{CALG}$ which is defined on objects as $\mathbb{L}(A) := A[\epsilon]$ and for a map $f : A \to B$, $\mathbb{L}(f) : A[\epsilon] \to B[\epsilon]$ is defined as:

$$\mathbb{L}(f)(a + b\epsilon) = f(a) + f(b)\epsilon$$

This gives us our tangent bundle and so $(k\text{-}\mathsf{CALG}, \mathbb{L})$ is a tangent category, where the remaining tangent structure is defined in [9, Sec 3.1]. We also note that this example can be generalized: for any symmetric monoidal category with distributive finite biproducts, its category of commutative monoids is a tangent category by generalizing the dual numbers construction.

▶ **Example 6.** For a commutative ring k, the category of affine schemes over k is a tangent categories [9, Sec 4]. Famously, the category of affine schemes over k is equivalent to k-CALG^{op}, so we may describe the tangent structure in terms of commutative k-algebras. For a commutative k-algebra A, we denote its module of Kähler differentials as $\Omega(A)$. Then define the "fibré tangente" (French for tangent bundle) of A, or tangent algebra of A [15, Section 2.6], as the free symmetric A-algebra over its modules of Kähler differentials

$$\mathsf{T}(A) := \mathsf{Sym}_A(\Omega(R)) = \bigoplus_{n=0}^{\infty} \Omega(A)^{\otimes_A^s n} = A \oplus \Omega(A) \oplus (\Omega(A) \otimes_A^s \Omega(A)) \oplus \ldots$$

where $\bigoplus$ is the coproduct of modules, and $\otimes^s_R$ is the symmetrized tensor product over A. Equivalently, $\mathsf{T}(A)$ is the A-algebra generated by the set $\{\mathsf{d}(a) |\ a \in R\}$ modulo the equations:

$$\mathsf{d}(1) = 0 \qquad \mathsf{d}(a+b) = \mathsf{d}(a) + \mathsf{d}(b) \qquad \mathsf{d}(ab) = a\mathsf{d}(b) + b\mathsf{d}(a)$$

which are the same equations that are modded out to construct the module of Kähler differentials of A. Thus, an arbitrary element of $\mathsf{T}(A)$ is a finite sum of monomials of the form $a\mathsf{d}(b_1)\dots\mathsf{d}(b_n)$, and so the algebra structure of $\mathsf{T}(A)$ is essentially the same as polynomials. This also induces a functor $\mathsf{T} : k\text{-}\mathsf{CALG} \to k\text{-}\mathsf{CALG}$ which maps a commutative k-algebra to its tangent algebra $\mathsf{T}(A)$, and a k-algebra morphism $f : A \to B$ to the k-algebra morphism $\mathsf{T}(f) : \mathsf{T}(A) \to \mathsf{T}(B)$ defined on generators as follows:

$$\mathsf{T}(f)(a) = f(a) \qquad \mathsf{T}(f)(\mathsf{d}(a)) = \mathsf{d}(f(a))$$

This gives us our tangent bundle and so $(k\text{-}\mathsf{CALG}^{op}, \mathbb{T})$ is a tangent category, where the remaining tangent structure is defined in [9, Sec 4.1].

There are many concepts from differential geometry which one can define in an arbitrary tangent category. The concept that plays a crucial role in the definition of reverse tangent categories is that of a **differential bundle**, which generalizes the idea of a smooth vector bundle from differential geometry.

▶ **Definition 7** ([4, Def 2.3]). *In a tangent category $(\mathbb{X}, \mathbb{T})$, a **differential bundle** is a quadruple $\mathcal{E} = (\mathsf{q} : E \to A, \sigma : E_2 \to E, \zeta : A \to E, \lambda : E \to \mathsf{T}(E))$ consisting of:*

- **(i)** *Objects A and E of $\mathbb{X}$;*
- **(ii)** *A map $\mathsf{q} : E \to A$ of $\mathbb{X}$, called the **projection**, such that for each $n \in \mathbb{N}$, the pullback of n copies of q exists; we denote this pullback as E_n with n projection maps $\pi_j : E_n \to E$, and for all $m \in \mathbb{N}$, T^m preserves these pullbacks;*
- **(iii)** *A map $\sigma : E_2 \to E$ of $\mathbb{X}$, called the **sum**;*
- **(iv)** *A map $\zeta : A \to E$ of $\mathbb{X}$, called the **zero**;*
- **(v)** *A map $\lambda : E \to \mathsf{T}(E)$ of $\mathbb{X}$, called the **lift**;*

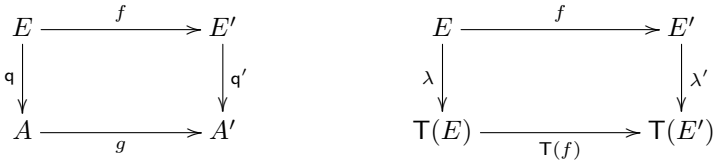
*such that the equalities and universal property in [4, Def 2.3] are satisfied. When there is no confusion, differential bundles will be written as $\mathcal{E} = (\mathsf{q} : E \to A, \sigma, \mathsf{z}, \lambda)$, and when the objects are specified simply as $\mathcal{E} = (\mathsf{q}, \sigma, \zeta, \lambda)$. If $\mathcal{E} = (\mathsf{q} : E \to A, \sigma, \mathsf{z}, \lambda)$ is a differential bundle, we also say that $\mathcal{E}$ is a **differential bundle over A**.*

If $\mathcal{E} = (\mathsf{q} : E \to A, \sigma, \mathsf{z}, \lambda)$ is a differential bundle, the object A is interpreted as a base space and the object E as the total space. The projection q is the analogue of the bundle projection from the total space to the base space, making E an "abstract bundle over A". The sum σ and the zero ζ make each fibre into a commutative monoid. Lastly, the lift λ is an analogue of the embedding of the total space into its tangent bundle (sometimes called the *small vertical lift*).

There are two possible kinds of morphism between differential bundles: one where the base objects can vary and one where the base object is fixed. The former is used as the maps in the category of all differential bundles in the tangent category. In either case, a differential bundle morphism is asked to preserve the projections and the lifts of the differential bundles.

▶ **Definition 8** ([4, Def 2.3]). *In a tangent category $(\mathbb{X}, \mathbb{T})$,*

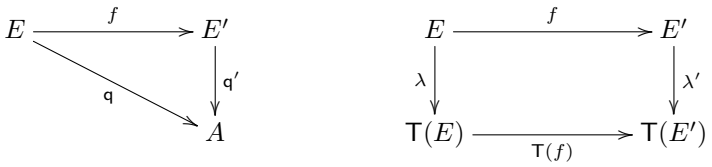
- **(i)** *A **linear differential bundle morphism** $(f, g) : \mathcal{E} = (\mathsf{q} : E \to A, \sigma, \zeta, \lambda) \to \mathcal{E}' = (\mathsf{q}' : E' \to A', \sigma', \zeta', \lambda')$ is a pair of maps $f : E \to E'$ and $g : A \to A'$ such that the following diagrams commute:*



(1)

Let $\mathsf{DBun}\,[(\mathbb{X}, \mathbb{T})]$ *be the category whose objects are differential bundles in* $(\mathbb{X}, \mathbb{T})$*, and whose maps are linear differential bundle morphisms between them.*

(ii) *A* ***linear A-differential bundle morphism*** $f : \mathcal{E} = (\mathsf{q} : E \to A, \sigma, \zeta, \lambda) \to \mathcal{E}' = (\mathsf{q}' : E' \to A, \sigma', \zeta', \lambda')$ *is a map* $f : E \to E'$ *such that* $(f, 1_A) : \mathcal{E} \to \mathcal{E}'$ *is a differential bundle morphism, that is, the following diagrams commute:*



(2)

Let $\mathsf{DBun}\,[A]$ *denote the subcategory of differential bundles over* A *and linear* A*-differential bundle morphisms between them.*

One does not need to assume that linear differential bundle morphisms preserve the sum and zero since, surprisingly, this follows from preserving the lift [4, Prop 2.16]. Other properties of differential bundle morphisms can be found in [4, Sec 2.5]. Here are some examples of differential bundles and morphisms between them:

▶ **Example 9.** In a tangent category $(\mathbb{X}, \mathbb{T})$, for any object A, its tangent bundle is a differential bundle over A, that is, $\mathcal{T}(A) := (\mathsf{p}_A : \mathsf{T}(A) \to A, \mathsf{s}_A, \mathsf{z}_A, \ell_A)$ is a differential bundle over A, and for every map $f : A \to B$, $\mathcal{T}(f) := (f, \mathsf{T}(f)) : \mathcal{T}(A) \to \mathcal{T}(B)$ is a linear differential bundle morphism [4, Ex 2.4]. As such, we obtain a functor $\mathcal{T} : \mathbb{X} \to \mathsf{DBun}\,[(\mathbb{X}, \mathbb{T})]$.

▶ **Example 10.** In $(\mathsf{SMAN}, \mathbb{T})$, differential bundles over a smooth manifold M correspond precisely to smooth vector bundles over M. Briefly, recall that a smooth vector bundle, in particular, consists of smooth manifolds M, called the base space, and E, called the total space, and a smooth surjection $\mathsf{q} : E \to M$, called the projection, such that for each point $x \in M$, the fibre $E_x = \{e \in E | \mathsf{q}(e) = x\}$ is a finite-dimensional $\mathbb{R}$-vector space. A smooth vector bundle morphism from $\mathsf{q} : E \to M$ to $\mathsf{q}' : E' \to M'$ consist of two smooth functions $f : M \to M'$ and $g : E \to E'$ such that $f(\mathsf{q}(e)) = \mathsf{q}'(g(e))$ for all $e \in E$ and the induced maps $g_x : E_x \to E'_{f(x)}$ are $\mathbb{R}$-linear maps. Let SVEC be the category of smooth vector bundles and smooth vector bundle morphisms between them. Every smooth vector bundle over M gives a differential bundle over M in $(\mathsf{SMAN}, \mathbb{T})$, and vice versa. As such, there is an equivalence $\mathsf{DBun}\,[(\mathsf{SMAN}, \mathbb{T})] \simeq \mathsf{SVEC}$. For full details, see [18].

▶ **Example 11.** In general, for an arbitrary Cartesian differential category, there is not necessarily a nice full description of all differential bundles. However, there is a nice class of differential bundles which plays an important role in the story of this paper. Recall that a Cartesian differential category $\mathbb{X}$ is also a **Cartesian left additive category** [2, Definition 1.2.1], which in particular means that every homset $\mathbb{X}(A, B)$ is a commutative monoid, with binary operation $+$ and zero 0. Then for every pair of objects A and X, their product $(\pi_0 : A \times X \to A, 1_A \times (\pi_0 + \pi_1), \langle 1_A, 0\rangle, \langle \pi_0, 0, 0, \pi_1\rangle)$ is a differential bundle over A. This

generalization the notion of trivial smooth vector bundles from differential geometry. Again we stress that not all differential bundles in a Cartesian differential category are necessarily of this form. That said, for SMOOTH, every differential bundle is of this form since smooth vector bundles over a Euclidean space is a trivial bundle.

▶ **Example 12.** In $(k\text{-}\mathsf{CALG}, \mathbb{L})$, differential bundles over a commutative k-algebra A correspond precisely to A-modules [9, Thm 3.10]. Briefly, let MOD be the category whose objects are pairs (A, M) consisting of a commutative k-algebra A and an A-module M, and whose maps $(f, g) : (A, M) \to (B, N)$ consist of a k-algebra morphism $f : A \to B$ and a k-linear map $g : M \to N$ such that $g(am) = f(a)g(m)$ for all $a \in A$ and $m \in M$. Then there is an equivalence $\mathsf{DBun}\left[(k\text{-}\mathsf{CALG}, \mathbb{L})\right] \simeq \mathsf{MOD}$, which in particular sends an A-module M to its nilpotent extension:

$$M[\varepsilon] = \{a + m\varepsilon |\ a \in A, m \in M, \varepsilon^2 = 0\}$$

For full details, see [9, Sec 3].

▶ **Example 13.** In $(k\text{-}\mathsf{CALG}^{op}, \mathbb{T}^{op})$, differential bundles over a commutative k-algebra A again correspond precisely to A-modules [9, Thm 4.15]. However this time, there is an equivalence $\mathsf{DBun}\left[(k\text{-}\mathsf{CALG}^{op}, \mathbb{T}^{op})\right] \simeq \mathsf{MOD}^{op}$ (or in other words $\mathsf{DBun}\left[(k\text{-}\mathsf{CALG}^{op}, \mathbb{T}^{op})\right]^{op} \simeq \mathsf{MOD}$), which in particular sends an A-module M to the free symmetric A-algebra over M:

$$\mathsf{Sym}_A(M) = \bigoplus_{n=0}^{\infty} M^{\otimes_A^{s\, n}} = A \oplus M \oplus (M \otimes_A^s M) \oplus \dots$$

For full details, see [9, Sec 4].

We conclude this section with some results about differential bundles which we will need in the later sections. The first is that the tangent bundle of a differential bundle is also a differential bundle, and the second is that the pullback along the projection of a differential bundle is again a differential bundle.

▶ **Proposition 14** ([4, Lem 2.5]). *In a tangent category $(\mathbb{X}, \mathbb{T})$, if $\mathcal{E} = (\mathsf{q} : E \to A, \sigma, \zeta, \lambda)$ is a differential bundle, then $\overline{\mathsf{T}}(\mathcal{E}) := (\mathsf{T}(\mathsf{q}) : \mathsf{T}(E) \to \mathsf{T}(A), \mathsf{T}(\sigma), \mathsf{T}(\zeta), \mathsf{T}(\lambda)\mathsf{c}_E)$ is a differential bundle[5], which we call the tangent bundle of $\mathcal{E}$. Similarly, if $(f, g) : \mathcal{E} \to \mathcal{E}'$ is a linear differential bundle morphism, then $\overline{\mathsf{T}}(f, g) := (\mathsf{T}(f), \mathsf{T}(g)) : \overline{\mathsf{T}}(\mathcal{E}) \to \overline{\mathsf{T}}(\mathcal{E}')$ is a linear differential bundle morphism. This induces a functor $\overline{\mathsf{T}} : \mathsf{DBun}\left[(\mathbb{X}, \mathbb{T})\right] \to \mathsf{DBun}\left[(\mathbb{X}, \mathbb{T})\right]$.*

▶ **Proposition 15** ([4, Lem 2.7]). *In a tangent category $(\mathbb{X}, \mathbb{T})$, let $\mathcal{E} = (\mathsf{q} : E \to A, \sigma, \zeta, \lambda)$ be a differential bundle and $f : X \to A$ is a map such that for each $n \in \mathbb{N}$, the pullback of n copies of q along f exists, which we denote as $X \times_M E_n$ with projection maps $\pi_0 : X \times_M E_n \to X$ and $\pi_{n+1} : X \times_M E_n \to E$, and for all $m \in \mathbb{N}$, T^m preserves these pullbacks. Then $X \times_M \mathcal{E} := (\pi_0 : X \times_M E_n \to X, 1_X \times_M \sigma, 1_X \times_M \zeta, 0_X \times_M \lambda)$ is a differential bundle and $(\pi_0, \pi_1) : X \times_M \mathcal{E} \to \mathcal{E}$ is a linear differential bundle morphism.*

3 Reverse Tangent Categories

In this section, we introduce the main novel concept of this paper: *reverse* tangent categories. As explained in the introduction, the way we define reverse tangent categories is by generalizing the definition of a Cartesian reverse differential category as a Cartesian differential category

[5] It is important to note that the canonical flip is used to the define the lift for the tangent bundle of a differential bundle.

with an involution from its linear fibration to its dual fibration (which we review in Ex 28). For a tangent category, the analogue of the linear fibration is replaced by a suitable fibration of differential bundles. As such, a reverse tangent category is a tangent category with an involution from its differential bundle fibration to its dual fibration. For a review of fibrations and their basic theory, see [14].

We first wish to build a fibration of differential bundles. Unfortunately, for an arbitrary tangent category, the forgetful functor from its category of differential bundles is not necessarily a fibration (as tangent categories do not assume that the necessary pullbacks exist). As such, we will need to specify a class of differential bundles that do form a fibration – which we call a system of differential bundles.

▶ **Definition 16.** *For a tangent category* $(\mathbb{X}, \mathbb{T})$*, a* ***system of differential bundles*** *consists of a collection of differential bundles,* $\mathcal{D}$*, such that:*

- **(i)** *For every object* A*,* $\mathcal{T}(A) \in \mathcal{D}$*;*
- **(ii)** *If* $\mathcal{E} \in \mathcal{D}$*, then so is its tangent bundle* $\overline{\mathsf{T}}(\mathcal{E}) \in \mathcal{D}$ *(see Prop. 14);*
- **(iii)** *If* $\mathcal{E} \in \mathcal{D}$*, then for any map* $f : X \to A$ *in* $\mathbb{X}$*, the pullback of* $\mathcal{E}$ *along* f *exists, and* $X \times_A \mathcal{E} \in \mathcal{D}$ *(see Prop. 15).*

Let $\mathsf{DBun}_{\mathcal{D}}[(\mathbb{X}, \mathbb{T})]$ *be the full subcategory of* $\mathsf{DBun}[(\mathbb{X}, \mathbb{T})]$ *whose objects are the differential bundles in* $\mathcal{D}$*, and let* $\mathsf{DBun}_{\mathcal{D}}[A]$ *be the full subcategory of* $\mathsf{DBun}[A]$ *whose objects are the differential bundles over* A *in* $\mathcal{D}$*. We denote the forgetful functor as* $\mathsf{U}_{\mathcal{D}} : \mathsf{DBun}_{\mathcal{D}}[(\mathbb{X}, \mathbb{T})] \to \mathbb{X}$ *which maps a differential bundles to its base object.*

▶ **Proposition 17.** *If* $\mathcal{D}$ *is a system of differential bundles on a tangent category* $(\mathbb{X}, \mathbb{T})$*, then the forgetful functor* $\mathsf{U}_{\mathcal{D}} : \mathsf{DBun}_{\mathcal{D}}[(\mathbb{X}, \mathbb{T})] \to \mathbb{X}$ *is a fibration, and the fibre over an object* A *is* $\mathsf{DBun}_{\mathcal{D}}[A]$*.*

Proof. This is immediate since $\mathcal{D}$ is closed under pullbacks; the Cartesian morphisms are precisely the pullback squares of differential bundles. ◀

▶ **Example 18.** For the tangent categories $(\mathsf{SMAN}, \mathbb{T})$, $(k\text{-}\mathsf{CALG}, \mathbb{L})$, and $(k\text{-}\mathsf{CALG}^{op}, \mathbb{T}^{op})$, the class of all differential bundles form a system of differential bundles. These recreate the previously known results that SVEC is a fibration over SMAN, that MOD is a fibration over $k\text{-}\mathsf{CALG}$, and that MOD^{op} is a fibration over $k\text{-}\mathsf{CALG}^{op}$ (or in other words, that MOD is a cofibration over $k\text{-}\mathsf{CALG}$).

▶ **Example 19.** For a Cartesian differential category $\mathbb{X}$, the differential bundles of the form $\pi_0 : A \times X \to A$ form a system of differential bundles for $(\mathbb{X}, \mathbb{T}_{\mathsf{D}})$. The resulting fibration for $\mathcal{P}$ corresponds to the canonical **linear fibration** [7, Def 16] of a Cartesian differential category, which plays a key role in characterizing Cartesian reverse differential categories. The linear fibration $\mathcal{L}[\mathbb{X}]$ has objects pairs (C, A) of objects of $\mathbb{X}$ and whose maps are pairs $(f, g) : (C, A) \to (D, B)$ where $f : C \to D$ and $g : C \times A \to B$ is **linear in its second argument** [7, Def 15], that is, $\langle \pi_0, 0, 0, \pi_1 \rangle \mathsf{D}[g] = g$. Relating this back to $\mathcal{P}$, it is straightforward to work out that by the two diagrams of (1), a linear differential bundle morphism of type $(\pi_0 : A \times X \to A) \to (\pi_0 : A' \times X' \to A')$ corresponds to a map $A \to A'$ and a map $A \times X \to X'$ which is linear in its second argument. Thus $\mathsf{DBun}_{\mathcal{P}}[(\mathbb{X}, \mathbb{T}_{\mathsf{D}})] \cong \mathcal{L}[\mathbb{X}]$ are equivalent as fibrations.

▶ **Example 20.** As an example of the above, in SMOOTH, the notion of linearity in the Cartesian differential category sense corresponds precisely to the classical notion of $\mathbb{R}$-linearity. Explicitly, a smooth function $F : \mathbb{R}^n \times \mathsf{R}^m \to \mathbb{R}^k$ is linear in its second argument in the above sense if and only if F is $\mathbb{R}$-linear in its second argument, that is,

$F(\vec{x}, r\vec{y} + s\vec{z}) = rF(\vec{x}, \vec{y}) + sF(\vec{x}, \vec{z})$. Now recall that every smooth vector bundle over $\mathbb{R}^n$ is a trivial vector bundle, and thus (up to isomorphism) of the form $\mathbb{R}^n \times \mathbb{R}^m$. Therefore, $\mathcal{P}$ is precisely the class of all differential bundles of $(\mathsf{SMOOTH}, \mathbb{T}_{\mathsf{D}})$, and so we have that $\mathsf{DBun}_{\mathcal{P}}\left[(\mathsf{SMOOTH}, \mathbb{T}_{\mathsf{D}})\right] = \mathsf{DBun}\left[(\mathsf{SMOOTH}, \mathbb{T}_{\mathsf{D}})\right] \cong \mathcal{L}\left[\mathbb{X}\right]$.

We now turn our attention to the notion of the dual fibration. For a fibration $\mathsf{F} : \mathbb{E} \to \mathbb{X}$, its **dual fibration** [14, Def 1.10.11] is the fibration $\mathsf{F}^{\circ} : \mathbb{E}^{\circ} \to \mathbb{X}$ where:

- **(i)** The objects of $\mathbb{E}^{\circ}$ are the same as the objects of $\mathbb{E}$;
- **(ii)** A map from A to B in $\mathbb{E}^{\circ}$ consists of an equivalence class of pairs $[(v, c)]$ where $v : C \to A$ is vertical and $c : C \to B$ is Cartesian, and where two pairs (v, c) and (v', c') are equivalent if there is a vertical isomorphism which make the relevant triangles commute;
- **(iii)** $\mathsf{F}^{\circ} : \mathbb{E}^{\circ} \to \mathbb{X}$ is defined on objects as F and on maps as $\mathsf{F}^{\circ}\left([(v, c)]\right) = \mathsf{F}(v)$.

The dual fibration of F has the key property that its fibre over A is the *opposite* category of the fibre of F over A. For more details about the dual of a fibration, see [14, Sec 1.10.11 – 13]. For a system of differential bundles, we can explicitly work out what its dual fibration will be. As this fibration consists of objects and maps as in the arrow category, this is essentially a modified version of the dual fibration of the arrow category; this dual fibration is known in various places as the category of containers [1] or dependent lenses [20], and has a close relationship to polynomial functors. In our case, the dual fibration of a system of differential bundles has the following form:

▶ **Proposition 21.** *If $\mathcal{D}$ is a system of differential bundles on a tangent category $(\mathbb{X}, \mathbb{T})$, then the dual fibration $\mathsf{U}^{\circ}_{\mathcal{D}} : \mathsf{DBun}^{\circ}_{\mathcal{D}}\left[(\mathbb{X}, \mathbb{T})\right] \to \mathbb{X}$ consists of:*

- **(i)** *Objects are differential bundles $\mathcal{E} \in \mathcal{D}$;*
- **(ii)** *A map $(f, g) : \mathcal{E} = (\mathsf{q} : E \to A, \sigma, \zeta, \lambda) \to \mathcal{E}' = (\mathsf{q}' : E' \to A', \sigma', \zeta', \lambda')$ in $\mathsf{DBun}^{\circ}_{\mathcal{D}}\left[(\mathbb{X}, \mathbb{T})\right]$ is a pair consisting of a map $f : A \to A'$ of $\mathbb{X}$ and a linear A'-differential bundle morphism $g : A \times_{A'} \mathcal{E}' \to \mathcal{E}$, where $A \times_{A'} \mathcal{E}'$ is the pullback bundle of $\mathcal{E}'$ along f; that is, a map $g : A \times_{A'} E' \to E$ in $\mathbb{X}$ such that the following diagrams commmute:*

$$\begin{array}{ccc} A \times_{A'} E' & \xrightarrow{\;g\;} & E \\ & {\scriptstyle \pi_0}\searrow & \downarrow{\scriptstyle \mathsf{q}} \\ & & A \end{array} \qquad \begin{array}{ccc} A \times_{A'} E' & \xrightarrow{\;g\;} & E \\ {\scriptstyle \mathsf{z} \times_{A'} \lambda}\downarrow & & \downarrow{\scriptstyle \lambda} \\ \mathsf{T}\left(A \times_{A'} E'\right) & \xrightarrow[\mathsf{T}(g)]{} & \mathsf{T}(E) \end{array} \tag{3}$$

- **(iii)** *The identity on $\mathcal{E}$ is the pair $(1_A : A \to A, \pi_1 : A \times_A E \to E)$;*
- **(iv)** *The composition of maps $(f : A \to A', g : A \times_{A'} E' \to E) : \mathcal{E} \to \mathcal{E}'$ and $(h : A' \to A'', k : A' \times_{A''} E'' \to E') : \mathcal{E}' \to \mathcal{E}''$ is the pair*

$$\left(fh,\ A \times_{A''} E'' \xrightarrow{\langle 1_A, (f \times_{A'} 1_{E''})k \rangle} A \times_{A'} E' \xrightarrow{\qquad g \qquad} E\right) : \mathcal{E} \to \mathcal{E}'' \tag{4}$$

- **(v)** $\mathsf{U}^{\circ}_{\mathcal{D}} : \mathsf{DBun}_{\mathcal{D}}\left[(\mathbb{X}, \mathbb{T})\right] \to \mathbb{X}$ *is defined on objects $\mathcal{E} = (\mathsf{q} : E \to A, \sigma, \zeta, \lambda)$ as $\mathsf{U}^{\circ}_{\mathcal{D}}(\mathcal{E}) = A$, and on maps as $\mathsf{U}^{\circ}_{\mathcal{D}}(f, g) = f$.*

Furthermore, note that when $A = A'$, for maps in $\mathsf{DBun}^{\circ}_{\mathcal{D}}\left[(\mathbb{X}, \mathbb{T})\right]$ of the form $(1_A, g) : \mathcal{E} \to \mathcal{E}'$, the domain of the second component is simply E', so we have that $g : E' \to E$. As such, the fibre over an object A is simply the opposite category $\mathsf{DBun}^{op}_{\mathcal{D}}\left[A\right]$.

It is worth emphasizing the form composition takes in the dual fibration: it is a mixture of a "forward" composite in the first component, and a "reverse" composite in the second component (with the k appearing before the g).

▶ **Example 22.** In [13], Higgins and Mackenzie call maps in the dual fibration of SVEC "comorphisms" of vector bundles [13, Def 1.1], and similarly call maps in the dual fibration of MOD^{op} "comorphisms" of modules [13, Def 2.1] (though the categories of "comorphisms" they define are the opposites of the ones we define in this paper).

The last ingredient in the definition of a reverse tangent category is an *involution* on the differential bundle fibration. Unlike for a Cartesian reverse differential category where the involution is a "dagger" and is asked to be the identity on objects [5, Def 33], the involution for a reverse tangent category need not be the identity on objects but is still required to be reflexive, which is captured by a natural transformation, and be compatible with the tangent bundle functor. Since a system of differential bundles $\mathcal{D}$ is closed under the tangent bundle, the induced functor $\overline{\mathsf{T}}$ from Prop 14 restricts to the class of $\mathcal{D}$, so we have a functor $\overline{\mathsf{T}}_{\mathcal{D}} : \mathsf{DBun}_{\mathcal{D}}[(\mathbb{X}, \mathbb{T})] \to \mathsf{DBun}_{\mathcal{D}}[(\mathbb{X}, \mathbb{T})]$, which is clearly also a fibration morphism. It follows from [7, Lem 32] that we also have a tangent bundle functor on the dual fibration, $\overline{\mathsf{T}}^{\circ}_{\mathcal{D}} : \mathsf{DBun}^{\circ}_{\mathcal{D}}[(\mathbb{X}, \mathbb{T})] \to \mathsf{DBun}^{\circ}_{\mathcal{D}}[(\mathbb{X}, \mathbb{T})]$.

▶ **Definition 23.** *For a tangent category $(\mathbb{X}, \mathbb{T})$ with a system of differential bundles $\mathcal{D}$, a* ***linear involution*** *is a pair $(*, \iota)$ consisting of a fibration morphism $(-)^* : \mathsf{DBun}_{\mathcal{D}}[(\mathbb{X}, \mathbb{T})] \to \mathsf{DBun}^{\circ}_{\mathcal{D}}[(\mathbb{X}, \mathbb{T})]$ where:*

- **(i)** *The image of a differential bundle $\mathcal{E} = (\mathsf{q} : E \to A, \sigma, \zeta, \lambda)$ is the differential bundle denoted as $\mathcal{E}^* = (\mathsf{q}^* : E^* \to A, \sigma^*, \zeta^*, \lambda^*)$, and $\mathcal{E}'$ is called the* ***dual bundle*** *of $\mathcal{E}$;*
- **(ii)** *The image of a linear differential bundle morphism $(f, g) : \mathcal{E} = (\mathsf{q} : E \to A, \sigma, \zeta, \lambda) \to \mathcal{E}' = (\mathsf{q}' : E' \to A', \sigma', \zeta', \lambda')$ is denoted as $(f, g)^* = (f, g^*)$ where $g^* : A \times_{A'} E'^* \to E^*$.*

and a natural isomorphism $\iota_{\mathcal{E}} : \mathcal{E} \to \mathcal{E}^{**}$ such that the following diagram should commute:

$$\begin{array}{ccc} \mathsf{DBun}_{\mathcal{D}}[(\mathbb{X}, \mathbb{T})] & \xrightarrow{(-)^*} & \mathsf{DBun}^{\circ}_{\mathcal{D}}[(\mathbb{X}, \mathbb{T})] \\ \Big\downarrow{\scriptstyle \overline{\mathsf{T}}} & & \Big\downarrow{\scriptstyle \overline{\mathsf{T}}^{\circ}} \\ \mathsf{DBun}_{\mathcal{D}}[(\mathbb{X}, \mathbb{T})] & \xrightarrow[(-)^*]{} & \mathsf{DBun}_{\mathcal{D}}[(\mathbb{X}, \mathbb{T})] \end{array} \tag{5}$$

▶ **Definition 24.** *A* ***reverse tangent category*** *is a quintuple $(\mathbb{X}, \mathbb{T}, \mathcal{D}, *, \iota)$ consisting of a tangent category $(\mathbb{X}, \mathbb{T})$ with a system of differential bundles $\mathcal{D}$ and a linear involution $(*, \iota)$.*

On the fibers, the involution induces an involutive functor $(-)^* : \mathsf{DBun}^{op}_{\mathcal{D}}[A] \to \mathsf{DBun}_{\mathcal{D}}[A]$. So for an A-linear differential bundle morphism $g : \mathcal{E} \to \mathcal{E}'$, applying the involution will result in an A-linear bundle morphism of type $g^* : \mathcal{E}'^* \to \mathcal{E}^*$, so in particular its underlying map is $g^* : E'^* \to E^*$.

Essentially, a reverse tangent category is a tangent category where for every differential bundle in the specified system there is a chosen "dual" differential bundle. In particular, we can consider the dual of the tangent bundle of an object, which we call the *reverse* tangent bundle of that object.

▶ **Definition 25.** *In a reverse tangent category $(\mathbb{X}, \mathbb{T}, \mathcal{D}, *, \iota)$, define the* ***reverse tangent functor*** *$\mathcal{T}^* : \mathbb{X} \to \mathsf{DBun}_{\mathcal{D}}[(\mathbb{X}, \mathbb{T})]^{\circ}$ as the following composite:*

$$\mathcal{T}^* := \mathbb{X} \xrightarrow{\mathcal{T}} \mathsf{DBun}_{\mathcal{D}}[(\mathbb{X}, \mathbb{T})] \xrightarrow{(-)^*} \mathsf{DBun}_{\mathcal{D}}[(\mathbb{X}, \mathbb{T})]^{\circ} \tag{6}$$

where on objects we denote:

$$\mathcal{T}^*(A) := (\mathsf{p}^*_A : \mathsf{T}^*(A) \to A, \mathsf{s}^*_A : \mathsf{T}^*_2(A) \to \mathsf{T}^*(A), \mathsf{z}^*_A : A \to \mathsf{T}^*(A), \ell^*_A : \mathsf{T}^*(A) \to \mathsf{TT}^*(A))$$

and on maps we denote

$$\mathcal{T}^*(f) = (f : A \to B, \mathsf{T}^*(f) : A \times_B \mathsf{T}^*(B) \to \mathsf{T}^*(A))$$

The object $\mathsf{T}^*(A)$ *is called the* ***reverse tangent bundle*** *of* A*.*

It is worth pointing out what the reverse tangent bundle functor does on maps. Given a map $f : A \to B$, we have that the second component of its image is a map of type $\mathsf{T}^*(f) : A \times_B \mathsf{T}^*(B) \to \mathsf{T}^*(A)$. This is exactly what one wants for gradient-based learning: given a point x of the domain and a cotangent vector over $f(x)$ in the *codomain*, $\mathsf{T}^*(f)$ produces a cotangent vector in the *domain*.

▶ Remark 26. It is also important to note that in general, the reverse tangent bundle does not induce a functor on the base category, since T^* is not functorial (either in the covariant or contravariant sense) with respect to the composition in $\mathbb{X}$. Moreover, the fact that T^* does not induce a functor on the base category is part of the reason why we have found it difficult to provide a direct description of a reverse tangent category (as was done for reverse cartesian differential categories in [5]). A tangent category has as part of its data natural transformations related to iterates of the tangent bundle functor. Appropriate analogs of these for reverse tangent categories have thus been difficult to find given that there is no endofunctor to iterate in the reverse situation. This is why we have instead chosen to define a reverse tangent as a tangent category with an appropriate involution operation.

We conclude this section with our main examples of reverse tangent categories.

▶ **Example 27.** Smooth manifolds form a reverse tangent category. Let us first describe the dual fibration of smooth vector bundles. Observe that if $\mathsf{q}' : F \to N$ is a smooth vector bundle and $f : M \to N$ is a smooth function, in local coordinates, elements of the pullback $M \times_N F$ are pairs (x, v) where $x \in M$ and $v \in F_{f(x)}$. As such, the dual fibration SVEC°, also called the category of star bundles [17, Sec 41.1], has objects smooth vector bundles $\mathsf{q} : E \to M$, and a map $\mathsf{q} : E \to M$ to $\mathsf{q}' : F \to N$ consists of smooth functions $f : M \to N$ and $g : M \times_N F \to E$ such that for every $x \in M$, $\mathsf{q}(g(x, v)) = x$, and the induced map $g_x : F_{f(x)} \to E_x$ is $\mathbb{R}$-linear, so we may write $g(x, v) = (x, g_x(v))$. There is an involution which sends a smooth vector bundle $\mathsf{q} : E \to M$ to the classical dual bundle $\mathsf{q}^* : E^* \to M$ from differential geometry, where the fibres of E^* are the $\mathbb{R}$-linear duals of the fibre of E, that is, E^*_x is the dual vector space of E_x in the classical linear algebra sense:

$$E^*_x = \{\phi : E_x \to \mathbb{R} |\ \phi \text{ } \mathbb{R}\text{-linear}\}$$

The involution sends a smooth vector bundle morphism $(f : M \to N, g : E \to F)$ to the pair $(f, g^* : M \times_N F^* \to E^*)$, where in local coordinates:

$$g^*(x, \phi) = (x, \phi(g_x(-))$$

So $(\mathsf{SMAN}, \mathbb{T}, \mathcal{D}, *, \iota)$ is a reverse tangent category. The reverse tangent bundle is given by the classical *cotangent bundle* $\mathsf{T}^*(M)$ from differential geometry, where:

$$\mathsf{T}^*(M) := \bigsqcup_{x \in M} \mathsf{T}^*_x(M)$$

So in local coordinates, elements of the cotangent bundle $\mathsf{T}^*(M)$ are pairs (m, ϕ) where $m \in M$ and $\phi \in \mathsf{T}^*_m(M)$, so $\phi : \mathsf{T}_m(M) \to \mathbb{R}$ is an $\mathbb{R}$-linear morphism. For a smooth function $f : M \to N$, we have that its image via the reverse tangent bundle is of type

$\mathsf{T}^*(f) : M \times_N \mathsf{T}^*(N) \to \mathsf{T}^*(M)$. In local coordinates, elements of $M \times_N \mathsf{T}^*(N)$ are pairs $(m, \phi : \mathsf{T}_{f(m)}(N) \to \mathbb{R})$, and so $\mathsf{T}^*(f)$ is defined is given as follows:

$$\mathsf{T}^*(f)(m, \phi) = (m, \phi\,(T_m(f)(-)))$$

▶ **Example 28.** Every Cartesian reverse differential category is a reverse tangent category. Briefly, a Cartesian reverse differential category [5, Def 13] can be defined as a category with finite products which in particular comes equipped with a **reverse differential combinator** R which associates every map $f : A \to B$ to a map of type $\mathsf{R}[f] : A \times B \to A$, called the reverse derivative of f. Alternatively, a Cartesian reverse differential category can be equivalently characterized as a Cartesian differential category equipped with a linear dagger [5, Thm 42]. Briefly, for a Cartesian differential category $\mathbb{X}$, a **linear dagger** [5, Def 39] is a fibration morphism $(-)^\dagger : \mathcal{L}[\mathbb{X}]^\circ \to \mathcal{L}[\mathbb{X}]$ which is the identity on objects and involutive, such that each fibre of the linear fibration has dagger biproducts – the dual of the linear fibration is described in detail in [7, Ex 34]. In particular, the linear dagger allows one to transpose linear arguments, that is, for every map $g : C \times A \to B$ which is linear in its second argument, the linear dagger gives a map $g^\dagger : C \times B \to A$ which is linear in its second argument. Now by using the linear dagger $\dagger$ for the linear involution and setting $\iota = 1$ (since the dagger is the identity on objects), we thus have that $(\mathbb{X}, \mathbb{T}_\mathsf{D}, \mathcal{P}, \dagger, 1)$ is a reverse tangent category. In particular, the reverse tangent bundle of an object A is $\mathsf{T}^*(A) = A \times A$, while for a map $f : A \to B$, its image via the reverse tangent bundle is of type $\mathsf{T}^*(f) : A \times B \to A \times A$ and can be nicely expressed using the reverse differential combinator as follows:

$$\mathsf{T}^*(f) = \langle \pi_0, \mathsf{R}[f] \rangle$$

▶ **Example 29.** SMOOTH is a Cartesian reverse differential category where for a smooth function $F : \mathbb{R}^n \to \mathbb{R}^m$, which recall is an m-tuple $F = \langle f_1, \ldots, f_m \rangle$ of smooth functions $f_i : \mathbb{R}^n \to \mathbb{R}$, its reverse derivative $\mathsf{R}[F] : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is defined as the n-tuple:

$$\mathsf{R}[F](\vec{x}, \vec{z}) := \left\langle \sum_{i=1}^{m} \frac{\partial f_i}{\partial x_1}(\vec{x}) z_i, \ldots, \sum_{i=1}^{m} \frac{\partial f_i}{\partial x_n}(\vec{x}) z_i \right\rangle$$

Thus $(\mathsf{SMOOTH}, \mathbb{T}_\mathsf{D}, \mathcal{P}, \dagger, 1)$ is a reverse tangent category, where in particular:

$$\mathsf{T}^*(\mathbb{R}^n) = \mathbb{R}^n \times \mathbb{R}^n \qquad\qquad \mathsf{T}^*(F)(\vec{x}, \vec{z}) = (\vec{x}, \mathsf{R}[F](\vec{x}, \vec{z}))$$

▶ **Example 30.** All of k-CALG will not form a reverse tangent category; instead, we must restrict to those algebras which are finitely generated free k-modules. So, let k-CALG_f and MOD_f be the full subcategories whose objects are finitely generated free k-modules. Clearly we still have that $(k\text{-}\mathsf{CALG}_f, \mathbb{L})$ is a tangent category and $\mathsf{DBun}\,[(k\text{-}\mathsf{CALG}_f, \mathbb{L})] \simeq \mathsf{MOD}_f$ is still a fibration of differential bundles, and thus all differential bundles form a system of differential bundles. The dual fibration MOD_f° has objects pairs (A, M) consisting of a commutative k-algebra A and an A-module M, and where a map is now a pair $(f, g) : (A, M) \to (B, N)$ consisting of a k-algebra morphism $f : A \to B$ and a k-linear map $g : N \to M$ such that $g(f(a)n) = ag(n)$. Since an A-module M is also a k-module, it make sense to consider the k-linear dual of M, which we denote as:

$$M^\circledast = \{\phi : M \to k |\ \phi\ k\text{-linear}\}$$

Note that $M^\circledast$ is also an A-module via the action $(a, \phi) \mapsto \phi(a \cdot -)$. Since M is a finitely generated free k-module, it is reflexive so we have the canonical isomorphism $M \cong M^{\circledast\circledast}$.

As such, this induces an involution which sends objects (A, M) to $(A, M^{\circledast})$, and maps $(f : A \to B, g : M \to N)$ to $(f : A \to B, g^{\circledast} : N^{\circledast} \to M^{\circledast})$ where:

$$g^{\circledast}(\phi) = \phi(g(-))$$

So $(k\text{-}\mathsf{CALG}, \mathbb{L}, \mathcal{D}, \circledast, \iota)$ is a reverse tangent category. For a commutative k-algebra A, its reverse tangent bundle is:

$$\perp^{\circledast}(A) = A^{\circledast}[\varepsilon] = \{a + \phi\varepsilon |\ a \in A, \phi : A \to k |\ \phi\ k\text{-linear and } \varepsilon^2 = 0\}$$

Now consider a k-algebra morphism $f : A \to B$, and note that:

$$A \times_B \perp^{\circledast}(B) = \{a + \phi\varepsilon |\ a \in A, \phi : V \to k \text{ where } \phi\ k\text{-linear and } \varepsilon^2 = 0\}$$

So $\mathsf{T}^{\circledast}(f) : A \times_B \perp^{\circledast}(B) \to \perp^{\circledast}(A)$ is defined as:

$$\perp^{\circledast}(f)(a + \phi\varepsilon) = a + \phi\left(f(-)\right)\epsilon$$

This example generalizes nicely to the star autonomous setting. Indeed, the category of commutative monoids of any star autonomous category with distributive finite biproducts is a reverse tangent category – thus providing a bountiful source of examples of reverse tangent categories.

▶ **Example 31.** Similarly, all of $k\text{-}\mathsf{CALG}^{op}$ will not form a reverse tangent category; instead, we must restrict to the subcategory of *smooth* algebras and the display system given by the finitely generated projective modules. This requires some setup. So briefly, a smooth k-algebra is a commutative k-algebra A whose associated affine scheme is smooth [21, Def 10.137.1]. Let $k\text{-}\mathsf{SmALG}$ be the full subcategory of $k\text{-}\mathsf{CALG}$ whose objects are the smooth k-algebras. Then to explain why $k\text{-}\mathsf{SmALG}$ is a tangent category, we need to explain why tangent algebras of smooth algebras are again smooth. Firstly, for a smooth k-algebra A and a finitely generated projective module M, $\mathsf{Sym}_A(M)$ is a smooth A-algebra [12, Prop 17.3.8]. Moreover, [21, Lemma 10.137.14] implies that smoothness is preserved via restriction of scalars, and therefore $\mathsf{Sym}_A(M)$ is also a smooth k-algebra. Now for a smooth k-algebra A, $\Omega(A)$ is a finitely generated projective A-module [21, Sec 10.142.(2)]. Therefore, we have that $\mathsf{T}(A)$ is a smooth k-algebra (and a smooth A-algebra). Furthermore, smoothness is also preserved via change of basis [21, Lemma 10.137.4], which implies that $\mathsf{T}_n(A)$ is also a smooth k-algebra (and a smooth A-algebra). So we conclude that $(k\text{-}\mathsf{SmALG}^{op}, \mathbb{T})$ is indeed a tangent category, and the differential bundles will again correspond to modules. For our system of differential bundles $\mathcal{D}$, we take the class of differential bundles that correspond to finitely generated projective modules. That $\mathcal{D}$ is indeed a system of differential bundles follows from the fact that the module of Kähler differentials of a smooth algebra is finitely generated projective, so $\mathcal{D}$ is closed under tangent bundles, and since the extension of scalars preserves finitely generated projective modules [11, Excercise 8.4], this implies that $\mathcal{D}$ is also closed under pullback (since the pullback in $k\text{-}\mathsf{CALG}^{op}$ in this case is given by $B \otimes_A \mathsf{Sym}_A(M) \cong \mathsf{Sym}_B(B \otimes_A M)$). We can give an alternative description of $\mathsf{DBun}\left[(k\text{-}\mathsf{SmALG}, \mathbb{T})\right]_{\mathcal{D}}$ similar to that of Ex 13. Let FMOD be the full subcategory of MOD whose objects are the pairs (A, M) consisting of a smooth k-algebra and a finitely generated projective A-module M[6]. Then we have that

[6] It is important to note the difference between Ex 30 and Ex 31. In the former, we consider modules that are finitely generated over the base ring, while in the latter we take modules that are finitely generated over the algebra.

$\mathsf{DBun}\left[(k\text{-}\mathsf{SmALG}, \mathbb{T})\right]_{\mathcal{D}} \simeq \mathsf{FMOD}^{op}$. Now the dual fibration $(\mathsf{FMOD}^{op})^{\circ}$ is easier understood via its opposite category $(\mathsf{FMOD}^{op})^{\circ op}$. This category has the same objects as FMOD, but where a map $(f, g) : (A, M) \to (B, N)$ consists a k-algebra morphism $f : A \to B$ and an A-linear map $g : N \to B \otimes_A M$ in the sense that $g(f(a)n) = ag(n)$. For an A-module M, consider its A-linear dual, which we denote as:

$$M^* = \{\phi : M \to A |\ \phi \ A\text{-linear}\}$$

Since M is a finitely generated free A-module, it is reflexive as an A-module so we have the canonical isomorphism $M \cong M^{**}$. As such, this induces an involution which sends objects (A, M) to (A, M^*). To describe what it does on a map, recall that a finitely generated projective A-module M has a finite generating set $\{e_1, ..., e_n\}$ and also a finite generating set $\{e_1^*, ..., e_n^*\}$ for M^*. Then the involution takes a map $(f : A \to B, g : M \to N)$ to $(f : A \to B, g^* : N^* \to B \otimes_A M^*)$ where g^* is defined as follows:

$$g^*(\phi) = \sum_{i=1}^{n} \phi(g(e_i)) \otimes_A e_i^*$$

So $(k\text{-}\mathsf{SmALG}^{op}, \mathbb{T}, \mathcal{D}, *, \iota)$ is a reverse tangent category. For a commutative k-algebra A, it is well known that the A-linear dual of $\Omega(A)$ is $\mathsf{DER}(A)$ the module of derivations on A – which recall is a k-linear map $\mathsf{D} : A \to A$ which satisfies the Leibniz rule:

$$\mathsf{D}(ab) = a\mathsf{D}(b) + b\mathsf{D}(a)$$

Therefore, for a smooth k-algebra A, we may take its reverse tangent bundle to be given as:

$$\mathsf{T}^*(A) = \mathsf{Sym}_A\left(\mathsf{DER}(A)\right)$$

To describe what the reverse tangent bundle does on an algebra morphism, we must use the fact since A is a smooth k-algebra morphism it is, by definition, a finitely presented k-algebra so $A \cong k[x_1, \ldots, x_n]/I$ where $k[x_1, \ldots, x_n]$ is the polynomial ring and I is a finitely generated ideal $I = (p_1(\vec{x}), \ldots, p_m(\vec{x}))$. Abusing notation slightly, let $\partial_i : A \to A$ be the derivation on A associated with differentiating polynomials with respect to the x_i variable (where we again abuse notation slightly and take $x_i \in A$). Then for a k-algebra morphism $f : A \to B$, applying T^* to it gives a k-algebra morphism of type $\mathsf{T}^*(f) : \mathsf{T}^*(B) \to B \otimes_A \mathsf{T}^*(A)$ defined as follows on generators $b \in B$ and $\mathsf{D} \in \mathsf{DER}(B)$:

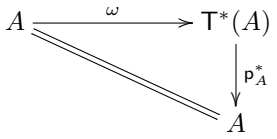
$$\mathsf{T}^*(f)(b) = b \otimes_A 1 \qquad \mathsf{T}^*(f)\left(\mathsf{D}\right) = \sum_{i=1}^{n} \mathsf{D}\left(f(x_i)\right) \otimes_A \partial_i$$

4 Some Theory of Reverse Tangent Categories

In this section, we provide some basic results that apply in any reverse tangent category. In particular, these results generalize key concepts about the cotangent bundle from classical differential geometry.

We begin with the notion of a covector field. In differential geometry, covector fields correspond to differential 1-forms. A key property of covector fields in differential geometry is that they can be pulled back; that is, a covector field on the codomain of a map can be pulled back to a covector field on the domain. The same is true in a reverse tangent category.

▶ **Definition 32.** *In a reverse tangent category* $(\mathbb{X}, \mathbb{T}, \mathcal{D}, *, \iota)$*, a* ***covector field*** *of an object* A *is a section of* $\mathsf{p}^*_A : \mathsf{T}^*(A) \to A$*, that is, a map* $\omega : A \to \mathsf{T}^*(A)$ *such that the following diagram commutes:*



(7)

▶ **Proposition 33** (Pullback of covector fields)**.** *In a reverse tangent category* $(\mathbb{X}, \mathbb{T}, \mathcal{D}, *, \iota)$*, if* $\omega : B \to \mathsf{T}^*(B)$ *is a covector field, then for any map* $f : A \to B$*, the composite*

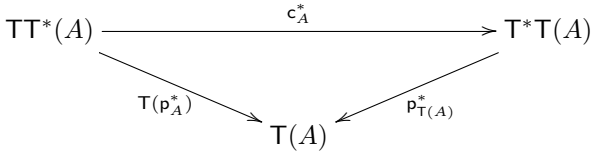
$$A \xrightarrow{\langle 1_A, f\omega \rangle} A \times_B \mathsf{T}^*(B) \xrightarrow{\mathsf{T}^*(f)} \mathsf{T}^*(A) \tag{8}$$

is a covector field on A*.*

Proof. Since $(f, \mathsf{T}^*(f))$ is a map in $\mathsf{DBun}^\circ_{\mathcal{D}}[(\mathbb{X}, \mathbb{T})]$, by the left diagram of (3) we easily compute that $\langle 1_A, f\omega \rangle \mathsf{T}^*(f)\mathsf{p}^*_A = \langle 1_A, f\omega \rangle \pi_0 = 1_A$, as required. ◀

Our next observation is that there is a canonical isomorphism between the composites of the tangent bundle and the reverse tangent bundle, generalizing a result for smooth manifolds [17, Sec 26.11].

▶ **Proposition 34.** *In a reverse tangent category* $(\mathbb{X}, \mathbb{T}, \mathcal{D}, *, \iota)$*, for every object* A*, there is a natural linear* $\mathsf{T}(A)$*-differential bundle isomorphism* $\mathsf{c}^*_A : \overline{\mathsf{T}}(\mathcal{T}^*(A)) \to \mathcal{T}^*(\mathsf{T}(A))$*, so in particular the following diagram commutes:*



(9)

Proof. In any tangent category, the canonical flip is an A-linear differential bundle isomorphism $\mathsf{c}_A : \mathcal{T}(\mathsf{T}(A)) \to \overline{\mathsf{T}}(\mathcal{T}(A))$. So c_A is an isomorphism in $\mathsf{DBun}_{\mathcal{D}}[A]$. Now by (5), the dual bundle of $\mathcal{T}(\mathsf{T}(A))$ is the tangent bundle of the reverse tangent bundle; that is, $\overline{\mathsf{T}}(\mathcal{T}^*(A))$ with projection $\mathsf{T}(\mathsf{p}^*_A)$, while the dual bundle of $\mathcal{T}(\mathsf{T}(A))$ is the reverse tangent bundle of the tangent bundle $\mathcal{T}^*(\mathsf{T}(A))$ with projection $\mathsf{p}^*_{\mathsf{T}(A)}$. As such, applying the induced involution $(-)^* : \mathsf{DBun}^{op}_{\mathcal{D}}[A] \to \mathsf{DBun}_{\mathcal{D}}[A]$ to c_A, we obtain an A-linear differential bundle isomorphism $\mathsf{c}^*_A : \overline{\mathsf{T}}(\mathcal{T}^*(A)) \to \mathcal{T}^*(\mathsf{T}(A))$. So in particular we have an isomorphism $\mathsf{c}^*_A : \mathsf{TT}^*(A) \to \mathsf{T}^*\mathsf{T}(A)$ such that $\mathsf{c}^*_A \mathsf{p}^*_{\mathsf{T}(A)} = \mathsf{T}(\mathsf{p}^*_A)$, as required. ◀

As mentioned in Remark 26, the reverse tangent bundle does not induce a functor on the base category. However, in differential geometry, while the cotangent bundle is similarly not functorial on all smooth functions, it is functorial on *étale* maps [17, pg. 346], which are useful generalizations of isomorphisms. We will now show that the same is true for the reverse tangent bundle in any reverse tangent category. A smooth function $f : M \to N$ between smooth manifolds is étale if for any $x \in M$, the tangent space at x is isomorphic to the tangent space at $f(x)$, that is $\mathsf{T}_x(M) \cong \mathsf{T}_{f(x)}(N)$. However, being étale can also be characterized in terms of a pullback square, specifically that the naturality square of the tangent bundle projection is a pullback. As such, the notion of an étale map can be easily defined in an arbitrary tangent category.

▶ **Definition 35.** *In a tangent category* $(\mathbb{X}, \mathbb{T})$*, a map* $f : A \to B$ *is* ***étale*** *if its associated* p *naturality square:*

$$\begin{array}{ccc} \mathsf{T}(A) & \xrightarrow{\mathsf{T}(f)} & \mathsf{T}(B) \\ {\scriptstyle \mathsf{p}_A}\downarrow & & \downarrow{\scriptstyle p_B} \\ A & \xrightarrow[f]{} & B \end{array} \tag{10}$$

is a pullback. Let $(\mathbb{X}, \mathbb{T})_{\acute{e}tale}$ *be the subcategory of étale maps of* $(\mathbb{X}, \mathbb{T})$*.*

It is straightforward to see that identity maps (and isomorphisms) are étale, and that the composition of étale maps is again étale. So $(\mathbb{X}, \mathbb{T})_{\text{étale}}$ is indeed well-defined. A slightly less obvious result is the following:

▶ **Lemma 36.** *In a tangent category* $(\mathbb{X}, \mathbb{T})$*, if a map* $f : A \to B$ *is étale and the square*

$$\begin{array}{ccc} X & \xrightarrow{\pi_1} & A \\ {\scriptstyle \pi_0}\downarrow & & \downarrow{\scriptstyle f} \\ C & \xrightarrow[g]{} & B \end{array}$$

is a pullback diagram which is preserved by T*, then* $\pi_0 : X \to C$ *is also étale.*

Proof. Consider the diagram

$$\begin{array}{ccccc} \mathsf{T}(X) & \xrightarrow{\mathsf{p}_X} & X & \xrightarrow{\pi_1} & A \\ {\scriptstyle \mathsf{T}(\pi_0)}\downarrow & & \downarrow{\scriptstyle \pi_0} & & \downarrow{\scriptstyle f} \\ \mathsf{T}(C) & \xrightarrow[\mathsf{p}_C]{} & C & \xrightarrow[g]{} & B \end{array}$$

We need to show that the left square is a pullback. However, the right square is a pullback by assumption, so by the pullback pasting lemma, it suffices to show that the outer rectangle is a pullback. However, by naturality of p, the outer rectangle can be rewritten as

$$\begin{array}{ccccc} \mathsf{T}(X) & \xrightarrow{\mathsf{T}(\pi_1)} & \mathsf{T}(A) & \xrightarrow{\mathsf{p}_A} & A \\ {\scriptstyle \mathsf{T}(\pi_0)}\downarrow & & \downarrow{\scriptstyle \mathsf{T}(f)} & & \downarrow{\scriptstyle f} \\ \mathsf{T}(C) & \xrightarrow[\mathsf{T}(g)]{} & \mathsf{T}(B) & \xrightarrow[\mathsf{p}_B]{} & B \end{array}$$

This is a pullback since the left square is a pullback by assumption and the right square is a pullback since f is étale. Thus the left square in the first diagram is a pullback, as required. ◀

In order to show that the reverse tangent bundle gives a functor on the subcategory of étale maps, we will also need the following useful observation:

▶ **Lemma 37.** *Let* $\mathcal{D}$ *be a system of differential bundles on a tangent category* $(\mathbb{X}, \mathbb{T})$*, and suppose that* $(f : A \to A', g : A \times_{A'} E' \to E) : \mathcal{E} \to \mathcal{E}'$ *is a Cartesian map in* $\mathsf{DBun}^{\circ}_{\mathcal{D}}[(\mathbb{X}, \mathbb{T})]$*. Then* g *is an isomorphism, so* $A \times_{A'} E' \cong E$*, and* $(f : A \to A', g^{-1}\pi_1 : E \to E') : \mathcal{E} \to \mathcal{E}'$ *is a Cartesian map in* $\mathsf{DBun}_{\mathcal{D}}[(\mathbb{X}, \mathbb{T})]$*.*

Proof. Cartesian maps in the dual fibration correspond to Cartesian maps in the starting fibration [16, Prop. 3.2]. In general, an equivalence class $[(v,c)]$ is Cartesian in the dual fibration if and only if the vertical component v is an isomorphism. As such, we have that $[(v,c)] = [(1, v^{-1}c)]$, where in particular, $v^{-1}c$ is a Cartesian map in the starting fibration. Thus, the desired result is obtained by translating this in terms of $\mathsf{DBun}_{\mathcal{D}}[(\mathbb{X},\mathbb{T})]$. So if (f,g) is Cartesian in $\mathsf{DBun}^{\circ}_{\mathcal{D}}[(\mathbb{X},\mathbb{T})]$, then its associated representation as an equivalence class is $[(1_A,g),(f,\pi_1)]$. This implies that $(1_A,g)$ is an isomorphism in $\mathsf{DBun}_{\mathcal{D}}[(\mathbb{X},\mathbb{T})]$, which in turn implies that g is an isomorphism. Then $[(1_A,g),(f,\pi_1)] = [(1_A,1_E),(f,g^{-1}\pi_1)]$, and in particular $(f,g^{-1}\pi_1)$ is a Cartesian map in $\mathsf{DBun}_{\mathcal{D}}[(\mathbb{X},\mathbb{T})]$. ◀

We can now prove our main result about étale maps:

▶ **Proposition 38** (Functoriality of T^* on étale maps)**.** *In a reverse tangent category $(\mathbb{X},\mathbb{T},\mathcal{D},*,\iota)$, if $f : A \to B$ is étale, then $\mathsf{T}^*(f) : A \times_B \mathsf{T}^*(B) \to \mathsf{T}^*(A)$ is an isomorphism. Furthermore, there is an endofunctor $\widehat{\mathsf{T}^*} : (\mathbb{X},\mathbb{T})_{étale} \to (\mathbb{X},\mathbb{T})_{étale}$ which maps an object A to its reverse tangent bundle $\mathsf{T}^*(A)$, and an étale map $f : A \to B$ to the map $\widehat{\mathsf{T}^*}(f) : \mathsf{T}^*(A) \to \mathsf{T}^*(B)$ which is defined as the composite $\widehat{\mathsf{T}^*}(f) := \mathsf{T}^*(f)^{-1}\pi_1$.*

Proof. Since f is étale, $(f,\mathsf{T}(f))$ is a pullback square, and hence Cartesian in $\mathsf{DBun}_{\mathcal{D}}[(\mathbb{X},\mathbb{T})]$. Since the involution $(-)^*$ is a fibration morphism, it sends Cartesian maps to Cartesian maps. As such, $(f,\mathsf{T}(f))^* = (f,\mathsf{T}^*(f))$ is Cartesian in $\mathsf{DBun}^{\circ}_{\mathcal{D}}[(\mathbb{X},\mathbb{T})]$. By Lemma 37, it follows that $\mathsf{T}^*(f)$ is an isomorphism. Thus $\mathsf{T}^*(f)^{-1}$ is also an isomorphism and hence étale, and by Lemma 36, $\pi_1 : A \times_B \mathsf{T}^*(B) \to \mathsf{T}^*(B)$ is also étale since it is a pullback of the étale map f. Hence $\widehat{\mathsf{T}^*}(f) := \mathsf{T}^*(f)^{-1}\pi_1$ is itself étale, as it is a composite of étale maps. It is then easy to check this assignment is functorial. ◀

In Ex 28, we explained how every Cartesian reverse differential category is a reverse tangent category. We conclude this section by going in the opposite direction. Looking at the forward side of the story, to do so we must work with Cartesian tangent categories [3, Def 2.8], which are tangent categories with finite products that are preserved by the tangent bundle functor T. Then to extract a Cartesian differential category from a Cartesian tangent category, one looks at the *differential objects* [4, Sec 3], which are the differential bundles over the terminal object 1. In particular, a differential object A has the property that $\mathsf{T}(A) \cong A \times A$. Then the full subcategory DO of differential objects and *all* maps between them is a Cartesian differential category [3, Thm 4.11], where the differential combinator is defined on a map $f : A \to B$ as follows:

$$\mathsf{D}[f] := \; A \times A \cong \mathsf{T}(A) \xrightarrow{\mathsf{T}(f)} \mathsf{T}(B) \cong B \times B \xrightarrow{\pi_1} B \tag{11}$$

As differential objects give a Cartesian differential category, to obtain a Cartesian reverse differential category, we need only give a linear dagger, which will be built using the linear involution. First note that for a Cartesian tangent category, for every object A, $\mathsf{DBun}[A]$ has finite biproducts. Now one of the axioms of a linear dagger is that the fibres of the linear fibration have dagger biproducts. Thus, we must ask that the involution preserves this biproduct structure:

▶ **Definition 39.** *A **Cartesian reverse tangent category** is a reverse tangent category $(\mathbb{X},\mathbb{T},\mathcal{D},*,\iota)$ such that $(\mathbb{X},\mathbb{T})$ is a Cartesian tangent category, $\mathcal{D}$ is closed under products, and for each object A, the induced involution $(-)^* : \mathsf{DBun}^{op}_{\mathcal{D}}[A] \to \mathsf{DBun}_{\mathcal{D}}[A]$ preserves the biproduct structure.*

In a Cartesian reverse differential category, since the dagger is the identity on objects, we have that the differential bundles in our chosen system are in fact self-dual. So to build a Cartesian reverse differential category from a Cartesian reverse tangent category, we consider the differential objects which are also self-dual.

▶ **Definition 40.** *In a Cartesian reverse tangent category* $(\mathbb{X}, \mathbb{T}, \mathcal{D}, *, \iota)$*, a* ***self-dual differential object*** *is a differential object* A *in* $\mathcal{D}$ *which comes equipped with a linear isomorphism* $A \cong A^*$*. Let* DO_{sd} *be the full subcategory of self-dual differential objects and all maps between them.*

▶ **Lemma 41.** *In a Cartesian reverse tangent category* $(\mathbb{X}, \mathbb{T}, \mathcal{D}, *, \iota)$*, if* A *is a* $*$*-self-dual differential object, then* $\mathsf{T}^*(A) \cong A \times A \cong \mathsf{T}(A)$*.*

Proof. Since A is a differential object and thus a differential bundle, by applying (5), we get that $\mathsf{T}^*(A) \cong \mathsf{T}(A^*) \cong \mathsf{T}(A) \cong A \times A$. ◀

We can now explain how the full subcategory of self-dual differential objects is a Cartesian reverse differential category.

▶ **Proposition 42.** *For a Cartesian reverse tangent category* $(\mathbb{X}, \mathbb{T}, \mathcal{D}, *, \iota)$*,* DO_{sd} *is a Cartesian reverse differential category where the reverse differential combinator* R *is defined on a map* $f : A \to B$ *as the following composite:*

$$\mathsf{R}[f] := \; A \times B \cong A \times_B \mathsf{T}^*(B) \xrightarrow{\mathsf{T}^*(f)} \mathsf{T}^*(A) \cong A \times A \xrightarrow{\pi_1} A \tag{12}$$

Proof. Since $\mathsf{DO}_{\mathrm{sd}}$ is a full subcategory of DO, it is clear that $\mathsf{DO}_{\mathrm{sd}}$ is also a Cartesian differential category with the same structure as DO. So it remains to construct a linear dagger for $\mathsf{DO}_{\mathrm{sd}}$. To do so, given a map $f : C \times A \to B$ which is linear in A, we must give a map $f^\dagger : C \times B \to A$ which is linear in B. Now $C \times A$ and $C \times B$ are differential bundles over C and $\langle \pi_0, f \rangle : C \times A \to C \times B$ is a map in $\mathsf{DBun}_{\mathcal{D}}[C]$. Applying the involution $(-)^* : \mathsf{DBun}_{\mathcal{D}}^{op}[A] \to \mathsf{DBun}_{\mathcal{D}}[A]$, we obtain a map $(\langle \pi_0, f \rangle)^* : C \times B^* \to C \times A^*$. Then define $f^\dagger$ as the following composite:

$$f^\dagger := \; C \times B \cong C \times B^* \xrightarrow{(\langle \pi_0, f \rangle)^*} C \times A^* \cong C \times A \xrightarrow{\pi_1} A \tag{13}$$

This map is linear in B since $(\langle \pi_0, f \rangle)^*$ is a C-linear differential bundle morphism. It is straightforward to check that this induces a linear dagger $\dagger$ on $\mathsf{DO}_{\mathrm{sd}}$. So we conclude that $\mathsf{DO}_{\mathrm{sd}}$ is a Cartesian reverse differential combinator, and the reverse differential combinator defined in (12) is precisely the dagger of the differential combinator defined in (11). ◀

▶ **Example 43.** $(\mathsf{SMAN}, \mathbb{T}, \mathcal{D}, *, \iota)$ is a Cartesian reverse tangent category, and its $*$-self-dual differential objects are precisely the Euclidean spaces $\mathbb{R}^n$. As such, the resulting Cartesian reverse differential category is precisely SMOOTH.

5 Future Work

One of the next major steps for reverse tangent categories is to apply these ideas to categorically study gradient-based learning and automatic differentiation on smooth manifolds. However, there are several other ways this work could be expanded upon:

(i) We have chosen to *define* a reverse tangent category as a tangent category with a certain kind of involution. However, this was not how Cartesian reverse differential categories were defined. Cartesian reverse differential categories were defined directly in terms of a reverse differential combinator R, and then shown to be equivalent to a Cartesian differential category with a certain kind of involution. It would be useful to have a direct description of a reverse tangent category in a similar fashion, that is, a structure involving a "reverse tangent bundle" functor from the base category to an appropriately defined category of differential bundles.

(ii) There is much more theoretical work that can be explored in an arbitrary reverse tangent category, especially by taking inspiration from results about the cotangent bundle in differential geometry. For example, a pseudo-Riemannian structure on a manifold can be defined as an isomorphism between its tangent bundle and its cotangent bundle; thus, one could similarly explore what can be done with such objects in an arbitrary reverse tangent category.

(iii) There are many ways in which tangent categories can be generated from existing ones. For example, the category of vector fields of a tangent category is again a tangent category [6, Prop 2.10]. It would be interesting to see how many of these constructions apply to reverse tangent categories, thus giving many more examples of this structure.

References

1 M. Abbott, T. Altenkirch, and N. Ghani. Categories of Containers. *Foundations of Software Science and Computation Structures*, pages 28–28, 2003. doi:10.1007/3-540-36576-1_2.

2 R. F. Blute, J. R. B. Cockett, and R. A. G. Seely. Cartesian Differential Categories. *Theory and Applications of Categories*, 22(23):622–672, 2009.

3 J. R. B. Cockett and G. S. H. Cruttwell. Differential Structure, Tangent Structure, and SDG. *Applied Categorical Structures*, 22(2):331–417, 2014. doi:10.1007/s10485-013-9312-0.

4 J. R. B. Cockett and G. S. H. Cruttwell. Differential Bundles and Fibrations for Tangent Categories. *Cahiers de Topologie et Géométrie Différentielle Catégoriques*, LIX:10–92, 2018.

5 J. R. B. Cockett, G. S. H. Cruttwell, J. D. Gallagher, J.-S. P. Lemay, B. MacAdam, G. Plotkin, and D. Pronk. Reverse derivative categories. In *CSL 2020*, volume 152, pages 18:1–18:16, 2020. doi:10.4230/LIPIcs.CSL.2020.18.

6 J. R. B. Cockett, G. S. H. Cruttwell, and J.-S. P. Lemay. Differential Equations in a Tangent Category I: Complete Vector Fields, Flows, and Exponentials. *Applied Categorical Structures*, 29:773–825, 2021. doi:10.1007/s10485-021-09629-x.

7 G. Cruttwell, J. Gallagher, J.-S. P. Lemay, and D. Pronk. Monoidal Reverse Differential Categories. *Mathematical Structures in Computer Science*, 32(10):1313–1363, 2022.

8 G. Cruttwell, J. Gallagher, and D. Pronk. Categorical Semantics of a Simple Differential Programming Language. *Electronic Proceedings in Theoretical Computer Science*, 333:289–310, 2021. doi:10.4204/EPTCS.333.20.

9 G. Cruttwell and J.-S. P. Lemay. Differential Bundles in Commutative Algebra and Algebraic Geometry. *arXiv preprint*, 2023. arXiv:2301.05542.

10 G.S.H. Cruttwell, B. Gavranovic, N. Ghani, P. Wilson, and F. Zanasi. Categorical Foundations of Gradient-Based Learning. *ESOP 2022*, pages 1–28, 2022. doi:10.1007/978-3-030-99336-8_1.

11 B. Farb and R. K. Dennis. *Noncommutative Algebra*, volume 144. Springer Science & Business Media, 2012. doi:10.1007/978-1-4612-0889-1.

12 A. Grothendieck. Éléments de géométrie algébrique IV. *Publications Mathématiques de l'IHÉS*, 28:5–255, 1966. doi:10.1007/BF02684778.

13 Philip J. Higgins and Kirril C. H. Mackenzie. Duality for Base-Changing Morphisms of Vector Bundles, Modules, Lie Algebroids and Poisson Structures. *Math. Proc. Camb. Phil. Soc.*, 114:471–488, 1993. doi:10.1017/S0305004100071760.

14 B. Jacobs. *Categorical Logic and Type Theory*. Elsevier, 1999.

15 B. Jubin. The Tangent Functor Monad and Foliations. *arXiv preprint*, 2014. arXiv:1401.0940.

16 A. Kock. The Dual Fibration in Elementary Terms. *arxiv*, 2015. arXiv:1501.01947.

17 I. Kolar, J. Slovak, and P. Michor. *Natural Operations in Differential Geometry*. Springer, 1993. doi:10.1007/978-3-662-02950-3.

18 B. MacAdam. Vector Bundles and Differential Bundles in the Category of Smooth Manifolds. *Applied categorical structures*, 29(2):285–310, 2021. doi:10.1007/s10485-020-09617-7.

19 J. Rosickỳ. Abstract Tangent Functors. *Diagrammes*, 12:JR1–JR11, 1984.

20 D. Spivak. Generalized Lens Categories via Functors $C^{op} \to$ Cat. *arxiv:1908.02202*, 2019.

21 The Stacks Project Authors. *Stacks Project*. https://stacks.math.columbia.edu, 2018.

22 P. Wilson and F. Zanasi. Reverse Derivative Ascent: A Categorical Approach to Learning Boolean Circuits. In Proceedings of the 3rd Annual International *Applied Category Theory Conference 2020,* Cambridge, USA, 6-10th July 2020, volume 333, pages 247–260. Open Publishing Association, 2021. doi:10.4204/EPTCS.333.17.

Intuitionistic Gödel-Löb Logic, à la Simpson: Labelled Systems and Birelational Semantics

Anupam Das
University of Birmingham, UK

Iris van der Giessen
University of Birmingham, UK

Sonia Marin
University of Birmingham, UK

Abstract

We derive an intuitionistic version of Gödel-Löb modal logic (GL) in the style of Simpson, via proof theoretic techniques. We recover a labelled system, ℓIGL, by restricting a non-wellfounded labelled system for GL to have only one formula on the right. The latter is obtained using techniques from cyclic proof theory, sidestepping the barrier that GL's usual frame condition (converse well-foundedness) is not first-order definable. While existing intuitionistic versions of GL are typically defined over only the box (and not the diamond), our presentation includes both modalities.

Our main result is that ℓIGL coincides with a corresponding semantic condition in birelational semantics: the composition of the modal relation and the intuitionistic relation is conversely well-founded. We call the resulting logic IGL. While the soundness direction is proved using standard ideas, the completeness direction is more complex and necessitates a detour through several intermediate characterisations of IGL.

2012 ACM Subject Classification Theory of computation → Proof theory

Keywords and phrases provability logic, proof theory, intuitionistic modal logic, cyclic proofs, non-wellfounded proofs, proof search, cut-elimination, labelled sequents

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.22

Related Version *Full Version*: `https://arxiv.org/abs/2309.00532`

Funding This work was partially supported by a UKRI Future Leaders Fellowship, "Structure vs Invariants in Proofs", project reference MR/S035540/1.

Acknowledgements We would like to thank Marianna Girlando and Jan Rooduijn for several valuable discussions on the topic. We particularly thank Marianna for her input during our reading group on intuitionistic modal logic which led to the preliminary ideas on which part of this paper is built.

1 Introduction

Gödel-Löb logic (GL) originates in the *provability* reading of modal logic: $\Box$ is interpreted as "it is provable that", in an arithmetical theory with suitable coding capacity, inducing its corresponding *provability logic*. Löb formulated a set of necessary conditions on the provability logic of Peano Arithmetic (PA), giving rise to GL, extending basic modal logic K by what we now call *Löb's axiom*: $\Box(\Box A \to A) \to \Box A$. Somewhat astoundingly GL turns out to be *complete* for PA's provability logic, a celebrated result of Solovay [36]: all that PA can prove about its own provability is already a consequence of a relatively simple (and indeed decidable) propositional modal logic.

Proof theoretically Löb's axiom represents a form of *induction*. Indeed, at the level of modal logic semantics, GL enjoys a correspondence with transitive relational structures that are *terminating*[1] [32]. Duly, in computer science, Löb's axiom has inspired several variants of modal type theories that extend simply-typed lambda calculus with some form of *recursion*. These range, for instance, from the seminal work of Nakano [26], to more recent explorations into guarded recursion [4] and intensional recursion [21]. Based in type theories, these developments naturally cast GL in a *constructive* setting, but little attention was given to analysing the induced intuitionistic modal logics. Indeed, for this reason, [8] has proposed a more foundational basis for studying computational interpretations of GL, by way of a sequent calculus for the logic of the topos of trees.

Returning to the provability reading of modal logic, several constructive variants of GL have been independently proposed (see [23, 15] for overviews). An important such logic is iGL (and its variants) which now enjoys a rich mathematical theory, from semantics (e.g., [23]) to proof theory (e.g., [16, 18]). Interestingly, while iGL is known to be sound for the provability logic of Heyting Arithmetic (HA), the intuitionistic counterpart of PA, it is not complete. The provability logic of HA has been recently announced by [25], currently under review.

One shortfall of all the above mentioned approaches is that they do not allow us to recover a bona fide computational interpretation of *classical* GL along, say, the Gödel-Gentzen negative translation (GG), a standard way of lifting interpretations of intuitionistic logics to their classical counterparts. Indeed it was recently observed that the "i" (or " constructive") traditions of intuitionistic modal logic are too weak to validate the GG translation [10, 11].

On the other hand modal logic's relational semantics effectively renders it a fragment of usual first-order predicate logic (FOL), the so-called *standard translation*. Interpreting this semantics in an intuitionistic meta-theory defines a logic that *does* validate GG, for the same reason that intuitionistic FOL GG-interprets classical FOL. This is the approach taken (and considerably developed) by Simpson [34], building on earlier work of Fischer Servi [14] and Plotkin and Stirling [29]. The resulting logic IK (and friends) enjoys a remarkably robust proof theory by way of *labelled systems*, which may be duly seen as a fragment of Gentzen's systems for intuitionistic FOL by way of the standard translation.

Contribution

In this work we develop an intuitionistic version of GL, following Simpson's methodology [34]. In particular, while logics such as iGL are defined over only the $\Box$, our logic is naturally formulated over both the $\Box$ and the $\Diamond$. A key stumbling block to this end, as noticed already in the classical setting by Negri in [27], is that GL's correspondence to terminating relations cannot seemingly be inlined within a standard labelled system: termination is not even FOL-definable. To side-step this barrier we draw from a now well-developed proof-theoretic approach to (co)induction: *non-wellfounded proofs* (e.g. [28, 7, 35, 2, 12, 9]). Such proofs allow infinite branches, and so are (typically) equipped with a *progress condition* that ensures sound reasoning. Starting from a standard labelled system for transitive relations, K4, we recover a labelled calculus ℓGL for GL by allowing non-wellfounded derivations under a typical progress condition. In fact, our progress condition is precisely the one from Simpson's *Cyclic Arithmetic* [35, 9]. Following (the same) Simpson, we duly recover an intuitionistic version ℓIGL of GL *syntactically* in a standard way: we restrict ℓGL to one formula on the right. Note that this presentation, albeit syntactic, does not constitute an *axiomatisation*, as the system ℓIGL is infinitary.

[1] Other authors refer to this property as *Noetherian* or *conversely well-founded*, but we opt for this simpler nomenclature.

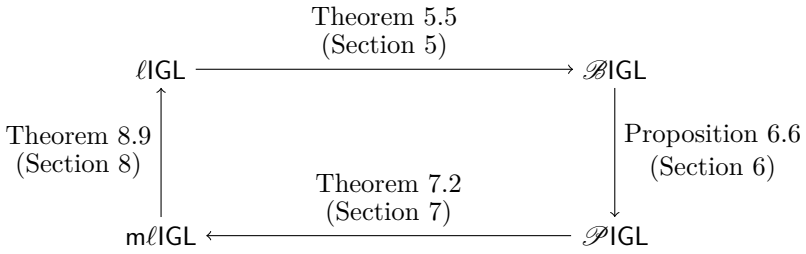


Figure 1 Summary of our main results, where arrows → denote inclusions of modal logics.

At the same time we can recover an intuitionistic version of GL *semantically* by suitably adapting the *birelational semantics* of intuitionistic modal logics, which combine the partial order $\leq$ of intuitionistic semantics with the accessibility relation R of modal semantics. Our semantic formulation $\mathscr{B}$IGL is duly obtained by reading the termination criterion of classical GL's semantics intuitionistically: the composition $\leq; R$ must be terminating. Our main result is that these two characterisations, ℓIGL and $\mathscr{B}$IGL, are indeed equivalent, and we duly dub the resulting logic IGL. Note that we do not address any *provability* reading of IGL in this work, being beyond scope.

The soundness direction is proved via standard techniques from intuitionistic modal logic and non-wellfounded proof theory. The completeness direction, on the other hand, is more cumbersome. To this end we exercise an intricate combination of proof theoretic techniques, necessitating two further (and ultimately equivalent) characterisations of IGL: semantically $\mathscr{P}$IGL, essentially a class of intuitionistic FOL structures, and syntactically mℓIGL, a *multi-succedent* variant of ℓIGL. These formulations facilitate a countermodel construction from failed proof search, inspired by Takeuti's for LJ [37]. Due to the non-wellfoundedness of proofs we employ a determinacy principle to organise the construction, a standard technique in non-wellfounded and cyclic proof theory (e.g. [28]). For the reduction to ℓIGL we devise a form of *continuous cut-elimination*, building on more recent ideas in non-wellfounded proof theory, cf. [2, 12]. Our " grand tour" of results is visualised in Figure 1, also indicating the organisation of this paper. Due to space constraints full proofs are omitted, but may be found in an associated preprint [13].

Other related work

The proof theory of GL is (in)famously complex. The first sequent calculus of GL was considered in [22] but its cut-elimination property was only finally settled (positively) in [17] after several attempts [38, 31, 24]. Intuitionistic versions of sequent calculi for GL are developed in [16, 18] and provide calculi for iGL. Labelled calculi [27] and *nested calculi* [30] have also been proposed. However all of these calculi are arguably unsatisfactory: the modality introduction rules are non-standard (which is atypical for labelled calculi) and, in particular, modal rules may change the polarity of formula occurrences, due to the inductive nature of Löb's axiom.

It is possible to recover a system for GL by admitting non-wellfounded proofs in a sequent calculus for K4, as observed by Shamkanov [33]. An intuitionistic version of this system has been studied by Iemhoff [20]. In these works both the base calculus and the corresponding correctness criterion are bespoke, rather than " recovered" from established foundations.

Non-wellfounded proofs originate in the study of modal logics with fixed points, in particular Niwinski and Walukiewicz's seminal work on the μ-calculus [28]. These ideas were later inlined into the proof theory of FOL with certain inductive definitions by Brotherston

and Simpson (e.g. [7]), a source of inspiration for the present work. As already mentioned, recent extensions of these ideas to PA [35, 9] and advances on cut-elimination [2, 12] are quite relevant to our development.

2 Preliminaries on (classical) modal logic

Throughout this work we work with a set Pr of **propositional symbols**, written p, q, etc., which we simultaneously construe as unary predicate symbols when working in predicate logic. For the latter we also assume a single binary relation symbol R and a countable set Var of **(individual) variables**, written x, y, etc.

We recall some preliminaries on classical modal logic in this section, but point the reader to general references such as [5, 6] for a more comprehensive background.

2.1 Language and semantics

The formulas of modal logic are generated by the following grammar:

$$A \quad ::= \quad p \in \mathsf{Pr} \mid \bot \mid A \wedge A \mid A \vee A \mid A \to A \mid \Box A \mid \Diamond A$$

As usual we write $\neg A := A \to \bot$, and employ standard bracketing conventions.

Modal formulas are interpreted over **relational frames** $\mathcal{F} = (W, R^{\mathcal{F}})$ formed of a non empty set of **worlds** W equipped with an **accessibility relation** $R^{\mathcal{F}} \subseteq W \times W$.[2] A **relational model** $\mathcal{M} = (W, R^{\mathcal{M}}, V)$ is a structure where $(W, R^{\mathcal{M}})$ is a frame with a **valuation** $V : W \to \mathcal{P}(\mathsf{Pr})$. Let $\mathcal{M} = (W, R^{\mathcal{M}}, V)$ be a relational model. For worlds $w \in W$ and formulas A we define the satisfaction judgement $\mathcal{M}, w \vDash A$ as follows:

- $\mathcal{M}, w \vDash p$ if $p \in V(w)$;
- $\mathcal{M}, w \nvDash \bot$;
- $\mathcal{M}, w \vDash A \wedge B$ if $\mathcal{M}, w \vDash A$ and $\mathcal{M}, w \vDash B$;
- $\mathcal{M}, w \vDash A \vee B$ if $\mathcal{M}, w \vDash A$ or $\mathcal{M}, w \vDash B$;
- $\mathcal{M}, w \vDash A \to B$ if $\mathcal{M}, w \vDash A$ implies $\mathcal{M}, w \vDash B$;
- $\mathcal{M}, w \vDash \Box A$ if for all v such that $wR^{\mathcal{M}}v$ we have $\mathcal{M}, v \vDash A$;
- $\mathcal{M}, w \vDash \Diamond A$ if there exists v such that $wR^{\mathcal{M}}v$ and $\mathcal{M}, v \vDash A$.

If $\mathcal{M}, w \vDash A$ for all $w \in W$ we simply write $\mathcal{M} \vDash A$, and if $(W, R^{\mathcal{M}}, V) \vDash A$ for all valuations V based on frame $\mathcal{F} = (W, R^{\mathcal{M}})$, we simply write $\mathcal{F} \vDash A$.

2.2 Axiomatisations and Gödel-Löb logic

Turning to syntax, let us now build up the modal logics we are concerned with axiomatically, before relating them to the semantics just discussed. The modal logic K is axiomatised by all the theorems of classical propositional logic (CPL) together with axiom (k) and closed under the rules (mp) (*modus ponens*) and nec (*necessitation*) from Figure 2. The logic $\mathsf{K4}$ is defined in the same way by further including the axiom ($\mathsf{4}$) from Figure 2.

Referring back to the earlier semantics, the following characterisation is well-known [5]:

▶ **Theorem 2.1** (K/$\mathsf{K4}$ characterisation)**.** *$\mathsf{K} \vdash A$ (resp. $\mathsf{K4} \vdash A$) iff A is satisfied in all relational frames (resp. with transitive accessibility relation).*

[2] We parameterise the relations by the frame or models to be able distinguish it from the fixed binary relation symbol R.

$$(\mathsf{k}) : \Box(A \to B) \to (\Box A \to \Box B) \qquad (\mathsf{mp})\ \frac{A \to B \quad A}{B} \qquad (\mathsf{nec})\ \frac{A}{\Box A}$$

$$(4) : \Box A \to \Box\Box A \qquad (\mathsf{löb}) : \Box(\Box A \to A) \to \Box A$$

Figure 2 Some modal axioms and rules.

The main subject of this work is an extension of K by the Löb axiom (löb), in Figure 2, a sort of induction principle:

▶ **Definition 2.2** (Gödel-Löb logic)**.** GL *is defined by extending* K4 *by the* ***Löb axiom*** (löb) *and is closed under* (mp) *and* (nec)*.*[3]

Semantically (löb) says that, as long as worlds satisfy A whenever all its successors (wrt. the accessibility relation) satisfy A, then A holds universally. This amounts to a " reverse' induction principle for the accessibility relation. Indeed we have an associated characterisation for GL just like that for K4.

We call a frame **terminating** if its accessibility relation has no infinite path.

▶ **Theorem 2.3** (GL characterisation)**.** $\mathsf{GL} \vdash A$ *iff all transitive terminating frames satisfy* A*.*

2.3 Labelled calculi and the standard translation

The relational semantics of modal logic may be viewed as a bona fide fragment of predicate logic. Recalling the predicate language we fixed at the start of the section, the **standard translation** is defined as follows: for individual variables x and modal formulas A we define the predicate formula $x : A$ by:

- $x : p$ is $p(x)$;
- $x : \bot$ is $\bot$;
- $x : A \star B$ is $(x : A) \star (x : B)$ for $\star \in \{\vee, \wedge, \to\}$;
- $x : \Diamond A$ is $\exists y(xRy \wedge (y : A))$;
- $x : \Box A$ is $\forall y(xRy \to (y : A))$.

This induces a well-behaved proof theory as a fragment of usual first-order predicate systems [27], adaptable to many extensions under correspondence theorems such as Theorem 2.1.

A **(labelled) sequent** is an expression $\mathbf{R}, \Gamma \Rightarrow \Delta$, where $\mathbf{R}$ is a set of **relational atoms**, i.e. formulas of form xRy, and Γ and Δ are multisets of **labelled formulas**, i.e. formulas of form $x : A$. We sometimes refer to the variable x in $x : A$ as a **label**. We write $\mathsf{Var}(S)$ for the subset of labels/variables that occur in a given sequent S. Similarly so for $\mathsf{Var}(\mathbf{R})$, etc.

Notationally, we have identified labelled formulas with the standard translation at the beginning of this section. This is entirely suggestive, as we can now easily distil systems for modal logics of interest by appealing to the sequent calculus LK for first-order predicate logic. In this vein, the labelled calculus ℓK for modal logic K is given in Figure 3. From here, under the aforementioned correspondence between K4 and transitive frames, we have a system ℓK4 for K4 that extends ℓK by the **transitivity rule**:

$$\mathsf{tr}\ \frac{\mathbf{R}, xRy, yRz, xRz, \Gamma \Rightarrow \Delta}{\mathbf{R}, xRy, yRz, \Gamma \Rightarrow \Delta}$$

[3] Alternatively, GL can be axiomatised by adding the (löb) axiom to K, as (4) can be derived from (löb).

Identity and cut:

$$\mathsf{id}\ \frac{}{\mathbf{R}, x : p \Rightarrow x : p} \qquad \mathsf{cut}\ \frac{\mathbf{R}, \Gamma \Rightarrow \Delta, x : A \quad \mathbf{R}, \Gamma', x : A \Rightarrow \Delta'}{\mathbf{R}, \Gamma, \Gamma' \Rightarrow \Delta, \Delta'}$$

Structural rules: *Relational structural rule:*

$$\mathsf{w}\text{-}l\ \frac{\mathbf{R}, \Gamma \Rightarrow \Delta}{\mathbf{R}, \Gamma, x : A \Rightarrow \Delta} \qquad \mathsf{c}\text{-}l\ \frac{\mathbf{R}, \Gamma, x : A, x : A \Rightarrow \Delta}{\mathbf{R}, \Gamma, x : A \Rightarrow \Delta} \qquad \mathsf{th}\ \frac{\mathbf{R}, \Gamma \Rightarrow \Delta}{\mathbf{R}, \mathbf{R}', \Gamma \Rightarrow \Delta}$$

$$\mathsf{w}\text{-}r\ \frac{\mathbf{R}, \Gamma \Rightarrow \Delta}{\mathbf{R}, \Gamma \Rightarrow \Delta, x : A} \qquad \mathsf{c}\text{-}r\ \frac{\mathbf{R}, \Gamma \Rightarrow \Delta, x : A, x : A}{\mathbf{R}, \Gamma \Rightarrow \Delta, x : A}$$

Propositional logical rules:

$$\bot\text{-}l\ \frac{}{\mathbf{R}, x : \bot, \Gamma \Rightarrow \Delta} \qquad \text{(no right rule for } \bot\text{)}$$

$$\to\text{-}l\ \frac{\mathbf{R}, \Gamma \Rightarrow \Delta, x : A \quad \mathbf{R}, \Gamma', x : B \Rightarrow \Delta'}{\mathbf{R}, \Gamma, \Gamma', x : A \to B \Rightarrow \Delta, \Delta'} \qquad \to\text{-}r\ \frac{\mathbf{R}, \Gamma, x : A \Rightarrow \Delta, x : B}{\mathbf{R}, \Gamma \Rightarrow \Delta, x : A \to B}$$

$$\wedge\text{-}l\ \frac{\mathbf{R}, \Gamma, x : A_i \Rightarrow \Delta}{\mathbf{R}, \Gamma, x : A_0 \wedge A_1 \Rightarrow \Delta}\ i \in \{0, 1\} \qquad \vee\text{-}l\ \frac{\mathbf{R}, \Gamma, x : A \Rightarrow \Delta \quad \mathbf{R}, \Gamma, x : B \Rightarrow \Delta}{\mathbf{R}, \Gamma, x : A \vee B \Rightarrow \Delta}$$

$$\vee\text{-}r\ \frac{\mathbf{R}, \Gamma \Rightarrow \Delta, x : A_i}{\mathbf{R}, \Gamma \Rightarrow \Delta, x : A_0 \vee A_1}\ i \in \{0, 1\} \qquad \wedge\text{-}r\ \frac{\mathbf{R}, \Gamma \Rightarrow \Delta, x : A \quad \mathbf{R}, \Gamma \Rightarrow \Delta, x : B}{\mathbf{R}, \Gamma \Rightarrow \Delta, x : A \wedge B}$$

Modal logical rules:

$$\Diamond\text{-}l\ \frac{\mathbf{R}, xRy, \Gamma, y : A \Rightarrow \Delta}{\mathbf{R}, \Gamma, x : \Diamond A \Rightarrow \Delta}\ y \text{ fresh} \qquad \Diamond\text{-}r\ \frac{\mathbf{R}, xRy, \Gamma \Rightarrow \Delta, y : A}{\mathbf{R}, xRy, \Gamma \Rightarrow \Delta, x : \Diamond A}$$

$$\Box\text{-}r\ \frac{\mathbf{R}, xRy, \Gamma \Rightarrow \Delta, y : A}{\mathbf{R}, \Gamma \Rightarrow \Delta, x : \Box A}\ y \text{ fresh} \qquad \Box\text{-}l\ \frac{\mathbf{R}, xRy, \Gamma, y : A \Rightarrow \Delta}{\mathbf{R}, xRy, \Gamma, x : \Box A \Rightarrow \Delta}$$

Figure 3 The standard labelled calculus $\ell\mathsf{K}$ for modal logic K.

For a labelled system S we write $\mathsf{S} \vdash \mathbf{R}, \Gamma \Rightarrow \Delta$, if there is a proof of $\mathbf{R}, \Gamma \Rightarrow \Delta$ using the rules from S. We write $\mathsf{S} \vdash x : A$, or even $\mathsf{S} \vdash A$, to mean $\mathsf{S} \vdash \varnothing \Rightarrow x : A$. Almost immediately from Theorem 2.1 and metatheorems for predicate logic, we have:

▶ **Proposition 2.4** (Soundness and completeness). *$\ell\mathsf{K} \vdash x : A$ (resp. $\ell\mathsf{K4} \vdash x : A$) iff $\mathsf{K} \vdash A$ (resp. $\mathsf{K4} \vdash A$).*

Naturally systems for many other modal logics can be readily obtained, when they correspond to simple frame properties, by adding further relational (structural) rules [27]. Indeed, let us call a labelled calculus *standard* if it does not extend $\ell\mathsf{K}$ by any new logical or (non-relational) structural rules, nor any new logical axioms. This terminology is suggestive, since it forces the left and right introduction rules for modalities to coincide with those induced by the *standard* translation. As remarked by Negri in [27], typical standard calculi cannot be complete for GL, as termination of a relation is not even first-order definable. We shall sidestep this barrier in the next section by making use of *non-wellfounded* systems.

3 Recovering a proof theoretic account for GL

In another branch of the proof theory literature, structural treatments of induction and well-foundedness have been developed in the guise of *non-wellfounded* and *cyclic* proofs, e.g. [28, 7, 2, 3, 35, 9]. Here non-wellfoundedness in proofs allows for inductive reasoning, and soundness is ensured by some global correctness condition. By incorporating these ideas into modal proof theory, one can design a non-wellfounded proof system for GL [33]. In this section we recover a *standard* labelled calculus for GL, in the sense of the preceding discussion. Our presentation is based on the correctness condition for *Cyclic Arithmetic* in [35, 9].

3.1 A standard calculus for GL, via non-wellfounded proofs

In what follows, we consider systems S that will typically be some fragment of the system ℓK4 given earlier. The definitions we give apply to all " non-wellfounded" systems of this work.

▶ **Definition 3.1** (Preproofs). *A* ***preproof*** *in a system* S *is a possibly infinite derivation generated from the rules of* S*.*

As preproofs may, in particular, be non-wellfounded, they may conclude fallacious theorems, so we require a correctness criterion. Appealing to the correspondence of GL over transitive terminating relations, we may directly import a "trace" condition first used in [35].

▶ **Definition 3.2** (Traces and proofs). *Fix a preproof P and an infinite branch $(S_i)_{i<\omega}$. A* ***trace*** *along $(S_i)_{i<\omega}$ is a sequence of variables $(x_i)_{i<\omega}$ such that either:*

- $x_{i+1} = x_i$*; or,*
- $x_i R x_{i+1}$ *appears in S_i (then i is a* ***progress point*** *of $(x_i)_{i<\omega}$).*

A trace is ***progressing*** *if it is not ultimately constant, i.e. if 3.2 above applies infinitely often. A branch $(S_i)_{i<\omega}$ is progressing if it has a progressing trace, and a preproof P is progressing, or simply is a* ***∞-proof****, if each infinite branch has a progressing trace.*
We write $\mathsf{S} \vdash^{\infty} \mathbf{R}, \Gamma \Rightarrow \Delta$ *if there is a ∞-proof in* S *of the sequent* $\mathbf{R}, \Gamma \Rightarrow \Delta$*.*

▶ **Example 3.3** (Löb and contra-Löb). An example of a ∞-proof of (löb) in ℓK4 is given in Figure 4, left. Here we have used bullets • to identify identical subproofs, up to the indicated renamings of variables. The preproof is indeed progressing: it has only one infinite branch, whose progress points are coloured red and trigger at each iteration of the •-loop. Notice that each sequent has only one formula on the right-hand side (RHS).

Another example of a ∞-proof in ℓK4 is the *contraposition* of (löb), given in Figure 4, right. We have merged several steps, and omitted some routine structural steps and initial sequents, a convention we shall continue to employ throughout this work. Again the preproof is indeed progressing, by the same argument as before. Notice, this time, that there are two formulas in the RHS of the premiss of •. Indeed, by basic inspection of proof search, there is no cut-free ∞-proof avoiding this feature.

▶ Remark 3.4 (On regularity). In non-wellfounded proof theory, special attention is often paid to the subset of *regular* preproofs, which may be written as finite (possibly cyclic) graphs, as in Example 3.3 above. Nonetheless these will play no role in the present work, as we are purely concerned with logical and proof theoretic investigations, not with effectivity.

The progress condition is invariant under expansion of relational contexts in a preproof: We may omit consideration of the thinning rule th (Figure 3) when reasoning about ∞-proofs:

▶ **Observation 3.5.** th *is eliminable in ∞-proofs of* S*.*

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{xRz, x:\Box(\Box p\to p)\Rightarrow z:p}\,\bullet[z/y]}{xRy, yRz, x:\Box(\Box p\to p)\Rightarrow z:p}\,\mathsf{tr}}{xRy, x:\Box(\Box p\to p)\Rightarrow y:\Box p}\,\Box\text{-}r}{xRy, x:\Box(\Box p\to p), y:\Box p\to p\Rightarrow y:p}\,\to\text{-}l}{xRy, x:\Box(\Box p\to p), x:\Box(\Box p\to p)\Rightarrow y:p}\,\Box\text{-}l}{xRy, x:\Box(\Box p\to p)\Rightarrow y:p}\,\mathsf{c}\text{-}l}{x:\Box(\Box p\to p)\Rightarrow x:\Box p}\,\Box\text{-}r}{\Rightarrow x:\Box(\Box p\to p)\to\Box p}\,\to\text{-}r \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{xRz, z:p\Rightarrow x:\Diamond(p\wedge\Box\neg p)}\,\bullet[z/y]}{xRy, yRz, y:p, z:p\Rightarrow x:\Diamond(p\wedge\Box\neg p)}\,\mathsf{tr}}{xRy, y:p\Rightarrow x:\Diamond(p\wedge\Box\neg p), y:\Box\neg p}\,\neg\text{-}r, \Box\text{-}r}{xRy, y:p\Rightarrow x:\Diamond(p\wedge\Box\neg p), y:p\wedge\Box\neg p}\,\wedge\text{-}r}{xRy, y:p\Rightarrow x:\Diamond(p\wedge\Box\neg p)}\,\Diamond\text{-}l}{x:\Diamond p\Rightarrow\Diamond(p\wedge\Box\neg p)}\,\Diamond\text{-}r}{\Rightarrow x:\Diamond p\to\Diamond(p\wedge\Box\neg p)}\,\to\text{-}r$$

Figure 4 An ∞-proof in $\ell\mathsf{K4}$ of Löb's axiom, left, and its contraposition, right. Identity steps on $y:p$ above the $\to$-l step, left, and the $\wedge$-r step, right, are omitted for space considerations.

3.2 Soundness and completeness

Let us now argue that our notion of ∞-proof for $\ell\mathsf{K4}$ is sound and complete for GL. The most interesting part is soundness, comprising a contradiction argument by infinite descent as is common in non-wellfounded proof theory, relying on the characterisation result of Theorem 2.3:

▶ **Proposition 3.6** (Soundness). *If* $\ell\mathsf{K4}\vdash^{\infty} x:A$ *then* $\mathsf{GL}\vdash A$.

On the other hand, thanks to Example 3.3 and the known completeness of $\ell\mathsf{K4}$ for $\mathsf{K4}$, we can use cut-rules to derive:

▶ **Proposition 3.7** (Completeness). *If* $\mathsf{GL}\vdash A$ *then* $\ell\mathsf{K4}\vdash^{\infty} x:A$.

These results motivate the following notation:

▶ **Definition 3.8.** $\ell\mathsf{GL}$ *is the class of* ∞*-proofs of* $\ell\mathsf{K4}$.

Henceforth for a class of ∞-proofs $\mathbf{P}$ we may write simply $\mathbf{P}\vdash S$ if $\mathbf{P}$ contains a ∞-proof of the sequent S. For instance, writing $\ell\mathsf{GL}\vdash S$ is the same as $\ell\mathsf{K4}\vdash^{\infty} S$.

4 Recovering intuitionistic versions of GL from syntax and semantics

In this section we propose two intuitionistic versions of GL, in the style of Simpson, respectively by consideration of the proof theory and semantics of GL discussed in the previous sections. Later sections are then devoted to proving the equivalence of these two notions.

4.1 An intuitionistic GL, via syntax

Following Gentzen, it is natural to define intuitionistic calculi based on their classical counterparts by restricting sequents to one formula on the RHS. However, when implementing this restriction to different starting calculi for modal logic K (based on sequents, nested sequents, labelled sequents) one ends up with different intuitionistic variants of K (see e.g. [10]). The restriction of the ordinary sequent calculus for K defines a logic which is not compatible with the standard translation in the way we described for K. On the other hand labelled calculi are well designed for this purpose.

▶ **Definition 4.1.** *The system* $\ell\mathsf{IK}$ *(resp.* $\ell\mathsf{IK4}$*) is the restriction of* $\ell\mathsf{K}$ *(resp.* $\ell\mathsf{K4}$*) to sequents in which exactly one formula occurs on the RHS.*

Note that the rules w-r and c-r cannot be used in ℓIK4, by the singleton restriction on the RHS of a sequent. In the style of Simpson [34], we can from here duly recover a standard intuitionistic analogue of GL, by restricting ℓGL to ∞-proofs with only singleton RHSs.

▶ **Definition 4.2.** *We write ℓIGL for the class of ∞-proofs of ℓIK4.*

▶ **Example 4.3** (Löb, revisited)**.** Recalling Example 3.3 earlier, note that the ∞-proof of Löb's axiom in Figure 4, left, indeed satisfies the singleton RHS restriction, and so $\ell\mathsf{IGL} \vdash (\mathsf{löb})$. On the other hand its contraposition, right, does not satisfy this restriction. We will see shortly in Example 4.8 that it is indeed *not* a theorem of ℓIGL.

4.2 An intuitionistic GL, via semantics

To give our semantic version of intuitionistic GL, we must first recall models of intuitionistic modal logic. *Birelational semantics* [29, 34] include models $\mathcal{B}$ with two relations, the intuitionistic $\leq$ and the modal $R^{\mathcal{B}}$ (parameterised with $\mathcal{B}$ to distinguish from the fixed predicate symbol R used in this paper), which duly gives it the capacity to model intuitionistic modal logics.

▶ **Definition 4.4** (Birelational semantics)**.** *A **birelational frame** $\mathcal{F}$ is a triple $(W, \leq, R^{\mathcal{F}})$, where W is a nonempty set of **worlds** equipped with a partial order $\leq$ and an **accessibility relation** $R^{\mathcal{F}} \subseteq W \times W$. We require the following frame conditions:*
(F1) *If $w \leq w'$ and $wR^{\mathcal{F}}v$, then there exists v' such that $v \leq v'$ and $w'R^{\mathcal{F}}v'$.*
(F2) *If $wR^{\mathcal{F}}v$ and $v \leq v'$, then there exists w' such that $w \leq w'$ and $w'R^{\mathcal{F}}v'$.*
*A **birelational model** is a tuple $(W, \leq, R^{\mathcal{B}}, V)$, where $(W, \leq, R^{\mathcal{B}})$ is a birelational frame and V is a **valuation** $W \to \mathcal{P}(\mathsf{Pr})$ that is monotone in $\leq$, i.e., $w \leq w'$ implies $V(w) \subseteq V(w')$.*

Let $\mathcal{B} = (W, \leq, R^{\mathcal{B}}, V)$ be a birelational model. For worlds $w \in W$ and formulas A we define the satisfaction judgement $\mathcal{B}, w \vDash A$ as follows:

- *$\mathcal{B}, w \vDash p$ if $p \in V(w)$;*
- *$\mathcal{B}, w \nvDash \bot$;*
- *$\mathcal{B}, w \vDash A \wedge B$ if $\mathcal{B}, w \vDash A$ and $\mathcal{B}, w \vDash B$;*
- *$\mathcal{B}, w \vDash A \vee B$ if $\mathcal{B}, w \vDash A$ or $\mathcal{B}, w \vDash B$;*
- *$\mathcal{B}, w \vDash A \to B$ if for all $w' \geq w$, if $\mathcal{B}, w' \vDash A$ then $\mathcal{B}, w' \vDash B$;*
- *$\mathcal{B}, w \vDash \Box A$ if for all $w' \geq w$ and for all v such that $w'R^{\mathcal{B}}v$ we have $\mathcal{B}, v \vDash A$;*
- *$\mathcal{B}, w \vDash \Diamond A$ if there exists v such that $wR^{\mathcal{B}}v$ and $\mathcal{B}, v \vDash A$.*

We write $\mathcal{B} \vDash A$ if $\mathcal{B}, w \vDash A$ for all $w \in W$.

Note that the clauses above are essentially determined by evaluating the standard translation of modal formulas within *intuitionistic* predicate models. This is why, say, evaluating a $\Box$ requires quantifying over all $w' \geq w$ (like $\forall$), but $\Diamond$ does not (like $\exists$).

▶ **Lemma 4.5** (Monotonicity lemma, [34])**.** *Let $\mathcal{B} = (W, \leq, R^{\mathcal{B}}, V)$ be a birelational model. For any formula A and $w, w' \in W$, if $w \leq w'$ and $\mathcal{B}, w \vDash A$, then $\mathcal{B}, w' \vDash A$.*

A soundness and completeness theorem is recovered in [34], similarly to the classical case (Proposition 2.4 and Theorem 2.1).

▶ **Theorem 4.6** ([34])**.** *$\ell\mathsf{IK} \vdash x : A$ (resp. $\ell\mathsf{IK4} \vdash x : A$) iff A is satisfied in all birelational models (resp. with transitive accessibility relation)*

We want to introduce a birelational counterpart of the transitive and terminating models of GL. The class of models that we will introduce here is different from the birelational semantics known for logic iGL [23, 1] which only require termination of the accessibility relation. Due to the *global* nature of $\Box$-evaluation in Definition 4.4, we here require the *composition* of $\leq$ and $R^{\mathcal{B}}$ to be terminating.

▶ **Definition 4.7.** *$\mathscr{B}$IGL is the class of birelational models $\mathcal{B} = (W, \leq, R^{\mathcal{B}}, V)$ such that:*

- *$R^{\mathcal{B}}$ is transitive; and,*
- *$(\leq; R^{\mathcal{B}})$ is terminating, i.e., there are no infinite paths $x_1 \leq y_1 R^{\mathcal{B}} x_2 \leq y_2 R^{\mathcal{B}} x_3 \dots$.*

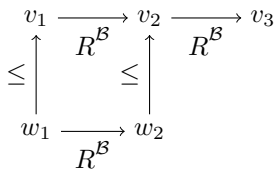
For a formula A, we write $\mathscr{B}\mathsf{IGL} \vDash A$ to mean that $\mathcal{B} \vDash A$ for all $\mathcal{B} \in \mathscr{B}\mathsf{IGL}$.

Note that the second condition, that $(\leq; R^{\mathcal{B}})$ is terminating, implies termination of $R^{\mathcal{B}}$ too, as $\leq$ is reflexive. However the converse does not in general hold. For instance if $W = (w_{ij})_{i<j<\omega}$ with:

- if $j \leq j'$ then $w_{ij} \leq w_{ij'}$;
- if $i < i'$ then $w_{ij} R^{\mathcal{B}} w_{i'j}$.

Here certainly $R^{\mathcal{B}}$ is terminating, as R-paths always have j fixed and i increasing and there are only j worlds of form w_{ij}. On the other hand there is an infinite $(\leq; R^{\mathcal{B}})$ path: $w_{01}, w_{12}, w_{23}, \dots$. The frame conditions of Definition 4.4 also hold.

▶ **Example 4.8** (Contra-Löb, revisited)**.** Recalling Examples 3.3 and 4.3 we indeed have that the contraposition of Löb's axiom, $\Diamond p \to \Diamond(p \wedge \Box\neg p)$, is not valid in $\mathscr{B}$IGL. It is falsified at world w_1 of the following $\mathscr{B}$IGL-model $\mathcal{B}$ where we assign p to all worlds. In the picture we omit transitive (and reflexive) edges of $R^{\mathcal{B}}$ (and $\leq$).



5 Soundness

The main result of this section is the soundness of ℓIGL with respect to $\mathscr{B}$IGL (Theorem 5.5 and Corollary 5.6). First we must extend the notion of satisfaction in birelational models to labelled sequents.

Let us fix a sequent $S = (\mathbf{R}, \Gamma \Rightarrow x : A)$ and a birelational model $\mathcal{B} = (W, \leq, R^{\mathcal{B}}, V)$ for the remainder of this section. An **interpretation** of S into $\mathcal{B}$ is a function $\mathcal{I} : \mathsf{Var}(S) \to W$ such that $\mathcal{I}(x) R^{\mathcal{B}} \mathcal{I}(y)$ whenever $xRy \in \mathbf{R}$. We write

$$\mathcal{B}, \mathcal{I} \vDash \mathbf{R}, \Gamma \Rightarrow x : A \quad \text{if} \quad \mathcal{B}, \mathcal{I}(x) \vDash A \text{ when } \mathcal{B}, \mathcal{I}(y) \vDash B \text{ for all } y : B \in \Gamma.$$

If $\mathcal{B}, \mathcal{I} \vDash S$ for all interpretations $\mathcal{I}$ of S into $\mathcal{B}$, we simply write $\mathcal{B} \vDash S$, and if $\mathcal{B} \vDash S$ for all models $\mathcal{B} \in \mathscr{B}\mathsf{IGL}$, we simply write $\mathscr{B}\mathsf{IGL} \vDash S$. See associated preprint [13] for a complete proof of the following.

▶ **Observation 5.1.** *$\mathcal{B} \vDash \varnothing, x : B_1, \dots, x : B_n \Rightarrow x : A$ iff $\mathcal{B} \vDash (B_1 \wedge \dots \wedge B_n) \to A$.*

In order to prove local soundness of ℓIK4 rules, we use a lifting lemma similarly to the one in [34] whose proof relies on the tree-like structure of $\mathbf{R}$.

▶ **Definition 5.2** ((Quasi-)tree-like). $\mathbf{R}$ *is a* ***tree*** *if there is* $x_0 \in \mathsf{Var}(\mathbf{R})$ *such that for each* $x \in \mathsf{Var}(\mathbf{R})$, $x \neq x_0$, *there is a unique sequence* $x_0 R x_1, x_1 R x_2, \ldots, x_m R x \in \mathbf{R}$. $\mathbf{R}$ *is a* ***quasi-tree*** *if there is some* $\mathbf{R}_0 \subseteq \mathbf{R} \subseteq \mathbf{R}_0^+$ *where* $\mathbf{R}_0$ *is a tree and* $\mathbf{R}_0^+$ *denotes the transitive closure of* $\mathbf{R}_0$. *Sequent* S *is* ***(quasi-)tree-like*** *if either* $\mathbf{R} = \varnothing$ *and* $\mathsf{Var}(\Gamma) \subseteq \{x\}$, *or* $\mathbf{R}$ *is a (resp., quasi-)tree and* $\mathsf{Var}(\Gamma) \cup \{x\} \subseteq \mathsf{Var}(\mathbf{R})$.

▶ **Lemma 5.3** (Lifting lemma). *Suppose* S *is quasi-tree-like. Let* $\mathcal{I}$ *be interpretation of* S *into* $\mathcal{B}$, $x \in \mathsf{Var}(S)$ *and* $w \geq \mathcal{I}(x)$. *There is an interpretation* $\mathcal{I}'$ *of* S *into* $\mathcal{B}$ *such that* $\mathcal{I}'(x) = w$ *and for all* $y \in \mathsf{Var}(S)$ *we have* $\mathcal{I}'(y) \geq \mathcal{I}(y)$.

Let us employ some conventions on ∞-proofs. Note that for every inference rule of $\ell\mathsf{IK4}$, except for th and cut, the premiss(es) are quasi-tree-like whenever the conclusion is. By Observation 3.5 we shall duly assume that th is not used. Note that this forces relational contexts to be *growing*, bottom-up: the relational context of a sequent always contains those below it. If a cut-formula $z : C$ has label z that does not occur in the conclusion, we may safely rename z to a variable that does. Thus we may assume that any ∞-proof in $\ell\mathsf{IGL}$ of $x : A$ has only quasi-tree-like sequents in it.

▶ **Proposition 5.4** (Local soundness). *Suppose* S *is quasi-tree-like. Let* $\mathcal{I}$ *be an interpretation of* S *into* $\mathcal{B}$ *such that* $\mathcal{B}, \mathcal{I} \not\models S$. *For any inference step of* $\ell\mathsf{IK4} \setminus \{\mathsf{th}\}$ *that* S *concludes, there is a premiss* S' *and an interpretation* $\mathcal{I}'$ *of* S' *into* $\mathcal{B}$ *such that* $\mathcal{B}, \mathcal{I}' \not\models S'$ *and* $\mathcal{I}'(z) \geq \mathcal{I}(z)$ *for all* $z \in \mathsf{Var}(S)$.

The proof uses some direct calculations in most cases and the lifting lemma (Lemma 5.3) to handle rules $\to$-r and $\Box$-r. We refer to the associated preprint [13] for the case analysis. From here, as in the classical setting for $\ell\mathsf{GL}$, we can employ a contradiction argument by infinite descent to conclude:

▶ **Theorem 5.5** (Soundness). *Suppose* S *is quasi-tree-like. Then* $\ell\mathsf{IGL} \vdash S$ *implies* $\mathscr{B}\mathsf{IGL} \models S$.

▶ **Corollary 5.6.** *If* $\ell\mathsf{IGL} \vdash x : A$ *then* $\mathscr{B}\mathsf{IGL} \models A$.

The remainder of this work is devoted to proving the converse result. The sections that follow structure the proof into the three parts according to the arrows indicated in Figure 1.

6 From birelational models to Kripke predicate models

Towards our countermodel construction in the next section we turn to *predicate models*, essentially via the standard translation, whose internal structure is richer than that of birelational models, thus providing useful invariants for the sequel.

▶ **Definition 6.1** (Predicate models). *A* ***Kripke structure*** *is a tuple*

$$(W, \leq, \{D_w\}_{w \in W}, \{\mathsf{Pr}_w\}_{w \in W}, \{R_w\}_{w \in W}), \text{ where}$$

- W *is a non-empty set of* worlds *partially ordered by* $\leq$;
- $\{D_w\}_{w \in W}$ *is a family of non-empty* ***domains****, such that* $D_w \subseteq D_{w'}$ *whenever* $w \leq w'$;
- $\{\mathsf{Pr}_w\}_{w \in W}$ *is a family of mappings* $\mathsf{Pr}_w : \mathsf{Pr} \to \mathcal{P}(D_w)$ *such that for each* $p \in \mathsf{Pr}$, $\mathsf{Pr}_w(p) \subseteq \mathsf{Pr}_{w'}(p)$ *whenever* $w \leq w'$;
- $\{R_w\}_{w \in W}$ *is a family of relations* $R_w \subseteq D_w \times D_w$ *such that* $R_w \subseteq R_{w'}$ *whenever* $w \leq w'$.

▶ **Definition 6.2** (Environment). *Let* $\mathcal{K}$ *be a Kripke structure. A* w***-environment*** *is a function* $\rho : \mathsf{Var} \to D_w$.

Note that a w-environment is also a w'-environment for any $w' \geq w$.

▶ **Definition 6.3** (Satisfaction). *For modal formula A, structure $\mathcal{K}$ and w-environment ρ, we inductively define the judgement $\mathcal{K}, w \vDash^{\rho} x : A$ as follows, where $\rho[x := d]$ is the map that sends variable x to d and agrees with ρ on all other variables:*

- $\mathcal{K}, w \vDash^{\rho} x : p$ *if* $\rho(x) \in \mathsf{Pr}_w(p)$;
- $\mathcal{K}, w \nvDash^{\rho} \bot$;
- $\mathcal{K}, w \vDash^{\rho} x : A \wedge B$ *if* $\mathcal{K}, w \vDash^{\rho} x : A$ *and* $\mathcal{K}, w \vDash^{\rho} x : B$;
- $\mathcal{K}, w \vDash^{\rho} x : A \vee B$ *if* $\mathcal{K}, w \vDash^{\rho} x : A$ *or* $\mathcal{K}, w \vDash^{\rho} x : B$;
- $\mathcal{K}, w \vDash^{\rho} x : A \to B$ *if for all* $w' \geq w$, *if* $\mathcal{K}, w' \vDash^{\rho} x : A$ *then* $\mathcal{K}, w' \vDash^{\rho} x : B$;
- $\mathcal{K}, w \vDash^{\rho} x : \Box A$ *if for all* $w' \geq w$ *and* $d \in D_{w'}$, *if* $\rho(x) R_{w'} d$, *then* $\mathcal{K}, w' \vDash^{\rho[y:=d]} y : A$;
- $\mathcal{K}, w \vDash^{\rho} x : \Diamond A$ *if there exists* $d \in D_w$ *such that* $\rho(x) R_w d$ *and* $\mathcal{K}, w \vDash^{\rho[y:=d]} y : A$.

We write $\mathcal{K} \vDash x : A$ if $\mathcal{K}, w \vDash^{\rho} x : A$ for all worlds w and w-environments ρ.

The monotonicity lemma also holds in Kripke structures.

▶ **Lemma 6.4** (Monotonicity lemma). *Let $\mathcal{K}$ be a Kripke structure. If $w \leq w'$ and $\mathcal{K}, w \vDash^{\rho} x : A$, then $\mathcal{K}, w' \vDash^{\rho} x : A$.*

To capture transitivity in Kripke structures, it is sufficient to require each R_w to be transitive, which can be considered as a local condition on Kripke structures. However interpreting termination requires us to consider the interactions between $\leq$ and R_w similarly to the previous section.

Let $\mathcal{K}$ be a Kripke structure. We write D_W for the set of ordered pairs of the form (w, d) with $w \in W$ and $d \in D_w$. We define the two binary relations $\leq_{D_W}, R_{D_W} \subseteq D_W \times D_W$ as:

- $(w, d) \leq_{D_W} (w', d')$ iff $w \leq w'$ and $d = d'$;
- $(w, d) R_{D_W} (w', d')$ iff $w = w'$ and $d R_w d'$.

▶ **Definition 6.5.** *Write $\mathscr{P}$IGL for the class of Kripke structures $\mathcal{K}$ satisfying the following:*

- *for all $w \in W$, R_w is* transitive*; and*
- *relation $(\leq_{D_W}; R_{D_W})$ is* terminating*, i.e., there are no infinite paths $(w_1, d_1) \leq_{D_W} (w_2, d_1) R_{D_W} (w_2, d_2) \leq_{D_W} (w_3, d_2) R_{D_W} (w_3, d_3) \ldots$.*

We write $\mathscr{P}\mathsf{IGL} \vDash A$ to mean that $\mathcal{K} \vDash x : A$ for all $\mathcal{K} \in \mathscr{P}\mathsf{IGL}$ and any $x \in \mathsf{Var}$.

The same construction converting a predicate structure into a birelational model from [34, Section 8.1.1] can be used to prove the following result.

▶ **Proposition 6.6.** *If $\mathscr{B}\mathsf{IGL} \vDash A$ then $\mathscr{P}\mathsf{IGL} \vDash A$.*

7 Completeness of a multi-succedent calculus via determinacy

Towards completeness we perform a countermodel construction using a proof search strategy based on an intuitionistic *multi-succedent* calculus. This is inspired by analogous arguments for intuitionistic predicate logic, e.g. in [37], but adapted to a non-wellfounded setting.

▶ **Definition 7.1** (Multi-succedent intuitionistic calculus). *The system $\mathsf{m}\ell\mathsf{IK4}$ is the restriction of $\ell\mathsf{K4}$ where the $\Box$-right and $\to$-right rules must have exactly one formula on the RHS. We write $\mathsf{m}\ell\mathsf{IGL}$ for the class of ∞-proofs of $\mathsf{m}\ell\mathsf{IK4}$.*

The remainder of this section is devoted to proving the following completeness result:

▶ **Theorem 7.2.** *If $\mathscr{P}\mathsf{IGL} \vDash A$ then $\mathsf{m}\ell\mathsf{IGL} \vdash x : A$.*

Here we informally describe the construction of the proof search tree. For a more formal treatment of parts below we refer to the associated preprint [13]. During bottom-up proof search we will always proceed according to the three following phases in order of priority: applications of rule tr, applications of invertible rules (other than tr), and application of non-invertible rules. The first two together we call the **invertible phase**, the other the **non-invertible phase**. In fact, invertibility of all rules except $\Box$-r and $\to$ -r is guaranteed by applying suitable contractions at the same time. For instance, we apply the following derivable " macro" rules for $\to$ on the left and $\Diamond$ on the right:

$$\to\text{-}l\ \frac{\mathbf{R}, \Gamma, x : A \to B \Rightarrow \Delta, x : A \quad \mathbf{R}, \Gamma, x : A \to B, x : B \Rightarrow \Delta}{\mathbf{R}, \Gamma, x : A \to B \Rightarrow \Delta} \qquad \Diamond\text{-}r\ \frac{\mathbf{R}, xRy, \Gamma \Rightarrow \Delta, x : \Diamond A, y : A}{\mathbf{R}, xRy, \Gamma \Rightarrow \Delta, x : \Diamond A}$$

By furthermore building weakening into the identity, i.e. allowing initial sequents of form $\mathbf{R}, \Gamma, x : p \Rightarrow \Delta, x : p$, structural rules become redundant for proof search. In the invertible phase we can apply the rules in any order. The non-invertible phase creates predecessor nodes for each possible rule instance of $\to$ -r and $\Box$-r.

In order to carry out our countermodel construction to show completeness of $\mathsf{m}\ell\mathsf{IGL}$, we rely on two features of the proof search space that digress from usual countermodel constructions in intuitionistic predicate logic [37]. The first important feature of this strategy is that the invertible phase is always finite and ends in so-called *saturated sequents*, i.e., sequents for which any bottom-up rule application (other than id and $\bot$-l) yields a premiss that is the same sequent, up to multiplicities (see [13] for formal definition). Looking ahead to the countermodel, worlds w are defined on the basis of invertible phases and will as a result all have a finite domain D_w.

▶ **Lemma 7.3.** *Following the proof search strategy described above, each invertible phase constructs a finite subtree that has saturated sequents at its leaves.*

Secondly, and perhaps more importantly, we employ a technique from non-wellfounded proof theory to help us organise the countermodel constructed from a failed proof search: we appeal to determinacy of a *proof search game*. To understand the motivation here, a classical countermodel-from-failed-proof search argument proceeds (very roughly) as follows: (1) assume a formula is not provable; (2) for each rule instance there must be an unprovable premiss; (3) continue in this way to construct an (infinite) " unprovable" branch; (4) extract a countermodel from this branch. In our setting we will need the branch obtained through the process above to be *not progressing* in order to deduce that the structure we extract is indeed one of $\mathscr{P}\mathsf{IGL}$. However the local nature of the process above does not at all guarantee that this will be the case. For this, we rely on *(lightface) analytic determinacy*, which is equivalent to the existence of $0^\sharp$ over ZFC [19], of the corresponding proof search game. As we only use it as a tool and it is not the main focus of our work, we refer to the associated preprint [13] for more details.

▶ **Proposition 7.4.** *Given an unprovable sequent S there is a subtree T of the proof search space rooted at S, closed under bottom-up non-invertible rule application*[4] *such that each infinite branch of T is not progressing.*

The properties in Lemma 7.3 and Proposition 7.4 enable us to construct a countermodel:

[4] I.e. if $S_0 \in T$ concludes some non-invertible step with premiss S_1, then also $S_1 \in T$.

▶ **Theorem 7.5** (Countermodel construction). *If* mℓIGL $\nvdash \mathbf{R}, \Gamma \Rightarrow \Delta$, *then there is a structure* $\mathcal{K} \in \mathscr{P}$IGL *with* w*-environment* ρ *such that for labelled formulas* $x : A$ *we have*

- *if* $x : A \in \Gamma$, *then* $\mathcal{K}, w \vDash^{\rho} x : A$, *and,*
- *if* $x : A \in \Delta$, *then* $\mathcal{K}, w \nvDash^{\rho} x : A$.

From here Theorem 7.2 easily follows. Notice that we did not use rule cut in the proof search strategy so we can actually conclude a stronger result:

▶ **Corollary 7.6.** mℓIGL *is* cut-free *complete over* $\mathscr{P}$IGL.

8 Completeness of ℓIGL via (partial) cut-elimination

To obtain completeness of ℓIGL for $\mathscr{B}$IGL, we will simulate mℓIGL using cuts in an extension of ℓIGL that allows reasoning over disjunctions of labelled formulas. We apply a (partial) cut-elimination procedure to eliminate these disjunctions, and then note that any resulting proof is already one of ℓIGL.

We shall use metavariables φ, ψ etc. to vary over *disjunctions* of labelled formulas. I.e. $\varphi, \psi, \ldots \quad ::= \quad (x : A) \quad | \quad \varphi \vee \psi$.

▶ **Definition 8.1.** *The system* $\vee\ell$IK4 *is the extension of* ℓIK4 *by duly adapting identity, cut, structural and* $\vee$ *rules to allow for* φ*-formulas. In particular it has the following* $\vee$ *rules:*

$$\vee\text{-}l\ \frac{\mathbf{R}, \Gamma, \varphi_0 \Rightarrow \psi \quad \mathbf{R}, \Gamma, \varphi_1 \Rightarrow \psi}{\mathbf{R}, \Gamma, \varphi_0 \vee \varphi_1 \Rightarrow \psi} \qquad \vee\text{-}r\ \frac{\mathbf{R}, \Gamma \Rightarrow \varphi_i}{\mathbf{R}, \Gamma \Rightarrow \varphi_0 \vee \varphi_1}\ i \in \{0, 1\}$$

The **degree** of a formula $(x_1 : A_1) \vee \cdots \vee (x_d : A_d)$ is d. The degree of a cut is the degree d of its cut-formula, in which case we say it is a d-cut. The degree of a preproof is the maximum degree of its cuts (when this is well-defined).

▶ **Proposition 8.2.** *If* mℓIK4 *has a cut-free* ∞*-proof of* $x : A$, $\vee\ell$IK4 *has one of bounded degree.*

Moreover, immediately from definitions, we have:

▶ **Observation 8.3.** *A* $\vee\ell$IK4 ∞*-proof containing only labelled formulas is a* ℓIK4 ∞*-proof.*

Thus, to conclude completeness of ℓIGL for $\mathscr{B}$IGL, it suffices to eliminate the use of disjunctions of labelled formulas in $\vee\ell$IK4 ∞-proofs. We will prove this by a partial cut-elimination procedure, reducing cuts over disjunctions of labelled formulas until they are on labelled formulas.

For the remainder of this section we work only with preproofs without thinning th, by Observation 3.5. Recall that this means that relational contexts are *growing*, bottom-up. For a sequent S write $\mathbf{R}_S$ for its relational context. I.e. if S is $\mathbf{R}, \Gamma \Rightarrow \varphi$, then $\mathbf{R}_S := \mathbf{R}$.

▶ **Lemma 8.4** (Invertibility). *If* $\vee\ell$IK4 *has a* ∞*-proof* P *of* $\mathbf{R}, \Gamma, \varphi_0 \vee \varphi_1 \Rightarrow \psi$ *then it also has* ∞*-proofs* P_i *of* $\mathbf{R}, \Gamma, \varphi_i \Rightarrow \psi$, *for* $i \in \{0, 1\}$. *Moreover, for each branch* $(S_i)_{i<\omega}$ *of* P_i *there is a branch* $(S_i')_{i<\omega}$ *of* P *such that* $\mathbf{R}_{S_i} \subseteq \mathbf{R}_{S_i'}$ *for all* $i < \omega$.

From here the key cut-reduction for φ-formulas is simply,

$$\text{cut}\ \frac{\vee\text{-}r\ \dfrac{\begin{array}{c}P\\ \mathbf{R}, \Gamma, \Rightarrow \chi_i\end{array}}{\mathbf{R}, \Gamma \Rightarrow \chi_0 \vee \chi_1} \qquad \begin{array}{c}Q\\ \mathbf{R}, \Gamma', \chi_0 \vee \chi_1 \Rightarrow \varphi\end{array}}{\mathbf{R}, \Gamma, \Gamma' \Rightarrow \varphi} \quad \rightsquigarrow \quad \text{cut}\ \frac{\begin{array}{c}P\\ \mathbf{R}, \Gamma \Rightarrow \chi_i\end{array} \qquad \begin{array}{c}Q_i\\ \mathbf{R}, \Gamma', \chi_i \Rightarrow \varphi\end{array}}{\mathbf{R}, \Gamma, \Gamma' \Rightarrow \varphi}$$

where Q_i is obtained by Lemma 8.4 above. Note that this reduction " produces' a cut of lower complexity. Commutative cut-reduction cases, where the cut-formula is not principal on the left, are standard and always produce. Note that, thanks to invertibility, we do not consider commutations over the right premiss of a φ-cut. However commutative cases may *increase* the heights of progress points; for instance when commuting over a $\Box$-l step,

$$\mathsf{cut}\,\dfrac{\Box\text{-}l\,\dfrac{\overset{P}{\vdots}\ \mathbf{R}, xRy, \Gamma, y : A \Rightarrow \chi}{\mathbf{R}, xRy, \Gamma, x : \Box A \Rightarrow \chi} \qquad \overset{Q}{\vdots}\ \mathbf{R}, xRy, \Gamma', \chi \Rightarrow \varphi}{\mathbf{R}, xRy, \Gamma, \Gamma', x : \Box A \Rightarrow \varphi} \quad \rightsquigarrow \quad \Box\text{-}l\,\dfrac{\mathsf{cut}\,\dfrac{\overset{P}{\vdots}\ \mathbf{R}, xRy, \Gamma, y : A \Rightarrow \chi \qquad \overset{Q}{\vdots}\ \mathbf{R}, xRy, \Gamma', \chi \Rightarrow \varphi}{\mathbf{R}, xRy, \Gamma, \Gamma', y : A \Rightarrow \varphi}}{\mathbf{R}, xRy, \Gamma, \Gamma', x : \Box A \Rightarrow \varphi}$$

observe that progress points in Q have been raised. Thus, to show that the limit of cut-reduction is progressing we will need appropriate invariants, requiring additional notions.

A **bar** of a preproof is a (necessarily finite, by König's Lemma) antichain intersecting each infinite branch. Each bar B induces a (necessarily finite) subtree $\lfloor B \rfloor$ of nodes beneath (and including) it. Given a cut-reduction $P \rightsquigarrow_{\mathsf{r}} P'$ we associate to each bar B of P a bar $\mathsf{r}(B)$ of P' in the natural way. In particular we have that $\mathsf{r}(B)$ satisfies the following properties:

▶ **Lemma 8.5** (Trace preservation). *If $P \rightsquigarrow_{\mathsf{r}} P'$ and B a bar of P, for any sequent $S' \in \mathsf{r}(B)$ there is a sequent $S \in B$ such that $\mathbf{R}_{S'} \supseteq \mathbf{R}_S$.*

This lemma allows us to keep track of a fixed amount of progress information during cut-elimination, sidestepping the issue that commutative cases raise progress points. Note that we really need the $\mathbf{R}_{S'} \supseteq \mathbf{R}_S$ due to the $\Diamond$-l commutative case:

$$\mathsf{cut}\,\dfrac{\Diamond\text{-}l\,\dfrac{\overset{P}{\vdots}\ \mathbf{R}, xRy, \Gamma, y : A \Rightarrow \chi}{\mathbf{R}, \Gamma, x : \Diamond A \Rightarrow \chi} \qquad \overset{Q}{\vdots}\ \mathbf{R}, \Gamma', \chi \Rightarrow \varphi}{\mathbf{R}, \Gamma, \Gamma', x : \Diamond A \Rightarrow \varphi} \quad \rightsquigarrow \quad \Diamond\text{-}l\,\dfrac{\mathsf{cut}\,\dfrac{\overset{P}{\vdots}\ \mathbf{R}, xRy, \Gamma, y : A \Rightarrow \chi \qquad \overset{xRy, Q}{\vdots}\ xRy, \mathbf{R}, \Gamma', \chi \Rightarrow \varphi}{\mathbf{R}, xRy, \Gamma, \Gamma', y : A \Rightarrow \varphi}}{\mathbf{R}, \Gamma, \Gamma', x : \Diamond A \Rightarrow \varphi}$$

where the preproof xRy, Q is obtained from Q by prepending xRy to the LHS of each sequent. From here we can effectively reduce infinitary cut-elimination to finitary cut-elimination, by eliminating cuts beneath higher and higher bars. The step case is given by:

▶ **Lemma 8.6** (Productivity). *For any ∞-proof P and bar B there is a sequence of cut-reductions $P = P_0 \rightsquigarrow_{\mathsf{r}_1} \cdots \rightsquigarrow_{\mathsf{r}_n} P_n$ such that $\mathsf{r}_n \cdots \mathsf{r}_1(B)$ has no d-cuts beneath it in P_n.*

The argument proceeds in a relatively standard way: by induction on the number of d-cuts in $\lfloor B \rfloor$, with a subinduction on the multiset of the distances of topmost d-cuts in $\lfloor B \rfloor$ from B, where the distance is the length of the shortest path from the cut to B. Applying Lemma 8.6 to higher and higher bars allows us to reduce cut-degrees as required:

▶ **Proposition 8.7** (Degree-reduction). *For each ∞-proof of $\vee\ell\mathsf{IK4}$ of $x : A$ of degree d, there is one of degree $< d$.*

Importantly here, the preservation of progress in the limit crucially relies on Lemma 8.5, under König's Lemma. Finally by induction on degree we have:

▶ **Corollary 8.8** (Partial cut-elimination). *If $\vee\ell\mathsf{IK4}$ has a bounded-degree ∞-proof of $x : A$, then it has one containing only labelled formulas.*

From here we have our desired converse to Theorem 5.5, following from Observation 8.3 and Propositions 6.6 and 8.2, Theorem 7.2 and Corollary 8.8:

▶ **Theorem 8.9** (Completeness). *If* $\mathscr{B}\mathsf{IGL} \vDash A$ *then* $\ell\mathsf{IGL} \vdash x : A$.

9 Conclusions

We have recovered several intuitionistic formulations of GL, both syntactically and semantically, in the tradition of Simpson, on intuitionistic modal logic [34], and Simpson, on non-wellfounded proofs [35]. We proved the equivalence of all these formulations, cf. Figure 1, motivating the following definition:

▶ **Definition 9.1.** IGL *is the modal logic given by any/all of the nodes of Figure 1.*

Thanks to the methodology we followed, IGL satisfies Simpson's *requirements* from [34]. IGL also interprets (classical) GL along the Gödel-Gentzen negative translation.

It would be interesting to examine IGL as a logic of provability, returning to the origins of GL. In particular IGL (even IK) has a *normal* $\Diamond$, distributing over $\vee$, so let us point out that it is not sound to interpret $\Diamond$ as consistency $\neg\Box\neg$. At the same time (effective) model-theoretic readings of the $\Diamond$ stumble on the consequence (already of IK) $\Box(A \to B) \to \Diamond A \to \Diamond B$. We expect IGL to rather correspond to the provability logic of some *model* of HA.

To this end it would be pertinent to develop a bona fide *axiomatisation* of IGL. As far as we know it is possible that IGL is simply the extension of IK4 by Löb's axiom but, similarly to other "I" logics, we expect that further axioms involving $\Diamond$ will be necessary. On a related note, we believe that a full cut-elimination result holds for ℓIGL, in particular by extending our argument for $\vee\ell$IGL. Crucial here is that no cut-reductions *delete* progress points, since we do not cut on relational atoms (cf. a key $\Box$ reduction). A full development of this is beyond the scope (and allocated space) of this work.

References

1 Mohammad Ardeshir and Motjaba Mojtahedi. The Σ_1-provability logic of HA. *Annals of Pure and Applied Logic*, 169(10):997–1043, 2018.

2 David Baelde, Amina Doumane, and Alexis Saurin. Infinitary proof theory: the multiplicative additive case. In *25th EACSL Annual Conference on Computer Science Logic, CSL 2016*, volume 62 of *LIPIcs*, pages 42:1–17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016.

3 Stefano Berardi and Makoto Tatsuta. Equivalence of inductive definitions and cyclic proofs under arithmetic. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–12. IEEE, 2017.

4 Lars Birkedal, Rasmus Ejlers Møgelberg, Jan Schwinghammer, and Kristian Støvring. First steps in synthetic guarded domain theory: step-indexing in the topos of trees. *Logical Methods in Computer Science*, 8, 2012.

5 Patrick Blackburn, Maarten De Rijke, and Yde Venema. *Modal logic*, volume 53. Cambridge University Press, 2001.

6 Patrick Blackburn, Johan van Benthem, and Frank Wolter, editors. *Handbook of Modal Logic*, volume 3 of *Studies in logic and practical reasoning*. North-Holland, 2007.

7 James Brotherston and Alex Simpson. Sequent calculi for induction and infinite descent. *Journal of Logic and Computation*, 21(6):1177–1216, 2011.

8 Ranald Clouston and Rajeev Goré. Sequent calculus in the topos of trees. In *Foundations of Software Science and Computation Structures: 18th International Conference, FOSSACS 2015*, pages 133–147. Springer, 2015.

9 Anupam Das. On the logical complexity of cyclic arithmetic. *Logical Methods in Computer Science*, 16, 2020.

10 Anupam Das and Sonia Marin. Brouwer meets Kripke: constructivising modal logic, 2022. Post on *The Proof Theory Blog* (accessed 2 August 2023). `https://prooftheory.blog/2022/08/19/brouwer-meets-kripke-constructivising-modal-logic/`.

11 Anupam Das and Sonia Marin. On intuitionistic diamonds (and lack thereof). In *Automated Reasoning with Analytic Tableaux and Related Methods*, volume 14278 of *Lecture Notes in Computer Science*, pages 283–301. Springer, 2023.

12 Anupam Das and Damien Pous. Non-wellfounded proof theory for (Kleene+Action) (Algebras+Lattices). In *27th EACSL Annual Conference on Computer Science Logic, CSL 2018*, volume 119 of *LIPIcs*, pages 19:1–19:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018.

13 Anupam Das, Iris van der Giessen, and Sonia Marin. Intuitionistic gödel-löb logic, à la simpson: labelled systems and birelational semantics, 2023. `arXiv:2309.00532`.

14 Gisèle Fischer Servi. On modal logic with an intuitionistic base. *Studia Logica*, 36:141–149, 1977.

15 Iris van der Giessen. *Uniform Interpolation and Admissible Rules: Proof-theoretic investigations into (intuitionistic) modal logics*. PhD thesis, Utrecht University, 2022.

16 Iris van der Giessen and Rosalie Iemhoff. Sequent calculi for intuitionistic Gödel–Löb logic. *Notre Dame Journal of Formal Logic*, 62(2):221–246, 2021.

17 Rajeev Goré and Revantha Ramanayake. Valentini's cut-elimination for provability logic resolved. In Carlos Areces and Robert Goldblatt, editors, *Proceedings of the 7th conference on Advances in Modal Logic*, pages 67–86. College Publications, 2008.

18 Rajeev Goré and Ian Shillito. Direct elimination of additive-cuts in GL4ip: verified and extracted. In *Advances in Modal Logic 14*, 2022.

19 Leo Harrington. Analytic determinacy and $0^{\#}$. *The Journal of Symbolic Logic*, 43(4):685–693, 1978. `doi:10.2307/2273508`.

20 Rosalie Iemhoff. Reasoning in circles. In Jan van Eijck, Joost J. Joosten, and Rosalie Iemhoff, editors, *Liber Amicorum Alberti. A Tribute to Albert Visser*, pages 165–178. College Publications, 2016.

21 G. Alex Kavvos. Intensionality, intensional recursion and the Gödel-Löb axiom. *FLAP*, 8(8):2287–2312, 2021.

22 Daniel Leivant. On the proof theory of the modal logic for arithmetic provability. *The Journal of Symbolic Logic*, 46(3):531–538, 1981.

23 Tadeusz Litak. Constructive modalities with provability smack. In Guram Bezhanishvili, editor, *Leo Esakia on Duality in Modal and Intuitionistic Logics*, volume 4 of *Outstanding Contributions to Logic*, pages 187–216. Springer Netherlands, 2014.

24 Anders Moen. The proposed algorithms for eliminating cuts in the provability calculus GLS do not terminate. In *The 13th Nordic Workshop in Programming Theory*, 2001.

25 Mojtaba Mojtahedi. On provability logic of HA, 2022. `doi:10.48550/arXiv.2206.00445`.

26 Hiroshi Nakano. A modality for recursion. In *Proceedings Fifteenth Annual IEEE Symposium on Logic in Computer Science (Cat. No. 99CB36332)*, pages 255–266. IEEE, 2000.

27 Sara Negri. Proof analysis in modal logic. *Journal of Philosophical Logic*, 34:507–544, 2005.

28 Damian Niwiński and Igor Walukiewicz. Games for the μ-calculus. *Theoretical Computer Science*, 163(1-2):99–116, 1996.

29 Gordon Plotkin and Colin Stirling. A framework for intuitionistic modal logics. In *Proceedings of the 1st Conference on Theoretical Aspects of Reasoning about Knowledge (TARK)*, pages 399–406, 1986.

30 Francesca Poggiolesi. A purely syntactic and cut-free sequent calculus for the modal logic of provability. *The Review of Symbolic Logic*, 2(4):593–611, 2009.

31 Katsumi Sasaki. Löb's axiom and cut-elimination theorem. *Academia Mathematical Sciences and Information Engineering Nanzan University*, 1:91–98, 2001.

32 Krister Segerberg. Results in non-classical propositional logic. *Uppsala: Filosofiska Föreningen och Filosofiska Institutionen vid Uppsala Universitet.*, 1971.

33 Daniyar S. Shamkanov. Circular proofs for the Gödel-Löb provability logic. *Mathematical Notes*, 96:575–585, 2014.

34 Alex K. Simpson. *The Proof Theory and Semantics of Intuitionistic Modal Logic*. PhD thesis, University of Edinburgh, 1994.

35 Alex K. Simpson. Cyclic arithmetic is equivalent to Peano arithmetic. In *Foundations of Software Science and Computation Structures Proceedings*, volume 10203 of *Lecture Notes in Computer Science*, pages 283–300, 2017.

36 Robert M. Solovay. Provability interpretations of modal logic. *Israel Journal of Mathematics*, 25:287–304, 1976.

37 Gaisi Takeuti. *Proof Theory*. New York, N.Y., U.S.A.: Sole distributors for the U.S.A. and Canada, Elsevier Science Pub. Co., 1975.

38 Silvio Valentini. The modal logic of provability: cut-elimination. *Journal of Philosophical logic*, pages 471–476, 1983.

Quantifiers Closed Under Partial Polymorphisms

Anuj Dawar
Department of Computer Science and Technology, University of Cambridge, UK

Lauri Hella
Faculty of Information Technology and Communication Sciences, Tampere University, Finland

Abstract

We study Lindström quantifiers that satisfy certain closure properties which are motivated by the study of polymorphisms in the context of constraint satisfaction problems (CSP). When the algebra of polymorphisms of a finite structure $\mathfrak{B}$ satisfies certain equations, this gives rise to a natural closure condition on the class of structures that map homomorphically to $\mathfrak{B}$. The collection of quantifiers that satisfy closure conditions arising from a fixed set of equations are rather more general than those arising as CSP. For any such conditions $\mathcal{P}$, we define a pebble game that delimits the distinguishing power of the infinitary logic with all quantifiers that are $\mathcal{P}$-closed. We use the pebble game to show that the problem of deciding whether a system of linear equations is solvable in $\mathbb{Z}/2\,\mathbb{Z}$ is not expressible in the infinitary logic with all quantifiers closed under a near-unanimity condition.

2012 ACM Subject Classification Theory of computation → Finite Model Theory

Keywords and phrases generalized quantifiers, constraint satisfaction problems, pebble games, finite variable logics, descriptive complexity theory

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.23

Funding *Anuj Dawar*: funded by UK Research and Innovation (UKRI) under the UK government's Horizon Europe funding guarantee: grant number EP/X028259/1.

1 Introduction

Generalized quantifiers, also known as Lindström quantifiers, have played a significant role in the development of finite model theory. The subject of finite model theory is the expressive power of logics in the finite, and Lindström quantifiers provide a very general and abstract method of constructing logics. We can associate with any isomorphism-closed class of structures $\mathcal{K}$, a quantifier $Q_{\mathcal{K}}$ so that the extension $L(Q_{\mathcal{K}})$ of a logic L with the quantifier $Q_{\mathcal{K}}$ is the *minimal* extension of L that can express the class $\mathcal{K}$, subject to certain natural closure conditions. For this reason, comparing the expressive power of logics with Lindström quantifiers is closely related to comparing the descriptive complexity of the underlying classes of structures.

Another reason for the significance of Lindström quantifiers is that we have powerful methods for proving inexpressibility in logics with such quantifiers. In particular, games, based on Hella's bijection games [17], are the basis of the most common inexpressivity results that have been obtained in finite model theory. The k,n-bijection game was introduced by Hella to characterize equivalence in the logic $L^k_{\infty\omega}(\mathbf{Q}_n)$, which is the extension of the infinitary logic with k variables by means of all n-ary Lindström quantifiers. A quantifier $Q_{\mathcal{K}}$ is n-ary if the class $\mathcal{K}$ is defined over a vocabulary σ in which all relation symbols have arity n or less. In particular, the $k,1$-bijection game, often called the k-pebble bijection game, characterizes equivalence in $L^k_{\infty\omega}(\mathbf{Q}_1)$ which has the same expressive power as $C^k_{\infty\omega}$, the k-variable infinitary logic with counting. Hella uses the k,n-bijection game to show that, for each n, there is an $(n+1)$-ary quantifier that is not definable in $L^k_{\infty\omega}(\mathbf{Q}_n)$ for any k.

The $k,1$-bijection game has been widely used to establish inexpressibility results for $C^k_{\infty\omega}$. The k,n-bijection game for $n>1$ has received relatively less attention. One reason is that, while equivalence in $C^k_{\infty\omega}$ is a polynomial-time decidable relation, which is in fact a relation

much studied on graphs in the form of the Weisfeiler-Leman algorithm, in contrast the relation induced by the k, n-bijection game for $n > 1$ reduces to isomorphism on graphs and is intractable in general. Nonetheless, there is some interest in studying, for example, the non-trivial equivalence induced by $L^k_{\infty\omega}(\mathbf{Q}_2)$ on structures with a ternary relation. Grochow and Levet [16] investigate this relation on finite groups.

A second reason why the logics $L^\omega_{\infty\omega}(\mathbf{Q}_n)$ have attracted less interest is that in finite model theory we are often interested in logics that are closed under vectorized first-order interpretations. This is especially so in descriptive complexity as the complexity classes we are trying to characterize usually have these closure properties. While $L^\omega_{\infty\omega}(\mathbf{Q}_1)$ is closed under first-order interpretations, this is not the case for $L^\omega_{\infty\omega}(\mathbf{Q}_n)$ for $n > 1$. Indeed, the closure of $L^\omega_{\infty\omega}(\mathbf{Q}_2)$ under interpretations already includes $\mathbf{Q}_n$ for all n and so can express all properties of finite structures. So, it seems that beyond $L^\omega_{\infty\omega}(\mathbf{Q}_1)$, interesting logics from the point of view of complexity necessarily include quantifiers of all arities.

One way of getting meaningful logics that include quantifiers of unbounded arity is to consider quantifiers restricted to stronger closure conditions than just closure under isomorphisms. In recent work, novel game-based methods have established new inexpresibilty results for such logics, i.e. logics with a wide class of quantifiers of unbounded arity, but satisfying further restrictions. An important example is the class of linear-algebraic quantifiers, introduced in [8] which is the closure under interpretations of binary quantifiers invariant under invertible linear maps over finite fields. Equivalence in the resulting logic is characterized by the invertible map games introduced in [10]. These games are used in a highly sophisticated way by Lichter [21] to demonstrate a polynomial-time property that is not definable in fixed-point logic with rank introduced in [9, 15]. The result is extended to the infinitary logic with all linear-algebraic quantifiers in [7].

Another example is the recent result of Hella [18] showing a hierarchy theorem for quantifiers based on *constraint satisfaction problems* (CSP), using a novel game. Recall that for a fixed relational structure $\mathfrak{B}$, $\mathsf{CSP}(\mathfrak{B})$ denotes the class of structures that map homomorphically to $\mathfrak{B}$. Hella establishes that, for each $n > 1$, there is a structure $\mathfrak{B}$ with $n + 1$ elements that is not definable in $L^\omega_{\infty\omega}(\mathbf{Q}_1, \mathbf{CSP}_n)$, where $\mathbf{CSP}_n$ denotes the collection of all quantifiers of the form $Q_{\mathsf{CSP}(\mathfrak{B}')}$ where $\mathfrak{B}'$ has at most n elements. Note that $\mathbf{CSP}_n$ includes quantifiers of all arities.

The interest in CSP quantifiers is inspired by the great progress that has been made in classifying constraint satisfaction problems in recent years, resulting in the dichotomy theorem of Bulatov and Zhuk [5, 24] stating that, for any structure $\mathfrak{B}$, $\mathsf{CSP}(\mathfrak{B})$ is either polynomial time computable, or NP-complete. The so-called algebraic approach to the classification of CSP has shown that the dividing line between these alternatives is completely determined by the algebra of polymorphisms of the structure $\mathfrak{B}$. More specifically, it is completely determined by the equational theory of this algebra. As we make explicit in Section 3 below, equations satisfied by the polymorphisms of $\mathfrak{B}$ naturally give rise to certain closure properties for the class of structures $\mathsf{CSP}(\mathfrak{B})$, which we describe by *partial polymorphisms*.

The notion of partial polymorphism, as well as that of polymorphism, goes back to Geiger and Bodnarchuk et al [14, 4], who proved a one-to-one correspondence between sets of relations that are closed under definability by conjunctions of atomic formulas (i.e., positive primitive formulas without existential quantification) and sets of partial functions that contain all projections and are closed under composition and restriction. Later Romov [22] formulated this correspondence as a Galois connection: a relation R is definable in a structure $\mathfrak{B}$ by a conjunction of atomic formulas if, and only if, every partial function that is a partial polymorphism of $\mathfrak{B}$ is also a partial polymorphism of R.

Partial polymorphism offer a more fine-grained tool for comparing the complexity of CSPs and related problems than the method based on total polymorphisms. Schnoor and Schnoor [23] showed that the above mentioned Galois connection for partial polymorphisms can be used for analysing the complexity of enumerating the solutions of $\mathsf{CSP}(\mathfrak{B})$. As an application, they proved that, for a Boolean $\mathsf{CSP}(\mathfrak{B})$, there exists an efficient enumeration algorithm if, and only if, $\mathsf{CSP}(\mathfrak{B})$ itself is polynomial time computable. Furthermore, Jonsson et al. [20] proved that, for two structures $\mathfrak{B}$ and $\mathfrak{C}$ with the same domain, if all partial polymorphisms of $\mathfrak{B}$ are also partial polymorphisms of $\mathfrak{C}$, then $\mathsf{CSP}(\mathfrak{C})$ can be reduced to $\mathsf{CSP}(\mathfrak{B})$ by a polynomial time reduction that increases the size of input structures by at most a constant. As a corollary, they proved a tight result on the relative time complexity of the corresponding CSPs: if $\mathsf{CSP}(\mathfrak{B})$ can be solved in time $2^{(c+\varepsilon)n}$ for every $\varepsilon > 0$, then so can $\mathsf{CSP}(\mathfrak{C})$.

A central aim of the present paper is to initiate the study of quantifiers closed under partial polymorphisms. We present a Spoiler-Duplicator pebble game, based on bijection games, which exactly characterises the expressive power of such quantifiers. More precisely, there is such a game for any suitable family $\mathcal{P}$ of partial polymorphisms. The exact definition of a quantifier being closed under the family $\mathcal{P}$ is given in Section 3; here we just remark that this notion is *not* based on the Galois connection between relations and partial polymorphisms. The definition the game and the proof of the characterization are given in Section 4.

As a case study, we consider the partial polymorphisms described by a *near-unanimity* condition. It is known since the seminal work of Feder and Vardi [13] that if a structure $\mathfrak{B}$ admits a near-unanimity polymorphism, then $\mathsf{CSP}(\mathfrak{B})$ has *bounded width*, i.e. it (or more precisely, its complement) is definable in Datalog. On the other hand, the problem of determining the solvability of a system of equations over the two-element field $\mathbb{Z}/2\mathbb{Z}$ is the classic example of a tractable CSP that is not of bounded width. Indeed, it is not even definable in $C^\omega_{\infty\omega}$ [2]. We show that the collection of quantifiers that are closed under near-unanimity partial polymorphisms is much richer than the classes $\mathsf{CSP}(\mathfrak{B})$ where $\mathfrak{B}$ has a near-unanimity polymorphism. The collection not only includes quantifiers which are not CSP, but it also includes CSP quantifiers which are not of bounded width, including intractable ones such as hypergraph colourability. Still, we are able to show that the problem of solving systems of equations over $\mathbb{Z}/2\mathbb{Z}$ is not definable in the extension of $C^\omega_{\infty\omega}$ with *all* quantifiers closed under near-unanimity partial polymorphisms. This sheds new light on the inter-definability of constraint satisfaction problems. For instance, while it follows from the arity hierarchy of [17] that the extension of $C^\omega_{\infty\omega}$ with a quantifier for graph 3-colourability still cannot define solvability of systems of equations over $\mathbb{Z}/2\mathbb{Z}$, our result shows this also for the extension of $C^\omega_{\infty\omega}$ with all hypergraph colourability quantifiers.

2 Preliminaries

We assume basic familiarity with logic, and in particular the logics commonly used in finite model theory (see [11], for example). We write $L^k_{\infty\omega}$ to denote the infinitary logic (that is, the closure of first-order logic with infinitary conjunctions and disjunctions) with k variables and $L^\omega_{\infty\omega}$ for $\bigcup_{k\in\omega} L^k_{\infty\omega}$. We are mainly interested in the extensions of these logics with generalized quantifiers, which we introduce in more detail in Section 2.1 below.

We use Fraktur letters $\mathfrak{A}, \mathfrak{B}, \ldots$ to denote structures and the corresponding Roman letters $A, B, \ldots$ to denote their universes. Unless otherwise mentioned, all structures are assumed to be finite. We use function notation, e.g. $f : A \to B$ to denote possibly *partial* functions. If $f : A \to B$ is a function and $\vec{a} \in A^m$ a tuple, we write $f(\vec{a})$ for the tuple in B^m obtained

by applying f to $\vec{a}$ componentwise. This extends to functions of arity greater than 1. Thus, if $f : A^n \to B$ is a function of arity n and $\vec{a}_1, \ldots, \vec{a}_m \in A^n$ is a sequence of n-tuples, then $f(\vec{a}_1, \ldots, \vec{a}_m) = (f(\vec{a}_1), \ldots, f(\vec{a}_m))$. It is sometimes convenient to think of the sequence $\vec{a}_1, \ldots, \vec{a}_m \in A^n$ as an $m \times n$ matrix M with $M_{ij} = (\vec{a}_i)_j$ and we may write $f(M)$ for $f(\vec{a}_1, \ldots, a_m)$. On the other hand if N is a $n \times m$ matrix with entries in A, we write $\hat{f}(N)$ to denote $f(N^T)$. That is, for $\vec{a}_1, \ldots, \vec{a}_n \in A^m$ $\hat{f}(\vec{a}_1, \ldots, \vec{a}_n)$ denotes $(f(\vec{b}_1), \ldots, f(\vec{b}_m))$, where $\vec{b}_i = (N^T)_i$ is the tuple of ith components of $\vec{a}_1, \ldots, \vec{a}_n$. For a matrix M, we write M_i to denote the vector formed by the ith row of M.

For a pair of structures $\mathfrak{A}$ and $\mathfrak{B}$, a *partial isomorphism* from $\mathfrak{A}$ to $\mathfrak{B}$ is a partial function $f : A \to B$ which is an isomorphism between the substructure of $\mathfrak{A}$ induced by the domain of f and the substructure of $\mathfrak{B}$ induced by the image of f. We write $\mathrm{PI}(\mathfrak{A}, \mathfrak{B})$ to denote the collection of all partial isomorphisms from $\mathfrak{A}$ to $\mathfrak{B}$.

We write $\mathbb{N}$ or ω to denote the natural numbers, and $\mathbb{Z}$ to denote the ring of integers. For any $n \in \mathbb{N}$, we write $[n]$ to denote the set $\{1, \ldots, n\}$. When mentioned without further qualification, a graph $G = (V, E)$ is simple and undirected. That is, it is a structure with universe V and one binary relation E that is irreflexive and symmetric. The *girth* of a graph G is the length of the shortest cycle in G.

A *hypergraph* is a pair (H, E) such that E is a set of subsets of H. (H, E) is *n-uniform* if $|e| = n$ for all $e \in E$. As usual, we treat an n-uniform hypergraph (H, E) as the corresponding relational structure (H, R), where $R := \{(v_1, \ldots, v_n) \in H^n \mid \{v_1, \ldots, v_n\} \in E\}$.

2.1 Generalized quantifiers

Let σ, τ be relational vocabularies with $\tau = \{R_1, \ldots, R_m\}$, and $\mathrm{ar}(R_i) = r_i$ for each $i \in [m]$. An interpretation $\mathcal{I}$ of τ in σ with parameters $\vec{z}$ is a tuple of σ-formulas $(\psi_1, \ldots, \psi_m)$ along with tuples $\vec{y}_1, \ldots, \vec{y}_m$ of variables with $|\vec{y}_i| = r_i$ for $i \in [m]$, such that the free variables of ψ_i are among $\vec{y}_i\vec{z}$. Such an interpretation defines a mapping that takes a σ-structure $\mathfrak{A}$, along with an interpretation α of the parameters $\vec{z}$ in $\mathfrak{A}$ to a τ-structure $\mathfrak{B} := \mathcal{I}(\mathfrak{A}, \alpha)$ as follows. The universe of $\mathfrak{B}$ is A, and the relations $R_i \in \tau$ are interpreted in $\mathfrak{B}$ by $R_i^{\mathfrak{B}} = \{\vec{b} \in A^{r_i} \mid (\mathfrak{A}, \alpha[\vec{b}/\vec{y}_i]) \models \psi_i\}$.

Let L be a logic and $\mathcal{K}$ a class of τ-structures. The extension $L(Q_\mathcal{K})$ of L by the *generalized quantifier* for the class $\mathcal{K}$ is obtained by extending the syntax of L by the following formula formation rule:

> For $\mathcal{I} = (\psi_1, \ldots, \psi_m)$ an interpretation of τ in σ with parameters $\vec{z}$, $\psi(\vec{z}) = Q_\mathcal{K}\vec{y}_1, \ldots, \vec{y}_m\mathcal{I}$ is a formula over the signature σ, with free variables $\vec{z}$. The semantics of the formula is given by $(\mathfrak{A}, \alpha) \models \psi(\vec{z})$, if, and only if, $\mathcal{I}(\mathfrak{A}, \alpha) \in \mathcal{K}$.

The extension $L(\mathbf{Q})$ of L by a collection $\mathbf{Q}$ of generalized quantifiers is defined by adding the rules above to L for each $Q_\mathcal{K} \in \mathbf{Q}$ separately.

The *type* of the quantifier $Q_\mathcal{K}$ is $(r_1, \ldots, r_m)$, and the *arity* of $Q_\mathcal{K}$ is $\max\{r_1, \ldots, r_m\}$. For the sake of simplicity, we assume in the sequel that the type of $Q_\mathcal{K}$ is *uniform*, i.e., $r_i = r_j$ for all $i, j \in [m]$. This is no loss of generality, since any quantifier $Q_\mathcal{K}$ is definably equivalent with another quantifier $Q_{\mathcal{K}'}$ of uniform type with the same arity. Furthermore, we restrict the syntactic rule of $Q_\mathcal{K}$ by requiring that $\vec{y}_i = \vec{y}_j$ for all $i, j \in [m]$. Then we can denote the formula obtained by applying the rule simply by $\varphi = Q_\mathcal{K}\vec{y}(\psi_1, \ldots, \psi_m)$.

Let $Q = Q_\mathcal{K}$ and $Q' = Q_{\mathcal{K}'}$ be generalized quantifiers. We say that Q is *definable* in $L(Q')$ if the defining class $\mathcal{K}$ is definable in $L(Q')$, i.e., there is a sentence φ of $L(Q')$ such that $\mathcal{K} = \{\mathfrak{A} \mid \mathfrak{A} \models \varphi\}$.

We write $\mathbf{Q}_n$ to denote the collection of all quantifiers of arity at most n. Hella [17] shows that for any n, there is a quantifier of arity $n+1$ that is not definable in $L^\omega_{\infty\omega}(\mathbf{Q}_n)$. The logic $L^\omega_{\infty\omega}(\mathbf{Q}_1)$ is equivalent to $C^\omega_{\infty\omega}$, the infinitary logic with counting. The notion of interpretation we have defined is fairly restricted in that it does not allow for *relativization* or *vectorizations* (see, e.g. [11, Def. 12.3.6]. The relativizations and vectorizations of a quantifer Q can always be seen as a *collection* of simple quantifiers of unbounded arity.

2.2 CSP and polymorphisms

Given relational structures $\mathfrak{A}$ and $\mathfrak{B}$ over the same vocabulary τ, a *homomorphism* $h : \mathfrak{A} \to \mathfrak{B}$ is a function that takes elements of A to elements of B and such that for every $R \in \tau$ of arity r and any $\vec{a} \in A^r$, $\vec{a} \in R^{\mathfrak{A}}$ implies $h(\vec{a}) \in R^{\mathfrak{B}}$. For a fixed structure $\mathfrak{B}$, we write $\mathsf{CSP}(\mathfrak{B})$ to denote the collection of structures $\mathfrak{A}$ for which there is some homomorphism $h : \mathfrak{A} \to \mathfrak{B}$. By the celebrated theorem of Bulatov and Zhuk, every class $\mathsf{CSP}(\mathfrak{B})$ is either decidable in polynomial time or NP-complete.

Given a τ-structure $\mathfrak{B}$ and $m \in \mathbb{N}$, we define a τ-structure $\mathfrak{B}^m$. Its universe is B^m and if R in τ is a relation of arity r, and $\vec{a}_1, \ldots, \vec{a}_r \in B^m$, then $(\vec{a}_1, \ldots, \vec{a}_r) \in R^{\mathfrak{B}^m}$ if, and only if, $(M^T)_j \in R^{\mathfrak{B}}$ for all $j \in [m]$ where M is the $r \times m$ matrix formed by $(\vec{a}_1, \ldots, \vec{a}_r)$. Then, a *polymorphism* of $\mathfrak{B}$ is a homomorphism $p : \mathfrak{B}^m \to \mathfrak{B}$ for some m. The collection of polymorphisms of $\mathfrak{B}$ forms an algebraic *clone* with universe B. It is known that the equational theory of this algebra completely determines the computational complexity of $\mathsf{CSP}(\mathfrak{B})$ (see [3] for an expository account).

A function $m : B^3 \to B$ is a *majority* function if it satisfies the equations $m(a, a, b) = m(a, b, a) = m(b, a, a) = a$ for all $a, b \in B$. More generally, for $\ell \geq 3$, a function $n : B^\ell \to B$ is a *near-unanimity* function of arity ℓ if for any ℓ-tuple $\vec{a}$, we have $n(\vec{a}) = a$ whenever at least $\ell - 1$ components of $\vec{a}$ are a. In particular, a near-unanimity function of arity 3 is a majority function. A function $M : B^3 \to B$ is a *Maltsev* function if it satisfies the identities $M(a, b, b) = M(b, b, a) = a$ for all $a, b \in B$.

For any structure $\mathfrak{B}$ which has a near-unanimity polymorphism, the class $\mathsf{CSP}(\mathfrak{B})$ is decidable in polynomial time, and definable in $L^\omega_{\infty\omega}$. If $\mathfrak{B}$ admits a Maltsev polymorphism, then $\mathsf{CSP}(\mathfrak{B})$ is also decidable in polynomial time, but may not be definable in $L^\omega_{\infty\omega}$ or $L^\omega_{\infty\omega}(\mathbf{Q}_1)$, its extension with all unary quantifiers. The classic example of a CSP with a Maltsev polymorphism that is not definable in $L^\omega_{\infty\omega}(\mathbf{Q}_1)$ is solvability of systems of equations over $\mathbb{Z}/2\mathbb{Z}$ with ℓ variables per equation. We can treat this as the class of structures $\mathsf{CSP}(\mathfrak{C}_\ell)$ where $\mathfrak{C}_\ell$ is the structure with universe $\{0, 1\}$ and two ℓ-ary relations $R_0 = \{(b_1, \ldots, b_\ell) \mid \sum_i b_i \equiv 0 \pmod 2\}$ and $R_1 = \{(b_1, \ldots, b_\ell) \mid \sum_i b_i \equiv 1 \pmod 2\}$.

If $\mathcal{K} = \mathsf{CSP}(\mathfrak{B})$ for some fixed structure $\mathfrak{B}$, we call $Q_\mathcal{K}$ a *CSP quantifier*. Write $\mathbf{CSP}_n$ for the collection of all CSP quantifiers $Q_\mathcal{K}$ where $\mathcal{K} = \mathsf{CSP}(\mathfrak{B})$ for a structure with at most n elements. Note that there is no restriction on the number or arity of relations in the signature of $\mathfrak{B}$ and thus $\mathbf{CSP}_n$ contains quantifiers of all arities. Hella [18] defines a pebble game that characterizes equivalence of structures in the logic $L^\omega_{\infty\omega}(\mathbf{Q}_1, \mathbf{CSP}_n)$ and shows that there is a structure $\mathfrak{B}$ on $n+1$ elements such that $\mathsf{CSP}(\mathfrak{B})$ is not definable in this logic.

3 Partial polymorphisms

Let τ be a relational vocabulary, and let $\mathfrak{C}$ be a τ-structure with a polymorphism p: $\mathfrak{C}^n \to \mathfrak{C}$. This gives rise to a closure condition on the class $\mathsf{CSP}(\mathfrak{C})$. In particular, suppose $\mathfrak{B} \in \mathsf{CSP}(\mathfrak{C})$ by a homomorphism $h : \mathfrak{B} \to \mathfrak{C}$. We can, in a sense, "close" $\mathfrak{B}$ under the polymorphism p

by including in each relation $R^{\mathfrak{B}}$ ($R \in \tau$) any tuple $\vec{a}$ for which $h(\vec{a}) = p(h(\vec{a}_1, \dots, \vec{a}_n))$ for some $\vec{a}_1, \dots, \vec{a}_n \in R_i^{\mathfrak{B}}$. The resulting structure $\mathfrak{B}'$ is still in $\mathsf{CSP}(\mathfrak{C})$ as is any structure $\mathfrak{A}$ with the same universe as $\mathfrak{B}$ and for which $R^{\mathfrak{A}} \subseteq R^{\mathfrak{B}'}$ for all $R \in \tau$.

Our aim is to generalize this type of closure property from CSP quantifiers to a larger class of generalized quantifiers. To formally define this, it is useful to introduce some notation. For reasons that will become clear, we use *partial* functions p.

▶ **Definition 1.** *Let $A \neq \emptyset$ be a set, and let p be a partial function $A^n \to A$.*
(a) *If $R \subseteq A^r$, then $p(R) := \{\hat{p}(\vec{a}_1, \dots, \vec{a}_n) \mid \vec{a}_1, \dots, \vec{a}_n \in R\}$.*
(b) *If $\mathfrak{A} = (A, R_1^{\mathfrak{A}}, \dots, R_m^{\mathfrak{A}})$, then we denote the structure $(A, p(R_1^{\mathfrak{A}}), \dots, p(R_m^{\mathfrak{A}}))$ by $p(\mathfrak{A})$.*

We say that p is a *partial polymorphism* of a τ-structure $\mathfrak{A}$ with domain A if for every $R \in \tau$, the relation $R^{\mathfrak{A}}$ is closed with respect to p, i.e., $p(R^{\mathfrak{A}}) \subseteq R^{\mathfrak{A}}$.

The reason for considering partial functions is that we are usually interested in polymorphisms that satisfy certain equations. The equations specify the polymorphism partially, but not totally. In other words, any polymorphism that extends the given partial function is a polymorphism satisfying the required equations. Thus, we can uniformly specify closure properties on our class of structures for polymorphisms satisfying the equations by only requiring closure for the partial function. This is illustrated in the examples below.

By a *family of partial functions* we mean a class $\mathcal{P}$ that contains a partial function $p_A \colon A^n \to A$ for every finite set A, where n is a fixed positive integer. We give next some important examples of families of partial functions that arise naturally from well-known classes of polymorphisms.

▶ **Example 2.**
(a) The *Maltsev family* $\mathcal{M}$ consists of the partial functions $M_A \colon A^3 \to A$ such that $M_A(a, b, b) = M_A(b, b, a) = a$ for all $a, b \in A$, and $M_A(a, b, c)$ is undefined unless $a = b$ or $b = c$. If a structure $\mathfrak{A}$ has a Maltsev polymorphism $p \colon A^3 \to A$, then clearly M_A is a restriction of p, whence it is a partial polymorphism of $\mathfrak{A}$.
(b) The family $\mathcal{MJ}$ of ternary *partial majority functions* consists of the partial functions $m_A \colon A^3 \to A$ such that $m_A(a, a, b) = m_A(a, b, a) = m_A(b, a, a) = a$ for all $a, b \in A$, and $m_A(a, b, c)$ is undefined if a, b and c are all distinct. If $\mathfrak{A}$ has a majority polymorphism, then m_A is a restriction of it, whence it is a partial polymorphism of $\mathfrak{A}$.
(c) More generally, for each $\ell \geq 3$ we define the family $\mathcal{N}_\ell$ of *ℓ-ary partial near-unanimity functions* $n_A^\ell \colon A^\ell \to A$ as follows:
- $n_A^\ell(a_1, \dots, a_\ell) = a$ if and only if $|\{i \in [n] \mid a_i = a\}| \geq \ell - 1$.

In particular, $\mathcal{MJ} = \mathcal{N}_3$.

We next give a formal definition for the closure property of generalized quantifiers that arises from a family of partial functions. In the definition we use the notation $\mathfrak{A} \leq \mathfrak{B}$ if $\mathfrak{A}$ and $\mathfrak{B}$ are τ-structures such that $A = B$ and $R^{\mathfrak{A}} \subseteq R^{\mathfrak{B}}$ for each $R \in \tau$. Furthermore, we define the union $\mathfrak{A} \cup \mathfrak{B}$ of $\mathfrak{A}$ and $\mathfrak{B}$ to be the τ-structure $\mathfrak{C}$ such that $C = A \cup B$ and $R^{\mathfrak{C}} = R^{\mathfrak{A}} \cup R^{\mathfrak{B}}$ for each $R \in \tau$. Note that we do not assume here that A and B are disjoint. On the contrary, we use the union $\mathfrak{A} \cup \mathfrak{B}$ specifically for structures $\mathfrak{A}$ and $\mathfrak{B}$ that share a common universe $A = B$.

▶ **Definition 3.** *Let $\mathcal{P}$ be a family of n-ary partial functions, and let $Q_{\mathcal{K}}$ be a generalized quantifier of vocabulary τ. We say that $Q_{\mathcal{K}}$ is $\mathcal{P}$-closed if the following holds for all τ-structures $\mathfrak{A}$ and $\mathfrak{B}$ with $A = B$:*
- *if $\mathfrak{B} \in \mathcal{K}$ and $\mathfrak{A} \leq p_A(\mathfrak{B}) \cup \mathfrak{B}$ for some $p_A \in \mathcal{P}$, then $\mathfrak{A} \in \mathcal{K}$.*

We denote the class of all $\mathcal{P}$-closed quantifiers by $\mathbf{Q}_{\mathcal{P}}$.

Note that the condition $\mathfrak{A} \leq p_A(\mathfrak{B}) \cup \mathfrak{B}$ holds if and only if for every $R \in \tau$ and every $\vec{a} \in R^{\mathfrak{A}} \setminus R^{\mathfrak{B}}$ there are tuples $\vec{a}_1, \ldots, \vec{a}_n \in R^{\mathfrak{B}}$ such that $\vec{a} = \widehat{p_A}(\vec{a}_1, \ldots, \vec{a}_n)$.

The quantifier $Q_{\mathcal{K}}$ is *downwards monotone* if $\mathfrak{A} \leq \mathfrak{B}$ and $\mathfrak{B} \in \mathcal{K}$ implies $\mathfrak{A} \in \mathcal{K}$. It follows directly from Definition 3 that all $\mathcal{P}$-closed quantifiers are downwards monotone.

▶ **Proposition 4.** *If $Q_{\mathcal{K}} \in \mathbf{Q}_{\mathcal{P}}$ for some family $\mathcal{P}$, then $Q_{\mathcal{K}}$ is downwards monotone.*

▶ Remark 5. As far as we know, the notion of $\mathcal{P}$-closed quantifiers (or classes) has not been considered earlier. In particular, as we mentioned in Section 1, a quantifier $Q_{\mathcal{K}}$ being $\mathcal{P}$-closed is not based on the Galois connection between partial polymorphisms and relations: by downwards monotonicity of $Q_{\mathcal{K}}$, the class $\mathcal{K}$ usually contains structures $\mathfrak{A}$ such that p_A is not a partial polymorphism of $\mathfrak{A}$. Note also that the structure $\mathfrak{D}$ with full relations (i.e., $R^{\mathfrak{D}} = D^r$ for each $R \in \tau$ of arity r) is usually not in $\mathcal{K}$ although p_D is a partial polymorphism of $\mathfrak{D}$.

We show next that there are quantifiers that are $\mathcal{P}$-closed for all families $\mathcal{P}$ of partial functions.

▶ **Proposition 6.** *Let $\mathcal{K}_0$ be the class of all $\{R\}$-structures $\mathfrak{A}$ such that $R^{\mathfrak{A}} = \emptyset$. Then $Q_{\mathcal{K}_0} \in \mathbf{Q}_{\mathcal{P}}$ for any family $\mathcal{P}$ of partial functions.*

Proof. If $\mathfrak{B} \in \mathcal{K}_0$, then $R^{\mathfrak{B}} = \emptyset$, whence $p_B(\mathfrak{B}) = \emptyset$. Thus, if $\mathfrak{A} \leq p_B(\mathfrak{B}) \cup \mathfrak{B}$, then $R^{\mathfrak{A}} = \emptyset$, and hence $\mathfrak{A} \in \mathcal{K}_0$. ◀

Note that in the case $\mathrm{ar}(R) = 1$, the quantifier $Q_{\mathcal{K}_0}$ of the proposition above is the negation of the existential quantifier: $\mathfrak{A} \models Q_{\mathcal{K}_0} x\, \varphi \iff \mathfrak{A} \models \neg\exists x\, \varphi$. Thus, for any family $\mathcal{P}$, the first-order quantifiers can be defined from a $\mathcal{P}$-closed quantifier using only negation.

Up to now we have not imposed any restrictions on the family $\mathcal{P}$. It is natural to require that the partial functions in $\mathcal{P}$ are uniformly defined, or at least that (A, p_A) and (B, p_B) are isomorphic if $|A| = |B|$. Such requirements are captured by the notions defined below.

▶ **Definition 7.** *Let $\mathcal{P}$ be a family of n-ary partial functions.*

(a) *$\mathcal{P}$ is* invariant *if it respects bijections: if $f\colon A \to B$ is a bijection and $a_1, \ldots, a_n \in A$, then $p_B(f(a_1), \ldots, f(a_n)) \simeq f(p_A(a_1, \ldots, a_n))$. Here the symbol $\simeq$ says that either both sides are defined and have the same value, or both sides are undefined.*

(b) *$\mathcal{P}$ is* strongly invariant *if it respects injections: if $f\colon A \to B$ is an injection and $a_1, \ldots, a_n \in A$, then $p_B(f(a_1), \ldots, f(a_n)) \simeq f(p_A(a_1, \ldots, a_n))$.*

(c) *$\mathcal{P}$ is* projective*, if it is strongly invariant and it is preserved by all functions: if $f\colon A \to B$ is a function and $a_1, \ldots, a_n \in A$ are such that $p_A(a_1, \ldots, a_n)$ is defined, then $p_B(f(a_1), \ldots, f(a_n)) = f(p_A(a_1, \ldots, a_n))$.*

It is easy to verify that $\mathcal{P}$ is invariant if, and only if, it is determined by equality types on each cardinality: there are quantifier free formulas in the language of equality $\theta^m_{\mathcal{P}}(\vec{x}, y)$ such that if $|A| = m$, then $p_A(\vec{a}) = b \iff A \models \theta^m_{\mathcal{P}}[\vec{a}/\vec{x}, b/y]$ holds for all $\vec{a} \in A^n$ and $b \in A$. Similarly, $\mathcal{P}$ is strongly invariant if, and only if, the same holds with a single formula $\theta_{\mathcal{P}} = \theta^m_{\mathcal{P}}$ for all $m \in \omega$.

Note that if the family $\mathcal{P}$ is strongly invariant, then for every finite set A, p_A is a *partial choice function*, i.e., $p_A(a_1, \ldots, a_n) \in \{a_1, \ldots, a_n\}$. Indeed, if $b := p_A(a_1, \ldots, a_n) \notin \{a_1, \ldots, a_n\}$ and $B = A \cup \{c\}$, where $c \notin A$, then using the identity function $f = \mathrm{id}_A$ of A in the condition $p_B(f(a_1), \ldots, f(a_n)) = f(p_A(a_1, \ldots, a_n))$, we get $p_B(a_1, \ldots, a_n) = b$. On the other hand, using the injection $f'\colon A \to B$ that agrees with id_A on $A \setminus \{b\}$ but maps b to c, we get the contradiction $p_B(a_1, \ldots, a_n) = c \neq b$.

▶ **Lemma 8.** *Let $\mathcal{P}$ be a family of n-ary partial choice functions. Then $Q_{\mathcal{K}} \in \mathbf{Q}_{\mathcal{P}}$ for any unary downwards monotone quantifier $Q_{\mathcal{K}}$. In particular this holds if $\mathcal{P}$ is strongly invariant.*

Proof. Let τ be the vocabulary of $\mathcal{K}$, and assume that $\mathfrak{B} \in \mathcal{K}$ and $\mathfrak{A} \leq p_A(\mathfrak{B}) \cup \mathfrak{B}$. Then for all $R \in \tau$ and $a \in R^{\mathfrak{A}} \setminus R^{\mathfrak{B}}$ there are $a_1, \dots, a_n \in A$ such that $p_A(a_1, \dots, a_n) = a$ and $a_i \in R^{\mathfrak{B}}$ for each $i \in [n]$. Since p_A is a partial choice function, we have $a \in \{a_1, \dots, a_n\}$, and hence $a \in R^{\mathfrak{B}}$. Thus we see that $\mathfrak{A} \leq \mathfrak{B}$, and consequently $\mathfrak{A} \in \mathcal{K}$, since $Q_{\mathcal{K}}$ is downwards monotone. ◀

It is easy to see that the families $\mathcal{M}$ and $\mathcal{N}_\ell$, $\ell \geq 3$, introduced in Example 2, are strongly invariant. Indeed, the defining formulas $\theta_{\mathcal{M}}$ and $\theta_{\mathcal{N}_\ell}$ are easily obtained from the identities that define these conditions. Thus, all unary downwards monotone quantifiers are $\mathcal{M}$-closed and $\mathcal{N}_\ell$-closed. For the families $\mathcal{N}_\ell$ we can prove a much stronger result:

▶ **Lemma 9.** *Let $\ell \geq 3$, and let $Q_{\mathcal{K}}$ be a downwards monotone quantifier of arity $r < \ell$. Then $Q_{\mathcal{K}} \in \mathbf{Q}_{\mathcal{N}_\ell}$.*

Proof. Let τ be the vocabulary of $\mathcal{K}$, and assume that $\mathfrak{B} \in \mathcal{K}$ and $\mathfrak{A} \leq n^\ell_A(\mathfrak{B}) \cup \mathfrak{B}$. Then for all $R \in \tau$ and $\vec{a} = (a_1, \dots, a_r) \in R^{\mathfrak{A}} \setminus R^{\mathfrak{B}}$ there are $\vec{a}_i = (a^1_i, \dots, a^r_i) \in R^{\mathfrak{B}}$, $i \in [\ell]$, such that $\widehat{n^\ell_A}(\vec{a}_1, \dots, \vec{a}_\ell) = \vec{a}$. Thus, for each $j \in [r]$ there is at most one $i \in [\ell]$ such that $a^j_i \neq a_j$, and hence there is at least one $i \in [\ell]$ such that $\vec{a} = \vec{a}_i$. This shows that $\mathfrak{A} \leq \mathfrak{B}$, and since $Q_{\mathcal{K}}$ is downwards monotone, we conclude that $\mathfrak{A} \in \mathcal{K}$. ◀

Using a technique originally due to Imhof for (upwards) monotone quantifiers (see [19]), we can show that any quantifier $Q_{\mathcal{K}}$ is definable by a downwards monotone quantifier of the same arity. Indeed, if the vocabulary of $\mathcal{K}$ is $\tau = \{R_1, \dots, R_m\}$, where $\mathrm{ar}(R_i) = r$ for all $i \in [m]$, we let $\tau' := \{S_1, \dots, S_m\}$ be a disjoint copy of τ, and $\tau^* := \tau \cup \tau'$. Furthermore, we let $\mathcal{K}^*$ be the class of all τ^*-structures $\mathfrak{A}$ such that $R^{\mathfrak{A}}_i \cap S^{\mathfrak{A}}_i = \emptyset$ for all $i \in [m]$, and $(A, R^{\mathfrak{A}}_1, \dots, R^{\mathfrak{A}}_m) \in \mathcal{K}$ or $R^{\mathfrak{A}}_i \cup S^{\mathfrak{A}}_i \neq A^r$ for some $i \in [m]$. Then $Q_{\mathcal{K}^*}$ is downwards monotone, and clearly $Q_{\mathcal{K}} \vec{x}\,(\psi_1, \dots, \psi_m)$ is equivalent with $Q_{\mathcal{K}^*} \vec{x}\,(\psi_1, \dots, \psi_m, \neg\psi_1, \dots, \neg\psi_m)$.

Using this observation, we get the following corollary to Lemmas 8 and 9.

▶ **Corollary 10.**
(a) *Let $\mathcal{P}$ be as in Lemma 8. Then $L^k_{\infty\omega}(\mathbf{Q}_{\mathcal{P}} \cup \mathbf{Q}_1) \leq L^k_{\infty\omega}(\mathbf{Q}_{\mathcal{P}})$.*
(b) *$L^k_{\infty\omega}(\mathbf{Q}_{\mathcal{N}_\ell} \cup \mathbf{Q}_{\ell-1}) \leq L^k_{\infty\omega}(\mathbf{Q}_{\mathcal{N}_\ell})$.*

As explained in the beginning of this section, the definition of $\mathcal{P}$-closed quantifiers was inspired by the closure property of a CSP quantifier $Q_{\mathsf{CSP}(\mathfrak{C})}$ that arises from a polymorphism of $\mathfrak{C}$. Thus, it is natural to look for sufficient conditions on the family $\mathcal{P}$ and the target structure $\mathfrak{C}$ for $Q_{\mathsf{CSP}(\mathfrak{C})}$ to be $\mathcal{P}$-closed. It turns out that the notions of projectivity and partial polymorphism lead to such a condition.

▶ **Proposition 11.** *Let $\mathcal{P}$ be a projective family of n-ary partial functions, and let $\mathfrak{C}$ be a τ-structure. If p_C is a partial polymorphism of $\mathfrak{C}$, then $Q_{\mathsf{CSP}(\mathfrak{C})} \in \mathbf{Q}_{\mathcal{P}}$.*

Proof. Assume that $\mathfrak{B} \in \mathsf{CSP}(\mathfrak{C})$ and $\mathfrak{A} \leq p_A(\mathfrak{B}) \cup \mathfrak{B}$. Then $A = B$ and there is a homomorphism $h\colon \mathfrak{B} \to \mathfrak{C}$. We show that h is a homomorphism $\mathfrak{A} \to \mathfrak{C}$, and hence $\mathfrak{A} \in \mathsf{CSP}(\mathfrak{C})$. Thus let $R \in \tau$, and let $\vec{a} \in R^{\mathfrak{A}}$. If $\vec{a} \in R^{\mathfrak{B}}$, then $h(\vec{a}) \in R^{\mathfrak{C}}$ by assumption. On the other hand, if $\vec{a} \in R^{\mathfrak{A}} \setminus R^{\mathfrak{B}}$, then there exist tuples $\vec{a}_1, \dots, \vec{a}_n \in R^{\mathfrak{B}}$ such that $\vec{a} = \widehat{p_A}(\vec{a}_1, \dots, \vec{a}_n)$. Since h is a homomorphism $\mathfrak{B} \to \mathfrak{C}$, we have $h(\vec{a}_i) \in R^{\mathfrak{C}}$ for each $i \in [n]$. Since p_C is a partial polymorphism of $\mathfrak{C}$, we have $\widehat{p_C}(h(\vec{a}_1), \dots, h(\vec{a}_n)) \in R^{\mathfrak{C}}$. Finally, since $\mathcal{P}$ is projective, we have $h(\vec{a}) = h(\widehat{p_A}(\vec{a}_1, \dots, \vec{a}_n)) = \widehat{p_C}(h(\vec{a}_1), \dots, h(\vec{a}_n))$, and hence $h(\vec{a}) \in R^{\mathfrak{C}}$. ◀

We can now apply Proposition 11 to the families introduced in Example 2.

▶ **Example 12.**

(a) Consider a constraint satisfaction problem $\mathsf{CSP}(\mathfrak{C})$ such that $\mathfrak{C}$ has a Maltsev polymorphism $p\colon \mathfrak{C}^3 \to \mathfrak{C}$. We show that $Q_{\mathsf{CSP}(\mathfrak{C})} \in \mathbf{Q}_{\mathcal{M}}$. As pointed out in Example 2, M_C is a partial polymorphism of $\mathfrak{C}$. Thus, by Proposition 11 it suffices to show that the Maltsev family $\mathcal{M}$ is projective.

Thus, assume that $f\colon A \to B$ is a function, and $M_A(a,b,c)$ is defined. Then $a = b$ and $M_A(a,b,c) = c$, or $b = c$ and $M_A(a,b,c) = a$. In the former case we have $f(a) = f(b)$, whence $M_B(f(a), f(b), f(c)) = f(c) = f(M_A(a,b,c))$. In the latter case we have $f(b) = f(c)$, whence $M_B(f(a), f(b), f(c)) = f(a) = f(M_A(a,b,c))$.

(b) The *n-uniform hypergraph m-colouring* problem is $\mathsf{CSP}(\mathfrak{H}_{n,m})$, where $\mathfrak{H}_{n,m} = ([m], R_{n,m})$ is the complete n-uniform hypergraph with m vertices, i.e.,

- $R_{n,m} := \{(v_1, \dots, v_n) \in [m]^n \mid v_i \neq v_j \text{ for all } 1 \leq i < j \leq m\}$.

We show that $Q_{\mathsf{CSP}(\mathfrak{H}_{n,m})} \in \mathbf{Q}_{\mathcal{MJ}}$ for all $n \geq 2$ and $m \geq n$. By Proposition 11 it suffices to show that $m_{[m]}$ is a partial polymorphism of $\mathfrak{H}_{n,m}$, and the family $\mathcal{MJ}$ is projective.

To see that $m_{[m]}$ is a partial polymorphism of $\mathfrak{H}_{n,m}$, assume that $\vec{a}_i = (a_i^1, \dots, a_i^n) \in R_{n,m}$ for $i \in [3]$, and $\vec{a} = (a_1, \dots, a_n) = \hat{m}_{[m]}(\vec{a}_1, \vec{a}_2, \vec{a}_3)$. By the definition of $m_{[m]}$, for each $j \in [n]$ we have $|\{i \in [3] \mid a_i^j = a_j\}| \geq 2$. Thus for any two distinct $j, k \in [n]$, there is $i \in [3]$ such that $a_j = a_i^j$ and $a_i^k = a_k$, whence $a_j \neq a_k$. Thus we have $\vec{a} \in R_{n,m}$.

To show that $\mathcal{MJ}$ is projective, assume that $f\colon A \to B$ is a function, and $m_A(a,b,c)$ is defined. Then $a = b = m_A(a,b,c)$, $a = c = m_A(a,b,c)$ or $b = c = m_A(a,b,c)$. In the first case we have $f(m_A(a,b,c)) = f(a) = f(b) = m_B(f(a), f(b), f(c))$, as desired. The two other cases are similar.

(c) In the same way we can show that the family $\mathcal{N}_\ell$ of partial near-unanimity polymorphisms is projective for any $\ell \geq 3$. We relax now the notion of hypergraph coloring as follows: Let $\mathfrak{H} = (H, R)$ be an n-uniform hypergraph, and let $k < n$. A *k-weak m-coloring* of $\mathfrak{H}$ is a function $f\colon H \to [m]$ such that for all $(u_1, \dots, u_n) \in R$ and all $i \in [m]$, $|\{u_1, \dots, u_n\} \cap f^{-1}[\{i\}]| \leq k$. Thus, instead of requiring that all vertices in a hyperedge $(u_1, \dots, u_n)$ must have different colors, a k-weak m-coloring allows up to k of them to have the same color. (Note that there are no restrictions on how many at most k-element subsets can be colored with a single color.) Observe now that there exists a k-weak m-coloring of $\mathfrak{H}$ if and only if $\mathfrak{H} \in \mathsf{CSP}(\mathfrak{H}^k_{n,m})$, where $\mathfrak{H}^k_{n,m} = ([m], R^k_{n,m})$ is the structure such that

- $R^k_{n,m} := \{(v_1, \dots, v_n) \in [m]^n \mid |\{v_i \mid i \in I\}| \geq 2 \text{ for all } I \subseteq [n] \text{ with } |I| = k+1\}$.

Note that $\mathfrak{H}^1_{n,m} = \mathfrak{H}_{n,m}$, whence $m_{[m]} = n^3_{[m]}$ is a partial polymorphism of $\mathfrak{H}^1_{n,m}$. It is straightforward to generalize this to $\ell > 3$: $n^\ell_{[m]}$ is a partial polymorhism of $\mathfrak{H}^{\ell-2}_{n,m}$. Thus by Proposition 11, the CSP quantifier $Q_{\mathsf{CSP}(\mathfrak{H}^{\ell-2}_{n,m})}$ is $\mathbf{Q}_{\mathcal{N}_\ell}$-closed.

▶ Remark 13. As shown in Example 12(b), the partial majority function $m_{[m]}$ is a partial polymorphism of the structure $\mathfrak{H}_{n,m}$. However, there does not exist any polymorphism $p\colon [m]^3 \to [m]$ that extends $m_{[m]}$. This can be verified directly, but it also follows from the fact that $\mathsf{CSP}(\mathfrak{C})$ is of bounded width for any $\mathfrak{C}$ that has a majority polymorphism ([13]), but $\mathsf{CSP}(\mathfrak{H}_{n,m})$ is not of bounded width. The same holds for the partial functions $n^\ell_{[m]}$ and the structures $\mathfrak{H}^k_{n,m}$ in Example 12(c).

4 Pebble game for $\mathcal{P}$-closed quantifiers

In this section we introduce a pebble game that characterizes equivalence of structures with respect to $L^{\omega}_{\infty\omega}(\mathbf{Q}_{\mathcal{P}})$, the extension of the infinitary k-variable logic $L^{\omega}_{\infty\omega}$ by the class of all $\mathcal{P}$-closed quantifiers.

We fix a family $\mathcal{P}$ of n-ary partial functions for the rest of the section. Given two structures $\mathfrak{A}$ and $\mathfrak{B}$ of the same vocabulary, and assignments α and β on $\mathfrak{A}$ and $\mathfrak{B}$, respectively, such that $\mathrm{dom}(\alpha) = \mathrm{dom}(\beta)$, we write $(\mathfrak{A}, \alpha) \equiv^{k}_{\infty\omega,\mathcal{P}} (\mathfrak{B}, \beta)$ if the equivalence

$$(\mathfrak{A}, \alpha) \models \varphi \iff (\mathfrak{B}, \beta) \models \varphi$$

holds for all formulas $\varphi \in L^{k}_{\infty\omega}(\mathbf{Q}_{\mathcal{P}})$ with free variables in $\mathrm{dom}(\alpha)$. If $\alpha = \beta = \emptyset$, we write simply $\mathfrak{A} \equiv^{k}_{\infty\omega,\mathcal{P}} \mathfrak{B}$ instead of $(\mathfrak{A}, \emptyset) \equiv^{k}_{\infty\omega,\mathcal{P}} (\mathfrak{B}, \emptyset)$.

The basic idea of our pebble game for a pair $(\mathfrak{A}, \mathfrak{B})$ of structures is the following. In each round Duplicator gives a bijection $f\colon A \to B$, just like in the bijection games of [17], but instead of using $\vec{b} = f(\vec{a})$ as answer for Spoiler's move $\vec{a} \in A^r$, she is allowed to give a sequence $\vec{b}_1, \ldots, \vec{b}_n \in B^r$ of alternative answers as long as $\vec{b} = \widehat{p_B}(\vec{b}_1, \ldots, \vec{b}_n)$. In particular, $\vec{b}$ need not be among the list $\vec{b}_1, \ldots, \vec{b}_n$. Spoiler completes the round by choosing one of these alternatives $\vec{b}_i$. Spoiler wins if $\vec{a} \mapsto \vec{b}_i$ is not a partial isomorphism; otherwise the game carries on from the new position. Note that this allows more freedom to Duplicator than in the ordinary k, n-bijection game. She can simulate a winning strategy in that game by simply playing at each move the sequence $\vec{b}_1, \ldots, \vec{b}_n$ where each $\vec{b}_i = \vec{b} = f(\vec{a})$.

Observe now that if Duplicator has a winning strategy for the first round of the game, then $f(\mathfrak{A}) \leq p_B(\mathfrak{B}) \cup \mathfrak{B}$. Indeed, if Spoiler chooses a tuple $\vec{a} \in R^{\mathfrak{A}}$, then Duplicator has to answer by either the tuple $f(\vec{a})$, or a sequence $\vec{b}_1, \ldots, \vec{b}_n \in B^r$ of tuples such that $f(\vec{a}) = \widehat{p_B}(\vec{b}_1, \ldots, \vec{b}_n)$; in the first case she loses if $f(\vec{a}) \notin R^{\mathfrak{B}}$, and in the second case she loses if $\vec{b}_i \notin R^{\mathfrak{B}}$ for some $i \in [n]$. Thus if Duplicator has a winning strategy in the one round game and $\mathfrak{B} \in \mathcal{K}$ for some $\mathcal{P}$-closed quantifier $Q_{\mathcal{K}}$, then $f(\mathfrak{A}) \in \mathcal{K}$, and since f is an isomorphism $\mathfrak{A} \to f(\mathfrak{A})$, also $\mathfrak{A} \in \mathcal{K}$. In other words, if $\mathfrak{B} \models Q_{\mathcal{K}}\vec{y}\,(R_1(\vec{y}), \ldots, R_m(\vec{y}))$, then $\mathfrak{A} \models Q_{\mathcal{K}}\vec{y}\,(R_1(\vec{y}), \ldots, R_m(\vec{y}))$. The reverse implication is obtained by using the move described above with the structures switched.

By allowing only k variables and repeating rounds indefinitely (unless Spoiler wins at some round), we obtain a game such that Duplicator having a winning strategy implies $\mathfrak{A} \equiv^{k}_{\infty\omega,\mathcal{P}} \mathfrak{B}$. However, in order to prove the converse implication we need to modify the rules explained above. This is because $p_B(\mathfrak{B}) \cup \mathfrak{B}$ is not necessarily closed with respect to the function p_B, and in the argument above it would equally well suffice that $f(\mathfrak{A}) \leq \mathfrak{C}$ for some structure $\mathfrak{C}$ that is obtained by applying p_B repeatedly to $\mathfrak{B}$. In the next definition we formalize the idea of such repeated applications.

▶ **Definition 14.** *Let $p\colon A^n \to A$ be a partial function, and let $R \subseteq A^r$. We define a sequence $\Gamma^i_p(R)$, $i \in \omega$, of r-ary relations on A by the following recursion:*

- $\Gamma^0_p(R) := R$; $\Gamma^{i+1}_p(R) := p(\Gamma^i_p(R)) \cup \Gamma^i_p(R)$.

Furthermore, we define $\Gamma^{\omega}_p(R) = \bigcup_{i\in\omega} \Gamma^i_p(R)$.

This is generalized to τ-structures in the natural way: for all $i \in \omega \cup \{\omega\}$, $\Gamma^i_p(\mathfrak{A})$ is the τ-structure $\mathfrak{C}$ such that $C = A$ and $R^{\mathfrak{C}} := \Gamma^i_p(R^{\mathfrak{A}})$ for each $R \in \tau$.

Note that since $\Gamma^i_p(R) \subseteq \Gamma^{i+1}_p(R)$ for all $i \in \omega$ (assuming A is finite) there exists $j \leq |A^r|$ such that $\Gamma^{\omega}_p(R) = \Gamma^j_p(R)$. Similarly for any finite structure $\mathfrak{A}$, $\Gamma^{\omega}_p(\mathfrak{A}) = \Gamma^j_p(\mathfrak{A})$ for some $j \leq |A^r|$, where r is the maximum arity of relations in $\mathfrak{A}$.

▶ **Lemma 15.** *Let $\mathcal{P}$ be a family of n-ary partial functions. A quantifier is $\mathcal{P}$-closed if and only if the implication*

$$\mathfrak{B} \in \mathcal{K} \text{ and } \mathfrak{A} \leq \Gamma^{\omega}_{p_A}(\mathfrak{B}) \implies \mathfrak{A} \in \mathcal{K}$$

holds for all structures $\mathfrak{A}$ and $\mathfrak{B}$ with $A = B$.

Proof. Assume first that $Q_{\mathcal{K}}$ is $\mathcal{P}$-closed, $\mathfrak{B} \in \mathcal{K}$ and $\mathfrak{A} \leq \Gamma^{\omega}_{p_A}(\mathfrak{B})$. We show first by induction on i that $\Gamma^{i}_{p_A}(\mathfrak{B}) \in \mathcal{K}$ for all $i \in \omega$. For $i = 0$ this holds by assumption. If $\Gamma^{i}_{p_A}(\mathfrak{B}) \in \mathcal{K}$, then $\Gamma^{i+1}_{p_A}(\mathfrak{B}) = p_A(\mathfrak{C}) \cup \mathfrak{C}$, for $\mathfrak{C} = \Gamma^{i}_{p_A}(\mathfrak{B})$, and hence $\Gamma^{i+1}_{p_A}(\mathfrak{B}) \in \mathcal{K}$ follows from the assumption that $Q_{\mathcal{K}}$ is $\mathcal{P}$-closed.

As noted above, there exists $j \in \omega$ such that $\Gamma^{\omega}_{p_A}(\mathfrak{B}) = \Gamma^{j}_{p_A}(\mathfrak{B})$. Thus we have $\mathfrak{A} \leq \Gamma^{j}_{p_A}(\mathfrak{B}) \leq \Gamma^{j+1}_{p_A}(\mathfrak{B}) = p_A(\Gamma^{j}_{p_A}(\mathfrak{B})) \cup \Gamma^{j}_{p_A}(\mathfrak{B})$. Since $\Gamma^{j}_{p_A}(\mathfrak{B}) \in \mathcal{K}$ and $\mathcal{K}$ is $\mathcal{P}$-closed, it follows that $\mathfrak{A} \in \mathcal{K}$.

Assume then that the implication

$$(*) \quad \mathfrak{B} \in \mathcal{K} \text{ and } \mathfrak{A} \leq \Gamma^{\omega}_{p_A}(\mathfrak{B}) \implies \mathfrak{A} \in \mathcal{K}$$

holds for all $\mathfrak{A}$ and $\mathfrak{B}$ with $A = B$. Assume further that $\mathfrak{B} \in \mathcal{K}$ and $\mathfrak{A} \leq p_A(\mathfrak{B}) \cup \mathfrak{B}$. By definition $p_A(\mathfrak{B}) \cup \mathfrak{B} = \Gamma^{1}_{p_A}(\mathfrak{B})$, and since $\Gamma^{1}_{p_A}(\mathfrak{B}) \leq \Gamma^{\omega}_{p_A}(\mathfrak{B})$, we have $\mathfrak{A} \leq \Gamma^{\omega}_{p_A}(\mathfrak{B})$. Thus $\mathfrak{A} \in \mathcal{K}$ follows from the implication $(*)$. ◀

We are now ready to give the formal definition of our pebble game for $\mathcal{P}$-closed quantifiers. Let k be a positive integer. Assume that $\mathfrak{A}$ and $\mathfrak{B}$ are τ-structures for a relational vocabulary τ. Furthermore, assume that α and β are assignments on $\mathfrak{A}$ and $\mathfrak{B}$, respectively, such that $\mathrm{dom}(\alpha) = \mathrm{dom}(\beta) \subseteq X$, where $X = \{x_1, \ldots, x_k\}$. The *k-pebble $\mathcal{P}$ game* for $(\mathfrak{A}, \alpha)$ and $(\mathfrak{B}, \beta)$ is played between *Spoiler* and *Duplicator*. We denote the game by $\mathrm{PG}^{\mathcal{P}}_k(\mathfrak{A}, \mathfrak{B}, \alpha, \beta)$, and we use the shorthand notation $\mathrm{PG}^{\mathcal{P}}_k(\alpha, \beta)$ whenever $\mathfrak{A}$ and $\mathfrak{B}$ are clear from the context.

▶ **Definition 16.** *The rules of the game $\mathrm{PG}^{\mathcal{P}}_k(\mathfrak{A}, \mathfrak{B}, \alpha, \beta)$ are the following:*

(1) *If $\alpha \mapsto \beta \notin \mathrm{PI}(\mathfrak{A}, \mathfrak{B})$, then the game ends, and Spoiler wins.*

(2) *If (1) does not hold, there are two types of moves that Spoiler can choose to play:*

- **Left $\mathcal{P}$-quantifier move:** *Spoiler starts by choosing $r \in [k]$ and an r-tuple $\vec{y} \in X^r$ of distinct variables. Duplicator responds with a bijection $f \colon B \to A$. Spoiler answers by choosing an r-tuple $\vec{b} \in B^r$. Duplicator answers by choosing $P \subseteq A^r$ such that $f(\vec{b}) \in \Gamma^{\omega}_{p_A}(P)$. Spoiler completes the round by choosing $\vec{a} \in P$. The players continue by playing $\mathrm{PG}^{\mathcal{P}}_k(\alpha', \beta')$, where $\alpha' := \alpha[\vec{a}/\vec{y}]$ and $\beta' := \beta[\vec{b}/\vec{y}]$.*
- **Right $\mathcal{P}$-quantifier move:** *Spoiler starts by choosing $r \in [k]$ and an r-tuple $\vec{y} \in X^r$ of distinct variables. Duplicator chooses next a bijection $f \colon A \to B$. Spoiler answers by choosing an r-tuple $\vec{a} \in A^r$. Duplicator answers by choosing $P \subseteq B^r$ such that $f(\vec{a}) \in \Gamma^{\omega}_{p_B}(P)$. Spoiler completes the round by choosing $\vec{b} \in P$. The players continue by playing $\mathrm{PG}^{\mathcal{P}}_k(\alpha', \beta')$, where $\alpha' := \alpha[\vec{a}/\vec{y}]$ and $\beta' := \beta[\vec{b}/\vec{y}]$.*

(3) *Duplicator wins the game if Spoiler does not win it in a finite number of rounds.*

We now prove that the game $\mathrm{PG}^{\mathcal{P}}_k$ indeed characterizes equivalence of structures with respect to the infinitary k-variable logic with all $\mathcal{P}$-closed quantifiers.

▶ **Theorem 17.** *Let $\mathcal{P}$ be an invariant family of partial functions. Then Duplicator has a winning strategy in $\mathrm{PG}^{\mathcal{P}}_k(\mathfrak{A}, \mathfrak{B}, \alpha, \beta)$ if, and only if, $(\mathfrak{A}, \alpha) \equiv^{k}_{\infty\omega,\mathcal{P}} (\mathfrak{B}, \beta)$.*

Proof.

$\Rightarrow$: We prove by induction on $\varphi \in L^{k}_{\infty\omega}(\mathbf{Q}_{\mathcal{P}})$ that (for any assignments α and β) if Duplicator has a winning strategy in $\mathrm{PG}^{\mathcal{P}}_k(\alpha, \beta)$, then $(\mathfrak{A}, \alpha) \models \varphi \iff (\mathfrak{B}, \beta) \models \varphi$.

- If φ is an atomic formula, the claim follows from the fact that Spoiler always wins the game $\mathrm{PG}_k^{\mathcal{P}}(\alpha,\beta)$ immediately if $\alpha \mapsto \beta \notin \mathrm{PI}(\mathfrak{A},\mathfrak{B})$.
- The cases $\varphi = \neg\psi$, $\varphi = \bigvee \Psi$ and $\varphi = \bigwedge \Psi$ are straightforward.
- By Proposition 6, the negation of the existential quantifier is in $\mathbf{Q}_{\mathcal{P}}$, and hence we do not need to consider the case $\varphi = \exists x_i \psi$ separately.
- Consider then the case $\varphi = Q_{\mathcal{K}}\vec{y}\mathcal{I}$ for some r-ary quantifier $Q_{\mathcal{K}} \in \mathbf{Q}_{\mathcal{P}}$ and interpretation $\mathcal{I} = (\psi_1,\dots,\psi_\ell)$. We start by assuming that $(\mathfrak{A},\alpha) \models \varphi$. Thus, $\mathcal{I}(\mathfrak{A},\alpha) := (A, R_1,\dots,R_\ell) \in \mathcal{K}$. Let Spoiler play in the game $\mathrm{PG}_k^{\mathcal{P}}(\alpha,\beta)$ a left $\mathcal{P}$-quantifier move with r and the tuple $\vec{y} \in X^r$, and let $f\colon B \to A$ be the bijection given by the winning strategy of Duplicator. Let $\mathcal{I}(\mathfrak{B},\beta) := (B, R'_1,\dots,R'_\ell)$, and for each $i \in [\ell]$, let $S_i := f(R'_i)$. We claim that $\mathfrak{D} := (A, S_1,\dots,S_\ell) \in \mathcal{K}$. Since f is an isomorphism $\mathcal{I}(\mathfrak{B},\beta) \to \mathfrak{D}$, it follows then that $(\mathfrak{B},\beta) \models \varphi$.
  To prove the claim it suffices to show that $\mathfrak{D} \le \Gamma^{\omega}_{p_A}(\mathcal{I}(\mathfrak{A},\alpha))$, since then $\mathfrak{D} \in \mathcal{K}$ by Lemma 15 and the assumption that $Q_{\mathcal{K}}$ is $\mathcal{P}$-closed. To show this, let $i \in [\ell]$ and $\vec{c} \in S_i$. We let Spoiler choose the tuple $\vec{b} = f^{-1}(\vec{c})$ as his answer to the bijection f. Thus, $(\mathfrak{B},\beta[\vec{b}/\vec{y}]) \models \psi_i$. Let $P \subseteq A^r$ be the answer of Duplicator. Then by the rules of the game $\vec{c} \in \Gamma^{\omega}_{p_A}(P)$, and Duplicator has a winning strategy in the game $\mathrm{PG}_k^{\mathcal{P}}(\alpha[\vec{a}/\vec{y}],\beta[\vec{b}/\vec{y}])$ for all $\vec{a} \in P$. Hence by induction hypothesis $(\mathfrak{A},\alpha[\vec{a}/\vec{y}]) \models \psi_i$, i.e., $\vec{a} \in R_i$, holds for all $\vec{a} \in P$. This shows that $S_i \subseteq \Gamma^{\omega}_{p_A}(R_i)$, and since this holds for all $i \in [\ell]$, we see that $\mathfrak{D} \le \Gamma^{\omega}_{p_A}(\mathcal{I}(\mathfrak{A},\alpha))$.
  By using the right $\mathcal{P}$-quantifier move in place of the left quantifier move, we can prove that $(\mathfrak{B},\beta) \models \varphi$ implies $(\mathfrak{A},\alpha) \models \varphi$. Thus, $(\mathfrak{A},\alpha) \models \varphi \iff (\mathfrak{B},\beta) \models \varphi$, as desired.

$\Leftarrow$: Assume then that $(\mathfrak{A},\alpha) \equiv^k_{\infty\omega,\mathcal{P}} (\mathfrak{B},\beta)$. Clearly it suffices to show that Duplicator can play in the first round of the game $\mathrm{PG}_k^{\mathcal{P}}(\alpha,\beta)$ in such a way that $(\mathfrak{A},\alpha') \equiv^k_{\infty\omega,\mathcal{P}} (\mathfrak{B},\beta')$ holds, where α' and β' are the assignments arising from the choices of Spoiler and Duplicator.

Assume first that Spoiler decides to play a left $\mathcal{P}$-quantifier move in the first round of $\mathrm{PG}_k^{\mathcal{P}}(\alpha,\beta)$. Let $\vec{y} \in X^r$ be the tuple of variables he chooses. Since A and B are finite, for each $\vec{a} \in A^r$ there is a formula $\Psi_{\vec{a}} \in L^k_{\infty\omega}(\mathbf{Q}_{\mathcal{P}})$ such that for any τ-structure $\mathfrak{C}$ of size at most $\max\{|A|,|B|\}$, any assignment γ on $\mathfrak{C}$, and any tuple $\vec{c} \in C^r$ we have

- $(\mathfrak{A},\alpha[\vec{a}/\vec{y}]) \equiv^k_{\infty\omega,\mathcal{P}} (\mathfrak{C},\gamma[\vec{c}/\vec{y}])$ if and only if $(\mathfrak{C},\gamma[\vec{c}/\vec{y}]) \models \Psi_{\vec{a}}$.

Let $\vec{c}_1,\dots,\vec{c}_\ell$ be a fixed enumeration of the set A^r, and let $\mathcal{I}$ be the interpretation $(\Psi_1,\dots,\Psi_m)$, where $\Psi_j := \Psi_{\vec{c}_j}$ for each $j \in [m]$. We define $\mathcal{K}$ to be the closure of the class $\{\mathfrak{D} \mid \mathfrak{D} \le \Gamma^{\omega}_{p_A}(\mathcal{I}(\mathfrak{A},\alpha))\}$ under isomorphisms. Note that if $\mathfrak{D} \le \Gamma^{\omega}_{p_A}(\mathcal{I}(\mathfrak{A},\alpha))$ and $\mathfrak{E} \le \Gamma^{\omega}_{p_A}(\mathfrak{D})$, then clearly $\mathfrak{E} \le \Gamma^{\omega}_{p_A}(\mathcal{I}(\mathfrak{A},\alpha))$. Hence by Lemma 15, the quantifier $Q_{\mathcal{K}}$ is $\mathcal{P}$-closed. Moreover, since $\mathcal{I}(\mathfrak{A},\alpha) \in \mathcal{K}$, we have $(\mathfrak{A},\alpha) \models Q_{\mathcal{K}}\vec{y}\mathcal{I}$, and consequently by our assumption, $(\mathfrak{B},\beta) \models Q_{\mathcal{K}}\vec{y}\mathcal{I}$. Thus, there is a structure $\mathfrak{D} \le \Gamma^{\omega}_{p_A}(\mathcal{I}(\mathfrak{A},\alpha))$ and an isomorphism $f\colon \mathcal{I}(\mathfrak{B},\beta) \to \mathfrak{D}$. We let Duplicator to use the bijection $f\colon B \to A$ as her answer to the choice $\vec{y}$ of Spoiler.

Let $\vec{b} \in B^r$ be the answer of Spoiler to f, and let $\vec{c} = f(\vec{b})$. Clearly $(\mathfrak{A},\alpha) \models \forall\vec{y} \bigvee_{j\in[\ell]} \Psi_j$, whence there exists $j \in [\ell]$ such that $(\mathfrak{B},\beta[\vec{b}/\vec{y}]) \models \Psi_j$, or in other words, $\vec{b} \in R_j^{\mathcal{I}(\mathfrak{B},\beta)}$. Since f is an isomorphism $\mathcal{I}(\mathfrak{B},\beta) \to \mathfrak{D}$, we have $\vec{c} \in R_j^{\mathfrak{D}}$. We let Duplicator to use $P := R_j^{\mathcal{I}(\mathfrak{A},\alpha)}$ as her answer to the choice $\vec{b}$ of Spoiler; this is a legal move since $\mathfrak{D} \le \Gamma^{\omega}_{p_A}(\mathcal{I}(\mathfrak{A},\alpha))$. Observe now that since $P = R_j^{\mathcal{I}(\mathfrak{A},\alpha)}$, we have $(\mathfrak{A},\alpha[\vec{a}/\vec{y}]) \models \Psi_{\vec{c}_j}$, and consequently $(\mathfrak{A},\alpha[\vec{c}_j/\vec{y}]) \equiv^k_{\infty\omega,\mathcal{P}} (\mathfrak{A},\alpha[\vec{a}/\vec{y}])$, for all $\vec{a} \in P$. On the other hand we also have $(\mathfrak{B},\beta[\vec{b}/\vec{y}]) \models \Psi_{\vec{c}_j}$, and hence $(\mathfrak{A},\alpha[\vec{c}_j/\vec{y}]) \equiv^k_{\infty\omega,\mathcal{P}} (\mathfrak{B},\beta[\vec{b}/\vec{y}])$. Thus the condition $(\mathfrak{A},\alpha') \equiv^k_{\infty\omega,\mathcal{P}} (\mathfrak{B},\beta')$, where $\alpha' = \alpha[\vec{a}/\vec{y}]$ and $\beta' = \beta[\vec{b}/\vec{y}]$, holds after the first round of $\mathrm{PG}_k^{\mathcal{P}}(\alpha,\beta)$ irrespective of the choice $\vec{a} \in P$ of Spoiler in the end of the round.

The case where Spoiler starts with a right $\mathcal{P}$-quantifier move is handled in the same way by switching the roles of $(\mathfrak{A}, \alpha)$ and $(\mathfrak{B}, \beta)$. ◀

5 Playing the game

In this section we use the game $\mathrm{PG}^k_{\mathcal{P}}$ to show inexpressibility of a property of finite structures in the infinitary finite variable logic $L^\omega_{\infty\omega}$ augmented by all $\mathcal{N}_\ell$-closed quantifiers. More precisely, we prove that the Boolean constraint satisfaction problem $\mathsf{CSP}(\mathfrak{C}_\ell)$, where $\mathfrak{C}_\ell$ is the structure with $C = \{0, 1\}$ and two ℓ-ary relations $R_0 = \{(b_1, \ldots, b_\ell) \mid \sum_{i\in[\ell]} b_i \equiv 0 \pmod 2\}$ and $R_1 = \{(b_1, \ldots, b_\ell) \mid \sum_{i\in[\ell]} b_i \equiv 1 \pmod 2\}$, is not definable in $L^\omega_{\infty\omega}(\mathbf{Q}_{\mathcal{N}_\ell})$.

In the proof of the undefinability of $\mathsf{CSP}(\mathfrak{C}_\ell)$ we use a variation of the well-known CFI construction, due to Cai, Fürer and Immerman [6]. Our construction is a minor modification of the one that was used in [17] for producing non-isomorphic structures on which Duplicator wins the k, n-bijection game. We start by explaining the details of the construction.

Let $G = (V, E, \leq^G)$ be a connected ℓ-regular ordered graph. For each vertex $v \in V$, we use the notation $E(v)$ for the set of edges adjacent to v and $\vec{e}(v) = (e_1, \ldots, e_\ell)$ for the tuple that lists $E(v)$ in the order $\leq^G$. The CFI structures we use have in the universe two elements $(e, 1)$ and $(e, 2)$ for each $e \in E$, and two ℓ-ary relations that connect such pairs (e, i) for edges e that are adjacent to some vertex $v \in V$.

▶ **Definition 18.** *Let $G = (V, E, \leq^G)$ be a connected ℓ-regular ordered graph and let $U \subseteq V$. We define a CFI structure $\mathfrak{A}_\ell(G, U) = (A_\ell(G), R_0^{\mathfrak{A}_\ell(G,U)}, R_1^{\mathfrak{A}_\ell(G,U)})$, where $\mathrm{ar}(R_0) = \mathrm{ar}(R_1) = \ell$, as follows.*

- $A_\ell(G) := E \times [2]$,
- $R_0^{\mathfrak{A}_\ell(G,U)} := \bigcup_{v\in V\setminus U} R(v) \cup \bigcup_{v\in U} \tilde{R}(v)$ *and* $R_1^{\mathfrak{A}_\ell(G,U)} := \bigcup_{v\in U} R(v) \cup \bigcup_{v\in V\setminus U} \tilde{R}(v)$, *where*
  - $R(v) := \{((e_1, i_1), \ldots, (e_\ell, i_\ell)) \mid (e_1, \ldots, e_\ell) = \vec{e}(v), \sum_{j\in[\ell]} i_j = 0 \pmod 2\}$, *and*
  - $\tilde{R}(v) := \{((e_1, i_1), \ldots, (e_\ell, i_\ell)) \mid (e_1, \ldots, e_\ell) = \vec{e}(v), \sum_{j\in[\ell]} i_j = 1 \pmod 2\}$.

For each $v \in V$, we denote the set $E(v) \times [2]$ by $A(v)$. Furthermore, we define $\mathfrak{A}_\ell(v) := (A(v), R(v), \tilde{R}(v))$ and $\tilde{\mathfrak{A}}_\ell(v) := (A(v), \tilde{R}(v), R(v))$.

By a similar argument as in the CFI structures constructed in [17] and [18] it can be proved that $\mathfrak{A}_\ell(G, U)$ and $\mathfrak{A}_\ell(G, U')$ are isomorphic if and only if $|U|$ and $|U'|$ are of the same parity. We choose $\mathfrak{A}^{\mathrm{ev}}_\ell(G) := \mathfrak{A}_\ell(G, \emptyset)$ and $\mathfrak{A}^{\mathrm{od}}_\ell(G) := \mathfrak{A}_\ell(G, \{v_0\})$ as representatives of these two isomorphism classes, where v_0 is the least element of V with respect to the linear order $\leq^G$. We show first that these structures are separated by $\mathsf{CSP}(\mathfrak{C}_\ell)$.

▶ **Lemma 19.** *$\mathfrak{A}^{\mathrm{ev}}_\ell(G) \in \mathsf{CSP}(\mathfrak{C}_\ell)$, but $\mathfrak{A}^{\mathrm{od}}_\ell(G) \notin \mathsf{CSP}(\mathfrak{C}_\ell)$.*

Proof. Let $h\colon A_\ell(G) \to \{0, 1\}$ be the function such that $h((e, 1)) = 1$ and $h((e, 2)) = 0$ for all $e \in E$. Then for any tuple $((e_1, i_1), \ldots, (e_\ell, i_\ell))$ the parity of $\sum_{j\in[\ell]} h((e_j, i_j))$ is the same as the parity of $\sum_{j\in[\ell]} i_j$. Thus, h is a homomorphism $\mathfrak{A}^{\mathrm{ev}}_\ell(G) \to \mathfrak{C}_\ell$.

To show that $\mathfrak{A}^{\mathrm{od}}_\ell(G) \notin \mathsf{CSP}(\mathfrak{C}_\ell)$, assume towards contradiction that $g\colon A_\ell(G) \to \{0, 1\}$ is a homomorphism $\mathfrak{A}^{\mathrm{od}}_\ell(G) \to \mathfrak{C}_\ell$. Then for every $e \in E$ necessarily $g((e, 1)) \neq g((e, 2))$. Furthermore, for every $v \in V \setminus \{v_0\}$, the number $n_v := |\{e \in E(v) \mid g((e, 2)) = 1\}|$ must be even, while the number n_{v_0} must be odd. Thus, $\sum_{v\in V} n_v$ must be odd. However, this is impossible, since clearly $\sum_{v\in V} n_v = 2|\{e \in E \mid g((e, 2)) = 1\}|$. ◀

Our aim is to prove, for a suitable graph G, that Duplicator has a winning strategy in $\mathrm{PG}^{\mathcal{N}_\ell}_k(\mathfrak{A}^{\mathrm{ev}}_\ell(G), \mathfrak{A}^{\mathrm{od}}_\ell(G), \emptyset, \emptyset)$. For the winning strategy, Duplicator needs a collection of well-behaved bijections. We define such a collection GB in Definition 23 below. One requirement is that the bijections preserve the first component of the elements $(e, i) \in A_\ell(G)$.

▶ **Definition 20.** *A bijection* $f\colon A_\ell(G) \to A_\ell(G)$ *is* edge preserving *if for every* $e \in E$ *and* $i \in [2]$, $f((e,i))$ *is either* $(e,1)$ *or* $(e,2)$.

For any edge preserving f and any $v \in V$ we denote by f_v the restriction of f to the set $E(v) \times [2]$. The *switching number* $\mathrm{swn}(f_v)$ of f_v is $|\{e \in E(v) \mid f_v((e,1)) = (e,2)\}|$. The lemma below follows directly from the definitions of $\mathfrak{A}_\ell(v)$ and $\tilde{\mathfrak{A}}_\ell(v)$.

▶ **Lemma 21.** *Let* $f\colon A_\ell(G) \to A_\ell(G)$ *be an edge preserving bijection, and let* $v \in V$.
(a) *If* $\mathrm{swn}(f_v)$ *is even, then* f_v *is an automorphism of the structures* $\mathfrak{A}_\ell(v)$ *and* $\tilde{\mathfrak{A}}_\ell(v)$.
(b) *If* $\mathrm{swn}(f_v)$ *is odd, then* f_v *is an isomorphism between the structures* $\mathfrak{A}_\ell(v)$ *and* $\tilde{\mathfrak{A}}_\ell(v)$.

Given an edge preserving bijection $f\colon A_\ell(G) \to A_\ell(G)$ we denote by $\mathrm{Odd}(f)$ the set of all $v \in V$ such that $\mathrm{swn}(f_v)$ is odd. Observe that $|\mathrm{Odd}(f)|$ is necessarily even.

▶ **Corollary 22.** *An edge preserving bijection* $f\colon A_\ell(G) \to A_\ell(G)$ *is an automorphism of the structures* $\mathfrak{A}_\ell^{\mathrm{ev}}(G)$ *and* $\mathfrak{A}_\ell^{\mathrm{od}}(G)$ *if and only if* $\mathrm{Odd}(f) = \emptyset$.

Proof. If $\mathrm{Odd}(f) = \emptyset$, then by Lemma 21(a) f_v is an automorphism of $\mathfrak{A}_\ell(v)$ and $\tilde{\mathfrak{A}}_\ell(v)$ for all $v \in V$. Clearly this means that f is an automorphism of $\mathfrak{A}_\ell^{\mathrm{ev}}(G)$ and $\mathfrak{A}_\ell^{\mathrm{od}}(G)$. On the other hand, if $v \in \mathrm{Odd}(f)$, then by Lemma 21(b), for any tuple $\vec{a} \in A(v)^\ell$, we have $\vec{a} \in R(v) \iff f(\vec{a}) \in \tilde{R}(v)$. Since $R(v) \cap \tilde{R}(v) = \emptyset$, it follows that f is not an automorphism of $\mathfrak{A}_\ell^{\mathrm{ev}}(G)$ and $\mathfrak{A}_\ell^{\mathrm{od}}(G)$. ◀

▶ **Definition 23.** *Let* $f\colon A_\ell(G) \to A_\ell(G)$ *be edge preserving bijection. Then* f *is a* good bijection *if either* $\mathrm{Odd}(f) = \emptyset$ *or* $\mathrm{Odd}(f) = \{v_0, v\}$ *for some* $v \in V \setminus \{v_0\}$. *We denote the set of all good bijections by* GB.

Note that if $f\colon A_\ell(G) \to A_\ell(G)$ is a good bijection, then there is exactly one vertex $v^* \in V$ such that f_{v^*} is not a partial isomorphism $\mathfrak{A}_\ell^{\mathrm{ev}}(G) \to \mathfrak{A}_\ell^{\mathrm{od}}(G)$. In case $\mathrm{Odd}(f) = \emptyset$, $v^* = v_0$, while in case $\mathrm{Odd}(f) = \{v_0, v\}$ for some $v \neq v_0$, $v^* = v$. We denote this vertex v^* by $\mathrm{tw}(f)$ (the *twist* of f).

Assume now that Duplicator has played a good bijection f in the game $\mathrm{PG}_k^{\mathcal{N}_\ell}$ on the structures $\mathfrak{A}_\ell^{\mathrm{ev}}(G)$ and $\mathfrak{A}_\ell^{\mathrm{od}}(G)$. Then it is sure that Spoiler does not win the game in the next position (α, β) if $(e,1)$ and $(e,2)$ are not in the range of α (and β) for any $e \in E(\mathrm{tw}(f))$. This leads us to the following notion.

▶ **Definition 24.** *Let* f *be a* good *bijection, and let* $F \subseteq E$. *Then* f *is* good for F *if* $E(\mathrm{tw}(f)) \cap F = \emptyset$. *We denote the set of all bijections that are good for* F *by* $\mathrm{GB}(F)$.

▶ **Lemma 25.** *If* $f \in \mathrm{GB}(F)$, *then* $f \restriction (F \times [2])$ *is a partial isomorphism* $\mathfrak{A}_\ell^{\mathrm{ev}}(G) \to \mathfrak{A}_\ell^{\mathrm{od}}(G)$.

Proof. Clearly $f \restriction (F \times [2]) \subseteq \bigcup_{v \in V \setminus \{\mathrm{tw}(f)\}} f_v$. By Lemma 21, f_v is an automorphism of $\mathfrak{A}_\ell(v)$ for any $v \in V \setminus \{\mathrm{tw}(f), v_0\}$, and if $v_0 \neq \mathrm{tw}(f)$, f_{v_0} is an isomorphism $\mathfrak{A}_\ell(v) \to \tilde{\mathfrak{A}}_\ell(v)$. The claim follows from this. ◀

Given a good bijection f with $\mathrm{tw}(f) = u$ and an E-path $P = (u_0, \ldots, u_m)$ from $u = u_0$ to $u' = u_m$, we obtain a new edge preserving bijection f_P by switching f on the edges $e_i := \{u_i, u_{i+1}\}$, $i < m$, of P: $f_P((e_i, j)) = (e_i, 3-j)$ for $i < m$, and $f_P(c) = f(c)$ for other $c \in A_\ell(G)$. Clearly f_P is also a good bijection, and $\mathrm{tw}(f_P) = u'$.

In order to prove that Duplicator has a winning strategy in $\mathrm{PG}_k^{\mathcal{N}_\ell}(\mathfrak{A}_\ell^{\mathrm{ev}}(G), \mathfrak{A}_\ell^{\mathrm{od}}(G), \emptyset, \emptyset)$ we need to assume that the graph G has a certain largeness property with respect the number k. We formulate this largeness property in terms of a game, $\mathrm{CRG}_k^\ell(G)$, that is a new variation of the *Cops&Robber games* used for similar purposes in [17] and [18].

▶ **Definition 26.** *The game* $\mathrm{CRG}_k^\ell(G)$ *is played between two players, Cop and Robber. The positions of the game are pairs* (F, u)*, where* $F \subseteq E$*,* $|F| \leq k$*, and* $u \in V$*. The rules of the game are the following:*

- *Assume that the position is* (F, u)*.*
- *If* $E(u) \cap F \neq \emptyset$*, the game ends and Cop wins.*
- *Otherwise Cop chooses a set* $F' \subseteq E$ *such that* $|F'| \leq k$*. Then Robber answers by giving mutually disjoint* $E \setminus (F \cap F')$*-paths* $P_i = (u, u_1^i, \ldots, u_{n_i}^i)$*,* $i \in [\ell]$*, from* u *to vertices* $u_i := u_{n_i}^i$*; here mutual disjointness means that* P_i *and* $P_{i'}$ *do not share edges for* $i \neq i'$ *(i.e.,* $u_1^i \neq u_1^{i'}$ *and* $\{u_j^i, u_{j+1}^i\} \neq \{u_{j'}^{i'}, u_{j'+1}^{i'}\}$ *for all* j *and* j'*). Then Cop completes the round by choosing* $i \in [\ell]$*. The next position is* (F', u_i)*.*

The intuition of the game $\mathrm{CRG}_k^\ell(G)$ is that Cop has k pebbles that he plays on edges of G forming a set $F \subseteq E$; these pebbles mark the edges e such that $(e, 1)$ or $(e, 2)$ is in the range of α or β in a position (α, β) of the game $\mathrm{PG}_k^{\mathcal{N}_\ell}$ on $\mathfrak{A}_\ell^{\mathrm{ev}}(G)$ and $\mathfrak{A}_\ell^{\mathrm{od}}(G)$. Robber has one pebble that she plays on the vertices of G; this pebble marks the vertex $\mathrm{tw}(f)$, where f is the good bijection played by Duplicator in the previous round of $\mathrm{PG}_k^{\mathcal{N}_\ell}$.

Cop captures Robber and wins the game if after some round (at least) one of his pebbles is on an edge that is adjacent to the vertex containing Robber's pebble. This corresponds to a position (α, β) in the game $\mathrm{PG}_k^{\mathcal{N}_\ell}$ such that $\alpha \mapsto \beta$ is potentially not a partial isomorphism. Otherwise Lemma 25 guarantees that $\alpha \mapsto \beta$ is a partial isomorphism. Cop can then move any number of his pebbles to new positions on G. While the pebbles Cop decides to move are still on their way to their new positions, Robber is allowed to prepare ℓ mutually disjoint escape routes along edges of G that do not contain any stationary pebbles of Cop. We show in the proof of Theorem 29 that these escape routes generate tuples $\vec{a}_1, \ldots, \vec{a}_\ell$ such that $f(\vec{b}) = \hat{q}(\vec{a}_1, \ldots, \vec{a}_\ell)$, where $q = n_{A_\ell(G)}^\ell$ and $\vec{b}$ is the tuple chosen by Spoiler after Duplicator played f. This gives Duplicator a legal answer $P = \{\vec{a}_1, \ldots, \vec{a}_\ell\}$ to $\vec{b}$. Then Spoiler completes the round by choosing one of the tuples in P. Correspondingly, in the end of the round of $\mathrm{CRG}_k^\ell(G)$ Cop chooses which escape route Robber has to use by blocking all but one of them.

▶ **Definition 27.** *Assume that* $u \in V$ *and* $F \subseteq E$ *is a set of edges such that* $|F| \leq k$*. We say that* u *is* k-safe for F *if Robber has a winning strategy in the game* $\mathrm{CRG}_k^\ell(G)$ *starting from position* (F, u)*.*

We prove next the existence of graphs G such that Robber has a winning strategy in the game $\mathrm{CRG}_k^\ell(G)$.

▶ **Theorem 28.** *For every* $\ell \geq 3$ *and every* $k \geq 1$*, there is an* ℓ*-regular graph* $G = (V, E)$ *such that every vertex* $v \in V$ *is* k*-safe for* $\emptyset$*.*

Proof. Clearly if Robber has a winning strategy in $\mathrm{CRG}_k^\ell(G)$, it also has a winning strategy in $\mathrm{CRG}_{k'}^\ell(G)$ for $k' < k$. Thus, it suffices to prove the theorem for $k \geq \ell$.

By a well-known result of Erdös and Sachs [12] (see also [1] for a more accessible construction), there exist ℓ-regular connected graphs of *girth* g for arbitrarily large g. Choose a positive integer d with $d > \frac{\log 2k}{\log(\ell-1)} + 1$ and let G be an ℓ-regular graph of girth $g > 6d$. We claim that any vertex v in G is k-safe for $\emptyset$.

To prove this, we show inductively that Robber can maintain the following invariant in any position (F, u) reached during the game:

$(*)$ for each edge $e \in F$, neither end point of e is within distance d of u in G.

Note that, from the assumption that $k \geq \ell$ and $d > \frac{\log 2k}{\log(\ell-1)} + 1$, it follows that $d \geq 2$. Thus, the invariant $(*)$ guarantees, in particular, that Cop does not win at any point.

Clearly the invariant $(*)$ is satisfied at the initial position, since F is empty. Suppose now that it is satisfied in some position (F, u) and Cop chooses a set F' in the next move. Let $C \subseteq V$ denote the set of end points of all edges in F'. Since $|F'| \leq k$, we have $|C| \leq 2k$.

Let $N \subseteq V$ denote the collection of vertices which are at distance at most $3d$ from u. By the assumption on the girth of G, the induced subgraph $G[N]$ is a tree. We can consider it as a rooted tree, with root u. Then, u has exactly ℓ children. All vertices in N at distance less than $3d$ from u have exactly $\ell - 1$ children (and one parent), and all vertices at distance exactly $3d$ from u are leaves of the tree. This allows us to speak, for instance, of the subtree rooted at a vertex u' meaning the subgraph of G induced by the vertices x in N such that the unique path from u to x in $G[N]$ goes through u'.

Let $u_1, \ldots, u_\ell$ be the children of u. For each i, let U_i denote the set of descendants of u_i that are at distance exactly d from u (and so at distance $d-1$ from u_i). Note that the collection $U_1, \ldots, U_\ell$ forms a partition of the set of vertices in N that are at distance exactly d from u. Each $x \in U_i$ is the root of a tree of height $2d$. Moreover, since the tree below u_i is $(\ell-1)$-regular, U_i contains exactly $(\ell-1)^{d-1}$ vertices. By the assumption that $d > \frac{\log 2k}{\log(\ell-1)} + 1$, it follows that $(\ell-1)^{d-1} > 2k \geq |C|$ and therefore each U_i contains at least one vertex x_i such that the subtree rooted at x_i contains no vertex in C. Let y_i be any descendant of x_i at distance d from x_i and let P_i denote the unique path in $G[N]$ from u to y_i. Robber's move is to play the paths $P_1, \ldots, P_\ell$. We now verify that this is a valid move, and that it maintains the required invariant $(*)$.

First, note that the paths $P_1, \ldots, P_\ell$ are paths in the tree $G[N]$ all starting at u and the second vertex in path P_i is u_i. It follows that the paths are pairwise edge disjoint. We next argue that no path P_i goes through an edge in $F \cap F'$. Indeed, by the inductive assumption, no endpoint of an edge in F appears within distance d of u and therefore the path from u to x_i does not go through any such vertex. Moreover, by the choice of x_i, no endpoint of an edge in F' appears in the subtree rooted at x_i and therefore the path from x_i to y_i does not go through any such vertex. Together these ensure that the path P_i does not visit any vertex that is an endpoint of an edge in $F \cap F'$.

Finally, to see that the invariant $(*)$ is maintained, note that all vertices that are at distance at most d from y_i are in the subtree of $G[N]$ rooted at x_i. The choice of x_i means this contains no vertex in C. This is exactly the condition that we wished to maintain. ◀

We are now ready to prove that a winning strategy for Robber in $\mathrm{CRG}_k^\ell(G)$ generates a winning strategy for Duplicator in the game $\mathrm{PG}_k^{\mathcal{N}_\ell}$ on the structures $\mathfrak{A}_\ell^{\mathrm{ev}}(G)$ and $\mathfrak{A}_\ell^{\mathrm{od}}(G)$.

▶ **Theorem 29.** *Let G be a connected ℓ-regular ordered graph. If v_0 is k-safe for the empty set, then Duplicator has a winning strategy in the game $\mathrm{PG}_k^{\mathcal{N}_\ell}(\mathfrak{A}_\ell^{\mathrm{ev}}(G), \mathfrak{A}_\ell^{\mathrm{od}}(G), \emptyset, \emptyset)$.*

Proof. We show that Duplicator can maintain the following invariant for all positions (α, β) obtained during the play of the game $\mathrm{PG}_k^{\mathcal{N}_\ell}(\mathfrak{A}_\ell^{\mathrm{ev}}(G), \mathfrak{A}_\ell^{\mathrm{od}}(G), \emptyset, \emptyset)$:

(†) There exists a bijection $f \in \mathrm{GB}(F_\alpha)$ such that $p := \alpha \mapsto \beta \subseteq f$ and $\mathrm{tw}(f)$ is k-safe for F_α, where $F_\alpha := \{e \in E \mid \mathrm{rng}(\alpha) \cap \{e\} \times [2] \neq \emptyset\}$.

Note that if (†) holds, then $p \subseteq f \restriction (F_\alpha \times [2])$ and, by Lemma 25, $f \restriction (F_\alpha \times [2]) \in \mathrm{PI}(\mathfrak{A}_\ell^{\mathrm{ev}}(G), \mathfrak{A}_\ell^{\mathrm{od}}(G))$, whence Spoiler does not win the game in position (α, β). Thus, maintaining the invariant (†) during the play guarantees a win for Duplicator.

Note first that (†) holds in the initial position $(\alpha, \beta) = (\emptyset, \emptyset)$ of the game: if $f_0 \in \mathrm{GB}$ is the bijection with $\mathrm{tw}(f_0) = v_0$, as $\emptyset \mapsto \emptyset = \emptyset \subseteq f_0$ and $\mathrm{tw}(f_0)$ is k-safe for $F_\emptyset = \emptyset$.

Assume then that (†) holds for a position (α, β), and assume that Spoiler plays a left $\mathcal{N}_\ell$-quantifier move by choosing $r \leq k$ and $\vec{y} \in X^r$. Duplicator answers this by giving the bijection f^{-1}. Let $\vec{b} = (b_1, \ldots, b_r) \in A_\ell(G)^r$ be the second part of Spoiler's move, and let

F' be the set $\{e \in E \mid \mathrm{rng}(\beta[\vec{b}/\vec{y}]) \cap \{e\} \times [2] \neq \emptyset\}$. Since $\mathrm{tw}(f)$ is k-safe for F_α, there are mutually disjoint $E \setminus (F_\alpha \cap F')$-paths P_i, $i \in [\ell]$, from $\mathrm{tw}(f)$ to some vertices u_i that are k-safe for the set F'. Let f_{P_i}, $i \in [\ell]$, be the good bijections obtained from f as explained before Definition 24. Now Duplicator answers the move $\vec{b}$ of Spoiler by giving the set $P = \{\vec{a}_1, \ldots, \vec{a}_\ell\}$ of r-tuples, where $\vec{a}_i := f_{P_i}^{-1}(\vec{b})$ for each $i \in [\ell]$.

To see that this is a legal move, observe that since the paths P_i are disjoint, for each $j \in [r]$ there is at most one $i \in [\ell]$ such that $f_{P_i}^{-1}(b_j) \neq f^{-1}(b_j)$. Thus we have $\hat{q}(\vec{a}_1, \ldots, \vec{a}_\ell) = f^{-1}(\vec{b})$, and hence $f^{-1}(\vec{b}) \in q(P) \subseteq \Gamma_q^\omega(P)$ for $q = n^\ell_{A_\ell(G)}$, as required. Let Spoiler complete the round of the game by choosing $i \in [\ell]$; thus, the next position is $(\alpha', \beta') := (\alpha[\vec{a}_i/\vec{y}], \beta[\vec{b}/\vec{y}])$. It suffices now to show that (†) holds for the position (α', β') and the bijection $f' := f_{P_i}$.

Note first that $F_{\alpha'} = F'$, since clearly $\mathrm{rng}(\alpha[\vec{a}_i/\vec{y}]) \cap \{e\} \times [2] \neq \emptyset$ if, and only if, $\mathrm{rng}(\beta[\vec{b}/\vec{y}]) \cap \{e\} \times [2] \neq \emptyset$. Thus, $\mathrm{tw}(f') = u_i$ is k-safe for $F_{\alpha'}$. This implies that $f' \in \mathrm{GB}(F_{\alpha'})$, since otherwise by Definition 26, Cop would win the game $\mathrm{CRG}_k(G)$ immediately in position $(F_{\alpha'}, \mathrm{tw}(f'))$. It remains to show that $p' := \alpha' \mapsto \beta'$ is contained in f'. For all components a_i^j of $\vec{a}_i$ we have $p'(a_i^j) = b_j = f'(a_i^j)$ by definition of $\vec{a}_i$. On the other hand, for any element $a \in \mathrm{dom}(p') \setminus \{a_i^1, \ldots, a_i^r\}$ we have $p'(a) = p(a) = f(a)$. Furthermore, since the path P_i does not contain any edges in $F_\alpha \cap F_{\alpha'}$, we have $f' \restriction (F_\alpha \cap F_{\alpha'}) \times [2] = f \restriction (F_\alpha \cap F_{\alpha'}) \times [2]$, and since clearly $a \in (F_\alpha \cap F_{\alpha'}) \times [2]$, we see that $f'(a) = f(a)$. Thus, $p'(a) = f'(a)$.

The case where Spoiler plays a right $\mathcal{N}_\ell$-quantifier move is similar. ◀

Note that the vocabulary of the structures $\mathfrak{A}_\ell^{\mathrm{ev}}(G)$ and $\mathfrak{A}_\ell^{\mathrm{od}}(G)$ consists of two ℓ-ary relation symbols. The presence of at least ℓ-ary relations is actually necessary: Duplicator cannot have a winning strategy in $\mathrm{PG}_{\ell-1}^{\mathcal{N}_\ell}$ on structures containing only relations of arity less than ℓ, since by Corollary 10(b), all properties of such structures are definable in $L_{\infty\omega}^{\ell-1}(\mathbf{Q}_{\mathcal{N}_\ell})$.

From Lemma 19, Theorem 28 and Theorem 29, we immediately obtain the result.

▶ **Theorem 30.** *For any $\ell \geq 3$, $\mathsf{CSP}(\mathfrak{C}_\ell)$ is not definable in $L_{\infty\omega}^\omega(\mathbf{Q}_{\mathcal{N}_\ell})$.*

Note that $\mathsf{CSP}(\mathfrak{C}_\ell)$ corresponds to solving systems of linear equations over $\mathbb{Z}/2\mathbb{Z}$ with all equations containing (at most) ℓ variables. Thus, as a corollary we see that solvability of such systems of equations cannot be expressed in $L_{\infty\omega}^\omega(\mathbf{Q}_{\mathcal{N}_\ell})$ for any ℓ. Furthermore, since systems of linear equations over $\mathbb{Z}/2\mathbb{Z}$ can be solved in polynomial time, we see that the complexity class PTIME is not contained in $L_{\infty\omega}^\omega(\mathbf{Q}_{\mathcal{N}_\ell})$ for any ℓ.

Finally, note that since the class $\mathsf{CSP}(\mathfrak{C}_\ell)$ is downwards monotone, by Lemma 9 the quantifier $Q_{\mathsf{CSP}(\mathfrak{C}_\ell)}$ is $\mathcal{N}_{\ell+1}$-closed. Thus, we get the following hierarchy result for the near-unanimity families $\mathcal{N}_\ell$ with respect to the arity ℓ of the partial functions.

▶ **Theorem 31.** *For every $\ell \geq 3$ there is a quantifier in $\mathbf{Q}_{\mathcal{N}_{\ell+1}}$ which is not definable in $L_{\infty\omega}^\omega(\mathbf{Q}_{\mathcal{N}_\ell})$.*

6 Conclusion

We have introduced new methods, in the form of pebble games, for proving inexpressibility in logics extended with generalized quantifiers. There is special interest in proving inexpressibility in logics with quantifiers of unbounded arity. We introduced a general method of defining such collections inspired by the equational theories of polymorphisms arising in the study of constraint satisfaction problems. Perhaps surprisingly, while the collection of CSP that have near-unanimity polymorphisms is rather limited (as they all have bounded width), the collection of quantifiers with the corresponding closure property is much richer, including even CSP that are intractable. The pebble game gives a general method of proving inexpressibility

that works for a wide variety of closure conditions. We were able to deploy it to prove that solvability of systems of equations over $\mathbb{Z}/2\mathbb{Z}$ is not definable using only quantifiers closed under near-unanimity conditions.

It would be interesting to use the pebble games we have defined to show undefinability with other collections of quantifiers closed under partial polymorphisms. Showing some class is not definable with quantifiers closed under partial Maltsev polymorphisms would be especially instructive. It would require using the pebble games with a construction that looks radically different from the CFI-like constructions most often used. This is because CFI constructions encode problems of solvability of equations over finite fields (or more generally finite rings), and all of these problems are Maltsev-closed.

References

1 N. Alon. Tools from higher algebra. In *Handbook of Combinatorics (Vol. 2)*, pages 1749–1783. MIT Press, 1996.

2 A. Atserias, A. Bulatov, and A. Dawar. Affine systems of equations and counting infinitary logic. *Theoretical Computer Science*, 410(18):1666–1683, 2009.

3 L. Barto, A. A. Krokhin, and R. Willard. Polymorphisms, and how to use them. In A. A. Krokhin and S. Zivný, editors, *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*, pages 1–44. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/DFU.Vol7.15301.1.

4 V. G. Bodnarchuk, L. A. Kaluzhnin, V. N. Kotov, and B. A. Romov. Galois theory for post algebras. i. *Cybernetics*, 5(3):243–252, 1969. doi:10.1007/BF01070906.

5 A. A. Bulatov. A dichotomy theorem for nonuniform CSPs. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 319–330, 2017. doi:10.1109/FOCS.2017.37.

6 J-Y. Cai, M. Fürer, and N. Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992.

7 A. Dawar, E. Grädel, and M. Lichter. Limitations of the invertible-map equivalences. *arXiv*, abs/2109.07218, 2021. arXiv:2109.07218.

8 A. Dawar, E. Grädel, and W. Pakusa. Approximations of isomorphism and logics with linear-algebraic operators. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019.*, pages 112:1–112:14, 2019. doi:10.4230/LIPIcs.ICALP.2019.112.

9 A. Dawar, M. Grohe, B. Holm, and B. Laubner. Logics with rank operators. In *Proc. 24th IEEE Symp. on Logic in Computer Science*, pages 113–122, 2009.

10 A. Dawar and B. Holm. Pebble games with algebraic rules. *Fundam. Inform.*, 150(3-4):281–316, 2017.

11 H-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer, 2nd edition, 1999.

12 P. Erdös and H. Sachs. Reguläre Graphen gegebener Taillenweite mit minimaler Knotenzahl. *Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg Math.-Natur. Reihe*, 12(251-257):22, 1963.

13 T. Feder and M.Y. Vardi. Computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory. *SIAM Journal of Computing*, 28:57–104, 1998.

14 David Geiger. Closed systems of functions and predicates. *Pacific Journal of Mathematics*, 27(1):95–100, 1968.

15 E. Grädel and W. Pakusa. Rank logic is dead, long live rank logic! *J. Symb. Log.*, 84:54–87, 2019. doi:10.1017/jsl.2018.33.

16 J. A. Grochow and M. Levet. On the descriptive complexity of groups without abelian normal subgroups. *arXiv*, abs/2209.13725, 2022. doi:10.48550/arXiv.2209.13725.

17 L. Hella. Logical hierarchies in PTIME. *Information and Computation*, 129:1–19, 1996.

18 L. Hella. The expressive power of CSP-quantifiers. In *31st EACSL Annual Conference on Computer Science Logic, CSL*, pages 25:1–25:19, 2023. doi:10.4230/LIPIcs.CSL.2023.25.

19 L. Hella and H. Imhof. Enhancing fixed point logic with cardinality quantifiers. *J. Log. Comput.*, 8(1):71–86, 1998. doi:10.1093/logcom/8.1.71.

20 Peter Jonsson, Victor Lagerkvist, Gustav Nordh, and Bruno Zanuttini. Strong partial clones and the time complexity of SAT problems. *J. Comput. Syst. Sci.*, 84:52–78, 2017. doi:10.1016/J.JCSS.2016.07.008.

21 M. Lichter. Separating rank logic from polynomial time. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS*. IEEE, 2021. doi:10.1109/LICS52264.2021.9470598.

22 B. A. Romov. The algebras of partial functions and their invariants. *Cybernetics*, 17(2):157–167, 1981. doi:10.1007/BF01069627.

23 Henning Schnoor and Ilka Schnoor. Partial polymorphisms and constraint satisfaction problems. In Nadia Creignou, Phokion G. Kolaitis, and Heribert Vollmer, editors, *Complexity of Constraints – An Overview of Current Research Themes [Result of a Dagstuhl Seminar]*, volume 5250 of *Lecture Notes in Computer Science*, pages 229–254. Springer, 2008. doi:10.1007/978-3-540-92800-3_9.

24 D. Zhuk. A proof of the CSP dichotomy conjecture. *J. ACM*, 67:30:1–30:78, 2020. doi:10.1145/3402029.

The Worst-Case Complexity of Symmetric Strategy Improvement

Tom van Dijk
Formal Methods and Tools, University of Twente, The Netherlands

Georg Loho
Discrete Mathematics and Mathematical Programming, University of Twente, The Netherlands

Matthew T. Maat
Discrete Mathematics and Mathematical Programming, University of Twente, The Netherlands

Abstract

Symmetric strategy improvement is an algorithm introduced by Schewe et al. (ICALP 2015) that can be used to solve two-player games on directed graphs such as parity games and mean payoff games. In contrast to the usual well-known strategy improvement algorithm, it iterates over strategies of both players simultaneously. The symmetric version solves the known worst-case examples for strategy improvement quickly, however its worst-case complexity remained open.

We present a class of worst-case examples for symmetric strategy improvement on which this symmetric version also takes exponentially many steps. Remarkably, our examples exhibit this behaviour for any choice of improvement rule, which is in contrast to classical strategy improvement where hard instances are usually hand-crafted for a specific improvement rule. We present a generalized version of symmetric strategy iteration depending less rigidly on the interplay of the strategies of both players. However, it turns out it has the same shortcomings.

2012 ACM Subject Classification Theory of computation → Logic and verification; Mathematics of computing → Discrete mathematics

Keywords and phrases Parity game, Mean payoff game, Symmetric strategy improvement, Strategy improvement, Worst-case complexity

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.24

Related Version *Preprint Version*: `https://doi.org/10.48550/arXiv.2309.02223`

Supplementary Material
Image (Animations): `https://github.com/MatthewMaat/SI-animations/tree/master/Example%20animations`, archived at `swh:1:dir:7eb5b94153b6d6d05acb741110c1dd902cd3dc8c`
Software (Source Code): `https://github.com/trolando/oink`
archived at `swh:1:dir:e20521063a75b985df2eb69595f8f317648fd9de`

1 Introduction

We study certain classes of infinite turn-based games on directed graphs between two players, also called infinitary payoff games, which includes parity games and discounted/mean payoff games. These games are interesting from an algorithmic perspective and from the viewpoint of complexity theory.

First, there are various problems that relate to solving these games. Solving parity games is important for formal verification and synthesis of programs, as many properties of programs are naturally specified by means of fixed points; parity games capture the expressive power of nested least and greatest fixed point operators. In particular, there are linear reductions between parity games and the model checking problem of the modal μ-calculus [11, 35]. Solving mean payoff games is equivalent to problems like solving energy games [4], deciding feasibility in tropical linear programming [1], scheduling with AND/OR precedence constraints [25], and the max-atoms problem [3].

Another notable aspect of these games is their complexity status. It is known that there is a polynomial-time reduction from parity games to mean payoff games, and from mean to discounted payoff games. Many classes of these games are known to be contained in the intersection of NP and coNP [6, 37], and parity games and mean payoff games have been shown to even lie in the intersection of UP and coUP [23]. However, the question whether there exists a polynomial-time algorithm for any of these games has been open for decades.

Related work. Many algorithms have been proposed for solving parity games and mean payoff games with the main algorithm classes being value iteration [19, 10, 14], strategy improvement [2, 34] and attractor-based algorithms [31, 36], where we list only a small part of the many papers. For most of these algorithms there are instances which take exponentially many steps; these are usually simple for value iteration, while work by Van Dijk [33] demonstrates an exponential lower bound to many attractor-based algorithms. Recently, it has been shown that parity games can actually be solved in quasi-polynomial time: after the breakthrough in [5], several other quasi-polynomial algorithms have been found, including [8, 13, 20, 27]. However, most of these approaches are likely to be inherently superpolynomial as demonstrated in [7].

Strategy improvement [2, 12, 26, 28, 29, 34] (also called strategy iteration or policy iteration) is considered to be a viable candidate for a polynomial-time algorithm for many classes of infinitary payoff games due to its inherent combinatorial nature. This method evaluates strategies by means of a function on the nodes in the graph called the *valuation*. It then iteratively makes changes to a strategy, improving the valuation, until an optimal strategy is found. When there are multiple options for improvements, the choice is made by a so-called *improvement rule*. There are a few valuations mainly used in the literature [2, 12, 28, 34]. Based on these, many well-known improvement rules have exponential worst-case running time as demonstrated by sophisticated worst-case constructions, in particular by Friedmann et al. (see e.g. [9, 15, 16]). The main idea behind the worst-case constructions is that one player can "trap" the other player repeatedly in different configurations so that the encountered strategies simulate a binary counter.

While infinitary payoff games are symmetric in the two players by their nature, only some algorithms explicitly exploit this symmetry. Most attractor-based algorithms for parity games are inherently symmetric by simultaneously considering the game from the perspective of both players, while value iteration and strategy improvement algorithms are mostly inherently asymmetric. Recently, a quasipolynomial symmetric algorithm for parity games was proposed by Jurdzinski et al. [21].

Our work is mainly motivated by a symmetric version of strategy iteration for infinitary payoff games established in [30]. This variant maintains two strategies, one for each player. The players then iteratively improve their strategy, using information from the best response to their opponent's strategy. This reduces the number of iterations needed in practice, and also does not have superpolynomial running time on the type of examples that were constructed for classic strategy improvement. The worst-case running time of this variant was unknown so far.

Our contribution. We develop a construction exhibiting exponential running time for symmetric strategy improvement (SSI). Our main result is the following:

▶ **Theorem 1.** *In the worst case, the number of iterations of symmetric strategy improvement in parity games, mean payoff games and discounted payoff games is exponential in the number of nodes and edges in the graph* independently *of the improvement rule. This holds for any of the currently used valuations in the literature.*

It is remarkable that the result holds for any improvement rule. This is different from regular strategy improvement, where the existence of a (theoretical) improvement rule for which the algorithm terminates in a linear number of iterations is guaranteed, see [15, Lemma 4.2]. By our Theorem 1, this does not exist for symmetric strategy iteration.

Moreover, we present a generalization of SSI which uses the valuation directly and not only the strategy of the opponent, allowing for more freedom to potentially overcome the exponential instances. However, we strengthen Theorem 1 with a subtle adapation of the worst-case instances to hold also for the generalization. This suggests that one needs a different approach involving more than only local information to benefit from insights in the interplay of strategies for both players.

Technical overview. To arrive at our main result, we derive a class of games for which SSI needs exponential running time. It is a careful adaptation (depicted in Figure 5) of the basic example from [2, 17] in such a way that the two players are both distracted by the other player's strategy. The key insight here is that the optimal counterstrategy to a bad strategy can also be a bad strategy itself. Hence restricting to moves from the optimal counterstrategy prevents them from making the crucial switches for achieving actual progress.

Our family of games has a self-similar structure. It requires the algorithm to solve a subgame first, and then after the important switches are made, solve the same subgame again, leading eventually to the exponential blowup of the number of iterations.

Recall that symmetric strategy iteration picks only edges of the optimal counter strategy. While this implicitly also uses the valuation, as it is defined via the subgraph arising from the optimal counterstrategy, the generalization directly compares the valuations of the nodes. Only those edges are considered, which provide a local improvement over the valuations of both players.

To derive the lower bound construction for the generalization, we introduce a new gadget (Figure 7). This replaces each of the iteratively arranged pairs of nodes in the former family. The gadget then forces the generalization to exhibit a similar behaviour as the original SSI.

Paper organization. We provide the necessary background on parity games and (symmetric) strategy improvement in Section 2. We introduce generalized symmetric strategy improvement in Section 3. Then, Section 4 presents the structure and iterations of the basic exponential instance for SSI. This is refined in Sections 5 and 6 to the generalized version of SSI and arbitrary improvement rules. We conclude with a discussion of potential extensions and limitations of our construction.

2 Preliminaries

Parity games. A parity game is a game played between two players, called player 0 and player 1. It is played on the nodes of a directed graph $G = (V, E)$, where the nodes are partitioned into $V = V_0 \cup V_1$, and V_i is controlled by player i. We assume every node has at least one outgoing edge. Associated with the nodes of the graph is a priority function $p\colon V \to Q$, where $Q \subset \mathbb{Z}$. At the start of the game, a pebble is placed on one of the nodes. A move is made by the player controlling the node that the pebble is currently on, and it consists of this player choosing an outgoing edge from this node. The pebble moves along the edge to the next node. The players keep making moves indefinitely. The winner of the game is decided by the largest node priority that the pebble encounters infinitely often. If it is even, player 0 wins, and if it is odd, player 1 wins.

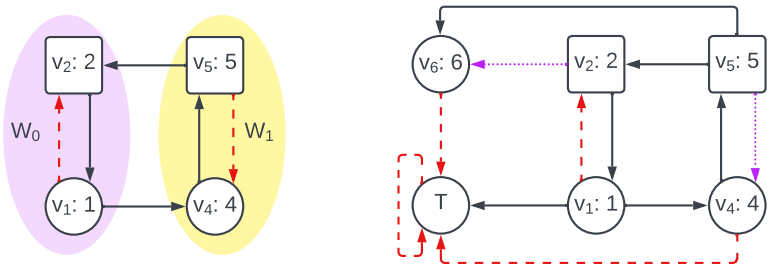


Figure 1 Left: A parity game. Priorities are shown in the nodes. Nodes controlled by player 0 are shown as circles, and nodes controlled by player 1 are squares. The sets of winning starting nodes for player 0 and 1 are W_0 and W_1, and the winning strategies are marked by dashed lines. Right: A sink parity game with a strategy σ (dashed) and its optimal counterstrategy $\bar{\sigma}$ (dotted).

It is well known that parity games are positionally determined, meaning that there always is a player that has a positional winning strategy. Positional means here that one only takes into account on which node the pebble currently is. Therefore, we define a strategy for player 0 as a function $\sigma\colon V_0 \to V$ (with the condition that $(v, \sigma(v)) \in E$ for all nodes $v \in V_0$). Similarly, a player 1 strategy is a function $\tau : V_1 \to V$ (with $(v, \tau(v)) \in E$ for all nodes $v \in V_1$).

Sink parity games. In this paper, we restrict ourselves to a class of parity games called *sink parity games* as it allows to simplify the arguments for valuations. This class has often been used to show lower bounds for parity game algorithms (see e.g. [15, 18]). Solving sink parity games is as hard as solving any parity game, see Lemma 4.

▶ **Definition 2.** *A* sink parity game *is a parity game that fulfills the following conditions:*

1. *There exists a so-called* sink node $\top$ *for which* $p(\top) < p(v)$ *for all nodes in* $V \backslash \{\top\}$*, and whose only outgoing edge is* $(\top, \top)$*.*
2. *There exists a player 0 strategy* σ *such that, when it is played, the highest priority (except* $\top$*) in any possible cycle is even.*
3. *There exists a player 1 strategy* τ *such that, when it is played, the highest priority (except* $\top$*) in any possible cycle is odd.*

We call player 0 and player 1 strategies that satisfy the above conditions *admissible*. If player 0 plays an admissible strategy σ and player 1 plays an admissible strategy τ, then the pebble must end up at $\top$. Otherwise, it would enter the same node outside $\top$ twice, which would create a cycle, and the highest priority of the cycle would have to be odd and even at the same time. This also implies that $\top$ is reachable for the other player when σ or τ is played. One may say that the result of best play in a sink parity game is a "tie".

Valuations of strategies. In the remainder of this section, we describe (symmetric) strategy improvement in parity games. We use the valuation of [12] and [24], adapted to sink parity games. The advantages of using this framework are that it is simpler to explain, and it focuses on the crucial second component of the most commonly used valuation by Jurdzinksi and Vöge [34]. In sink parity games and their related mean/discounted payoff games, the valuation that we describe here is equivalent to the other used versions of strategy improvement for parity games and mean payoff games [2, 28, 34].

Now, suppose player 0 and player 1 both fix a strategy σ and τ, respectively. Then, given a starting node v, the course of the game is fixed. We want to evaluate how "good" this outcome is for player 0. This is expressed in the *play value* $\Theta_{\sigma,\tau}(v)$. If the pebble does not reach $\top$, then it must eventually follow some cycle. If the highest priority in the cycle is even, then this is very good for player 0, so we assign ∞ to this play. If the priority is odd, we assign it value $-\infty$. Otherwise, the pebble follows a path to $\top$. In this case we establish the play value by counting how often each priority on this path is encountered. The even player aims to encounter many high even priorities and little high odd priorities on this path. The following definition formalizes this.

▶ **Definition 3.** *Let σ and τ be a player 0 and player 1 strategy, respectively. Their* play value *is a function $\Theta_{\sigma,\tau} : V \to \mathbb{Z}^Q_{\geq 0} \cup \{-\infty, \infty\}$, with $\mathbb{Z}^Q_{\geq 0}$ the set of nonnegative integer vectors indexed by the priority set Q. It is defined as follows:*

- *Suppose the nodes encountered, starting from node v, are $v_1(= v), v_2, \ldots, v_k, \top, \top, \ldots$ (with $v_k \neq \top$). Then for any $q \in Q$, the q-element of $\Theta_{\sigma,\tau}(v)$ (the component of the vector in $\mathbb{Z}^Q_{\geq 0}$ indexed by q) is given by $(\Theta_{\sigma,\tau}(v))_q = |\{j \leq k : p(v_j) = q\}|$.*
- *If $\top$ is not reached starting from v and player 0 wins, then $\Theta_{\sigma,\tau}(v) = \infty$.*
- *If $\top$ is not reached starting from v and player 1 wins, then $\Theta_{\sigma,\tau}(v) = -\infty$.*

We can compare play values by how "good" they are for player 0. This is done by a total order $\trianglelefteq$ on $\mathbb{Z}^Q_{\geq 0} \cup \{-\infty, \infty\}$. The smallest element for $\trianglelefteq$ is $-\infty$ and the largest is ∞. The order $\trianglelefteq$ compares the play values in $\mathbb{Z}^Q_{\geq 0}$ lexicographically, but different for even and odd indices. To be precise, suppose we have B and C in $\mathbb{Z}^Q_{\geq 0}$, and that $B \neq C$. Let q' be the highest priority q for which $B_{q'} \neq C_{q'}$. We then say that $B \triangleleft C$ if either $B_{q'} < C_{q'}$ and q' is even, or $B_{q'} > C_{q'}$ and q' is odd. So if $B \triangleleft C$, then B has either less of some high even priority or more of some high odd priority, so B is "worse" for player 0. Then we use the concept of play value to evaluate strategies. We evaluate a player 0 admissible strategy σ by an optimal player 1 response $\bar{\sigma}$. To be precise, the valuation of an admissible strategy σ is a function $\Xi_\sigma : V \to \mathbb{Z}^Q_{\geq 0}$ given by: $\Xi_\sigma(v) = \min_{\trianglelefteq}\{\Theta_{\sigma,\tau}(v) \mid \tau \text{ player 1 strategy}\} = \Theta_{\sigma,\bar{\sigma}}(v)$

A strategy τ attaining the minimum in the above equation is an *(optimal) counterstrategy*. Note that there exists a player 1 strategy $\bar{\sigma}$ that simultaneously attains the minimum for all nodes in V (equation (11) in [22]). In general, this strategy might not be unique, but we pick one arbitrarily to be $\bar{\sigma}$ if there are multiple options. Moreover, an optimal response $\bar{\sigma}$ can be computed efficiently. Because we have a sink parity game and an admissible σ, the minimum is never ∞ or $-\infty$, so we can regard the range of Ξ_σ as $\mathbb{Z}^Q_{\geq 0}$.

The valuation of an admissible player 1 strategy τ is defined similarly using an optimal response $\bar{\tau}$ from player 0: $\Xi_\tau(v) = \max_{\trianglelefteq}\{\Theta_{\sigma,\tau}(v) \mid \sigma \text{ player 0 strategy}\} = \Theta_{\bar{\tau},\tau}$.

Figure 1 shows an example of a sink parity game with a strategy and its counterstrategy. For example, we have $\Xi_\sigma(v_2) = (0, 1, 0, 0, 0, 1)$, as the play resulting from σ and its optimal counterstrategy $\bar{\sigma}$ goes through the nodes with priorities 2 and 6. Likewise, $\Xi_\sigma(v_4) = (0, 0, 0, 1, 0, 0)$, hence $\Xi_\sigma(v_2) \triangleright \Xi_\sigma(v_4)$.

Strategy improvement. The core idea behind strategy improvement is to make so-called improving moves. Improving moves for player 0 are given by the edges (v, v') with $v \in V_0$ for which $\Xi_\sigma(v') \triangleright \Xi_\sigma(\sigma(v))$. That means that, with a new strategy picking v' after v, player 0 can send the pebble to a node with higher valuation than it currently does in σ. We denote the set of improving moves for σ by I_σ. Player 0 creates a new strategy σ' from σ by making improving moves, which means $\sigma'(v) = v'$ for a number of improving moves $(v, v') \in I_\sigma$ and $\sigma'(v) = \sigma(v)$ everywhere else. Of course, there may be multiple improving edges per node.

Algorithm 1 Strategy improvement.

1: Start with some admissible strategy σ
2: Find an optimal counterstrategy $\bar{\sigma}$ to σ, compute Ξ_σ and I_σ
3: **if** $I_\sigma = \emptyset$ **then return** σ
4: **else**
5: Let σ' be the strategy obtained from σ by applying all improving moves from $f(I_\sigma)$
6: $\sigma \leftarrow \sigma'$, **go to** 2
7: **end if**

The choice which edges to use to improve is decided by an *improvement rule*, which is a function $f : \mathcal{P}(E) \to \mathcal{P}(E)$ that takes as input a set of improving edges, and outputs a set of improving edges subject to the following conditions:

- If $|S| > 0$, then $|f(S)| > 0$.
- $f(S) \subseteq S$ for all $S \in \mathcal{P}(E)$.
- Every node has at most one outgoing edge in $f(S)$.

By [22, Lemma 5.7], for any choice of improving edges, we have $\Xi_{\sigma'}(v) \trianglerighteq \Xi_\sigma(v)$ for every $v \in V$, and $\Xi_{\sigma'}(v) \triangleright \Xi_\sigma(v)$ for at least one node v. Hence we increase the valuation of the strategy. Clearly, the new strategy is also admissible as its valuation is not $-\infty$. Additionally, if σ has no improving moves, then we know that Ξ_σ is pointwise maximal in the space of valuations ([22, Lemma 5.8]). This leads to the strategy improvement algorithm (Algorithm 1).

We can also define improving moves for player 1, by saying (v, v') with $v \in V_1$ is an improving move if $\Xi_\tau(v') \triangleleft \Xi_\tau(\tau(v))$. We denote the set of improving moves for player 1 by I_τ. Similar to before, if τ' is obtained from τ by making improving moves, $\Xi_{\tau'}(v) \trianglelefteq \Xi_\tau(v)$ for every $v \in V$, and $\Xi_{\tau'}(v) \triangleleft \Xi_\tau(v)$ for at least one node v. It is well-known that, if σ is an optimal player 0 strategy (maximizing Ξ_σ pointwise) and τ an optimal player 1 strategy (minimizing Ξ_τ pointwise), then $\Xi_\sigma = \Xi_\tau$.

Now why are we interested in finding the strategy σ that yields the highest valuation Ξ_σ in a sink parity game? The winner of a sink parity game is already known, since the best both players can do is go to the sink node $\top$. However, even in a sink parity game, finding the optimal strategy (yielding the $\trianglelefteq$-best Ξ_σ) is as difficult as solving parity games, as noted before in [16]. It is similar to the reduction to the longest shortest path problem in [2] and escape games in [29].

▶ **Lemma 4.** *Deciding the winning starting sets and the winning strategies in a parity game can be polynomial-time reduced to finding a player 0 strategy σ in a sink parity game that maximizes Ξ_σ.*

Proof. Suppose we have a parity game $G = (V = V_0 \cup V_1, E)$ with priority function $p : V \to \mathbb{Z}$. We may assume that there are no cycles within V_0 or within V_1. This is because we can always add nodes with small priorities controlled by player 0 in the middle of cycles in V_1 and vice versa without affecting the outcome of the game. Now we construct a parity game $G' = (V', E')$ from G, by adding two extra nodes, $\top$ and w. We extend p by choosing $p(\top)$ smaller than all other priorities, and taking for $p(w)$ an even number higher than all other priorities. We add an edge from every node in V_0 to $\top$, from every node in V_1 to w, from w to $\top$ and from $\top$ to $\top$.

This is clearly a sink parity game, since player 0 has an admissible strategy by always going to $\top$, and player 1 has an admissible strategy by going to w. Now let σ be an optimal player 0 strategy that maximizes Ξ_σ pointwise, and let $\bar{\sigma}$ be player 1's optimal response.

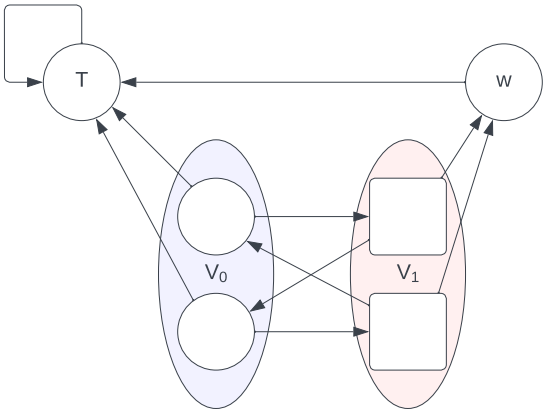


Figure 2 Reduction to a sink parity game.

Since $p(w)$ is even and very large, if player 1 can avoid entering $\top$ through w, they will do so. Define the subgraph G'_σ by the graph with the same node set as G' and with edge set $\{(v, \sigma(v)) : v \in V'_0\} \cup \{(v, v') \in E' : v \in V'_1\}$. Suppose for a node v that $(\Xi_\sigma(v))_{p(w)} = 1$. This implies that in G'_σ, the node $\top$ is only reachable from v through w. In particular, this means that player 1 would always have to end in a cycle in V if they would not have the option of going to w. Because σ is admissible, this cycle has an even highest priority. This implies that, in the original game G, player 0 wins the game that starts at v by playing σ. If, on the other hand, $(\Xi_\sigma(v))_{p(w)} = 0$, this implies that $(\Xi_\tau(v))_{p(w)} = 0$, and we can argue in the same way that player 0 can only reach cycles of odd priority in V if player 1 plays τ. Hence τ wins for player 1 in the parity game G that starts from v. So we found the winning starting sets and winning strategies for both players in G. (Note that we could also have made the above construction with $p(w)$ odd and connecting player 0 nodes to w and player 1 nodes with $\top$.) See also Example 6. ◀

Symmetric strategy improvement. The symmetric strategy improvement (SSI) algorithm was introduced by Schewe et al. in [30] as a symmetric version of strategy improvement. The algorithm maintains and improves two strategies simultaneously: a player 0 strategy σ and a player 1 strategy τ. It uses an optimal counterstrategy $\bar{\tau}$ (by player 0) to τ to select improving moves for σ, and an optimal counterstrategy $\bar{\sigma}$ (by player 1) to σ to select improvements for τ. Note that this could be applied to a broader class of games than just (sink) parity games, in particular mean and discounted payoff games. It is described in Algorithm 2 where we expicitly include the choice of an improvement rule $f \colon \mathcal{P}(E) \to \mathcal{P}(E)$.

It is clear that the algorithm terminates, since any improving move improves the valuation for the respective player, and there is only a finite number of strategies (having a fixed valuation) for both players. The following lemma implies that the algorithm only terminates when the resulting pair of strategies (σ, τ) is optimal for the players. It is implicitly proven in [30, Lemma 3].

▶ **Lemma 5.** *Suppose σ is a non-optimal player 0 strategy or τ is a non-optimal player 1 strategy. Let $\bar{\sigma}$ and $\bar{\tau}$ be optimal counterstrategies to σ and τ, respectively. Then at least one of the sets $I_\sigma \cap \{(v, \bar{\tau}(v)) | v \in V_0\}$ and $I_\tau \cap \{(v, \bar{\sigma}(v)) | v \in V_1\}$ is nonempty.*

Algorithm 2 Symmetric strategy improvement.

1: Start with some pair of admissible strategies σ,τ
2: Find counterstrategies $\bar{\sigma}$ and $\bar{\tau}$ and compute I_σ and I_τ
3: $I \leftarrow f\left((I_\sigma \cap \{(v, \bar{\tau}(v)) | v \in V_0\}) \cup (I_\tau \cap \{(v, \bar{\sigma}(v)) | v \in V_1\})\right)$
4: Let σ', τ' be result of applying all improving moves from I to σ, τ
5: **if** $\sigma = \sigma'$ **and** $\tau = \tau'$ **then return** σ, τ
6: **else**
7: $\quad \sigma \leftarrow \sigma'$, $\tau \leftarrow \tau'$, **go to** 2
8: **end if**

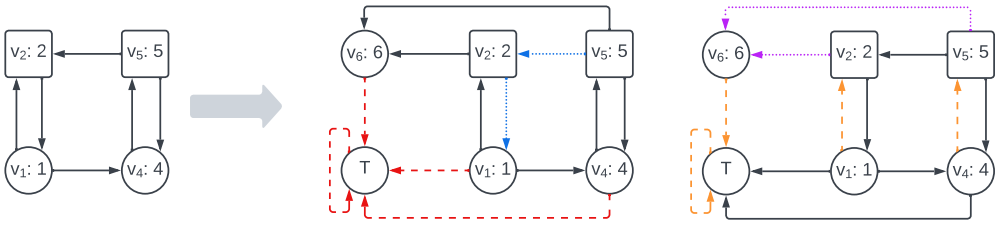


Figure 3 Left: a parity game. Middle: strategy σ (dashed) and counterstrategy $\bar{\sigma}$ (dotted) in the resulting sink parity game. Right: Strategy τ (dashed) and counterstrategy $\bar{\tau}$ (dotted).

▶ **Example 6.** We illustrate the reduction to a sink parity game and the symmetric strategy improvement algorithm with an example. Suppose we have a parity game as on the left in Figure 3. This can be transformed into a sink parity game as in Lemma 4. Two copies of the resulting sink parity game are shown on the right in Figure 3. One copy shows a trivial admissible player 0 strategy σ, with its optimal response $\bar{\sigma}$. The other one shows an admissible τ with its optimal response $\bar{\tau}$.

Now we look at what symmetric strategy improvement does if we start with the pair of strategies shown in Figure 3. We denote the valuations Ξ_σ and Ξ_τ in this game by $(a_1, a_2, a_4, a_5, a_6)$, where a_i stands for $(\Xi_\sigma)_i$ or $(\Xi_\tau)_i$. Now we find the improving moves. Player 0 has two improving moves: the first is (v_1, v_2), as
$\Xi_\sigma(v_2) = (1, 1, 0, 0, 0) \triangleright \Xi_\sigma(\sigma(v_1)) = \Xi_\sigma(\top) = (0, 0, 0, 0, 0)$;
and the other improving move is (v_1, v_4). Since $\bar{\tau}(v_1) = v_2$, the symmetric strategy improvement algorithm will switch player 0's choice in v_1 to v_2. Player 1 has one improving move, namely (v_5, v_4), as $\Xi_\tau(v_4) = (0, 0, 1, 1, 1) \triangleleft (0, 0, 0, 0, 1) = \Xi_\tau(v_6)$. However, $\bar{\sigma}(v_5) = v_2$, so $I_\tau \cap \{(v, \bar{\sigma}(v)) | v \in V_0\}$ is empty, and we do not change τ in this iteration. So in the first iteration, the only switch that is made is changing $\sigma(v_1)$ to v_2. The reader can verify that in the second iteration of the symmetric strategy improvement algorithm, only the switch (v_5, v_4) is made for player 1, and that after that, the algorithm terminates. Then, we have a pair of optimal strategies σ and τ as shown on the left in Figure 4. We can now infer the winning sets W_0 and W_1 of the original game from the 6-element of the valuations of the nodes. Also, the optimal strategies in the sink game form winning strategies in the original game.

3 Generalized symmetric strategy improvement

It was an intriguing insight by Schewe et al. how the interplay between strategies of both players can be used to overcome the known hard instances of strategy improvement. Since the evaluation of the goodness of a strategy relies on the valuation, we go one step further

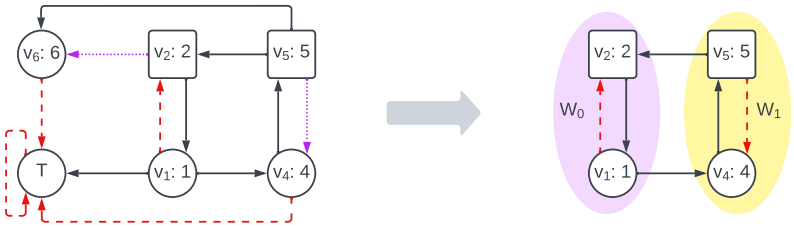


Figure 4 Left: optimal strategies σ (dashed) and τ (dotted), found after performing symmetric strategy improvement. Right: the resulting winning sets and winning strategies (dashed) in the original game.

Algorithm 3 Generalized symmetric strategy improvement.

1: Start with some admissible strategies σ and τ
2: Find I_σ, I_τ, $J_\sigma(\tau)$ and $J_\tau(\sigma)$
3: $I \leftarrow f\left((I_\sigma \cap J_\sigma(\tau)) \cup (I_\tau \cap J_\tau(\sigma))\right)$
4: Let σ', τ' be result of applying all improving moves from I to σ, τ
5: **if** $\sigma = \sigma'$ **and** $\tau = \tau'$ **then return** σ, τ
6: **else**
7: $\quad \sigma \leftarrow \sigma'$, $\tau \leftarrow \tau'$, **go to** 2
8: **end if**

and directly use the interplay between the valuations arising from the strategies of the two players to make the improvements. While this can overcome the first basic family of hard instances presented in Section 4, we show in Section 5 that actually both versions can still be forced to take exponentially many steps. In this way, we extend the result of Theorem 1.

Our selection of improving edges contains the original set but is bigger in general. Instead of basing the choices just on optimal counterstrategies, we construct sets $J_\sigma(\tau)$ and $J_\tau(\sigma)$ by directly comparing locally the valuations of adjacent nodes:

$$J_\sigma(\tau) = \{(v, w) : \ v \in V_0 \wedge \Xi_\tau(w) \trianglerighteq \Xi_\tau(\sigma(v))\},$$
$$J_\tau(\sigma) = \{(v, w) : \ v \in V_1 \wedge \Xi_\sigma(w) \trianglelefteq \Xi_\sigma(\tau(v))\}.$$

These notions allow us to state our generalized version in Algorithm 3. They are in some sense the counterparts of I_σ and I_τ, recall that $I_\sigma = \{(v, w) : \ v \in V_0 \wedge \Xi_\sigma(w) \triangleright \Xi_\sigma(\sigma(v))\}$. One obtains the original SSI back in this context by choosing only those improvement rules that select only edges used by $\bar{\sigma}$ and $\bar{\tau}$.

Correctness of generalized symmetric strategy improvement. Like for normal SSI, it is clear that the algorithm terminates. We are left to show that there is always an improving move possible if the pair of strategies is not optimal. We do so by showing that all the improving moves that were possible in SSI are also possible in the generalization. Correctness of Algorithm 3 then follows from Lemma 5.

▶ **Lemma 7.** *For any pair of strategies σ, τ, we have that any edge $(v, \bar{\sigma}(v))$ is in $J_\tau(\sigma)$, and any edge $(v, \bar{\tau}(v))$ is in $J_\sigma(\tau)$.*

Proof. We do so by contradiction. Suppose there is an edge $(v, \bar{\sigma}(v))$ that is not in $J_\tau(\sigma)$. This means by definition of $J_\tau(\sigma)$ that $\Xi_\sigma(\bar{\sigma}(v)) \triangleright \Xi_\sigma(\tau(v))$. Note that the valuation of σ, which is Ξ_σ, is equal to $\Theta_{\sigma\bar{\sigma}}$. Now consider the strategy subgraph $G_\sigma := (V, E_\sigma)$ with

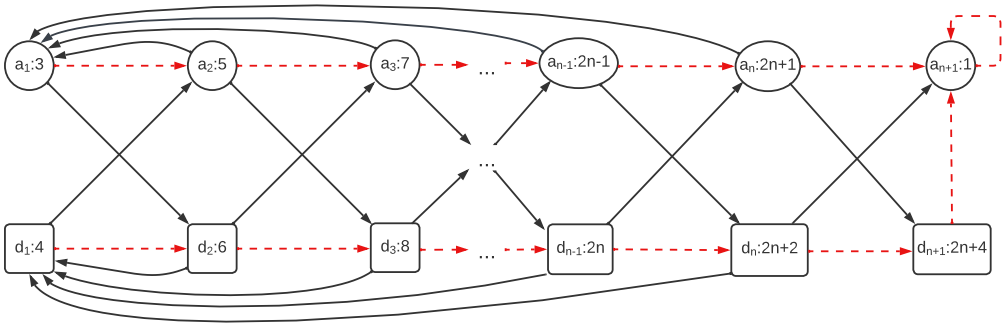


Figure 5 The graph G_n, with the priorities written in the nodes. The initial strategies σ_0 and τ_0 are dashed.

$E_\sigma := \{(v, w) : v \in V_1\} \cup \{(v, \sigma(v)) : v \in V_0\})$. In G_σ, the valuation of $\bar{\sigma}$ is equal to $\Theta_{\sigma\bar{\sigma}} = \Xi_\sigma$, as there is only one player 0 strategy possible. But then $\Xi_\sigma(\bar{\sigma}(v)) \triangleright \Xi_\sigma(\tau(v))$ implies that $(v, \tau(v))$ is an improving move for player 1 strategy $\bar{\sigma}$ in the game G_σ. However, this is not possible, since $\bar{\sigma}$ is defined to be an optimal counterstrategy to σ. We conclude that $(v, \bar{\sigma}(v)) \in J_\tau(\sigma)$. The proof of $(v, \bar{\tau}(v)) \in J_\sigma(\tau)$ is analogous. ◀

4 Counterexample for worst-case complexity

We present a family of parity games for which the original SSI needs exponentially many iterations, where we restrict to the SWITCH-ALL improvement rule at first. This rule selects one improving edge for every node that has an outgoing improving edge in S. We generalize this to arbitrary improvement rules in Section 6. Some animations of the iterations of the examples presented in Sections 4 and 5 can be found at `https://github.com/MatthewMaat/SI-animations/tree/master/Example%20animations`

We use a sequence of sink parity games $(G_n)_{n\in\mathbb{N}}$ whose nodes and edges of G_n are listed in Table 1 and Figure 5. The main structure is similar to the mean payoff game from [2] and [17], with the main difference being that we have backward edges to the "start" of the game.

Initial strategies We define σ_0 and τ_0 by $\sigma_0(a_i) = a_{i+1}$ and $\tau_0(d_i) = d_{i+1}$ for $i = 1, 2, \ldots, n$, $\sigma_0(a_{n+1}) = a_{n+1}$ and $\tau_0(d_{n+1}) = a_{n+1}$.

G_n is a sink parity game. The node a_{n+1} is the sink node with low priority. Also, σ_0 is an admissible player 0 strategy. If the pebble enters any node a_i if σ_0 is played, it will end at the sink. The only way player 1 could try to avoid this is by trying to keep the pebble within $d_1, d_2, \ldots, d_n$, but then this will create a cycle with even highest priority. Hence σ_0 is admissible. Likewise, τ_0 is admissible, so we have a sink parity game.

Table 1 Nodes and edges of G_n.

Node	Player	Priority	Successors
a_1	Player 0	3	a_2, d_2
a_i, $i = 2, \ldots, n$	Player 0	$2i + 1$	a_1, a_{i+1}, d_{i+1}
d_1	Player 1	4	a_2, d_2
d_i, $i = 2, \ldots, n$	Player 1	$2i + 2$	d_1, a_{i+1}, d_{i+1}
a_{n+1}	Player 0	1	a_{n+1}
d_{n+1}	Player 1	$2n + 4$	a_{n+1}

Optimal strategies of the players. Recall that a strategy having a high valuation (for player 0) corresponds to being able to pass through node with high even priorities and avoiding nodes with high odd priorities. Therefore, to maximize their valuation, player 0 lets the pebble pass d_{n+1}, which has the largest (and even) priority in the game. Playing a strategy σ that achieves this when starting from a node v yields $(\Xi_\sigma(v))_{2n+4} = 1$. This means the valuation is larger than any strategy that does not achieve this. There is only one strategy where player 0 can do this from every starting node a_i, namely the strategy with $\sigma(a_i) = a_{i+1}$ for $i < n$ and $\sigma_{a_n} = d_{n+1}$. Ironically, this differs from σ_0 in only one edge. However, as we will see, we avoid making this switch for a long time. Likewise, player 1's goal is to avoid d_{n+1}. Their only strategy to avoid it from every node d_i is to pick $\tau(d_i) = d_{i+1}$ for $i < n$, and $\tau(d_n) = a_{n+1}$. Again, player 1 is only one switch from optimal when starting with τ_0.

The remainder of this section is dedicated to the proof of the following proposition, where we consider iterations in 5 phases. In Section 6, we discuss how our main result follows from this.

▶ **Proposition 8.** *Suppose SSI with the SWITCH-ALL rule on the game graph G_n starts with the strategy pair σ_0, τ_0. Then after $2^{n+1} - 3$ iterations, the optimal strategies σ and τ are found. The optimal strategies for both players do not appear in any earlier iteration.*

Proof. The reader can verify that one needs only one iteration to reach the optimum in G_1. We assume the claim to be true for G_{j-1} and show that it holds for G_j to conclude the proof by induction. We do this by showing that the iterations of SSI are as follows:

1. $2^j - 3$ iterations with switches from $a_1, a_2, \ldots, a_{j-1}$ and $d_1, d_2, \ldots, d_{j-1}$
2. One iteration where the only switch made is (a_j, a_1)
3. One iteration where the only switch made is (d_j, a_{j+1})
4. One iteration where the only switch made is (a_j, d_{j+1})
5. $2^j - 3$ iterations with switches from $a_1, a_2, \ldots, a_{j-1}$ and $d_1, d_2, \ldots, d_{j-1}$

Then clearly the total number of iterations is $2 \cdot (2^j - 3) + 3 = 2^{j+1} - 3$. Now, we elaborate on these 5 steps, where the first $2^j - 3$ iterations need extra insights captured in the following three observations.

▶ **Observation 9.** *As long as $\sigma(a_j) = a_{j+1}$, we have $\bar{\sigma}(d_j) = d_1$. Likewise, as long as $\tau(d_j) = d_{j+1}$, we have $\bar{\tau}(a_j) = a_1$*

Proof. Suppose $\sigma(a_j) = a_{j+1}$, and we look at what $\bar{\sigma}$ could be. Recall that player 1's goal is to reach a_{j+1} without passing d_{j+1}. Starting from d_j, player 1 can do so in two ways: by picking $\bar{\sigma}(d_j) = a_{j+1}$, or by picking $\bar{\sigma}(d_j) = d_1$ and then choosing $\bar{\sigma}$ for $d_1, d_2, \ldots, d_{j-1}$ such that the pebble ends up at a_j. The first option gives $(\Xi_\sigma(d_j))_{2j+1} = 0$ while the second one gives $(\Xi_\sigma(d_j))_{2j+1} = 1$. Hence, the latter gives a lower valuation, and $\bar{\sigma}(d_j) = d_1$. We can prove similarly that while $\tau(d_j) = d_{j+1}$, we have $\bar{\tau}(a_j) = a_1$. ◀

▶ **Observation 10.** *Suppose $\sigma(a_j) = a_{j+1}$ and $\tau(d_j) = d_{j+1}$. Then $\Xi_\sigma(a_j) \lhd \Xi_\sigma(d_j)$ and $\Xi_\tau(a_j) \lhd \Xi_\tau(d_j)$. In both cases, the valuation vectors compared differ in their q-position where $q \geq 2j + 1$.*

Proof. From the proof of Observation 9 we know that when the pebble starts at d_j and players play $\sigma, \bar{\sigma}$, the pebble goes to d_1, then to a_j and directly to a_{j+1}. Hence $\Xi_\sigma(a_j)$ and $\Xi_\sigma(d_j)$ differ in their $p(d_j) = 2j + 2$-component, where the latter valuation has a 1. Then it follows that $\Xi_\sigma(a_j) \lhd \Xi_\sigma(d_j)$, as the values in the vector corresponding to smaller priorities are insignificant when comparing these two valuations. We can likewise see that $\Xi_\tau(a_j) \lhd \Xi_\tau(d_j)$ because they differ in their $p(a_j) = 2j + 1$-component. ◀

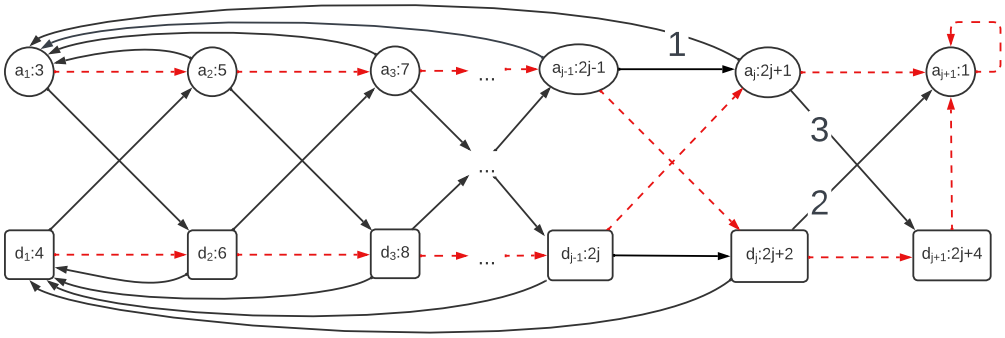


Figure 6 The strategies σ, τ (dashed) after $2^j - 3$ iterations of SSI. Edges that are switched in the next 3 iterations are marked with 1,2,3.

▶ **Observation 11.** *Suppose $\sigma(a_j) = a_{j+1}$ and $\tau(d_j) = d_{j+1}$. Then edge (a_j, a_1) is improving only if $\sigma(a_i) = a_{i+1}$ for $i < j - 1$ and $\sigma(a_{j-1}) = d_j$. Edge (d_j, d_1) is never improving.*

Proof. Suppose player 0 switches the improving edge (a_j, a_1) while σ is different than specified above. The resulting strategy should also be admissible. However, player 1 can play strategy $\bar{\sigma}(d_i) = d_{i+1}$ for $i < j - 1$ and $\bar{\sigma}(d_{j-1}) = a_j$, creating a cycle with highest priority $p(a_j) = 2j + 1$. So the new strategy is not admissible, so (a_j, a_1) could not have been improving. Likewise, suppose (d_j, d_1) is improving for some strategy τ and we switch it. Then player 0 can play $\bar{\tau}(a_i) = a_{i+1}$ for $i < j - 1$ and $\bar{\tau}(a_{j-1}) = d_j$. This either creates a cycle with highest priority $p(d_j) = 2n + 2$ or player 1 creates another cycle within $d_1, d_2, \ldots, d_{j-1}$. In both cases, the resulting player 1 strategy cannot be admissible, so the edge could not have been improving. ◀

From Observations 9 and 11, we can conclude that the strategies at a_j and d_j will not change until σ fulfills the conditon of Observation 11 ($\sigma(a_i) = a_{i+1}$ for $i < j - 1$ and $\sigma(a_{j-1}) = d_j$). Moreover, from Observation 10, we notice that at this first part of the algorithm we may as well remove a_{j+1} and d_{j+1}, and set $p(a_j) = 1$ and add edges (d_j, a_j), (a_j, a_j) without changing any choices of SSI. But now we are left with an exact copy of G_{j-1}, so by induction hypothesis, we know that SSI needs $2^j - 3$ iterations to reach the optimal pair of strategies there. The optimal player 0 strategy in G_{j-1} is $\sigma(a_i) = a_{i+1}$ for $i < j - 1$ and $\sigma(a_{j-1}) = d_j$, which we only reach after $2^j - 3$ iterations. So only then do we need to consider the original graph G_j again and are we able to switch edge (a_j, a_1) by Observation 11.

The $(2^j - 2)$-th iteration There are no improving moves in $a_1, \ldots, a_{j-1}$ or in $d_1, \ldots, d_{j-1}$, as the strategies are 'optimal' strategies in G_{j-1}. However, we do not have optimal strategies in G_j yet, so by Lemma 5, there must be at least one edge switched by SSI. The only choice left is edge (a_j, a_1), so this edge is now switched.

The $(2^j - 1)$-th iteration We have $\sigma = \bar{\tau}$ (recall $\bar{\tau}$ from Observation 9). Since τ is the same as in the last iteration, the nodes $d_1, \ldots, d_{j-1}$ do not have improving moves. Only the edge (d_j, a_{j+1}) can be switched. Since the strategies are not optimal yet, again Lemma 5 implies that we switch exactly this edge in the $(2^j - 1)$-th iteration.

The 2^j-th iteration At the start of the 2^j-th iteration, we observe that player 1's strategy τ is equal to the counterstrategy $\bar{\sigma}$. This is since player 1 can only reach the sink (avoiding d_{j+1}) by playing $\bar{\sigma}(d_j) = a_{j+1}$, and only with this current strategy can player 1 additionally

pass node a_j on the way there if player 0 plays σ. Hence, SSI does not make any improving moves for player 1. If player 0 makes any switch in the nodes $a_1, a_2, \ldots, a_{j-1}$, then this always allows player 1 to create a cycle with odd highest priority, so this cannot be an improving move. Therefore, SSI can only make a switch in a_j. The only improving move in a_j is (a_j, d_{j+1}), so exactly this edge is switched in iteration 2^j.

The final $2^j - 3$ iterations Notice that we will never again make switches in nodes a_j and d_j. Moreover, $\Xi_\sigma(a_j) \triangleright \Xi_\sigma(d_j)$ and $\Xi_\tau(a_j) \triangleright \Xi_\tau(d_j)$. We might as well replace d_j by a sink node of priority 1, and a_j by a node with priority $2j+2$ with an edge to the sink (and remove a_{j+1}, d_{j+1}). This is without changing any future iterations of SSI. But then we have again a copy of G_{j-1} with its respective starting strategies. Using the induction hypothesis again, SSI takes another $2^j - 3$ iterations. It reaches the optimal strategies only in the last iteration. These strategies are also optimal in G_j, and this completes the proof of Proposition 8. ◀

5 Adapted counterexample for generalization of symmetric strategy improvement

We consider the worst-case performance of the generalized version of SSI, when the SWITCH-ALL rule is used. The generalized version can solve the games from the previous section quickly, as it considers more improving moves. However, in this section we show that the generalized version still has exponential running time on a suitably modified version of the counterexample.

The overall structure is the same as for the original counterexample, but the nodes a_i and b_i are replaced by gadgets as shown in Figure 7. The full construction is described in Table 2. The nodes a_i and d_i (except the sink) have priorities larger than N and the other nodes have priority smaller than N, so the priorities of a_i and d_i are still the most important ones.

The function of these gadgets is to make it harder for the players to switch. Now for example, instead of just making a switch at a_i, player 0 has to switch their choice at both c_i and m_i to significantly change the course of the game. Additionally, the nodes e_i, f_i, k_i, l_i make improving moves seem very insignificant with respect to differences in valuation. Similar to Proposition 8, the following proposition holds:

▶ **Proposition 12.** *Suppose generalized SSI on the game graph G_n starts with σ_0, τ_0. Then after $7 \cdot 2^{n-1} - 5$ iterations, the optimal strategies σ and τ are found. The optimal strategies for both players do not appear in any earlier iteration.*

Table 2 Nodes, edges and initial strategies of the adapted counterexample. We have $N = 16n+16$, and i ranges from 1 to n. Nodes between brackets mean that they are only a successor if $i > 1$.

Node	Player	Priority	Successors	Node	Player	Priority	Successors
a_i	Player 0	$N + 2i - 1$	c_i	a_{n+1}	Player 0	1	a_{n+1}
d_i	Player 1	$N + 2i$	h_i	d_{n+1}	Player 1	$N + 2n + 2$	a_{n+1}
c_i	Player 0	$14i + 1$	$e_i, m_i, (a_1)$	e_i	Player 1	$14i + 4$	m_i, a_{i+1}
m_i	Player 0	$14i + 3$	$f_i, c_i, (a_1)$	f_i	Player 1	$14i + 6$	c_i, d_{i+1}
g_i	Player 1	$14i + 8$	$k_i, h_i, (d_1)$	k_i	Player 0	$14i + 11$	h_i, a_{i+1}
h_i	Player 1	$14i + 10$	$l_i, g_i, (d_1)$	l_i	Player 0	$14i + 13$	g_i, d_{i+1}

$$\sigma_0(a_i) = c_i \quad \tau_0(d_i) = h_i \quad \sigma_0(a_{n+1}) = a_{n+1} \quad \tau_0(d_{n+1}) = a_{n+1}$$
$$\sigma_0(c_i) = e_i \quad \sigma_0(m_i) = c_i \quad \tau_0(g_i) = h_i \quad \tau_0(h_i) = l_i$$
$$\tau_0(e_i) = a_{i+1} \quad \tau_0(f_i) = d_{i+1} \quad \sigma_0(k_i) = a_{i+1} \quad \sigma_0(l_i) = d_{i+1}$$

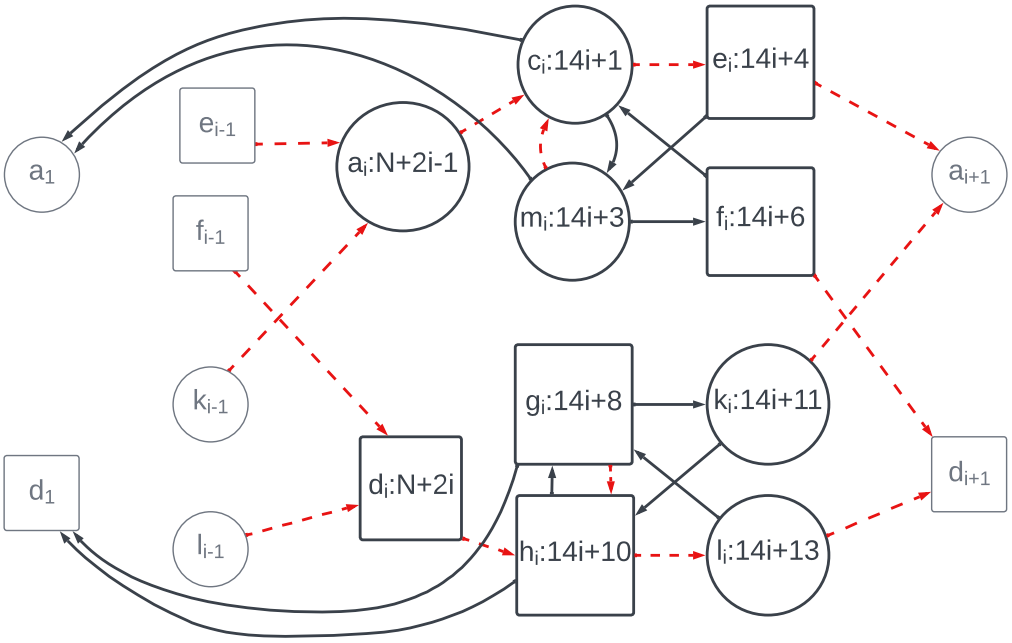


Figure 7 Subgraph for generalized SSI that replaces a_i and b_i when $i > 1$. Initial strategies σ_0 and τ_0 are dashed. Nodes that do not have all their incident edges shown are small and grey.

Proof. We show this by induction similar to before. For $n = 1$, the reader can verify that in the first iteration we switch (m_1, f_1) and (g_1, k_1), and in the second iteration we switch (h_1, g_1) and (c_1, m_1) to reach the optimum. For the induction step, we assume the lemma holds for G_{j-1}, and show that the iterations of generalized SSI on G_j are as follows:

1. $7 \cdot 2^{j-1} - 5$ iterations with switches in nodes x_i with index $i < j$
2. One iteration where only (c_j, a_1) and (m_j, a_1) are switched
3. One iteration where only (g_j, k_j) is switched
4. One iteration where only (h_j, g_j) is switched
5. One iteration where only (m_j, f_j) is switched
6. One iteration where only (c_j, m_j) is switched
7. $7 \cdot 2^{j-1} - 5$ iterations with switches in nodes x_i with index $i < j$

Similar to the proof of Proposition 8, we can argue about the phase of the algorithm where no switches are yet made in the nodes c_j, m_j, g_j and l_j. We can observe that edges (c_j, a_1) and (m_j, a_1) are part of $\bar{\tau}$ but not yet improving until player 0 plays a specific strategy, and (g_j, d_1) and (h_j, d_1) are part of $\bar{\sigma}$ and never improving. So in the first phase, the only improving moves with index j are (m_j, f_j) and (g_j, k_j). But we do not make these moves for a reason that is similar to Observation 9. If we consider the path of the pebble when τ and $\bar{\tau}$ are played, then starting from f_j, we go immediately to d_{j+1}, while from c_j we go back to a_1 and through d_j to d_{j+1}. So $\Xi_\tau(c_j)$ is much bigger (because of the $p(d_j) = N + 2i$) than $\Xi_\tau(f_j)$. Hence generalized SSI never makes the switch (m_j, f_j) in the beginning. Likewise, the move (g_j, k_j) is postponed by the algorithm. Therefore, we can again pretend that a_j is the sink and d_j only has an edge towards the sink and use the induction hypothesis to show that this first phase takes $7 \cdot 2^{j-1} - 5$ iterations.

Now going back to the induction proof, the next five iterations of the algorithm are similar to the three iterations in the middle of the counterexample for SSI, except that we need two switches per module instead of one.

Table 3 Number of iterations of the implementation of symmetric strategy improvement in Oink [32] for the counterexample of Figure 7.

n	1	2	3	4	5	6	7	8	9	10
number of iterations G_n	4	10	11	16	25	44	81	156	305	604

For the last part, we can again pretend that d_j is the sink and a_j a node with even priority with an edge to the sink. This yields a copy of G_{j-1}, but with nodes like c_{j-1} and e_{j-1} and (f_{j-1}) switched. This, however, does not affect any choices of the algorithm (as the valuations of c_i and e_i are always very close). Then using the induction hypothesis, it follows that this last phase takes $7 \cdot 2^{j-1} - 5$ iterations. This completes the induction proof. ◀

An efficient implementation of SSI and of the worst-case example presented here can be found in Oink [32]. The implementation of SSI in Oink is slightly different from the original algorithm, as it contains optimizations that are not present in the original algorithm, but the worst-case example presented here works nonetheless. See Table 3. It is clear that with increasing n, the number of iterations roughly doubles. To obtain these numbers, we used the command line `counter_symsi n | oink --ssi`. The numbers are the same with the preprocessor `--scc` for SCC decomposition.

6 Concluding the proof

We look at the last details to prove Theorem 1. So far, we assumed that the improvement rule SWITCH-ALL is used. It turns out that in many iterations of (generalized) SSI, there is only one improving move. This implies the following result.

▶ **Lemma 13.** *The results of Propositions 8 and 12 hold for any improvement rule f.*

Proof. We prove this by arguing that if we use SWITCH-ALL on the graphs G_n for both constructions, then either there is only one switch possible, or every switch but one switch is irrelevant. Here irrelevant means that whether or not we make the switch, the further course of the algorithm does not change, hence the number of iterations can only increase if we decide not to make some improving switches.

First, we consider the class of games G_n that proved Proposition 8 (Figure 5). If we use the SWITCH-ALL rule, then by following the induction proof we find that we make two switches per iteration whenever we make switches at a_1 and d_1 (following from the induction basis), and we make one switch in every other case (from the induction step). Note however, that not making a switch in d_1 never has any effect on the course of the algorithm. Observations 9, 10, 11 still hold if no switch is made in d_1, and so does the rest of the induction step. It also follows that if only d_1 is switched and a_1 not, then in the next iteration a_1 has the only possible switch. In all cases, symmetric strategy improvement takes at least as many iterations as for SWITCH-ALL.

Next we consider the games G_n from the proof of Proposition 12 (Figure 7). There are two cases where there are multiple switches when SWITCH-ALL is used. First, the modules attached to a_1 and d_1 are switched at the same time. However, for the same reason as above, we can ignore the second module. Secondly, (c_i, a_1) and (m_i, a_1) are switched at the same time. Note that if (c_i, a_1) is switched, then any switch in m_i becomes irrelevant as m_i is not reachable at that moment for player 0, and we make the switch (m_i, f_i) three iterations later. If, on the other hand, only (m_i, a_1) is switched while $\sigma(c_i) = e_i$, then nothing significant changes in the game and in the next iteration we will still have to switch (c_i, a_1). Finally,

if (m_i, a_1) is switched while $\sigma(c_i) = m_i$ and $\sigma(m_i) = f_i$, then the switch (c_i, a_1) becomes irrelevant after the switch as it only removes one $14i + 3$-priority from the valuation. So in all cases, there is only one relevant switch. We conclude that using an improvement rule for this G_n takes at least as many iterations as for SWITCH-ALL. ◀

Note that the number of nodes and edges of G_n for regular SSI is $2n+2$ and $6n$, respectively. Also, the running time of SSI cannot be more than the total number of strategies for both players, which is exponential in the number of nodes and edges. Hence the worst-case running time of symmetric strategy improvement for parity games is exponential in the number of nodes and edges. Moreover, we already noted that in sink parity games, the valuations from the literature are equivalent. Furthermore, the result for mean and discounted payoff games can be shown analogously to [15, Theorem 4.19]. This concludes the proof of Theorem 1.

Finally, the number of nodes and edges of our game G_n for generalized SSI is also linear in n. This yields the following result.

▶ **Theorem 14.** *The results from Theorem 1 also hold for generalized SSI.*

Note on further increasing the number of switches

In the adapted counterexample, generalized symmetric strategy improvement still makes "too few" switches, because it puts off making some good switches. That opens the question if one could further increase the number of switches that generalized symmetric strategy improvement makes. A next logical change would be to always make an improving move in a node if one can, and use the opponent's strategy only for deciding which improving move to make. However, in a run of regular strategy improvement on the switch-best counterexample from [15], there is always exactly one improving move per node. [1] The main idea behind Friedmann's example is that the nodes representing significant bits are distracted by unnecessary switches, resulting in the insignificant bits always being switched first, which creates a binary counter-like behaviour. It follows that the just mentioned generalization would behave the same on this game for player 0 as regular strategy improvement with the SWITCH-ALL rule. Hence there would still be an exponential-time worst-case example. This time, one could say that it is because the algorithm is switching "too many" edges.

7 Discussion

We showed that both symmetric strategy improvement (SSI) and a generalization have exponential worst-case time complexity. The reason is that they can make too few crucial switches, as they are distracted by their opponent's (bad) strategy. In our example, the opponent's valuation always steers the players to make "bad" improving moves. Hence, even using the local information of the other player's valuation is not always useful. Remarkably, no improvement rule can fix this issue. Furthermore, we presented a generalization of SSI and its worst-case example which allows for more flexibility.

The parity game example presented in Figure 5 occurs to be trivial to solve for various natural implementations. One could think of preprocessing techniques like SCC decomposition, removing self-loops or choosing an initial strategy with some heuristic (in fact, the

[1] There is one exception in the iteration after their so-called deceleration lane resets, here we can choose to which node in the deceleration lane we go. However, for this generalization, both from the perspective of Ξ_σ and Ξ_τ we get a higher valuation if we switch to the highest even node in the lane like in regular strategy improvement. Hence it would make sense to assume that this always happens.

construction in the proof of Lemma 4 gives such a heuristic that would solve the parity game quickly). There are further tweaks like propagating information about which nodes are won through the graph by attracting towards them. However, we argue that the main principles of our counterexamples are robust against such tricks. Imagine the parity game of Figure 5 to be part of a larger parity game, where a_{n+1} looks much higher valued than d_{n+1} at the beginning. Suppose that in reality d_{n+1} is winning for player 0 (or player 0 has a strategy such that it has a large valuation in the version of SSI used here), and a_{n+1} is winning for player 1 (or player 1 can let it have a low valuation). Then, until SSI discovers the true value of a_{n+1} and d_{n+1}, the best possible strategies to play from $a_1, a_2, \ldots$ and $d_1, d_2, \ldots$ are in fact the strategies we use as initial strategies in Section 4. So most likely, the preprocessing would pick these strategies or we quickly end up with them after some iterations. Secondly, SCC decomposition can be tricked by adding edges back from the rest of the game to a_1, d_1 that are bad choices for the player that can choose them, and the propagation trick is stopped by the modules from Figure 7. It is likely that other optimizations, assuming that they have a weakness, will be stopped by adding modules that exploit this weakness in a similar way. Note, however, that structures like those in our hard instances are unlikely to show up in practice. As observed by the authors of SSI [30], the number of iterations of SSI is low on randomly generated instances.

Our work leaves open if there is actually any way to use the interplay of strategies of the two players not ending up with exponential worst-case running time. Our constructions suggest that the restriction imposed by using local information of the opponent's strategy might always be exploited to lure the iterations into an exponential trap.

References

1 Marianne Akian, Stéphane Gaubert, and Alexander E. Guterman. Tropical polyhedra are equivalent to mean payoff games. *Int. J. Algebra Comput.*, 22(1), 2012. doi:10.1142/S0218196711006674.

2 Henrik Björklund and Sergei G. Vorobyov. A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games. *Discret. Appl. Math.*, 155(2):210–229, 2007. doi:10.1016/j.dam.2006.04.029.

3 Manuel Bodirsky and Marcello Mamino. Tropically convex constraint satisfaction. *Theory Comput. Syst.*, 62(3):481–509, 2018. doi:10.1007/s00224-017-9762-0.

4 L. Brim, J. Chaloupka, L. Doyen, R. Gentilini, and J. F. Raskin. Faster algorithms for mean-payoff games. *Form. Methods Syst. Des.*, 38(2):97–118, April 2011. doi:10.1007/s10703-010-0105-x.

5 Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. Deciding parity games in quasipolynomial time. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 252–263. ACM, 2017. doi:10.1145/3055399.3055409.

6 Anne Condon. The complexity of stochastic games. *Information and Computation*, 96(2):203–224, 1992. doi:10.1016/0890-5401(92)90048-K.

7 Wojciech Czerwinski, Laure Daviaud, Nathanaël Fijalkow, Marcin Jurdzinski, Ranko Lazic, and Pawel Parys. Universal trees grow inside separating automata: Quasi-polynomial lower bounds for parity games. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2333–2349. SIAM, 2019. doi:10.1137/1.9781611975482.142.

8 Daniele Dell'Erba and Sven Schewe. Smaller progress measures and separating automata for parity games. *Frontiers Comput. Sci.*, 4, 2022. doi:10.3389/fcomp.2022.936903.

9 Yann Disser, Oliver Friedmann, and Alexander V. Hopp. An exponential lower bound for zadeh's pivot rule. *Math. Program.*, 199(1):865–936, 2023. doi:10.1007/s10107-022-01848-x.

10 Dani Dorfman, Haim Kaplan, and Uri Zwick. A Faster Deterministic Exponential Time Algorithm for Energy Games and Mean Payoff Games. In *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 114:1–114:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ICALP.2019.114.

11 E. Allen Emerson, Charanjit S. Jutla, and A. Prasad Sistla. On model checking for the μ-calculus and its fragments. *Theor. Comput. Sci.*, 258(1-2):491–522, 2001. doi:10.1016/S0304-3975(00)00034-7.

12 John Fearnley. Efficient parallel strategy improvement for parity games. In *Computer Aided Verification – 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part II*, volume 10427 of *Lecture Notes in Computer Science*, pages 137–154. Springer, 2017. doi:10.1007/978-3-319-63390-9_8.

13 John Fearnley, Sanjay Jain, Bart de Keijzer, Sven Schewe, Frank Stephan, and Dominik Wojtczak. An ordered approach to solving parity games in quasi-polynomial time and quasi-linear space. *Int. J. Softw. Tools Technol. Transf.*, 21(3):325–349, 2019. doi:10.1007/s10009-019-00509-3.

14 Nathanaël Fijalkow, Paweł Gawrychowski, and Pierre Ohlmann. Value Iteration Using Universal Graphs and the Complexity of Mean Payoff Games. In *45th International Symposium on Mathematical Foundations of Computer Science (MFCS 2020)*, volume 170 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 34:1–34:15, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.MFCS.2020.34.

15 Oliver Friedmann. *Exponential Lower Bounds for Solving Infinitary Payoff Games and Linear Programs*. PhD thesis, Ludwig Maximilians University Munich, 2011. URL: http://edoc.ub.uni-muenchen.de/13294/.

16 Oliver Friedmann. A superpolynomial lower bound for strategy iteration based on snare memorization. *Discret. Appl. Math.*, 161(10-11):1317–1337, 2013. doi:10.1016/j.dam.2013.02.007.

17 V. A. Gurvich, A. V. Karzanov, and L. G. Kachivan. Cyclic games and an algorithm to find minimax cycle means in directed graphs. *USSR Computational Mathematics and Mathematical Physics*, 28(5):85–91, 1988. URL: http://www.maths.lse.ac.uk/Personal/stengel/ussrGKK.pdf.

18 Thomas Dueholm Hansen. *Worst-case analysis of strategy iteration and the simplex method*. PhD thesis, Aarhus University, Denmark, 2012. URL: https://pure.au.dk/portal/files/52807524/PhD_dissertation_Thomas_Dueholm_Hansen.pdf.

19 Marcin Jurdzinski. Small progress measures for solving parity games. In *STACS 2000, 17th Annual Symposium on Theoretical Aspects of Computer Science, Lille, France, February 2000, Proceedings*, volume 1770 of *Lecture Notes in Computer Science*, pages 290–301. Springer, 2000. doi:10.1007/3-540-46541-3_24.

20 Marcin Jurdzinski and Ranko Lazic. Succinct progress measures for solving parity games. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–9. IEEE Computer Society, 2017. doi:10.1109/LICS.2017.8005092.

21 Marcin Jurdziński, Rémi Morvan, Pierre Ohlmann, and K. S. Thejaswini. A symmetric attractor-decomposition lifting algorithm for parity games. *arXiv e-prints*, page arXiv:2010.08288, October 2020. doi:10.48550/arXiv.2010.08288.

22 Marcin Jurdzinski and Jens Vöge. A discrete stratety improvement algorithm for solving parity games. *BRICS Report Series*, 7(48), June 2000. doi:10.7146/brics.v7i48.20215.

23 Marcin Jurdziński. Deciding the winner in parity games is in up $\cap$ co-up. *Information Processing Letters*, 68(3):119–124, 1998. doi:10.1016/S0020-0190(98)00150-1.

24 Michael Luttenberger. Strategy iteration using non-deterministic strategies for solving parity games. *CoRR*, abs/0806.2923, 2008. arXiv:0806.2923.

25 Rolf H. Möhring, Martin Skutella, and Frederik Stork. Scheduling with AND/OR precedence constraints. *SIAM J. Comput.*, 33(2):393–415, 2004. doi:10.1137/S009753970037727X.

26 Pierre Ohlmann. *Monotonic Graphs for Parity and Mean-Payoff Games*. PhD thesis, University of Paris, 2021. URL: https://www.irif.fr/~ohlmann/contents/these.pdf.

27 Pawel Parys. Parity games: Zielonka's algorithm in quasi-polynomial time. In *44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019, August 26-30, 2019, Aachen, Germany*, volume 138 of *LIPIcs*, pages 10:1–10:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.MFCS.2019.10.

28 Anuj Puri. Theory of hybrid systems and discrete event systems., 1995. URL: https://www.elibrary.ru/item.asp?id=5408583.

29 Sven Schewe. An optimal strategy improvement algorithm for solving parity and payoff games. In *Computer Science Logic, 22nd International Workshop, CSL 2008, 17th Annual Conference of the EACSL, Bertinoro, Italy, September 16-19, 2008. Proceedings*, volume 5213 of *Lecture Notes in Computer Science*, pages 369–384. Springer, 2008. doi:10.1007/978-3-540-87531-4_27.

30 Sven Schewe, Ashutosh Trivedi, and Thomas Varghese. Symmetric strategy improvement. In *Automata, Languages, and Programming – 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part II*, volume 9135 of *Lecture Notes in Computer Science*, pages 388–400. Springer, 2015. doi:10.1007/978-3-662-47666-6_31.

31 Tom van Dijk. Attracting tangles to solve parity games. In *Computer Aided Verification – 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings, Part II*, volume 10982 of *Lecture Notes in Computer Science*, pages 198–215. Springer, 2018. doi:10.1007/978-3-319-96142-2_14.

32 Tom van Dijk. Oink: An implementation and evaluation of modern parity game solvers. In *Tools and Algorithms for the Construction and Analysis of Systems – 24th International Conference, TACAS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings, Part I*, volume 10805 of *Lecture Notes in Computer Science*, pages 291–308. Springer, 2018. doi:10.1007/978-3-319-89960-2_16.

33 Tom van Dijk. A parity game tale of two counters. In *Proceedings Tenth International Symposium on Games, Automata, Logics, and Formal Verification, GandALF 2019, Bordeaux, France, 2-3rd September 2019*, volume 305 of *EPTCS*, pages 107–122, 2019. doi:10.4204/EPTCS.305.8.

34 Jens Vöge and Marcin Jurdziński. A discrete strategy improvement algorithm for solving parity games. In *Computer Aided Verification*, pages 202–215, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

35 Igor Walukiewicz. Monadic second order logic on tree-like structures. In *STACS 96, 13th Annual Symposium on Theoretical Aspects of Computer Science, Grenoble, France, February 22-24, 1996, Proceedings*, volume 1046 of *Lecture Notes in Computer Science*, pages 401–413. Springer, 1996. doi:10.1007/3-540-60922-9_33.

36 Wieslaw Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theor. Comput. Sci.*, 200(1-2):135–183, 1998. doi:10.1016/S0304-3975(98)00009-7.

37 Uri Zwick and Mike Paterson. The complexity of mean payoff games on graphs. *Theor. Comput. Sci.*, 158(1&2):343–359, 1996. doi:10.1016/0304-3975(95)00188-3.

The Produoidal Algebra of Process Decomposition

Matt Earnshaw
Tallinn University of Technology, Estonia

James Hefford
University of Oxford, UK

Mario Román
Tallinn University of Technology, Estonia

Abstract

We characterize a universal normal produoidal category of monoidal contexts over an arbitrary monoidal category. In the same sense that a monoidal morphism represents a process, a monoidal context represents an incomplete process: a piece of a decomposition, possibly containing missing parts. In particular, symmetric monoidal contexts coincide with monoidal lenses and endow them with a novel universal property. We apply this algebraic structure to the analysis of multi-party protocols in arbitrary theories of processes.

2012 ACM Subject Classification Theory of computation → Categorical semantics

Keywords and phrases monoidal categories, profunctors, lenses, duoidal categories

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.25

Related Version *Full Version*: `https://arxiv.org/abs/2301.11867` [20]

Funding Matt Earnshaw and Mario Román were supported by the European Social Fund Estonian IT Academy research measure (project 2014-2020.4.05.19-0001). James Hefford is supported by University College London and the EPSRC [grant number EP/L015242/1].

Acknowledgements We thank Pawel Sobocinski, Fosco Loregian, Chad Nester and David Spivak for discussion. We thank the CSL reviewers for suggestions that lead to considerable improvements.

1 Introduction

Theories of processes, such as *stochastic*, *partial* or *linear* functions, are a foundational tool in computer science. They help us model interacting systems using a solid mathematical structure. Any theory of processes involving operations for *sequential* and *parallel composition*, satisfying reasonable axioms, forms a *monoidal category*. Monoidal categories are versatile: they can be used in the description of quantum circuits [2], stochastic processes [11, 24], relational queries [9] and non-terminating processes [13], among other applications [16].

At the same time, monoidal categories have two intuitive, sound and complete calculi: the first in terms of *string diagrams* [40], and the second in terms of their *linear type theory* [63]. String diagrams are a 2-dimensional syntax in which processes are represented by boxes, and their inputs and outputs are connected by wires. The type theory of symmetric monoidal categories is the basis of the more specialized *arrow do-notation* used in functional programming languages [37, 51], which becomes *do-notation* for Kleisli categories of commutative monads [46, 28].

This manuscript studies an algebra of context for monoidal categories. Context is of central importance in computer science: we model not only processes but also the environment in which they act. While the algebra of 1-dimensional context is commonplace in applications like parsing [45], the same concept was missing for 2-dimensional syntaxes, which are still less frequent in computer science [21]. Let us showcase monoidal categories, their string diagrams and the use of do-notation in the description of a protocol.

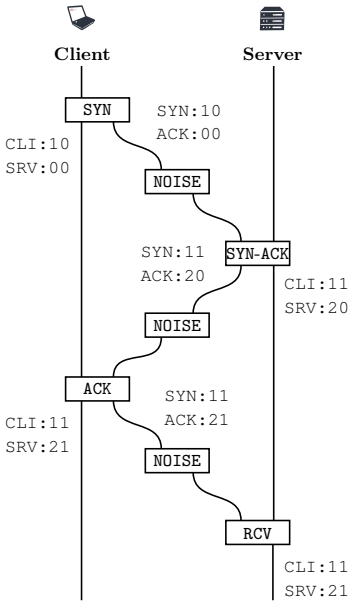


Figure 1 TCP Three way handshake.

1.1 Protocol Description

The Transmission Control Protocol (TCP) is a connection-based communication protocol. Every connection begins with a *three-way handshake*: an exchange of messages that synchronizes the state of both parties. This handshake is defined in RFC793 to have three steps: `SYN`, `SYN-ACK` and `ACK` [55]. The client initiates the communication by sending a synchronization packet (`SYN`) to the server. The synchronization packet contains a pseudorandom number associated to the session, the Initial Sequence Number of the client (`CLI`). The server acknowledges this packet and sends a message (`ACK`) containing its own sequence number (`SRV`) together with the client's sequence number plus one (`CLI`+1). These two form the `SYN-ACK` message. Finally, the client sends a final `ACK` message with the server's sequence number plus one, $\texttt{SRV} + 1$. When the protocol works correctly, both client and server end up with the pair $(\texttt{CLI} + 1, \texttt{SRV} + 1)$.

This protocol is traditionally described in terms of a communication diagram (Figure 1). This diagram can be taken seriously as a formal mathematical object: it is a string diagram describing a *morphism* in a monoidal category. The implementation of each component of the protocol is traditionally written as pseudocode. This pseudocode can also be taken seriously as the expression of a morphism in the same monoidal category, possibly with extra structure: in this case, a commutative *Freyd category* (Figure 2a, see the full version [20, Appendix C.1] [46]). That is, symmetric monoidal categories admit two different internal languages, and we can use both to interpret formally the traditional description of a protocol in terms of string diagrams and pseudocode.

1.2 Types for Message Passing

The last part in formalizing a multi-party protocol in terms of monoidal categories is to actually separate its component parties. For instance, the three-way handshake can be split into the client, the server and a channel. Here is where the existing literature in monoidal

categories seems to fall short: the parts resulting from the decomposition of a monoidal morphism are not necessarily monoidal morphisms themselves (see the full version [20, Figure 14] for the diagrammatic representation). We say that these are only *monoidal contexts*.

Contrary to monoidal morphisms, which only need to declare their input and output types, monoidal contexts need *behavioural types* [54, 38] that specify the order and type of the exchange of information along their boundary. A monoidal context may declare intermediate *send* ($!A$) and *receive* ($?A$) types, separated by a sequencing operator ($\triangleleft$). For instance, the channel is a monoidal morphism just declaring that it takes an input message (**Msg**) and produces another output message; but the client is a monoidal context that transforms its memory type, **Client**, at the same time it *sends*, *receives* and then *sends* a message; and the server transforms its memory type, **Server** while, dually to the client, it *receives*, *sends* and then *receives* a message (Figure 2b).

(a)

```
syn :: (Cli, Srv) ~>
  ((Cli, Srv), (Syn, Ack))
syn(client, server) = do
  client <- random
  return ((cli, 0), (cli, 0))
```

(b)

$$\in \mathcal{LC}\left(\begin{smallmatrix}\textbf{Client}\\ \textbf{Client}\end{smallmatrix}; \textbf{!Msg} \triangleleft \textbf{?Msg} \triangleleft \textbf{!Msg}\right);$$

$$\in \mathcal{LC}\left(\begin{smallmatrix}\textbf{Server}\\ \textbf{Server}\end{smallmatrix}; \textbf{?Msg} \triangleleft \textbf{!Msg} \triangleleft \textbf{?Msg}\right);$$

$$\texttt{NOISE} \in \mathbb{C}\left(\textbf{Msg}; \textbf{Msg}\right);$$

Figure 2 (a) Implementation of `SYN`. (b) Types for the three parties.

Session types [36], including the send ($!A$) and receive ($?A$) polarized types, have been commonplace in logics of message passing. Cockett and Pastro [14] already proposed a categorical semantics for message-passing which, however, needs to go beyond monoidal categories, into *linear actegories* and *polyactegories*.

Our claim is that, perhaps surprisingly, monoidal categories already have the necessary algebraic structure to define *monoidal contexts* and their send-receive polarized types. Latent to any monoidal category, there exists a universal category of contexts with polarized types ($!/?$) and parallel/sequence operators ($\otimes/\triangleleft$).

1.3 Reasoning with Contexts

This manuscript introduces the notion of monoidal context and symmetric monoidal context; and it explains how dinaturality allows us to reason with them. In the same way that we reason with monoidal morphisms using string diagrams, we can reason about monoidal contexts using *incomplete string diagrams* [4, 59].

For instance, consider the following fact about the TCP three-way handshake: the client does not need to store a starting `SRV` number for the server, as it will be overwritten as soon as the real one arrives. This fact only concerns the actions of the client, and it is independent of the server and the channel. We would like to reason about it preserving this modularity, and this is what the incomplete diagrams in Figure 3 achieve.

Here, we define $\texttt{SYN}^* = \texttt{SYN} \fatsemi \texttt{PRJ}$ to be the same as the `SYN` process but projecting out only the client `CLI` number. We also define a new $\texttt{ACK}^*$ that ignores the server `SRV` number, so that $\texttt{ACK} = \texttt{PRJ} \fatsemi \texttt{ACK}^*$. These two equations are enough to complete our reasoning.

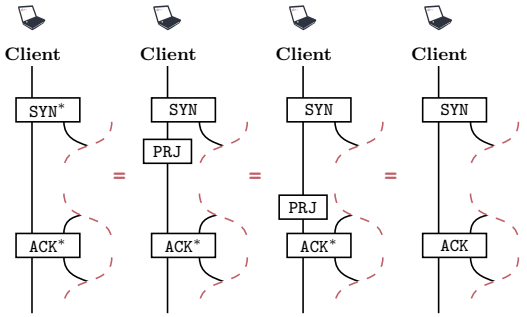


Figure 3 Reasoning only with the client.

Monoidal contexts and their incomplete diagrams are defined to be convenient tuples of morphisms, e.g. (SYN|ACK) in our example; what makes them interesting is the equivalence relation we impose on them: this equivalence relation makes the pair (SYN ⨟ PRJ|ACK*) equal to (SYN|PRJ ⨟ ACK*). *Dinaturality* is the name we give to this relation, and we will see how it arises canonically from the algebra of profunctors.

1.4 The Produoidal Algebra of Monoidal Context

Despite the relative popularity of string diagrams and other forms of formal 2-dimensional syntax, the algebra of incomplete monoidal morphisms has remained obscure. This manuscript elucidates this algebra: we show that, as monoidal morphisms together with their string diagrams form *monoidal categories*, monoidal contexts together with their incomplete string diagrams form *normal produoidal categories*. Normal produoidal categories were a poorly understood categorical structure, for which we provide examples. Let us motivate "normal produoidal categories" by parts.

First, the *"duoidal"* part. Monoidal contexts can be composed sequentially and in parallel, but also nested together to fill the missing parts. Nesting is captured by categorical composition, so we need specific tensors for both sequential ($\lhd$) and parallel ($\otimes$) composition. This is what duoidal categories provide. Duoidal categories are categories with two monoidal structures, e.g. $(\lhd, N)$ and $(\otimes, I)$. These two monoidal structures are in principle independent but, whenever they share the same unit ($I \cong N$), they become well-suited to express process dependence [62]: they become *"normal"*.

Finally, the *"pro-"* prefix. It is not that we want to impose this structure on top of the monoidal one, but we want to capture the structure morphisms already form. The two tensors $(\lhd, \otimes)$ do not necessarily exist in the original category; in technical terms, they are not *representable* or *functorial*, but *virtual* or *profunctorial*. This makes us turn to the produoidal categories of Booker and Street [10, 66].

Not only is all of this algebra present in monoidal contexts. Monoidal contexts are the *canonical* such algebra; in a precise sense given by universal properties. The slogan for the main result of this manuscript (Theorem 5.4) is that

> Monoidal contexts are the *free* normalization of the *cofree* produoidal category over a monoidal category.

1.5 Related Work

Far from being the proposal of yet another paradigm, monoidal contexts form a novel algebraic characterization of a widespread paradigm. We argue that the idea of monoidal contexts has been recurrent in the literature, just never appearing explicitly and formally. Our main contribution is to universally characterize an algebra of monoidal contexts, in the form of a *normal produoidal* category. In fact, recently, there have been multiple implicit applications of monoidal contexts. Kissinger and Uijlen [41] describe higher order quantum processes using contexts with holes in compact closed monoidal categories. Ghani, Hedges, Winschel and Zahn [27] describe economic game theory in terms of *lenses* and incomplete processes in cartesian monoidal categories. Bonchi, Piedeleu, Sobociński and Zanasi [8] study contextual equivalence in their monoidal category of affine signal flow graphs. Di Lavore, de Felice and Román [19] define *monoidal streams* by iterating monoidal context coalgebraically.

Category theory. Street already noted that the endoprofunctors of a monoidal category had a duoidal structure [66]; Pastro and Street described a promonoidal structure on lenses [50] and Garner and López-Franco contributed a partial normalization procedure for duoidal categories [25]. We build on top of this literature, putting it together, spelling out existence proofs, popularizing its produoidal counterpart and providing multiple new results and constructions that were previously missing (e.g. Theorems 3.8, 4.3, and 5.4).

Language theory. Motivated by language theory and the Chomsky-Schützenberger theorem, Melliès and Zeilberger [45] were the first to present the multicategorical *splice-contour* adjunction. We are indebted to their exposition, which we extend to the promonoidal and produoidal cases. Earnshaw and Sobociński [21] described a congruence on formal languages of string diagrams using monoidal contexts. We prove how monoidal contexts arise from an extended produoidal splice-contour adjunction; unifying these two threads.

Session types. Session types [35, 36] are the mainstay type formalism for communication protocols, and they have been extensively applied to the π-calculus [60]. Our approach is not set up to capture all of the features of a fully fledged session type theory [43]. Arguably, this makes it more general in what it does: it always provides a universal way of implementing send ($\mathbf{!}A$) and receive ($\mathbf{?}A$) operations in an arbitrary theory of processes represented by a monoidal category. For instance, recursion and the internal/external choice duality [26, 54] are not discussed, although they could be considered as extensions in the same way they are to monoidal categories: via trace [29] and linear distributivity [15].

Lenses and incomplete diagrams. Lenses are a notion of bidirectional transformation [23] that can be cast in arbitrary monoidal categories. The first mention of monoidal lenses separate from their classical database counterparts [39] is due to Pastro and Street [50], who identify them as an example of a promonoidal category. However, it was with a different monoidal structure [56] that they became popular in recent years, spawning applications not only in bidirectional transformations [23] but also in functional programming [53, 12], open games [27], polynomial functors [49] and quantum combs [31]. Relating this monoidal category of lenses with the previous promonoidal category of lenses was an open problem; and the promonoidal structure was mostly ignored in applications. We solve this problem, proving that lenses are a universal normal symmetric produoidal category (the symmetric monoidal contexts), which endows them with a novel algebra and a novel universal property. This also extends work on the relation between *incomplete diagrams*, *comb-shaped diagrams*, and *lenses* [57, 59].

Finally, Nester et al. have recently proposed a syntax for lenses and message-passing [48, 7] and lenses themselves have been applied to protocol specification [67]. Spivak [65] also discusses the multicategory of *wiring diagrams*, later used for incomplete diagrams [52] and

related to lenses [61]. The promonoidal categories we use can be seen as multicategories with an extra coherence property. In this sense, we contribute the missing algebraic structure of the universal multicategory of *wiring diagrams relative to a monoidal category*.

1.6 Contributions

Our main contribution is the universal characterization of a produoidal category of *monoidal contexts* over a monoidal category (Theorem 5.4).

We construct an adjunction between monoidal categories and produoidal categories in Section 3, and we characterize spliced monoidal arrows as the cofree produoidal category over a monoidal category (Theorem 3.8); in order to do this, we also introduce a version of the splice-contour construction that creates an adjunction between categories and promonoidal categories, the interested reader can follow the full version [20, Appendix B].

We introduce the free normalization of an arbitrary produoidal category (Theorem 4.3). Normalization had been only described for well-behaved duoidal categories [25]; we show that any produoidal category can be normalized and we construct an idempotent normalization monad. We universally characterize the algebra of monoidal contexts as a free normalization (Theorem 5.4) in Section 5; we universally characterize the algebra of monoidal lenses as a free symmetric normalization (Theorem 6.2) in Section 6. Finally, we introduce an interpretation of send/receive types (**!**/**?**) (Proposition 6.5) in terms of monoidal lenses.

2 Preliminaries: Profunctors and Dinaturality

Profunctors describe families of processes indexed by the input and output types of a category. Since they will be our main tool in the following, we give a brief introduction. More details can be found in the full version of this paper [20, Appendix A].

▶ **Definition 2.1.** *A* profunctor $P\colon \mathbb{B}_0 \times \ldots \times \mathbb{B}_m \nrightarrow \mathbb{A}_0 \times \ldots \times \mathbb{A}_n$ *is a functor*

$$P\colon \mathbb{A}_0^{op} \ldots \times \mathbb{A}_n^{op} \times \mathbb{B}_0 \times \ldots \times \mathbb{B}_m \to \mathbf{Set}.$$

For our purposes, a profunctor $P(A_0, \ldots, A_n; B_0, \ldots, B_m)$ is a family of processes indexed by contravariant inputs $A_0, \ldots, A_n$ and covariant outputs $B_0, \ldots, B_m$. The profunctor is endowed with jointly functorial left $(\succ_0, \ldots, \succ_n)$ and right $(\prec_0, \ldots, \prec_m)$ actions of the morphisms of $\mathbb{A}_0, \ldots, \mathbb{A}_n$ and $\mathbb{B}_0, \ldots, \mathbb{B}_m$, respectively [5, 44].[1]

Composing profunctors is subtle: the same processes could arise as the composite of different pairs of processes, so we need to impose an equivalence relation. Imagine we try to connect two different processes:

$$p \in P(A_0, \ldots, A_n; B_0, \ldots, B_m), \text{ and } q \in Q(C_0, \ldots, C_k; D_0, \ldots, D_h);$$

and we have some morphism $f\colon B_i \to C_j$ that translates the i-th output port of p to the j-th input port of q. Let us write $({}_i|_j)$ for this connection operation. Note that we could connect them in two different ways: we could

- change *the output of the first process* $p \prec_i f$ before connecting both, $(p \prec_i f)\,{}_i|_j\, q$;
- or change *the input of the second process* $f \succ_j q$ before connecting both, $p\,{}_i|_j\,(f \succ_j q)$.

[1] We simply use $(\prec/\succ)$ without any subscript whenever the input/output is unique.

These are different descriptions, made up of two different components. However, they essentially describe the same process [19]: they are *dinaturally equal*. Indeed, profunctors are canonically endowed with this notion of equivalence [5, 44], precisely equating these two descriptions. Profunctors, and their elements, are thus composed *up to dinatural equivalence*.

▶ **Definition 2.2** (Dinatural equivalence). *Consider two profunctors* $P\colon \mathbb{B}_0 \times \ldots \times \mathbb{B}_m \nrightarrow \mathbb{A}_0 \times \ldots \times \mathbb{A}_n$ *and* $Q\colon \mathbb{D}_0 \times \ldots \times \mathbb{D}_h \nrightarrow \mathbb{C}_0 \times \ldots \times \mathbb{C}_k$ *such that* $\mathbb{B}_i = \mathbb{C}_j$*; and let* $\mathbf{S}^{i,j}_{P,Q}(A;D)$ *be the set*

$$\sum_{X \in \mathbb{B}_i} P(A_0 \ldots A_n; B_0 \ldots X \ldots B_m) \times Q(C_0 \ldots X \ldots C_k; D_0 \ldots D_h).$$

Dinatural equivalence, $(\sim)$, *on the set* $\mathbf{S}^{i,j}_{P,Q}(A;D)$ *is the smallest equivalence relation satisfying* $(p \prec_i f \,|_j\, q) \sim (p \,|_i\, f \succ_j q)$*. The* coend *is defined as this coproduct quotiented by dinaturality,* $\mathbf{S}^{i,j}_{P,Q}(A;D)/(\sim)$*, and written as an integral.*

$$\int^{X \in \mathbb{C}_j} P(A_0 \ldots A_n; B_0 \ldots X \ldots B_m) \times Q(C_0 \ldots X \ldots C_k; D_0 \ldots D_h).$$

▶ **Definition 2.3** (Profunctor composition). *Consider two profunctors* $P\colon \mathbb{B}_0 \times \ldots \times \mathbb{B}_m \nrightarrow \mathbb{A}_0 \times \ldots \times \mathbb{A}_n$ *and* $Q\colon \mathbb{D}_0 \times \ldots \times \mathbb{D}_h \nrightarrow \mathbb{C}_0 \times \ldots \times \mathbb{C}_k$ *such that* $\mathbb{B}_i = \mathbb{C}_j$*; their* composition *along ports* i *and* j *is a profunctor* $P \diamond Q : \mathbb{B}_0 \times \ldots \times \mathbb{D}_0 \times \ldots \times \mathbb{D}_h \times \ldots \times \mathbb{B}_m \nrightarrow \mathbb{C}_0 \times \ldots \times \mathbb{A}_0 \times \ldots \times \mathbb{A}_n \times \ldots \times \mathbb{C}_k$*; we write it marking this connection*

$$P(A_0 \ldots A_n; B_0 \ldots \bullet \ldots B_m) \diamond Q(C_0 \ldots \bullet \ldots C_k; D_0 \ldots D_h),$$

and it is defined as the coproduct of the product of both profunctors, indexed by the common variable, and quotiented by dinatural equivalence,

$$\int^{X \in \mathbb{C}} P(A_0 \ldots A_n; B_0 \ldots X \ldots B_m) \times Q(C_0 \ldots X \ldots C_k; D_0 \ldots D_h).$$

3 Parallel-Sequential Context

Monoidal categories are an algebraic structure for sequential and parallel composition: they contain a "tensoring" operator on morphisms, $(\otimes)$, apart from the usual sequencing, $(\fatsemi)$, and identities (id).

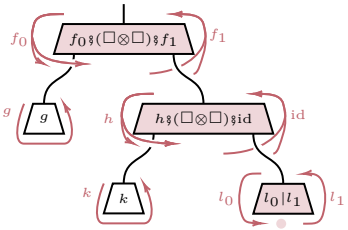


Figure 4 Example decomposition.

Assume a monoidal morphism factors as follows: $f_0 \fatsemi (g \otimes (h \fatsemi (k \otimes (l_0 \fatsemi l_1)))) \fatsemi f_1$. We can say that this morphism came from dividing everything between f_0 and f_1 by a tensor. That is, from a context $f_0 \fatsemi (\square \otimes \square) \fatsemi f_1$. We filled the first hole of this context with a g, and then proceeded to split the second part as $h \fatsemi (\square \otimes \square) \fatsemi \mathrm{id}$. Finally, we filled the first part with k and the second one we filled with l_0, id_I, and l_1.

This section studies decomposition of morphisms in a *monoidal* category, in the same way we study decomposition of morphisms in a category (see the full version [20, Appendix B]). We present an algebraic structure for decomposing both sequential and parallel compositions: *produoidal categories*.

3.1 Produoidal Categories

Produoidal categories, first defined by Booker and Street [10], provide an algebraic structure for the interaction of sequential and parallel decomposition. A produoidal category $\mathbb{V}$ not only contains *morphisms*, $\mathbb{V}(X;Y)$, as in a category, but also *sequential splits*, $\mathbb{V}(X;Y_0 \lhd Y_1)$, and *sequential units*, $\mathbb{V}(X;N)$, provided by a promonoidal structure; and *parallel splits*, $\mathbb{V}(X;Y_0 \otimes Y_1)$ and *parallel units*, $\mathbb{V}(X;I)$, provided by another promonoidal structure.

These splits must be coherent. For instance, imagine we want to decompose X (sequentially) into Y_0, Y_1 and Y_2. Decomposing X into Y_0 and something ($\bullet$), and then decomposing that something into Y_1 and Y_2 *should be doable in essentially the same ways* as decomposing X into something ($\bullet$) and Y_2, and then decomposing that something into Y_0 and Y_1. Formally, we are saying that,

$$\mathbb{V}(X;Y_0 \lhd \bullet) \diamond \mathbb{V}(\bullet;Y_1 \lhd Y_2) \cong \mathbb{V}(X;\bullet \lhd Y_2) \diamond \mathbb{V}(\bullet;Y_0 \lhd Y_1),$$

and, in fact, we just write $\mathbb{V}(X;Y_0 \lhd Y_1 \lhd Y_2)$ for the set of such decompositions. This is precisely what we ask for in a promonoidal structure.

▶ **Definition 3.1** (Produoidal category)**.** *A* produoidal category *is a category* $\mathbb{V}$ *endowed with two promonoidal structures,*

$$\mathbb{V}(\bullet;\bullet \otimes \bullet)\colon \mathbb{V} \times \mathbb{V} \nrightarrow \mathbb{V}, \text{ and } \mathbb{V}(\bullet;I)\colon 1 \nrightarrow \mathbb{V},$$
$$\mathbb{V}(\bullet;\bullet \lhd \bullet)\colon \mathbb{V} \times \mathbb{V} \nrightarrow \mathbb{V}, \text{ and } \mathbb{V}(\bullet;N)\colon 1 \nrightarrow \mathbb{V},$$

such that one laxly distributes over the other. This is to say that it is endowed with the following natural laxators*:* $\psi_2\colon \mathbb{V}(\bullet;(X \lhd Y) \otimes (Z \lhd W)) \to \mathbb{V}(\bullet;(X \otimes Z) \lhd (Y \otimes W))$, $\psi_0\colon \mathbb{V}(\bullet;I) \to \mathbb{V}(\bullet;I \lhd I)$, $\varphi_2\colon \mathbb{V}(\bullet;N \otimes N) \to \mathbb{V}(\bullet;N)$, *and* $\varphi_0\colon \mathbb{V}(\bullet;I) \to \mathbb{V}(\bullet;N)$*. Laxators, together with unitors and associators, must satisfy coherence conditions (see [20, Definition J.7]). Denote by* **Produo** *the category of produoidal categories and produoidal functors.*

▶ Remark 3.2 (Nesting profunctorial structures). Notation for nesting functorial structures, say ($\lhd$) and ($\otimes$), is straightforward: we use expressions like $(X_1 \otimes Y_1) \lhd (X_2 \otimes Y_2)$ without a second thought. Nesting the profunctorial (or *virtual*) structures ($\lhd$) and ($\otimes$) is more subtle: defining $\mathbb{V}(\bullet;X \otimes Y)$ and $\mathbb{V}(\bullet;X \lhd Y)$ for each pair of objects X and Y does not itself define what something like $\mathbb{V}(\bullet;(X_1 \otimes Y_1) \lhd (X_2 \otimes Y_2))$ means. Recall that, in the profunctorial case, $X_1 \lhd Y_1$ and $X_1 \otimes Y_1$ are not objects themselves: they are just names for the profunctors $\mathbb{V}(\bullet;X_1 \lhd Y_1)$ and $\mathbb{V}(\bullet;X_1 \otimes Y_1)$, which are not *representable*.

Instead, when we write $\mathbb{V}(\bullet;(X_1 \otimes Y_1) \lhd (X_2 \otimes Y_2))$, we formally mean the composition of profunctors $\mathbb{V}(\bullet;\bullet_1 \lhd \bullet_2) \diamond \mathbb{V}(\bullet_1;X_1 \otimes Y_1) \diamond \mathbb{V}(\bullet_2;X_2 \otimes Y_2)$. By convention, nesting profunctorial structures means profunctor composition in this text.

3.2 Monoidal Contour of a Produoidal Category

Any produoidal category freely generates a monoidal category, its *monoidal contour*. Contours form a monoidal category of paths around the decomposition trees of the produoidal category. Contours follow a pleasant geometric pattern, where we follow the shape of the decomposition, both in the parallel and sequential dimensions, to construct both sequential and parallel compositions for a monoidal category.

▶ **Definition 3.3** (Monoidal contour). *The* contour *of a produoidal category* $\mathbb{B}$ *is the monoidal category* $\mathcal{D}\mathbb{B}$ *presented by two objects,* X^L *(left-handed) and* X^R *(right-handed), for each object* $X \in \mathbb{B}_{obj}$*; and generated by arrows that arise from* contouring *both sequential and parallel decompositions of the promonoidal category.*

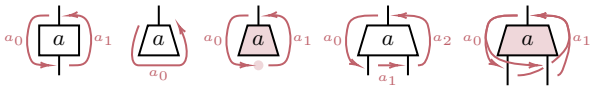
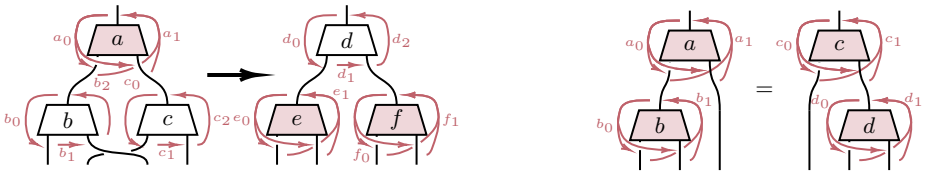


Figure 5 Generators of the monoidal category of contours.

Specifically, it is presented by the following generators *(i)* a pair of morphisms $a_0 \in \mathcal{D}\mathbb{B}(A^L; X^L)$, $a_1 \in \mathcal{D}\mathbb{B}(X^R; A^R)$ for each morphism $a \in \mathbb{B}(A; X)$; *(ii)* a morphism $a_0 \in \mathcal{D}\mathbb{B}(A^L; A^R)$, for each sequential unit $a \in \mathbb{C}(A; N)$; *(iii)* a pair of morphisms $a_0 \in \mathcal{D}\mathbb{B}(A^L; I)$ and $a_0 \in \mathcal{D}\mathbb{B}(I; A^R)$, for each parallel unit $a \in \mathbb{B}(A; I)$; *(iv)* a triple of morphisms $a_0 \in \mathcal{D}\mathbb{B}(A^L; X^L)$, $a_1 \in \mathcal{D}\mathbb{B}(X^R; Y^L)$, $a_2 \in \mathcal{D}\mathbb{B}(Y^R; A^R)$ for each sequential split $a \in \mathbb{B}(A; X \triangleleft Y)$; and *(v)* a pair of morphisms $a_0 \in \mathcal{D}\mathbb{B}(A^L; X^L \otimes Y^L)$ and $a_1 \in \mathcal{D}\mathbb{B}(X^R \otimes Y^R; A^R)$ for each parallel split $a \in \mathbb{B}(A; X \otimes Y)$, see Figure 5.

We impose some equations that arise naturally from the associator and unitor of the sequential structure ($\triangleleft$), as done by Melliès and Zeilberger [45]; but moreover, we also impose some new equations, coming from the parallel structure ($\otimes$), as depicted in Figures 6a and 6b. We refer the interested reader to the full version [20] for the full list of equations.



(a) **(b)**

Figure 6 Equations coming from laxators (a) and associators (b).

▶ **Proposition 3.4.** *Monoidal contour extends to a functor* $\mathcal{D} : \mathbf{Produo} \to \mathbf{Mon}$.

Proof. See the full version [20, Proposition E.3]. ◀

3.3 Produoidal Category of Spliced Monoidal Arrows

We want to go the other way around: given a monoidal category, what is the produoidal category that tracks decomposition of arrows in that monoidal category? This subsection finds a right adjoint to the monoidal contour construction: the produoidal category of *spliced monoidal arrows*.

▶ **Definition 3.5.** *Let* $(\mathbb{C}, \otimes, I)$ *be a monoidal category. The produoidal category of* spliced monoidal arrows, $\mathcal{T}\mathbb{C}$, *has as objects pairs of objects of* $\mathbb{C}$. *It uses the following profunctors to define*

- *morphisms,* $\mathcal{T}\mathbb{C}\left(\begin{smallmatrix}A\\B\end{smallmatrix}; \begin{smallmatrix}X\\Y\end{smallmatrix}\right) = \mathbb{C}(A; X) \times \mathbb{C}(Y, B)$;
- *sequential splits,* $\mathcal{T}\mathbb{C}(\begin{smallmatrix}A\\B\end{smallmatrix}; \begin{smallmatrix}X \triangleleft X'\\Y\end{smallmatrix}) = \mathbb{C}(A; X) \times \mathbb{C}(Y; X') \times \mathbb{C}(Y'; B)$;
- *parallel splits,* $\mathcal{T}\mathbb{C}(\begin{smallmatrix}A\\B\end{smallmatrix}; \begin{smallmatrix}X \otimes X'\\Y\end{smallmatrix}) = \mathbb{C}(A; X \otimes X') \times \mathbb{C}(Y \otimes Y'; B)$;
- *sequential units,* $\mathcal{T}\mathbb{C}(\begin{smallmatrix}A\\B\end{smallmatrix}; N) = \mathbb{C}(A; B)$;
- *and parallel units,* $\mathcal{T}\mathbb{C}(\begin{smallmatrix}A\\B\end{smallmatrix}; I) = \mathbb{C}(A; I) \times \mathbb{C}(I; B)$.

In other words, morphisms are pairs of arrows *written as* $f \fatsemi \Box \fatsemi g \in \mathcal{T}\mathbb{C}\left(\begin{smallmatrix} A \\ B \end{smallmatrix}; \begin{smallmatrix} X \\ Y \end{smallmatrix}\right)$. *Sequential splits are* triples of arrows, *written as* $f \fatsemi \Box \fatsemi g \fatsemi \Box \fatsemi h \in \mathcal{T}\mathbb{C}\left(\begin{smallmatrix} A \\ B \end{smallmatrix}; \begin{smallmatrix} X \\ Y \end{smallmatrix} \triangleleft \begin{smallmatrix} X' \\ Y' \end{smallmatrix}\right)$. *Parallel splits are* pairs of arrows, *written as* $f \fatsemi (\Box \otimes \Box) \fatsemi h \in \mathcal{T}\mathbb{C}\left(\begin{smallmatrix} A \\ B \end{smallmatrix}; \begin{smallmatrix} X \\ Y \end{smallmatrix} \otimes \begin{smallmatrix} X' \\ Y' \end{smallmatrix}\right)$. *Sequential units are* arrows, *written simply as* $f \in \mathcal{T}\mathbb{C}\left(\begin{smallmatrix} A \\ B \end{smallmatrix}; N\right)$. *parallel units are pairs of arrows, written as* $f \parallel g \in \mathcal{T}\mathbb{C}\left(\begin{smallmatrix} A \\ B \end{smallmatrix}; I\right)$.

Finally, the laxators are defined by plugging the different pieces and reinterpreting the relative position of the holes. Let us give an example of what this means; we refer the interested reader to the full version [20, Appendix E.2] for full details.

▶ **Example 3.6.** For instance, the last laxator takes parallel sequences of holes, $f_0 \fatsemi ((h_0 \fatsemi \Box \fatsemi h_1 \fatsemi \Box \fatsemi h_2) \otimes (k_0 \fatsemi \Box \fatsemi k_1 \fatsemi \Box \fatsemi k_2)) \fatsemi f_1$ into sequences of parallel holes, $f_0 \fatsemi (h_0 \otimes k_0) \fatsemi (\Box \otimes \Box) \fatsemi (h_1 \otimes k_1) \fatsemi (\Box \otimes \Box) \fatsemi (h_2 \otimes k_2) \fatsemi f_1$.

▶ Remark 3.7. The produoidal algebra of spliced arrows is a natural construction: abstractly, we know that there is a duoidal structure on the endomodules of any monoidal category [18, 66] – this is its explicit produoidal counterpart. What may be more surprising is that spliced arrows have themselves a universal property as part of an adjunction.

▶ **Theorem 3.8.** *Spliced monoidal arrows form a produoidal category with their sequential and parallel splits, units, and suitable coherence morphisms and laxators. Spliced monoidal arrows extend to a functor* $\mathcal{T} : \mathbf{Mon} \to \mathbf{Produo}$. *The monoidal contour and the produoidal splice are left and right adjoints to each other, respectively.*

Proof. See the full version [20, Propositions E.4 and E.9 and Theorem E.10]. ◀

▶ Remark 3.9. When $\mathbb{C}$ is a *symmetric* monoidal category, then $\mathcal{T}\mathbb{C}$ is moreover a symmetric produoidal category with symmetry defined using the symmety of $\mathbb{C}$.

3.4 Representable Parallel Structure

A produoidal category has two tensors, and neither is, in principle, representable. However, the cofree produoidal category over a category we have just constructed happens also to have a representable tensor, $(\otimes)$. Spliced monoidal arrows form a monoidal category.

▶ **Proposition 3.10.** *Parallel splits and parallel units of spliced monoidal arrows are representable profunctors. That is,* $\mathcal{T}\mathbb{C}\left(\begin{smallmatrix} A \\ B \end{smallmatrix}; \begin{smallmatrix} X \\ Y \end{smallmatrix} \otimes \begin{smallmatrix} X' \\ Y' \end{smallmatrix}\right) \cong \mathcal{T}\mathbb{C}\left(\begin{smallmatrix} A \\ B \end{smallmatrix}; \begin{smallmatrix} X \otimes X' \\ Y \otimes Y' \end{smallmatrix}\right)$, *and* $\mathcal{T}\mathbb{C}\left(\begin{smallmatrix} A \\ B \end{smallmatrix}; I\right) \cong \mathcal{T}\mathbb{C}\left(\begin{smallmatrix} A \\ B \end{smallmatrix}; \begin{smallmatrix} I \\ I \end{smallmatrix}\right)$.

In fact, these sets are equal by definition. However, we argue that there is a good reason to work in the full generality of produoidal categories: produoidal categories can always be *normalized*.

Normalization is a procedure to mix both tensors of a duoidal category, $(\otimes)$ and $(\triangleleft)$, but not every duoidal category has a normalization [25]. It is folklore that one loses nothing by regarding non-representable produoidal structures as representable *duoidal structures on presheaves*, dismissing that they are moreover *closed* [18]; thus, one would expect only some produoidal categories to be normalizable. Against folklore, we prove that every produoidal category, representable or not, has a *universal normalization*, a normal produoidal category which may be again representable or not (Theorem 4.3). We use this procedure to universally characterize *monoidal contexts* in Section 5, which form a produoidal category without representable structure.

▶ Remark 3.11. This means $\mathcal{T}\mathbb{C}$ has the structure of a *virtual duoidal category* [64] or *monoidal multicategory*, defined by Aguiar, Haim and López Franco [3] as a pseudomonoid in the cartesian monoidal 2-category of multicategories.

4 Interlude: Normalization

Produoidal categories seem to contain too much structure: of course, we want to split things in two different ways, sequentially ($\triangleleft$) and in parallel ($\otimes$); but that does not necessarily mean that we want to keep track of two different types of units, parallel (I) and sequential (N). The atomic components of our decomposition algebra should be the same, without having to care if they are *atomic for sequential composition* or *atomic for parallel composition*.

Fortunately, there exists an abstract procedure that, starting from any produoidal category, constructs a new produoidal category where both units are identified. This procedure is known as *normalization*, and the resulting produoidal categories are called *normal*.

▶ **Definition 4.1** (Normal produoidal category). *A* normal produoidal category *is a produoidal category where the laxator* $\varphi_0 \colon \mathbb{V}(\bullet; I) \to \mathbb{V}(\bullet; N)$ *is an isomorphism. Normal produoidal categories form a category* **nProduo** *with produoidal functors between them and endowed with fully faithful forgetful functor* $\mathcal{U} \colon \mathbf{nProduo} \to \mathbf{Produo}$.

▶ **Theorem 4.2** (Normalization construction). *Let* $\mathbb{V}_{\otimes,I,\triangleleft,N}$ *be a produoidal category. The profunctor* $\mathcal{N}\mathbb{V}(\bullet;\bullet) = \mathbb{V}(\bullet; N \otimes \bullet \otimes N)$ *forms a promonad [33, 58]. Moreover, the Kleisli category of this promonad is a normal produoidal category with the following splits and units:* $\mathcal{N}\mathbb{V}(A; B \otimes_N C) = \mathbb{V}(A; N \otimes B \otimes N \otimes C \otimes N)$; $\mathcal{N}\mathbb{V}(A; B \triangleleft_N C) = \mathbb{V}(A; (N \otimes B \otimes N) \triangleleft (N \otimes C \otimes N))$; $\mathcal{N}\mathbb{V}(A; I_N) = \mathbb{V}(A; N)$; *and* $\mathcal{N}\mathbb{V}(A; N_N) = \mathbb{V}(A; N)$.

Proof. See the full version [20, Theorem F.1]. ◀

Garner and López Franco [25] introduced a partial normalization procedure for duoidal categories. We contribute a general normalization procedure for produoidal categories and we characterize it universally. Produoidal normalization behaves slightly better than duoidal normalization: it always succeeds, and we prove that it forms an idempotent monad (Theorem 4.3). The technical reason for this improvement is that the original duoidal normalization required the existence of certain coequalizers in $\mathbb{V}$; produoidal normalization uses coequalizers in **Set**. See the full version [20, Appendix F.4] for an outline of the relation between the two procedures.

▶ **Theorem 4.3** (Free normal produoidal). *Normalization extends to an idempotent monad. Moreover, normalization determines an adjunction between produoidal categories and normal produoidal categories,* $\mathcal{N} \colon \mathbf{Produo} \rightleftharpoons \mathbf{nProduo} \colon \mathcal{U}$. *That is,* $\mathcal{N}\mathbb{V}$ *is the free normal produoidal category over* $\mathbb{V}$.

Proof. See the full version [20, Theorems F.3 and F.5]. ◀

In the previous Section 3, we constructed the produoidal category of spliced monoidal arrows, which distinguishes between morphisms and morphisms with a hole in the monoidal unit. This is because the latter hole splits the morphism in two parts. Normalization equates both; it sews these two parts. In Section 5, we explicitly construct monoidal contexts, the normalization of spliced monoidal arrows.

▶ Remark 4.4. Normalization is a generic procedure that applies to any produoidal category, it does not matter if the parallel split ($\otimes$) is symmetric or not. However, when $\otimes$ happens to be symmetric, we can also apply a more specialized normalization procedure: symmetric normalization. See the full version [20, Appendix F.2].

5 Monoidal Context: Mixing $\triangleleft$ and $\otimes$ by normalization

Monoidal contexts formalize the notion of an incomplete morphism in a monoidal category. The category of monoidal contexts will have a rich algebraic structure: we shall be able to still compose contexts sequentially and in parallel and, at the same time, we shall be able to fill a context using another monoidal context. Perhaps surprisingly, then, the category of monoidal contexts is not even monoidal.

We justify this apparent contradiction in terms of profunctorial structure: the category is not monoidal, but it does have two promonoidal structures that precisely represent sequential and parallel composition. These structures form a normal produoidal category. In fact, we show it to be the normalization of the produoidal category of spliced monoidal arrows. This section constructs explicitly this normal produoidal category of monoidal contexts.

5.1 The Category of Monoidal Contexts

A monoidal context, $\mathcal{MC}\left(\begin{smallmatrix}A\\B\end{smallmatrix};\begin{smallmatrix}X\\Y\end{smallmatrix}\right)$, represents a process from A to B with a hole admitting a process from X to Y. In this sense, monoidal contexts are similar to spliced monoidal arrows. The difference with spliced monoidal arrows is that monoidal contexts allow for communication to happen to the left and to the right of this hole.

▶ **Definition 5.1** (Monoidal context)**.** *Let $(\mathbb{C},\otimes,I)$ be a monoidal category.* Monoidal contexts *are the elements of the profunctor $\mathcal{MC}\left(\begin{smallmatrix}A\\B\end{smallmatrix};\begin{smallmatrix}X\\Y\end{smallmatrix}\right)=\mathbb{C}(A;\bullet_1\otimes X\otimes\bullet_2)\diamond\mathbb{C}(\bullet_1\otimes Y\otimes\bullet_2;B)$ over $\mathbb{C}^{op}\times\mathbb{C}$.*

In other words, a *monoidal context* from A to B, *with a hole* from X to Y, is an equivalence class consisting of a pair of objects $M,N\in\mathbb{C}_{\text{obj}}$ and a pair of morphisms $f\in\mathbb{C}(A;M\otimes X\otimes N)$ and $g\in\mathbb{C}(M\otimes Y\otimes N;B)$, quotiented by dinaturality of M and N (Figure 8). We write monoidal contexts as

$$(f\fatsemi(\mathrm{id}_M\otimes\blacksquare\otimes\mathrm{id}_N)\fatsemi g)\in\mathcal{MC}\left(\begin{smallmatrix}A\\B\end{smallmatrix};\begin{smallmatrix}X\\Y\end{smallmatrix}\right).$$

In this notation, dinaturality explicitly means $(f\fatsemi(m\otimes\mathrm{id}_X\otimes n)\fatsemi(\mathrm{id}_W\otimes\blacksquare\otimes\mathrm{id}_H)\fatsemi g)=(f\fatsemi(\mathrm{id}_M\otimes\blacksquare\otimes\mathrm{id}_N)\fatsemi(m\otimes\mathrm{id}_Y\otimes n)\fatsemi g)$.

▶ Remark 5.2. Even when we introduce $(\mathrm{id}\otimes\blacksquare\otimes\mathrm{id})$ as a piece of suggestive notation, we can still write $(g\otimes\blacksquare\otimes h)$ unambiguously, because of dinaturality: $(g\otimes\mathrm{id}\otimes h)\fatsemi(\mathrm{id}\otimes\blacksquare\otimes\mathrm{id})=(\mathrm{id}\otimes\blacksquare\otimes\mathrm{id})\fatsemi(g\otimes\mathrm{id}\otimes h)$.

5.2 The Normal Produoidal Algebra of Monoidal Contexts

▶ **Definition 5.3.** *Let us endow monoidal contexts with a normal produoidal structure. The category of monoidal contexts, $\mathcal{MC}$, has as objects pairs of objects of $\mathbb{C}$. Units are defined by $\mathcal{MC}\left(\begin{smallmatrix}A\\B\end{smallmatrix};N\right)=\mathbb{C}(A;B)$. We use the following profunctors to define sequential splits and parallel splits,*

$$\mathcal{MC}\left(\begin{smallmatrix}A\\B\end{smallmatrix};\begin{smallmatrix}X\\Y\end{smallmatrix}\triangleleft\begin{smallmatrix}X'\\Y'\end{smallmatrix}\right)=\mathbb{C}(A;\bullet_1\otimes X\otimes\bullet_2)\diamond\mathbb{C}(\bullet_1\otimes Y\otimes\bullet_2;\bullet_3\otimes X'\otimes\bullet_4)\diamond\mathbb{C}(\bullet_3\otimes Y'\otimes\bullet_4;B);$$
$$\mathcal{MC}\left(\begin{smallmatrix}A\\B\end{smallmatrix};\begin{smallmatrix}X\\Y\end{smallmatrix}\otimes\begin{smallmatrix}X'\\Y'\end{smallmatrix}\right)=\mathbb{C}(A;\bullet_1\otimes X\otimes\bullet_2\otimes X'\otimes\bullet_3)\diamond\mathbb{C}(\bullet_1\otimes Y\otimes\bullet_2\otimes Y'\otimes\bullet_3;B).$$

In other words, sequential splits are triples of arrows quotiented by dinaturality and written as $f\fatsemi(\mathrm{id}\otimes\blacksquare\otimes\mathrm{id})\fatsemi g\fatsemi(\mathrm{id}\otimes\blacksquare\otimes\mathrm{id})\fatsemi h$. Parallel splits are pairs of arrows quotiented by dinaturality and written as $f\fatsemi(\mathrm{id}\otimes\blacksquare\otimes\mathrm{id}\otimes\blacksquare\otimes\mathrm{id})\fatsemi g$. Units are simply arrows $f\colon A\to B$. Morphisms are pairs of arrows, written as $f\fatsemi(\mathrm{id}\otimes\blacksquare\otimes\mathrm{id})\fatsemi g$, and also quotiented by dinaturality. Figure 7 gives the diagrammatic representations of these components. Dinaturality for sequential splits and parallel splits is depicted in Figure 8.

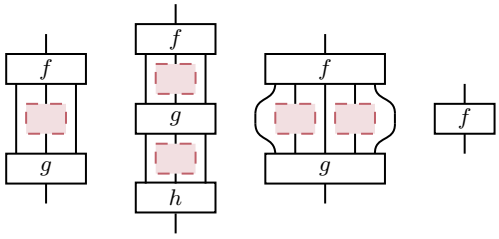


Figure 7 Morphisms, sequential and parallel splits, and units of the splice monoidal arrow produoidal category.

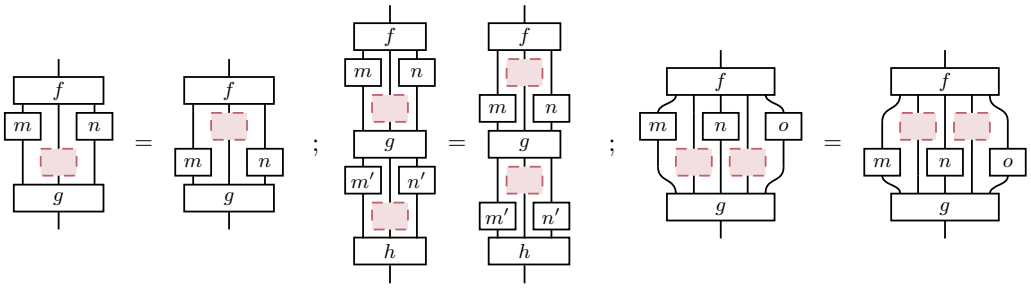


Figure 8 Dinaturality of morphisms, and sequential and parallel splits of monoidal contexts.

▶ **Theorem 5.4.** *The category of monoidal contexts forms a normal produoidal category with its units, sequential and parallel splits. Monoidal contexts are the free normalization of the cofree produoidal category over a category. In other words, monoidal contexts are the normalization of spliced monoidal arrows,* $\mathcal{NT}\mathbb{C} \cong \mathcal{M}\mathbb{C}$.

Proof. See the full version [20, Propositions G.4 and G.5 and Theorem G.13]. ◀

6 Monoidal Lenses

Monoidal lenses are *symmetric* monoidal contexts. Again, the category of monoidal lenses has a rich algebraic structure; and again, most of this structure exists only virtually in terms of profunctors. In this case, though, the monoidal tensor *does* indeed exist: contrary to monoidal contexts, monoidal lenses form also a monoidal category. This is perhaps why applications of monoidal lenses have grown popular in recent years [56], with applications in decision theory [27], supervised learning [17, 22] and most notably in functional data accessing [42, 53, 6, 12]. The promonoidal structure of lenses was ignored, even when, after now identifying for the first time its relation to the monoidal structure of lenses, we argue that it could be potentially useful in these applications: e.g. in multi-stage decision problems, or in multi-stage data accessors.

This section explicitly constructs the normal symmetric produoidal category of *monoidal lenses*. We describe it for the first time by a universal property: it is the free symmetric normalization of the cofree produoidal category.

6.1 The Normal Symmetric Produoidal Algebra of Monoidal Lenses

▶ **Definition 6.1** (Monoidal Lens). *Let* $(\mathbb{C}, \otimes, I)$ *be a symmetric monoidal category. A monoidal lens of type* $\mathcal{L}\mathbb{C}\left(\begin{smallmatrix}A\\B\end{smallmatrix};\begin{smallmatrix}X\\Y\end{smallmatrix}\right)$ *represents a process in a symmetric monoidal category with a hole admitting a process from* X *to* Y *[56]. Explicitly,* monoidal lenses *are the elements of the profunctor* $\mathcal{L}\mathbb{C}\left(\begin{smallmatrix}A\\B\end{smallmatrix};\begin{smallmatrix}X\\Y\end{smallmatrix}\right) = \mathbb{C}(A; \bullet \otimes X) \diamond \mathbb{C}(\bullet \otimes Y; B)$ *over* $\mathbb{C}^{op} \times \mathbb{C}$.

In other words, a *monoidal lens* from A to B, *with a hole* from X to Y, is an equivalence class consisting of a pair of objects $M, N \in \mathbb{C}_{\text{obj}}$ and a pair of morphisms $f \in \mathbb{C}(A; M \otimes X)$ and $g \in \mathbb{C}(M \otimes Y; B)$, quotiented by dinaturality of M. We write monoidal lenses as $f \fatsemi (\mathrm{id}_M \otimes \blacksquare) \fatsemi g \in \mathcal{LC}\left(\begin{smallmatrix} A \\ B \end{smallmatrix}; \begin{smallmatrix} X \\ Y \end{smallmatrix}\right)$.

▶ **Theorem 6.2.** *Monoidal lenses form a normal symmetric produoidal category with the units given by* $\mathcal{LC}\left(\begin{smallmatrix} A \\ B \end{smallmatrix}; N\right) = \mathbb{C}(A; B)$*, and the following sequential and parallel splits.*

$$\mathcal{LC}\left(\begin{smallmatrix} A \\ B \end{smallmatrix}; \begin{smallmatrix} X \triangleleft X' \\ Y \end{smallmatrix}\right) = \mathbb{C}(A; \bullet_1 \otimes X) \diamond \mathbb{C}(\bullet_1 \otimes Y; \bullet_2 \otimes X') \diamond \mathbb{C}(\bullet_2 \otimes Y'; B);$$
$$\mathcal{LC}\left(\begin{smallmatrix} A \\ B \end{smallmatrix}; \begin{smallmatrix} X \otimes X' \\ Y \end{smallmatrix}\right) = \mathbb{C}(A; \bullet_1 \otimes X \otimes X') \diamond \mathbb{C}(\bullet_1 \otimes Y \otimes Y'; B).$$

Monoidal lenses are the free symmetric normalization of the cofree symmetric produoidal category over a symmetric monoidal category.

Proof. See the full version [20, Proposition H.1 and Theorem H.9]. ◀

▶ Remark 6.3 (Representable parallel structure). The parallel splitting structure of monoidal lenses is representable, $\mathcal{LC}\left(\begin{smallmatrix} A \\ B \end{smallmatrix}; \begin{smallmatrix} X \otimes X' \\ Y \end{smallmatrix}\right) = \mathcal{LC}\left(\begin{smallmatrix} A \\ B \end{smallmatrix}; \begin{smallmatrix} X \otimes X' \\ Y \otimes Y' \end{smallmatrix}\right)$. Lenses over a symmetric monoidal category are known to be monoidal [56, 30], but it remained unexplained why a similar structure was not present in non-symmetric lenses. The contradiction can be solved by noting that both symmetric and non-symmetric lenses are indeed *promonoidal*, even if only symmetric lenses provide a representable tensor.

▶ Remark 6.4 (Session notation for lenses). We will write $!A = \left(\begin{smallmatrix} A \\ I \end{smallmatrix}\right)$ and $?B = \left(\begin{smallmatrix} I \\ B \end{smallmatrix}\right)$ for the objects of the produoidal category of lenses that have a monoidal unit as one of its objects. These are enough to express all objects because $!A \otimes ?B = \left(\begin{smallmatrix} A \\ B \end{smallmatrix}\right)$.

▶ **Proposition 6.5.** *Let* $(\mathbb{C}, \otimes, I)$ *be a symmetric monoidal category. There exist monoidal functors* $(!)\colon \mathbb{C} \to \mathcal{LC}$ *and* $(?)\colon \mathbb{C}^{op} \to \mathcal{LC}$*. Moreover, they satisfy the following properties definitionally:* $\mathbb{C}(\bullet; ?A \triangleleft ?B) \cong \mathbb{C}(\bullet; ?A \otimes ?B)$*;* $!(A \otimes B) = !A \otimes !B$*;* $\mathbb{C}(\bullet; !A \triangleleft !B) \cong \mathbb{C}(\bullet; !A \otimes !B)$*;* $?(A \otimes B) = ?A \otimes ?B$*; and* $\mathbb{C}(\bullet; !A \triangleleft ?B) \cong \mathbb{C}(\bullet; !A \otimes ?B)$.

Proof. See the full version [20, Proposition H.7]. ◀

6.2 Protocol Analysis

Let us go back to our running example (Figure 1). We can now declare that the client and server have the following types, representing the order in which they communicate,

$$\text{} \in \mathcal{LC}\left(\begin{smallmatrix} \text{Client} \\ \text{Client} \end{smallmatrix}; !\text{Msg} \triangleleft ?\text{Msg} \triangleleft !\text{Msg}\right); \quad \text{  } \in \mathcal{LC}\left(\begin{smallmatrix} \text{Server} \\ \text{Server} \end{smallmatrix}; ?\text{Msg} \triangleleft !\text{Msg} \triangleleft ?\text{Msg}\right).$$

Moreover, we can use the duoidal algebra to compose them. Indeed, tensoring client and server, we get the following codomain type: $(!\text{Msg} \triangleleft ?\text{Msg} \triangleleft !\text{Msg}) \otimes (?\text{Msg} \triangleleft !\text{Msg} \triangleleft ?\text{Msg})$. We then apply the laxators to mix inputs and outputs, obtaining $(!\text{Msg} \otimes ?\text{Msg}) \triangleleft (?\text{Msg} \otimes !\text{Msg}) \triangleleft (!\text{Msg} \otimes ?\text{Msg})$, and we finally apply the unitors to fill the communication holes with noisy channels of type $\text{Msg} \to \text{Msg}$.

$$\psi_2\left(\text{  } \otimes \text{  }\right) \prec^3_\lambda \texttt{NOISE}^3 \in \mathcal{LC}\left(\begin{smallmatrix} \text{Client} \otimes \text{Server} \\ \text{Client} \otimes \text{Server} \end{smallmatrix}; N\right).$$

We end up obtaining the protocol as a single morphism $\text{Client} \otimes \text{Server} \to \text{Client} \otimes \text{Server}$ in whatever category we are using to program. Assuming the category of finite stochastic maps, this single morphism represents the distribution over the possible outcomes of the protocol. Finally, by dinaturality, we can reason over independent parts of the protocol.

▶ Remark 6.6. Let $(\ \diagup\!\!\blacksquare\ ) = (\mathtt{SYN} \fatsemi (\mathrm{id} \otimes \blacksquare) \fatsemi \mathtt{ACK} \fatsemi (\mathrm{id} \otimes \blacksquare))$. The equalities in Figure 1 are a consequence of dinaturality over $\mathtt{PRJ}$, which acts as the interchange law for incomplete morphisms.

$$\mathtt{SYN} \fatsemi (\mathrm{id} \otimes \blacksquare) \fatsemi \mathtt{ACK} \fatsemi (\mathrm{id} \otimes \blacksquare) = \mathtt{SYN}^* \fatsemi (\mathtt{PRJ} \otimes \mathrm{id}) \fatsemi \blacksquare \fatsemi \mathtt{ACK} \fatsemi (\mathrm{id} \otimes \blacksquare) =$$
$$\mathtt{SYN}^* \fatsemi (\mathrm{id} \otimes \blacksquare) \fatsemi (\mathtt{PRJ} \otimes \mathrm{id}) \fatsemi \mathtt{ACK} \fatsemi (\mathrm{id} \otimes \blacksquare) = \mathtt{SYN}^* \fatsemi (\mathrm{id} \otimes \blacksquare) \fatsemi \mathtt{ACK}^* \fatsemi (\mathrm{id} \otimes \blacksquare).$$

7 Conclusions

Monoidal contexts are an algebra of incomplete processes, commonly generalizing lenses [56] and spliced arrows [45]. In the same way that the π-calculus allows input/output channels of an abstract model of computation, monoidal contexts allow input/output communication on arbitrary theories of processes, such as stochastic or partial functions, quantum processes or relational queries.

Monoidal contexts form a normal produoidal category: a highly structured and rich categorical algebra. Moreover, they are the universal such algebra on a monoidal category. This is good news for applications: the literature on concurrency is rich in frameworks; but the lack of *canonicity* may get us confused when trying to choose, design, or compare among them, as Abramsky [1] has pointed out. Precisely characterizing the universal property of a model addresses this concern. This is also good news for the category theorist: not only is this an example shedding light on a relatively obscure structure; it is a paradigmatic such one.

We rely on two mathematical ideas: *monoidal* and *duoidal* categories on one hand, and *dinaturality* and *profunctorial* structures on the other. *Monoidal categories*, which could be accidentally dismissed as a toy version of cartesian categories, show that their string diagrams can bootstrap our conceptual understanding of new fundamental process structures, while keeping an abstraction over their implementation that cartesian categories cannot afford. Duoidal categories are such an example: starting to appear insistently in computer science [62, 34], they capture the posetal structure of process dependency and communication. *Dinaturality*, virtual structures and profunctors, even if sometimes judged arcane, show again that they can canonically model a notion as concrete as process composition.

7.1 Further Work

Dependencies. Shapiro and Spivak [62] prove that normal symmetric duoidal categories with certain limits additionally have the structure of *dependence categories*: they can not only express dependence structures generated by $(\lhd)$ and $(\otimes)$, but arbitrary poset-mediated dependence structures. Produoidal categories are better behaved: the limits always exist, and we only require these are preserved by the coend (see the full version for details [20]). Weakening dependence categories in this way combines the ideas of Shapiro and Spivak [62] with those of Hefford and Kissinger [32], who employ virtual objects to deal with the non-existence of tensor products in models of spacetime.

Language theory. Melliès and Zeilberger [45] used a multicategorical form of splice-contour adjunction to give a novel proof of the Chomsky-Schützenberger representation theorem, generalized to context-free languages in categories. Our produoidal splice-contour adjunction (Section 3), combined with recent work on languages of morphisms in monoidal categories [21] opens the way for a monoidal version of the Chomsky-Schützenberger theorem.

String diagrams for concurrency. Nester et al. [48, 7] have recently introduced an alternative description of lenses in terms of *proarrow equipments*, which have a good 2-dimensional syntax [47] we can use for send/receive types (**!**/**?**). We have shown how this structure arises universally in symmetric monoidal categories. It remains as further work to determine a good 2-dimensional syntax for concurrent programs with *iteration* and *internal/external choice*.

References

1 Samson Abramsky. What are the fundamental structures of concurrency?: We still don't know! In Luca Aceto and Andrew D. Gordon, editors, *Proceedings of the Workshop "Essays on Algebraic Process Calculi", APC 25, Bertinoro, Italy, August 1-5, 2005*, volume 162 of *Electronic Notes in Theoretical Computer Science*, pages 37–41. Elsevier, 2005. doi:10.1016/j.entcs.2005.12.075.

2 Samson Abramsky and Bob Coecke. Categorical quantum mechanics. In Kurt Engesser, Dov M. Gabbay, and Daniel Lehmann, editors, *Handbook of Quantum Logic and Quantum Structures*, pages 261–323. Elsevier, Amsterdam, 2009. doi:10.1016/B978-0-444-52869-8.50010-4.

3 Marcelo Aguiar, Mariana Haim, and Ignacio López Franco. Monads on higher monoidal categories. *Applied Categorical Structures*, 26(3):413–458, June 2018. doi:10.1007/s10485-017-9497-8.

4 Bruce Bartlett, Christopher L. Douglas, Christopher J. Schommer-Pries, and Jamie Vicary. Modular categories as representations of the 3-dimensional bordism 2-category, 2015. arXiv:1509.06811.

5 Jean Bénabou. Distributors at work. *Lecture notes written by Thomas Streicher*, 11, 2000.

6 Guillaume Boisseau and Jeremy Gibbons. What you needa know about yoneda: Profunctor optics and the yoneda lemma (functional pearl). *Proceedings of the ACM on Programming Languages*, 2(ICFP):1–27, 2018.

7 Guillaume Boisseau, Chad Nester, and Mario Román. Cornering optics. In *Proceedings Fifth International Conference on Applied Category Theory, ACT 2022, Glasgow, United Kingdom, 18-22 July 2022*, volume abs/2205.00842, 2022. doi:10.48550/arXiv.2205.00842.

8 Filippo Bonchi, Robin Piedeleu, Pawel Sobocinski, and Fabio Zanasi. Graphical affine algebra. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*, pages 1–12. IEEE, 2019. doi:10.1109/LICS.2019.8785877.

9 Filippo Bonchi, Jens Seeber, and Pawel Sobocinski. Graphical conjunctive queries. In Dan R. Ghica and Achim Jung, editors, *27th EACSL Annual Conference on Computer Science Logic, CSL 2018, September 4-7, 2018, Birmingham, UK*, volume 119 of *LIPIcs*, pages 13:1–13:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.CSL.2018.13.

10 Thomas Booker and Ross Street. Tannaka duality and convolution for duoidal categories. *Theory and Applications of Categories*, 28(6):166–205, 2013.

11 Kenta Cho and Bart Jacobs. Disintegration and Bayesian Inversion via String Diagrams. *Mathematical Structures in Computer Science*, pages 1–34, March 2019. doi:10.1017/S0960129518000488.

12 Bryce Clarke, Derek Elkins, Jeremy Gibbons, Fosco Loregiàn, Bartosz Milewski, Emily Pillmore, and Mario Román. Profunctor optics, a categorical update. *CoRR*, abs/2001.07488, 2020. arXiv:2001.07488.

13 J. Robin B. Cockett and Stephen Lack. Restriction categories I: categories of partial maps. *Theoretical Computer Science*, 270(1-2):223–259, 2002. doi:10.1016/S0304-3975(00)00382-0.

14 J. Robin B. Cockett and Craig A. Pastro. The logic of message-passing. *Sci. Comput. Program.*, 74(8):498–533, 2009. doi:10.1016/j.scico.2007.11.005.

15 J. Robin B. Cockett and Robert A. G. Seely. Weakly distributive categories. *Journal of Pure and Applied Algebra*, 114(2):133–173, 1997.

16 Bob Coecke, Tobias Fritz, and Robert W. Spekkens. A mathematical theory of resources. *Inf. Comput.*, 250:59–86, 2016. doi:10.1016/j.ic.2016.02.008.

17 Geoffrey S. H. Cruttwell, Bruno Gavranović, Neil Ghani, Paul Wilson, and Fabio Zanasi. Categorical foundations of gradient-based learning. In *European Symposium on Programming*, pages 1–28. Springer, Cham, 2022.

18 Brian Day. On closed categories of functors. In *Reports of the Midwest Category Seminar IV*, volume 137, pages 1–38, Berlin, Heidelberg, 1970. Springer Berlin Heidelberg. doi:10.1007/BFb0060438.

19 Elena Di Lavore, Giovanni de Felice, and Mario Román. Monoidal streams for dataflow programming. In *Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '22, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3531130.3533365.

20 Matt Earnshaw, James Hefford, and Mario Román. The produoidal algebra of process decomposition, 2023. arXiv:2301.11867.

21 Matthew Earnshaw and Pawel Sobociński. Regular Monoidal Languages. In Stefan Szeider, Robert Ganian, and Alexandra Silva, editors, *47th International Symposium on Mathematical Foundations of Computer Science (MFCS 2022)*, volume 241 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 44:1–44:14, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.MFCS.2022.44.

22 Brendan Fong and Michael Johnson. Lenses and learners. *arXiv preprint*, 2019. arXiv:1903.03671.

23 J. Nathan Foster, Michael B. Greenwald, Jonathan T. Moore, Benjamin C. Pierce, and Alan Schmitt. Combinators for bidirectional tree transformations: A linguistic approach to the view-update problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 29(3):17–es, 2007.

24 Tobias Fritz. A synthetic approach to Markov kernels, conditional independence and theorems on sufficient statistics. *Advances in Mathematics*, 370:107239, 2020. arXiv:1908.07021.

25 Richard Garner and Ignacio López Franco. Commutativity. *Journal of Pure and Applied Algebra*, 220(5):1707–1751, 2016.

26 Simon J. Gay and Malcolm Hole. Types and subtypes for client-server interactions. In S. Doaitse Swierstra, editor, *Programming Languages and Systems, 8th European Symposium on Programming, ESOP'99, Held as Part of the European Joint Conferences on the Theory and Practice of Software, ETAPS'99, Amsterdam, The Netherlands, 22-28 March, 1999, Proceedings*, volume 1576 of *Lecture Notes in Computer Science*, pages 74–90. Springer, 1999. doi:10.1007/3-540-49099-X_6.

27 Neil Ghani, Jules Hedges, Viktor Winschel, and Philipp Zahn. Compositional game theory. In Anuj Dawar and Erich Grädel, editors, *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 472–481. ACM, 2018. doi:10.1145/3209108.3209165.

28 René Guitart. Tenseurs et machines. *Cahiers de topologie et géométrie différentielle catégoriques*, 21(1):5–62, 1980. URL: http://www.numdam.org/item/CTGDC_1980__21_1_5_0/.

29 Masahito Hasegawa. *Models of sharing graphs: a categorical semantics of let and letrec*. PhD thesis, University of Edinburgh, UK, 1997. URL: http://hdl.handle.net/1842/15001.

30 Jules Hedges. Coherence for lenses and open games. *arXiv preprint*, 2017. arXiv:1704.02230.

31 James Hefford and Cole Comfort. Coend optics for quantum combs. *arXiv preprint*, 2022. arXiv:2205.09027, doi:10.48550/ARXIV.2205.09027.

32 James Hefford and Aleks Kissinger. On the pre- and promonoidal structure of spacetime. *arXiv preprint*, 2022. doi:10.48550/arXiv.2206.09678.

33 Chris Heunen and Bart Jacobs. Arrows, like monads, are monoids. In Stephen D. Brookes and Michael W. Mislove, editors, *Proceedings of the 22nd Annual Conference on Mathematical Foundations of Programming Semantics, MFPS 2006, Genova, Italy, May 23-27, 2006*, volume 158 of *Electronic Notes in Theoretical Computer Science*, pages 219–236. Elsevier, 2006. doi:10.1016/j.entcs.2006.04.012.

34 Chris Heunen and Jesse Sigal. Duoidally enriched Freyd categories. *arXiv preprint*, 2023. arXiv:2301.05162.

35 Kohei Honda. Types for dyadic interaction. In Eike Best, editor, *CONCUR '93, 4th International Conference on Concurrency Theory, Hildesheim, Germany, August 23-26, 1993, Proceedings*, volume 715 of *Lecture Notes in Computer Science*, pages 509–523. Springer, 1993. doi:10.1007/3-540-57208-2_35.

36 Kohei Honda, Nobuko Yoshida, and Marco Carbone. Multiparty asynchronous session types. In George C. Necula and Philip Wadler, editors, *Proceedings of the 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2008, San Francisco, California, USA, January 7-12, 2008*, pages 273–284. ACM, 2008. doi:10.1145/1328438.1328472.

37 John Hughes. Generalising monads to arrows. *Science of Computer Programming*, 37(1-3):67–111, 2000. doi:10.1016/S0167-6423(99)00023-4.

38 Hans Hüttel, Ivan Lanese, Vasco T. Vasconcelos, Luís Caires, Marco Carbone, Pierre-Malo Deniélou, Dimitris Mostrous, Luca Padovani, António Ravara, Emilio Tuosto, Hugo Torres Vieira, and Gianluigi Zavattaro. Foundations of session types and behavioural contracts. *ACM Comput. Surv.*, 49(1):3:1–3:36, 2016. doi:10.1145/2873052.

39 Michael Johnson, Robert Rosebrugh, and Richard J. Wood. Lenses, fibrations and universal translations. *Mathematical structures in computer science*, 22(1):25–42, 2012.

40 André Joyal and Ross Street. The geometry of tensor calculus, I. *Advances in Mathematics*, 88(1):55–112, 1991. doi:10.1016/0001-8708(91)90003-P.

41 Aleks Kissinger and Sander Uijlen. A categorical semantics for causal structure. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–12. IEEE Computer Society, 2017. doi:10.1109/LICS.2017.8005095.

42 Edward Kmett. lens library, version 4.16. *Hackage https://hackage. haskell. org/package/lens-4.16*, 2018, 2012.

43 Naoki Kobayashi, Benjamin C. Pierce, and David N. Turner. Linearity and the pi-calculus. In Hans-Juergen Boehm and Guy L. Steele Jr., editors, *Conference Record of POPL'96: The 23rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Papers Presented at the Symposium, St. Petersburg Beach, Florida, USA, January 21-24, 1996*, pages 358–371. ACM Press, 1996. doi:10.1145/237721.237804.

44 Fosco Loregian. *(Co)end Calculus*. London Mathematical Society Lecture Note Series. Cambridge University Press, 2021. doi:10.1017/9781108778657.

45 Paul-André Melliès and Noam Zeilberger. Parsing as a Lifting Problem and the Chomsky-Schützenberger Representation Theorem. In *MFPS 2022-38th conference on Mathematical Foundations for Programming Semantics*, 2022.

46 Eugenio Moggi. Notions of computation and monads. *Inf. Comput.*, 93(1):55–92, 1991. doi:10.1016/0890-5401(91)90052-4.

47 David Jaz Myers. String diagrams for double categories and equipments, 2016. doi:10.48550/arXiv.1612.02762.

48 Chad Nester. Concurrent Process Histories and Resource Transducers. *Logical Methods in Computer Science*, Volume 19, Issue 1, January 2023. doi:10.46298/lmcs-19(1:7)2023.

49 Nelson Niu and David I. Spivak. Polynomial functors: A general theory of interaction. *In preparation*, 2022.

50 Craig Pastro and Ross Street. Doubles for Monoidal Categories. *arXiv preprint*, 2007. arXiv:0711.1859.

51 Ross Paterson. A new notation for arrows. In Benjamin C. Pierce, editor, *Proceedings of the Sixth ACM SIGPLAN International Conference on Functional Programming (ICFP '01), Firenze (Florence), Italy, September 3-5, 2001*, pages 229–240. ACM, 2001. doi:10.1145/507635.507664.

52 Evan Patterson, David I. Spivak, and Dmitry Vagner. Wiring diagrams as normal forms for computing in symmetric monoidal categories. *Electronic Proceedings in Theoretical Computer Science*, pages 49–64, February 2021.

53 Matthew Pickering, Jeremy Gibbons, and Nicolas Wu. Profunctor optics: Modular data accessors. *Art Sci. Eng. Program.*, 1(2):7, 2017. doi:10.22152/programming-journal.org/2017/1/7.

54 Benjamin C. Pierce and Davide Sangiorgi. Typing and subtyping for mobile processes. In *Proceedings of the Eighth Annual Symposium on Logic in Computer Science (LICS '93), Montreal, Canada, June 19-23, 1993*, pages 376–385. IEEE Computer Society, 1993. doi:10.1109/LICS.1993.287570.

55 J. Postel. Transmission control protocol. RFC 793, RFC Editor, September 1981. doi:10.17487/RFC0793.

56 Mitchell Riley. Categories of Optics. *arXiv preprint*, 2018. arXiv:1809.00738.

57 Mario Román. Comb Diagrams for Discrete-Time Feedback. *CoRR*, abs/2003.06214, 2020. arXiv:2003.06214.

58 Mario Román. Promonads and string diagrams for effectful categories. In *ACT '22: Applied Category Theory, Glasgow, United Kingdom, 18–22 July, 2022*, volume abs/2205.07664, 2022. doi:10.48550/arXiv.2205.07664.

59 Mario Román. Open diagrams via coend calculus. *Electronic Proceedings in Theoretical Computer Science*, 333:65–78, February 2021. doi:10.4204/eptcs.333.5.

60 Davide Sangiorgi and David Walker. *The Pi-Calculus – A theory of mobile processes*. Cambridge University Press, 2001.

61 Patrick Schultz, David I. Spivak, and Christina Vasilakopoulou. Dynamical systems and sheaves. *Applied Categorical Structures*, 28(1):1–57, 2020.

62 Brandon T. Shapiro and David I. Spivak. Duoidal structures for compositional dependence. *arXiv preprint*, 2022. arXiv:2210.01962.

63 Michael Shulman. Categorical logic from a categorical point of view. *Available on the web*, 2016. URL: https://mikeshulman.github.io/catlog/catlog.pdf.

64 Michael Shulman. Duoidal category (nlab entry), section 2, 2017. , Last accessed on 2022-12-14. URL: https://ncatlab.org/nlab/show/duoidal+category.

65 David I. Spivak. The operad of wiring diagrams: formalizing a graphical language for databases, recursion, and plug-and-play circuits. *CoRR*, abs/1305.0297, 2013. arXiv:1305.0297.

66 Ross Street. Monoidal categories in, and linking, geometry and algebra. *Bulletin of the Belgian Mathematical Society-Simon Stevin*, 19(5):769–820, 2012.

67 André Videla and Matteo Capucci. Lenses for composable servers. *CoRR*, abs/2203.15633, 2022. doi:10.48550/arXiv.2203.15633.

Extensions and Limits of the Specker-Blatter Theorem

Eldar Fischer
Faculty of Computer Science, Technion – Israel Institute of Technology, Haifa, Israel

Johann A. Makowsky
Faculty of Computer Science, Technion – Israel Institute of Technology, Haifa, Israel

Abstract

The original Specker-Blatter Theorem (1983) was formulated for classes of structures $\mathcal{C}$ of one or several binary relations definable in Monadic Second Order Logic MSOL. It states that the number of such structures on the set $[n]$ is modularly C-finite (MC-finite). In previous work we extended this to structures definable in CMSOL, MSOL extended with modular counting quantifiers. The first author also showed that the Specker-Blatter Theorem does not hold for one quaternary relation (2003).

If the vocabulary allows a constant symbol c, there are n possible interpretations on $[n]$ for c. We say that a constant c is *hard-wired* if c is always interpreted by the same element $j \in [n]$. In this paper we show:

- **(i)** The Specker-Blatter Theorem also holds for CMSOL when hard-wired constants are allowed. The proof method of Specker and Blatter does not work in this case.
- **(ii)** The Specker-Blatter Theorem does not hold already for $\mathcal{C}$ with one ternary relation definable in First Order Logic FOL. This was left open since 1983.

Using hard-wired constants allows us to show MC-finiteness of counting functions of various restricted partition functions which were not known to be MC-finite till now. Among them we have the restricted Bell numbers $B_{r,A}$, restricted Stirling numbers of the second kind $S_{r,A}$ or restricted Lah-numbers $L_{r,A}$. Here r is an non-negative integer and A is an ultimately periodic set of non-negative integers.

2012 ACM Subject Classification Theory of computation → Finite Model Theory; Mathematics of computing → Enumeration

Keywords and phrases Specker-Blatter Theorem, Monadic Second Order Logic, MC-finiteness

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.26

Related Version *Full Version*: `https://arxiv.org/abs/2206.12135`

Funding *Eldar Fischer*: Israel Science Foundation ISF 879/22

1 Introduction

A sequence of natural numbers $s(n)$ is *C-finite* if it satisfies a linear recurrence relation with constant coefficients. $s(n)$ is MC-finite if it satisfies a linear recurrence relation with constant coefficients modulo m for each m separately. A C-finite sequence $s(n)$ must have limited growth: $s(n) \leq 2^{cn}$ for some constant c. No such bound exists for MC-finite sequences: for every monotone increasing sequence $s(n)$ the sequence $s'(n) = n!s(n)$ is MC-finite.

A typical example of a C-finite sequence is the sequence $f(n)$ of Fibonacci numbers. A typical example of an MC-finite sequence which is not C-finite is the sequence $B(n)$ of Bell numbers. The Bell number $B(n)$ counts the number of partitions of the set $[n]$ of the numbers $\{1, 2, \ldots, n\}$. Let $Eq(n)$ be number of equivalence relations over $[n]$. Clearly, $B(n) = Eq(n)$. Let $Eq_2(n)$ be the number of equivalence relations on $[n]$ with exactly two equivalence classes of the same size. $Eq_2(n)$ is not MC-finite since the value of $Eq_2(n)$ is odd iff n is an even power of 2, see [3].

In [23] G. Pfeiffer discusses counting other transitive relations besides $Eq(n)$, in particular, partial orders $PO(n)$, quasi-orders (aka preorders) $QO(n)$ and just transitive relations $Tr(n)$. Using a growth argument one can see that none of these functions is C-finite. It follows directly from the Specker-Blatter Theorem stated below, see Corollary 2, that $PO(n)$, $QO(n)$ and $Tr(n)$ are MC-finite. However, to the best of our knowledge, this has not been stated in the literature. This may be due to the fact that no explicit formulas for these functions are known. The Specker-Blatter Theorem establishes MC-finiteness even in the absence of explicit formulas. It derives MC-finiteness solely from the assumption that $\mathcal{C}$ is definable in Monadic Second Order Logic (MSOL), or in MSOL augmented by modular counting quantifiers (CMSOL).

The present paper grew out of our study of modular recurrence relations for restricted partition functions, [11]. We provide a short review of the Specker-Blatter Theorem, and show how to extend its applicability by extending the allowed vocabulary to include constants with a fixed interpretation ("hard-wired"). The reduction allowing this extension can be made to work in the other direction. Using it we also close the gap between the Specker-Blatter Theorem and its known limits, left open in [12], by constructing am FOL statement over a single ternary relation for which the theorem does not hold.

Formal definitions with more examples and details about C-finite and MC-finite sequences are given in Section 6.

2 Background in logic

We generally follow the notation of [8], and assume basic knowledge of model theory. Standard texts for Finite Model Theory are [8, 21]. In the following, we always refer to a set $\bar{R} = \{R_1, \ldots, R_{\ell_{\bar{R}}}\}$ of distinct binary relation symbols, a set $\bar{a} = \{a_1, \ldots, a_{\ell_{\bar{a}}}\}$ of distinct constant symbols, and so on. By $a \in \bar{a}$ we mean that there exists $1 \leq i \leq \ell_{\bar{a}}$ for which $a = a_i$. We also use the shorthand $[n] = \{1, \ldots, n\}$.

Let $\tau = \bar{R} \cup \bar{a}$ be a vocabulary, i.e., a set of non-logical constants. We denote by FOL(τ) the set of first order formulas with its non-logical constants in τ. If τ is clear from the context, we omit it. We denote by MSOL(τ) the set of Monadic Second Order Logic, obtained from FOL by allowing unary relation variables and quantification over them. The logic CMSOL is obtained from MSOL by allowing also quantification of the form $\mathrm{C}_{m,a} x \phi(x)$, which are interpreted by

$$\mathcal{A} \models \mathrm{C}_{m,a} x \phi(x) \text{ iff } |\{a \in A : \phi(a)\}| \equiv a \mod m.$$

In the following we will be interested in the set of models of a logic sentence ϕ over a vocabulary τ whose universe is $[n]$ for any natural number n. We denote this set by

$$\mathcal{C}_\phi = \{\mathfrak{M} = ([n], A_1, \ldots, A_m) : n \in \mathbb{N}, A_i \in [n]^{\rho_i}, \mathfrak{M} \models \phi\}.$$

3 The original Specker-Blatter Theorem

Let ϕ_E be the formula in First Order Logic (FOL) which says that $E(x, y)$ is an equivalence relation. $Eq(n)$ can be written as

$$Eq(n) = |\{E \subseteq [n]^2 : ([n], E) \models \phi_E\}|.$$

$PO(n)$, $QO(n)$ and $Tr(n)$ can be written in a similar way.

The original Specker-Blatter Theorem from 1981, [1, 2, 3, 26], gives a general criterion for certain integer sequences to be MC-finite. Let $\bar{R} = \{R_1, \ldots, R_{\ell_{\bar{R}}}\}$ be a finite set of relation symbols of arities $\rho_1, \ldots, \rho_{\ell_{\bar{R}}}$ respectively, and ϕ be a formula of Monadic Second Order Logic (MSOL) using relation symbols from R without free variables.

Let $Sp_\phi(n)$ be the number of ways we can interpret the relation symbols in R on $[n]$ such that the resulting structures where A_i is the interpretation of R_i satisfies ϕ. Formally

$$Sp_\phi(n) = |\{A_i \subseteq [n]^{\rho_i}, i \leq m : ([n], A_1 \ldots, A_m) \models \phi\}|.$$

▶ **Theorem 1** (Specker-Blatter). *Let $\bar{R}$ be a finite set of binary relations and ϕ be a formula of* MSOL$(\bar{R})$ *using relation symbols in $\bar{R}$. Then the sequence $Sp_\phi(n)$ is MC-finite.*

▶ **Corollary 2.** *The sequences counting the number of partial orders $PO(n)$, quasi-orders $QO(n)$, and transitive relations $Tr(n)$ on $[n]$, are MC-finite.*

The idea behind the proof of the Specker-Blatter theorem consists of two parts, both of which use the assertion that $\tau = \bar{R}$ contains only binary relation symbols. Unary symbols can also be incorporated, since these can be simulated with binary symbols in a way that preserves the number of satisfying models.

The first part is combinatorial and applies to any family $\mathcal{C}$ of structures over the vocabulary τ satisfying a property that we outline below. For such a family, we let $Sp_{\mathcal{C}}(n)$ be the number of members of $\mathcal{C}$ whose universe is $[n]$. In particular, $Sp_\phi(n)$ is just a shorthand for $Sp_{\mathcal{C}_\phi}(n)$.

A pointed $\bar{R}$-structure is an $\bar{R}$-structure $\mathcal{A} = ([n], A_1, \ldots, A_\mu, a)$ with an additional distinguished point $a \in [n]$. Given a pointed $\bar{R}$-structure $\mathcal{A}_1$ with universe $[n_1]$ and an $\bar{R}$-structure $\mathcal{A}_2$ with universe $[n_2]$ we define $\mathcal{A} = Subst(\mathcal{A}_1, a, \mathcal{A}_2)$ as follows:

- **(i)** The universe A of $\mathcal{A}$ is the disjoint union of A_1 and A_2 with the point a removed. It can be assumed to be the set $[n_1 + n_2 - 1]$.
- **(ii)** The binary relations are defined such that $\mathcal{A}_2$ is a module in $\mathcal{A}$, i.e., for $u \in A_1 \setminus \{a\}$ and $v \in A_2$ and $R \in \bar{R}$, the relation $R(u, v)$ holds in $\mathcal{A} = Subst(\mathcal{A}_1, a, \mathcal{A}_2)$ iff $R(u, a)$ holds in $\mathcal{A}_1$. For $u, v \in A_1 \setminus \{a\}$ (respectively $u, v \in A_2$), $R(u, v)$ holds in $\mathcal{A}$ iff it holds in $\mathcal{A}_1$ (respectively $\mathcal{A}_2$).

By using an arbitrary enumeration of all possible pointed $\bar{R}$-structures and all possible (non-pointed) $\bar{R}$-structures, we construct an $\mathbb{N} \times \mathbb{N}$ matrix $M_{\mathcal{C}}$ over $\{0, 1\}$, by setting for every i and j the value $M_{\mathcal{C}}(i, j)$ to be the indicator as to whether the substitution of the j'th structure in the i'th pointed structure results in a member of $\mathcal{C}$. The main combinatorial part is the following.

▶ **Theorem 3** (Specker-Blatter, combinatorial version). *Let $\bar{R}$ be a finite set of binary relations and $\mathcal{C}$ be a class of finite $\bar{R}$-structures whose substitution rank is finite under $\mathbb{Z}_{p^q}$ for any prime number p and $q \in \mathbb{N}$. Then the sequence $Sp_{\mathcal{C}}(n)$ is MC-finite.*

The above applies to an uncountable number of families $\mathcal{C}$. Theorem 1 follows from it by the following lemma, which forms the second part of the original proof:

▶ **Lemma 4.** *Let $\bar{R}$ be a finite set of binary relations and $\mathcal{C}$ be a finite class of R-structures defined by an R-sentences ϕ in* MSOL. *Then the substitution rank of $\mathcal{C}$ is finite.*

In [15] it is shown that the lemma still holds if MSOL is replaced by CMSOL. On the other hand, when considering relations of arity higher than 2, the substitution operation is no longer well-defined as it is written here. As it later turned out, this is not a merely technical limitation, but an essential one.

Also, it is not clear how to handle hard-wired constant in the definition of the substitution operation. In this paper, instead of incorporating the hard-wired constants directly into the original mechanism, we show a reduction from the question of the original count to a sum of counts over other sentences that do not involve the constants. This approach turns out to be useful also in the other direction, of proving a new *limit* on the Specker-Blatter theorem.

4 Previous limitations and extensions

Limitations and extensions of the Specker-Blatter Theorem have been previously discussed in [15, 14].

It is well known that Eulerian graphs and regular graphs of even degree are not definable in MSOL, but they are definable in CMSOL. In [15], the Specker-Blatter Theorem was shown to hold also for CMSOL. It follows in particular that $Eul(n)$, which counts the number of Eulerian graphs over $[n]$ (i.e. connected graphs all of whose degrees are even), is also MC-finite.

In [13] the first author showed that the Specker-Blatter Theorem does not hold for quaternary relations:

▶ **Theorem 5** (E. Fischer, 2002)**.** *There is an* FOL*-sentence with only one quaternary relation symbol ϕ, such that $Sp_\phi(n)$ is not an MC-sequence.*

The question of whether Specker-Blatter Theorem holds in the presence of ternary relation symbols remained open.

5 Main new results

Due to space constraints, some proofs are deferred to the full version of this paper[1] [17].

The Bell numbers $B(n)$ and the Stirling numbers of the second kind $S_k(n)$ for fixed k can be shown to be MC-finite using the Specker-Blatter Theorem. A. Broder in 1984, [4], introduced the restricted Bell numbers $B_r(n)$ and the restricted Stirling numbers of the second kind $S_{k,r}(n)$. Let $r \in \mathbb{N}^+$. $S_{k,r}(n)$ is defined as the number of set partitions of $[r+n]$ into $k+r$ blocks with the additional condition that the first r elements are in distinct blocks. $B_r(n)$ is defined as

$$B_r(n) = \sum_k S_{k,r}(n).$$

The class of equivalence relations on $[r+n]$ where the first r elements are in different equivalence classes is definable in FOL with one binary relation and r hard-wired constants. The Specker-Blatter Theorem does not directly apply to this case. In [11] it is shown how to circumvent this obstacle in the case of one equivalence relation. It followed that both $S_{k,r}(n)$ and $B_r(n)$ are MC-finite.

In this paper we prove a more general theorem:

▶ **Theorem 6** (Elimination of hard-wired constants)**.**

(i) *Let τ consist of a finite set of (hard-wired) constant symbols $\bar{a}$, unary relations symbols $\bar{U}$, and binary relation symbols $\bar{R}$. For every class $\mathcal{C}$ of τ-structures there exist classes $\mathcal{C}_1, \ldots, \mathcal{C}_r$ of τ'-structures, where τ'-contains only a finite number $r(\bar{a}, \bar{U}, \bar{R})$ of binary relation symbols, such that*

[1] The full version can be downloaded at `https://arxiv.org/abs/2206.12135`.

$$Sp_{\mathcal{C}}(n) = \sum_{i=1}^{r} Sp_{\mathcal{C}_i}(n).$$

Equality here is not modular.

(ii) *Furthermore, if $\mathcal{C}$ is* FOL*-definable (*MSOL*-definable,* CMSOL*-definable), so are the $\mathcal{C}_i$.*

▶ **Corollary 7.** *Let τ consist of a finite set of (hard-wired) constant symbols $\bar{a}$, unary relations symbols $\bar{U}$, and binary relation symbols $\bar{R}$, and let $\mathcal{C}$ be class of finite τ-structures definable in* CMSOL*. Then the sequence $Sp_{\mathcal{C}}(n)$ is MC-finite.*

The proof of Theorem 6 is given in Section 7. In Section 8 we state Theorem 19, which is an extension of Theorem 6 that works for higher arities and other logics, and sketch its proof. The full proof details of Theorem 19 are deferred to [17]. The extension to higher arities is needed for proving Theorem 8 below.

We have seen in Theorem 5 that the Specker-Blatter Theorem does not hold for a single quaternary relation. The question of whether Specker-Blatter Theorem holds in the presence of a single ternary relation symbol remained open. Our second main result here answers this.

▶ **Theorem 8** (Ternary Counter-Example)**.** *There is a* FOL*-sentence ϕ with only one ternary relation symbol (and some lower arity relations), such that $Sp_{\phi}(n)$ is not an MC-sequence.*

The proof of Theorem 8 first produces a sentence ψ which also uses one symbol for a hard-wired constant. This will be shown in Section 9. To construct ϕ without the hard-wired constants, we deploy the aforementioned Theorem 19, which provides a sentence with one ternary relation and several lower arity relations. We can then also eliminate all relations except the ternary one, to arrive at Theorem 31 stated at Section 9, whose proof is deferred to the full version [17]. A sketch thereof is still provided.

We conclude this paper with Section 10, containing a summary and open problems.

6 More details about C-finite and MC-finite sequences of integers

A sequence of integers $s(n)$ is *C-finite*[2] if there are constants $p, q \in \mathbb{N}$ and $c_i \in \mathbb{Z}$, $0 \leq i \leq p-1$ such that for all $n \geq q$ the linear recurrence relation below holds for $s(n)$.

$$s(n+p) = \sum_{i=0}^{p-1} c_i s(n+i).$$

A sequence of integers $s(n)$ is modular C-finite, abbreviated as *MC-finite*, if for every $m \in \mathbb{N}$ there are constants $p_m, q_m \in \mathbb{N}^+$ such that for every $n \geq q_m$ there is a linear recurrence relation

$$s(n+p_m) \equiv \sum_{i=0}^{p_m-1} c_{i,m} s(n+i) \bmod m$$

with constant coefficients $c_{i,m} \in \mathbb{Z}$.

We denote by $s^m(n)$ the sequence $s(n) \bmod m$. Note that the coefficients $c_{i,m}$ and both p_m and q_m generally do depend on m.

[2] These are also called constant-recursive sequences or linear-recursive sequences in the literature.

▶ **Proposition 9.** *The sequence $s(n)$ is MC-finite iff $s^m(n)$ is ultimately periodic for every m.*

Proof. MC-finiteness implies periodicity. The converse is from [24]. ◀

Clearly, if a sequence $s(n)$ is C-finite then it is also MC-finite with $r_m = r$ and $c_{i,m} = c_i$ for all m. The converse is not true as there are uncountably many MC-finite sequences, but only countably many C-finite sequences with integer coefficients, see Proposition 11 below.

▶ **Example 10.**

(i) The Fibonacci sequence is C-finite.

(ii) If $s(n)$ is C-finite then it has at most simple exponential growth. There is $c \in \mathbb{N}^+$ such that $s(n) \leq 2^{cn}$ for all $n \in \mathbb{N}$, see e.g. [9, 19].

(iii) The Bell numbers $B(n)$ are *not C-finite*, but are *MC-finite*.

(iv) Let $f(n)$ be any integer sequence. The sequence $s_1(n) = 2 \cdot f(n)$ is ultimately periodic modulo 2, but not necessarily MC-finite.

(v) Let $g(n)$ be any integer sequence which is not almost everywhere zero. The sequence $s_2(n) = n! \cdot g(n)$ is MC-finite but not C-finite due to its growth.

(vi) The sequence $s_3(n) = \frac{1}{2}\binom{2n}{n}$ is not MC-finite: $s_3(n)$ is odd if and only if n is a power of 2 (Lucas, 1878). A proof may be found in [18, Exercise 5.61] or in [26].

(vii) The Catalan numbers $C(n) = \frac{1}{n+1}\binom{2n}{n}$ are not MC-finite, since $C(n)$ is odd iff n is a Mersenne number, i.e., $n = 2^m - 1$ for some m, see [20, Chapter 13].

(viii) Let p be a prime and $f(n)$ monotone increasing. The sequence $s(n) = p \cdot f(n) + z(n)$, where $z(n)$ is defined to equal 1 if n is a power of p and to equal 0 for any other n, is monotone increasing but not ultimately periodic modulo p, hence not MC-finite.

▶ **Proposition 11.**

(i) *There are uncountably many monotone increasing sequences which are MC-finite, and uncountably many which are not MC-finite.*

(ii) *Almost all integer sequences (under a suitable measure) are not MC-finite.*

Proof. (i) follows from Example 10 (v) and (viii). (ii) follows from almost all integer sequences being being absolutely normal (see [9]); the full proof is deferred to [17]. ◀

7 Proving the reduction

7.1 Introduction

In the following we consider extending the language with "hard-wired" constants. Specifically, assume that we have a class $\mathcal{C}$ that is defined by a sentence ϕ involving a set of constant symbols $\bar{a}$, unary symbols $\bar{U}$ and binary symbols $\bar{R}$. The function $f_{\mathcal{C}}(n)$ is defined as the number of models over the universe $[n + \ell_{\bar{a}}]$ which satisfy ϕ, for which a_i is interpreted as $n + i$ for all $i \in [\ell_{\bar{a}}]$. Note the distinction from the non-hard-wired setting, where we would have had to also count the possible interpretations of the constants.

Our main result is an expression for the function $f_{\mathcal{C}}(n)$ (when constants are allowed) that is based on counting functions for classes that do not utilize constants. We first show this reduction for languages using only unary and binary relations. The reduction preserves many of the common logics, in particular an FOL expression would be reduced to functions involving FOL expressions, and so on. This extends the Specker-Blatter theorem to languages involving hard-wired constants, allowing modular ultimate periodicity proofs of new functions.

In this section we prove Theorem 6. For convenience we state it again as Theorem 12.

▶ **Theorem 12** (Reducing model counts to the case without constants). *For any class $\mathcal{C}$ defined by an FOL (resp. MSOL, CMSOL) sentence involving a set of constant symbols $\bar{a}$, unary symbols $\bar{U}$ and binary symbols $\bar{R}$, there exist classes $\mathcal{C}_1, \ldots, \mathcal{C}_r$ (where r depends on the original language), definable by FOL (resp. MSOL, CMSOL) sentences involving $\bar{U}'$ (which contains $\bar{U}$), $\bar{R}$ and no constants, satisfying $f_{\mathcal{C}}(n) = \sum_{i=1}^{r} f_{\mathcal{C}_i}(n)$ for all $n \in \mathbb{N}$.*

The following is the immediate corollary it produces for the Specker-Blatter Theorem, which is a restatement of Corollary 2.

▶ **Corollary 13** (Extended Specker-Blatter Theorem). *For a class $\mathcal{C}$ definable in CMSOL with (hard-wired) constants, unary and binary relation symbols only, the function $f_{\mathcal{C}}$ is MC-finite.*

Theorem 12 is proved by induction over the number of constants. The basis, $\ell_{\bar{a}} = 0$, is trivial (with $\bar{U}' = \bar{U}$, $r = 1$ and $\mathcal{C}_1 = \mathcal{C}$). The induction step is provided by the following.

▶ **Lemma 14** (Removing a single constant). *For any class $\mathcal{C}$ defined by an FOL (resp. MSOL, CMSOL) sentence involving a set of constant symbols $\bar{a}$ with $\ell_{\bar{a}} > 0$, unary symbols $\bar{U}$ and binary symbols $\bar{R}$, there exist classes $\mathcal{C}_1, \ldots, \mathcal{C}_r$ (where r depends on the original language), definable by FOL (resp. MSOL, CMSOL) sentences over the language $(\bar{a}', \bar{U}', \bar{R}')$, where $\bar{a}' = \bar{a} \setminus \{a_{\ell_{\bar{a}}}\}$, $\bar{U}' = \bar{U} \cup \bar{I} \cup \bar{O}$ where $\ell_{\bar{I}} = \ell_{\bar{O}} = \ell_{\bar{R}}$, and $\bar{R}' = \bar{R}$, satisfying $f_{\mathcal{C}}(n) = \sum_{i=1}^{r} f_{\mathcal{C}_i}(n)$ for all $n \in \mathbb{N}$.*

The main idea in the proof of this lemma is to encode the "interaction" of the constant $a_{\ell_{\bar{a}}}$ with the rest of the universe using the additional unary relations. For every $i \in [\ell_{\bar{R}}]$, we will use the new relation I_i to hold every $x \neq a_{\ell_{\bar{a}}}$ for which (x, a) was in R_i, and the relation O_i to hold every $x \neq a_{\ell_{\bar{a}}}$ for which (a, x) was in R_i.

We cannot directly keep track whether (a, a) was in R_i, or whether a was in U_i for $i \in [\ell_{\bar{U}}]$, so we count the number of models for each of these options separately. This sets $r = 2^{\ell_{\bar{U}} + \ell_{\bar{R}}}$. Instead of a running index, we index each such option with a set $\mathfrak{U} \subseteq [\ell_{\bar{U}}]$ denoting which of the relations in $\bar{U}$ include the constant to be removed $a = a_{\ell_{\bar{a}}}$, and a set $\mathfrak{R} \subseteq [\ell_{\bar{R}}]$ denoting which of the relations in $\bar{R}$ include (a, a). Using these we can define the case where a model $\mathfrak{N}$ over the language $(\bar{a}', \bar{U}', \bar{R})$ with universe $[n + \ell_{\bar{a}} - 1]$ corresponds (along with $\mathfrak{U}$ and $\mathfrak{R}$) to an "original model" $\mathfrak{M}$ with universe $[n + \ell_{\bar{a}}]$ over the original language.

▶ **Definition 15.** *Given a model $\mathfrak{M}$ over the language $(\bar{a}, \bar{U}, \bar{R})$ with universe $[n + \ell_{\bar{a}}]$, a model $\mathfrak{N}$ over the language $(\bar{a}', \bar{U}', \bar{R})$ with universe $[n + \ell_{\bar{a}} - 1]$, and sets $\mathfrak{U} \subseteq [\ell_{\bar{U}}]$ and $\mathfrak{R} \subseteq [\ell_{\bar{R}}]$, where (as always) in both models every constant a_i is interpreted to be $n + i$, we say that $(\mathfrak{N}, \mathfrak{U}, \mathfrak{R})$* correspond *to $\mathfrak{M}$ if the following hold.*

- *For every $U \in \bar{U}$ and $x \in [n + \ell_{\bar{a}} - 1]$, we have $\mathfrak{N} \models U(x)$ if and only if $\mathfrak{M} \models U(x)$.*
- *For every $i \in [\ell_{\bar{U}}]$, we have $i \in \mathfrak{U}$ if and only if $\mathfrak{M} \models U_i(a)$.*
- *For every $R \in \bar{R}$ and $x, y \in [n + \ell_{\bar{a}} - 1]$, we have $\mathfrak{N} \models R(x, y)$ if and only if $\mathfrak{M} \models R(x, y)$.*
- *For every $i \in [\ell_{\bar{R}}]$ and $x \in [n + \ell_{\bar{a}} - 1]$, we have $\mathfrak{N} \models I_i(x)$ if and only if $\mathfrak{M} \models R_i(x, a)$.*
- *For every $i \in [\ell_{\bar{R}}]$ and $x \in [n + \ell_{\bar{a}} - 1]$, we have $\mathfrak{N} \models O_i(x)$ if and only if $\mathfrak{M} \models R_i(a, x)$.*
- *For every $i \in [\ell_{\bar{R}}]$, we have $i \in \mathfrak{R}$ if and only if $\mathfrak{M} \models R_i(a, a)$.*

It is important to note, for the purpose of counting, the following observation.

▶ **Observation 16.** *Definition 15 provides a bijection between the set of possible models $\mathfrak{M}$ over the universe $[n + \ell_{\bar{a}}]$ (where the constants are interpreted as in Definition 15), and the set of possible triples $(\mathfrak{N}, \mathfrak{U}, \mathfrak{R})$ where $\mathfrak{N}$ is a model over $[n + \ell_{\bar{a}} - 1]$ (where the constants are interpreted as in Definition 15) and $\mathfrak{U} \subseteq [\ell_{\bar{U}}]$ and $\mathfrak{R} \subseteq [\ell_{\bar{R}}]$.*

Suppose we are given an expression $\phi(\bar{x})$ where $\bar{x} = \{x_1, \ldots, x_{\ell_{\bar{x}}}\}$ is a set of variable symbols over the language $(\bar{a}, \bar{U}, \bar{R})$, as well as a set $\mathfrak{U} \subseteq [\ell_{\bar{U}}]$ and a set $\mathfrak{R} \subseteq [\ell_{\bar{R}}]$. We will construct, by induction over the structure of ϕ, several expressions, where one of which is an expression $\phi'_{\mathfrak{U},\mathfrak{R}}(\bar{x})$ over the language $(\bar{a}', \bar{U}', \bar{R})$. It will be constructed so that for any $\mathfrak{M}$ over the language $(\bar{a}, \bar{U}, \bar{R})$ with universe $[n+\ell_{\bar{a}}]$ and $\mathfrak{N}$ over the language $(\bar{a}', \bar{U}', \bar{R})$ with universe $[n + \ell_{\bar{a}} - 1]$, where $(\mathfrak{N}, \mathfrak{U}, \mathfrak{R})$ correspond to $\mathfrak{M}$, and any fixing of $x_1, \ldots, x_{\ell_{\bar{x}}} \in [n + \ell_{\bar{a}} - 1]$, we will have $\mathfrak{M} \models \phi(\bar{x})$ if and only if $\mathfrak{N} \models \phi'_{\mathfrak{U},\mathfrak{R}}(\bar{x})$.

Lemma 14 then immediately follows from the case $\ell_{\bar{x}} = 0$ (i.e. where ϕ is a sentence). To be precise, for a class $\mathcal{C}$ defined by a sentence ϕ over the language $(\bar{a}, \bar{U}, \bar{R})$, we obtain $f_{\mathcal{C}}(n) = \sum_{\mathfrak{U} \subseteq [\ell_{\bar{U}}], \mathfrak{R} \subseteq [\ell_{\bar{R}}]} f_{C_{\mathfrak{U},\mathfrak{R}}}(n)$, where $\mathcal{C}_{\mathfrak{U},\mathfrak{R}}$ is the class respectively defined by $\phi'_{\mathfrak{U},\mathfrak{R}}(\bar{x})$ over the language $(\bar{a}', \bar{U}', \bar{R})$.

To sustain the induction, the above will not be enough. This is because we need to account under the model $\mathfrak{N}$ also for the case where some variables are "assigned the value $a = a_{\ell_{\bar{a}}}$", a value which does not exist in its universe (it exists only in that of $\mathfrak{M}$). We henceforth consider also a set $\mathfrak{X} \subseteq [\ell_{\bar{x}}]$, and denote the set of variable symbols $x_{\mathfrak{X}} = \{x_i : i \in \mathfrak{X}\}$. In our induction we will construct the expressions $\phi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}})$, where $\phi'_{\mathfrak{U},\mathfrak{R}}(\bar{x})$ is just the special case $\phi'_{\emptyset,\mathfrak{U},\mathfrak{R}}(\bar{x})$. With models $\mathfrak{M}$ and $\mathfrak{N}$ as above and a fixing of the variables in $\bar{x} \setminus x_{\mathfrak{X}}$, denote by $\bar{x}_{\mathfrak{X}\to a}$ the completion of this fixing to all of $\bar{x}$ that is obtained by fixing x_i to be equal to a for all $i \in \mathfrak{X}$. We will then have $\mathfrak{M} \models \phi(\bar{x}_{\mathfrak{X}\to a})$ if and only if $\mathfrak{N} \models \phi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}})$.

The rest of this section is concerned with the recursive definition of $\phi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}})$. There is a subsection for the base cases, a subsection for Boolean connectives, and a subsection for each class of quantifiers (first order quantifiers, counting quantifiers, and monadic second order quantifiers). In every construction we argue (at times trivially) that we keep the correspondence invariant, namely that $\mathfrak{M} \models \phi(\bar{x}_{\mathfrak{X}\to a})$ if and only if $\mathfrak{N} \models \phi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}})$ whenever $\mathfrak{M}$ and $(\mathfrak{N}, \mathfrak{U}, \mathfrak{R})$ satisfy the correspondence condition of Definition 15.

7.2 The base constructions

We use the Boolean "true" and "false" statements in the following, so for formality's sake they are also considered as atomic statements here. Clearly, if $\phi(\bar{x})$ is simply the "true" statement $\top$ (respectively the "false" statement $\bot$), then setting $\phi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}})$ to $\top$ (respectively $\bot$) gives us the equivalent statement satisfying the correspondence invariant.

For $i \in [\ell_{\bar{U}}]$ and $j \in [\ell_{\bar{x}}]$, let us now consider the expression $\phi(\bar{x}) = U_i(x_j)$. To produce $\phi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}})$, we partition to cases according to whether $j \in \mathfrak{X}$. In the case where $j \notin \mathfrak{X}$, we simply set $\phi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}}) = U_i(x_j)$ as well, which clearly satisfies the invariant for $(\mathfrak{N}, \mathfrak{U}, \mathfrak{R})$ correlated with $\mathfrak{M}$ (recall that the "if and only if" condition in this case should hold when the value of x_i is in $[n + \ell_{\bar{a}} - 1]$).

Similarly, for $i \in [\ell_{\bar{U}}]$ and $j \in [\ell_{\bar{a}} - 1]$, for the expression $\phi(\bar{x}) = U_i(a_j)$, we produce $\phi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}}) = U_i(a_j)$, noting that in our setting the value of a_j is guaranteed to be equal to $n + j \in [n + \ell_{\bar{a}} - 1]$.

Now consider the expression $\phi(\bar{x}) = U_i(x_j)$ for the case where $x_j \in \mathfrak{X}$. Recall that in this case $\phi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}})$ should not depend on x_j. Moreover, to preserve the invariant for corresponding sets and models, $\phi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}})$ should hold if and only if $U_j(a)$ holds (recall that we use the shorthand $a = a_{\ell_{\bar{a}}}$ throughout). We hence define $\phi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}})$ to be $\top$ ("true") if $i \in \mathfrak{U}$, and define it to be $\bot$ ("false") if $i \notin \mathfrak{U}$.

The remaining case for a unary relation is the expression $\phi(\bar{x}) = U_i(a)$. Again, we define $\phi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}})$ to be $\top$ if $i \in \mathfrak{U}$, and define it to be $\bot$ if $i \notin \mathfrak{U}$.

We now move on to handle the atomic expressions involving a binary relation R_i where $i \in [\ell_{\bar{R}}]$. The first case here is the one analogous to the first case we discussed involving a unary relation. Namely, it is the case where $\phi(\bar{x}) = R_i(x_j, x_k)$ where both $j \notin \mathfrak{X}$ and $k \notin \mathfrak{X}$. In this case we set $\phi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}}) = R_i(x_j, x_k)$, and argue the same argument as above about satisfying the correspondence invariant.

The next four cases we discuss resemble the last two cases we discussed about a unary relation. Namely, these are the cases where $\phi(\bar{x}) = R_i(x_j, x_k)$ with $j, k \in \mathfrak{X}$, $\phi(\bar{x}) = R_i(x_j, a)$ or $\phi(\bar{x}) = R_i(a, x_j)$ with $j \in \mathfrak{X}$, and $\phi(\bar{x}) = R_i(a, a)$. In all these cases the resulting expression should reflect on whether $\mathfrak{M} \models R_i(a, a)$, which for the corresponding $(\mathfrak{N}, \mathfrak{U}, \mathfrak{R})$ is handled by the set $\mathfrak{R}$. We hence set $\phi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}}) = \top$ if $i \in \mathfrak{R}$, and set $\phi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}}) = \bot$ if $i \notin \mathfrak{R}$.

Next we handle the cases where $\phi(\bar{x}) = R_i(x_j, x_k)$ with $j \notin \mathfrak{X}$ and $k \in \mathfrak{X}$, and $\phi(\bar{x}) = R_i(x_j, a)$ with $j \notin \mathfrak{X}$. For both this cases, for the correspondence invariant to follow we need to look at whether $\mathfrak{M} \models R_i(x_j, a)$, where the value of x_j is in $[n + \ell_{\bar{a}} - 1]$. For the corresponding $(\mathfrak{N}, \mathfrak{U}, \mathfrak{R})$ this occurs if and only if $\mathfrak{N} \models I_i(x_j)$, where we recall that I_i is a relation from $\bar{U}' \setminus \bar{U}$. We therefor set $\phi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}}) = I_i(x_j)$ in these cases. Similarly, for the cases $\phi(\bar{x}) = R_i(a_j, x_k)$ and $\phi(\bar{x}) = R_i(a_j, a)$, where $j \in [\ell_{\bar{a}} - 1]$ and $k \in \mathfrak{X}$, we set $\phi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}}) = I_i(a_j)$.

Moving on to the remaining cases for a binary relation, we consider $\phi(\bar{x}) = R_i(x_k, x_j)$ with $j \notin \mathfrak{X}$ and $k \in \mathfrak{X}$, and $\phi(\bar{x}) = R_i(a, x_j)$ with $j \notin \mathfrak{X}$. These are analogous to the cases handled in the last paragraph, only here we use O_i instead of I_i. We set $\phi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}}) = O_i(x_j)$ in these two cases. Finally, for the cases $\phi(\bar{x}) = R_i(x_k, a_j)$ and $\phi(\bar{x}) = R_i(a, a_j)$, where $j \in [\ell_{\bar{a}} - 1]$ and $k \in \mathfrak{X}$, we set $\phi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}}) = O_i(a_j)$.

The final atomic formula to consider is the "builtin relation" of equality. We skip all cases involving only constants (e.g. $a_i = a_j$), since these are equivalent to $\top$ or $\bot$. We also skip cases that are equivalent by the symmetry of the equality relation to those that we discuss.

First, if $\phi(\bar{x})$ is $x_i = x_j$ or $x_i = a_k$ for $i, j \notin \mathfrak{X}$ and $k \in [\ell_{\bar{a}} - 1]$, then since we are dealing with values that are guaranteed to be in $[n + \ell_{\bar{a}} - 1]$ (the universe of $\mathfrak{N}$), we set $\phi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}})$ respectively to $x_i = x_j$ or $x_i = a_k$ as well (so it is "unaltered" from $\phi(\bar{x})$).

On the other hand, if $\phi(\bar{x})$ is $x_i = x_j$ or $x_i = a$ for $i, j \in \mathfrak{X}$, then for the correspondence principle to hold, we need $\mathfrak{N} \models \phi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}})$ to hold if $\mathfrak{M} \models (a = a)$. In other words, we have to set $\phi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}}) = \top$ here.

The final cases are those where $\phi(\bar{x})$ is $x_i = x_j$ or $x_i = a$ for $i \notin \mathfrak{X}$ and $j \in \mathfrak{X}$. For the correspondence principle to hold, we need $\mathfrak{N} \models \phi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}})$ to hold if and only if $\mathfrak{M} \models (x_i = a)$. However, we make here the subtle yet important observation that this should occur for any value that x_i can take from the universe of $\mathfrak{N}$, which does not include a. Therefor, we can (and should) set $\phi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}}) = \bot$ in these cases.

7.3 Boolean connectives

Handling Boolean connectives is the most straightforward part of this construction. For example, suppose that we have $\phi(\bar{x}) = \neg\psi(\bar{x})$ for some expression $\psi(\bar{x})$, for which we have already established (by the induction hypothesis) that $\mathfrak{M} \models \psi(\bar{x}_{\mathfrak{X}\to a})$ if and only if $\mathfrak{N} \models \psi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}})$ whenever $\mathfrak{M}$ and $(\mathfrak{N}, \mathfrak{U}, \mathfrak{R})$ correspond. Here we can clearly set $\phi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}}) = \neg\psi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}})$, and obtain that $\mathfrak{M} \models \phi(\bar{x}_{\mathfrak{X}\to a})$ if and only if $\mathfrak{N} \models \phi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}})$ whenever $\mathfrak{M}$ and $(\mathfrak{N}, \mathfrak{U}, \mathfrak{R})$ correspond.

The same idea and analysis follow for all other Boolean connectives. For example, for the expression $\phi(\bar{x}) = \psi_1(\bar{x}) \wedge \psi_2(\bar{x})$, we set $\phi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}}) = \psi'_{1,\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}}) \wedge \psi'_{2,\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}})$.

7.4 First order quantifiers

To deal with quantifiers over variables, we make some assumptions on the structure of our expressions that can easily be justified by the appropriate variable substitutions. Namely, we require that every quantified variable is quantified only once in the expression, and is not used at all outside the scope of the quantification. In particular, this means that the set $\mathfrak{X}$ that appears in the subscript of our expression cannot contain a reference to the quantified variable.

For notational convenience, when $\phi(\bar{x})$ is our formula, we denote by $x = x_{\ell_{\bar{x}}+1}$ the quantified variable. So the two cases that we consider in this subsection are the existential quantification $\phi(\bar{x}) = \exists_x \psi(\bar{x} \cup \{x\})$ and the universal quantification $\phi(\bar{x}) = \forall_x \psi(\bar{x} \cup \{x\})$, and for both of them we would like to construct a corresponding $\phi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}})$ where $\mathfrak{X} \subseteq [\ell_{\bar{x}}]$.

In the existential case, we want $\mathfrak{N} \models \phi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}})$ to occur whenever there is at least one value of x for which $\mathfrak{M} \models \psi(\bar{x} \cup \{x\})$. For the values of x within $[n + \ell_{\bar{a}} - 1]$, by the induction hypothesis, this is covered by the expression $\exists_x \psi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \cup \{x\} \setminus x_{\mathfrak{X}})$. However, there is one possible value of x not covered in this way, and that is the value $n + \ell_{\bar{a}}$, which we identify with the constant a. But by the induction hypothesis, $\mathfrak{M} \models \psi(\bar{x} \cup \{x\})$ for $x = a$ if and only if $\mathfrak{N} \models \psi'_{\mathfrak{X} \cup \{\ell_{\bar{x}}+1\},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}})$. The combined expression that satisfies the correspondence invariant is hence $\phi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}}) = \exists_x \psi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \cup \{x\} \setminus x_{\mathfrak{X}}) \vee \psi'_{\mathfrak{X} \cup \{\ell_{\bar{x}}+1\},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}})$.

The universal case follows an analogous argument, only here $\mathfrak{M} \models \psi(\bar{x} \cup \{x\})$ needs to hold for all values of x, those in $[n + \ell_{\bar{a}} - 1]$ as well as the value of a. The combined expression is $\phi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}}) = \forall_x \psi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \cup \{x\} \setminus x_{\mathfrak{X}}) \wedge \psi'_{\mathfrak{X} \cup \{\ell_{\bar{x}}+1\},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}})$.

7.5 Modular counting quantifiers

We briefly recall the definition of a modular counting quantifier. Given $\phi(\bar{x}) = \mathrm{C}_x^{r,m} \psi(\bar{x} \cup \{x\})$, this expression is said to hold in $\mathfrak{M}$ for a specific assignment to the variable of $\bar{x}$, if the size of the set $\{x : \mathfrak{M} \models \psi(\bar{x} \cup \{x\})\}$ is congruent to r modulo m. As with the previous subsection, we assume that the quantified variable is not used outside the quantification scope, and that no variable is quantified more than once. We again denote for notational convenience the quantified variable by $x = x_{\ell_{\bar{x}}+1}$, and note that $\mathfrak{X} \subseteq [\ell_{\bar{x}}]$ cannot include a reference to x.

When working with $(\mathfrak{N}, \mathfrak{U}, \mathfrak{R})$ that corresponds to $\mathfrak{M}$, to obtain the original modular count, we have to count the set (satisfying the induction hypothesis) $\{x : \mathfrak{N} \models \psi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \cup \{x\} \setminus x_{\mathfrak{X}})\}$, as well as check whether $\mathfrak{N} \models \psi'_{\mathfrak{X} \cup \{\ell_{\bar{x}}+1\},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}})$ (which if true adds 1 to the count). This gives $(\mathrm{C}_x^{r-1,m} \psi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \cup \{x\} \setminus x_{\mathfrak{X}}) \wedge \psi'_{\mathfrak{X} \cup \{\ell_{\bar{x}}+1\},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}})) \vee (\mathrm{C}_x^{r,m} \psi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \cup \{x\} \setminus x_{\mathfrak{X}}) \wedge \neg \psi'_{\mathfrak{X} \cup \{\ell_{\bar{x}}+1\},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}}))$ as the combined expression for $\phi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}})$.

7.6 Monadic second order quantifiers

Here we deal with quantifiers over unary relations. The cases we cover are the existential quantification $\phi(\bar{x}) = \exists_U \psi(\bar{x})$ and the universal quantification $\phi(\bar{x}) = \forall_U \psi(\bar{x})$, where U is a new unary relation that does not appear in the language $(\bar{a}, \bar{U}, \bar{R})$ of $\phi(\bar{x})$, while being part of the language of $\psi(\bar{x})$. As before, we assume that the quantified relation symbol U appears only in the scope of this quantification, and is not quantified anywhere else, and again denote for convenience $U = U_{\ell_{\bar{U}}+1}$. In particular, when analyzing expressions of the type $\psi'_{\mathfrak{X},\mathfrak{U}',\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}})$, we may allow $\mathfrak{U}'$ to contain $[\ell_{\bar{U}} + 1]$ (the same is not allowed for the expression $\phi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}})$, whose language does not contain U).

Consider now the family of possible models $\mathfrak{M}'$ that extend $\mathfrak{M}$ with an interpretation of the relation U. Now consider $(\mathfrak{N}', \mathfrak{U}', \mathfrak{R}')$ which correspond to $\mathfrak{M}'$, in relation to $(\mathfrak{N}, \mathfrak{U}, \mathfrak{R})$ which correspond to $\mathfrak{M}$. Referring to Definition 15, every relation already appearing in $\bar{U}$ will

have the same interpretation in $\mathfrak{N}$ and $\mathfrak{N}'$. Also, $\mathfrak{R}' = \mathfrak{R}$, since the binary relations are the same in the languages of both models. Additionally, from the definition, the interpretation of $U = U_{\ell_{\bar{U}}+1}$ in $\mathfrak{N}'$ is the restriction of its interpretation in $\mathfrak{M}'$ to $[n + \ell_{\bar{a}} - 1]$. As for the final ingredient $\mathfrak{U}'$, for every $i \in [\ell_{\bar{U}}]$, the condition on whether it is in $\mathfrak{U}$ or in $\mathfrak{U}'$ is the same. However, $\mathfrak{U}'$ may also include $\ell_{\bar{U}} + 1$ according to whether $\mathfrak{M}' \models U(a)$. So considering all possible models $\mathfrak{M}'$, we have two possibilities. Either $\mathfrak{U}' = \mathfrak{U}$, or $\mathfrak{U}' = \mathfrak{U} \cup \{\ell_{\bar{U}} + 1\}$.

We can now construct our expression that corresponds to all models extending $\mathfrak{M}$. For the existential case we have $\phi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}}) = \exists_U \psi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}}) \vee \exists_U \psi'_{\mathfrak{X},\mathfrak{U}\cup\{\ell_{\bar{U}}+1\},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}})$, and for the universal one we have $\phi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}}) = \forall_U \psi'_{\mathfrak{X},\mathfrak{U},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}}) \wedge \forall_U \psi'_{\mathfrak{X},\mathfrak{U}\cup\{\ell_{\bar{U}}+1\},\mathfrak{R}}(\bar{x} \setminus x_{\mathfrak{X}})$.

8 Handling relations of other arities

8.1 Nullary relations and a many-one version of the reduction

Before we consider relations of higher arities, let us show how incorporating nullary ("arity zero") relations can replace the reduction of Theorem 6 into a many-one reduction. That is, instead of a reduction into of the original $Sp_\phi(n)$ into a finite sum $\sum_{i=1}^{r} Sp_{\phi_i}(n)$, we will have a reduction into a single $Sp_{\phi'}(n)$ where ϕ' may also involve nullary relations. Later, when we consider higher arity relations, nullary relations add much needed consistency to the notation.

Formally, for a nullary relation Z, the corresponding atomic formula is $Z()$, and a model $\mathfrak{M}$ over a language that includes Z interprets this formula as either true or false, that is, either $\mathfrak{M} \models Z()$ or $\mathfrak{M} \models \neg Z()$.

Note that nullary relations can be simulated using higher arity relations. To replace a nullary relation Z in the language with a unary relation U (while preserving the model count), the logical expression under discussion should be conjuncted with "$\forall_x \forall_y (U(x) \leftrightarrow U(y))$", and then every instance of "$Z()$" in the formula should be replaced with "$\exists_x U(x)$".

The corresponding reduction theorem is the following.

▶ **Theorem 17** (Many-one reduction to the case without constants). *For any class $\mathcal{C}$ defined by an FOL (resp. MSOL, CMSOL) sentence involving a set of constant symbols $\bar{a}$, nullary symbols $\bar{Z}$, unary symbols $\bar{U}$ and binary symbols $\bar{R}$, there exists a class $\mathcal{C}'$ definable by an FOL (resp. MSOL, CMSOL) sentence involving $\bar{Z}'$, $\bar{U}'$ (which contain $\bar{Z}$ and $\bar{U}$ respectively), $\bar{R}$ and no constants, satisfying $f_{\mathcal{C}}(n) = f_{\mathcal{C}'}(n)$ for all $n \in \mathbb{N}$.*

Also here, the theorem follows from a single constant removal lemma, which is used for an inductive argument over $\ell_{\bar{a}}$.

▶ **Lemma 18** (Removing a single constant in a many-one manner). *For any class $\mathcal{C}$ defined by an FOL (resp. MSOL, CMSOL) sentence involving a set of constant symbols $\bar{a}$ with $\ell_{\bar{a}} > 0$, nullary symbols $\bar{Z}$, unary symbols $\bar{U}$ and binary symbols $\bar{R}$, there exists a class $\mathcal{C}'$, definable by an FOL (resp. MSOL, CMSOL) sentence over the language $(\bar{a}', \bar{Z}', \bar{U}', \bar{R}')$, where $\bar{a}' = \bar{a} \setminus \{a_{\ell_{\bar{a}}}\}$, $\bar{Z}' = \bar{Z} \cup \bar{S} \cup \bar{D}$ where $\ell_S = \ell_U$ and $\ell_D = \ell_R$, $\bar{U}' = \bar{U} \cup \bar{I} \cup \bar{O}$ where $\ell_{\bar{I}} = \ell_{\bar{O}} = \ell_{\bar{R}}$, and $\bar{R}' = \bar{R}$, satisfying $f_{\mathcal{C}}(n) = f_{\mathcal{C}'}(n)$ for all $n \in \mathbb{N}$.*

The proofs are deferred to the full version of this paper [17]. We provide here a "sketch by example" on how this is done.

Adding nullary relations essentially allows us to get rid of the role of $\mathfrak{U}$ and $\mathfrak{R}$, so we will only inductively construct the expressions $\phi'_{\mathfrak{X}}$ and let nullary relations "hold the information" as to whether some relations hold with a substituted to all their variables.

If for instance we consider a binary relation $R(x,y)$ in the original vocabulary of ϕ, then the new vocabulary will include $R(x,y)$ (holding the information about the contents of R that does not involve the constant a), the unary relations $I(x)$ (holding the information about $R(x,a)$ for all $x \neq a$) and $O(y)$ (holding the information about $R(y,a)$), and the nullary relation $Z()$ that holds the information whether $R(a,a)$ holds.

In the inductive construction, $\phi'_{\mathfrak{X}}(\bar{x} \setminus x_{\mathfrak{X}})$ will tell us whether ϕ holds where all variables enumerated by $\mathfrak{X}$ are assigned the constant a, while all other variables are guaranteed to be different from a.

As an example, for the expression $\phi(x) = \exists_y R(x,y)$, we will have $\phi'_{\emptyset}(x) = (\exists_y R(x,y)) \vee I(x)$, which goes over the two options for the existence of y for which $R(x,y)$ holds: the first option being $y \neq a$, and the second one being $y = a$. Similarly we will have $\phi'_{\{1\}}() = (\exists y O(y)) \vee Z()$, which goes over the two options for the existence of y for which $R(a,y)$ holds.

For a universal quantifier the reduction would similarly go over the two cases, but use a conjunction this time. Thus for $\psi(x) = \forall_y R(x,y)$ we will have $\psi'_{\emptyset}(x) = (\forall_y R(x,y)) \wedge I(x)$, and $\psi'_{\{1\}}() = (\forall y O(y)) \wedge Z()$.

8.2 Handling higher arity relations and extended logics

The mechanism behind the proof of Theorem 17 can be extended higher arity relations, as well as more expressive logics, such as Second Order Logic (SOL) and Guarded Second Order Logic (GSOL)[3].

▶ **Theorem 19** (Many-one reduction allowing higher arity)**.** *For any class $\mathcal{C}$ defined by an FOL (resp. MSOL, CMSOL, GSOL, SOL) sentence involving a set of constant symbols $\bar{a}$, and relation symbols $\bar{R}$ of arbitrary arities, there exists a class $\mathcal{C}'$ definable by an FOL (resp. MSOL, CMSOL, GSOL, SOL) sentence involving $\bar{R}'$, which contains $\bar{R}$, has the same maximum arity as $\bar{R}$, and has no new relations of maximum arity, satisfying $f_{\mathcal{C}}(n) = f_{\mathcal{C}'}(n)$ for all $n \in \mathbb{N}$.*

Also here, this follows by induction using a corresponding extension of Lemma 18, which allows us to eliminate hard-wired constants one at a time. As a way of demonstrating the proof of this theorem (deferred to [17]), we explain how the single elimination lemma works when considering a ternary relation $R(x,y,z)$.

As before, for eliminating a hard-wired constant a we need to add lower arity relations to R. Namely, since after the reduction $R(x,y,z)$ itself would refer only to values different from a, we add a lower arity relation for every option of substituting the value a in any of these variables. This would add $\binom{3}{1} = 3$ binary relations which we will denote here by $R_1(y,z)$, $R_2(x,z)$ and $R_3(x,y)$, in addition to $\binom{3}{2} = 3$ unary relations which we will denote by $R_{12}(z)$, $R_{13}(y)$ and $R_{23}(x)$, and a single nullary relation which we will denote by $R_{123}()$.

Thus, given for example the expression $\phi(x,y) = \exists_z R(x,y,z)$, we would have for example $\phi'_{\emptyset}(x,y) = (\exists_z R(x,y,z)) \vee R_3(x,y)$, $\phi'_{\{1\}}(y) = (\exists_z R_1(y,z)) \vee R_{13}(y)$, $\phi'_{\{2\}}(x) = (\exists_z R_2(x,z)) \vee R_{23}(x)$ and $\phi'_{\{1,2\}}() = (\exists_z R_{12}(z)) \vee R_{123}()$.

Going another recursion level, for $\psi(x) = \exists_{y,z} R(x,y,z) = \exists_y \phi(x)$, we would correspondingly have $\psi'_{\emptyset}(x) = (\exists_{y,z} R(x,y,z)) \vee (\exists_z R_2(x,z)) \vee (\exists_y R_3(x,y)) \vee R_{23}(x)$, and $\psi'_{\{1\}}() = (\exists_{y,z} R_1(y,z)) \vee (\exists_z R_{12}(z)) \vee (\exists_y R_{13}(y)) \vee R_{123}()$.

[3] All discussion of these logics is deferred to [17], since the limit to the Specker-Blatter theorem discussed here only uses FOL.

9 An FOL-definable class $\mathcal{C}$ where $f_{\mathcal{C}}(n)$ is not MC-finite

In this section we negatively settle the question of whether the Specker-Blatter theorem holds for classes whose language contains only ternary and lower-arity relations.

9.1 Using one hard-wired constant

We first construct a class whose language includes a single ternary relation and a single hard-wired constant. Our counterexample builds on ideas used in [13].

▶ **Theorem 20** (Ternary relation counterexample with a constant). *There exists an FOL sentence $\phi_{\mathcal{M}}$ over the language (a, R), where a is a single (hard-wired) constant and R is a single relation of arity 3, so that the corresponding class $\mathcal{C}$ satisfies $f_{\mathcal{C}}(n-1) = 0$ for any n that is not a power of 2, and $f_{\mathcal{C}}(n-1) \equiv 1 \pmod 2$ for $n = 2^m$ for every $m \in \mathbb{N}$. In particular, $f_{\mathcal{C}}$ is not ultimately periodic modulo 2.*

The statement uses $f_{\mathcal{C}}(n-1)$ instead of $f_{\mathcal{C}}(n)$, but recalling the definition of $f_{\mathcal{C}}$, this refers to the universe $[n-1] \cup \{a\}$ whose size is n. We explain later how to modify this class to produce a counterexample modulo other prime numbers p instead of 2.

By Theorem 19, we have the following immediate corollary that does away with the constant, at the price of adding some additional smaller arity relations. This corollary is effectively a restatement of Theorem 8.

▶ **Corollary 21** (Ternary counterexample without constants). *There exists an FOL sentence $\phi'_{\mathcal{M}}$ over the language $(\bar{R})$, where $\bar{R}$ includes one relation of arity 3 and other relations of lower arities, so that the corresponding class $\mathcal{C}$ satisfies $f_{\mathcal{C}}(n) = 0$ for every n for which $n+1$ is not a power of 2, and $f_{\mathcal{C}}(n) \equiv 1 \pmod 2$ for $n = 2^m - 1$ for every $m \in \mathbb{N}$. In particular, $f_{\mathcal{C}}$ is not ultimately periodic modulo 2.*

At the end of this section we sketch how to further reduce the language so that it includes only one ternary relation and no lower arity relations. The full details are deferred to [17].

9.2 The first construction

The starting point of the construction is a structure that is defined over a non-constant length sequence (and hence not yet expressible in FOL) of unordered graphs. This definition follows the streamlining by Specker [25] of the original construction from [13].

▶ **Definition 22** (Iterated matching sequence). *Given a set V of vertices, An* iterated matching sequence *is a sequence of graphs over V, identified by their edge sets $\bar{E} = E_1, \ldots, E_{\ell_{\bar{E}}}$, satisfying the following for every $1 \leq i \leq \ell_{\bar{E}}$.*

- *The connected components of E_i are (vertex-disjoint) complete bipartite graphs.*
- *The two vertex classes of every complete bipartite graph in E_i as above are two connected components of $\bigcup_{j=1}^{i-1} E_j$ (for $i = 1$ this means that E_1 is a matching).*
- *Every connected component of $\bigcup_{j=1}^{i-1} E_j$ is a vertex class of some bipartite graph of E_i (so in particular E_1 is a perfect matching).*

An iterated matching sequence $\bar{E}$ is full *if every vertex pair $u, v \in V$ (where $u \neq v$) appears in some E_i.*

The following properties of iterated matching sequences are easily provable by induction.

▶ **Observation 23.** *For an iterated matching $\bar{E}$, every E_i corresponds to a perfect matching over the set of connected components of $\bigcup_{j=1}^{i-1} E_j$. Additionally, every connected component of $\bigcup_{j=1}^{i} E_j$ is a clique with exactly 2^i vertices.*

The above implies that there can be a full iterated matching sequence over $[n]$ if and only if n is a power of 2, in which case $\ell_{\bar{E}} = \log_2(n)$. Denoting the number of possible full iterated matching sequences over $[n]$ by $f_{\mathcal{M}}(n)$, note the following lemma.

▶ **Lemma 24** (see [25])**.** *For every n which is not a power of 2 we have $f_{\mathcal{M}}(n) = 0$, while $f_{\mathcal{M}}(n) \equiv 1 \pmod 2$ for $n = 2^m$ for every $m \in \mathbb{N}$.*

The rest of this section concerns the construction of a sentence $\phi_{\mathcal{M}}$ over a language with one constant and one ternary relation, so that the corresponding class $\mathcal{C}$ satisfies $f_{\mathcal{C}}(n-1) = f_{\mathcal{M}}(n)$. In the original construction utilizing a quaternary relation Q, essentially we had $(u, v, x, y) \in Q$ if $(u, v) \in E_i$ and $(x, y) \in E_{i-1}$ for some $1 < i \leq \ell_{\bar{E}}$, or $(u, v) \in E_1$ and $x = y$. For the construction here, we only have a ternary relation R, and we encode the placement of (u, v) within $\bar{E}$ by the set $\{w : (u, v, w) \in R\}$. We will have to utilize the hard-wired constant a to make sure that there is exactly one way to encode every full iterated matching sequence.

9.3 Setting up and referring to an order over the vertex pairs

We simulate the structure of a full iterated matching sequence over $[n]$ (where $n \in [n]$ is identified with the constant a) by assigning "ranks" to pairs of members of $[n]$, which we consider as vertices, where each pair (x, y) is assigned the set $r_{x,y} = \{z : (x, y, z) \in R\}$. First we need to make sure that "graphness" is satisfied, which means that $r_{x,y}$ is symmetric and is empty for loops.

$$\phi_{\text{graph}} = \forall_{x,y,z}(R(x, y, z) \to (x \neq y \wedge R(y, x, z)))$$

Next we make sure that every two vertex pairs have ranks that are comparable by containment. This means that for every (x_1, y_1) and (x_2, y_2) either $r_{x_1,y_1} \subseteq r_{x_2,y_2}$ or $r_{x_2,y_2} \subseteq r_{x_1,y_1}$.

$$\phi_{\text{comp}} = \forall_{x_1,y_1,x_2,y_2} \neg\exists_{z_1,z_2}(R(x_1, y_1, z_1) \wedge \neg R(x_2, y_2, z_1) \wedge R(x_2, y_2, z_2) \wedge \neg R(x_1, y_1, z_2))$$

Finally, we want every non-loop vertex pair to have a non-empty rank, and moreover for it to include the constant a. This is crucial, because a will eventually serve as an "anchor" making sure that there is only one way to assign ranks when encoding a full iterated matching sequence using the ternary relation R.

$$\phi_{\text{full}} = \forall_{x,y}((x \neq y) \to R(x, y, a))$$

It is a good time to sum up the full statement that sets up our pair ranks.

$$\phi_{\text{rank}} = \phi_{\text{graph}} \wedge \phi_{\text{comp}} \wedge \phi_{\text{full}}$$

Whenever this statement is satisfied, we can use it to construct expressions that compare ranks. We will use the following expressions, which compare the ranks of (x_1, y_1) and (x_2, y_2), when we formulate further conditions on R that will eventually force it to conform to a full iterated matching sequence. Note that conveniently, these comparison expression also work against loops (whose "rank", the empty set, is considered to be the lowest).

$$\begin{aligned}
\phi_{=}(x_1, y_1, x_2, y_2) &= \forall_z(R(x_1, y_1, z) \leftrightarrow R(x_2, y_2, z)) \\
\phi_{\leq}(x_1, y_1, x_2, y_2) &= \forall_z(R(x_1, y_1, z) \to R(x_2, y_2, z)) \\
\phi_{<}(x_1, y_1, x_2, y_2) &= \phi_{\leq}(x_1, y_1, x_2, y_2) \wedge \neg\phi_{=}(x_1, y_1, x_2, y_2)
\end{aligned}$$

9.4 Making the ordered pairs correspond to an iterated matching

In this subsection we consider a ternary relation R that is known to satisfy ϕ_{rank} as defined in Subsection 9.3, and impose further conditions that will force it to correspond to an iterated matching sequence (which will also be full by virtue of every pair having a rank).

For every rank appearing in R, that is for every set A which is equal to $r_{x,y}$ for some $x, y \in [n]$, we refer to the set of vertex pairs having this rank as E_i, where i is the number of ranks that appear in R (including the empty set, which is the "rank" of loops) and are strictly contained in A. So in particular $E_0 = \{(x, x) : x \in [n]\}$, and E_1 for example would be the set of vertex pairs that have the smallest non-empty set as their ranks.

We first impose the restriction that for any i, the graph defined by $\bigcup_{j=1}^{i} E_j$ is a transitive graph, that is a disjoint union of cliques. By Observation 23 this is a necessary condition for $\bar{E}$ to be an iterated matching sequence (note that allowing also the 0-ranked loops does not change the condition). This is the same as saying that for any three vertices x, y, z, it cannot be the case that the rank of (x, z) is larger than the maximum ranks of (x, y) and (y, z).

$$\phi_{\text{trans}} = \forall_{x,y,z}(\phi_{\leq}(x, z, x, y) \vee \phi_{\leq}(x, z, y, z))$$

Whenever R satisfies the above, it is not hard to add the restriction that E_i consists of disjoint complete bipartite graphs such that each of them connects exactly two components of $\bigcup_{j=1}^{i-1} E_j$, with all such components being covered. First we state that if some rank A exists, that is, there exists some (x, y) for which $A = r_{x,y}$, then every vertex z is a part of an edge with such rank.

$$\phi_{\text{cover}} = \forall_{x,y}\forall_{z}\exists_{w}\phi_{=}(x, y, z, w)$$

Then, using the prior knowledge that all connected components of both $\bigcup_{j=1}^{i-1} E_j$ and $\bigcup_{j=1}^{i} E_j$ are cliques, to make sure that every connected component of E_i is exactly a bipartite graph encompassing two components of $\bigcup_{j=1}^{i-1} E_j$, it is enough to state that it contains no triangles, excluding of course "triangles" of the type (x, x, x).

$$\phi_{\text{part}} = \forall_{x,y,z}((x \neq y) \rightarrow \neg(\phi_{=}(x, y, y, z) \wedge \phi_{=}(x, y, x, z)))$$

All of the above is sufficient to guarantee that the relation R corresponds to a full iterated matching sequence. However, as things stand now there can be many relations that correspond to the same iterated matching. This occurs because we still have unwanted freedom in choosing the sets that correspond to the possible ranks. To remove this freedom, we now require that the rank of every pair (x, y) for $x \neq y$ consists of exactly one connected component of the union of the lower ranked pairs. This will be sufficient, because by ϕ_{full} the only option for the rank would be the connected component that contains the constant a.

Noting that by ϕ_{trans} these components are cliques, it is enough to require that every member of $r_{x,y}$ is connected via a lower rank edge to a, while every vertex that is connected to a member of $r_{x,y}$ via a lower rank edge is also a member of $r_{x,y}$. We obtain the following statement.

$$\phi_{\text{anchor}} = \forall_{x,y,z}(R(x, y, z) \rightarrow (\phi_{<}(z, a, x, y) \wedge \forall_{w}(\phi_{<}(z, w, x, y) \rightarrow R(x, y, w))))$$

The final statement that counts the number of full iterated matching sequences, and hence provides the example proving Theorem 20 is the following.

$$\phi_{\mathcal{M}} = \phi_{\text{rank}} \wedge \phi_{\text{trans}} \wedge \phi_{\text{cover}} \wedge \phi_{\text{part}} \wedge \phi_{\text{anchor}}$$

9.5 Adapting the example to other primes

We show here how to adapt the FOL sentence from Theorem 20 to provide a sequence that is not ultimately periodic modulo p for any prime number $p \geq 2$. The analogous corollary about removing the constant also follows.

▶ **Theorem 25** (Ternary relation counterexample for $p \geq 2$)**.** *For any prime number p, there exists an FOL sentence $\phi_{\mathcal{M}_p}$ over the language (a, R), where a is a (hard-wired) constant and R is a relation of arity 3, so that the corresponding class $\mathcal{C}_p$ satisfies $f_{\mathcal{C}_p}(n-1) = 0$ for every n that is not a power of p, and $f_{\mathcal{C}_p}(n-1) \equiv 1 \pmod{p}$ for $n = p^m$ for every $m \in \mathbb{N}$. In particular, $f_{\mathcal{C}_p}$ is not ultimately periodic modulo p.*

The construction follows the same lines as the extension from $p = 2$ to $p \geq 2$ in previous works. For completeness we give some details on how it works with respect to the version of [25]. The basic idea is to use a "matching" of p-tuples instead of pairs.

▶ **Definition 26.** *A* p-matching *over the vertex set $[n]$ is a spanning graph, each of whose connected components is either a clique with p vertices or a single vertex. A* perfect p-matching *is a p-matching in which there are no single vertex components (in other words, it is a partition of $[n]$ into sets of size p).*

The following is not hard to prove.

▶ **Lemma 27.** *There are no perfect p-matchings over $[n]$ unless n is a multiple of p, in which case their number is congruent to 1 modulo p.*

Proof. The case where n is not a multiple of p is trivial. Otherwise, consider the number of possible partitions of the set $[p]$ to a sequence of subsets of sizes $i_1, \ldots, i_r$, where $\sum_{k=1}^{r} i_k = p$. Note that unless $i_1 = p$ (and hence $r = 1$), the number of such partitions is divisible by $\binom{p}{i_1}$, which is divisible by p (since p is a prime).

Denoting by $f_{M_p}(n)$ the number of perfect p-matchings over $[n]$, We consider for any p-matching its restriction to $[p]$ (which corresponds to a partition of $[p]$ – the reason we need to consider the partitions as sequences rather than as unordered families of sets is that we need to consider which sets in the restriction of the p-matching over $[n] \setminus [p]$ they are "attached" to). This implies that $f_{M_p}(n) \equiv f_{M_p}(n-p) \pmod{p}$ for every $n > p$, allowing us to prove by induction that $f_{M_p}(n) \equiv 1 \pmod{p}$ if p divides n. ◀

The definition of an iterated p-matching sequence is what one would expect.

▶ **Definition 28** (Iterated p-matching sequence)**.** *Given a set V of vertices, An* iterated p-matching sequence *is a sequence graphs over V, identified by their edge sets $\bar{E} = E_1, \ldots, E_{\ell_{\bar{E}}}$, satisfying the following for every $1 \leq i \leq \ell_{\bar{E}}$.*

- *The connected components of E_i are (vertex-disjoint) complete p-partite graphs.*
- *The p vertex classes of every complete p-partite graph in E_i as above are p connected components of $\bigcup_{j=1}^{i-1} E_j$ (for $i = 1$ this means that E_1 is a p-matching).*
- *Every connected component of $\bigcup_{j=1}^{i-1} E_j$ is a vertex class of some p-partite graph of E_i (so in particular E_1 is a perfect p-matching).*

An iterated matching sequence $\bar{E}$ is full *if every vertex pair $u, v \in V$ (where $u \neq v$) appears in some E_i.*

Again we have the following properties, analogous to those of iterated matching sequences.

▶ **Observation 29.** *For an iterated p-matching $\bar{E}$, every E_i corresponds to a perfect p-matching over the set of connected components of $\bigcup_{j=1}^{i-1} E_j$. Additionally, every connected component of $\bigcup_{j=1}^{i} E_j$ is a clique with exactly p^i vertices.*

The above implies that there can be a full iterated matching sequence over $[n]$ if and only if n is a power of p, in which case $\ell_{\bar{E}} = \log_p(n)$. Denoting the number of possible full iterated matching sequences over $[n]$ by $f_{\mathcal{M}_p}(n)$, note the following lemma.

▶ **Lemma 30.** *For every n that is not a power of p we have $f_{\mathcal{M}}(n) = 0$, while for $n = p^m$ for every $m \in \mathbb{N}$ we have $f_{\mathcal{M}_p}(n) \equiv 1 \pmod{p}$.*

Proof. The case where n is not a power of p was already discussed above. The case $n = p^m$ is proved by induction over m using Lemma 27. ◀

From here on the construction of $\phi_{\mathcal{M}_p}$ is identical to that of $\phi_{\mathcal{M}}$ in Subsection 9.3 and Subsection 9.4, with the only exceptions being the replacements for ϕ_{cover} and ϕ_{part}.

To construct ϕ_{cover_p}, we need to state that for every existing rank, each vertex is a part of a size p clique consisting of edges from this rank.

$$\phi_{\text{cover}_p} = \forall_{x,y} \forall_{z_1} \exists_{z_2,\ldots,z_p} \bigwedge_{1 \le i < j \le p} \phi_{=}(x, y, z_i, z_j)$$

To construct ϕ_{part_p}, we need to state that no E_i may contain a clique with $p+1$ vertices.

$$\phi_{\text{part}_p} = \forall_{z_1,\ldots,z_{p+1}} ((z_1 \neq z_2) \to \neg(\bigwedge_{1 \le i < j \le p+1} \phi_{=}(z_1, z_2, z_i, z_j))$$

The final expression is the following.

$$\phi_{\mathcal{M}_p} = \phi_{\text{rank}} \wedge \phi_{\text{trans}} \wedge \phi_{\text{cover}_p} \wedge \phi_{\text{part}_p} \wedge \phi_{\text{anchor}}$$

9.6 Reducing the example further to have a single relation

We provide here a sketch on how to produce, starting with Theorem 8, a sentence with a single relation that provides a class that is not MC-finite.

▶ **Theorem 31** (A sentence with a single relation)**.** *For every prime number $p \geq 1$ there exists an* FOL*-sentence ϕ_p over a language consisting of a single relation of arity 3, so that for the class $\mathcal{C}$ corresponding to ϕ_p, its counting function $f_{\mathcal{C}}(n)$ is not ultimately periodic modulo p.*

Starting with an expression ϕ that results from invoking Theorem 25 over $\phi_{\mathcal{M}_p}$, we explain how to reduce it further to an expression that involves a single ternary relation. This transformation is ad-hoc and uses certain specific features and symmetries of models satisfying $\phi_{\mathcal{M}_p}$, and their reflection in the corresponding models satisfying ϕ. The full details require delving into the specifics of the proof of Theorem 19, and are deferred to [17].

As $\phi_{\mathcal{M}_p}$ involves a single ternary relation R and a single constant a, the resulting ϕ involves a corresponding ternary relation, as well as three binary relations, three unary relations, and a single nullary relation. Since the lower order relations result from substituting the constant at some of the places of the relation R (while restricting the other places to hold values different from a), looking at the working of $\phi_{\mathcal{M}_p}$ allows us to immediately rule out most options for the lower arity relations.

For example, the nullary relation would correspond to whether $R(a, a, a)$ holds, so it must evaluate to $\bot$ ("false") for all models of ϕ (since $\phi_{\mathcal{M}_p}$ implies $\neg R(x, x, y)$ for all x and y, equal or unequal to a). Thus we may just remove it and replace its occurrences in ϕ with the $\bot$ symbol.

Similar considerations allow us to eliminate the unary relation corresponding to $R(a,a,x)$ for $x \neq a$ (always false), and the unary relations corresponding to $R(a,x,a)$ and $R(x,a,a)$ (always true by ϕ_{full} since $x \neq a$).

Next, the binary relation corresponding to $R(x,y,a)$ for $x,y \neq a$, while not constant, can be "fully deduced" and replaced with $x \neq y$ by ϕ_{full} and ϕ_{graph}.

This leaves us with the two relations corresponding to $R(x,a,y)$ and $R(a,x,y)$. By first noting that for satisfying models they are equal to each other (by ϕ_{graph}), we can reduce them to a single relation. In the final step, we "fold" this relation into the ternary relation, after noting that R has to satisfy $\neg R(x,x,y)$ for all x,y. The last operation requires first replacing the occurrences of $R(x,y,z)$ in the sentence with "$(x \neq y) \wedge R(x,y,z)$", which now "frees" this part of R to be used instead of the binary relation. This does not change the number of satisfying models, since we make sure that this "region" of R is completely used through a bijection for the role of the binary relation that it replaces.

10 Conclusions and open problems

In this work we have extended the Specker-Blatter Theorem to classes of τ-structures definable in CMSOL for vocabularies τ which contain a finite number of hard-wired constants, unary and binary relation symbols, Corollary 7. We have also shown that it does not hold already when τ consists of only one ternary relation symbol, Theorem 31. We note that in [15, 16] we have shown that for $\mathcal{C}$ definable in CMSOL such that all structures have degree bounded by a constant d, $S_{\mathcal{C}}(n)$ is always MC-finite. The degree of a structure $\mathcal{A}$ is defined via the Gaifman graph of $\mathcal{A}$. With this the MC-finiteness of $S_{\mathcal{C}}(n)$ for CMSOL-definable classes of τ-structures as a function of τ is completely understood. Applications of our results in this paper to restricted Bell numbers and various restricted partition functions are given in [11].

A sequence of integers $s(n)$ is MC-finite if for every $m \in \mathbb{N}^+$ there are constants $r(m), p(m) \in \mathbb{N}^+$ and coefficients $\alpha_1(m), \ldots, \alpha_{p(m)} \in \mathbb{N}^+$, such that for all $n \geq r(m)$ we have

$$s(n+p(m)+1) \equiv \sum_{i=0}^{p(m)} \alpha_i s(i) \mod m.$$

The Specker-Blatter Theorem gives little information on the constants $r(m), p(m)$ or the coefficients $\alpha_1(m), \ldots, \alpha_{p(m)}$. These in particular depend on the substitution rank of the class $\mathcal{C}$. In fact Theorem 3 gives a very bad estimate of the substitution rank in the case of binary relation symbols. The constants are computable, but it is not known whether they are always computable in feasible time or whether their size is bounded by an elementary function. In the presence of constants the substitution rank is not defined. Our main Theorem 12 allows to eliminate the constants, and therefore gives a formula for which the substitution rank is defined. However, due to the increased complexity of the resulting formula, the estimate of the substitution rank will be even worse.

▶ **Problem 32.** *Given a sentence ϕ in* CMSOL(τ) *where τ consists only of constants, unary and binary relation symbols,*

- **(i)** *what is the time complexity of computing the constants $r(m), p(m)$ and the coefficients $\alpha_1(m), \ldots, \alpha_{p(m)}$?*
- **(ii)** *what can we say about the size of these constants?*

The proof of Theorem 3 depends on the Feferman-Vaught Theorem which also holds for CMSOL(τ) for any finite relational τ, [10, 22]. In our context, the Feferman-Vaught Theorem allows to check whether a formula of CMSOL(τ) holds in $Subst(\mathcal{A}_1, a, \mathcal{A}_2)$ by checking a

sequence of CMSOL(τ)-formulas in $\mathcal{A}_1$ and $\mathcal{A}_2$ independently. This sequence is called a reduction sequence, cf. [14]. In [6] it is shown that even for FOL(τ) the size of the reduction sequences for the Feferman-Vaught Theorem cannot, in general, be bounded by an elementary function.

▶ **Problem 33.** *Does there exist an elementary function $f(k)$, so that for any sentence ϕ in* CMSOL(τ) *where τ consists only of constants, unary and binary relation symbols, the size of the constants $r(m)$ and $p(m)$ is bounded by $f(\max\{|\phi|, m\})$?*

The Specker-Blatter Theorem also applies to hereditary, monotone and minor-closed graph classes, provided they are definable using a finite set of forbidden (induced) subgraphs or minors. In the first two cases such a class is FOL-definable. In the case of a minor-closed class, B. Courcelle showed that it is MSOL-definable, see [5]. By the celebrated theorem of N. Robertson and P. Seymour, [7], every minor-closed class of graphs is definable by a finite set of forbidden minors. However, there are monotone (hereditary) classes of graphs where a finite set of forbidden (induced) subgraphs does not suffice.

▶ **Problem 34.** *Are there hereditary or monotone classes of graphs $\mathcal{C}$ such that $Sp_{\mathcal{C}}(n)$ is not MC-finite?*

An analogue question arises when we replace graphs by finite relational τ-structures. In this case one speaks of classes of τ-structures closed under substructures. Every class of finite τ-structures $\mathcal{C}$ closed under substructures can be characterized by a set of forbidden substructures. If this set is finite, $\mathcal{C}$ is again FOL-definable, and the Specker-Blatter Theorem applies.

▶ **Problem 35.**

(i) *Let τ be a relational vocabulary. Are there substructure closed classes $\mathcal{C}$ of τ-structures such that $Sp_{\mathcal{C}}(n)$ is not MC-finite?*

(ii) *Same question when all the relations are at most binary?*

References

1 C. Blatter and E. Specker. Le nombre de structures finies d'une théorie à charactère fini. *Sciences Mathématiques, Fonds Nationale de la recherche Scientifique, Bruxelles*, pages 41–44, 1981.

2 C. Blatter and E. Specker. Modular periodicity of combinatorial sequences. *Abstracts of the AMS*, 4:313, 1983.

3 C. Blatter and E. Specker. Recurrence relations for the number of labeled structures on a finite set. In E. Börger, G. Hasenjaeger, and D. Rödding, editors, *In Logic and Machines: Decision Problems and Complexity*, volume 171 of *Lecture Notes in Computer Science*, pages 43–61. Springer, 1984.

4 Andrei Z Broder. The r-stirling numbers. *Discrete Mathematics*, 49(3):241–259, 1984.

5 B. Courcelle and J. Engelfriet. *Graph Structure and Monadic Second-order Logic, a Language Theoretic Approach*. Cambridge University Press, 2012.

6 Anuj Dawar, Martin Grohe, Stephan Kreutzer, and Nicole Schweikardt. Model theory makes formulas large. In *International Colloquium on Automata, Languages, and Programming*, pages 913–924. Springer, 2007.

7 R. Diestel. *Graph Theory*. Graduate Texts in Mathematics. Springer, 3 edition, 2005.

8 H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer Verlag, 1995.

9 Graham Everest, Alfred J van der Poorten, Igor Shparlinski, Thomas Ward, et al. *Recurrence sequences*, volume 104. American Mathematical Society Providence, RI, 2003.

10 S. Feferman and R. Vaught. The first order properties of algebraic systems. *Fundamenta Mathematicae*, 47:57–103, 1959.

11 Yuval Filmus, Eldar Fischer, Johann A. Makowsky, and Vsevolod Rakita. MC-finiteness of restricted set partitions, 2023. `arXiv:2302.08265`.

12 E. Fischer. The Specker-Blatter theorem does not hold for quaternary relations. *Journal of Combinatorial Theory, Series A*, 103:121–136, 2003.

13 E. Fischer. The Specker-Blatter theorem does not hold for quaternary relations. *Journal of Combinatorial Theory, Series A*, 103:121–136, 2003.

14 E. Fischer, T. Kotek, and J.A. Makowsky. Application of logic to combinatorial sequences and their recurrence relations. In M. Grohe and J.A. Makowsky, editors, *Model Theoretic Methods in Finite Combinatorics*, volume 558 of *Contemporary Mathematics*, pages 1–42. American Mathematical Society, 2011.

15 E. Fischer and J. A. Makowsky. The Specker-Blatter theorem revisited. In *COCOON*, volume 2697 of *Lecture Notes in Computer Science*, pages 90–101. Springer, 2003.

16 Eldar Fischer, Tomer Kotek, and Johann A Makowsky. Application of logic to combinatorial sequences and their recurrence relations. *Model Theoretic Methods in Finite Combinatorics*, 558:1–42, 2011.

17 Eldar Fischer and Johann A. Makowsky. Extensions and limits of the specker-blatter theorem, 2022. `arXiv:2206.12135`.

18 Ronald L Graham, Donald E Knuth, and Oren Patashnik. *Concrete mathematics: a foundation for computer science*. Addison-Wesley, 1989.

19 Manuel Kauers and Paule. *The Concrete Tetrahedron: Symbolic Sums, Recurrence Equations, Generating Functions, Asymptotic Estimates*. Springer, 2011.

20 Thomas Koshy. *Catalan numbers with applications*. Oxford University Press, 2008.

21 L. Libkin. *Elements of Finite Model Theory*. Springer, 2004.

22 J.A. Makowsky. Algorithmic uses of the Feferman-Vaught theorem. *Annals of Pure and Applied Logic*, 126.1-3:159–213, 2004.

23 Götz Pfeiffer. Counting transitive relations. *Journal of Integer Sequences*, 7(2):3, 2004.

24 James A Reeds and Neil JA Sloane. Shift register synthesis (modulo m). *SIAM Journal on Computing*, 14(3):505–513, 1985.

25 E. Specker. Modular counting and substitution of structures. *Combinatorics, Probability and Computing*, 14:203–210, 2005.

26 Ernst Specker. Application of logic and combinatorics to enumeration problems. In *Ernst Specker Selecta*, pages 324–350. Springer, 1990.

Going Deep and Going Wide: Counting Logic and Homomorphism Indistinguishability over Graphs of Bounded Treedepth and Treewidth

Eva Fluck
RWTH Aachen University, Germany

Tim Seppelt
RWTH Aachen University, Germany

Gian Luca Spitzer
RWTH Aachen University, Germany

Abstract

We study the expressive power of first-order logic with counting quantifiers, especially the k-variable and quantifier-rank-q fragment C_q^k, using homomorphism indistinguishability. Recently, Dawar, Jakl, and Reggio (2021) proved that two graphs satisfy the same C_q^k-sentences if and only if they are homomorphism indistinguishable over the class $\mathcal{T}_q^k$ of graphs admitting a k-pebble forest cover of depth q. Their proof builds on the categorical framework of game comonads developed by Abramsky, Dawar, and Wang (2017). We reprove their result using elementary techniques inspired by Dvořák (2010). Using these techniques we also give a characterisation of guarded counting logic. Our main focus, however, is to provide a graph theoretic analysis of the graph class $\mathcal{T}_q^k$. This allows us to separate $\mathcal{T}_q^k$ from the intersection of the graph class $\mathcal{TW}_{k-1}$, that is graphs of treewidth less or equal $k-1$, and $\mathcal{TD}_q$, that is graphs of treedepth at most q if q is sufficiently larger than k. We are able to lift this separation to the semantic separation of the respective homomorphism indistinguishability relations. A part of this separation is to prove that the class $\mathcal{TD}_q$ is homomorphism distinguishing closed, which was already conjectured by Roberson (2022).

2012 ACM Subject Classification Mathematics of computing → Graph theory; Theory of computation → Finite Model Theory

Keywords and phrases Treewidth, treedepth, homomorphism indistinguishability, counting first-order logic

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.27

Related Version *Full Version*: `https://arxiv.org/abs/2308.06044` [9]

Funding *Tim Seppelt*: German Research Foundation (DFG) via RTG 2236/2 (UnRAVeL), European Union (ERC, SymSim, 101054974)
Gian Luca Spitzer: German Research Foundation (DFG) via RTG 2236/2 (UnRAVeL)

Acknowledgements We would like to thank Martin Grohe and Daniel Neuen for fruitful discussions.

1 Introduction

Since the 1980s, first-order logic with counting quantifiers C plays a decisive role in finite model theory. In this extension of first-order logic with quantifiers $\exists^{\geq t} x$ ("there exists at least t many x"), properties which can be expressed in first-order logic only with formulae of length depending on t can be expressed succinctly. Of particular interest are the k-variable and quantifier-depth-q fragments C^k and C_q of C, which enjoy rich connections to graph algorithms [8], algebraic graph theory [7, 16], optimisation [16, 27], graph neural networks [21, 30, 15], and category theory [5, 1].

The intersection of these fragments, the fragment $\mathsf{C}_q^k := \mathsf{C}^k \cap \mathsf{C}_q$ of all C-formulae with k-variables and quantifier-depth q, has received much less attention [25]. In this work, we study the expressivity of C_q^k using homomorphism indistinguishability.

Homomorphism indistinguishability is an emerging framework for measuring the expressivity of equivalence relations comparing graphs. Two graphs G and H are *homomorphism indistinguishable* over a graph class $\mathcal{F}$ if for all $F \in \mathcal{F}$ the number of homomorphisms from F to G is equal to the number of homomorphisms from F to H. Many natural equivalence relations between graphs including isomorphism [19], quantum isomorphism [20], cospectrality [7], and feasibility of integer programming relaxations for graph isomorphism [16, 27] can be characterised as homomorphism indistinguishability relations over certain graph classes. Establishing such characterisations is intriguing since it allows to use tools from structural graph theory to study equivalence relations between graphs [26, 28]. Furthermore, the expressivity of homomorphism counts themselves is of practical interest [23, 14].

Equivalence with respect to C^k and C_q has been characterised by Dvořák [8] and Grohe [13] as homomorphism indistinguishability over the classes $\mathcal{TW}_{k-1}$ of graphs of treewidth $\leq k-1$ and $\mathcal{TD}_q$ of graphs of treedepth $\leq q$, respectively. Recently, Dawar, Jakl, and Reggio [5] proved that two graphs satisfy the same C^k_q-sentences if and only if they are homomorphism indistinguishable over the class $\mathcal{T}^k_q$ of graphs admitting a k-pebble forest cover of depth q. Their proof builds on the categorical framework of game comonads developed in [1].

As a first step, we reprove their result using elementary techniques inspired by Dvořák [8]. The general idea is to translate between sentences in C and graphs from which homomorphism are counted in an inductive fashion. By carefully imposing structural constraints, we are able to extend the original correspondence from [8] between C^k and graphs of treewidth at most $k-1$ to C_q and graphs of treedepth at most q, reproducing a result of [13], and finally to C^k_q and $\mathcal{T}^k_q$. This simple and uniform proof strategy also yields the following result on guarded counting logic GC^k_q. Guarded counting logic plays a crucial role in the theory of properties of higher arity expressible by graph neural networks [15]. Towards this goal we introduce a new graph class called $\mathcal{GT}^k_q$, which is closely related to $\mathcal{T}^k_q$.

▶ **Theorem 1.** *Let $k, q \geq 1$. Two graphs G and H are GC^k_q-equivalent if and only if they are homomorphism indistinguishable over $\mathcal{GT}^k_q$.*

The main contribution of this work, however, concerns the relationship between the graph classes $\mathcal{T}^k_q$ and the class $\mathcal{TW}_{k-1} \cap \mathcal{TD}_q$ of graphs which have treewidth at most $k-1$ *and* treedepth at most q. Given the results of [8, 13], one might think that elementary equivalence with respect to sentences in $\mathsf{C}^k_q = \mathsf{C}^k \cap \mathsf{C}_q$ is characterised by homomorphism indistinguishability with respect to $\mathcal{TW}_{k-1} \cap \mathcal{TD}_q$. The central result of this paper asserts that this intuition is wrong. As a fist step towards this, we prove that the graph class $\mathcal{T}^k_q$ and $\mathcal{TW}_{k-1} \cap \mathcal{TD}_q$ are distinct if q is sufficiently larger than k. All logarithms in this work are to the base 2.

▶ **Theorem 2.** *For $q \geq 3$, $\mathcal{T}^2_q \subsetneq \mathcal{TW}_1 \cap \mathcal{TD}_q$, and for $2 \leq k-1 \leq \frac{q}{3+\log q}$, $\mathcal{T}^k_q \subsetneq \mathcal{TW}_{k-1} \cap \mathcal{TD}_q$.*

Towards Theorem 2, we give an equivalent characterisation of $\mathcal{T}^k_q$ via a monotone cops-and-robber game, which is essentially the standard game for treewidth where one additionally counts the number of rounds the cops need to capture the robber. Here, "monotone" refers to a restriction of Cops, who is only allowed to move cops that are not adjacent to the current escape-space of the Robber. Building on [10], we then prove that $\mathcal{T}^k_q$ is a proper subclass of $\mathcal{TW}_{k-1} \cap \mathcal{TD}_q$, for q sufficiently larger than k. Additionally, we provide an analysis of various notions designed to restrict both width and depth of a decomposition and show that all of them are equivalent. Adding to the original definition of $\mathcal{T}^k_q$ via k-pebble forest covers of depth q, which can be interpreted as treedepth decompositions augmented by a width

measure, we introduce a way to measure the depth of tree decompositions. Finally, we define k-construction trees of elimination depth q, another equivalent notion, which relates to the machinery used by Dvořák [8].

However, the, let us say syntactical, separation of the graph classes $\mathcal{T}_q^k$ and $\mathcal{TW}_{k-1} \cap \mathcal{TD}_q$ from Theorem 2 does not suffice to separate their homomorphism indistinguishability relations semantically. In fact, it could well be that all graphs which are homomorphism indistinguishable over $\mathcal{T}_q^k$ are also homomorphism indistinguishable over $\mathcal{TW}_{k-1} \cap \mathcal{TD}_q$.

That such phenomena do not arise under certain mild assumptions was recently conjectured by Roberson [26]. His conjecture asserts that every graph class which is closed under taking minors and disjoint unions is homomorphism distinguishing closed. Here, a graph class $\mathcal{F}$ is *homomorphism distinguishing closed* if it satisfies the following maximality condition: For every graph $F \notin \mathcal{F}$, there exists two graphs G and H which are homomorphism indistinguishable over $\mathcal{F}$ but have different numbers of homomorphism from F.

Since $\mathcal{T}_q^k$, $\mathcal{TW}_{k-1}$, and $\mathcal{TD}_q$ are closed under disjoint unions and minors, the confirmation of Roberson's conjecture would readily imply the semantic counterpart of Theorem 2. Unfortunately, Roberson's conjecture is wide open and has been confirmed only for the class of all planar graphs [26], $\mathcal{TW}_{k-1}$ [22], and for graph classes which are essentially finite [28]. Guided by [22], we add to this short list of examples:

▶ **Theorem 3.** *For $q \geq 1$, the class $\mathcal{TD}_q$ is homomorphism distinguishing closed.*

Combining this with the results of [22], we get that $\mathcal{TW}_{k-1} \cap \mathcal{TD}_q$ is homomorphism distinguishing closed as well. We then set out to separate homomorphism indistinguishability over $\mathcal{T}_q^k$ and $\mathcal{TW}_{k-1} \cap \mathcal{TD}_q$. Despite not being able to prove that $\mathcal{T}_q^k$ is homomorphism distinguishing closed, we prove that the homomorphism distinguishing closure of $\mathcal{T}_q^k$, i.e. the smallest homomorphism distinguishing closed superclass of $\mathcal{T}_q^k$, is a proper subclass of $\mathcal{TW}_{k-1} \cap \mathcal{TD}_q$, for q sufficiently larger than k. Written out, Theorem 4 asserts that whenever q is sufficiently large in terms of k, then there exist graphs which are homomorphism indistinguishable over $\mathcal{T}_q^k$ but not over $\mathcal{TW}_{k-1} \cap \mathcal{TD}_q$.

▶ **Theorem 4.** *For $q \geq 3$, $\mathrm{cl}(\mathcal{T}_q^2) \subsetneq \mathcal{TW}_1 \cap \mathcal{TD}_q$, and for $2 \leq k-1 \leq \frac{q}{3+\log q}$, $\mathrm{cl}(\mathcal{T}_q^k) \subsetneq \mathcal{TW}_{k-1} \cap \mathcal{TD}_q$.*

Besides obtaining Theorem 4, we distil the challenge of proving that $\mathcal{T}_q^k$ is homomorphism distinguishing closed to the question whether the monotone variant of the cops-and-robber game is equivalent to the non-monotone variant. In general this equivalence between the monotone and non-monotone variant of a graph searching game is a non-trivial property. There are games where the two variants are equivalent, such as the games corresponding to treewidth [29] and treedepth [11], as well as games where they are not, such as games corresponding to directed treewidth [18] or hypertreewidth [12].

2 Preliminaries

Notation. By $[k]$ we denote the set $\{1, \dots, k\}$. For a finite set X, we write 2^X for the power set of X. For a function f, we denote the domain of f by $\mathrm{dom}(f)$. The *image* of f is the set $\mathrm{img}(f) \coloneqq \{f(x) \mid x \in \mathrm{dom}(f)\}$. The *restriction* of a function $f\colon A \to C$ to some set $B \subseteq A$ is the function $f|_B\colon B \to C$ with $f|_B(x) = f(x)$ for $x \in B$. For functions $f\colon A \to C$, $g\colon B \to C$ that agree on $A \cap B$, we write $f \sqcup g$ for the union of f and g, that is, the function mapping x to $f(x)$ if $x \in A$ and to $g(x)$ if $x \in B$.

We use bold letters to denote tuples. The tuple elements are denoted by the corresponding regular letter together with an index. For example, $\boldsymbol{a}$ stands for the tuple $(a_1, \dots, a_n)$.

Graphs and Labels. A graph G is a tuple $(V(G), E(G))$, where $V(G)$ is a finite set of vertices and $E(G) \subseteq \binom{V(G)}{2}$ is the set of edges. We usually write uv or vu to denote the edge $\{u, v\} \in E(G)$. Unless otherwise specified, all graphs are assumed to be simple: They are undirected, unweighted and contain neither loops nor parallel edges. We denote the class of all graphs by $\mathcal{G}$.

A *k-labelled graph* G is a graph together with a partial function $\nu_G \colon [k] \rightharpoonup V(G)$ that assigns labels from the finite set $[k] = \{1, \dots, k\}$ to vertices of G. A label thus occurs at most once in a graph, a single vertex can have multiple labels, and not all labels have to be assigned. By $L_G = \operatorname{img}(\nu_G)$ we denote the set of labelled vertices of G. A graph where every vertex has at least one label is called *fully labelled*. We denote the class of all k-labelled graphs by $\mathcal{G}_k$.

For $\ell \in [k]$ and $v \in V(G)$, we write $G(\ell \to v)$ to denote the graph obtained from G by setting $\nu_{G(\ell \to v)}(\ell) = v$. We can *remove* a label ℓ from a graph G, which yields a copy G' of G where $\nu_{G'}(\ell) = \bot$ and $\nu_{G'}(\ell') = \nu_G(\ell')$ for all $\ell' \neq \ell$. The *product*[1] $G_1 G_2$ of two labelled graphs is the graph obtained by taking the disjoint union of G_1 and G_2, identifying vertices with the same label, and suppressing any parallel edges that might be created.

We call H a *subgraph* of G if H can be obtained from G by removing vertices and edges. H is a *minor* of G if it can be obtained from G by removing vertices, removing edges, and *contracting* edges. We contract an edge uv by removing it and identifying u and v. For labelled graphs, the new vertex is labelled by the union of labels of u and v.

A graph is *connected* if there exists a path between any two vertices. A *tree* is a graph where any two vertices are connected by exactly one path. The disjoint union of one or more trees is called a *forest*. A *rooted tree* (T, r) is a tree T together with some designated vertex $r \in V(T)$, the *root* of T. A *rooted forest* $(F, \boldsymbol{r})$ is a disjoint union of rooted trees. The *height* of a rooted tree is equal to the number of vertices on the longest path from the root to the leaves. The height of a rooted forest is the maximum height over all its connected components.

At times, the following alternative definition is more convenient. We can view a rooted forest $(F, \boldsymbol{r})$ as a pair $(V(F), \preceq)$, where $\preceq$ is a partial order on $V(F)$ and for every $v \in V(F)$ the elements of the set $\{u \in V(F) \mid u \preceq v\}$ are pairwise comparable: The minimal elements of $\preceq$ are precisely the roots of F, and for any rooted tree (T, r) that is part of F we let $v \preceq w$ if v is on the unique path from r to w.

The height of a rooted forest $(F, \boldsymbol{r})$ is then given by the length of the longest $\preceq$-chain. A rooted tree (T', r') is a *subtree* of a tree (T, r) if $V(T') \subseteq V(T)$ and $\preceq^{T'}$ is the restriction of $\preceq^{T}$ to $V(T')$. Note that the subgraph of T induced by $V(T')$ might not be a tree, since the vertices of T' can be interleaved with vertices that do not belong to T'. We call a subtree T' of T *connected* if its induced subgraph on T is connected.

Homomorphisms. A *homomorphism* from a graph F to a graph G is a map $h \colon V(F) \to V(G)$ satisfying $uv \in E(F) \implies h(u)h(v) \in E(G)$. For k-labelled graphs, we additionally require that $h(\nu_F(\ell)) = \nu_G(\ell)$ for all $\ell \in \operatorname{dom}(\nu_F)$. We denote the set of homomorphisms from F to G by $\operatorname{Hom}(F, G)$. The number of homomorphisms from F to G we denote by $\hom(F, G) \coloneqq |\operatorname{Hom}(F, G)|$. We write $\operatorname{Hom}(F, G; a_1 \mapsto b_1, \dots, a_n \mapsto b_n)$ to denote the set of homomorphism $h \colon F \to G$ satisfying $h(a_i) = b_i$ for $i \in [n]$. Two graphs G and H are *homomorphism indistinguishable* over a graph class $\mathcal{F}$ if $\hom(F, G) = \hom(F, H)$ for all $F \in \mathcal{F}$.

[1] Categorically speaking, this is a coproduct of labelled graphs or a pushout of graphs.

Logic of Graphs. We will mainly consider *counting first-order logic* C. C extends regular first-order logic FO by quantifiers $\exists^{\geq t}$, for $t \in \mathbb{N}$. Consequently, we can build a C-formula in the usual way from atomic formulae; variables $x_1, x_2, \ldots$; logical operators $\wedge, \vee, \rightarrow, \neg$; and quantifiers $\forall, \exists, \exists^{\geq t}$. The atomic formulae in the language of graphs are $E\alpha\beta$ and $\alpha = \beta$ for arbitrary variables α, β.

An occurrence of a variable x is called *free* if it is not in the scope of any quantifier. The *free variables* $\mathrm{free}(\varphi)$ of a formula φ are precisely those that have a free occurrence in φ. A formula without free variables is called a *sentence*. We often write $\varphi(x_1, \ldots, x_n)$ to denote that the free variables of φ are among $x_1, \ldots, x_n$. For a graph G, it usually depends on the interpretation of the free variables whether $G \models \varphi(x_1, \ldots, x_n)$. We write $G, v_1, \ldots, v_n \models \varphi(x_1, \ldots, x_n)$ or $G \models \varphi(v_1, \ldots, v_n)$ if G satisfies φ when x_i is interpreted by v_i. We might also give an explicit *interpretation function* $\mathfrak{I}\colon \mathrm{free}(\varphi) \to V(G)$, writing $G, \mathfrak{I} \models \varphi$.

We generalise the notion of C-equivalence, writing $G, v_1, \ldots, v_n \equiv_{\mathsf{C}} H, w_1, \ldots, w_n$ to denote that for all formulae $\varphi(x_1, \ldots, x_n) \in \mathsf{C}$ it holds that $G \models \varphi(v_1, \ldots, v_n) \Leftrightarrow H \models \varphi(w_1, \ldots, w_n)$. Note that for labelled graphs, such an interpretation function is implicit: If the indices of $\mathrm{free}(\varphi)$ are a subset of the labels of G, then we can interpret the variables x_i by the vertex with the label i, that is, $\mathfrak{I}(x_i) = \nu(i)$. The semantics of C can then be stated succinctly in terms of label assignments.

▶ **Definition 5** (C semantics of labelled graphs). *Let $\varphi \in \mathsf{C}$ and let G be a labelled graph, such that $\nu(i) \in V(G)$ for all $x_i \in \mathrm{free}(\varphi)$. Then $G \models \varphi$ if*

- $\varphi = (x_i = x_j)$ *and* $\nu(i) = \nu(j)$,
- $\varphi = Ex_ix_j$ *and* $\nu(i)\nu(j) \in E(G)$,
- $\varphi = \neg\psi$ *and* $G \not\models \psi$,
- $\varphi = \psi \vee \vartheta$ *and* $G \models \psi$ *or* $G \models \vartheta$, *or*
- $\varphi = \exists^{\geq t} x_\ell \psi(x_\ell)$ *and there exist distinct* $v_1, \ldots, v_t$, *such that* $G(\ell \to v_i) \models \psi$ *for all* $i \in [t]$.

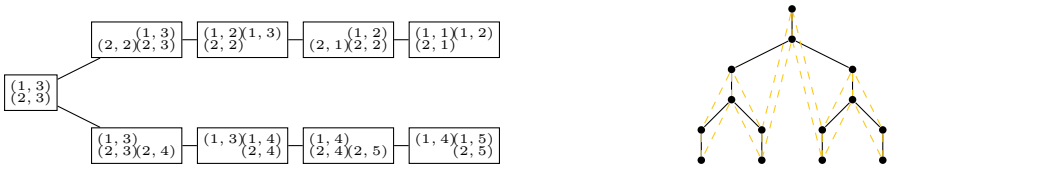
Note that for labelled graphs this is equivalent to extending the standard semantics of FO by the following rule: It is $G, v_1, \ldots, v_n \models \exists^{\geq t} y \psi(x_1, \ldots, x_n, y)$ if there exist distinct elements $u_1, \ldots, u_t \in V(G)$ such that $G \models \psi(v_1, \ldots, v_n, u_i)$ for all $i \in [t]$.

We sometimes write $\exists^{=t} x\varphi(x)$ for $\exists^{\geq t} x\varphi(x) \wedge \neg\exists^{\geq t+1} x\varphi(x)$. We also write $\top$ for $\forall x(x = x)$ and $\bot$ for $\neg\top$. As we already did above, we will often restrict ourselves to the connectives $\neg, \vee$ and the quantifier $\exists^{\geq t}$. This set of symbols is indeed equally expressive by De Morgan's laws and observing that $\exists x\varphi(x) \equiv \exists^{\geq 1} x\varphi(x)$ and $\forall x\varphi(x) \equiv \neg\exists x\neg\varphi(x)$.

The quantifier rank $\mathrm{qr}(\cdot)$ of a formula is defined inductively as follows. It is $\mathrm{qr}(\varphi) = 0$ for atomic formulae φ, $\mathrm{qr}(\neg\varphi) = \mathrm{qr}(\varphi)$, $\mathrm{qr}(\varphi \vee \psi) = \max\{\mathrm{qr}(\varphi), \mathrm{qr}(\psi)\}$ and $\mathrm{qr}(\exists^{\geq t} x\varphi) = 1 + \mathrm{qr}(\varphi)$. The *quantifier-rank-$q$ fragment* C_q of counting first order logic consists of all formulae of quantifier rank at most q.

Instead of restricting the quantifer rank, we can also restrict the number of distinct variables that are allowed to occur in a formula. By C^k we denote the *k-variable fragment* of C, consisting of all formulae using at most k different variables. Similarly, the *k-variable quantifier-rank-q fragment* is defined as $\mathsf{C}^k_q := \mathsf{C}^k \cap \mathsf{C}_q$. Note that these are purely syntactic definitions.

Treewidth and Treedepth. Treewidth is a structural graph parameter that measures how close a graph is to being a tree. It is usually defined in terms of *tree decompositions*.



(a) Tree decomposition of the grid $G_{2\times 5}$ of width 3.

(b) A forest cover of the grid $G_{2\times 7}$ of height 6. The edges of the original grid are dashed.

Figure 1 Tree decomposition and forest cover of grids.

▶ **Definition 6.** *A tree decomposition (T, β) of a graph G is a tree T together with a function $\beta\colon V(T) \to 2^{V(G)}$ satisfying*

- $\bigcup_{t\in V(T)} \beta(t) = V(G)$,
- *for all $uv \in E(G)$ there is a $t \in V(T)$ with $u, v \in \beta(t)$,*
- *for all $v \in V(G)$ the set $\beta^{-1}(v) = \{t \in V(T) \mid v \in \beta(t)\}$ is connected in T.*

The sets $\beta(t)$ are called *bags*. The *width* of a tree decomposition is $\max_{t\in V(T)} |\beta(t)| - 1$. The *treewidth* $\mathrm{tw}(G)$ of a graph G is the minimum width over all tree decompositions of G. We denote the class of graphs of treewidth at most k by $\mathcal{TW}_k$. For an example of a tree decomposition, see Figure 1a.

Treedepth, on the other hand, can be thought of as measuring how close a graph is to being a star. Alternatively, we may think of it as extending the notion of height beyond rooted forests. It is defined for a graph G as the minimum height of a forest F over the vertices of G, such that all edges in G have an ancestor-descendant relationship in F.

▶ **Definition 7.** *A* forest cover *of a graph G is a rooted forest $(F, \boldsymbol{r})$ with $V(F) = V(G)$, such that for every edge $uv \in E(G)$ it holds that either $u \preceq v$ or $v \preceq u$.*

The *treedepth* $\mathrm{td}(G)$ of G is the minimum height of a forest cover of G. We denote the class of all graphs of treedepth at most q by $\mathcal{TD}_q$. For an example of a forest cover, see Figure 1b.

It is possible to construct a tree decomposition from a forest cover $(F, \boldsymbol{r})$. This is achieved by considering a path of bags, each containing the vertices on a path from $\boldsymbol{r}$ to the leaves of F. It is not hard to see that there is an ordering of these bags that satisfies the conditions of Definition 6. This yields the following relation between treedepth and treewidth.

▶ **Fact 8.** *For every graph G, it holds that* $\mathrm{tw}(G) \leq \mathrm{td}(G) - 1$.

Both treewidth and treedepth enjoy characterisations in terms of node searching games, the so called *cops-and-robber games*. The general cops-and-robber game is played on a graph G by Cops, controlling a number of cops; and Robber, controlling a single robber. The cops and the robber are positioned on vertices of G. The goal of Cops is to place a cop on the robber's position, while the robber tries to avoid capture by moving along paths free from cops. The players play in rounds where first Cops announces the next position(s) of the cops (with possible restriction on how many cops may be moved and where they may be positioned) and then Robber moves the robber along some path avoiding all vertices where before and after his move there is a cop. Treewidth can be characterised by the minimum number of cops needed to capture the robber where neither the movement of Cops nor Robber is restricted (see e.g. [29]). A well-known characterisation of treedepth is the minimum number of cops needed if Cops is not allowed to move a cop after it is positioned on the graph (see e.g. [11]). It is equivalent to count the number of rounds the game is played, without restricting the number of cops that can be used by Cops, as long as only one cop can be moved per round. Therefore we use the following unified definition.

▶ **Definition 9** (q-round k-cops-and-robber game). *Let G be a graph and let $k, q \geq 1$. The* monotone q-round k-cops-and-robber game mon-$\mathrm{CR}_q^k(G)$ *is defined as follows:*

We play the game on G' which is constructed from G by adding a disjoint k-vertex clique K. The cop positions are sets $X \in \binom{V(G')}{k}$, the robber position is a vertex $v \in G$. The game is initiated with all cops positioned on K and the robber on any vertex in $V(G)$ of his choice. If the cops are at positions X and robber at vertex v we write (X, v) for the position of the game. For $X \subseteq V(G')$ and $v \in G$, we call the connected component γ_v^X of the graph $G \setminus X$, with $v \in \gamma_v^X$, the robber escape space. *If the cop strategy only depends on γ_v^X and not the precise vertex that the robber occupies, we write (X, γ_v^X) for the position of the game.*

In round $i \leq q$,

- *Cops can move from the set X_i to a set X_{i+1} if $|X_i \cap X_{i+1}| = k - 1$ and $\gamma_v^{X_i} \supseteq \gamma_v^{X_i \cap X_{i+1}}$,*
- *Robber moves along some (possibly empty) path $v_i P v_{i+1}$, where no (inner) vertex is in $X_i \cap X_{i+1}$.*
- *Cops wins if $v_{i+1} \in X_{i+1}$.*

Robber wins if Cops has not won after q rounds.

If we drop the condition $\gamma_v^{X_i} \supseteq \gamma_v^{X_i \cap X_{i+1}}$ in the movement of the cop, we call this the non-monotone *variant of the game and write $\mathrm{CR}_q^k(G)$.*

We write $\mathrm{CR}_q(G)$ instead of $\mathrm{CR}_q^q(G)$ and $\mathrm{CR}^k(G)$ instead of $\mathrm{CR}_{|V(G)|}^k(G)$. Treewidth and treedepth can be characterised in terms of winning strategies for these games.

▶ **Lemma 10** ([29, Theorem 1.4] and [11, Theorem 4]). *Let G be a graph. Let $k, q \geq 1$.*

1. *G has treewidth at most k iff Cops has a winning strategy for $\mathrm{CR}^{k+1}(G)$ iff Cops has a winning strategy for* mon-$\mathrm{CR}^{k+1}(G)$.
2. *G has treedepth at most q iff Cops player has a winning strategy for $\mathrm{CR}_q(G)$ iff Cops has a winning strategy for* mon-$\mathrm{CR}_q(G)$.

3 Graph Decompositions Accounting for Treewidth and Treedepth Simultaneously

In this section, we reconcile treewidth and treedepth by introducing graph decompositions which account simultaneously for depth and width. These efforts yield various equivalent characterisations of the graph class $\mathcal{T}_q^k$, a subclass both of $\mathcal{TW}_{k-1}$ and $\mathcal{TD}_q$, the classes of graphs of treewidth $\leq k-1$ and treedepth $\leq q$, respectively. By introducing a variant of the standard cops-and-robber game which captures $\mathcal{T}_q^k$ and adapting a result from [10], we show that $\mathcal{T}_q^k$ is a proper subclass of $\mathcal{TW}_{k-1} \cap \mathcal{TD}_q$ if q is sufficiently larger than k.

3.1 Four Characterisations for $\mathcal{T}_q^k$

We start with the original definition of the class $\mathcal{T}_q^k$, which incorporates width into forest covers from treedepth. This definition has first been introduced as k-traversal in [1].

▶ **Definition 11.** *Let G be graph and $k \geq 1$. A* k-pebble forest cover *of G is a tuple $(F, \boldsymbol{r}, p)$, where $(F, \boldsymbol{r})$ is a rooted forest over the vertices $V(G)$ and a pebbling function $p\colon V(G) \to [k]$ such that:*

- *If $uv \in E(G)$, then $u \preceq v$ or $v \preceq u$ in $(F, \boldsymbol{r})$.*
- *If $uv \in E(G)$ and $u \prec v$ in $(F, \boldsymbol{r})$, then for every $w \in V(G)$ with $u \prec w \preceq v$ in $(F, \boldsymbol{r})$ it holds that $p(u) \neq p(w)$.*

$(F, \boldsymbol{r}, p)$ has depth $q \geq 1$ if $(F, \boldsymbol{r})$ has height q. We write $\mathcal{T}_q^k$ for the class of all graphs G admitting a k-pebble forest cover of depth q.

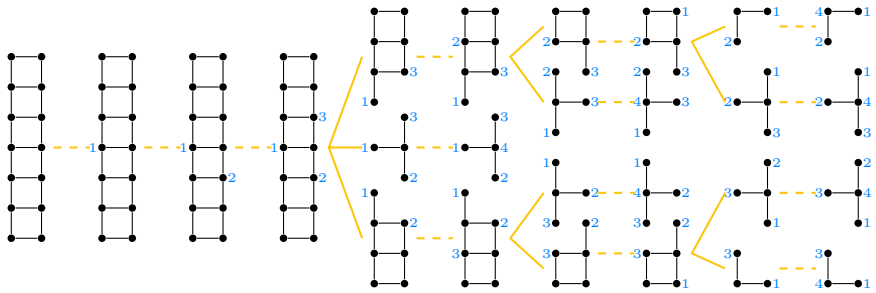


Figure 2 A 4-construction tree for the grid $G_{2\times 7}$ of elimination depth 6. Edges entering elimination nodes are dashed.

The class $\mathcal{T}_q^k$ can also be defined by measuring the depth of a tree decomposition (T, β). Crucially, it does not suffice to take the height of T into account since this notion is not robust. For example, it is well known that one can alter a tree decomposition by subdividing any edge multiple times and copying the bag of the child node. This transformation does neither change the width of the decomposition, nor does it affect the information how to decompose the graph. However, the height of the tree will change drastically under this transformation. It turns out that the following is the right definition:

▶ **Definition 12.** *Let G be a graph. A tuple (T, r, β) is a* rooted tree decomposition *of G if (T, β) is a tree decomposition of G and $r \in V(T)$. The* depth *of (T, β) is*

$$\mathrm{dp}(T,\beta) \coloneqq \min_{r\in V(T)} \mathrm{dp}(T,r,\beta) \qquad \textit{where} \qquad \mathrm{dp}(T,r,\beta) \coloneqq \max_{v\in V(T)} \left|\bigcup_{t\preceq v} \beta(t)\right|.$$

Lastly we define a construction inspired by Dvořák [8], that enables us to use their proof technique to study the expressive power of first-order logic with counting quantifiers using homomorphism indistinguishability (see Figure 2).

▶ **Definition 13.** *Let G be a (possibly labelled) graph. A k-*construction tree *for G is a tuple (T, λ, r), where T is a tree rooted at r and $\lambda\colon V(T) \to \mathcal{G}_k$ is a function assigning k-labelled graphs to the nodes of T such that:*

1. *$\lambda(r) = G$,*
2. *all leaves $\ell \in V(T)$ are assigned fully labelled graphs,*
3. *all internal nodes $t \in V(T)$ with exactly one child t' are* elimination nodes*, that is $\lambda(t)$ can be obtained from $\lambda(t')$ by removing one label, and*
4. *all internal nodes $t \in V(T)$ with more than one child are* product nodes*, that is $\lambda(t)$ is the product of its children.*

The elimination depth *of a construction tree (T, λ, r) is the maximum number of elimination nodes on any path from the root r to a leaf. If G has a k-construction tree of elimination depth $\leq q$, we say that G is* (k, q)-constructible*. We write $\mathcal{L}_q^k$ for the class of all k-labelled (k, q)-constructible graphs.*

It turns out that all three notions coincide.

▶ **Theorem 14.** *Let $k, q \geq 1$. For every graph G, the following are equivalent:*

1. *G is (k, q)-constructible,*
2. *G has a tree decomposition of width $k - 1$ and depth q,*
3. *$G \in \mathcal{T}_q^k$, that is G admits a k-pebble forest cover of depth q.*

The equivalence of Items 1 and 2 is proven by a carefully choosing a tree decomposition such tha the bags are identified with the labelled vertices of the construction tree. For the equivalence of Items 2 and 3, we follow the proof of [3, Theorem 19] and observe that their construction preserves depth. Details can be found in the full version [9].

▶ **Corollary 15.** *Let $k, q \geq 1$. The class $\mathcal{T}_q^k$ is minor-closed, closed under taking disjoint unions, and a subclass of $\mathcal{TW}_{k-1} \cap \mathcal{TD}_q$.*

Given Theorem 14, Corollary 15 is immediate. Dawar, Jakl, and Reggio reduced the proofs of the results of Grohe and Dvořák [13, 8] to a "combinatorial core" [5, Remark 17], which amounts to showing that the classes $\mathcal{TW}_k$ and $\mathcal{TD}_q$ are closed under contracting edges. To that end, Corollary 15 illustrates the benefits of characterising $\mathcal{T}_q^k$ in terms of tree decompositions (Definition 12): Proving that pebble forest covers are preserved under edge contractions requires a non-trivial amount of bookkeeping while the analogous statement for tree decomposition is straightforward.

We conclude with a characterisation of $\mathcal{T}_q^k$ in terms of a cops-and-robber game.

▶ **Lemma 16.** *The Cops win the game* mon-$\mathrm{CR}_q^k(G)$ *if and only if $G \in \mathcal{T}_q^k$.*

The proof of this lemma follows the same strategies as the proof for the monotone version of the cops-and-robber game for treewidth (see for example [24]). Details can be found in the full version [9].

3.2 Separating $\mathcal{T}_q^k$ from $\mathcal{TW}_k \cap \mathcal{TD}_q$ Syntactically

We aim to show that the graph class $\mathcal{T}_q^k$ is a proper subclass of $\mathcal{TW}_k \cap \mathcal{TD}_q$. Since $\mathcal{T}_q^q = \mathcal{TD}_q = \mathcal{TW}_{q-1} \cap \mathcal{TD}_q$, one can only hope to separate the classes if q is larger than k. Using the characterisations of $\mathcal{T}_q^k$ via a cops-and-robber game, we show that there are indeed graphs which do not admit a decomposition where the width is bounded by the treewidth and simultaneously the depth bounded by the treedepth. The graph we consider is the $(h \times \ell)$-grid $G_{h\times\ell}$ with $h < \ell$. It is well known that $\mathrm{tw}(G_{h\times\ell}) = h$ and $\mathrm{td}(G_{h\times\ell}) \leq h\lceil\log(\ell+1)\rceil$, cf. Figure 1. We give a lower bound to the number of rounds q that the robber can survive in a, possibly non-monotone, game $\mathrm{CR}_q^{h+1}(G_{h\times\ell})$, which is linear in both ℓ and h.

▶ **Lemma 17.** *For $1 < h < \ell - 2$ and $q \leq \frac{h(\ell-h+2)}{4}$, Robber wins the game $\mathrm{CR}_q^{h+1}(G_{h\times\ell})$.*

The proof of Lemma 17 builds upon [10]. The winning strategy of Robber is to always stay in the component with the most vertices. We find a lower bound on the size of this component in terms of the number of rounds played and prove that Cops can only force this bound to shrink by two vertices each round, for the majority of the rounds. We additionally show that for $h > 3$ Cops can indeed force the component to shrink by two vertices each round and thus in this case the bound given in Lemma 17 is tight up to an additive term depending only on h.

For $h = 1$, the proof idea of Lemma 17 is not applicable as on a path there are separators of size two that separate the path into three components of roughly equal size. Despite that, one may observe that such a separator does not benefit Cops as from such a position he would always have to combine two of these components into a larger one. Thus Cops can only move along the path and shrink the escape space of Robber by one vertex. This case is covered in the original proof of [10].

▶ **Lemma 18** ([10, Theorems 5 and 7])**.** *Let $\ell \geq 1$. Robber wins the game $\mathrm{CR}_q^2(G_{1\times\ell})$ if and only if $q \leq \lceil\frac{\ell-1}{2}\rceil$.*

With an appropriate choice of ℓ and a small alteration to the graph $G_{k-1\times\ell}$ that ensures that the treedepth of the graph is exactly q, we can prove the following:

▶ **Theorem 2.** *For $q \geq 3$, $\mathcal{T}_q^2 \subsetneq \mathcal{TW}_1 \cap \mathcal{TD}_q$, and for $2 \leq k-1 \leq \frac{q}{3+\log q}$, $\mathcal{T}_q^k \subsetneq \mathcal{TW}_{k-1} \cap \mathcal{TD}_q$.*

Detailed proofs can be found in the full version [9]. The reader should note that the proof of the lower bound on the number of rounds even holds for the non-monotone game, which in turn allows us to lift this result to the semantic level of homomorphism indistinguishability.

4 Homomorphism Indistinguishability

In this section, we turn to investigating $\mathcal{T}_q^k$ in terms of homomorphism indistinguishability. It turns out that the representation of $\mathcal{T}_q^k$ in terms of construction trees offers a great framework for obtaining characterisations of logical equivalence. The general idea will be to use these trees to inductively construct C-formulae that capture homomorphism counts. Not only does this approach generalise results from [8, 13], it also yields an intuitive characterisation of C_q^k-equivalence. This provides a more elementary proof of a result from [5].

Moreover, the constructive nature of our proof strategy proves fruitful in obtaining additional characterisations of fragments of C. The general idea is to impose natural restrictions on the construction trees, such that a fragment $\mathsf{L} \subsetneq \mathsf{C}$ already suffices to capture homomorphism counts. By choosing these restrictions carefully, the resulting subclass of $\mathcal{T}_q^k$ is then still large enough to capture L-equivalence. We illustrate this point by giving a characterisation of *guarded* counting logic GC.

We conclude by *semantically* separating $\mathcal{T}_q^k$ and $\mathcal{TW}_{k-1} \cap \mathcal{TD}_q$. More formally, we show that, for q sufficiently larger than k, there exist graphs G and H which are homomorphism indistinguishable over $\mathcal{T}_q^k$ but have different numbers of homomorphisms from some graph in $\mathcal{TW}_{k-1} \cap \mathcal{TD}_q$.

4.1 Homomorphism Indistinguishability over $\mathcal{T}_q^k$ is C_q^k-Equivalence

In his 2010 paper [8], Dvořák showed that C^k-equivalence is equivalent to homomorphism indistinguishability over $\mathcal{TW}_k$. It turns out that his techniques generalise remarkably well to construction trees. To begin with, we make a few observations on how the operations that make up construction trees interact with homomorphism counts.

First, observe that when a graph F is fully labelled there can be at most one homomorphism $h\colon F \to G$, which is entirely determined by the label positions in G.

▶ **Observation 19.** *Let F be a fully labelled graph and let L_F denote the set of labels. Then there exists a unique homomorphism $h\colon F \to G$ if for all labels $i, j \in L_F$*

- $\nu_F(i) = \nu_F(j) \implies \nu_G(i) = \nu_G(j)$,
- $\nu_F(i)\nu_F(j) \in E(F) \implies \nu_G(i)\nu_G(j) \in E(G)$.

Further, for $h \in \mathrm{Hom}(F_1F_2, G)$ the restrictions $h|_{V(F_1)}$ and $h|_{V(F_2)}$ are homomorphisms, and since two homomorphisms $g\colon F_1 \to G$ and $h\colon F_2 \to G$ must agree on vertices with the same label, $g \sqcup h$ is well-defined and a homomorphism from F_1F_2 to G. This implies the following for products.

▶ **Observation 20.** *For labelled graphs F_1, F_2, G, it holds that* $\hom(F_1F_2, G) = \hom(F_1, G) \cdot \hom(F_2, G)$.

Finally, we can also relate the homomorphism counts from graphs F and F', whenever F' is obtained from F by removing some label ℓ. Then in any homomorphism $h\colon F' \to G$ the image of $\nu_F(\ell)$ is no longer necessarily $\nu_G(\ell)$. Hence, we can obtain $\hom(F,G)$ by moving the label ℓ to different vertices in G and tallying up the homomorphisms from F to those graphs. We may write this succinctly as

$$\hom(F',G) = \sum_{v \in V(G)} \hom(F, G(\ell \to v)),$$

or slightly more verbose as follows.

▶ **Observation 21.** *Let F' be the graph obtained from F by removing a single label ℓ. Then $\hom(F',G) = m$ if and only if there exists a decomposition $m = \sum_i c_i m_i$ with $c_i, m_i \in \mathbb{N}$, such that:*

- *There exist exactly c_i vertices v with $\hom(F, G(\ell \to v)) = m_i$.*
- *There exist exactly $c := \sum_i c_i$ vertices v with $\hom(F, G(\ell \to v)) \neq 0$.*

The crucial insight is that the conditions above are all definable in C. In particular, the condition for fully labelled graphs can be expressed as a conjunction of atomic formulae using at most $|L_F|$ different variables. This allows us to prove the following lemma by induction over a construction tree. The proofs of the lemmas in this section can be found in the full version [9].

▶ **Lemma 22.** *Let $F \in \mathcal{L}_q^k$ be a k-labelled graph, and let $m \geq 0$. Then there exists a formula $\varphi_m \in \mathsf{C}_q^k$ such that for each k-labelled graph G with $L_F \subseteq L_G$, $G \models \varphi_m$ if and only if $\hom(F,G) = m$.*

Ideally, we would like to prove the converse in a similar manner. Given some C_q^k-formula ψ that distinguishes two graphs G and H, construct a graph $F \in \mathcal{L}_q^k$ with $\hom(F,G) \neq \hom(F,H)$ by induction over the structure of ψ. While graphs are too rigid in this regard, such a construction will be possible using *linear combinations* of graphs.[2]

For a class of (labelled) graphs $\mathcal{F}$, we let $\mathbb{R}\mathcal{F}$ be the class of finite formal linear combinations with real coefficients of graphs $F \in \mathcal{F}$. We linearly extend the function hom to $\mathbb{R}\mathcal{G}$ by defining

$$\hom(\mathfrak{F}, G) = \hom(\sum_i c_i F_i, G) := \sum_i c_i \cdot \hom(F_i, G),$$

for $\mathfrak{F} = \sum_i c_i F_i$.

The following observation shows that homomorphism indistinguishability over $\mathcal{F}$ and over $\mathbb{R}\mathcal{F}$ is essentially the same. This allows us to reason about linear combinations instead of graphs.

▶ **Observation 23.** *Let G, H be graphs and let $\mathfrak{F} \in \mathbb{R}\mathcal{F}$. If $\hom(\mathfrak{F}, G) \neq \hom(\mathfrak{F}, H)$, then there is already an $F \in \mathcal{F}$ with $\hom(F,G) \neq \hom(F,H)$.*

The product of two linear combinations is defined in the natural way, where the graph products distribute over the sum. We also remove any graphs with loops that might have been created from the resulting linear combination. This definition preserves the property that $\hom(\mathfrak{F}_1\mathfrak{F}_2, H) = \hom(\mathfrak{F}_1, H)\hom(\mathfrak{F}_2, H)$ and admits the following interpolation lemma.

[2] These linear combinations are called "quantum graphs" in [8].

▶ **Lemma 24** ([8, Lemma 5]). *Let $\mathcal{F}$ be a class of graphs and let $\mathfrak{F} \in \mathbb{R}\mathcal{F}$. If S^-, S^+ are disjoint finite sets of real numbers, then there exists a linear combination $\mathfrak{F}[S^-; S^+] \in \mathbb{R}\mathcal{G}$, such that for any graph G*

- $\hom(\mathfrak{F}[S^-; S^+], G) = 1$ *if* $\hom(\mathfrak{F}, G) \in S^+$, *and*
- $\hom(\mathfrak{F}[S^-; S^+], G) = 0$ *if* $\hom(\mathfrak{F}, G) \in S^-$.

Moreover, if $\mathcal{F}$ is closed under taking products then $\mathfrak{F}[S^-; S^+] \in \mathbb{R}\mathcal{F}$.

With this result, we may construct for a formula $\psi \in \mathsf{C}^k_q$ and $n \in \mathbb{N}$ a linear combination $\mathfrak{F}_{\psi,n}$ such that for all graphs G of size n it holds that $\hom(\mathfrak{F}_{\psi,n}, G) = 1$ if $G \models \psi$ and $\hom(\mathfrak{F}_{\psi,n}, G) = 0$ otherwise. We say that $\mathfrak{F}_{\psi,n}$ *models ψ for graphs of size n.*

▶ **Lemma 25.** *Let $k, q \geq 1$ and let φ be a C^k_q-formula. Then for every $n \geq 1$ there exists an $\mathfrak{F} \in \mathbb{R}\mathcal{L}^k_q$ modelling φ for graphs of size n.*

The proof is by structural induction on φ and exploits how homomorphism counts change under label deletions and taking products. Interpolation is used to define negation and to renormalise homomorphism counts to 0 or 1. The construction has the property that labels in the components of $\mathfrak{F}$ correspond to free variables of φ. This correspondence yields the following corollary.

▶ **Corollary 26.** *Let $k, q \geq 1$ and let φ be a C^k_q-sentence. Then for every $n \geq 1$ there exists an $\mathfrak{F} \in \mathbb{R}\mathcal{T}^k_q$ modelling φ for graphs of size n.*

We can now prove the main result of this section.

▶ **Theorem 27.** *Let $k, q \geq 1$. Two graphs G and H are C^k_q-equivalent if and only if they are homomorphism indistinguishable over $\mathcal{T}^k_q$.*

Proof. Suppose there exists a graph $F \in \mathcal{T}^k_q \subseteq \mathcal{L}^k_q$ with $\hom(F, G) \neq \hom(F, H)$. Then by Lemma 22 there exist C^k_q-sentences φ^F_m for $m \geq 0$ such that $G \models \varphi^F_m$ iff $\hom(F, G) = m$. Consequently, there exists an m with $G \models \varphi^F_m$ and $H \not\models \varphi^F_m$, so G and H cannot satisfy the same C^k_q-sentences.

Suppose now that there exists a sentence $\varphi \in \mathsf{C}^k_q$ with $G \models \varphi$ and $H \not\models \varphi$. Without loss of generality, we may assume $|G| = |H| =: n$. Then, by Corollary 26, there is an $\mathfrak{F} \in \mathbb{R}\mathcal{T}^k_q$ that models φ for graphs of size n, that is, $\hom(\mathfrak{F}, G) \neq \hom(\mathfrak{F}, H)$. By Observation 23, this already implies the existence of an $F \in \mathcal{T}^k_q$ with $\hom(F, G) \neq \hom(F, H)$. ◀

By dropping the restriction on one of the two parameters in Theorem 27, we recover the original results of Dvořák [8] and Grohe [13]:

▶ **Corollary 28.** *Let $k, q \geq 1$. Let G and H be graphs.*

1. *G and H are C^k-equivalent iff they are homomorphism indistinguishable over $\mathcal{TW}_{k-1}$.*
2. *G and H are C_q-equivalent iff they are homomorphism indistinguishable over $\mathcal{TD}_q$.*

4.2 Guarded Fragments

Given the constructive nature of the arguments in Section 4.1, it is interesting to investigate whether the same strategy can be used to obtain results for different fragments of C by restricting construction trees in some way. An example where this works well is guarded counting logic GC.

In the guarded fragment GC, quantifiers are restricted to range over the neighbours of a vertex. Formally, we require that quantifiers only occur in the form $\exists^{\geq t} y(Exy \wedge \psi(z_1, \ldots, z_n, y))$, where x and y are distinct variables.

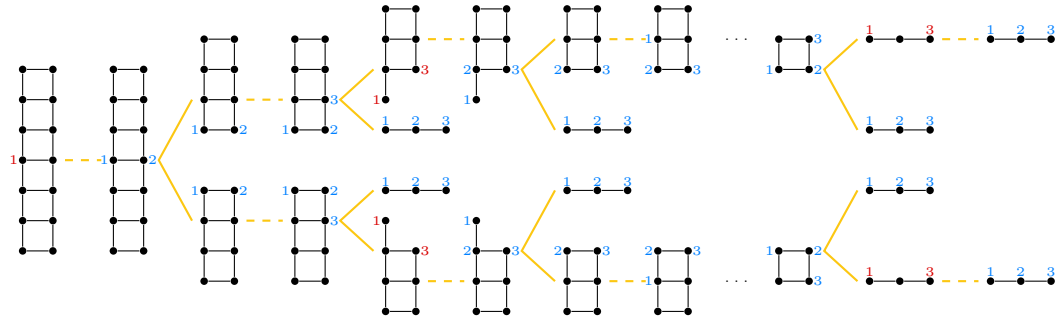


Figure 3 A guarded 3-construction tree of elimination depth 7 for the grid $G_{2\times 7}$ with one labelled vertex. Edges entering elimination nodes are dashed. At every labelled graph, those labels that may be removed are marked blue, those that may not be removed are marked red. The dotted omitted part of the construction tree follows the same pattern.

Since GC-formulae necessarily have a free variable, it is not immediately obvious how to define GC-equivalence on graphs. One option is to consider GC-equivalence of graphs together with a distinguished vertex. This works, and we will, in fact, obtain a characterisation for precisely this relation. However, we would prefer to study the landscape of homomorphism indistinguishability relations on graphs without additional structure. The following natural definition of GC-equivalence allows us to lift our result to graphs without a dinstinguished vertex.

▶ **Definition 29** (GC-equivalence). *Let G and H be unlabelled graphs. We say that G and H are GC-equivalent, in symbols $G \equiv_{\mathsf{GC}} H$, if there exists a bijection $f\colon V(G) \to V(H)$ such that $G, v \models \varphi(x) \iff H, f(v) \models \varphi(x)$ for all $v \in V(G)$ and $\varphi \in \mathsf{GC}$.*

To apply our proof strategy to GC, we need to restrict the construction trees such that guarded quantifiers suffice to express homomorphism counts. Observe that the quantifiers are only needed to describe how the number of homomorphisms $F \to G$ changes by removing a label from F. More precisely, we make use of the fact that removing a label ℓ from F is the same as moving it around in G and tallying up the resulting homomorphisms. Now if ℓ is adjacent to some other label ℓ', then the only positions of ℓ in G that contribute to the final homomorphism count are adjacent to ℓ'. Consequently, it will suffice to quantify over the neighbours of ℓ'.

▶ **Definition 30.** *Let $k, q \geq 1$. By $\mathcal{GL}_q^k$ we denote the class of k-labelled graphs that admit a k-construction tree of elimination-depth q with the additional restriction that labels can only be removed if they have a labelled neighbour.*

We observe that in Figure 2 there are nodes where labels without labelled neighbors are removed. In Figure 3, we depict a construction tree without such nodes of the same graph. We remark that all graphs in $\mathcal{GL}_q^k$ are labelled, as a single label can never be removed. Under these restrictions, the argument from Lemma 22 goes through using only guarded quantifiers.

▶ **Lemma 31.** *Let $F \in \mathcal{GL}_q^k$. Then for each $m \geq 0$ there is a formula $\varphi_m \in \mathsf{GC}_q^k$ such that for appropriately labelled graphs G it holds that $\hom(F, G) = m$ iff $G \models \varphi_m$.*

The proof of the converse – showing that there exists for each $\psi \in \mathsf{GC}_q^k$ an $\mathfrak{F} \in \mathbb{R}\mathcal{GL}_q^k$ modelling ψ – also goes through nearly unchanged.

▶ **Lemma 32.** *Let $\varphi \in \mathsf{GC}_q^k$. Then there is an $\mathfrak{F} \in \mathbb{R}\mathcal{GL}_q^k$ modelling φ for graphs of size n.*

The analogues of these two lemmas already sufficed to prove Theorem 27. Here, however, we still need to be mindful of any remaining labels. Concretely, Lemma 31 and Lemma 32 imply the following for GC sentences.

▶ **Corollary 33.** *Let G, v and H, w be graphs together with a single labelled vertex. Then the following are equivalent.*
1. *For all $\psi(x) \in \mathsf{GC}_q^k$, it holds $G, v \models \psi(x) \iff H, w \models \psi(x)$.*
2. $\hom(F, G) = \hom(F, H)$ *for all $F \in \mathcal{GL}_q^k$.*

While this is already a nice result, ideally we would like to make a statement about general, unlabelled, graphs. Fortunately, simply removing all labels from $F \in \mathcal{GL}_q^k$ turns out to induce the equivalence relation described in Definition 29.

Let us denote by $\mathcal{GT}_q^k$ the class of graphs in $\mathcal{GL}_q^k$ with all labels removed. Then we can state the following theorem, characterising GC_q^k-equivalence in terms of homomorphism indistinguishability. The details can be found in the full version [9].

▶ **Theorem 1.** *Let $k, q \geq 1$. Two graphs G and H are GC_q^k-equivalent if and only if they are homomorphism indistinguishable over $\mathcal{GT}_q^k$.*

We remark that in [2] the logic GC was studied with comonadic means. In this work, winning strategies for Duplicator in guarded bisimulation games were characterised as coKleisli morphisms with respect to a suitably defined comonad. This is in contrast to the comonadic Lovász-type theorem of [5] which applies to logical equivalences which can be characterised as coKleisli isomorphisms. Thus, Theorem 1 does not seem to be immediate from [2, 5].

4.3 Separating $\mathcal{T}_q^k$ from $\mathcal{TW}_{k-1} \cap \mathcal{TD}_q$ Semantically

By Theorem 2, the graph class $\mathcal{T}_q^k$ is a proper subclass of $\mathcal{TW}_{k-1} \cap \mathcal{TD}_q$. Despite that, it could well be that the homomorphism indistinguishability relations of the two graph classes (and via Theorem 27 also C_q^k-equivalence) coincide, i.e. $G \equiv_{\mathcal{T}_q^k} H$ if and only if $G \equiv_{\mathcal{TW}_{k-1} \cap \mathcal{TD}_q} H$ for all graphs G and H. It turns out that this is not the case.

In general, establishing that the homomorphism indistinguishability relations $\equiv_{\mathcal{F}_1}$ and $\equiv_{\mathcal{F}_2}$ of two graph classes $\mathcal{F}_1 \neq \mathcal{F}_2$ are distinct is a notoriously hard task. Pivotal tools for accomplishing this were introduced by Roberson in [26]. He defines the *homomorphism distinguishing closure* $\mathrm{cl}(\mathcal{F})$ of a graph class $\mathcal{F}$ as the graph class

$$\mathrm{cl}(\mathcal{F}) \coloneqq \{F \text{ graph} \mid \forall G, H.\ G \equiv_{\mathcal{F}} H \implies \hom(F, G) = \hom(F, H)\}.$$

A graph class $\mathcal{F}$ is *homomorphism distinguishing closed* if $\mathcal{F} = \mathrm{cl}(\mathcal{F})$. This is the case if and only if for every $F \notin \mathcal{F}$ there exist two graphs G and H homomorphism indistinguishable over $\mathcal{F}$ and satisfying that $\hom(F, G) \neq \hom(F, H)$. Therefore, homomorphism distinguishing closed graph classes may be thought of as maximal in terms of homomorphism indistinguishability.

Roberson conjectures that *every graph class which is closed under taking minors and disjoint unions is homomorphism distinguishing closed.* A confirmation of this conjecture would aid separating homomorphism indistinguishability relations and in turn all equivalence relations between graphs which have such characterisations, cf. [27]. In particular, it would readily imply that $\equiv_{\mathcal{T}_q^k}$ and $\equiv_{\mathcal{TW}_{k-1} \cap \mathcal{TD}_q}$ are distinct, cf. Corollary 15. Unfortunately, the conjecture's assertion is only known to be true for the class of planar graphs [26], $\mathcal{TW}_k$ [22] and graph classes arising from finite graph classes [28]. Towards separating $\equiv_{\mathcal{T}_q^k}$ and $\equiv_{\mathcal{TW}_{k-1} \cap \mathcal{TD}_q}$, we first add to this list by proving the following:

▶ **Theorem 3.** *For $q \geq 1$, the class $\mathcal{TD}_q$ is homomorphism distinguishing closed.*

The proof of Theorem 3 follows the proof in [22] of the assertion that the class $\mathcal{TW}_k$ is homomorphism distinguishing closed for all $k \geq 0$. Central to it is a construction of highly similar graphs from [26] which is reminiscent of the CFI-construction [4]. With these ingredients, it suffices to prove that Duplicator wins the model comparison game characterising C_q-equivalence on these CFI-like graphs constructed over a graph of high treedepth. To that end, we build a Duplicator strategy from a Robber strategy for the game from Definition 9. The connection between model comparison and node searching games via CFI-constructions is well-known [17, 6].

Crucial for the aforementioned argument is that Robber wins the *non-monotone* node searching game characterising bounded treedepth. Indeed, it cannot be assumed that Cops plays monotonously since he must shadow Spoiler's moves. Since we are unable to establish a non-monotone node searching game characterising $\mathcal{T}_q^k$, we cannot conclude along the same lines that $\mathcal{T}_q^k$ is homomorphism distinguishing closed. Nevertheless, we separate $\equiv_{\mathcal{T}_q^k}$ and $\equiv_{\mathcal{TW}_{k-1} \cap \mathcal{TD}_q}$. The details are deferred to the full version [9].

▶ **Theorem 4.** *For $q \geq 3$, $\mathrm{cl}(\mathcal{T}_q^2) \subsetneq \mathcal{TW}_1 \cap \mathcal{TD}_q$, and for $2 \leq k-1 \leq \frac{q}{3+\log q}$, $\mathrm{cl}(\mathcal{T}_q^k) \subsetneq \mathcal{TW}_{k-1} \cap \mathcal{TD}_q$.*

Proving that $\mathcal{T}_q^k$ is characterised by Robber winning the (non-monotone) game CR_q^k, c.f. Lemma 16, would immediately imply that $\mathcal{T}_q^k$ is homomorphism distinguishing closed.

In the introduction, we mentioned that it is tempting to assume that C_q^k-equivalence coincides with homomorphism indistinguishability over $\mathcal{TW}_{k-1} \cap \mathcal{TD}_q$ because $\mathsf{C}_q^k = \mathsf{C}^k \cap \mathsf{C}_q$. However, our results imply that there are properties definable in both C^k and C_q that are not definable in C_q^k.

▶ **Corollary 34.** *For $2 \leq k-1 \leq \frac{q}{3+\log q}$, there exist sentences $\varphi \in \mathsf{C}^k$ and $\psi \in \mathsf{C}_q$ such that $\varphi \equiv \psi$, but for all sentences $\vartheta \in \mathsf{C}_q^k$ it holds that $\vartheta \not\equiv \varphi$.*

Proof. By Theorem 4, for suitable k, q, there exist graphs G and H such that $G \equiv_{\mathcal{T}_q^k} H$ and there exists $F \in \mathcal{TW}_{k-1} \cap \mathcal{TD}_q$ such that $m := \hom(F, G) \neq \hom(F, H)$. By Lemma 22 and Theorem 14, there exist sentences $\varphi \in \mathsf{C}^k$ and $\psi \in \mathsf{C}_q$ such that $\hom(F, K) = m \iff K \models \varphi \iff K \models \psi$ for every graph K. However, this property cannot be defined in C_q^k since G and H satisfy the same C_q^k-sentences by Theorem 27. ◀

5 Outlook

We studied the expressive power of the counting logic fragment C_q^k with tools from homomorphism indistinguishability. After giving an elementary and uniform proof of theorems from [8, 13, 5], we showed that the graph class $\mathcal{T}_q^k$, whose homomorphism indistinguishability relation characterises C_q^k-equivalence, is a proper subclass of $\mathcal{TW}_{k-1} \cap \mathcal{TD}_q$. Finally, we showed that homomorphism indistinguishability over $\mathcal{T}_q^k$ is not the same as homomorphism indistinguishability over $\mathcal{TW}_{k-1} \cap \mathcal{TD}_q$.

The main problem remaining open is to tighten Theorem 4 by proving that the graph class $\mathcal{T}_q^k$ is homomorphism distinguishing closed, as predicted by Roberson's conjecture. Our contribution in this direction is a reduction to a purely graph theoretic problem: Proving that the class $\mathcal{T}_q^k$ is characterised by a *non-monotone* cops-and-robber game, cf. Lemma 16, would be sufficient to yield this claim. Exploring whether intertwining node searching and model comparison games can help to verify Roberson's conjecture in other cases seems a tempting direction for future research.

With slight reformulations, our results might yield insights into the ability of the Weisfeiler–Leman algorithm to determine subgraph counts after a fixed number of rounds [25, 22].

References

1. Samson Abramsky, Anuj Dawar, and Pengming Wang. The pebbling comonad in Finite Model Theory. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavík, Iceland, June 20-23, 2017*, pages 1–12. IEEE Computer Society, 2017. doi:10.1109/LICS.2017.8005129.
2. Samson Abramsky and Dan Marsden. Comonadic Semantics for Guarded Fragments. In *Proceedings of the 36th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '21. IEEE Press, 2021. doi:10.1109/LICS52264.2021.9470594.
3. Samson Abramsky and Nihil Shah. Relating structure and power: Comonadic semantics for computational resources. *Journal of Logic and Computation*, 31(6):1390–1428, September 2021. doi:10.1093/logcom/exab048.
4. Jin-Yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, December 1992. doi:10.1007/BF01305232.
5. Anuj Dawar, Tomáš Jakl, and Luca Reggio. Lovász-Type Theorems and Game Comonads. In *2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–13, June 2021. doi:10.1109/LICS52264.2021.9470609.
6. Anuj Dawar and David Richerby. The power of counting logics on restricted classes of finite structures. In Jacques Duparc and Thomas A. Henzinger, editors, *Computer Science Logic*, pages 84–98. Springer Berlin Heidelberg, 2007. doi:10.1007/978-3-540-74915-8_10.
7. Holger Dell, Martin Grohe, and Gaurav Rattan. Lovász Meets Weisfeiler and Leman. *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, pages 40:1–40:14, 2018. doi:10.4230/LIPICS.ICALP.2018.40.
8. Zdeněk Dvořák. On recognizing graphs by numbers of homomorphisms. *Journal of Graph Theory*, 64(4):330–342, August 2010. doi:10.1002/jgt.20461.
9. Eva Fluck, Tim Seppelt, and Gian Luca Spitzer. Going Deep and Going Wide: Counting Logic and Homomorphism Indistinguishability over Graphs of Bounded Treedepth and Treewidth, 2023. doi:10.48550/arXiv.2308.06044.
10. Martin Fürer. Weisfeiler-Lehman Refinement Requires at Least a Linear Number of Iterations. In Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen, editors, *Automata, Languages and Programming, 28th International Colloquium, ICALP 2001, Crete, Greece, July 8-12, 2001, Proceedings*, volume 2076 of *Lecture Notes in Computer Science*, pages 322–333. Springer, 2001. doi:10.1007/3-540-48224-5_27.
11. Archontia C. Giannopoulou, Paul Hunter, and Dimitrios M. Thilikos. LIFO-search: A min–max theorem and a searching game for cycle-rank and tree-depth. *Discrete Applied Mathematics*, 160(15):2089–2097, October 2012. doi:10.1016/j.dam.2012.03.015.
12. Georg Gottlob, Nicola Leone, and Francesco Scarcello. Robbers, marshals, and guards: game theoretic and logical characterizations of hypertree width. *Journal of Computer and System Sciences*, 66(4):775–808, 2003. Special Issue on PODS 2001. doi:10.1016/S0022-0000(03)00030-8.
13. Martin Grohe. Counting Bounded Tree Depth Homomorphisms. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '20, pages 507–520, New York, NY, USA, 2020. Association for Computing Machinery. event-place: Saarbrücken, Germany. doi:10.1145/3373718.3394739.
14. Martin Grohe. word2vec, node2vec, graph2vec, X2vec: Towards a Theory of Vector Embeddings of Structured Data. In Dan Suciu, Yufei Tao, and Zhewei Wei, editors, *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2020, Portland, OR, USA, June 14-19, 2020*, pages 1–16. ACM, 2020. doi:10.1145/3375395.3387641.
15. Martin Grohe. The Logic of Graph Neural Networks. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*, pages 1–17. IEEE, 2021. doi:10.1109/LICS52264.2021.9470677.

16 Martin Grohe, Gaurav Rattan, and Tim Seppelt. Homomorphism Tensors and Linear Equations. In Mikołaj Bojańczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, volume 229 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 70:1–70:20, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2022.70.

17 Lauri Hella. Logical Hierarchies in PTIME. *Information and Computation*, 129(1):1–19, August 1996. doi:10.1006/inco.1996.0070.

18 Stephan Kreutzer and Sebastian Ordyniak. Digraph decompositions and monotonicity in digraph searching. *Theor. Comput. Sci.*, 412(35):4688–4703, 2011. doi:10.1016/j.tcs.2011.05.003.

19 László Lovász. Operations with structures. *Acta Mathematica Academiae Scientiarum Hungarica*, 18(3):321–328, September 1967. doi:10.1007/BF02280291.

20 Laura Mančinska and David E. Roberson. Quantum isomorphism is equivalent to equality of homomorphism counts from planar graphs. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 661–672, 2020. doi:10.1109/FOCS46700.2020.00067.

21 Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks. *AAAI*, 33:4602–4609, July 2019. doi:10.1609/aaai.v33i01.33014602.

22 Daniel Neuen. Homomorphism-Distinguishing Closedness for Graphs of Bounded Tree-Width, April 2023. doi:10.48550/arXiv.2304.07011.

23 Hoang Nguyen and Takanori Maehara. Graph homomorphism convolution. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 7306–7316. PMLR, 13–18 July 2020. URL: https://proceedings.mlr.press/v119/nguyen20c.html.

24 Roman Rabinovich. *Graph complexity measures and monotonicity*. PhD thesis, RWTH Aachen University, 2013. URL: https://publications.rwth-aachen.de/record/230227.

25 Gaurav Rattan and Tim Seppelt. Weisfeiler–Leman and Graph Spectra. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2268–2285. Society for Industrial and Applied Mathematics, 2023. doi:10.1137/1.9781611977554.ch87.

26 David E. Roberson. Oddomorphisms and homomorphism indistinguishability over graphs of bounded degree, June 2022. doi:10.48550/arXiv.2206.10321.

27 David E. Roberson and Tim Seppelt. Lasserre Hierarchy for Graph Isomorphism and Homomorphism Indistinguishability. In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *50th International Colloquium on Automata, Languages, and Programming (ICALP 2023)*, volume 261 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 101:1–101:18, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2023.101.

28 Tim Seppelt. Logical Equivalences, Homomorphism Indistinguishability, and Forbidden Minors. In Jérôme Leroux, Sylvain Lombardy, and David Peleg, editors, *48th International Symposium on Mathematical Foundations of Computer Science (MFCS 2023)*, volume 272 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 82:1–82:15, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.MFCS.2023.82.

29 Paul D. Seymour and Robin Thomas. Graph Searching and a Min-Max Theorem for Tree-Width. *J. Comb. Theory, Ser. B*, 58(1):22–33, 1993. doi:10.1006/jctb.1993.1027.

30 Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? In *International Conference on Learning Representations*, 2019. URL: https://openreview.net/forum?id=ryGs6iA5Km.

Realizability Models for Large Cardinals

Laura Fontanella
Univ. Paris Est Créteil, LACL, F-94010, France

Guillaume Geoffroy
Université Paris Cité, laboratoire IRIF, France

Richard Matthews
Univ. Paris Est Créteil, LACL, F-94010, France

Abstract

Realizabilty is a branch of logic that aims at extracting the computational content of mathematical proofs by establishing a correspondence between proofs and programs. Invented by S.C. Kleene in the 1945 to develop a connection between intuitionism and Turing computable functions, realizability has evolved to include not only classical logic but even set theory, thanks to the work of J-L. Krivine. Krivine's work made possible to build realizability models for Zermelo-Frænkel set theory, ZF, assuming its consistency. Nevertheless, a large part of set theoretic research involves investigating further axioms that are known as *large cardinals axioms*; in this paper we focus on four large cardinals axioms: the axioms of (strongly) inaccessible cardinal, Mahlo cardinals, measurable cardinals and Reinhardt cardinals. We show how to build realizability models for each of these four axioms assuming their consistency relative to ZFC or ZF.

2012 ACM Subject Classification Theory of computation → Logic; Theory of computation → Proof theory; Theory of computation → Type theory

Keywords and phrases Logic, Classical Realizability, Set Theory, Large Cardinals

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.28

Funding *Richard Matthews*: DIM RFSI 21R03101S.

Acknowledgements We would like to thank Jean-Louis Krivine for many fruitful discussions that set the main ideas for this work.

1 Introduction

Realizability is an extension of the *proofs-as-programs* correspondence also known as the *Curry-Howard isomorphism*. In realizability, a theory (or a logical system) is interpreted in a model of computation by establishing a correspondence between formulae and programs in a way that is compatible with the rules of deduction. For instance, a realizer of an implication $A \to B$ is a program which, whenever applied to a realizer of A, returns a realizer of B. The origins of realizability date back to S.C. Kleene's work in constructive mathematics in 1945 [11]: Kleene's realizability formalized the intuitionistic view that proofs are algorithms (computable functions) by interpreting proofs in Heyting arithmetic as recursive functions. In the 90's, the work of T. Griffin [6] led to pass the barrier of intuitionistic logic and to extend the Curry-Howard correspondence to classical logic by using the λ_c-calculus, an extension of λ-calculus that formalizes computation in the programming language Scheme (for a presentation of the λ-calculus we refer to Barendregt's book [1], for a presentation of the λ_c-calculus we refer to [3]). J.-L. Krivine developed a method for realizing not only classical logic, but even Zermelo-Frænkel set theory, ZF (see [12] and [13]) using *realizability algebras* which are generalizations of the notion of Boolean algebra involving programs and stacks (realizability algebras will be presented in Section 2). Krivine's technique combines

intuitionistic set theory, IZF, with a double negation translation of formulas. In this matter the work of H. Friedman [4] on IZF was crucial as it showed that IZF is equiconsistent with ZF and gave a natural way to interpret the classical theory within the intuitionistic one.

The method can also be seen as a generalization of the method of forcing in set theory. This is because every Boolean-valued model can be naturally interpreted as a realizability algebra; moreover, this interpretation is done in such a way that the two resulting models prove the same statements in some precise sense (we refer to section 19 of [19] for the details of this translation). Nevertheless, from a computational point of view, forcing models are not very informative since all realizers are interpreted as the same element (the bottom element of the boolean algebra).

For a long time it remained an open problem whether or not it was possible to realize the Axiom of Choice, AC; recent work of Krivine [16] shows that it is indeed possible to build a realizability model for AC (although, it remains unclear what would be an explicit realizer for AC in this model). Research in contemporary set theory is not limited to the axioms of ZF or ZFC (i.e. ZF plus AC), with an active area of research being the study of *large cardinals axioms*. These are strong axioms of infinity that assert the existence of uncountable cardinals with various closure properties. Large cardinals axioms can be ordered by their consistency strength and they all entail the existence of a *set* which satisfies all of the axioms of ZF. It follows by K. Gödel's second incompleteness theorem that the existence of large cardinals cannot be proven within ZF. Nevertheless, these axioms have many applications to various areas of mathematics and computer science (for a more detailed presentation of large cardinals we refer to A. Kanamori's book [10])

We address the problem of whether or not large cardinals axioms can be realized, and we focus on four major large cardinals notions: (strongly) *inaccessible cardinals*, *Mahlo cardinals*, *measurable cardinals* and *Reinhardt cardinals*. Inaccessible cardinals are uncountable cardinals that imply the existence of uncountable Grothendieck universes (see [24]). Mahlo cardinals imply strong reflection properties which have been used in type theories, such as Agda, to produce type universes which contain inductive-recursive types. The strongest known version of type theory for which there exists a constructive justification is a system of Martin-Löf type theory with a Mahlo universe, MLM, which was introduced by A. Setzer (see [22]). M. Rathjen showed that constructive set theory plus the axiom that asserts the existence of Mahlo cardinals has a canonical interpretation in Setzer's type theory (see [21]). A measurable cardinal is a cardinal κ for which there exists a non-trivial κ-additive, 0-1-valued measure on the power set of κ. As proved by A. Blass in [2], the existence of measurable cardinals is equivalent to the existence of an exact functor $F : \mathrm{Set} \to \mathrm{Set}$ that is not naturally isomorphic to the identity. Reinhardt cardinals generalize the notion of measurable cardinal and imply the existence of a non-trivial elementary embedding of the universe of sets into itself. The existence of Reinhardt cardinals is inconsistent with ZFC by Kunen's inconsistency theorem [17], so they are defined only in the context of ZF.

In this paper we show that these four large cardinals axioms can be realized within Krivine's framework. We consider for each of these large cardinal axioms φ an equivalent large cardinal axiom φ^* in the context of ZF, then we build realizability models for the theory "ZF plus φ^*" assuming its consistency. The work of H. Friedman and A. Ščedrov [5] provided a suitable formulation of large cardinals in intuitionistic set theory which made it possible to integrate these large cardinals notions into Krivine's machinery. We shall point out that not only do we build realizaiblity models for these large cardinals notions assuming their consistency with ZFC or ZF, but we prove that *any* realizability algebra of size less than the large cardinals considered preserves these large cardinals axioms.

The paper is structured as follows. We first introduce the technique for building realizability models for ZF in Sections 2–4. In Section 5 we illustrate the method of reish names, or *recursive* names, for transferring properties of sets and functions to realizability algebras. In Section 6 we discuss some useful relativization properties for transitive sets. In Section 7 we show how to preserve the axiom of inaccessible cardinals by realizability algebras. In Section 8, we show how to preserve the axiom of Mahlo cardinals by realizability algebras. Section 9 is devoted to realizability models for the second ordered set theory, GB. Finally, in Section 10 we show how to preserve by realizability algebras the axioms of measurable and Reinhardt cardinals.

2 Realizability algebras

In this section, we present *Realizability Algebras*, which are the main building blocks for the construction of realizability models for set theory. We shall being by briefly explaining the main intuition behind this construction. We start with a model of set theory and we will use programs and stacks to evaluate the potential truth and falsity values of set theoretic statements. For computational reasons we work with a non extensional version of set theory, called ZF_ε, that involve two membership relations: the usual one and a strict non-extensional relation. We will use the terms of the λ_c-calculus (a variant of λ-calculus that include as a term the operator *call-with-current-continuation*) to evaluate the truth value of formulas in the language of ZF_ε. On the other hand, we will use stacks, namely sequences of λ_c-terms, to evaluate the falsity values of such formulas. Truth values and falsity values will be related to each other, so that a λ_c-term is in the truth value of a formula (we say that it "realizes the formula"), if it is somehow "incompatible" with every stack in the falsity value of that formula. These definitions will respect certain logical constraints such as no stack can be in the falsity value of $\top$, and every stack is in the falsity value of $\bot$. Then we choose some privileged collection of λ_c-terms that we call *realizers* and we will show that the set of formulas that are realized by some realizer forms a consistent theory which includes ZF_ε and is closed under the rules of derivation of classical natural deduction. Finally, a realizability model will be a model of such a theory. Since ZF_ε is a conservative extension of ZF, such a model will induce a model of ZF.

The main ingredients of realizability algebras are λ_c*-terms*, *stacks* and *processes* which we define next. We will give the definition in full generality, in particular allowing for non-empty sets of *special instructions* and *stack bottoms*. These are customisable constants which can be added to our realizability algebras to ensure the models satisfies additional principles. For example, if the algebra is countable and contains the special instruction *quote* then one can prove Dependent Choice holds in the model. However, all the statements in this paper will be realized by terms of the λ_c-calculus. Therefore we will not need any special instructions but the arguments will still go through if they are present.

▶ **Definition 1.** *Let* V *be a model of* ZF *and let* A, B *be two sets in* V*:*

We let $\Lambda^{open}_{A,B}$ *and* $\Pi_{A,B}$ *denote the elements of* V *defined by the following grammars, modulo* α*-equivalence. Their elements are called respectively* λ_c-terms *and* stacks*:*

$\Lambda^{open}_{A,B}$ (λ_c**-terms**) :

t, u	$::=$	x	*(variable; we choose a set of variables that is countable in* V*)*
	$\mid$	tu	*(application)*
	$\mid$	$\lambda x.t$	*(abstraction;* x *is a variable and* t *is a* λ_c*-term)*
	$\mid$	cc	*(call-with-current-continuation)*
	$\mid$	k_π	*(continuation constants;* π *is a stack)*
	$\mid$	ξ_α	*(special instructions;* $\alpha \in A$*)*

$\Pi_{A,B}$ (**Stacks**) :

$$\begin{array}{rcll} \pi & ::= & \omega_\beta & \textit{(stack bottoms; } \beta \in B\textit{)} \\ & | & t.\pi & \textit{(}t \textit{ is a closed } \lambda_c\textit{-term and } \pi \textit{ is a stack)} \end{array}$$

- $\Lambda_{A,B} \in \mathrm{V}$ *denotes the set of all closed* λ_c*-terms,*
- $\mathcal{R}_{A,B} \in \mathrm{V}$ *denotes the set of all closed* λ_c*-terms that contain no occurrence of a continuation constant. Such terms are called* realizers*.*
- $\Lambda_{A,B} \star \Pi_{A,B} \in \mathrm{V}$ *denotes the cartesian product* $\Lambda_{A,B} \times \Pi_{A,B}$*. Its elements are called* processes*. We will write* $t \star \pi$ *for* $(t, \pi) \in \Lambda_{A,B} \star \Pi_{A,B}$*.*

Application on λ_c-terms is left associative ($tu_1u_2 \cdots u_n$ means $(\cdots((tu_1)u_2)\cdots)u_n$) and has higher priority than abstraction ($\lambda x.tu$ means $\lambda x.(tu)$). We define some rules of reduction on the set of processes through the notion of *evaluation*.

▶ **Definition 2.** *Let* V *be a model of* ZF *and let* A, B *be two sets in* V*.*

- $\prec_{A,B} \in \mathrm{V}$ *is called the* evaluation preorder *and denotes the smallest preorder on* $\Lambda_{A,B} \star \Pi_{A,B}$ *such that:*

$$\begin{array}{rcll} tu \star \pi & \succ_{A,B} & t \star u.\pi & \textit{(push)} \\ \lambda x.t \star u.\pi & \succ_{A,B} & t[x := u] \star \pi & \textit{(grab)} \\ \mathrm{cc} \star t.\pi & \succ_{A,B} & t \star \mathrm{k}_\pi.\pi & \textit{(save)} \\ \mathrm{k}_{\pi'} \star t.\pi & \succ_{A,B} & t \star \pi' & \textit{(restore).} \end{array}$$

Note that there is no evaluation rule for the special instructions, thus $\prec_{A,B}$ treats the special instructions as inert constants; depending on the context we may define other evaluation relations with specific evaluation rules for the special instructions. If A and B can be well-ordered (which is always the case if V satisfies the Axiom of Choice), then the cardinality (from the point of view of V) of $\Lambda_{A,B}$, $\Pi_{A,B}$, $\mathcal{R}_{A,B}$ and $\Lambda_{A,B} \star \Pi_{A,B}$ is the maximum of the cardinality of A, the cardinality of B and $\aleph_0$.

▶ **Definition 3.** *Let* V *be a model of* ZF*. A* realizability algebra *in* V *is a tuple* $\mathcal{A} = (A, B, \perp\!\!\!\perp)$ *such that:*

- $\mathcal{A} \in \mathrm{V}$ *(i.e.* $A \in \mathrm{V}$*,* $B \in \mathrm{V}$ *and* $\perp\!\!\!\perp \in \mathrm{V}$*);*
- $\perp\!\!\!\perp$ *is a subset of* $\Lambda_{A,B} \star \Pi_{A,B}$ *that is a final segment for* $\succ_{A,B}$*, i.e. if* $t \star \pi \succ_{A,B} t' \star \pi'$ *and* $t' \star \pi' \in \perp\!\!\!\perp$*, then* $t \star \pi \in \perp\!\!\!\perp$*. It is called the* pole *of the realizability algebra.*

Given a model V of ZF, recall that the *Von Neumann hierarchy* is a collection of sets V_α indexed by ordinals and defined by transfinite recursion as follows: $\mathrm{V}_\alpha = \bigcup_{\beta<\alpha} \mathcal{P}(\mathrm{V}_\beta)$. The Axiom of Foundation implies that $\mathrm{V} = \bigcup_{\alpha \in \mathrm{ORD}} \mathrm{V}_\alpha$; thus every set belongs to some V_α and the *rank* ordinal of a set a, denoted $\mathrm{rk}(a)$, is the least ordinal α such that $a \in \mathrm{V}_\alpha$. We call the *footprint* of $\mathcal{A}$ the ordinal $\mathrm{fp}(\mathcal{A}) \coloneqq \mathrm{rk}_{\mathrm{V}}(\mathcal{A}) + \omega_1^{\mathrm{V}}$ where ω_1^{V} is the least uncountable ordinal in V. We assume that the sets $\Lambda_{A,B}^{\mathrm{open}}$ and $\Pi_{A,B}$ were constructed in such a way that their ranks are strictly less than $\mathrm{fp}(\mathcal{A})$. When there is no ambiguity, we will drop the indices A, B and simply write Λ, Π, $\mathcal{R}$, etc.

3 The theory ZF_ε

In order to define a realizability model for classical set theory, we consider a non-extensional conservative extension of the usual set theory. This theory was originally formulated by Friedman in [4] in his proof that ZF is equiconsistent with IZF and notably contains two distinct membership relations: $\in$ which behaves like the standard membership relation, and ε which is a form of "*strong membership*".

Throughout this paper, we will work in first-order logic *without equality*: individuals languages may contain a symbol that happens to be written "=", but such a symbol has no special status. In particular, models are not required to interpret it by "meta" equality. In addition, we will assume that the only primitive logical constructions are $\to$, $\top$, $\bot$, and $\forall$; for $\vee$, $\wedge$, and $\exists$, we will use De Morgan's encoding. Thus:

- $\varphi \wedge \psi$ means $(\varphi \to \psi \to \bot) \to \bot$,
- $\varphi \vee \psi$ means $(\varphi \to \bot) \to (\psi \to \bot) \to \bot$,
- $\exists x\ \varphi(x)$ means $(\forall x\ (\varphi(x) \to \bot)) \to \bot$.

We will denote by $\mathcal{L}_\in$ the first-order language over the signature $\{\in, \simeq\}$ where $\in$ and $\simeq$ are binary relation symbols. The language of ZF_ε requires two distinct symbols for the membership relation, $\in$ and ε (the former will refer to the usual extensional membership relation, the latter will correspond to a strict non-extensional membership relation); however, for computational reasons it is better to take as primitives the negative versions of those symbols, thus the language of ZF_ε which is denoted $\mathcal{L}_\varepsilon$, is the first-order language over the signature $\{\notin, \subseteq, \not\varepsilon, \neq\}$, where all 4 symbols are binary relation symbols. It can be proven that $\neq$ is definable from $\not\varepsilon$ via the Leibniz equality and is therefore not necessary in the signature, however we include it here for practical purposes. $\mathrm{Fml}_\in$ and Fml_ε denote the collection of all formulas in $\mathcal{L}_\in$ and $\mathcal{L}_\varepsilon$ respectively. In the language $\mathcal{L}_\varepsilon$, we will use the following abbreviations:

Abreviation	Meaning	Abreviation	Meaning
$a \,\varepsilon\, b$	$a \not\varepsilon b \to \bot$	$a \simeq b$	$(a \subseteq b) \wedge (b \subseteq a)$
$a \in b$	$a \notin b \to \bot$	$\forall x \,\varepsilon\, a\ \varphi(x)$	$\forall x\ (x \,\varepsilon\, a \to \varphi(x))$
$a = b$	$a \neq b \to \bot$	$\exists x \,\varepsilon\, a\ \varphi(x)$	$(\forall x\ (\varphi(x) \to x \notin a)) \to \bot$

In particular, by a slight abuse of notation, we will consider $\mathrm{Fml}_\in$ to be a subset of Fml_ε. ZF denotes the usual set theory, written in the language $\mathcal{L}_\in$, i.e. ZF is a subset of $\mathrm{Fml}_\in$, while ZF_ε denotes non-extensional set theory, as defined by Krivine, written in the language $\mathcal{L}_\varepsilon$ (i.e. ZF_ε is a subset of Fml_ε). In a nutshell, the axioms of ZF_ε state that:

- An equivalent presentation of the axioms of ZF *minus the Axiom of Extensionality* (essentially the double negation) are satisfied *over the signature* $\{\notin, \neq\}$ (rather than $\{\in, \simeq\}$).
- $\in$ is the extensional collapse of ε: $x \in y$ *iff* there is $x' \,\varepsilon\, y$ such that $x \simeq x'$;
- $\subseteq$ is the extensional inclusion: $x \subseteq y$ *iff* for every $z \,\varepsilon\, x$, we have $z \in y$;
- $\simeq$ is extensional equivalence: two sets are $\simeq$-equal *iff* they have the same $\in$-elements.

For full details, including the list of the axioms of $\mathcal{L}_\varepsilon$, we refer the reader to [14]; see also Friedman's earlier account in [4]. As proven in [4], ZF_ε is a conservative extension of ZF:

▶ **Theorem 4.** *Let φ be a closed formula in $\mathcal{L}_\in$: φ is a consequence of* ZF *if and only if it is a consequence of* ZF_ε.

A proof of this fact can be found in [12]. For further details we refer to [19].

Whenever $\mathcal{L}$ is a first-order language that contains $\mathcal{L}_\varepsilon$, we will denote by $\mathrm{ZF}_\varepsilon^\mathcal{L}$ the theory obtained by taking ZF_ε and enriching all the axiom schemas to include the formulas of $\mathcal{L}$.

4 Construction of realizability models

Our construction of realizability models follows the presentation in [19]. Let V be a model of Zermelo-Frænkel set theory, ZF, and let $\mathcal{A} = (\Lambda, \Pi, \perp\!\!\!\perp)$ be a realizability algebra in V.

We define $\mathrm{N}^{\mathcal{A},\mathrm{V}} \subseteq \mathrm{V}$ as follows: for any ordinal $\alpha \in \mathrm{V}$, let $\mathrm{N}_\alpha^{\mathcal{A},\mathrm{V}} \coloneqq \bigcup_{\beta<\alpha} \mathcal{P}(\mathrm{N}_\beta^{\mathcal{A},\mathrm{V}} \times \Pi)$, then let $\mathrm{N}^{\mathcal{A},\mathrm{V}} \coloneqq \bigcup_{\alpha\in\mathrm{Ord}} \mathrm{N}_\alpha^{\mathcal{A},\mathrm{V}}$, where Ord denotes the class of ordinals in V. The elements of $\mathrm{N}^{\mathcal{A},\mathrm{V}}$ are called $(\mathcal{A},\mathrm{V})$-*names*. Note that for all α, $\mathrm{N}_\alpha^{\mathcal{A},\mathrm{V}} \in \mathrm{V}$, but $\mathrm{N}^{\mathcal{A},\mathrm{V}} \notin \mathrm{V}$ ($\mathrm{N}^{\mathcal{A},\mathrm{V}}$ is a *proper class* in V). We will generally drop the exponents and simply write N_α and N. Given an element $a \in \mathrm{N}$ we let $\mathrm{dom}(a) \coloneqq \{b \mid \exists \pi \in \Pi\ (b,\pi) \in a\} \in \mathrm{V}$.

A function $f : \mathrm{N}^n \to \mathrm{N}$ is said to be $\mathcal{A}$-*definable* if there is a formula $\varphi(z, x_1, \dots, x_n, y) \in \mathrm{Fml}_\varepsilon$ and $c \in \mathrm{V}_{\mathrm{fp}(\mathcal{A})}$ such that, for any $a_1, \dots, a_n \in \mathrm{N}$ and $b \in \mathrm{V}$, $\mathrm{V} \models \varphi(c, a_1, \dots, a_n, b)$ if and only if $b = f(a_1, \dots, a_n)$. Let $\mathcal{L}_\varepsilon^{\mathcal{A}}$ be the language obtained from $\mathcal{L}_\varepsilon$ by adding for each $\mathcal{A}$-definable function $f : \mathrm{N}^n \to \mathrm{N}$ an n-ary function symbol "f".

The *realizability interpretation* of $\mathcal{L}_\varepsilon^{\mathcal{A}}$ in $\mathcal{A}$ consists of the following.

▶ **Definition 5.** *To each closed formula φ in $\mathcal{L}_\varepsilon^{\mathcal{A}}$ with parameters in* N*, we associate a* truth value $|\varphi| \subseteq \Lambda$ *and a* falsity value $\|\varphi\| \subseteq \Pi$*, they are defined jointly by induction on the complexity of φ:*

- $|\varphi| \coloneqq \{t \in \Lambda \mid \forall \pi \in \|\varphi\|, t \star \pi \in \perp\!\!\!\perp\}$*;*
- $\|\top\| \coloneqq \emptyset$ *and* $\|\bot\| \coloneqq \Pi$*;*
- $\|a \notin b\| \coloneqq \{\pi \in \Pi \mid (a,\pi) \in b\}$;
- $\|a \neq b\| \coloneqq \|\top\|$ *if* $a \neq b$*,* $\|\bot\|$ *otherwise;*
- $\|a \notin b\| \coloneqq \bigcup_{c\in\mathrm{dom}(b)} \{t.t'.\pi \mid (c,\pi) \in b, t \in |a \subseteq c|, t' \in |c \subseteq a|\}$*;*
- $\|a \subseteq b\| \coloneqq \bigcup_{c\in\mathrm{dom}(a)} \{t.\pi \mid (c,\pi) \in a, t \in |c \notin b|\}$*;*
- $\|\psi \to \theta\| \coloneqq \{t.\pi \mid t \in |\psi|, \pi \in \|\theta\|\}$*;*
- $\|\forall x\ \varphi(x)\| \coloneqq \bigcup_{a\in\mathrm{N}} \|\varphi[a/x]\|$.

For atomic formulas, we identify the closed terms a and b with their valuations in N*. Formally, $\|a \notin b\|$ and $\|a \subseteq b\|$ are defined by induction on the pair* $(\max(\mathrm{rk}_\mathrm{N}(a), \mathrm{rk}_\mathrm{N}(b)), \min(\mathrm{rk}_\mathrm{N}(a), \mathrm{rk}_\mathrm{N}(b)))$ *under the product order, where* $\mathrm{rk}_\mathrm{N}(c) \coloneqq \min\{\alpha \mid c \in \mathrm{N}_{\alpha+1}\}$.

We say that a closed λ_c-term t realizes *a closed formula φ with parameters in* N *and write $t \Vdash \varphi$, whenever $t \in |\varphi|$.*

By standard set-theoretic arguments, for each formula $\varphi(\vec{x})$ in $\mathcal{L}_\varepsilon^{\mathcal{A}}$ (without parameters), there exist formulas $\varphi_\Pi(p, \vec{x})$ and $\varphi_\Lambda(p, \vec{x})$ in $\mathcal{L}_\in$ with parameters in $\mathrm{V}_{\mathrm{fp}(\mathcal{A})}$ such that for all sequences of sets $\vec{a} \in \mathrm{V}$, for all $\pi \in \Pi$ and $t \in \Lambda$,

$$\pi \in \|\varphi(\vec{a})\| \Leftrightarrow \mathrm{V} \models \varphi_\Pi(\pi, \vec{a}), \text{ and } t \in |\varphi(\vec{a})| \Leftrightarrow \mathrm{V} \models \varphi_\Lambda(t, \vec{a}).$$

Now, we would like to associate to $\mathcal{A}$ a "realizability theory" consisting of all closed formulas which are realized. However, for all $t \star \pi \in \perp\!\!\!\perp$, the λ_c-term $\mathrm{k}_\pi\, t$ realizes the formula $\bot$. Therefore, in order to obtain a realizability theory that is not automatically inconsistent, we will need to exclude terms of this shape; this is where the set $\mathcal{R}$ of *realizers* comes into play (i.e. the closed λ_c terms containing no continuation constant):

▶ **Definition 6.** *The* realizability theory of $(\mathcal{A},\mathrm{V})$*, denoted by $T_{\mathcal{A},\mathrm{V}}$, is the set of all closed formulas, φ, of $\mathcal{L}_\varepsilon^{\mathcal{A}}$ with parameters in* N *such that there exists $t \in \mathcal{R}$ such that t realizes φ.*

The following facts are standard (see e.g. [14]):

- the realizability theory of $(\mathcal{A},\mathrm{V})$ is closed under classical deduction, (i.e. if $\varphi \in T_{\mathcal{A},\mathrm{V}}$ and φ entails ψ in classical logic, then $\psi \in T_{\mathcal{A},\mathrm{V}}$);
- this theory is consistent if and only if for every $t \in \mathcal{R}$ there is a stack π such that $t \star \pi \notin \perp\!\!\!\perp$;
- this theory is generally not complete.

▶ **Theorem 7.** *Let* V *be a model of* ZF *and* $\mathcal{A}$ *a realizability algebra in* V*. The realizability theory of* $(\mathcal{A}, \mathrm{V})$ *contains* $\mathrm{ZF}_\varepsilon^{\mathcal{L}_\varepsilon^{\mathcal{A}}}$*. In particular, it contains* ZF_ε*, and therefore* ZF*).*

We refer to [14] for a proof of this, or [19] for an alternative proof using the setup given. This justifies the following definition:

▶ **Definition 8.** *A* realizability model of ZF_ε *is a pair* $\mathcal{N} = (\mathrm{V}, \mathcal{A})$*, with* V *a model of* ZF *and* $\mathcal{A}$ *a realizability algebra in* V*. We write* $\mathcal{N} \Vdash \varphi$ *for "*$T_{\mathcal{A},\mathrm{V}}$ *contains* φ*".*

Sometimes, we will argue within models of the realizability theory $T_{\mathcal{A},\mathrm{V}}$ and by abuse of language we will call *realizability model* any model of $T_{\mathcal{A},\mathrm{V}}$.

5 Reish Names and Pairing

In this section we present the method of *reish names*, or *recursive* names, which is used for transferring properties of sets of the ground model to sets in a realizability model. Given a ground model set a, the *gimel of* a, $\gimel(a)$, is defined as $\gimel(a) \coloneqq a \times \Pi$, but $\reish(a)$ shall instead apply this process recursively to all elements of a. This will have the benefit that $\reish(a)$, and each one of its elements, is always an element of N.

▶ **Definition 9.** *For* $x \in \mathrm{V}$ *we define* $\reish(x) \coloneqq \{(\reish(y), \pi) \mid y \in x,\ \pi \in \Pi\}$.

This method will not in general give a straightforward interpretation of the ground model elements. For example, it is unclear if $\reish(\omega)$ is an extensional name for the first limit ordinal in the ZF structure. However, it is a useful tool to transfer certain properties of sets of the ground model into the realizability model.

▶ **Proposition 10.** *If* $a \subseteq b$ *then* $\mathcal{N} \Vdash \reish(a) \subseteq \reish(b)$.

Proof. Let v_0 be a realizer such that $v_0 \Vdash \forall x(x \subseteq x)$ and set $v_1 \coloneqq \lambda f.(f(v_0))(v_0)$. It is easy to see that $v_1 \Vdash \reish(a) \subseteq \reish(b)$. ◀

▶ **Observation 11.** *If* $a \in b$ *then* $\mathbf{I} \Vdash \reish(a)\, \varepsilon\, \reish(b)$*, where* $\mathbf{I} = \lambda f.f$ *is the identity term. Thus, if* $a \in b$ *then* $\mathcal{N} \Vdash \reish(a) \not\subseteq \reish(b)$.

This construction then allows us to define a proper class of ordinals in a realizability model. For this we use the definition of ordinals as *transitive sets of transitive sets*, which can easily be seen to be equivalent to *a transitive set well-ordered by the* $\in$-relation.

▶ **Definition 12.** *(*ZF_ε*) We say that a set* a *is a* ε-ordinal *if it is a* ε*-transitive set of* ε*-transitive sets. That is,* $\forall x\, \varepsilon\, a\, \forall y\, \varepsilon\, x\, (y\, \varepsilon\, a)$ *and* $\forall z\, \varepsilon\, a\, \forall x\, \varepsilon\, z\, \forall y\, \varepsilon\, x\, (y\, \varepsilon\, z)$.

Note that, over ZF_ε, this definition is not equivalent to the definition of ordinals as transitive sets well-ordered by the ε-relation. As an example, consider the realizability model constructed at the end of [15] in which $\reish(2)$ has size 4. In this case there are two ordinals, $a, b\, \varepsilon\, \reish(2)$, such that $a \notin b$, $b \notin a$. However, $(a \cup \{a\}) \cup (b \cup \{b\})$ is an ε-ordinal on which the ε-relation does not linearly order the set.

▶ **Proposition 13.** *Suppose that* $\mathcal{N} = (\mathrm{N}, \notin, \not\subseteq, \subseteq)$ *is a model of* ZF_ε*. Then for any* $a \in \mathrm{N}$*:*

1. *If* a *is a* ε*-transitive set, then it is a* $\in$*-transitive set,*
2. *If* a *is a* ε*-ordinal, then it is a* $\in$*-ordinal.*

Proof. Suppose that a is a ε-transitive set and take $c \in b \in a$. Then there exists some $x \varepsilon a$ such that $x \simeq b$ and there exists some $y \varepsilon x$ such that $y \simeq c$. Since a is assumed to be ε-transitive, $y \varepsilon a$. Therefore $x \in a$ by definition of $\in$.

Next, suppose that a is a ε-ordinal. We have already shown that a is $\in$-transitive so it suffices to prove that every $b \in a$ is $\in$-transitive. So let $d \in c \in b \in a$. Then we can find $z \varepsilon y \varepsilon x \varepsilon a$ such that $x \simeq b$, $y \simeq c$ and $z \simeq d$. Since a is a ε-ordinal, $z \varepsilon x$ and therefore $d \in x$. Finally, $d \in x$ and $x \simeq b$ gives us $d \in b$, as required. ◀

▶ **Proposition 14.** *If δ is an ordinal in* V *then $\mathcal{N} \Vdash \ulcorner(\delta)$ is a ε-ordinal.*

Proof. Let δ be an ordinal in V. We show that $\ulcorner(\delta)$ is a ε-transitive set; the fact that it consists of ε-transitive sets will follow by a similar argument. To do this, we show that $\mathbf{I} \Vdash \forall x^{\ulcorner(\delta)} \forall y (y \notin \ulcorner(\delta) \to y \notin x)$. Fix $\beta \in \delta$, $c \in \mathrm{N}$, $t \Vdash c \notin \ulcorner(\delta)$ and $\pi \in \|c \notin \ulcorner(\beta)\|$. Now $\|c \notin \ulcorner(\beta)\| = \{\sigma \mid (c, \sigma) \in \ulcorner(\beta)\}$. Since this set is non-empty, it must be the case that $\|c \notin \ulcorner(\beta)\| = \Pi$ and $c = \ulcorner(\gamma)$ for some $\gamma \in \beta$. Therefore, $\|c \notin \ulcorner(\delta)\| = \|\ulcorner(\gamma) \notin \ulcorner(\delta)\| = \Pi$ hence $t \star \pi \in \perp\!\!\!\perp$, from which the result follows. ◀

We need a method to encode ordered pairs in the realizability structure; for this we introduce a function $\mathtt{op}$ satisfying $\mathcal{N} \Vdash$ "$\mathtt{op}(a, b)$ is the ordered pair of a and b" for any $a, b \in \mathrm{N}$. This definition is based on the *Wiener pairing function* which is $(a, b) = \{\{\{a\}, \emptyset\}, \{\{b\}\}\}$. Here 0 denotes the λ-term $\lambda x.\lambda y.y$ and 1 the λ-term $\lambda x.\lambda y.xy$.

▶ **Definition 15.** *For $a, b \in \mathrm{N}$, we define*

- *the* singleton *of a as the set* $\mathtt{sng}(a) \coloneqq \{a\} \times \Pi$,
- *the* unordered pair *of a and b as the set* $\mathtt{up}(a, b) \coloneqq \{(a, \underline{0}.\pi) \mid \pi \in \Pi\} \cup \{(b, \underline{1}.\pi) \mid \pi \in \Pi\}$,
- *the* ordered pair *of a and b as the set* $\mathtt{op}(a, b) \coloneqq \mathtt{up}(\mathtt{up}(\mathtt{sng}(a), \ulcorner(0)), \mathtt{sng}(\mathtt{sng}(b)))$.

Note that the three functions $\mathtt{sng} : \mathrm{N} \to \mathrm{N}$, $\mathtt{up}, \mathtt{op} : \mathrm{N}^2 \to \mathrm{N}$ *are $\mathcal{A}$-definable.*

▶ **Theorem 16** ([19])**.** *The following are realizable in $\mathcal{N}$:*

- $\forall x_1 \forall x_2 \forall y_1 \forall y_2 \, (\mathtt{op}(x_1, y_1) \simeq \mathtt{op}(x_2, y_2) \to (x_1 \simeq x_2 \wedge y_1 \simeq y_2))$.
- $\forall x_1 \forall x_2 \forall y_1 \forall y_2 \, (x_1 \simeq x_2 \to (y_1 \simeq y_2 \to \mathtt{op}(x_1, y_1) \simeq \mathtt{op}(x_2, y_2)))$.

While we do not include a proof of this fact, we refer to [19] for all necessary details.

6 Relativization over transitive sets

In this setion we introduce a method to relativize formulas to certain objects in a realizability model. Relativization is a simple, but power, technique which provides a way to interpret a formula internally in a given transitive set or class. Given a formula φ and set M, the relativised formula φ^M is essentially constructed by replacing all unbounded quantifiers with quantifiers bounded by M, that is $\forall x$ becomes $\forall x \in M$.

Let $\mathcal{A}$ be a realizability algebra and construct the class of names, N. Given a transitive set M containing $\mathcal{A}$, we set $\mathrm{M}^{\mathcal{A}} \coloneqq \{(a, \pi) \mid a \in \mathrm{M} \cap \mathrm{N}, \pi \in \Pi\}$. Namely, $\mathrm{M}^{\mathcal{A}}$ is a name for the set of names that are in M and $\mathrm{dom}(\mathrm{M}^{\mathcal{A}}) = \mathrm{M} \cap \mathrm{N}$. We can then relativize to M any formula φ in the language of $\mathcal{L}_{\varepsilon}^{\mathcal{A}}$, which we denote by φ^{M}, by replacing universal quantifiers $\forall x$ by bounded quantifiers $\forall x^{\mathrm{M}^{\mathcal{A}}}$ defined in the usual way.

▶ **Definition 17.** *Suppose that* M *is a transitive set containing $\mathcal{A}$ and φ is a formula in* $\mathrm{Fml}_{\varepsilon}$*. We define* $\|\forall x^{\mathrm{M}^{\mathcal{A}}} \varphi(x)\| = \bigcup_{c \in \mathrm{dom}(\mathrm{M}^{\mathcal{A}})} \|\varphi(c)\|$.

It is easy to see that this restricted quantifier $\forall x^{\mathrm{M}^{\mathcal{A}}}$ corresponds to $\forall x \, \varepsilon \, \mathrm{M}^{\mathcal{A}}$.

▶ **Proposition 18.** *Suppose that* M *is a transitive set which contains* $\mathcal{A}$*. Then*
1. $\lambda f.\lambda g.gf \Vdash \forall x^{\mathrm{M}^{\mathcal{A}}}\, \varphi(x) \to \forall x(\neg\varphi(x) \to x \notin \mathrm{M}^{\mathcal{A}})$,
2. $\lambda f.\mathrm{cc}(\lambda k.fk) \Vdash \forall x(\neg\varphi(x) \to x \notin \mathrm{M}^{\mathcal{A}}) \to \forall x^{\mathrm{M}^{\mathcal{A}}}\, \varphi(x)$.

Proof. First, suppose that $t \Vdash \forall x^{\mathrm{M}^{\mathcal{A}}}\, \varphi(x)$, $s \Vdash \neg\varphi(b)$ for some $b \in \mathrm{N}$ and $\pi \in \|b \notin \mathrm{M}^{\mathcal{A}}\|$. Since $(\pi, b) \in \mathrm{M}^{\mathcal{A}}$ and M is transitive, we must have $b \in \mathrm{M} \cap \mathrm{N}$ and therefore $t \star \sigma \in \perp\!\!\!\perp$ for any $\sigma \in \|\varphi(b)\|$. It follows that $t \Vdash \varphi(b)$, hence $\lambda f.\lambda g.gf \star t.s.\pi \succ s \star t.\pi \in \perp\!\!\!\perp$, as required.

For the second claim, suppose that $t \Vdash \forall x(\neg\varphi(x) \to x \notin a)$ and $\pi \in \|\forall x^{\mathrm{M}^{\mathcal{A}}}\, \varphi(x)\|$. Fix $b \in \mathrm{M} \cap \mathrm{N}$ such that $\pi \in \|\varphi(b)\|$. We have

$$\|\forall x(\neg\varphi(x) \to x \notin \mathrm{M}^{\mathcal{A}})\| = \bigcup_{c\in\mathrm{N}} \|\neg\varphi(c) \to c \notin \mathrm{M}^{\mathcal{A}}\| = \bigcup_{c\in\mathrm{N}} \{s.\sigma \mid s \Vdash \neg\varphi(c), \sigma \in \|c \notin \mathrm{M}^{\mathcal{A}}\|\}.$$

Since $\pi \in \|\varphi(b)\|$, $\mathrm{k}_\pi \Vdash \neg\varphi(b)$. Moreover, $(b, \pi) \in \mathrm{M}^{\mathcal{A}}$ thus $\mathrm{k}_\pi.\pi \in \|\forall x(\neg\varphi(x) \to x \notin \mathrm{M}^{\mathcal{A}})\|$. It follows that $\lambda f.\mathrm{cc}(\lambda k.fk) \star t.\pi \succ \mathrm{cc} \star (\lambda k.tk).\pi \succ \lambda k.tk \star \mathrm{k}_\pi.\pi \succ t \star \mathrm{k}_\pi.\pi \in \perp\!\!\!\perp$. ◀

One can easily observe that if M is a transitive set containing $\mathcal{A}$, then $\mathrm{M}^{\mathcal{A}}$ is realized to be a ε-transitive set, and thus also a transitive set by Proposition 13.

▶ **Proposition 19.** *For every transitive set* M *which contains* $\mathcal{A}$*,* $\mathbf{I} \Vdash \forall x^{\mathrm{M}^{\mathcal{A}}} \forall y(y \notin \mathrm{M}^{\mathcal{A}} \to y \notin x)$*. Thus* $\mathcal{N}$ *realizes that* $M^{\mathcal{A}}$ *is a* ε*-transitive set.*

▶ **Theorem 20.** *Let* M *be a transitive class which contains* $\mathcal{A}$*, then for all sets* $a_1, \dots, a_n$ *in* $\mathrm{M} \cap \mathrm{N}$*, and for every formula* $\varphi \in \mathrm{Fml}_\varepsilon$*,* $\|\varphi^{\mathrm{M}^{\mathcal{A}}}(a_1, \dots, a_n)\| = \|\varphi(a_1, \dots, a_n)\|^{\mathrm{M}}$.

Proof. We procede by induction on the formula, ignoring the parameters to simplify notation.

Let $\varphi(x, y) \equiv x \notin y$ and fix $a, b \in \mathrm{M} \cap \mathrm{N}$. Then we have $\|(a \notin b)^{\mathrm{M}^{\mathcal{A}}}\| = \|a \notin b\|^{\mathrm{V}} = \{\pi \in \Pi \mid (a, \pi) \in b\} = \|a \notin b\|^{\mathrm{M}}$, since M is a transitive class containing Π.

We will prove the cases $\varphi(x, y) \equiv x \notin y$ and $\varphi(x, y) \equiv x \subseteq y$ by simultaneous induction on the lexicographical order of the pair of ranks of a and b. So, fix $a, b \in \mathrm{M} \cap \mathrm{N}$. Then we have $\|(a \notin b)^{\mathrm{M}^{\mathcal{A}}}\| = \|a \notin b\|^{\mathrm{V}} = \bigcup_{c\in\mathrm{dom}(b)} \{t.t'.\pi \mid (c, \pi) \in b, t \Vdash a \subseteq c, t' \Vdash c \subseteq a\}$. Now, $t \Vdash c \subseteq a$ means $\forall \sigma \in \|c \subseteq a\| (t \star \sigma \in \perp\!\!\!\perp)$, by the induction hypothesis this is equivalent to $\forall \sigma \in \|c \subseteq a\|^{\mathrm{M}} (t \star \sigma \in \perp\!\!\!\perp)$ which corresponds to $(t \Vdash c \subseteq a)^{\mathrm{M}}$. Thus we have $\|(a \notin b)^{\mathrm{M}^{\mathcal{A}}}\| = \bigcup_{c\in\mathrm{dom}(b)} \{t.t'.\pi \mid (c, \pi) \in b, (t \Vdash a \subseteq c)^{\mathrm{M}}, (t' \Vdash c \subseteq a)^{\mathrm{M}}\} = \|a \notin b\|^{\mathrm{M}}$.

Similarly for the second case, by applying the induction hypothesis we have $\|(a \subseteq b)^{\mathrm{M}^{\mathcal{A}}}\| = \bigcup_{c\in\mathrm{dom}(a)} \{t.\pi \mid (c, \pi) \in a, (t \Vdash c \notin b)^{\mathrm{M}}\} = \|a \subseteq b\|^{\mathrm{M}}$.

The cases $\varphi \equiv \psi \to \chi$ and $\varphi \equiv \forall x\, \psi(x)$ follow easily from the induction hypothesis. ◀

We will apply these results in particular to the transitive sets of the Von Neumann hierarchy. We end this section by showing that if $\mathcal{A}$ is an element of V_γ then $\mathrm{V}_\gamma^{\mathcal{A}}$ is simply the construction of the names internally in V_γ.

▶ **Lemma 21.** *Let* γ *be a limit ordinal such that* $\mathcal{A} \in \mathrm{V}_\gamma$*. Then* $\mathrm{V}_\gamma^{\mathcal{A}} = \bigcup \mathrm{N}_\gamma$ *and* $\mathrm{N}_\gamma = (\mathrm{N})^{\mathrm{V}_\gamma}$.

Proof. First, observe that for every $a \in \mathrm{N}$, $\mathrm{rk}_{\mathrm{N}}(a) \leq \mathrm{rk}_{\mathrm{V}}(a) \leq \max\{\mathrm{rk}_{\mathrm{N}}(a), \mathrm{rk}_{\mathrm{V}}(\Pi)\} + 2$ where $\mathrm{rk}_{\mathrm{N}}(a)$ is the minimal α for which $a \in \mathrm{N}_\alpha$ and $\mathrm{rk}_{\mathrm{V}}(a)$, the minimal α for which $a \in \mathrm{V}_\alpha$, is the standard rank of a in V. From this it follows that $\mathrm{V}_\gamma \cap \mathrm{N} = \mathrm{N}_\gamma$ since γ is a limit ordinal and $\mathcal{A} \in \mathrm{V}_\gamma$, therefore $\mathrm{V}_\gamma^{\mathcal{A}} = \{(a, \pi) \mid a \in \mathrm{N}_\gamma, \pi \in \Pi\} = \bigcup \mathrm{N}_\gamma$.

We now prove inductively that $\mathrm{N}_\alpha = (\mathrm{N}_\alpha)^{\mathrm{V}_\gamma}$ for all $\alpha \leq \gamma$, starting with $\mathrm{N}_0 = \emptyset = (\mathrm{N}_0)^{\mathrm{V}_\gamma}$. So fix $\alpha \leq \gamma$ and suppose that the claim holds for all $\beta < \alpha$. Then

$$\mathrm{N}_\alpha = \bigcup_{\beta<\alpha} \mathcal{P}(\mathrm{N}_\beta \times \Pi) = \bigcup_{\beta<\alpha} \mathcal{P}^{\mathrm{V}_\gamma}((\mathrm{N}_\beta)^{\mathrm{V}_\gamma} \times \Pi) = (\mathrm{N}_\alpha)^{\mathrm{V}_\gamma}.$$ ◀

7 Realizing Inaccessibles

In this section, we assume the consistency of the theory "ZFC plus there is an inaccessible cardinal" and, from that, we show how to build a *realizability* model for the equiconsistent theory "ZF plus there is an inaccessible set".

We recall that over ZFC an uncountable cardinal κ is (strongly) inaccessible if it is a regular cardinal which is a strong limit, namely whenever $\alpha < \kappa$, $2^\alpha < \kappa$. However, in models where the Axiom of Choice fails this is no longer a satisfactory definition, for example if 2^ω is not well-ordered then no such (well-orderable) cardinals can possibly exist. Therefore, it is preferable to take an alternative definition.

From a structural point of view, the defining property of an inaccessible cardinal is that it provides a very robust model of set theory. Namely, if κ is inaccessible then V_κ is a Grothendieck Universe which contains ω. For our purposes we will take a slightly different, but equivalent, definition which is that a set will be *inaccessible* if it is a transitive model of full second-order ZF; this definition is motivated by [5, Definition 1].

▶ **Definition 22.** *We call a set z* inaccessible *if it satisfies the following:*

- *Transitivity:* $\forall u \in z \forall v \in u (v \in z)$.
- *Empty Set:* $\exists u \in z \forall v (v \notin u)$.
- *Pairing:* $\forall u \in z \forall v \in z \exists w \in z \, (u \in w \wedge v \in w)$.
- *Unions:* $\forall u \in z \exists v \in z \forall w \, (w \in v \leftrightarrow \exists x \in u (w \in x))$.
- *Infinity:* $\forall a \in z \exists u \in z \, (a \in u \wedge \forall v \in u \exists w \in u (v \in w))$.
- *Weak Power Set:* $\forall u \in z \exists v \in z \forall w \exists x \in v \forall y \, (y \in x \leftrightarrow (y \in u \wedge y \in w))$.
- *Second-order Collection:*

$$\forall a \in z \forall f (\forall x \in a \, \exists y \in z \, ((x, y) \in f) \rightarrow \exists b \in z \forall x \in a \, \exists y \in b \, ((x, y) \in f)).$$

The proof of the following proposition is standard and justifies calling such sets inaccessible.

▶ **Proposition 23.** *Over* ZFC *the following are equivalent:*

- *z is inaccessible,*
- *z is a Grothendieck Universe containing ω,*
- *$z = \mathrm{V}_\kappa$ for some inaccessible cardinal κ.*

Moreover, it is known that if V is a model of ZF with an inaccessible set z, then $z \cap \mathrm{L}$ is an inaccessible set in the constructible universe L, which is a model of ZFC. In fact, $z \cap \mathrm{L} = \mathrm{L}_\kappa$ where κ is an inaccessible cardinal in L. Therefore ZF with an inaccessible set is equiconsistent with ZFC plus an inaccessible cardinal.

We now consider any realizability algebra $\mathcal{A}$ in a model V of ZF (the ground model) with an inaccessible set z such that $\mathcal{A} \in z$. We shall give an appropriate translation of inaccessible sets to the language of ZF_ε, which we call ε-inaccessible sets. We shall then show that in any realizability model $\mathcal{N}$, $z^{\mathcal{A}}$ is a ε-inaccessible set and in the corresponding ZF structure $(\mathrm{N}, \in, \simeq)$, $z^{\mathcal{A}}$ is an inaccessible set.

▶ **Definition 24.** *In a model of $T_{\mathcal{A},\mathrm{V}}$, we call a set z* ε-inaccessible *if it satisfies the following:*

- *ε-Transitivity:* $\forall u \, \varepsilon \, z \forall v \, \varepsilon \, u (v \, \varepsilon \, z)$.
- *ε-Empty Set:* $\exists u \, \varepsilon \, z \forall v (v \notin u)$.
- *ε-Pairing:* $\forall u \, \varepsilon \, z \forall v \, \varepsilon \, z \exists w \, \varepsilon \, z \, (u \, \varepsilon \, w \wedge v \, \varepsilon \, w)$.
- *ε-Unions:* $\forall u \, \varepsilon \, z \exists v \, \varepsilon \, z \forall w \, (w \, \varepsilon \, v \leftrightarrow \exists x \, \varepsilon \, u (w \, \varepsilon \, x))$.

- *ε-Infinity:* $\forall a\,\varepsilon\, z\,\exists u\,\varepsilon\, z\,(a\,\varepsilon\, u \wedge \forall v\,\varepsilon\, u \exists w\,\varepsilon\, u(v\,\varepsilon\, w))$.
- *ε-Weak Power Set:* $\forall u\,\varepsilon\, z\,\exists v\,\varepsilon\, z\,\forall w\,\exists x\,\varepsilon\, v\,\forall y\,(y\,\varepsilon\, x \leftrightarrow (y\,\varepsilon\, u \wedge y\,\varepsilon\, w))$.
- *ε-Second-order Collection:*

$$\forall a\,\varepsilon\, z \forall f(\forall x\,\varepsilon\, a\,\exists y\,\varepsilon\, z\,(\mathtt{op}(x,y)\,\varepsilon\, f) \to \exists b\,\varepsilon\, z\,\forall x\,\varepsilon\, a\,\exists y\,\varepsilon\, b\,(\mathtt{op}(x,y)\,\varepsilon\, f)).$$

▶ **Lemma 25.** *If z is an inaccessible set in* V *and $\mathcal{A} \in z$ is a realizability algebra, then for the corresponding realizability model $\mathcal{N} = (\mathrm{V}, \mathcal{A})$ we have $\mathcal{N} \Vdash z^{\mathcal{A}}$ is a ε-inaccessible set.*

Proof. Firstly, by Proposition 19 we have that $z^{\mathcal{A}}$ is a ε-transitive set. For all of the axioms except for Second-order Collection it suffices to verify that $\mathrm{dom}(z^{\mathcal{A}})$ is closed by certain relevant names. For this, we observe that since z is a transitive set which is closed under Weak Power Set and Second-order Collection, we have that $z = \mathrm{V}_{\mathrm{rank}(z)}$. Therefore, z is also closed under Separation and, by Lemma 21, $z^{\mathcal{A}} = (\mathrm{N})^z$.

For Empty Set the relevant name is $\emptyset$ which is in $z \cap \mathrm{N}$ be definition. Given $a, b \in z \cap \mathrm{N}$ the name for the pair is $\{a, b\} \times \Pi \in z \cap \mathrm{N}$. Given $a \in z \cap \mathrm{N}$ the name for the union is $\{(c, \sigma) \mid \exists (x, \pi) \in a\ (c, \sigma) \in x\} \in z \cap \mathrm{N}$. Given $a \in z \cap \mathrm{N}$ the name for the infinite set containing a is $\{(a^n, \pi) \mid n \in \omega, \pi \in \Pi\}$ where $a^0 \coloneqq a$ and $a^{n+1} \coloneqq \{a^n\} \times \Pi$. It is clear that all of these names are in $z \cap \mathrm{N}$. Finally, given $a \in z \cap \mathrm{N}$ the name for the Weak Power Set of a is $\mathcal{P}(\mathrm{dom}(a) \times \Pi) \times \Pi$.

It remains to prove the axiom of Second-order Collection. Fix $a \in z \cap \mathrm{N}$ and $f \in \mathrm{N}$. Since z is an inaccessible set, by Second-order Collection in z we can find a set $Y \in z$ such that

$$\forall (x, \pi) \in a \forall t \in \Lambda \exists y \in z \cap \mathrm{N}(t \Vdash \mathtt{op}(x, y)\,\varepsilon\, f) \to \forall (x, \pi) \in a \forall t \in \Lambda \exists y \in Y (t \Vdash \mathtt{op}(x, y)\,\varepsilon\, f).$$

Let $b \coloneqq \{(y, \pi) \mid \exists t \in \Lambda\ \exists x\ ((x, \pi) \in a,\ t \Vdash \mathtt{op}(x, y)\,\varepsilon\, f,\ y \in Y)\} \in z \cap \mathrm{N}$. It will suffice to prove that for any $x \in \mathrm{N}$, $\|\forall y(\mathtt{op}(x, y)\,\varepsilon\, f \to x \notin a)\| \subseteq \|\forall y(\mathtt{op}(x, y)\,\varepsilon\, f \to x \notin b)\|$. For this, fix $t.\pi \in \|\forall y(\mathtt{op}(x, y)\,\varepsilon\, f \to x \notin a)\|$. Then we can fix some $c \in \mathrm{N}$ such that $t \Vdash \mathtt{op}(x, c)\,\varepsilon\, f$ and $(x, \pi) \in a$. By the definition of Y, this means that there exists a $c' \in Y$ such that $t \Vdash \mathtt{op}(x, c')\,\varepsilon\, f$ and $(x, \pi) \in a$ from which it follows that $(c', \pi) \in b$. Thus $t.\pi \in \|\forall y(\mathtt{op}(x, y)\,\varepsilon\, f \to x \notin b)\|$. ◀

▶ **Theorem 26.** *Let $\mathcal{N} = (\mathrm{V}, \mathcal{A})$ be a realizability model, then*
$\mathcal{N} \Vdash \forall z(z$ is an ε-inaccessible set $\to z$ is an inaccessible set$)$

Proof. We argue within a realizability model $\mathcal{N} = (\mathrm{N}, \notin, \notin, \subseteq)$. Suppose that z is a ε-inaccessible set. We want to show that z is an inaccessible set in $(\mathrm{N}, \in, \simeq)$. It is easy to see that z satisfies every condition except for possibly Second-order Collection. In order to prove this axiom, note that Second-order Collection is equivalent to the statement

$$\forall a \in z\,\forall f\,\exists b \in z\,\forall x \in a\,(\exists y \in z\,((x, y) \in f) \to \exists y \in b\,((x, y) \in f)).$$

So fix $a\,\varepsilon\, z$ and f, we define $f' = \{\mathtt{op}(x, y) \mid x\,\varepsilon\, a, y\,\varepsilon\, z, \mathtt{op}(x, y) \in f\}$. Since z satisfies ε-Second-order Collection, we can find some $b\,\varepsilon\, z$ such that $\forall x\,\varepsilon\, a(\exists y\,\varepsilon\, z\,\mathtt{op}(x, y)\,\varepsilon\, f' \to \exists y\,\varepsilon\, b\,\mathtt{op}(x, y)\,\varepsilon\, f')$. We shall show that this same set b witnesses Second-order Collection for f in ZF. By Theorem 16, we know that for every x, y in the realizability model, $\mathtt{op}(x, y) \simeq (x, y)$. Fix $x \in a$ and suppose that $\exists y \in z((x, y) \in f)$. Then, we can find $x'\,\varepsilon\, a$ and $y'\,\varepsilon\, z$ such that $x \simeq x'$ and $y \simeq y'$. Therefore $(x, y) \simeq \mathtt{op}(x', y')$, hence $\mathtt{op}(x', y')\,\varepsilon\, f'$. By definition of b, we can find some $y''\,\varepsilon\, b$ such that $\mathtt{op}(x', y'')\,\varepsilon\, f'$, thus $(x', y'') \in f$ by definition of f'. Since $x \simeq x'$, we have $(x, y'') \simeq (x', y'')$, thus $(x, y'') \in f$ as required. ◀

▶ **Corollary 27.** *Let $\mathcal{N} = (\mathrm{V}, \mathcal{A})$ be a realizability model. Assume that there is an inaccessible set z in* V *such that $\mathcal{A} \in z$. Then $\mathcal{N} \Vdash$ ZF + there exists an inaccessible set.*

▶ Remark 28. One should observe that the statement $z^{\mathcal{A}}$ is a ε-inaccessible set can be expressed by a single sentence. Therefore, given a realizability algebra $\mathcal{A}$, there exists a single realizer θ such that whenever z is an inaccessible set with $\mathcal{A} \in z$, $\theta \Vdash$ "$z^{\mathcal{A}}$ is a ε-inaccessible set".

8 Realizing Mahlo cardinals

In this section, we show that from the consistency of the theory "ZFC plus there is a Mahlo cardinal" we can build a *realizability* model for the equiconsistent theory "ZF plus there is a Mahlo set". Recall that κ is a Mahlo cardinal if $\{\alpha \in \kappa \mid \alpha \text{ is strongly inaccessible}\}$ is stationary in κ. However, as in the inaccessible case, it is beneficial to use the following, more structural, definition which was first formulated by Lévy in [18] and which is the version used by Friedman and Ščedrov [5, Definition 2].

▶ **Definition 29.** *A* Mahlo *set is an inaccessible set z such that for every $u \in z$ and for every binary relation R, there is an inaccessible set $v \in z$ such that*

1. $u \in v$,
2. R reflects to v, *which means that* $\forall x \in v(\exists y \in z\,(x,y) \in R \to \exists y \in v\,(x,y) \in R)$.

The proof of the following proposition is standard and justifies calling such sets Mahlo.

▶ **Proposition 30** (Lévy, [18, Theorem 3])**.** *Over* ZFC*, z is a Mahlo set iff $z = \mathrm{V}_\kappa$ for some Mahlo cardinal κ.*

Moreover, as with the inaccessible case, if V is a model of ZF with a Mahlo set z, then $z \cap \mathrm{L}$ remains a Mahlo set in L. Therefore ZF with a Mahlo set is equiconsistent with ZFC plus a Mahlo cardinal.

▶ **Definition 31.** *In a model of $T_{\mathcal{A},\mathrm{V}}$, we say that z is a* ε-Mahlo set *if z is a ε-inaccessible set and for every $u \,\varepsilon\, z$ and every binary relation R, there is a ε-inaccessible set $v \,\varepsilon\, z$ such that*

1. $u \,\varepsilon\, v$,
2. $\forall x \,\varepsilon\, v(\exists y \,\varepsilon\, z\ \mathtt{op}(x,y) \,\varepsilon\, R \to \exists y \,\varepsilon\, v\ \mathtt{op}(x,y) \,\varepsilon\, R)$.

▶ **Lemma 32.** *Let $\mathcal{N} = (\mathrm{V}, \mathcal{A})$ be a realizability model and suppose that z is a Mahlo set in* V *such that $\mathcal{A} \in z$, then $\mathcal{N} \Vdash z^{\mathcal{A}}$ is a ε-Mahlo set.*

Proof. By Remark 28 we know that whenever v is an inaccessible set such that $\mathcal{A} \in v$, $v^{\mathcal{A}}$ is realized to be a ε-inaccessible set by a realizer that does not depend on v. In particular, this means that $z^{\mathcal{A}}$ is realized to be a ε-inaccessible set. To realize that $z^{\mathcal{A}}$ is a ε-Mahlo set, we fix $a \in z \cap \mathrm{N}$ and $R \in \mathrm{N}$. First, we define $R' \coloneqq \{((x,\pi),y) \mid (\mathtt{op}(x,y),\pi) \in R,\ y \in \mathrm{N}\}$. Since z is a Mahlo set in the ground model, we can find an inaccessible set v such that $a, \mathcal{A} \in v$ and R' reflects to v. The following hold:

1. $\mathbf{I} \Vdash v^{\mathcal{A}} \,\varepsilon\, z^{\mathcal{A}}$ and $\mathbf{I} \Vdash a \,\varepsilon\, v^{\mathcal{A}}$,
2. $v^{\mathcal{A}}$ is realized to be a ε-inaccessible set by a realizer that does not depend on v.

We want to realize that, in $\mathcal{N}$, R reflects to $v^{\mathcal{A}}$. To do this, it suffices to show that

$$\mathbf{I} \Vdash \forall x^{v^{\mathcal{A}}}(\forall y^{v^{\mathcal{A}}}\, \mathtt{op}(x,y) \notin R \to \forall y^{z^{\mathcal{A}}}\, \mathtt{op}(x,y) \notin R).$$

Fix t, π such that $t \Vdash \forall y^{v^{\mathcal{A}}}\, \mathtt{op}(x,y) \notin R$ and $\pi \in \|\forall y^{z^{\mathcal{A}}}\, \mathtt{op}(x,y) \notin R\|$. There is $y \in z \cap \mathrm{N}$ such that $(\mathtt{op}(x,y),\pi) \in R$, thus $((x,\pi),y) \in R'$. Since R' reflects to v there is $y' \in v$ such that $(\mathtt{op}(x,\pi),y') \in R'$. This means that $(\mathtt{op}(x,y'),\pi) \in R$ and $y' \in \mathrm{N}$. So $y' \in v \cap \mathrm{N}$, hence $\pi \in \|\forall y^{v^{\mathcal{A}}}\, \mathtt{op}(x,y) \notin R\|$ and $t \star \pi \in \bot\!\!\!\bot$. ◀

▶ **Theorem 33.** *Let $\mathcal{N} = (\mathrm{V}, \mathcal{A})$ be a realizability model, then $\mathcal{N} \Vdash \forall z(z$ is a ε-Mahlo set $\to z$ is a Mahlo set$)$*

Proof. We work within a realizability model. Suppose that z is a ε-Mahlo set. By Theorem 26 we know that z is an inaccessible set. Fix $u \in z$ and let R be a binary relation. Let $R_\varepsilon \coloneqq \{\mathtt{op}(x, y) \mid x, y \,\varepsilon\, z, \mathtt{op}(x, y) \in R\}$. Since z is a ε-Mahlo set we can fix a ε-inaccessible (and hence inaccessible) set $v \,\varepsilon\, z$ such that $u \,\varepsilon\, v$ and R_ε reflects to v. By Theorem 16, we know that for every x, y in the realizability model, $\mathtt{op}(x, y) \simeq (x, y)$. Since $u \,\varepsilon\, v \,\varepsilon\, z$ we have $u \in v \in z$. For the final property, fix $x \in v$ and suppose that $(x, y) \in R$ for some $y \in z$. Next, take $x' \,\varepsilon\, v$ and $y' \,\varepsilon\, z$ such that $x \simeq x'$ and $y \simeq y'$. Then, by definition, $\mathtt{op}(x', y') \,\varepsilon\, R_\varepsilon$ so, since R_ε reflects to v, we can find some $y'' \,\varepsilon\, v$ for which $\mathtt{op}(x', y'') \,\varepsilon\, R_\varepsilon$. Unpacking the definition of R_ε this means that $(x', y'') \in R$. Therefore, since $x \simeq x'$ we have that there exists some $y'' \in v$ for which $(x, y'') \in R$, as required. ◀

▶ **Corollary 34.** *Let $\mathcal{N} = (\mathrm{V}, \mathcal{A})$ be a realizability model. Assume that there is a Mahlo set z in* V *such that $\mathcal{A} \in z$. Then $\mathcal{N} \Vdash$ ZF + there exists a Mahlo set.*

9 Extending Realizability to Classes

Gödel-Bernays set theory (GB) is an extension of Zermelo-Frænkel set theory (ZF) with a built-in notion of *classes* – arbitrary collections of sets that may be too big to be sets themselves. In the section, we will show how to similarly extend ZF_ε to a theory GB_ε that supports classes, and we will show how to construct realizability models of GB_ε.

We will work in *two-sorted* first-order logic (without equality): one sort will represent *sets*, and the other, *classes*. We will use lowercase letters for set variables, and uppercase letters for class variables. Quantification over sets will be denoted by "$\forall^0$" and "$\exists^0$", and quantification over classes by "$\forall^1$" and "$\exists^1$" (though we may drop the exponents when there is no ambiguity).

Let $\mathcal{L}^2_\in$ denote the first-order language over the signature $\{\in^0, \in^1, \simeq\}$, where $\in^0$ and $\simeq$ are relation symbols of arity Set × Set, and $\in^1$ is a relation symbol of arity Set × Class. The reason why we need two versions of $\in$ is that both sets and classes can contain sets. Likewise, let $\mathcal{L}^2_\varepsilon$ denote the first-order language over the signature $\{\notin^0, \notin^1, \subseteq, \not\ni^0, \not\ni^1, \neq^0, \neq^1\}$, where $\notin^0$, $\subseteq$, $\not\ni^0$ and $\neq^0$ are relation symbols of arity Set × Set, $\not\ni^1$ and $\notin^1$ are relation symbols of arity Set × Class, and $\neq^1$ is a relation symbol of arity Class × Class. Since the context always makes it clear which "version" of a given relation symbol is being used, we will systematically drop these exponents and simply write $\notin$ and $\not\ni$.

The theory GB_ε is the theory over $\mathcal{L}^2_\varepsilon$ generated by the following axioms:

1. The axioms of ZF_ε, with the axioms schemas extended to all formulas of the language $\mathcal{L}^2_\varepsilon$ that contain *no quantifications over classes*.
2. Class Separation: $\forall^1 A\ \forall^0 b\ \exists^0 a\ \forall^0 x\ (x \,\varepsilon\, a \leftrightarrow x \,\varepsilon\, A \wedge x \,\varepsilon\, b)$.
3. Class Induction: $\forall^1 A\ \Big((\forall^0 x\ ((\forall^0 y \,\varepsilon\, x\ y \,\varepsilon\, A) \to x \,\varepsilon\, A)) \to \forall^0 z\ (z \,\varepsilon\, A)\Big)$.
4. Elementary Class Comprehension: $\forall^1 A\ \forall^0 u\ \exists^1 B\ \forall^0 x\ (x \,\varepsilon\, B \leftrightarrow \varphi(x, u, A))$ for every formula $\varphi(x, u, A)$ with no quantifications over classes.
5. Class Collection: $\forall^1 A\ \forall^0 u\ \forall^0 a\ \exists^0 b\ \forall^0 x \,\varepsilon\, a\ \Big((\exists^0 y\ \varphi(x, y, u, A)) \to (\exists^0 y \,\varepsilon\, b\ \varphi(x, y, u, A))\Big)$, for every formula $\varphi(x, y, u, A)$ with no quantifications over classes.
6. Definition of $\in^1$: $\forall^1 A\ \forall^0 x\ \Big(x \in A \leftrightarrow \exists^0 y\ (y \,\varepsilon\, A \wedge y \simeq x)\Big)$.

We refer the read to the end of Chapter 6 of [9] and Chapter 4 of [20] for more details on GB and second-order set theories in general. By a simple generalisation of the ZF case, we can see that the theory GB_ε is a conservative extension of the standard theory GB.

▶ **Theorem 35.** *Let φ be a closed formula in $\mathcal{L}^2_\in$, then $\mathrm{GB} \vdash \varphi$ if and only if $\mathrm{GB}_\varepsilon \vdash \varphi$.*

Whenever $\mathcal{L}^2$ is a first-order language that contains $\mathcal{L}^2_\varepsilon$, we will denote by $\mathrm{GB}_\varepsilon^{\mathcal{L}^2}$ the theory obtained by taking GB_ε and enriching all the axiom schemas to include all the formulas of $\mathcal{L}^2$ with no quantifications over classes.

Realizability Models with Classes

Let $(\mathrm{V}, \mathcal{C})$ be a model of GB and let $\mathcal{A} = (\Lambda, \Pi, \perp\!\!\!\perp)$ be a realizability algebra in V. Let:
- $\mathrm{N} \coloneqq \bigcup_{\alpha \in \mathrm{Ord}} \mathrm{N}_\alpha \in \mathcal{C}$, where $\mathrm{N}_\alpha \coloneqq \bigcup_{\beta<\alpha} \mathcal{P}(\mathrm{N}_\beta \times \Pi) \in \mathrm{V}$ as before,
- $\mathcal{D} \coloneqq \{X \in \mathcal{C} \mid X \subseteq \mathrm{N}\} \subseteq \mathcal{C}$.

As in Section 4, we let $\mathcal{L}^{\mathcal{A},2}_\varepsilon$ denote the language obtained by adding a function symbol f for each $\mathcal{A}$-definable function $f : \mathrm{N}^n \to \mathrm{N}$.

▶ **Definition 36.** *We extend Definition 5 to all formulas in $\mathcal{L}^{\mathcal{A},2}_\varepsilon$ with parameters in $(\mathrm{N}, \mathcal{D})$:*
- $\|a \notin^1 B\| \coloneqq \{\pi \in \Pi \mid (a, \pi) \in^1 B\}$;
- $\|A \neq^1 B\| \coloneqq \|\top\|$ *if* $A \neq B$, $\|\perp\|$ *otherwise;*
- $\|a \notin^1 B\| \coloneqq \bigcup_{c \in \mathrm{dom}(B)} \{t.t'.\pi \mid (c, \pi) \in^1 B,\ t \in |a \subseteq c|,\ t' \in |c \subseteq a|\}$;
- $\|\forall^1 X\, \varphi(X)\| \coloneqq \bigcup_{A \in \mathcal{D}} \|\varphi[A/X]\|$.

▶ Remark 37. By standard set-theoretic arguments, for each formula $\varphi(x_1, \dots, x_m, Y_1, \dots Y_n)$ in $\mathcal{L}^2_\varepsilon$, there are formulas $\varphi_\Pi(p, x_1, \dots, x_m, Y_1, \dots Y_n)$ and $\varphi_\Lambda(p, x_1, \dots, x_m, Y_1, \dots Y_n)$ in $\mathcal{L}^2_\in$ with parameters in $\mathrm{V}_{\mathrm{fp}(\mathcal{A})}$ such that for all $a_1, \dots, a_m \in \mathrm{V}$, all $B_1, \dots, B_n \in D$, all $\pi \in \Pi$, and all $t \in \Lambda$, we have

$$\pi \in \|\varphi(a_1, \dots, a_m, B_1, \dots, B_n)\| \quad \textit{iff} \quad \mathrm{V} \models \varphi_\Pi(\pi, a_1, \dots, a_m, B_1, \dots, B_n)$$
$$\textit{and} \quad t \in |\varphi(a_1, \dots, a_m, B_1, \dots, B_n)| \quad \textit{iff} \quad \mathrm{V} \models \varphi_\Lambda(t, a_1, \dots, a_m, B_1, \dots, B_n).$$

▶ **Definition 38.** *The* realizability theory of $(\mathcal{A}, \mathrm{V}, \mathcal{C})$*, denoted by $T_{\mathcal{A},\mathrm{V},\mathcal{C}}$, is the set of all closed formulas φ of $\mathcal{L}^{\mathcal{A},2}_\varepsilon$ with parameters in $(\mathrm{N}, \mathcal{D})$ such that there exists $t \in \mathcal{R}$ such that t realizes φ.*

▶ **Proposition 39.** *Let $(\mathrm{V}, \mathcal{C})$ be a model of* GB *and let $\mathcal{A} \in \mathrm{V}$ be a realizability algebra. The realizability theory of $(\mathcal{A}, \mathrm{V}, \mathcal{C})$ is closed under classical deduction and contains GB_ε.*

This justifies the following definition:

▶ **Definition 40.** *A* realizability model of GB_ε *is a tuple $\mathcal{N} = (\mathrm{V}, \mathcal{C}, \mathcal{A})$, with $(\mathrm{V}, \mathcal{C})$ a model of* GB *and $\mathcal{A}$ a realizability algebra in* V*. We write $\mathcal{N} \Vdash \varphi$ for "$T_{\mathcal{A},\mathrm{V},\mathcal{C}}$ contains φ".*

10 Realizing measurable and Reinhardt cardinals

In this section, we work with the theory GB and we build realizability models for measurable and Reinhardt cardinals. First, we will consider a model V of GB with Choice that contains a measurable cardinal and show how to *realize* the existence of a measurable cardinal. A cardinal κ is said to be measurable if there exists a non-principal, κ-complete ultrafilter over

κ. The reason we work with Choice in the ground model is to use the Łoś Theorem to define from the ultrafilter a class function $j\colon \mathrm{V} \to \mathrm{M}$ where M is a transitive inner model of ZFC, j is not the identity and for every formula $\varphi(x_1, \dots, x_n)$ and sets $a_1, \dots, a_n$ one has

$$\varphi(a_1, \dots, a_n) \text{ if and only if } \mathrm{M} \models \varphi(j(a_1), \dots, j(a_n)).$$

We call δ the *critical point* of an embedding j if $\forall \alpha \in \delta$, $j(\alpha) = \alpha$ and $j(\delta) > \delta$, that is to say δ is the first ordinal moved by j. It is well-known that κ is a measurable cardinal if and only if it is the critical point of such an elementary embedding.

What we shall realize is that the existence of such an embedding transfers nicely to realizability models and their corresponding extensional models of GB. The reason we work in a second-order theory is that formally the embedding $j\colon \mathrm{V} \to \mathrm{M}$ is a class function over our model. It is known that without Choice the Łoś Theorem need no longer hold and therefore the existence of an embedding might not be definable in a first-order way. On the other hand, it is easy to see that under ZF, if j is a non-trivial elementary embedding with critical point κ then there is a non-trivial κ-complete ultrafilter on κ defined by $U = \{X \subseteq \kappa \mid \kappa \in j(X)\}$. We refer the reader to [8] for more details on measurable cardinals without Choice.

When we are working in GB_ε it no longer needs to be the case that there is a unique critical point because we may have sets $a, b \in \mathrm{N}$ such that $\mathcal{N}$ believes a and b are ordinals with $a \simeq b$, the embedding will fix every element of both a and b while being non-trivial on them and yet $a \neq b$. Therefore, we shall instead refer to a critical point of some embedding. We shall show that there exists an embedding $j^\star$ and see that $\ulcorner(\kappa)$ is a critical point of $j^\star$. Then, when we restrict ourselves to the model of GB, the extensionality of $\in$ will give us the required uniqueness.

So let us fix such an embedding $j\colon \mathrm{V} \to \mathrm{M}$ and denote its critical point by κ. Let $\mathcal{A}$ be any realizability algebra such that $\mathcal{A} \in \mathrm{V}_\kappa$ (which implies $\mathrm{fp}(\mathcal{A}) < \kappa$). Then j will induce a non-trivial elementary embedding j^* of the realizability model into a transitive subclass of the realizability model that satisfies GB_ε where j^* is defined as

$$j^* = \{(\mathtt{op}(x, j(x)), \pi) \mid \pi \in \Pi\}. \tag{$\star$}$$

▶ **Definition 41.** *In a model of $T_{\mathcal{A},\mathrm{V},\mathcal{C}}$, we say that an ordinal a is* a critical point of j^* *if $\forall x \,\varepsilon\, a(\mathtt{op}(x, x) \,\varepsilon\, j^\star)$ and there exists some set b such that $\mathtt{op}(a, b) \,\varepsilon\, j^\star$ and $a \,\varepsilon\, b$.*

An important fact we will use is that the elementarity of j implies that $t \Vdash \varphi(a)$ if and only if $t \Vdash \varphi(j(a))$ (by Remark 37). In addition, since $\mathcal{A} \in \mathrm{V}_\kappa$, we have $\mathcal{P}(\Pi) \in \mathrm{M}$ so $\|\varphi\|^{\mathrm{M}} = \|\varphi\|$.

▶ **Proposition 42.** *For all formulas φ in $\mathcal{L}_\varepsilon^{\mathcal{A}}$ and all $a_1, \dots, a_n \in \mathrm{N}$, $\|\varphi(a_1, \dots, a_n)\| = \|\varphi(j(a_1), \dots, j(a_n))\|$. Hence $\mathcal{N} \Vdash \varphi(a_1, \dots, a_n) \leftrightarrow \varphi(j(a_1), \dots, j(a_n))$.*

From this it is easy to see that $j^\star$ is realized to be a class elementary embedding.

▶ **Theorem 43.** *Suppose that $(\mathrm{V}, \mathcal{C})$ is a model of* GB *and $\mathcal{A} \in \mathrm{V}_\kappa$ where κ is the critical point of a non-trivial elementary embedding $j\colon \mathrm{V} \to \mathrm{M}$ for some class function j and transitive class* M*. Then $\mathcal{N}$ realizes every axiom of GB_ε plus:*

1. *$\mathcal{N} \Vdash j^\star$ is a class function compatible with $\simeq$, namely*

$$\mathcal{N} \Vdash \forall x_1 \forall x_2 \forall y_1 \forall y_2((x_1 \simeq x_2 \wedge \mathtt{op}(x_1, y_1) \,\varepsilon\, j^\star \wedge \mathtt{op}(x_2, y_2) \,\varepsilon\, j^\star) \to y_1 \simeq y_2).$$

2. *$\mathcal{N} \Vdash \forall x \forall y(\mathtt{op}(x, y) \,\varepsilon\, j^\star \to y \,\varepsilon\, \mathrm{M}^{\mathcal{A}}$,*
3. *$\mathcal{N} \Vdash \forall \alpha^{\ulcorner(\kappa)} \mathtt{op}(\alpha, \alpha) \,\varepsilon\, j^\star \wedge \exists b(\ulcorner(\kappa) \,\varepsilon\, b \wedge \mathtt{op}(\ulcorner(\kappa), b) \,\varepsilon\, j^\star$ (i.e.$\ulcorner(\kappa)$ is a critical point of $j^\star$),*
4. *For every formula $\varphi(x_1, \dots, x_n)$ in Fml_ε we have*

$$\mathcal{N} \Vdash \forall a_1, \dots, a_n \exists b_1, \dots, b_n(\mathtt{op}(a_1, b_1) \,\varepsilon\, j^\star \wedge \dots \wedge \mathtt{op}(a_1, b_1) \,\varepsilon\, j^\star \wedge (\varphi(\vec{a}) \leftrightarrow \varphi^{\mathrm{M}^{\mathcal{A}}}(\vec{b}))).$$

▶ **Theorem 44.** *Suppose that* $(\mathrm{V},\mathcal{C})$ *is a model of* GB *plus Choice with a measurable cardinal* κ*, and* $\mathcal{A}\in\mathrm{V}_\kappa$ *is a realizability algebra. Let* $\mathcal{N}=(\mathrm{V},\mathcal{A})$ *be the corresponding realizability model. Then,* $\mathcal{N}\Vdash$ GB + *there exists a measurable cardinal.*

Proof. Let $j\colon \mathrm{V}\to\mathrm{M}$ be an elementary embedding with critical point κ witnessing that κ is a measurable cardinal and define $j^\star$ as in $(\star)$. By Theorem 35 and Proposition 39, we have $\mathcal{N}\Vdash$ GB. Theorem 43(1) proves that $j^\star$ is a function compatible with the extensional equality. Theorem 43(3) implies that $j^\star$ is non-trivial and $\neg(\kappa)$ is a critical point. Theorem 43(4) implies that $j^\star$ is elementary for formulas in the language of $\mathcal{L}_\varepsilon$ (and hence $\mathcal{L}_\in$). Therefore, we can realize that there is an elementary embedding $j^\star\colon \mathrm{N}\to\mathrm{M}^{\mathcal{A}}$ with critical point $\neg(\kappa)$. Finally, from this it follows that there is a non-principal $\neg(\kappa)$-complete ultrafilter on $\neg(\kappa)$ so $\neg(\kappa)$ is indeed a measurable cardinal. ◀

We end this section by showing that the above analysis naturally generalizes to larger cardinal notions involving the notion of elementary embedding, in particular Reinhardt cardinals. So suppose that $(\mathrm{V},\mathcal{C})$ is a model of GB that contains a Reinhardt cardinal κ, this means that κ is the critical point of a non-trivial elementary embedding j of the universe into itself. It is important here that we work over GB because if we only consider Reinhardt cardinals in a purely first-order setting then any proper class must be definable by some formula. It is then possible to show that there are no definable elementary embedding of the universe into itself by work of Suzuki [23] or see [7] for an extended discussion on the metamathematics of dealing with Reinhardt cardinals.

▶ **Theorem 45.** *Suppose that* $(\mathrm{V},\mathcal{C})$ *is a model of* GB *with a Reinhardt cardinal* κ*, and* $\mathcal{A}\in\mathrm{V}_\kappa$ *is a realizability algebra. Let* $\mathcal{N}=(\mathrm{V},\mathcal{A})$ *be the corresponding realizability model, then* $\mathcal{N}\models$ GB + *there exists a Reinhardt cardinal.*

Proof. Let $j\colon \mathrm{V}\to\mathrm{V}$ be an elementary embedding witnessing that κ is a Reinhardt cardinal, with j^* defined as in $(\star)$. By Theorem 35 and Proposition 39, we have $\mathcal{N}\Vdash$ GB. As before, from Theorem 43 one can realize that $j^\star$ is a non trivial function which is compatible with the extensional equality and has $\neg(\kappa)$ as a critical point. An easy generalization of Theorem 43(4) implies that j^* is elementary for GB formulas. Therefore, the existence of a Reinhardt cardinal is realized. ◀

11 Conclusion

We have shown how to realize the axioms of inaccessible, Mahlo, measurable and Reinhardt cardinals assuming their consistency relative to ZFC or ZF. We have reformulated each of these axioms in the context of ZF or GB and we have proven that the corresponding notions are preserved by any realizability algebra whose size is smaller than the large cardinals considered, in particular by any countable realizability algebra. Note that in each one of these four scenarios, no special instructions were needed, the axioms considered are realized by pure terms of the λ_c-calculus. This may be counterintuitive since large cardinals axioms are very strong axioms which entail the consistency of ZFC, yet our results show that despite their strength, large cardinals axioms do not add a computational content to the realizability machinery whenever the algebra is small enough (namely countable or of size smaller than the large cardinal considered). On the other hand, it remains an open problem to determine what would happen for a larger realizability algebra (equipotent with the large cardinal considered or larger). For example, in the case of forcing, if the forcing has size κ then it may collapse κ to be bijective with a smaller cardinal. Hence special instructions may be needed in order to prevent the large cardinal from collapsing and to preserve its properties.

There is opportunity for future work involving realizability models for large cardinals. The method presented to realize measurable and Reinhardt cardinals focused on preservation of non trivial elementary embeddings; this could be easily adapted to realize rank-into-rank embeddings, which correspond to some of the strongest known large cardinals axioms not known to be inconsistent in ZFC. Preservation of other large cardinal notions could be investigated such as Ramsey cardinals, weakly compact, strongly compact, supercompact cardinals and many others. Clearly, assuming larger cardinals we can get realizability models for those large cardinals notions: for instance if we assume the consistency of ZFC with a measurable cardinal, we can realize the existence of a measurable cardinal and in particular of a Ramsey cardinal, but it remains an open problem whether Ramsey cardinals can be *preserved* by realizability algebras, namely whether starting from a model of ZFC with a Ramsey cardinal κ such that $|\mathcal{A}| < \kappa$, one can realize the existence of a Ramsey cardinal in the corresponding realizability model.

References

1 Hendrik Pieter Barendregt. *The lambda calculus - its syntax and semantics*, volume 103 of *Studies in logic and the foundations of mathematics*. North-Holland, 1985.

2 Andreas Blass. Exact functors and measurable cardinals. *Pacific journal of mathematics*, 63(2):335–346, 1976.

3 Matthias Felleisen, Daniel P. Friedman, Eugene E. Kohlbecker, and Bruce F. Duba. A syntactic theory of sequential control. *Theor. Comput. Sci.*, 52:205–237, 1987. doi:10.1016/0304-3975(87)90109-5.

4 Harvey Friedman. The consistency of classical set theory relative to a set theory with intu1tionistic logic. *The Journal of Symbolic Logic*, 38(2):315–319, 1973.

5 Harvey Friedman and Andrej Ščedrov. Large sets in intuitionistic set theory. *Annals of pure and applied logic*, 27(1):1–24, 1984.

6 Timothy G. Griffin. A formulae-as-type notion of control. In *Proceedings of the 17th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 47 , 58, 1989.

7 Joel David Hamkins, Greg Kirmayer, and Norman Lewis Perlmutter. Generalizations of the Kunen inconsistency. *Annals of Pure and Applied Logic*, 163(12):1872–1890, 2012.

8 Yair Hayut and Asaf Karagila. Critical cardinals. *Israel journal of mathematics*, 236(1):449–472, 2020.

9 Thomas Jech. *Set Theory*. Springer Monographs in Mathematics. Springer Berlin, Heidelberg, 2003.

10 Akihiro Kanamori. *The higher infinite: large cardinals in set theory from their beginnings*. Springer Science & Business Media, 2008.

11 Stephen Cole Kleene. On the interpretation of intuitionistic number theory. *The Journal of Symbolic Logic*, 10(4):109–124, 1945.

12 Jean-Louis Krivine. Typed lambda-calculus in classical Zermelo-Fraenkel set theory. *Archive of Mathematical Logic*, 40(3):189–205, 2001.

13 Jean-Louis Krivine. Realizability in classical logic. *Panoramas et synthéses, Sociètè Mathématique de France*, 27:197–229, 2009.

14 Jean-Louis Krivine. Realizability algebras II: new models of ZF+ DC. *Logical Methods in Computer Science*, 8:1–28, 2012.

15 Jean-Louis Krivine. Realizability algebras III: some examples. *Mathematical Structures in Computer Science*, 28(1):45–76, 2018.

16 Jean-Louis Krivine. A program for the full axiom of choice. *Logical Methods in Computer Science*, 17(3):1–22, 2021.

17 Kenneth Kunen. Elementary embeddings and infinitary combinatorics. *The Journal of Symbolic Logic*, 36(3):407–413, 1971.

18 Azriel Lévy. Axiom schemata of strong infinity in axiomatic set theory. *Pacific journal of mathematics*, 10(1):223–238, 1960.

19 Richard Matthews. A Guide to Krivine Realizability for Set Theory. *arxiv preprint*, 2023. `arXiv:2307.13563`.

20 Elliott Mendelson. *Introduction to Mathematical Logic*. CRC Press, 5 edition, 2009.

21 Michael Rathjen. Realizing Mahlo set theory in type theory. *Archive for Mathematical Logic*, 42:89–101, 2003.

22 Anton Setzer. Extending Martin-Löf type theory by one Mahlo universe. *Archive for Mathematical Logic*, 39(3):155–181, 2000.

23 Akira Suzuki. No elementary embedding from V into V is definable from parameters. *The Journal of Symbolic Logic*, 64(4):1591–1594, 1999.

24 Neil H. Williams. On Grothendieck universes. *Compositio Mathematicae*, 21(1):1–3, 1969.

The Kleene-Post and Post's Theorem in the Calculus of Inductive Constructions

Yannick Forster
Inria, Nantes Université, LS2N, Nantes, France

Dominik Kirst
Ben-Gurion University of the Negev, Beer-Sheva, Israel
Saarland University, Saarland Informatics Campus, Saarbrücken, Germany

Niklas Mück
Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
MPI-SWS, Saarland Informatics Campus, Saarbrücken, Germany

Abstract

The Kleene-Post theorem and Post's theorem are two central and historically important results in the development of oracle computability theory, clarifying the structure of Turing reducibility degrees. They state, respectively, that there are incomparable Turing degrees and that the arithmetical hierarchy is connected to the relativised form of the halting problem defined via Turing jumps.

We study these two results in the calculus of inductive constructions (CIC), the constructive type theory underlying the Coq proof assistant. CIC constitutes an ideal foundation for the formalisation of computability theory for two reasons: First, like in other constructive foundations, computable functions can be treated via axioms as a purely synthetic notion rather than being defined in terms of a concrete analytic model of computation such as Turing machines. Furthermore and uniquely, CIC allows consistently assuming classical logic via the law of excluded middle or weaker variants on top of axioms for synthetic computability, enabling both fully classical developments and taking the perspective of constructive reverse mathematics on computability theory.

In the present paper, we give a fully constructive construction of two Turing-incomparable degrees à la Kleene-Post and observe that the classical content of Post's theorem seems to be related to the arithmetical hierarchy of the law of excluded middle due to Akama et. al. Technically, we base our investigation on a previously studied notion of synthetic oracle computability and contribute the first consistency proof of a suitable enumeration axiom. All results discussed in the paper are mechanised and contributed to the Coq library of synthetic computability.

2012 ACM Subject Classification Theory of computation → Constructive mathematics; Theory of computation → Type theory

Keywords and phrases Constructive mathematics, Computability theory, Logical foundations, Constructive type theory, Interactive theorem proving, Coq proof assistant

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.29

Supplementary Material *Software (Source Code)*: `https://github.com/uds-psl/coq-synthetic-computability/tree/code-paper-kleene-post-post`
archived at `swh:1:dir:f11ba122d0e1aa97b47e89dd70353fa42a88c7f9`

Funding *Yannick Forster*: received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 101024493.
Dominik Kirst: is supported by a Minerva Fellowship of the Minerva Stiftung Gesellschaft für die Forschung mbH.

Acknowledgements We want to thank Felix Jahn, Gert Smolka, Dominique Larchey-Wendling, and the participants of the TYPES '22 conference for many fruitful discussions about Turing reducibility, Ian Shillito and the anonymous reviewers of this paper for helpful feedback, as well as Martin Baillon, Yann Leray, Assia Mahboubi, Pierre-Marie Pédrot, and Matthieu Piquerez for discussions about notions of continuity. The central inspiration to start working on Turing reducibility in type theory

is due to Andrej Bauer's talk at the Wisconsin logic seminar in February 2021. Furthermore, the first two authors want to thank Benjamin Kaminski, his research group, and the royals of the castle for hosting their nostalgic stay in Saarbrücken to finish writing this paper.

1 Introduction

We study two well-known results in computability theory from the perspective of synthetic mathematics: the Kleene-Post theorem [31], stating that there are incomparable Turing degrees,[1] and Post's theorem [41], establishing a close link between Turing jumps and the arithmetical hierarchy due to Kleene [30] and Mostowski [35]. Both have been historically important: The former clarifies that Turing degrees are not linearly ordered, whereas the latter links a purely logical characterisation of sets of numbers with oracle computations.

Although still covered in the standard canon of computability textbooks [43, 38, 45], these theorems pose an interesting benchmark for the synthetic approach since they involve higher-order notions like Turing reductions and relative semi-decidability, that are not obvious to represent synthetically, see e.g. the discussion in the PhD thesis of the first author [12, §9.2]. Furthermore, both results also crucially rely on an enumeration of oracle machines, which has not been established as a consistent axiom in synthetic computability yet. Therefore in previous work [15], we have first suggested a careful definition of oracle computability and conjectured that it allows to derive the consistency of such an enumeration. In the present paper, we confirm this conjecture by constructing an enumeration from the well-known axiom Church's thesis (CT) [33] and then use this enumeration to prove and analyse the Kleene-Post and Post's theorem.

Synthetic computability exploits the fact that in a constructive foundation of mathematics only computable functions are definable a priori. Non-computable functions arise a posteriori when combining function existence principles such as countable choice or unique choice with classical axioms like the law of excluded middle (LEM) or weaker counter-parts such as the limited principle of omniscience (LPO) or the weak limited principle of omniscience (WLPO). In constructive settings, where such classical principles would have to be explicitly assumed, the theory of computable functions can thus be studied by considering the whole function space as computable: a so-called synthetic approach allowing for a concise but precise mathematical development. In contrast, in classical settings such as ZFC set theory one has to resort to an analytic model of computability like Turing machines or one of its equivalents, cluttering formal definitions and proofs with computability conditions.

Synthetic approaches to computability have been expressed in several dialects of constructive mathematics: Markov's work in the Russian school of constructivism relies explicitly on a computational background theory [34]. Kreisel's formulation of the axiom CT internalises the fact that every (definable) function is computable in a model of computation [33], e.g. working over intuitionistic Heyting arithmetic. Working in Bishop-style constructive mathematics, Richman [42] suggests an axiom stating that the partial function space is enumerable, which can be stated without even defining models of computation. Bauer [2, 3] works in the effective topos [23] where the set of enumerable sets is enumerable, formulated as the axiom EA, and Swan and Uemura [46] establish the consistency of CT for univalent type theory.

[1] The seminal 1954 paper by Kleene and Post establishes various other results besides this one. In particular, it also proves that both constructed degrees Turing reduce to the halting problem. We follow the terminology to only use the incomparability part of the result used e.g. in the textbooks by Odifreddi [38] and Cooper [6] as well as more recent work on (classical) reverse mathematics by Sanders [44] and Brattka et. al [5].

For the specific case of Turing reductions, Bauer [4] characterises an oracle computation by a higher-order functional with certain continuity and computability conditions. However, due to countable choice being present, his setting based on the effective topos is inherently anti-classical, i.e. no axioms like LPO or even WLPO can be assumed consistently on top of EA, so one is bound to fully constructive reasoning. Recently proposing an alternative definition, Swan [47] works around the incompatibility of univalent mathematics with classical axioms in synthetic computability due to the presence of unique choice [11] by characterising oracle computations via 0-truncated $\neg\neg$-sheafification. As this implements a negative translation making constructive distinctions invisible, one is bound to fully classical reasoning. Thus, both settings are unusable for a sub-classical logical analysis of computability theory in the style of constructive reverse mathematics [24, 10]. Offering a solution, such an analysis is possible in our setting since the calculus of inductive constructions (CIC) [7, 8, 39] provides a universe of (possibly classical) propositions mostly disconnected from the (possibly computable) function spaces, so we can freely assume and distinguish classical axioms together with the base axiom for synthetic computability.

In the case of the Kleene-Post theorem, we report on a fully constructive proof that can be obtained by standard techniques of modelling mathematics in a constructive foundation. In the case of Post's theorem, we localise the use of classical logic: Based on Akama et. al's arithmetical hierarchy of the law of excluded middle [1] we derive that Σ_n-LEM is sufficient to obtain Post's theorem up to the same level n. This remains a preliminary analysis, however, since we do not prove that Post's theorem at level n in turn implies Σ_n-LEM. For many auxiliary results, e.g. for closure properties of the arithmetical hierarchy, we conjecture that weaker axioms would suffice. In particular, we observe a seemingly weaker variant of Markov's principle at play that appears not to have been treated in the literature before.

A preliminary proof of the Kleene-Post theorem with an assumed enumerator for a weaker definition of Turing reductions has been discussed in an extended abstract at TYPES '22 [26]. The same abstract discusses a proof of Post's theorem with a similar assumption and using the full law of excluded middle, based on the Bachelor's thesis of the third author [37].

Contributions. We contribute a consistency proof of an enumeration axiom for oracle computable functionals, derived from the well-known axiom CT and its fully synthetic variant EPF [13, 12]. Based on this axiom, we give a fully constructive synthetic proof of the Kleene-Post theorem [31] following Odifreddi [38] and a synthetic definition of the Turing jump. We then give the first formal definition of the arithmetical hierarchy in constructive type theory and mechanise several proofs due to Akama et. al [1] about the arithmetical hierarchy of classical axioms. We use axioms from the Σ_n-level of this hierarchy to prove Post's theorem [41] and discuss perspectives regarding a reverse analysis. Lastly, we give a more traditional definition of the arithmetical hierarchy via a syntactic modeling of first-order logic and prove that these hierarchies are equivalent if and only if CT holds. All proofs are machine-checked using the Coq proof assistant [48] and all statements in this PDF are hyperlinked with the HTML version of the proofs.

Outline. After collecting some preliminary definitions and notations in Section 2, we recall the concept and basic properties of synthetic oracle computability of [15] in Section 3. We next derive an enumerator of oracle computations from established axioms for synthetic computability (Section 4), followed by a first application of the enumerator to derive the Kleene-Post theorem (Section 5). To prepare the second application regarding Post's theorem (Section 9), we first study Turing jumps (Section 6), the arithmetical hierarchy (Section 7),

and classical assumptions characterising the structure of arithmetical sets (Section 8). We complement the technical development with a purely syntactic definition of the arithmetical hierarchy in Section 10 and conclude in Section 11.

2 Preliminaries

We use inductive types of natural numbers $\mathbb{N}$ and booleans $\mathbb{B}$ with constructors true and false, lists X^* with constructors $[\,]$ and $x :: l$ and concatenation operation $l_1 +\!\!+ l_2$, the sum type $X + Y$ with constructors $\mathsf{inl}\, x$ and $\mathsf{inr}\, x$, and vectors X^n with same notations as for lists.

We use a type of partial functions $X \rightharpoonup Y$, and write $fx \triangleright y$ if fx is defined with value y. The concrete implementation of partial functions is not important. Mathematically, we abstract away from details, whereas in the Coq formalisation we work against an abstract interface of partial functions, that can for instance be instantiated using step-indexing.

Two crucial properties are that the graph of partial functions should be testable via step-indexing and that one can perform unbounded search:

Lemma 1. *Partial functions have the following properties:*

1. *There is a function* $\epsilon\colon (X \rightharpoonup Y) \to \mathbb{N} \to X \to Y \to \mathbb{B}$ *with* $fx \triangleright y \leftrightarrow \exists n.\ \epsilon\ f\ n\ x\ y = \mathsf{true}$.
2. *There is a function* $\mu\colon (\mathbb{N} \rightharpoonup \mathbb{B}) \rightharpoonup \mathbb{N}$ *with* $\mu f \triangleright n \leftrightarrow fn \triangleright \mathsf{true} \land \forall m < n.\ fm \triangleright \mathsf{false}$.

As is common in type theory we work with predicates instead of sets, a formality that does not introduce any mathematical overhead or relevant change of presentation. Predicates are defined as functions into the universe of propositions, i.e. $p\colon X \to \mathbb{P}$. We define the complement of a predicate as $\overline{p}x := \neg px$. The universe of propositions is impredicative in CIC, which plays no essential role. More relevantly, propositions in CIC cannot in general be analysed in computations, meaning that e.g. projection functions of type $(\exists n : \mathbb{N}.\ px) \to \mathbb{N}$ can only be defined in special circumstances, not in general.

We now define basic notions of synthetic computability theory [16, 2]: decidability, semi-decidability, and many-one reducibility.
A predicate $p\colon X \to \mathbb{P}$ is decidable if it is reflected by a boolean function:

$$\mathcal{D}(p) := \exists f\colon X \to \mathbb{B}.\ \forall x\colon X.\ px \leftrightarrow fx = \mathsf{true}$$

We define semi-decidability using partial functions into the type $\mathbb{1}$ with only element $\star$:

$$\mathcal{S}(p) := \exists g\colon X \rightharpoonup \mathbb{1}.\ \forall x\colon X.\ px \leftrightarrow gx \triangleright \star$$

Lastly, a predicate $p\colon X \to \mathbb{P}$ is many-one reducible to $q\colon Y \to \mathbb{P}$ if p can be encoded into q:

$$p \preceq_m q := \exists f\colon X \to Y.\ \forall x.\ px \leftrightarrow q(fx)$$

The biggest deviation from paper presentations of computability theory is that we do not consider equivalences classes w.r.t. any reducibility, which are notoriously hard to treat in formalised approaches, but rather talk about concrete representatives of equivalence classes.

3 Oracle Computability and Turing Reducibility

We use the synthetic definition of oracle computability introduced in prior work [15]. It is based on a notion of computability of functionals $F\colon (Q \to A \to \mathbb{P}) \to I \to O \to \mathbb{P}$. The argument $R\colon Q \to A \to \mathbb{P}$ is to be read as the oracle relating questions $q : Q$ to answers $a : A$, $i\colon I$ is the input to the computation, and $o\colon O$ is the output. Technically, synthetic oracle computability

of such functionals is based on a notion of *continuity* via a partial and more extensional variant of dialogue trees [49]. Conceptually, considering oracle computations via continuous functionals was introduced by Kleene [29] and Kreisel [32]. Kleene calls such functionals *countable*, since they can analytically be represented in a model of computation. Synthetically, they become countable using axioms for synthetic computability as discussed in Section 4.

A functional $F\colon (Q\to A\to\mathbb{P})\to I\to O\to\mathbb{P}$. is considered (oracle)-computable if there is an underlying computation tree $\tau\colon I\to A^*\rightharpoonup Q+O$ capturing the extensional behaviour of F by

$$\forall Rxb.\ FRxb \leftrightarrow \exists qs\ as.\ \tau x\ ;\ R\vdash\ qs\ ;\ as\wedge \tau\ x\ as\triangleright \mathsf{out}\ b$$

where the *interrogation relation* $\sigma; R\vdash qs; as$ is inductively defined for $\sigma\colon A^*\rightharpoonup Q+O$ as

$$\frac{}{\sigma\ ;\ R\vdash [\,]\ ;\ [\,]} \qquad \frac{\sigma\ ;\ R\vdash\ qs\ ;\ as \qquad \sigma\, as\triangleright \mathsf{ask}\ q \qquad Rqa}{\sigma\ ;\ R\vdash\ qs +\!\!+ [q]\ ;\ as +\!\!+ [a]}$$

and where we write $\mathsf{ask}\ q$ and $\mathsf{out}\ o$ for the respective injections into the sum type $Q+O$.

Intuitively, a computation tree takes as input a list of answers given already by the oracle. It can then (1) ask another question to the oracle, (2) return an output, or (3) compute forever while neither asking a question or returning an output. The computation on an input then is described by a sequence of runs of the tree, first given the empty list $[\,]$ as input, and then subsequently the list of all answers produced by the oracle. We use τ as letter for trees that take input and σ for trees that do not.

We now define Turing reducibility from p to q as computable functionals that map the characteristic relation of q to the characteristic relation of p. To this end, given $r\colon Z\to\mathbb{P}$, we define its characteristic relation $\hat{r}\colon Z\to\mathbb{B}\to\mathbb{P}$ as

$$\hat{r}zb := \textbf{if}\ b\ \textbf{then}\ rz\ \textbf{else}\ \neg rz$$

Then a Turing reduction from $p\colon X\to\mathbb{P}$ to $q\colon Y\to\mathbb{P}$ is an oracle-computable functional $F\colon (Y\to\mathbb{B}\to\mathbb{P})\to X\to\mathbb{B}\to\mathbb{P}$ such that $\forall xb.\,\hat{p}xb\leftrightarrow F\hat{q}xb$ and we write $p\preceq_{\mathsf{T}} q$ if such F exists.

Technically we do not work explicitly with trees in this paper, but rely on the following properties [15], establishing essentially the closure under a certain function algebra, containing variants of applications, constants, identity, branching, composition, and unbounded search.

Lemma 2. *The following functionals are computable provided the given functionals are:*

1. $\lambda Rio.\ F(gi)o$ *of type* $(Q\to A\to\mathbb{P})\to I\to O\to\mathbb{P}$ *given* $g\colon I\to I'$ *and* $F\colon (Q\to A\to\mathbb{P})\to I\to O\to\mathbb{P}$,
2. $\lambda Rio.\ \bot$ *of type* $(Q\to A\to\mathbb{P})\to I\to O\to\mathbb{P}$,
3. $\lambda Rio.\ fi\triangleright o$ *of type* $(Q\to A\to\mathbb{P})\to I\to O\to\mathbb{P}$ *given* $f\colon I\rightharpoonup O$,
4. $\lambda Rio.\ fi = o$ *of type* $(Q\to A\to\mathbb{P})\to I\to O\to\mathbb{P}$ *given* $f\colon I\to O$,
5. $\lambda Rio.\ o = v$ *of type* $(Q\to A\to\mathbb{P})\to I\to O\to\mathbb{P}$ *given* $v\colon O$,
6. $\lambda Rio.\ Rio$ *of type* $(I\to O\to\mathbb{P})\to I\to O\to\mathbb{P}$,
7. $\lambda Rio.$ **if** fi **then** $F_1 Rio$ **else** $F_2 Rio$ *of type* $(Q\to A\to\mathbb{P})\to I\to O\to\mathbb{P}$ *given* F_1, F_2 *of the same type and* $f : I\to\mathbb{B}$,
8. $\lambda Rio.\ \exists o'\colon O'.\ F_1\ R\ i\ o'\wedge F_2\ R\ (i,o')\ o$ *of type* $(Q\to A\to\mathbb{P})\to I\to O\to\mathbb{P}$ *given* $F_1\colon (Q\to A\to\mathbb{P})\to (I\to O'\to\mathbb{P})$ *and* $F_2\colon (Q\to A\to\mathbb{P})\to ((I\times O')\to O\to\mathbb{P})$,
9. $\lambda Rin.\ R\ (i,n)\ \mathsf{true}\wedge\forall m<n.\ R\ (i,m)\ \mathsf{false}$ *of type* $((I\times\mathbb{N})\to\mathbb{B}\to\mathbb{P})\to I\to\mathbb{N}\to\mathbb{P}$.

Lemma 3. *If* $p\preceq_m q$, *then* $p\preceq_{\mathsf{T}} q$.

Proof. Let f be given such that $\forall x.\ px\leftrightarrow q(fx)$. Define $FRxb := R(fx)b$, which is computable with Lemma 2 (**1**) and (**6**). We have that $\hat{p}xb\leftrightarrow F\hat{q}xb$ by case analysis on b. ◀

Turing reducibility can be seen as decidability of p relative to q. Similarly, we also define the central notion of relative semi-decidability. A predicate $p\colon X\to\mathbb{P}$ is semi-decidable in $q\colon Y\to\mathbb{P}$ if there is an oracle-computable functional $F\colon (Y\to\mathbb{B}\to\mathbb{P})\to(X\to\mathbb{1}\to\mathbb{P})$ such that $\forall x.\ px \leftrightarrow F\hat{q}x\star$ and we write $\mathcal{S}_q(p)$ if such F exists.

Lemma 4. *If $p \preceq_{\mathsf{T}} q$, then $\mathcal{S}_q(p)$ and $\mathcal{S}_q(\overline{p})$ for the complement $\overline{p}\,x := \neg p\,x$.*

Proof. See Lemma 3 in [15]. ◀

4 Enumeration Axiom for Synthetic Oracle Computability

Non-relative synthetic computability uses an axiomatic assumption of an enumerator of all partial functions [17], or equivalently of all enumerable predicates [14, 2], or equivalently a step-indexed interpreter enumerating all total functions [13]. All are consequences of the well-known axiom CT [33] stating that all total functions of type $\mathbb{N}\to\mathbb{N}$ are computable, which is consistent in type theory [46, 13].

For relative synthetic computability, we introduce a novel axiom and derive its consistency from the consistency of CT by constructing an enumerator of computable functionals based on an enumerator of partial functions.

The axiom EPF, itself a consequence of CT [13] and thus consistent, states that there is an enumerator $\theta\colon\mathbb{N}\to(\mathbb{N}\rightharpoonup\mathbb{N})$ universal for partial functions.

$$\mathsf{EPF} := \exists\theta\colon\mathbb{N}\to(\mathbb{N}\rightharpoonup\mathbb{N}).\ \forall f\colon\mathbb{N}\rightharpoonup\mathbb{N}.\exists c\colon\mathbb{N}.\ \forall xv.\ \theta_c x\rhd v \leftrightarrow fx\rhd v$$

Note that EPF implies the existence and undecidability of a synthetic variant of the self-halting problem $\mathcal{K}x := \exists v.\ \theta_x x\rhd v$ [12], and thus in particular we have e.g. $\mathcal{K}\not\preceq_{\mathsf{T}}\emptyset$.

For various results we will need a version working with families of functions $f\colon\mathbb{N}\to(\mathbb{N}\rightharpoonup\mathbb{N})$ and consequently coding functions $\gamma\colon\mathbb{N}\to\mathbb{N}$ such that f_n and $\theta_{\gamma n}$ agree, in line with the parametric version of the axiom EA used by Forster and Jahn [14]. Intuitively, this is related to having an *s-m-n* operator for θ. We can construct such a stronger form of θ directly by using a bijection between $\mathbb{N}$ and $\mathbb{N}\times\mathbb{N}$. This pairing function is written as $\langle x,y\rangle$ and we use the notation $f\langle x,y\rangle := \dots$ to define a function which takes as argument one single natural number, and uses the inverse of the pairing function to decompose it into x and y implicitly.

Lemma 5. EPF *is equivalent to the following parametric form:*

$$\exists\theta\colon\mathbb{N}\to(\mathbb{N}\rightharpoonup\mathbb{N}).\ \forall f\colon\mathbb{N}\to(\mathbb{N}\rightharpoonup\mathbb{N}).\exists\gamma\colon\mathbb{N}\to\mathbb{N}.\ \forall nxv.\ \theta_{\gamma n}x\rhd v \leftrightarrow f_n x\rhd v$$

Proof. The direction from right to left is immediate. From left to right, take θ' which enumerates all partial functions and define $\theta_{\langle c,n\rangle}x := \theta'_c\langle n,x\rangle$. Now given a family $f\colon\mathbb{N}\to(\mathbb{N}\rightharpoonup\mathbb{N})$, use universality of θ' on the function $g\langle n,x\rangle := f_n x$ to obtain c with $*:\forall av.\ \theta'_c a\rhd v \leftrightarrow ga\rhd v$. Then for $\gamma n := \langle c,n\rangle$ we have $\theta_{\gamma n}x\rhd v \overset{\gamma}{\leftrightarrow} \theta_{\langle c,n\rangle}x\rhd v \overset{\theta}{\leftrightarrow} \theta'_c\langle n,x\rangle\rhd v \overset{*}{\leftrightarrow} g\langle n,x\rangle\rhd v \overset{g}{\leftrightarrow} f_n x\rhd v$. ◀

We now explicitly explain the construction for an enumeration of all relative semi-deciders, but the same construction works for arbitrary computable functionals based on retracts of $\mathbb{N}$. To define an enumeration of all semi-deciders, we use retractions $\iota_{1,2}\colon X_{1,2}\to\mathbb{N}$ and $\rho_{1,2}\colon\mathbb{N}\to X_{1,2}$ with $\forall n.\ \iota_{1,2}(\rho_{1,2}n) = n$ for $X_1 := \mathbb{N}\times\mathbb{B}^*$ and $X_2 := \mathbb{N}+\mathbb{1}$.

We define $\xi\colon\mathbb{N}\to(\mathbb{N}\to\mathbb{B}^*\rightharpoonup\mathbb{N}+\mathbb{1})$ which then consequently parametrically enumerates every family of trees τ by:

$$\xi_c xl := \theta_c(\iota_1(x,l)) \gg\!= \lambda v.\ \mathsf{ret}\ (\rho_2 v) \quad \textit{with} \quad \forall\tau.\ \exists\gamma.\ \forall n\,x\,l\,v.\ \xi_{\gamma n}xl\rhd v \leftrightarrow \tau_n xl\rhd v$$

The bind notation $\gg\!=$ evaluates the left hand side to a value if possible, and then passes its value to the function on the right hand side. Given a tree $\tau\colon \mathbb{N}\to\mathbb{B}^*\rightharpoonup\mathbb{N}+\mathbb{1}$ we define $\hat{\tau}Rx := \exists qs\, as.\ \sigma\ ;\ R \vdash\ qs\ ;\ as \wedge \tau\ x\ as \triangleright \mathsf{out}\ \star$ and finally $\Xi_c Rx := \widehat{\xi_c} Rx$.

Lemma 6. *The relation $\lambda R(c,x)o.\ \Xi_c Rx$ is computable.*

Proof. Use $\lambda(c,x)l.\ \xi_c x l$. ◀

Lemma 7. *Given a family of trees, i.e. $\tau_i\colon \mathbb{N}\to\mathbb{B}^*\rightharpoonup\mathbb{N}+\mathbb{1}$, there exists a function $\gamma\colon\mathbb{N}\to\mathbb{N}$ such that $\forall iRx.\ \Xi_{\gamma i}Rx \leftrightarrow \hat{\tau}_i Rx$.*

Corollary 8. *Given a computable F there exists a code c such that $\forall Rx.\ FRx\star \leftrightarrow \Xi_c Rx$.*

We can give a similar enumerator for Turing reductions, which will be necessary for the Kleene-Post theorem. Note that we do not require the enumerator there to be parametric.

Theorem 9. *There is an enumerator of functionals $\chi\colon \mathbb{N}\to(\mathbb{N}\to\mathbb{B}\to\mathbb{P})\to\mathbb{N}\to\mathbb{B}\to\mathbb{P}$ such that*

1. *χ_c is computable and*
2. *given a computable F there exists a code c such that $\forall Rxb.\ FRxb \leftrightarrow \chi_c Rxb$.*

In the remainder of this paper, we just need to assume the enumerators Ξ and χ without any knowledge of their implementation. So their availability can be treated as an axiom for synthetic oracle computability and the construction provided in this section amounts to a consistency proof. In fact, this axiom likewise implies EPF:

Lemma 10. *The statement of Theorem 9 implies* EPF.

Proof. Instead of proving EPF, we prove the following version, which is equivalent [12]:

$$\mathsf{EPF}_{\mathbb{B}} := \exists\theta\colon\mathbb{N}\to(\mathbb{N}\rightharpoonup\mathbb{B}).\ \forall f\colon\mathbb{N}\rightharpoonup\mathbb{B}.\exists c\colon\mathbb{N}.\ \forall xv.\ \theta_c x\triangleright v \leftrightarrow fx\triangleright v$$

The proof is straightforwad by using that one can turn any function $f\colon\mathbb{N}\rightharpoonup\mathbb{B}$ into an oracle-computable F, such that $F\ (\lambda xv.\bot)$ agrees with f. ◀

5 The Kleene-Post Theorem

To establish incomparable Turing degrees, we adapt the proof given in Odifreddi's textbook [38] to our synthetic setting. Compared to the next sections, we here focus more on the intuition of the synthetic and constructive setting and omit some formal details. The usual strategy is to obtain said degrees as the unions $A := \bigcup_{n:\mathbb{N}} s_n$ and $B := \bigcup_{n:\mathbb{N}} t_n$ of cumulative increasing sequences s_n and t_n of boolean strings (interpreted implicitly as the relation arising from relating i to b for a string $b_0,\dots b_i\dots b_k$) such that the former take care that no χ_n induces a reduction $B \preceq_T A$ and the latter conversely rule out $A \preceq_T B$. Naturally, in our synthetic setting we are not able to define these sequences as functions $\mathbb{N}\to\mathbb{B}^*$, as this would force A and B decidable. Instead, we characterise both sequences simultaneously with an inductive predicate $\rightsquigarrow\colon \mathbb{N}\to\mathbb{B}^*\to\mathbb{B}^*\to\mathbb{P}$ such that $n\rightsquigarrow(s,t)$ represents s_n as s and t_n as t, by adding to the base case $0\rightsquigarrow([],[])$ the following inductive rules:

$$\frac{2n\rightsquigarrow(s,t)\qquad s'\sqsupseteq s\qquad \chi_n\, s'\,|t|\,b\qquad \forall u<s'.\,\neg\chi_n\,u\,|t|\,b}{2n+1\rightsquigarrow(s',t+\!\!+[\neg b])}\mathrm{E1}\qquad \frac{2n\rightsquigarrow(s,t)\qquad \neg(\exists s' b.\ s'\sqsupseteq s\wedge\chi_n\,s'\,|t|\,b)}{2n+1\rightsquigarrow(s,t+\!\!+[\mathsf{false}])}\mathrm{E2}$$

$$\frac{2n+1 \rightsquigarrow (s,t) \qquad t' \sqsupseteq t \qquad \chi_n\, t' \,|s|\, b \qquad \forall u < t'.\, \neg\chi_n\, u \,|s|\, b}{2n+2 \rightsquigarrow (s +\!\!+ [\neg b], t')}\mathrm{O1} \qquad \frac{2n+1 \rightsquigarrow (s,t) \qquad \neg(\exists t' b.\, t' \sqsupseteq t \wedge \chi_n\, t' \,|s|\, b)}{2n+2 \rightsquigarrow (s +\!\!+ [\mathsf{false}], t)}\mathrm{O2}$$

In every even step with $2n \rightsquigarrow (s,t)$ the sequences are extended such that χ_n applied to any prefix of A differs from any prefix of B at position $|t|$, either by flipping the result if χ_n already converges on some extension $s' \sqsupseteq s$ least with respect to some ordering $u < u'$ of strings (E1) or by setting a dummy value if χ_n diverges on all extensions (E2). Dually, in every odd step with $2n+1 \rightsquigarrow (s',t')$ it is taken care that χ_e applied to any prefix of B differs from any prefix of A.

We first show that the relation $n \rightsquigarrow (s,t)$ indeed captures a (classically total) cumulative increasing sequence of boolean strings:

Lemma 11. *The following properties of $n \rightsquigarrow (s,t)$ hold.*

1. *For every n there not not exist s and t with $n \rightsquigarrow (s,t)$.*
2. *If $n \rightsquigarrow (s,t)$ and $n' \rightsquigarrow (s',t')$ for $n \le n'$, then $s \sqsubseteq s'$ and $t \sqsubseteq t'$.*
3. *If $2n \rightsquigarrow (s,t)$ then $n \le |s|$ and $n \le |t|$.*

Proof. We give a detailed proof of (**1**) since it relies on careful constructive reasoning, the proofs of (**2**) and (**3**) are by routine arguments. The proof of (**1**) is by induction on n, in the case of 0 we just choose $s := []$ and $t := []$ and conclude with $0 \rightsquigarrow ([],[])$. In the case of $n+1$, we assume that there are no s' and t' with $n+1 \rightsquigarrow (s',t')$ and, given that we then want to derive a contradiction, may use the inductive hypothesis to assume positively that there are s and t with $n \rightsquigarrow (s,t)$. Since we still derive a contradiction, we can perform a case analysis whether or not there are $s' \sqsupseteq s$ and b with $\chi_n\, s' \,|t|\, b$, given that $\neg\neg(P \vee \neg P)$ holds constructively for every $P : \mathbb{P}$. If not, we just set $s' := s$ and $t' := t +\!\!+ [\mathsf{false}]$ and conclude $n+1 \rightsquigarrow (s',t')$ with (E2). If so, given the negative goal we can actually find a least such s', given that

$$(\exists n.\, p\, n) \to \neg\neg\exists n.\, p\, n \wedge \forall n' < n.\, p\, n$$

holds constructively for every $p : \mathbb{N} \to \mathbb{P}$. So for the least s', we set $t' := t +\!\!+ [\neg b]$ and conclude $n+1 \rightsquigarrow (s',t')$ with (E1). ◀

We next formally define the incomparable degrees A and B by

$$A\, x := \exists n\, s\, t.\, n \rightsquigarrow (s,t) \wedge s_x = \mathsf{true} \quad B\, x := \exists n\, s\, t.\, n \rightsquigarrow (s,t) \wedge t_x = \mathsf{true}$$

where s_x denotes the x-th element in s and is false other otherwise and state the central lemma used to show $B \not\leq_T A$, a dual version yields $A \not\leq_T B$.

Lemma 12. *If $2n \rightsquigarrow (s,t)$ and $2n+1 \rightsquigarrow (s',t')$, then $\hat{B} \,|t|\, b$ implies $\neg\chi_n\, \hat{A} \,|t|\, b$.*

Proof. We analyse how $2n+1 \rightsquigarrow (s',t')$ could have been derived from $2n \rightsquigarrow (s,t)$.

- In the case (E1), we have that $t' = t +\!\!+ [\neg b']$ and $\chi_n\, s' \,|t|\, b'$ for s' being the least such extension of s. By the former and the assumption $\hat{B} \,|t|\, b$ we derive $b = \neg b'$ using $t' \subseteq \hat{B}$ and $t'_{|t|} = \neg b'$. But then the further assumption $\chi_n\, \hat{A} \,|t|\, \neg b'$ is in conflict with $\chi_n\, s' \,|t|\, b'$ via monotonicity of oracle computations [15, Lemma 41], using $s' \subseteq \hat{A}$.
- In the case (E2), we have that $t' = t +\!\!+ [\mathsf{false}]$ and there is no extension $s' \sqsupseteq s$ with $\chi_n\, s' \,|t|\, b'$. However, if we now assume that $\chi_n\, \hat{A} \,|t|\, b$, then by modulus-continuity of oracle computations [15, Lemma 1] there is a finite prefix of $\hat{A}$ determining the outcome of $\chi_n\, \hat{A}$, in fact there is N such that $\chi_n\, s_N \,|t|\, b$ for all $N \rightsquigarrow (s_N, t_N)$. Now given the

negative goal, we can use (**1**) of Lemma 11 to actually obtain such s_N and t_N. Then by comparing $2n$ with N we either obtain $s \sqsubseteq s_N$ or $s_N \sqsubseteq s$ by (**2**) of Lemma 11, so in either case we find an extension $s' \sqsupseteq s$ with $\chi_n\, s'\, |t|\, b'$, in contradiction to the assumption. ◀

From this lemma the Kleene-Post theorem then follows immediately.

Theorem 13. *There are predicates A and B such that neither $A \preceq_T B$ nor $B \preceq_T A$.*

Proof. Suppose that $B \preceq_T A$, so $\chi_c\, \hat{A}\, x\, b \leftrightarrow \hat{B} x b$ for some c. Given that we have to derive a contradiction, we can argue classically enough to obtain $2n \rightsquigarrow (s,t)$, $2n+1 \rightsquigarrow (s',t')$, and $\hat{B}\, |t|\, b$. Then by Lemma 12 we obtain $\neg \chi_c\, \hat{A}\, |t|\, b$, contradicting $\chi_c\, \hat{A}\, |t|\, b \leftrightarrow \hat{B} |t| b$. That also $A \not\preceq_T B$ follows similarly. ◀

6 The Turing Jump

The Turing jump is the relativised equivalent to the self-halting problem: a number c is contained in the Turing jump of a predicate q if the c-th oracle machine halts on input c while given q as oracle. Formally, we define the Turing jump q' of a predicate $q : \mathbb{N} \to \mathbb{P}$ as

$$q'\, c := \Xi_c\, \hat{q}\, c\, \star\,.$$

Crucially, the jump is semi-decidable in the predicate, but its complement is not.

Lemma 14. *$\mathcal{S}_q(q')$ and $\neg \mathcal{S}_q(\overline{q'})$.*

Proof. Take $\lambda R c o.\ \Xi_c R c$ for $\mathcal{S}_q(q')$. For $\neg \mathcal{S}_q(\overline{q'})$, let F be computable and $\forall x.\ \neg q' x \leftrightarrow F \hat{q} x \star$. By definition, $\forall x.\ \neg q' x \leftrightarrow \neg \Xi_x \hat{q} x$. Using Corollary 8 we have c such that $\forall x.\ F \hat{q} x \star \leftrightarrow \Xi_c \hat{q} x$ and thus in particular $\neg \Xi_c \hat{q} c \leftrightarrow \Xi_c \hat{q} c$ – a contradiction. ◀

We now prove two standard results: First, that the Turing jump of a predicate is strictly higher in the order of Turing reducibility than the predicate itself, and secondly, that semi-decidability of p in q can be expressed as many-one reducibility of p to q'. To do so, we define an alternative Turing jump q° given $q : \mathbb{N} \to \mathbb{P}$ corresponding to a relativised halting problem, rather than a relativised self-halting problem q', as $q^\circ \langle c, x \rangle := \Xi_c\, \hat{q}\, x$.

Lemma 15. *$q' \preceq_m q^\circ$ and $q^\circ \preceq_m q'$.*

Proof. The first is by $\lambda c. \langle c, c \rangle$. For the second, use Lemma 7 for $\tau_{\langle c,x \rangle} n l := \xi_c x l$. ◀

Lemma 16. *$q \preceq_m q^\circ$*

Proof. Note that $\lambda R x o.\ R\, x\, \mathsf{true}$ is computable. Via Corollary 8 this means we have c with $\forall R x o.\ \Xi_c R x \leftrightarrow R x \mathsf{true}$. Now $\lambda x.\ \langle c, x \rangle$ is the wanted many-one reduction. ◀

Lemma 17. *$q \preceq_\mathsf{T} q'$ and $q' \not\preceq_\mathsf{T} q$.*

Proof. The first part is straightforward with the last two lemmas. For the second part, if $q' \preceq_\mathsf{T} q$ we have with Lemma 4 that $\overline{q'}$ is semi-decidable in q, contradicting Lemma 14. ◀

This lemma summarises the mentioned fact Turing jumps are strictly higher in terms of Turing reducibility. We next establish the announced connection between relative semi-decidability, (many-one) reducibility, and Turing jumps.

Lemma 18. *If $\mathcal{S}_q(p)$, then $p \preceq_m q^\circ$.*

Proof. Let F be computable and $\forall x.\, px \leftrightarrow F\hat{q}x\star$. Using Corollary 8, we have c such that $\Xi_c\hat{q}x \leftrightarrow F\hat{q}x\star$. Now take $\lambda x.\, \langle c, x\rangle$ as reduction. ◀

Lemma 19. *If $p \preceq_m q'$, then $\mathcal{S}_q(p)$.*

Proof. Let f be the many-one reduction. It then suffices to prove that $\lambda x.\, q'(fx)$ is semi-decidable in q. Take $\lambda Rco.\, \Xi_{fc}\, R\, (fc)$. ◀

Corollary 20. *$\mathcal{S}_q(p)$ if and only if $p \preceq_m q'$.*

Corollary 21. *If $p \preceq_\mathsf{T} q$, then $p' \preceq_m q'$.*

We now define iterated Turing jumps from a predicate q:

$$q^{(0)} := q \qquad\qquad q^{(n+1)} := (q^{(n)})'$$

In textbooks, one usually uses the empty predicate $\emptyset$ as basis, since it is Turing equivalent to any other decidable predicate. In the context of many-one reductions, however not all decidable predicates are equivalent: only non-trivial ones are. To obtain a more uniform treatment that does not have to consider special cases for decidable predicates, we use as basis the predicate that has sole element 0:

$$\mathbb{0} := \lambda x.\, x = 0$$

We also remark that, working explicitly with the previously discussed axiom EPF would allow us to prove that $\mathcal{K}$ is many-one equivalent to $\mathbb{0}^{(1)}$.

7 The Arithmetical Hierarchy

The arithmetical hierarchy was developed independently by Kleene [30] and Mostowski [35], using models of computation to define the 0 level. In line with the rest of this paper, we here define a synthetic variant relying on type-theoretic functions. In Section 10 we discuss the connection to an alternative definition where the 0 level uses quantifier-free logical formulas of a formal first-order syntax. Informally, a predicate $p\colon \mathbb{N}^k \to \mathbb{P}$ is in Σ_n if

$$\forall v.\, pv \leftrightarrow \exists x_1.\forall x_2.\ldots.Qx_n.\, f([x_n,\ldots,x_2,x_1] +\!\!+ v) = \mathsf{true}$$

where Q_n is either $\exists$ or $\forall$ depending on n being odd or even, i.e. if p can be characterised by a first-order formula with n quantifier alternations before a boolean function application, starting with an existential quantification. The definition of Π_n is dual. Formally, we use mutually defined inductive predicates:

$$\frac{\forall v : \mathbb{N}^k.\, pv \leftrightarrow fv = \mathsf{true}}{\Sigma_0^k p} \qquad \frac{\Pi_n^{k+1} q \qquad \forall v : \mathbb{N}^k.\, pv \leftrightarrow \exists x.\, q(x :: v)}{\Sigma_{n+1}^k p}$$

$$\frac{\forall v : \mathbb{N}^k.\, pv \leftrightarrow fv = \mathsf{true}}{\Pi_0^k p} \qquad \frac{\Sigma_n^{k+1} q \qquad \forall v : \mathbb{N}^k.\, pv \leftrightarrow \forall x.\, q(x :: v)}{\Pi_{n+1}^k p}$$

One usually defines $\Delta_n^k p := \Sigma_n^k p \wedge \Pi_n^k p$, but we do not use this notion technically. From now on, we leave out the arity k since it is always clear from context. In the remainder of this section we collect closure properties that hold constructively, in the next section we continue with some non-constructive closure properties.

The hierarchy treats predicates extensionally:

Lemma 22. *Whenever* $\forall v.\ p_1 v \leftrightarrow p_2 v$*, then* $\Sigma_n p_1$ *implies* $\Sigma_n p_2$ *and* $\Pi_n p_1$ *implies* $\Pi_n p_2$*.*

The 1 level of the hierarchy can be characterised using semi-decidability:

Lemma 23. *A predicate* p *is semi-decidable if and only if* $\Sigma_1 p$*.*

Proof. From left to right, let p be semi-decidable, i.e. $\forall v : \mathbb{N}^k.\ pv \leftrightarrow fv \triangleright \star$. The result follows using Lemma 1 (**1**), because $fv \triangleright \star \leftrightarrow \exists n.\ \epsilon\, f\, n\, v \star = \mathsf{true}$. From right to left, we use unbounded search μ from Lemma 1 (**2**). ◀

By straightforward induction, we also obtain that the hierarchy is cumulative:

Lemma 24. *If* $n \leq m$ *then* $\Sigma_n \subseteq \Sigma_m$ *as well as* $\Pi_n \subseteq \Pi_m$*.*

Furthermore, it is closed under many-one reductions:

Lemma 25. *If* $p_1 \preceq_m p_2$ *then* $\Sigma_n p_2$ *implies* $\Sigma_n p_1$ *and* $\Pi_n p_2$ *implies* $\Pi_n p_1$*.*

Proof. By mutual induction. The base cases are immediate. We prove that if $p_1 \preceq_m p_2$ via f, $\forall v.\ p_2 v \leftrightarrow \exists x.\ q(x :: v)$, and $\Pi_n q$, then $\Sigma_{n+1} p_1$, the other case is similar.

We have that $\forall v.\ p_1 v \leftrightarrow \exists x.\ q(x :: fv)$. By the induction hypothesis, it thus suffices to prove $(\lambda(x :: v).\ q(x :: fv)) \preceq_m q$, which is trivial. ◀

Note that in this proof, we use the notation $\lambda(x :: v) \ldots$, which defines a function taking as argument a non-empty vector, i.e. an argument of type X^{n+1}. We now show further closure properties of the levels, crucially making use of Lemma 25 and pairing $\langle \cdot, \cdot \rangle$.

Lemma 26. $\Sigma_n(\lambda v.\ \exists xy.\ p(x :: y :: v))$ *if* $\Pi_n p$*, and* $\Pi_n(\lambda v.\ \forall xy.\ q(x :: y :: v))$ *if* $\Sigma_n q$*.*

Corollary 27. $\Sigma_n p$ *implies* $\Sigma_n(\lambda v.\ \exists x.\ p(x :: v))$ *and* $\Pi_n q$ *implies* $\Pi_n(\lambda v.\ \forall x.\ q(x :: v))$*.*

By induction we then have:

Lemma 28. $\Sigma_n \subseteq \Pi_{n+1}$ *and* $\Pi_n \subseteq \Sigma_{n+1}$*.*

Lemma 29. *If* $\Sigma_n p_1$ *and* $\Sigma_n p_2$ *then* $\Sigma_n(\lambda v. p_1 v \wedge p_2 v)$*. If* $\Pi_n q_1$ *and* $\Pi_n q_2$ *then* $\Pi_n(\lambda v. q_1 v \wedge q_2 v)$*.*

Proof. By mutual induction. The base cases are easy.

We just show one inductive case, the other one is dual. Let $\forall v.\ p_1 v \leftrightarrow \exists x.\ q_1'(x :: v)$, $\forall v.\ p_2 v \leftrightarrow \exists y.\ q_2'(y :: v)$, and $\Pi_n q_1'$ as well as $\Pi_n q_2'$, then $\Sigma_{n+1}(\lambda v.\ p_1 v \wedge p_2 v)$.

Note that we have that $\lambda v.\ p_1 v \wedge p_2 v \leftrightarrow \exists xy.\ q_1'(x :: v) \wedge q_2'(y :: v)$. Using Lemma 26 and the induction hypothesis, it suffices to prove that $\Pi_n \lambda(x :: y :: v).\ q_1'(x :: v)$ and $\Pi_n \lambda(x :: y :: v).\ q_2'(x :: v)$, which follows from $\Pi_n q_1'$, $\Pi_n q_2'$, and Lemma 25. ◀

The following two lemmas have similarly technical proofs, we omit the details.

Lemma 30. *Let* $f : \mathbb{N} \to \mathbb{B}$*. If* $\Sigma_n p_1$ *and* $\Sigma_n p_2$*, then* $\Sigma_n(\lambda(x :: v).\ \mathbf{if}\ fx\ \mathbf{then}\ p_1 v\ \mathbf{else}\ p_2 v)$*. If* $\Pi_n q_1$ *and* $\Pi_n q_2$ *then* $\Pi_n(\lambda(x :: v).\ \mathbf{if}\ fx\ \mathbf{then}\ q_1 v\ \mathbf{else}\ q_2 v)$*.*

Lemma 31. *If* $\Sigma_n p$*, then* $\Sigma_n(\lambda(N :: v).\ \forall x < N.\ p(x :: v))$*. Moreover, if* $\Pi_n q$*, then* $\Pi_n(\lambda(N :: v).\ \forall x < N.\ q(x :: v))$*.*

We prove closure under disjunction for Σ_n now:

Lemma 32. *If* $\Sigma_n p_1$ *and* $\Sigma_n p_2$ *then* $\Sigma_n(\lambda v. p_1 v \vee p_2 v)$*.*

Proof. With $p_1 v \vee p_2 v \leftrightarrow \exists n.\ \mathbf{if}\ n = 0\ \mathbf{then}\ p_1 v\ \mathbf{else}\ p_2 v$ from Lemma 30 and corollary 27. ◀

The proof for Π_n requires classical logic. We thus only give it in the next section, where we introduce a fine-grained characterisation of classical logic for the arithmetical hierarchy, allowing to state a stronger theorem for closure of Π_n under disjunction.

8 An Arithmetical Hierarchy of Classical Axioms

The most common axiom to enable classical logical reasoning in constructive foundations is the law of excluded middle (LEM), or, equivalently, double negation elimination (DNE). Both LEM and DNE can be weakened to apply to certain propositions only. If restricted to apply to Σ_1 propositions only, one obtains the limited principles of omniscience (LPO) and Markov's principle (MP), respectively.

$$\mathsf{LEM} := \forall P{:}\,\mathbb{P}.\ P \vee \neg P \qquad \mathsf{LPO} := \forall f{:}\,\mathbb{N}{\to}\mathbb{B}.\ (\exists n.\ fn = \mathsf{true}) \vee \neg(\exists n.\ fn = \mathsf{true})$$

$$\mathsf{DNE} := \forall P : \mathbb{P}.\ \neg\neg P \to P \qquad \mathsf{MP} := \forall f{:}\,\mathbb{N}{\to}\mathbb{B}.\ \neg\neg(\exists n.\ fn = \mathsf{true}) \to (\exists n.\ fn = \mathsf{true})$$

LPO implies MP, but the converse is well-known to not be provable [9].

Akama, Berardi, Hayashi, and Kohlenbach [1] introduce relativisations of the law of excluded middle and double negation elimination to the arithmetical hierarchy, and prove that they are in relation as displayed in Figure 1. We prove these implications formally in CIC and using Coq, with the following definitions:

$$\Sigma_n\text{-LEM} := \forall k.\forall p{:}\,\mathbb{N}^k.\ \Sigma_n p \to \forall v.pv \vee \neg pv \qquad \Sigma_n\text{-DNE} := \forall k.\forall p{:}\,\mathbb{N}^k.\ \Sigma_n p \to \forall v.\neg\neg pv \to pv$$

$$\Pi_n\text{-LEM} := \forall k.\forall p{:}\,\mathbb{N}^k.\ \Pi_n p \to \forall v.pv \vee \neg pv \qquad \Pi_n\text{-DNE} := \forall k.\forall p{:}\,\mathbb{N}^k.\ \Pi_n p \to \forall v.\neg\neg pv \to pv$$

On the 0 and 1 levels these axioms have well-known connections to LPO and LEM:

Lemma 33. *The following hold*

1. Σ_0-LEM *holds constructively, and thus all* 0 *levels of the axioms,*
2. Σ_1-LEM $\leftrightarrow$ LPO*,*
3. Σ_1-DNE $\leftrightarrow$ MP*,*
4. Π_1-LEM $\leftrightarrow$ WLPO *with* $\mathsf{WLPO} := \forall f{:}\,\mathbb{N}{\to}\mathbb{B}.\ (\forall n.\ fn = \mathsf{false}) \vee \neg(\forall n.\ fn = \mathsf{false})$*,*
5. Π_1-DNE *holds constructively.*

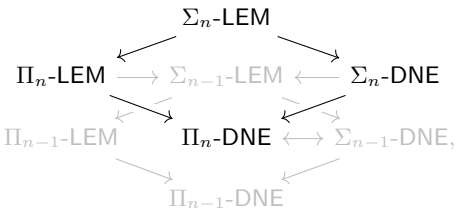


Figure 1 Arithmetical hierarchy of the law of excluded middle and related principles [1].

To prove the implications from Figure 1, we need that the arithmetical hierarchy is closed under complements and that Π_n is closed under disjunction (which holds constructively for Σ_n, see Lemma 32). We begin with the closures under complement:

Lemma 34. *If* $\Sigma_n p$ *and* Π_n-DNE*, then* $\Pi_n \overline{p}$*, and if* $\Pi_n p$ *and* Σ_n-DNE*, then* $\Sigma_n \overline{p}$*.*

Proof. By mutual induction. The base cases hold constructively.

For the first inductive case, we prove that if $\forall v.\ pv \leftrightarrow \exists x.\ p'(x :: v)$, $\Sigma_n p'$, and Σ_{n+1}-DNE, then $\Sigma_{n+1}\overline{p}$. By the inductive hypothesis we have $\Pi_n \overline{p}$ provided Π_n-DNE, which is a consequence of Σ_{n+1}-DNE. It thus suffices to prove that $\forall v.\ \neg pv \leftrightarrow \forall x.\ \neg p'(x :: v)$, which holds constructively.

For the other case, we prove that if $\forall v.\ pv \leftrightarrow \forall x.\ p'(x :: v)$, $\Sigma_n p'$, and Σ_{n+1}-DNE, then $\Sigma_{n+1}\overline{p}$. By the induction hypothesis we have $\Pi_n\overline{p}$ provided Π_n-DNE, which is a consequence of Σ_{n+1}-DNE. It thus suffices to prove that $\forall v.\ \neg pv \leftrightarrow \exists x.\ \neg p'(x :: v)$. The direction from right to left holds constructively. For the direction from left to right, assume $\neg pv$ and prove $\exists x.\ \neg p(x :: v)$. Since by the inductive hypothesis we have that $\Sigma_{n+1}(\lambda v.\ \exists x.\ \neg p'(x :: v)$ we can use Σ_{n+1}-DNE and it suffices to prove $\neg\neg\exists x.\ \neg p(x :: v)$, which follows constructively from $\neg pv$. ◀

It is straightforward to prove that Π_n is closed under disjunction assuming Π_n-LEM:

Lemma 35. *If* Π_n-LEM, $\Pi_n p$, *and* $\Pi_n q$ *then* $\Pi_n(\lambda v. pv \vee qv)$.

We now prove the implications from Figure 1.

Lemma 36.
1. Σ_n-LEM $\to$ Σ_n-DNE
2. Π_n-LEM $\to$ Π_n-DNE
3. Σ_n-DNE $\leftrightarrow$ Π_{n+1}-DNE
4. Π_{n+1}-LEM $\to$ Σ_n-LEM
5. Σ_n-LEM $\to$ Π_n-LEM
6. Σ_{n+1}-DNE $\to$ Σ_n-LEM

Proof. (1) and (2) are immediate, because in general $P \vee \neg P \to \neg\neg P \to P$. (3) is by induction. (4) follows by $\Sigma_n \subseteq \Pi_{n+1}$.

(5) has a more interesting proof: Assume Σ_n-LEM. By the previous implications, we can then use Σ_n-DNE and Π_n-DNE. Let $\Pi_n p$ and $v : \mathbb{N}^k$. We have to prove $pv \vee \neg pv$. By Lemma 34 and Σ_n-DNE, we have $\Sigma_n\overline{p}$. Using Σ_n-LEM we have $\neg pv \vee \neg\neg pv$. Using Π_n-DNE, we have $pv \vee \neg pv$.

For (6), assume Σ_{n+1}-DNE and $\Sigma_n p$. We prove $pv \vee \neg pv$ by applying Σ_{n+1}-DNE. Because Σ_{n+1} is closed under disjunction by Lemma 32, it suffices to prove that p is in Σ_{n+1}, which is trivial, and that $\overline{p}$ is, which follows from Lemma 34. ◀

Note that the converses of the implications are not provable: For (1), MP does not imply LPO (see e.g. [21]). For (2), Π_1-LEM is WLPO, which is not provable, and Π_1-DNE is provable. For (4), since Σ_0-LEM is provable, but Π_1-LEM is WLPO. For (5), since (at level 1) WLPO is strictly weaker than LPO [21]. We furthermore cannot prove that Σ_1-DNE implies Π_1-LEM, because MP does not imply WLPO. For (6), since Σ_0-LEM is provable, but Σ_1-DNE is MP.

It seems that for both closure properties we proved the assumptions are stronger than necessary. To prove closure under disjunction, it would suffice to assume DNE for disjunctions where both sides of the disjunct are Π_n formulas [1], but we avoid introducing this axiom.

We conjecture that for the purpose of proving that the arithmetical hierarchy is closed under negation, Σ_n-DNE is also strictly stronger than necessary. This is because at level 1, the theorem is equivalent to an axiom which seems to be weaker than MP (i.e. Σ_1-DNE).

Lemma 37. $\Pi_1 p \to \Sigma_1\overline{p}$ *iff* $\forall f : \mathbb{N}\to\mathbb{B}.\exists g : \mathbb{N}\to\mathbb{B}.\ \neg\neg(\exists n.\ fn = \mathsf{true}) \leftrightarrow (\exists n.\ gn = \mathsf{true})$.

This principle can be seen as an "anonymised" Markov's principle. It is an obvious consequence of MP (with $g := f$), but it seems to be strictly weaker. We are not aware of this axiom appearing in the literature, and we conjecture it to be non-provable. It can also be seen as the level 1 instance of closure under double negation for the hierarchy.

We say that the arithmetical hierarchy is closed under double negations at level n if $\Sigma_n p$, then $\Sigma_n\overline{\overline{p}}$, and if $\Pi_n p$, then $\Pi_n\overline{\overline{p}}$. And, as before, it is closed under complements at level n if $\Sigma_n p$, then $\Pi_n\overline{p}$, and if $\Pi_n p$, then $\Sigma_n\overline{p}$.

Lemma 38. *If the arithmetical hierarchy is closed under complements at level n, it is closed under double negations at level n.*

For the converse, we need to assume that the hierarchy is closed under double negations for all levels $m \leq n$, because closure at level $n+1$ seems not to imply closure at level n. Furthermore, the proof seems to require Π_n-DNE – a principle strictly weaker than Σ_n-DNE which we used to prove closure under negation.

Lemma 39. *Given Π_n-DNE, if the arithmetical hierarchy is closed under double negation at all levels $m \leq n$, it is closed under complements at level n.*

We conjecture that above level 1, this is not an equivalence, i.e. that some axiom potentially weaker than Π_n-DNE is needed.

9 Post's Theorem

By composition of previous results, we now derive the statements comprising Post's theorem. We first explicitly capture the connection of relative semi-decidability and the arithmetical hierarchy in the next two lemmas.

Lemma 40. *Assume Π_n-LEM. If $\Sigma_{n+1}p$, then p is semi-decidable relative to some q in Π_n.*

Proof. Let Σ_{n+1}, i.e. there is q in Π_n such that $pv \leftrightarrow \exists x.q(x :: v)$. We show that p is semi-decidable in q by linearly searching for x, i.e. pick

$$FRvo := \exists x.R\ (x :: v)\ \mathsf{true} \wedge \forall x' < x.R\ (x' :: v)\ \mathsf{false}$$

which is oracle-computable by Lemma 2 (**9**). We need to prove that $(\exists x.q(x :: v)) \leftrightarrow F\hat{q}v\star$. The direction from right to left is immediate. For the direction from left to right we have to prove that given x with $q(x :: v)$, i.e. $\hat{q}(x :: v)\mathsf{true}$, there is a *least* x such that $\hat{q}(x :: v)\mathsf{true}$. This follows from Π_n-LEM and $\Pi_n q$. ◀

Lemma 41. *Assume Σ_n-DNE. If p is semi-decidable relative to some q in Π_n, then $\Sigma_{n+1}p$.*

Proof. Let $\forall v.\ pv \leftrightarrow F\hat{q}v\star$ for an oracle-computable F. This means we have τ such that

$$\forall v.\ pv \leftrightarrow \exists qs\ as.\ \tau v\ ;\hat{q} \vdash qs\ ;as \wedge \tau\ v\ as \triangleright \mathsf{out}\ \star\ .$$

Note that $\lambda(as :: v).\ \tau\ v\ as \triangleright \mathsf{out}\ \star$ is in $\Sigma_1 \subseteq \Sigma_{n+1}$, because it is trivially semi-decidable, which makes Lemma 23 applicable. Using Corollary 27, it then suffices to prove that $\lambda(qs :: as :: v).\ \tau v\ ;\hat{q} \vdash qs\ ;as$ is in Σ_{n+1}.

To do so, we prove that

$$\tau v\ ;\hat{q} \vdash qs\ ;as \leftrightarrow \forall n < |as|.\ \tau\ v\ (as\ \lfloor_n) \triangleright \mathsf{ask}\ qs_n \wedge \hat{q}(qs_n)(as_n)$$

where $as\ \lfloor_n$ is the list with the first n elements of as, and qs_n and as_n are the n-th element of qs and as, respectively. The direction from left to right is by induction on the interrogation, from right to left by induction on as.

Using Lemma 31 and once again that $\triangleright$ is semi-decidable, it suffices to prove that $\lambda(y :: b :: [\,]).\ \hat{q}yb$ is in Σ_{n+1}. Using Lemma 30, we have to prove that both q and its complement are in Σ_{n+1}. The former holds because q is in Π_n and Lemma 28. The latter holds because q is in Π_n, and thus its complement is in Σ_n using Σ_n-DNE and Lemma 34, and thus in Σ_{n+1} using Lemma 24. ◀

Incidentally, apart from the 0 case, the inductive structure of all remaining proofs for Theorem 43 is unchanged as long as the following holds:

Lemma 42. $\Sigma_0\mathbb{0}$

Then finally, Post's theorem can be stated as follows:

Theorem 43. *The following hold assuming* Σ_n-LEM*:*

1. $\Sigma_{n+1}p$ *if and only if* $\mathcal{S}_q(p)$ *for some* q *in* Π_n*,*
2. $\Sigma_{n+1}p$ *if and only if* $\mathcal{S}_q(p)$ *for some* q *in* Σ_n*,*
3. $\Sigma_n\mathbb{0}^{(n)}$*,*
4. *if* $\Sigma_n p$ *then* $p \preceq_m \mathbb{0}^{(n)}$*, so in particular* $p \preceq_{\mathsf{T}} \mathbb{0}^{(n)}$*,*
5. $\Sigma_{n+1}p$ *if and only if* $\mathcal{S}_{\mathbb{0}^{(n)}}(p)$*.*

Note that, in light of Lemma 42, by further relativising the 0 level of the arithmetical hierarchy to relative decidability one can obtain a relativised form of Post's theorem from an arbitrary base predicate, that does not even necessarily have to be arithmetical.

10 The Syntactic Arithmetical Hierarchy

We have defined the arithmetical hierarchy in a synthetic way, by defining the 0 level using boolean functions rather than predicates decidable in a model of computation. Another natural definition is syntactic, purely in terms of first-order logic, where the 0 level is characterised by quantifier-free formulas, and where the Δ_1 class can alternatively be characterised by predicates that are decidable in a model of computation.

We define this syntactic arithmetical hierarchy now, show that it is included in the previously defined synthetic arithmetical hierarchy, and show that the converse inclusion is equivalent to the well-known constructive axiom CT, a strengthening of the axiom EPF.

As a basis, we employ the Coq library of first-order logic [27, 25] and recall the basic framework. We use a de-Bruijn representation of first-order formulas φ over the term language of arithmetic (i.e. with constant 0, unary successor function S, and binary operation symbols for addition and multiplication). We use a type of variables V on paper, which we implement using de Bruijn indices as $V := \mathbb{N}$ in Coq.

$$t : \mathsf{term} ::= x \mid n \mid t_1 \dot{+} t_2 \mid t_1 \dot{\times} t_2 \qquad x{:}V, n{:}\mathbb{N}$$

$$\varphi : \mathsf{form} ::= \dot{\bot} \mid t_1 \dot{=} t_2 \mid \varphi_1 \dot{\wedge} \varphi_2 \mid \varphi_1 \dot{\vee} \varphi_2 \mid \varphi_1 \dot{\to} \varphi_2 \mid \dot{\forall}\varphi \mid \dot{\exists}\varphi$$

A formula is quantifier-free if it does not use the constructors $\dot{\exists}$ and $\dot{\forall}$.

We furthermore use a Tarski-style satisfaction predicate $\rho \vDash \varphi$ for $\rho < : V \to \mathbb{N}$ being an assignment from de Bruijn indices to natural numbers which maps the terms to their respective interpretation of natural numbers, and formulas to their respective interpretations in the meta-logic.

$$[\![x]\!]_\rho := \rho x \qquad [\![n]\!]_\rho := n \qquad [\![t_1 \dot{+} t_2]\!]_\rho := [\![t_1]\!]_\rho + [\![t_2]\!]_\rho \qquad [\![t_1 \dot{\times} t_2]\!]_\rho := [\![t_1]\!]_\rho \cdot [\![t_2]\!]_\rho$$

$$\rho \vDash \dot{\bot} := \bot \qquad \rho \vDash t_1 \dot{=} t_2 := [\![t_1]\!]_\rho = [\![t_2]\!]_\rho \qquad \rho \vDash \varphi_1 \dot{\wedge} \varphi_2 := (\rho \vDash \varphi_1) \wedge (\rho \vDash \varphi_2)$$

$$\rho \vDash \varphi_1 \dot{\vee} \varphi_2 := (\rho \vDash \varphi_1) \wedge (\rho \vDash \varphi_2) \qquad \rho \vDash \varphi_1 \dot{\to} \varphi_2 := (\rho \vDash \varphi_1) \to (\rho \vDash \varphi_2)$$

$$\rho \vDash \dot{\forall}\varphi := \forall d.(d; \rho) \vDash \varphi \qquad \rho \vDash \dot{\exists}\varphi := \exists d.(d; \rho) \vDash \varphi$$

We then define when a formula φ is $\dot{\mathbf{\Sigma}}_n$ or respectively $\dot{\mathbf{\Pi}}_n$.[2]

$$\frac{\varphi \text{ is quantifier-free}}{\dot{\mathbf{\Sigma}}_n\varphi} \qquad \frac{\dot{\mathbf{\Pi}}_n\varphi}{\dot{\mathbf{\Sigma}}_{n+1}\dot{\exists}\varphi} \qquad \frac{\dot{\mathbf{\Sigma}}_{n+1}\varphi}{\dot{\mathbf{\Sigma}}_{n+1}\dot{\exists}\varphi}$$

$$\frac{\varphi \text{ is quantifier-free}}{\dot{\mathbf{\Pi}}_n\varphi} \qquad \frac{\dot{\mathbf{\Sigma}}_n\varphi}{\dot{\mathbf{\Pi}}_{n+1}\dot{\forall}\varphi} \qquad \frac{\dot{\mathbf{\Pi}}_{n+1}\varphi}{\dot{\mathbf{\Pi}}_{n+1}\dot{\forall}\varphi}$$

Based on this, we define

$$\dot{\mathbf{\Sigma}}^k_n p := \exists\varphi.\ \dot{\mathbf{\Sigma}}_n\varphi \land \forall v{:}\,\mathbb{N}^k.\ pv \leftrightarrow \rho_v \vDash \varphi \qquad \dot{\mathbf{\Pi}}^k_n p := \exists\varphi.\ \dot{\mathbf{\Pi}}_n\varphi \land \forall v{:}\,\mathbb{N}^k.\ pv \leftrightarrow \rho_v \vDash \varphi$$

where ρ_v is a function mapping i to the i-th value in v (and 0 if v is not long enough). As before, we leave out k from here on.

Lemma 44. *Satisfaction of quantifier-free formulas is decidable.*

Lemma 45. *The syntactic arithmetical hierarchy is included in the synthetic arithmetical hierarchy.*

Proof. We prove by mutual induction that

$$(\dot{\mathbf{\Sigma}}_n\varphi \to \Sigma_n(\lambda v.\ \rho_v \vDash \varphi)) \land (\dot{\mathbf{\Pi}}_n\varphi \to \Pi_n(\lambda v.\ \rho_v \vDash \varphi)).$$

The base cases follows by the last lemma. The cases of adding $\dot{\exists}$ to a $\dot{\mathbf{\Pi}}_n$ formula, or $\dot{\forall}$ to a $\dot{\mathbf{\Sigma}}_n$ formula follow by definition of Σ and Π. The cases of adding $\dot{\exists}$ to a $\dot{\mathbf{\Sigma}}_n$ formula, or $\dot{\forall}$ to a $\dot{\mathbf{\Pi}}_n$ formula follow by Corollary 27. ◀

The reverse direction is not provable without axioms. In fact, it is equivalent to the axiom Church's thesis (CT), stating that all functions are computable in a model of computation. The axiom EPF we have used is a direct consequence of CT [11].

Let $\phi^n_c x$ be the execution of the c-th μ-recursive function according to an enumeration, on input x for n steps.

$$\mathsf{CT} := \forall f{:}\,\mathbb{N}{\to}\mathbb{N}.\exists c{:}\,\mathbb{N}.\ \forall x.\exists n.\ \phi^n_c x = 1 + (fx)$$

Lemma 46. *Assuming* CT, $\dot{\mathbf{\Sigma}}_1(\lambda v.\ fv = \mathsf{true})$.

Proof. See Hermes and Kirst [22] and Kirst and Peters [28]. ◀

Lemma 47. *Assuming* CT, $\dot{\mathbf{\Pi}}_1(\lambda v.\ gv = \mathsf{true})$.

Proof. Let g be given. Using the last lemma with $fv := \neg_{\mathbb{B}} gv$ we have that $\dot{\mathbf{\Sigma}}_1(\lambda v.\neg_{\mathbb{B}} gv = \mathsf{true})$. The claim follows by proving that $\dot{\mathbf{\Sigma}}_1 p \to \dot{\mathbf{\Pi}}_n(\lambda v.\ \neg pv)$ in general, which holds constructively without axioms. ◀

Corollary 48. *Assuming* CT, $\Sigma_1 \subseteq \dot{\mathbf{\Sigma}}_1$ *and* $\Pi_1 \subseteq \dot{\mathbf{\Pi}}_1$.

[2] We use boldface fonts with a dot here to distinguish from the synthetic arithmetical hierarchy as clearly as possible. Note that there is no connection to the so-called boldface and lightface pointclasses from descriptive set theory, which are also based on hierarchies of formulas classified via quantifier alternations.

Note that for $n = 0$, the hiearchies are not equivalent, because not every decidable predicate can be presented as quantifier-free formula[3] This however does not affect Post's theorem, since it only talks about $n > 0$.

Lemma 49. *Assuming* CT*, the synthetic arithmetical hierarchy is included in the syntactic arithmetical hierarchy on every level* $n > 0$*.*

Proof. By induction, using the last lemma. ◀

To avoid reintroducing classical axioms for the syntactic arithmetical hierarchy, we re-state Post's theorem by using classical logic in general:

▶ **Theorem 50.** *The following hold assuming the law of excluded middle and* CT*:*

1. $\dot{\mathbf{\Sigma}}_{n+1} p$ *if and only if* $\mathcal{S}_q(p)$ *for a* q *in* $\dot{\mathbf{\Pi}}_n$*,*
2. $\dot{\mathbf{\Sigma}}_{n+1} p$ *if and only if* $\mathcal{S}_q(p)$ *for a* q *in* $\dot{\mathbf{\Sigma}}_n$*,*
3. $\dot{\mathbf{\Sigma}}_n \mathbb{0}^{(n)}$*,*
4. *if* $\dot{\mathbf{\Sigma}}_n p$ *then* $p \preceq_m \mathbb{0}^{(n)}$*, so in particular* $p \preceq_{\mathsf{T}} \mathbb{0}^{(n)}$*,*
5. $\dot{\mathbf{\Sigma}}_{n+1} p$ *if and only if* $\mathcal{S}_{\mathbb{0}^{(n)}}(p)$*.*

▶ **Lemma 51.** *If the synthetic arithmetical hierarchy is included in the syntactic arithmetical hierarchy on level* $1 > 0$*, then* CT *holds.*

Proof. Assume $\Sigma_1 \subseteq \dot{\mathbf{\Sigma}}_1$. The axiom that semi-decidable predicates are semi-decided by a μ-recursive function is equivalent to CT [12]. Let $p : \mathbb{N} \to \mathbb{P}$ be semi-decidable. Then it is in Σ_1. By assumption, it then is in $\dot{\mathbf{\Sigma}}_1$. Thus, it can be semi-decided by a μ-recursive function. ◀

Lemma 52. *Given* LEM*, whenever a predicate* p *is definable via a first-order formula* φ *one can compute* n *and either a proof of* $\dot{\mathbf{\Sigma}}_n p$ *or* $\dot{\mathbf{\Pi}}_n p$*.*

Note that the assumption of LEM could be weakened to $\dot{\mathbf{\Sigma}}_n$-LEM [20, 19].

11 Conclusion

In this paper, we use the definition of oracle computability of [15] mostly abstractly, i.e. through the closure properties in Lemma 2 as interface. While the invariants for the constructions underlying these properties are intricate and often tedious, utilising the properties themselves is straightforward. The only explicit uses of the underlying computation trees are Lemma 41 and the construction of the enumerators Ξ and χ from EPF. The proof of Post's theorem can then be seen as a sanity check for our synthetic definition of Turing reducibility: It agrees with analytic Turing reducibility on arithmetical predicates.

The Coq mechanisation, contributing roughly 3k lines of code (2k for the Kleene-Post and Post's theorem, 1k for the syntactic arithmetical hierarchy) to the Coq library of synthetic computability, has proven beneficial besides the goals of formalising the foundations of theoretical computer science and a constructive reverse analysis: Having a mechanisation allows for quickly changing definitions without the need for extensive manual re-checking. We have utilised this e.g. when changing the base of the hierarchy to $\mathbb{0}$ instead of $\emptyset$, which was almost automatic. Furthermore, identifying and tracking classical assumptions manually would be cumbersome, and is almost for free using a proof assistant.

[3] The situation would be different if we would allow for bounded quantification on the 0 level of the hierarchy, but we syntactically disallow any quantification to appear.

Bauer [4] and Swan [47] suggest definitions of oracle computability and Turing reducibility in unpublished work. Our definition of oracle computability is stronger than Bauer's [15, Lemmas 2 and 42]. However, the resulting notion of Turing reducibility might still be the same – it hinges on the constructability of an enumerator for Bauer's notion. Additionally, our definition can be adapted to Swan's setting in univalent type theory, potentially implying his definition of Turing computability. Vice versa we expect the implication to be unprovable because it would require a form of double negation elimination stronger than MP, which seemingly implies LPO and thus is inconsistent in univalent synthetic computability. Swan's definition cannot be exactly reproduced in our setting, since it requires higher inductive types. A version of CIC with higher inductives but without univalence could be suitable.

For future work, it first would be interesting to frame more results from the 1954 Kleene-Post paper, e.g. that there are countably many incomparable Turing degrees with the order structure of the rationals, in our setting of synthetic computability. Secondly, we would like to carry out the full analysis of what classical axioms are implied by, and thus equivalent to, Post's theorem. Lastly, it would be intriguing to give a synthetic solution to Post's problem [40], e.g. via the Friedberg-Muchnik theorem and the priority method [18, 36].

References

1 Yohji Akama, Stefano Berardi, Susumu Hayashi, and Ulrich Kohlenbach. An arithmetical hierarchy of the law of excluded middle and related principles. In *19th IEEE Symposium on Logic in Computer Science (LICS 2004), 14-17 July 2004, Turku, Finland, Proceedings*, pages 192–201. IEEE Computer Society, 2004. doi:10.1109/LICS.2004.1319613.

2 Andrej Bauer. First steps in synthetic computability theory. *Electronic Notes in Theoretical Computer Science*, 155:5–31, 2006. doi:10.1016/j.entcs.2005.11.049.

3 Andrej Bauer. On fixed-point theorems in synthetic computability. *Tbilisi Mathematical Journal*, 10(3):167–181, 2017. doi:10.1515/tmj-2017-0107.

4 Andrej Bauer. Synthetic mathematics with an excursion into computability theory (slide set). *University of Wisconsin Logic seminar*, 2020. URL: http://math.andrej.com/asset/data/madison-synthetic-computability-talk.pdf.

5 Vasco Brattka, Matthew Hendtlass, and Alexander P. Kreuzer. On the uniform computational content of computability theory. *Theory of Computing Systems*, 61(4):1376–1426, August 2017. doi:10.1007/s00224-017-9798-1.

6 S Barry Cooper. *Computability theory*. CRC Press, 2003. doi:10.1201/9781315275789.

7 Thierry Coquand. Metamathematical investigations of a calculus of constructions. Technical Report RR-1088, INRIA, 1989. URL: https://hal.inria.fr/inria-00075471.

8 Thierry Coquand and Gérard P Huet. The calculus of constructions. *Information and Computation*, 76(2/3):95–120, 1988. doi:10.1016/0890-5401(88)90005-3.

9 Hannes Diener. Constructive Reverse Mathematics. *arXiv:1804.05495 [math]*, 2020. arXiv:1804.05495.

10 Hannes Diener and Hajime Ishihara. Bishop-style constructive reverse mathematics. In *Theory and Applications of Computability*, pages 347–365. Springer International Publishing, 2021. doi:10.1007/978-3-030-59234-9_10.

11 Yannick Forster. Church's Thesis and Related Axioms in Coq's Type Theory. In Christel Baier and Jean Goubault-Larrecq, editors, *29th EACSL Annual Conference on Computer Science Logic (CSL 2021)*, volume 183 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 21:1–21:19, Dagstuhl, Germany, 2021. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.CSL.2021.21.

12 Yannick Forster. *Computability in Constructive Type Theory*. PhD thesis, Saarland University, 2021. doi:10.22028/D291-35758.

13 Yannick Forster. Parametric Church's Thesis: Synthetic computability without choice. In *International Symposium on Logical Foundations of Computer Science*, pages 70–89. Springer, 2022. doi:10.1007/978-3-030-93100-1_6.

14 Yannick Forster and Felix Jahn. Constructive and Synthetic Reducibility Degrees: Post's Problem for Many-One and Truth-Table Reducibility in Coq. In Bartek Klin and Elaine Pimentel, editors, *31st EACSL Annual Conference on Computer Science Logic (CSL 2023)*, volume 252 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 21:1–21:21, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.CSL.2023.21.

15 Yannick Forster, Dominik Kirst, and Niklas Mück. Oracle computability and turing reducibility in the calculus of inductive constructions, 2023. arXiv:2307.15543.

16 Yannick Forster, Dominik Kirst, and Gert Smolka. On synthetic undecidability in Coq, with an application to the Entscheidungsproblem. In *Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs – CPP 2019*. ACM Press, 2019. doi:10.1145/3293880.3294091.

17 Yannick Forster, Fabian Kunze, and Nils Lauermann. Synthetic Kolmogorov Complexity in Coq. In June Andronick and Leonardo de Moura, editors, *13th International Conference on Interactive Theorem Proving (ITP 2022)*, volume 237 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:19, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ITP.2022.12.

18 R. M. Friedberg. Two recursively enumerable sets of incomparable degrees of unsovlability (solution of Post's problem), 1944. *Proceedings of the National Academy of Sciences*, 43(2):236–238, February 1957. doi:10.1073/pnas.43.2.236.

19 Makoto Fujiwara and Taishi Kurahashi. Prenex normal form theorems in semi-classical arithmetic. *The Journal of Symbolic Logic*, 86(3):1124–1153, 2021. doi:10.1017/jsl.2021.47.

20 Makoto Fujiwara and Taishi Kurahashi. Prenex normalization and the hierarchical classification of formulas, 2023. arXiv:2302.11808.

21 Matt Hendtlass and Robert Lubarsky. Separating fragments of WLEM, LPO, and MP. *The Journal of Symbolic Logic*, 81(4):1315–1343, 2016. doi:10.1017/jsl.2016.38.

22 Marc Hermes and Dominik Kirst. An analysis of Tennenbaum's theorem in constructive type theory. In Amy P. Felty, editor, *7th International Conference on Formal Structures for Computation and Deduction, FSCD 2022, August 2-5, 2022, Haifa, Israel*, volume 228 of *LIPIcs*, pages 9:1–9:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.FSCD.2022.9.

23 J. Martin E. Hyland. The effective topos. In *The L. E. J. Brouwer Centenary Symposium, Proceedings of the Conference held in Noordwijkerhout*, pages 165–216. Elsevier, 1982. doi:10.1016/s0049-237x(09)70129-6.

24 Hajime Ishihara. Reverse mathematics in Bishop's constructive mathematics. *Philosophia Scientae*, CS 6:43–59, September 2006. doi:10.4000/philosophiascientiae.406.

25 Dominik Kirst. *Mechanised Metamathematics: An Investigation of First-Order Logic and Set Theory in Constructive Type Theory*. PhD thesis, Saarland University, 2022. doi:10.22028/D291-39150.

26 Dominik Kirst, Yannick Forster, and Niklas Mück. Synthetic Versions of the Kleene-Post and Post's Theorem. 28th International Conference on Types for Proofs and Programs (TYPES 2022), 2022. URL: https://types22.inria.fr/files/2022/06/TYPES_2022_paper_65.pdf.

27 Dominik Kirst, Johannes Hostert, Andrej Dudenhefner, Yannick Forster, Marc Hermes, Mark Koch, Dominique Larchey-Wendling, Niklas Mück, Benjamin Peters, Gert Smolka, and Dominik Wehr. A Coq library for mechanised first-order logic. In *Coq Workshop*, 2022. URL: https://coq-workshop.gitlab.io/2022/abstracts/Coq2022-01-01-first-order-logic.pdf.

28 Dominik Kirst and Benjamin Peters. Gödel's theorem without tears – Essential incompleteness in synthetic computability. In Bartek Klin and Elaine Pimentel, editors, *31st EACSL Annual Conference on Computer Science Logic, CSL 2023, February 13-16, 2023, Warsaw, Poland*, volume 252 of *LIPIcs*, pages 30:1–30:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.CSL.2023.30.

29 S. C. Kleene. Countable functionals. *Journal of Symbolic Logic*, 27(3):81–100, 1959. doi:10.2307/2964658.

30 Stephen C. Kleene. Recursive predicates and quantifiers. *Transactions of the American Mathematical Society*, 53(1):41–73, 1943. doi:10.1090/S0002-9947-1943-0007371-8.

31 Steven C. Kleene and Emil L. Post. The upper semi-lattice of degrees of recursive unsolvability. *The Annals of Mathematics*, 59(3):379, May 1954. doi:10.2307/1969708.

32 Georg Kreisel. Interpretation of analysis by means of constructive functionals of finite types. In A. Heyting, editor, *Constructivity in Mathematics*, pages 101–128. North-Holland Pub. Co., 1959.

33 Georg Kreisel. Mathematical logic. *Lectures in modern mathematics*, 3:95–195, 1965. doi:10.2307/2315573.

34 Andrei A. Markov. The theory of algorithms. *Trudy Matematicheskogo Instituta Imeni VA Steklova*, 42:3–375, 1954. doi:10.1007/978-94-017-3477-6.

35 Andrzej Mostowski. On definable sets of positive integers. *Fundamenta Mathematicae*, 34:81–112, 1947. doi:10.4064/fm-34-1-81-112.

36 Albert Abramovich Muchnik. On strong and weak reducibility of algorithmic problems. *Sibirskii Matematicheskii Zhurnal*, 4(6):1328–1341, 1963.

37 Niklas Mück. *The Arithmetical Hierarchy, Oracle Computability, and Post's Theorem in Synthetic Computability*. Bachelor's thesis, Saarland University, 2022. URL: https://ps.uni-saarland.de/~mueck/bachelor.php.

38 Piergiorgio Odifreddi. *Classical recursion theory: The theory of functions and sets of natural numbers*. Elsevier, 1992.

39 Christine Paulin-Mohring. Inductive definitions in the system Coq rules and properties. In *International Conference on Typed Lambda Calculi and Applications*, pages 328–345. Springer, 1993. doi:10.1007/BFb0037116.

40 Emil L. Post. Recursively enumerable sets of positive integers and their decision problems. *bulletin of the American Mathematical Society*, 50(5):284–316, 1944. doi:10.1090/S0002-9904-1944-08111-1.

41 Emil L. Post. Degrees of recursive unsolvability – Preliminary report. In *Bulletin of the American Mathematical Society*, volume 54:7, pages 641–642. American Mathematical Society (AMS), 1948.

42 Fred Richman. Church's thesis without tears. *The Journal of symbolic logic*, 48(3):797–803, 1983. doi:10.2307/2273473.

43 Hartley Rogers. *Theory of Recursive Functions and Effective Computability*. MIT Press, Cambridge, MA, USA, 1987.

44 Sam Sanders. Refining the taming of the reverse mathematics zoo. *Notre Dame Journal of Formal Logic*, 59(4), January 2018. doi:10.1215/00294527-2018-0015.

45 Robert I. Soare. *Recursively enumerable sets and degrees: A study of computable functions and computably generated sets*. Springer Science & Business Media, 1999.

46 Andrew Swan and Taichi Uemura. On Church's thesis in cubical assemblies. *arXiv preprint*, 2019. arXiv:1905.03014.

47 Andrew W Swan. Oracle modalities. *Second International Conference on Homotopy Type Theory (HoTT 2023)*, 2023. URL: https://hott.github.io/HoTT-2023/abstracts/HoTT-2023_abstract_35.pdf.

48 The Coq Development Team. The Coq proof assistant, June 2023. doi:10.5281/zenodo.8161141.

49 Jaap van Oosten. Partial combinatory algebras of functions. *Notre Dame Journal of Formal Logic*, 52(4):431–448, 2011. doi:10.1215/00294527-1499381.

A Many-Sorted Epistemic Logic for Chromatic Hypergraphs

Éric Goubault
LIX, CNRS, École Polytechnique, IP-Paris, Palaiseau Cedex, France

Roman Kniazev
LIX, CNRS, École Polytechnique, IP-Paris, Palaiseau Cedex, France
Université Paris-Saclay, ENS Paris-Saclay, CNRS, LMF, 91190, Gif-Sur-Yvette, France
Université Paris Cité, CNRS, IRIF, F-75013, Paris, France

Jérémy Ledent
Université Paris Cité, CNRS, IRIF, F-75013, Paris, France

Abstract

We propose a many-sorted modal logic for reasoning about knowledge in multi-agent systems. Our logic introduces a clear distinction between participating agents and the environment. This allows to express local properties of agents and global properties of worlds in a uniform way, as well as to talk about the presence or absence of agents in a world. The logic subsumes the standard epistemic logic and is a conservative extension of it. The semantics is given in chromatic hypergraphs, a generalization of chromatic simplicial complexes, which were recently used to model knowledge in distributed systems. We show that the logic is sound and complete with respect to the intended semantics. We also show a further connection of chromatic hypergraphs with neighborhood frames.

2012 ACM Subject Classification Theory of computation → Modal and temporal logics

Keywords and phrases Modal logics, epistemic logics, multi-agent systems, hypergraphs

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.30

Funding *Éric Goubault*: The author was partially supported by Agence de l'Innovation de Défense – AID – via Centre Interdisciplinaire d'Etudes pour la Défense et la Sécurité – CIEDS – (project 2021 – FARO).

Acknowledgements This work benefited from various discussions during Dagstuhl Seminar 23272.

1 Introduction

Epistemic logic is a modal logic that allows formal reasoning about knowledge and belief in multi-agent systems. At its core lies the *knowledge operator* denoted $K_a\varphi$, which means that "agent a knows that the formula φ holds", or simply "a knows φ" for short. Since the work of [16], epistemic logic has expanded in many directions, studying various operators such as common knowledge [11], distributed knowledge [6], as well as studying how knowledge evolves over time, as in public announcement logics [7], temporal epistemic logics [12], or dynamic epistemic logics [5]. A common feature of those approaches is that they rely on the classic "possible worlds" semantics of normal modal logics, based of Kripke structures. Indeed, a model for multi-agent epistemic logic $\mathsf{S5}_n$ usually consists of a set of *possible worlds* W, and for each agent a, an equivalence relation $\sim_a \subseteq W \times W$ called the *indistinguishability relation* of agent a. The intended semantics of such a model is that an agent a *knows* that a formula φ is true, then φ is true in every possible world that is indistinguishable from the real world for that agent. Formally, this means that the satisfaction relation is defined as follows, given a model M and a world $w \in W$ (thought of as the "real world"):

$$M, w \models K_a\varphi \quad \text{iff} \quad M, w' \models \varphi \text{ for every world } w' \in W \text{ such that } w \sim_a w' \tag{1}$$

A recent line of work [9, 4, 3, 10, 18, 8] has been developing a new notion of model for epistemic logic called *simplicial models*. This approach was closely inspired by connections with distributed computing, where simplicial complexes have been very successful in modeling various models of computation [13]. Compared to Hintikka's possible worlds semantics, this new approach represents a shift in perspective. Rather than focusing on the *worlds* (a.k.a. global states, in distributed computing terms) as the primary object of study, we instead focus on the agents' *points of view* about the world (a.k.a. local states). A possible world can then be defined as a set of *compatible* points of view, one for each agent.

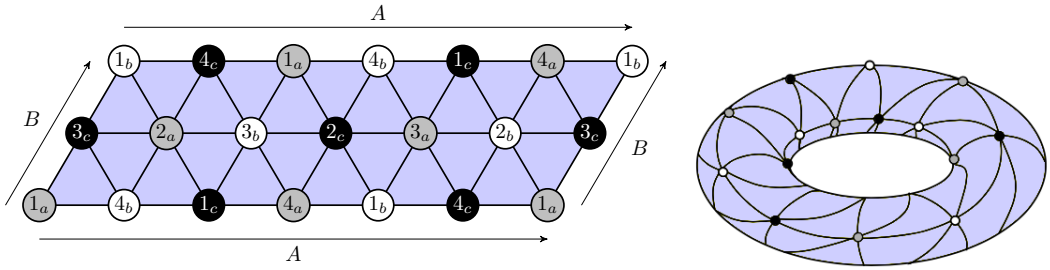
Worlds and views. To illustrate the distinction between possible worlds, and local views about the possible worlds, we use a classic example originally from distributed computing [14]. Since this paper is not concerned with distributed computing, we instead tell a story about a card game. Assume there are three agents, $\mathcal{A} = \{a, b, c\}$, and a deck of four cards, $\{1, 2, 3, 4\}$. We deal one card to each agent, and keep the remaining card hidden, so that each agent only knows its own card. Our goal is to model this (static) epistemic situation.

In the standard Kripke model semantics, the main step is to identify the possible worlds: each possible distribution of the cards constitutes a possible world. There are 24 such distributions, which we can denote by $W = \{123, 124, 132, 134, 142, 143, 213, 214, \ldots\}$, where for example the world "123" denotes the situation where agent a (resp. b, c) has received card number 1 (resp. 2, 3). To get a Kripke model, one must also define the indistinguishability relations for each agent. For instance, we have $123 \sim_a 132$. Indeed, from the point of view of agent a, who holds the same card number 1 in both of those worlds, these two worlds are indistinguishable. One can check that $\sim_a$ defined in this way is indeed an equivalence relation, with four equivalence classes, and similarly for $\sim_b$ and $\sim_c$.

In simplicial models, however, the central notion is that of *local view*. From the point of view of agent a, who sees only his own card, there are four possible situations: he can be given cards 1, 2, 3 or 4. We call these the *views* of a, denoted by $V_a = \{1_a, 2_a, 3_a, 4_a\}$. Similarly for the other two agents, we have $V_b = \{1_b, 2_b, 3_b, 4_b\}$ and $V_c = \{1_c, 2_c, 3_c, 4_c\}$. In order to get a model, one must moreover define which of those views are *compatible*. Indeed, a possible world can now be seen as a set of views, one for each agent. But not every combination is allowed: in our example, $\{1_a, 1_b, 1_c\}$ is not a compatible set of views, because at most one agent can be given the card number 1. As before, there are 24 sets of compatible views, corresponding to the 24 possible worlds: $\{1_a, 2_b, 3_c\}$, $\{1_a, 2_b, 4_c\}$, etc.

Surprisingly, shifting our focus from worlds to local views reveals an underlying geometric structure in this model. Indeed, the structure described above, with a set of views and a n-ary compatibility relations between those views, is known in mathematics as an *abstract simplicial complex*[1]. Simplicial complexes provide a combinatorial description of topological spaces. In the picture below, each local view is represented as a vertex, and each set of compatible views is represented as a triangle between the corresponding three vertices. The resulting shape is that of a triangulated torus, represented on the right in 3D view, and on the left as a flattened view with some repeated vertices. Notice that we use colors to indicate agent names, as we will do in the rest of the paper.

[1] Technically, we did not explicitly require the downward-closure property of a simplicial complex. We will come back to this later when we move on to hypergraph models.



This idea led to the definition of a *pure simplicial model* in [9]. Readers unfamiliar with simplicial complexes need not worry about technical details, as we will be using hypergraphs instead in this paper. In the following, we assume the number of agents is $|\mathcal{A}| = n + 1$.

▶ **Definition 1** ([9]). *A pure simplicial model $M = (V, S, \chi, \ell)$ is given by:*

- *(V, S) is a pure simplicial complex of dimension n.*
- *$\chi : V \to \mathcal{A}$ assigns agents to vertices, s.t. every simplex has vertices of different colors.*
- *$\ell : V \to 2^{\mathsf{Ap}}$ assigns sets of atomic propositions to vertices.*

For this notion of model, the satisfaction relation $M, w \models \varphi$ is defined below, by induction on the structure of the formula φ. Note that now, the worlds w, w' are facets of the simplicial model, i.e., sets of compatible vertices. With that in mind, the previous condition (1) defining the knowledge operator in Kripke models, is replaced by condition (2). There, the condition $a \in \chi(w \cap w')$ can be read as "w and w' share an a-colored vertex", that is, agent a has the same local point of view in both worlds. Another discrepancy with respect to Kripke models is the case of atomic propositions: since the labelling ℓ decorates vertices, not worlds, with truth values of atomic propositions, we say that a proposition is true in a world when it is true in one of the vertices.

$$
\begin{array}{lll}
M, w \models p & \text{iff} & p \in \bigcup_{v \in w} \ell(v) \\
M, w \models \varphi \wedge \psi & \text{iff} & M, w \models \varphi \text{ and } M, w \models \psi \\
M, w \models \neg\varphi & \text{iff} & M, w \not\models \varphi \\
M, w \models K_a\varphi & \text{iff} & M, w' \models \varphi \text{ for every world } w' \text{ such that } a \in \chi(w \cap w') \qquad (2)
\end{array}
$$

In Definition 1, the requirement to be "pure of dimension n" ensures that every agent is present in every world of the model. While this is a standard assumption in the epistemic logic literature, it is often not the case in distributed computing. Indeed, when we study computational models where processes may crash, one usually ends up with an impure simplicial model (see e.g. [15]).

Impure simplicial models. The idea of having a different set of agents (processes) in different possible worlds (executions) is ubiquitous in distributed computing. This situation might occur when a process crashes during the execution of a protocol; or simply when the set of participating agents is not known in advance (say, a server concurrently answering requests from various clients). In reference to the idea of crashed processes, and to be consistent with previous work on the topic [3, 10, 18, 8], we will say that an agent can be "alive" or "dead" in a given world. But note that for the time being, we only model static situations, so we could also say that agents can be "present" or "absent".

In the epistemic logic literature, the topic of non-participating agents has not been thoroughly studied. It was briefly considered, e.g. in [6], where it is called a "nonrigid set of agents". This formalism is not very handy to work with, as it simply consists of extra data

on top of the model indicating which agents are alive; and it is easy to circumvent the issue entirely by considering a special local state for crashed processes. However, in simplicial models, it is quite natural and straightforward to model worlds with non-participating agents: we simply remove from Definition 1 the requirement that the model must be pure. This simple idea led to a line of research on *impure simplicial models* [3, 10, 18, 8].

While it is clear which class of models we want to consider, we quickly run into issues when we try to define the semantics of epistemic logic formulas on these models. The crux of the matter is that we have to decide how to define the satisfaction relation $w \models K_a\varphi$, in a world w where agent a is dead. Two ways of dealing with this have been proposed.

The first approach, called the three-valued semantics [3, 18], claims that such a formula should be undefined in world w. Formulas can then be either true, false or undefined, hence the name "three-valued". Defining when formulas are well-defined is not trivial, as knowledge operators can be nested and evaluating the satisfaction relation will explore the various possible worlds of the model. So one first needs to inductively define a judgment $w \bowtie \varphi$, meaning that the formula φ is well-defined in world w; and then we can define the satisfaction relation $w \models \varphi$ on top of it. The resulting logic called $\mathsf{S5}_n^{\bowtie}$ is fully axiomatized in [18]. It is a non-normal modal logic, where axiom $\mathbf{K} : K_a(\varphi \to \psi) \to (K_a\varphi \to K_a\psi)$, and even modus ponens, do not always hold. However, it retains the axiom of truth, $\mathsf{T} : K_a\varphi \to \varphi$.

The second approach, called the two-valued semantics [10, 8], is obtained by closely following the correspondence with Kripke models. As such, it yields a normal modal logic, where axiom K holds. However, the axiom of truth is lost, and only a weaker version remains, saying that alive agents are truthful: $\mathsf{alive}(a) \Rightarrow (K_a\varphi \Rightarrow \varphi)$. The reason for this is that dead agents know every formula: when agent a is dead in world w, the judgment $w \models K_a\varphi$ is vacuously true, because there is no world w' satisfying the condition in (2). The resulting logic is called $\mathsf{KB4}_n$; however, small design choices in how we define the models can result in additional axioms, as has been thoroughly investigated in [8].

This situation is quite unsatisfactory, as both approaches seem to have pros and cons, and there is no obvious way to tell which one might turn out to be more useful in practice. In this paper, we introduce a many-sorted logic that avoids entirely the problem of undefined formulas. Rather than being a third proposal, it subsumes and unifies the previous two approaches. But before we present the syntax of our logic, let us argue in favor of moving from simplicial complexes to hypergraphs.

From simplicial complexes to hypergraphs. In pure simplicial models (Definition 1), the vertices represent points of view of individual agents, and the facets (a.k.a. maximal simplexes) represent the possible worlds. There are also simplexes of lower dimension, but they do not seem to have a meaning. They are only here to preserve the geometric structure of the model: in order to have a triangle, we must also have the three edges of the triangle.

When we move on to impure simplicial models, we are now allowing worlds of different dimensions. So there is no good reason why only the facets of the model should represent worlds. In fact, as was shown in [10], having only facets as worlds results in some dubious axioms. The idea that all simplexes, not only the facets, could represent worlds was briefly discussed in [4], and later studied in [3, 18]. The idea was further generalized in [8], where models are equipped with additional data (called a covering) allowing to explicitly say which simplexes are worlds or not. Thus, models can be *minimal* (only the facets are worlds), *maximal* (all simplexes are worlds), or anything in-between. For instance in Figure 1 (left), the model has only one facet (the blue triangle) but three worlds: w_1 is the triangle itself, where three agents are alive; w_2 is an edge, where only two agents are alive; and w_3 is a vertex, where only one agent is alive.

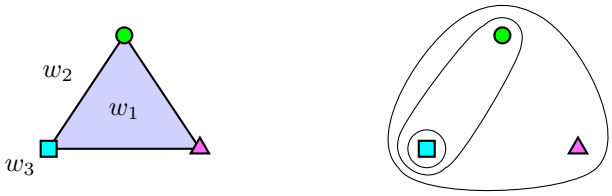


Figure 1 A simplicial model with three worlds, and the equivalent hypergraph representation.

Still, the simplicial model depicted on the left has some simplexes that play no epistemic role (the two edges and two vertices with no label). This introduces a strange discrepancy between simplexes that represent worlds, and simplexes that do not. Our proposal, which is quite mild from a technical standpoint, is to simply get rid of those extra simplexes. The resulting structure is called a hypergraph.

▶ **Definition 2.** *A (simple) hypergraph H is given by a pair (V, E), where V is a set of vertices, and $E \subseteq 2^V$ is a set of hyperedges (or just "edges", when clear from context).*

Note that the only difference between a hypergraph and a simplicial complex, is that the set E of hyperedges does not have to be downward-closed. A hypergraph is depicted in Figure 1 (right). It consists of three vertices $V = \{x, y, z\}$ (named from left to right), and three hyperedges $E = \{\{x\}, \{x, y\}, \{x, y, z\}\}$. Hyperedges are represented by a closed curve around the corresponding vertices, not unlike a Venn diagram. By moving from simplicial complexes to hypergraphs, we seem to lose the geometric intuition behind simplicial complexes. But it can be easily recovered by computing the downward closure of E: if (V, E) is a hypergraph, then $(V, {\downarrow}E)$ is a simplicial complex.

▶ Remark 3. For readers familiar with the hierarchy of models introduced in [8], hypergraph models fit nicely within that picture, as a strict subclass of epistemic covering models. They are not as general as simplicial sets, as there cannot be complex connectivity between hyperedges. They can be either minimal, or maximal, or in-between. Simple hypergraphs as in Definition 2 give rise to proper models, but as we will see later, we can also model non-proper behavior by allowing several hyperedges with the same sets of vertices.

Many-sorted epistemic logic. We now describe the main contribution of the paper. We propose a new syntax for epistemic logic formulas where agents can be dead or alive. The central idea is to introduce several sorts of formulas[2]: *world formulas* are to be interpreted in a world of the model, and *agent formulas* are interpreted in a point of view of a particular agent (i.e. in a vertex, for hypergraph models). We usually denote world formulas in capital letters Φ, Ψ, and agent formulas in lowercase, with a subscript indicating the name of an agent φ_a, ψ_a. Thus, our logic has $|\mathcal{A}| + 1$ sorts, one for each agent $a \in \mathcal{A}$, and an extra one for the world formulas.

A key observation is the following. Let us look again at the semantics of the knowledge operator for simplicial models, condition (2). It says that an agent a knows φ in a world w, precisely when φ holds in every world w' in which a has the same point of view. In fact, this definition does not refer to the "real world" w; it only refers to the point of view of a in this

[2] A familiar example of a many-sorted modal logic is CTL*, where the syntax is divided between state formulas and path formulas.

world! This suggests that the knowledge operator K_a should really be interpreted not in a world, but in a point of view of agent a. This gives us the following syntax for agent a's formulas, where p_a ranges over atomic propositions concerning agent a:

$$\varphi_a ::= p_a \mid \neg\varphi \mid \varphi \wedge \psi \mid \mathsf{K}_a\Phi$$

This gives us the first $|\mathcal{A}|$ sorts of our logic, with one sort of agent formulas for each $a \in \mathcal{A}$. Note that agent formulas can only talk about one particular agent. For instance, the expression $\mathsf{K}_a\Phi \wedge \mathsf{K}_b\Psi$ is not a syntactically valid formula. Also note that Φ is a world formula. We now explain the syntax of world formulas.

Since agents can be alive or dead, we cannot talk about the knowledge of a specific agent in world formulas, or we will run into the issue of how to define the knowledge of a dead agent. Instead, we introduce two new modal operators, E_a and A_a, that can test whether an agent exists in this world. As the names indicate, E_a has an existential flavor, while A_a is its universal counterpart. However, note that they are not binders: E_a should **not** be read as "there exists an agent a such that". Rather, the intuitive meaning of those operators is the following:

- $\mathsf{E}_a\varphi_a$: "there exists a point of view for agent a such that φ_a holds".
- $\mathsf{A}_a\varphi_a$: "for every point of view of agent a, φ_a holds".

The syntax of world formulas is as follows, where p_e denotes atomic propositions that do not talk about specific agents ('e' stands for *environment* here).

$$\Phi ::= p_e \mid \neg\Phi \mid \Phi \wedge \Psi \mid \mathsf{E}_a\varphi_a \mid \mathsf{A}_a\varphi_a \qquad \text{where } a \in \mathcal{A}$$

Note that, unlike with agent formulas, we have $|\mathcal{A}|$-many modal operators to choose from. So, for example, the following world formula is syntactically valid: $\mathsf{E}_a\mathsf{K}_a\Phi \wedge \mathsf{A}_b\mathsf{K}_b\Psi$. It is read "there exists a point of view of agent a where a knows Ψ, and for every point of view of agent b, b knows Ψ". Observe how whenever we want to talk about the knowledge of an agent, we are forced to explicitly quantify over the points of view of that agent. This avoids entirely the question of "undefined formulas", where we had to make an arbitrary decision about the meaning of knowledge for dead agents.

▶ Remark 4. It might seem strange to have a modality A_a quantifying over "all points of view of agent a", when there can be at most one point of view per agent in a given world. First, note that when agent a is absent in a world, the operator A_a is vacuously true, while E_a is false. So these two operators do behave differently in hypergraph models. Secondly, one could consider an extension of hypergraph models where an agent can have multiple points of view about the world. We briefly explore this idea in Section 4.3, where we relate it to the neighborhood semantics of epistemic logic [17].

In some sense, our logic can be viewed as a *refinement* of the usual knowledge operator K_a into two distinct operators: $\mathsf{A}_a\mathsf{K}_a\Phi$ is the 2-valued semantics of [10, 8], which is vacuously true when agent a is dead; while $\mathsf{E}_a\mathsf{K}_a\Phi$ is closer to the 3-valued semantics of [3, 18]

Related work. As we already explained, this work is directly related to the line of work on simplicial models [9], especially those that deal with impure simplicial complexes [3, 10, 18, 8]. Recently, a single-sorted epistemic logic on hypergraphs was considered in [2] to study weakly aggregative logics. There, vertices of hypergraphs are not colored as they are interpreted as worlds, so these models lie in-between epistemic frames and neighborhood frames. A framework that uses adjoint modalities was studied in [19] in the context of epistemic

modalities "agent is uncertain about" and "agent has information that". In interpreted systems [6], epistemic frames are generated by explicitly modeling the local states of agents and global states of the environment. However, at the level of syntax, no difference is made between local properties of the agents, and global properties of the environment.

Plan of the paper. In Section 2, we start by describing the syntax of the logic 2CH and its semantics in chromatic hypergraphs. We give an axiomatization of the logic in Section 3, where we prove the completeness result in Theorem 23. In Section 4, we relate hypergraph models with partial epistemic models by showing an isomorphism of categories (Theorem 26). We use this equivalence to formulate a translation from $\mathsf{KB4}_n$-formulas into 2CH-formulas, showing that the latter is a conservative extension of the former (Theorem 30).

2 Two-level chromatic hypergraph logic 2CH

We introduce a many-sorted refinement of multi-agent epistemic logic $\mathsf{KB4}_n$ studied in [10]. Our logic has $|\mathcal{A}|+1$ sorts of formulas: $|\mathcal{A}|$ sorts of *agent* formulas, one per agent, and one sort of *world* formulas. Since this logic is intended to be interpreted on chromatic hypergraphs (see Section 2.2), we name it the *2-level Chromatic Hypergraph Logic*, 2CH. The intended interpretation of agent formulas is to describe local information that belongs to a point of view of a specific agent. On the other hand, world formulas talk about properties of the environment, or world. As we will see in Section 3, this many-sorted logic, or two-level logic, embeds faithfully the logic $\mathsf{KB4}_n$; but it also makes explicit (and not up to model interpretation as in [10]) the various choices about how much agents can observe each other's presence or absence.

2.1 Syntax

Fix a finite set $\mathcal{A}$ of agents. For each $a \in \mathcal{A}$, we have a set Ap_a of atomic propositions about agent a. We also have a set Ap_e of atomic propositions for the environment. We use lowercase letters with subscripts $\varphi_a, \psi_a, \dots$ to denote agent formulas, and uppercase letters $\Phi, \Psi, \dots$ for world formulas.

▶ **Definition 5.** *The language of the logic* 2CH *is defined as follows. For each agent $a \in \mathcal{A}$, there is a sort of agent formulas generated by the following grammar:*

$$\varphi_a ::= p_a \mid \neg\varphi \mid \varphi \wedge \psi \mid \widehat{\mathsf{K}}_a\Phi \qquad \text{where } p_a \in \mathsf{Ap}_a$$

The sort of world formulas is generated by the following grammar:

$$\Phi ::= p_e \mid \neg\Phi \mid \Phi \wedge \Psi \mid \mathsf{E}_a\varphi_a \qquad \text{where } a \in \mathcal{A} \text{ and } p_e \in \mathsf{Ap}_e$$

▶ Remark 6. We emphasize that this is truly an $(|\mathcal{A}|+1)$-sorted logic, not a 2-sorted one. In particular, it is important to notice that the indexes in the operators $\widehat{\mathsf{K}}_a$ and E_a play a very different role. To illustrate that, let us fix two distinct agents $a, b \in \mathcal{A}$. To write an a-sorted agent formula φ_a, we are only allowed to refer to agent a's atomic propositions; and we are only allowed to use the modal operator $\widehat{\mathsf{K}}_a$, with index a. No reference to agent b is allowed. Similarly, an agent formula φ_b of sort b is not allowed to refer to agent a. Crucially, the following formula is ill-formed: $\widehat{\mathsf{K}}_a\Phi \wedge \widehat{\mathsf{K}}_b\Psi$. It cannot be generated by the grammar φ_a of a-sorted formulas, nor by the grammar φ_b of b-sorted formulas. Thus, it is not a valid formula in our logic. On the other hand, the sort of world formulas behaves much closer to the usual language of multi-agent epistemic logic. In a world formula, we are allowed to choose between $|\mathcal{A}|$-many modal operators E_a. Thus, a world formula of the form $\mathsf{E}_a\varphi_a \wedge \mathsf{E}_b\varphi_b$ is allowed.

We will use standard propositional connectives like true, $\vee$, $\Rightarrow$, defined as usual. There are also dual modalities: $\mathsf{K}_a\Phi := \neg\widehat{\mathsf{K}}_a\neg\Phi$, and $\mathsf{A}_a\varphi_a := \neg\mathsf{E}_a\neg\varphi_a$. Modalities are read as follows: $\widehat{\mathsf{K}}_a$ means "agent a considers possible that", K_a means "agent a knows that", E_a means "there exists a point of view of agent a such that", and A_a means "for all points of view of agent a". We call $\widehat{\mathsf{K}}_a$ and E_a *existential* modalities and K_a and A_a *universal* modalities.

2.2 Semantics

A *hypergraph* is a generalization of a graph, where instead of just edges between pairs of vertices, one has *hyperedges* that can connect multiple vertices at once. In the introduction, we defined *simple* hypergraphs (Definition 2), to explain the proximity with simplicial complexes. In fact, we will be slightly more general than that, and allow multiple hyperedges to have the same set of vertices; this will allow us to model non-proper behavior. Namely, a (non-simple) hypergraph H is a triple (V, E, P), where V is the set of vertices, E is the set of hyperedges, and $P : E \to 2^V$ assigns to each hyperedge a set of vertices.

In the context of multi-agent systems, we need moreover to consider *chromatic* hypergraphs, where each vertex is assigned an agent name. This could be done by adding an extra piece of data $\chi : V \to \mathcal{A}$, as in chromatic simplicial complexes [9]. Instead, we tweak the definition a little bit in order to make explicit the set of vertices assigned to each individual agent. This is similar to the definition of chromatic semi-simplicial sets in [8].

▶ **Definition 7.** *A chromatic hypergraph H is a tuple $(E, \{V_a\}_{a\in\mathcal{A}}, \{\mathsf{pr}_a\}_{a\in\mathcal{A}})$, where:*

- *E is a set of hyperedges,*
- *for all $a \in \mathcal{A}$, V_a is the set of* views *of agent a,*
- *for each agent $a \in \mathcal{A}$, $\mathsf{pr}_a : E \to V_a$ is a surjective partial function. Additionally, we require that for each $e \in E$, $\mathsf{pr}_a(e)$ is defined for at least one $a \in \mathcal{A}$.*

Indeed, defining $V = \bigcup_{a\in\mathcal{A}} V_a$ as the total set of vertices of a chromatic hypergraph, and $P(e) = \{\mathsf{pr}_a(e) \mid a \in \mathcal{A}\} \subseteq 2^V$, we can view H as a regular hypergraph. We will use the words *view* and *vertex* interchangeably, as well as *world* and *hyperedge*. Given a world $e \in E$, when $\mathsf{pr}_a(e)$ is undefined, we say that agent a is *dead* in e. Otherwise, a is *alive* in e, and we call $\mathsf{pr}_a(e)$ the view of a in e. Moreover, when $\mathsf{pr}_a(e) = v$, we say that v belongs to e, or that e contains v, and occasionally write $v \in_a e$. If a hyperedge consists of views $v_0, \dots v_{n-1}$ then we say that these views are *compatible*. Let us explain each condition imposed on pr_a.

- pr_a is surjective: every view belongs to at least one world.
- pr_a is partial: not all agents are required to be alive in a world.
- pr_a is functional: every world contains at most one view of each agent.
- $\mathsf{pr}_a(e)$ is defined for at least one $a \in \mathcal{A}$: every world contains at least one alive agent.

▶ Remark 8. In chromatic hypergraphs, views and worlds are of equal importance: they are both described explicitly in the sets E and $(V_a)_{a\in\mathcal{A}}$. This contrasts with Kripke structures, where the set of worlds is explicit, but the views are implicit in the indistinguishability relations. Recent work on simplicial models has highlighted the importance of considering the views as a first-class notion. Here, we take this idea one step further with our two-level syntax, where world formulas talk about properties of the world, and agent formulas talk about properties of an agent's point of view. We also avoid choosing between *local* vs. *global* atomic propositions (in the sense of [9]): agent formulas can talk about (local) properties of agent $a \in \mathcal{A}$ using atoms in Ap_a, and world formulas can talk about (global) properties of the world using atoms in Ap_e.

▶ **Example 9.** Three examples of chromatic hypergraphs are depicted in Figure 2. The three agents a, b, c are represented as colored shapes ●, ■, ▲, respectively. In all three hypergraphs, there is only one view per agent.

On the leftmost figure, all possible combinations of views are compatible, giving 7 hyperedges. Intuitively, in this situation, the agents do not know whether other agents exist. A scenario like this is standard in distributed computing, where agents are processes, and they do not know whether other processes are concurrently running.

The hypergraph depicted in the middle has a single hyperedge containing the three views. In this situation, all agents know that everyone is alive, as it is the only possible world. This represents a scenario where every agent has guarantees that the other two agents are running.

The rightmost figure is a hypergraph with three hyperedges, each of which contains two views. It represents a situation where the points of views are pairwise compatible, but not all three of them are compatible. That is, there is no possible world that realizes all three of them at once. This could model a scenario where each agent receives a message from one of the other two agents, but they do not know who sent the message. Another interpretation might be in a quantum setting, as an example of *contextuality* [1].

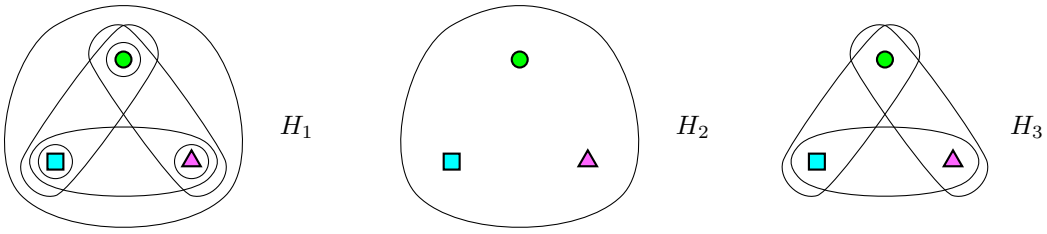


Figure 2 Three examples of chromatic hypergraphs H_1, H_2, H_3.

▶ **Definition 10.** *A chromatic hypergraph model is a tuple* $(H, \{\ell_a\}_{a\in\mathcal{A}}, \ell_e)$*, where* H *is a chromatic hypergraph, and* $\ell_a : \mathsf{Ap}_a \to P(V_a)$*,* $\ell_e : \mathsf{Ap}_e \to P(E)$ *are valuation functions.*

From now on, we sometimes omit the adjective "chromatic" when clear from context.

We can now define the semantics of 2CH formulas with respect to chromatic hypergraph models. Given a hypergraph model H, the satisfaction relations are defined for every sort by mutual induction. As expected, world formulas are interpreted in a world $e \in E$, and agent formulas are interpreted in a point of view $v \in V_a$ of that agent.

$$
\begin{array}{lll}
H, v \models_a p_a & \text{iff} & v \in \ell_a(p_a) \\
H, v \models_a \neg\varphi & \text{iff} & H, v \not\models_a \varphi \\
H, v \models_a \varphi \wedge \psi & \text{iff} & H, v \models_a \varphi \text{ and } H, v \models_a \psi \\
H, v \models_a \widehat{\mathsf{K}}_a \Phi & \text{iff} & H, e \models_e \Phi \text{ for some } e \in E \\
 & & \text{such that } \mathsf{pr}_a(e) = v
\end{array}
\qquad
\begin{array}{lll}
H, e \models_e p_e & \text{iff} & e \in \ell_e(p_e) \\
H, e \models_e \neg\Phi & \text{iff} & H, e \not\models_e \Phi \\
H, e \models_e \Phi \wedge \Psi & \text{iff} & H, e \models_e \Phi \text{ and } H, e \models_e \Psi \\
H, e \models_e \mathsf{E}_a\varphi & \text{iff} & H, v \models_a \varphi \text{ for some } v \in V_a \\
 & & \text{such that } \mathsf{pr}_a(e) = v
\end{array}
$$

2.3 Examples

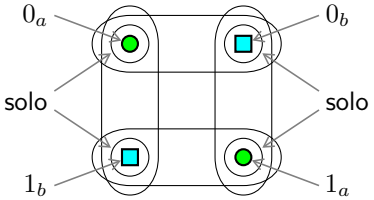
Let us illustrate the semantics of 2CH on a few examples.

▶ **Example 11.** Consider again the three hypergraphs from Example 9, which we denote by H_1, H_2, H_3, from left to right. For now, we do not worry about atomic propositions: we just need to assume that the sets Ap_* are non-empty, in order to get the constant $\mathsf{true} = p \vee \neg p$. Also, recall that the agents are depicted as $a =$ ●, $b =$ ■, and $c =$ ▲.

In the model H_1, let us consider the hyperedge $e = \{●, ■\}$. Then we have $H_1, e \models \mathsf{E}_a\mathsf{true}$ and $H_1, e \models \mathsf{E}_b\mathsf{true}$, but $H_1, e \not\models \mathsf{E}_c\mathsf{true}$. Indeed, the edge e does not contain a point of view of agent c, i.e., $\mathsf{pr}_c(e)$ is undefined. In fact, the world formula "$\mathsf{E}_a\mathsf{true}$" is satisfied exactly in the worlds where agent a is alive. So let us write $\mathsf{alive}(a) := \mathsf{E}_a\mathsf{true}$, so that $H, e \models \mathsf{alive}(a)$ iff e contains a point of view of a, that is, iff $\mathsf{pr}_a(e)$ is defined.

We can now talk about whether agents know that other agents are alive. Let $v = ●$ be the (unique) point of view of a in the model. In model H_1, all combinations of alive and dead agents are possible. So as expected, $H_1, v \models \neg\mathsf{K}_a\mathsf{alive}(b)$. In model H_2 however, a knows that all agents are alive, since this is the only possible world: $H_2, v \models \mathsf{K}_a(\mathsf{alive}(b) \wedge \mathsf{alive}(c))$. Finally in model H_3 the situation is more complicated: a knows that another agent is alive, but does not know which one. Thus, $H_3, v \models \mathsf{K}_a(\mathsf{alive}(b) \vee \mathsf{alive}(c)) \wedge \neg\mathsf{K}_a\mathsf{alive}(b) \wedge \neg\mathsf{K}_a\mathsf{alive}(c)$.

▶ **Example 12** (2-agent binary input model with solo executions)**.** As an example where atomic propositions play a role, we consider the situation where two agents are given a binary input value, either 0 or 1. Moreover, there can be solo executions (a.k.a. initial crash failures, in distributed computing), so that agents do not know if they are running alone or not. As before, the agents are depicted as $a = ●$ and $b = ■$. The sets of atomic propositions are $\mathsf{Ap}_a = \{0_a, 1_a\}$, $\mathsf{Ap}_b = \{0_b, 1_b\}$, and $\mathsf{Ap}_e = \{\mathsf{solo}\}$. The agent atomic propositions hold in the points of view indicated on the picture below. The environment atomic proposition solo holds in the four singleton hyperedges where only one agent is alive.



Let v be the top-left vertex, where agent a has input value 0. Then by definition $H, v \models 0_a$. Moreover, a does not know whether agent b is alive: $H, v \models \neg\mathsf{K}_a\mathsf{E}_b\mathsf{true}$, which we could also reformulate as $H, v \models \neg\mathsf{K}_a\mathsf{solo}$, that is, a does not know whether this is a solo execution. We can also say that a considers possible that b is alive with value 1: $H, v \models \widehat{\mathsf{K}}_a\mathsf{E}_b 1_b$. As a last example, we can express that a knows that if b is alive, its value is either 0 or 1: $H, v \models \mathsf{K}_a(\neg\mathsf{solo} \Rightarrow \mathsf{E}_b(0_b \vee 1_b))$. Alternatively, we could have used the operator A_b to express the same fact without a conditional: $H, v \models \mathsf{K}_a\mathsf{A}_b(0_b \vee 1_b)$.

2.4 Safe and unsafe knowledge

In traditional epistemic logics, formulas are interpreted in a world of the model. In the logic 2CH, to talk about the knowledge of an agent in a world, we first need to quantify over the points of view of this agent. Since there are two quantifiers E_a and A_a, we obtain two different knowledge operators on worlds, which we call *safe* and *unsafe* knowledge.

$$K_a^{\text{safe}}\Phi := \mathsf{E}_a\mathsf{K}_a\Phi \qquad\qquad K_a^{\text{unsafe}}\Phi := \mathsf{A}_a\mathsf{K}_a\Phi$$

These two operators only differ in the knowledge of dead agents. Indeed, given a world e and an agent a which is dead in e (i.e., $\mathsf{pr}_a(e)$ is undefined), we have $H, e \not\models K_a^{\text{safe}}\Phi$ (dead agents know nothing), whereas $H, e \models K_a^{\text{unsafe}}\Phi$ (dead agents know everything). However, when the agent a is alive in e, the two notions agree: $H, e \models K_a^{\text{safe}}\Phi \iff H, e \models K_a^{\text{unsafe}}\Phi$.

3 Axiomatics

The logic 2CH has all the usual inference rules of classical propositional logic (such as modus ponens) together with classical tautologies. It also has the following rules for modalities. We annotate the $\vdash$ symbol with the sort of the corresponding formula, $a \in \mathcal{A}$ for agent sorts and e for the world sort.

$$\frac{\vdash_e \Phi}{\vdash_a \mathsf{K}_a\Phi}\ \textsc{Nec-a} \qquad \frac{\vdash_a \varphi}{\vdash_e \mathsf{A}_a\varphi}\ \textsc{Nec-e} \qquad \frac{\vdash_e \Phi \to \Psi}{\vdash_a \heartsuit\Phi \to \heartsuit\Psi}\ \text{RM} \qquad \frac{\vdash_a \varphi \to \psi}{\vdash_e \heartsuit\varphi \to \heartsuit\psi}\ \text{RM'}$$

$$\frac{\vdash_e \Phi \to \mathsf{A}_a\psi}{\vdash_a \widehat{\mathsf{K}}_a\Phi \to \psi}\ \textsc{Adj-1} \qquad \frac{\vdash_a \varphi \to \mathsf{K}_a\Psi}{\vdash_e \mathsf{E}_a\varphi \to \Psi}\ \textsc{Adj-2}$$

where $\heartsuit \in \{\mathsf{E}_a, \mathsf{A}_a\}$ for rule RM, and $\heartsuit \in \{\widehat{\mathsf{K}}_a, \mathsf{K}_a\}$ for rule RM'. The first two rules are necessitation rules. The next two rules are monotonicity rules. The last two rules are called adjunction rules, and the double horizontal bar indicates that they go in both directions: top-to-bottom and bottom-to-top. They describe the interaction between the two pairs of modalities. Finally, we have the following axiom schemes for modalities:

- $\vdash_a \varphi \to \widehat{\mathsf{K}}_a\mathsf{E}_a\varphi$: every point of view belongs to some world;
- $\vdash_a \widehat{\mathsf{K}}_a\mathsf{E}_a\varphi \to \varphi$: every world has at most one point of view of a given agent;
- $\vdash_e \bigvee_{a\in\mathcal{A}} \mathsf{E}_a\mathsf{true}$: every world contains at least one point of view.

As we will see, the axiom schemes for modalities correspond to the defining properties of chromatic hypergraphs. Therefore, we will call the first axiom scheme *surjecitivity*, the second one *functionality*, and the third one *non-emptiness*.

▶ **Proposition 13.** *The logic* 2CH *is sound with respect to chromatic hypergraphs.*

3.1 Playing with the logic 2CH

In this logic, we can show that the universal modalities satisfy axiom K:

▶ **Proposition 14.** *For* $\heartsuit \in \{\mathsf{A}_a, \mathsf{K}_a\}$*, the axiom* $\mathsf{K}_\heartsuit$ *holds:* $\heartsuit(\varphi \to \psi) \to (\heartsuit\varphi \to \heartsuit\psi)$.

We give a list of useful formulas that are derivable in the logic 2CH.

▶ **Proposition 15.** *The following statements are derivable in the two-level logic* 2CH*:*

1. $\mathsf{E}_a\varphi \to \mathsf{A}_a\varphi$
2. $\mathsf{K}_a\Phi \to \mathsf{K}_a\mathsf{E}_a\mathsf{K}_a\Phi$
3. $\mathsf{E}_a\mathsf{K}_a\Phi \to \Phi$
4. $\Phi \to \mathsf{A}_a\widehat{\mathsf{K}}_a\Phi$
5. $\varphi \to \mathsf{K}_a\mathsf{E}_a\varphi$
6. $\mathsf{K}_a\Phi \to \widehat{\mathsf{K}}_a\Phi$

Here is an intuitive explanation of these formulas:

1. If φ holds in a point of view of a, then it holds in all points of view of a. That is, there can be at most one point of view per world.
2. This is a form of positive introspection: if an agent knows something, then he knows that he knows it. However, we can put stress on the last "he", that is, "if he knows something, then he knows that *he* knows it".
3. This is a form of veracity: if a fact about a world is known by someone, then it is true.
4. If a certain fact holds in a world, then the agents in this world consider this fact possible. This is related to negative introspection.
5. Agents know the local facts about themselves.
6. This is the usual modal axiom D. It reflects the fact that every point of view belongs to at least one world.

In [9], the use of local atomic propositions leads to a so-called assumption of locality, $K_a(p_{a,x}) \vee K_a(\neg p_{a,x})$. In hypergraph models, valuations are local by construction:

▶ **Proposition 16.** *For any $p_a \in \mathsf{Ap}_a$, the formula $\mathsf{K}_a\mathsf{E}_a p_a \vee \mathsf{K}_a\mathsf{E}_a\neg p_a$ is derivable in* 2CH.

Proof. By adjunction rules, we have $\widehat{\mathsf{K}}_a\mathsf{A}_a p_a \to p_a$ and $p_a \to \mathsf{K}_a\mathsf{E}_a p_a$. By cut rule, we have $\widehat{\mathsf{K}}_a\mathsf{A}_a p_a \to \mathsf{K}_a\mathsf{E}_a p_a$. This is equivalent to $\mathsf{K}_a\mathsf{E}_a p_a \vee \mathsf{K}_a\mathsf{E}_a\neg p_a$ by propositional logic. ◀

Note that the above proof does not use the fact that p_a is atomic, so in fact, an agent can decide *any* formula about itself. That is, for any φ, $\mathsf{K}_a\mathsf{E}_a\varphi \vee \mathsf{K}_a\mathsf{E}_a\neg\varphi$ is derivable.

3.2 Completeness

The proof of completeness uses the standard canonicity argument, extended to the many-sorted case. There is nothing surprising: the canonical model consists of the maximal consistent sets of formulas, now of several sorts. These sets satisfy standard properties, and together form a chromatic hypergraph.

▶ **Definition 17.** *A set of formulas S_* of sort $*$ is* inconsistent *if* false *can be derived from it. Otherwise, it is called consistent. A consistent set of formulas is* maximal *if it is not a proper subset of any other consistent set of formulas.*

▶ **Proposition 18.** *Maximal consistent sets (MCS) of formulas of sort $*$ are closed under modus ponens: for any formula ξ of sort $*$, either $\xi \in S_*$ or $\neg\xi \in S_*$, and every (non-maximal) consistent set of formulas of sort $*$ is contained in a maximal consistent set of formulas of sort $*$.*

▶ **Definition 19.** *The canonical hypergraph model consists of the following:*
- *the set of hyperedges is $E = \{S_e \mid S_e$ is a MCS of sort $e\}$;*
- *for each agent a, the set of vertices is $V_a = \{S_a \mid S_a$ is a MCS of sort $a\}$;*

together with relations $R_a \subseteq E \times V_a$ for every agent a, which are defined as follows: $S_E R_a S_a$ iff for all formulas Φ, if $\Phi \in S_E$, then $\widehat{\mathsf{K}}_a\Phi \in S_a$. The valuation function is defined by: $\ell_*(p_*) = \{S_* \mid p_* \in S_*\}$.

▶ **Proposition 20.** *In the canonical model, if $S_E R_a S_a$, if $\mathsf{K}_a\Phi \in S_a$, then $\Phi \in S_E$.*

▶ **Lemma 21.** *The canonical model is a chromatic hypergraph, that is, R_a is surjective, functional, and for any S_E there is S_a such that $S_E R_a S_a$ for at least one a.*

Proof. First, suppose that $S_E R_a S_a$, $S_E R_a S'_a$, and $S_a \neq S'_a$. It means that there is a formula φ which is in S_a, but is not in S'_a. By adjoint axiom and modus ponens, $\mathsf{K}_a\mathsf{E}_a\varphi$ is in S_a. By Proposition 20, $\mathsf{E}_a\varphi$ is in S_E. Using the definition of the canonical model, $\widehat{\mathsf{K}}_a\mathsf{E}_a\varphi$ belongs to S'_a. From there, by functionality axiom and modus ponens, $\varphi \in S'_a$, which is a contradiction. Thus, $S_a = S'_a$.

Second, we need to show that in the canonical model every vertex belongs to a hyperedge. Assume this is not the case, that is, there is a vertex S_a that does not belong to any hyperedge. It means that there is no maximal consistent set of formulas that contains $S = \{\Phi \mid \mathsf{K}_a\Phi \in S_a\}$. In particular, it means that S is itself not consistent, that is, there is a finite set of formulas $\{\Phi_i\}$ such that $\bigwedge_i \Phi_i \to \mathsf{false}$ is derivable. By applying necessitation, we get that $\mathsf{K}_a(\bigwedge_i \Phi_i \to \mathsf{false})$ is derivable, thus belongs to S_a. As K_a distributes over conjunction, and every $\mathsf{A}_a\Phi_i$ is in S_a, we get that $\mathsf{K}_a \bigwedge_i \Phi_i$ is in S_a. Applying modus ponens, we get that $\mathsf{K}_a\mathsf{false}$ is in S_a. Using surjectivity axiom, we get that false is in S_a, that is S_a is not consistent, which is a contradiction.

Lastly, we need to show that every hyperedge contains some vertex. Suppose it is not the case, that is there is a hyperedge S_E that does not contain any vertex. It means that for every a, the set $\widehat{\mathsf{K}}_a S_E$ is not consistent. Thus, for all a, there is a finite set of formulas $\{\Phi_i^a\} \subset S_E$, such that $\bigwedge_i \widehat{\mathsf{K}}_a \Phi_i^a \to \mathsf{false}$ is derivable. We now show that this implies that S_E is not consistent. By applying necessitation, we get that $\mathsf{A}_a(\bigwedge_i \widehat{\mathsf{K}}_a \Phi_i^a \to \mathsf{false})$ is derivable for every a. By (K) and modus ponens, we derive $\mathsf{A}_a(\bigwedge \widehat{\mathsf{K}}_a \Phi_i^a) \to \mathsf{A}_a\mathsf{false}$. Combining them all together, we have that $\bigwedge_a \bigwedge_i (\mathsf{A}_a \widehat{\mathsf{K}}_a \Phi_i^a) \to \bigwedge_a \mathsf{A}_a\mathsf{false}$ is derivable too. The antecedent is in S_E because every Φ_i^a is in S_E and $\Phi \to \mathsf{A}_a \widehat{\mathsf{K}}_a \Phi$ is an axiom. Thus, $\bigwedge_a \mathsf{A}_a\mathsf{false}$ is in S_E, which means that S_E is not consistent since $\bigvee_a \mathsf{E}_a\mathsf{true}_a$ is an axiom, which is its negation. We have a contradiction, which means that every hyperedge contains some vertex. ◀

▶ **Lemma 22.** *In the canonical model, $S_* \models_* \xi$ iff $\xi \in S_*$.*

▶ **Theorem 23.** *The logic* 2CH *is complete with respect to chromatic hypergraph models.*

4 Links to related work

4.1 Equivalence with partial epistemic frames

Let us recall first the definition of a partial epistemic frame, which has been one of the main models used in the study of epistemic logics such as $\mathsf{KB4}_n$ in [10]:

▶ **Definition 24.** *Given the set of agents $\mathcal{A}$, a partial epistemic frame $\mathcal{M}$ consists of a set of worlds M together with a family of partial equivalence relations $\{\sim_a\}_{a\in\mathcal{A}}$, such that for every $w \in M$, $w \sim_a w$ for at least one $a \in \mathcal{A}$. A morphism of partial epistemic frames is a function $f : M \to M'$ such that for every $a \in \mathcal{A}$ and $w, w' \in M$, $w \sim_a w'$ implies $f(w) \sim_a f(w')$.*

We can transform a partial epistemic frame into a chromatic hypergraph, and vice versa, using the following construction. Suppose we are given a partial epistemic frame $\mathcal{M}$. We construct a chromatic hypergraph $\eta(\mathcal{M})$ by setting $E = M$ and $V_a = M/_{\sim_a}$, that is the set of hyperedges is exactly the set of worlds, and the set of vertices of color a is the set of equivalence classes of $\sim_a$. We then set $\mathsf{pr}_a(w)$ to be $[w]_a$, that is the equivalence class of w under $\sim_a$. It is easy to check that this indeed defines a chromatic hypergraph.

Conversely, given a chromatic hypergraph H, we can construct a partial epistemic frame $\kappa(H)$. We set the set of worlds M to be equal to the set of hyperedges of H, and $e \sim_a e'$ if and only if e and e' share an a-colored vertex. This yields a partial equivalence relation.

These maps can be seen as the dual hypergraph construction: if $H = (V, E)$ is a (non-chromatic) hypergraph, then H^* is the hypergraph (E, V) where the hyperedges are the vertices of H and the vertices are the hyperedges of H. Partial epistemic frames can be seen as hypergraphs that have colored hyperedges which are defined by equivalence classes. The correspondence is exemplified in Figure 3.

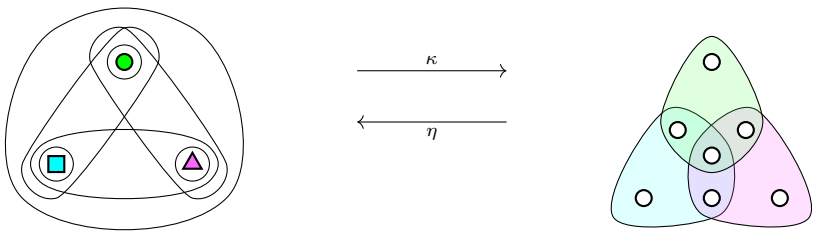


Figure 3 Example of correspondence between chromatic hypergraphs and frames.

In fact, η and κ can be extended to morphisms of partial epistemic frames and chromatic hypergraphs, giving an equivalence of categories. First, we need to define the corresponding morphisms of chromatic hypergraphs:

▶ **Definition 25.** *A morphism of hypergraphs $f : H \to H'$ is a family of functions $f_a : V_a \to V'_a$ for each agent a, together with a function $f_e : E \to E'$ such that for all $a \in \mathcal{A}$, if $\mathsf{pr}_a(e) = v$ then $\mathsf{pr}'_a(f_e(e)) = f_a(v)$.*

▶ **Theorem 26.** *The category of partial epistemic frames is isomorphic to the category of chromatic hypergraphs. In particular, for any chromatic hypergraph H, $\eta(\kappa(H))$ is isomorphic to H, and for any partial epistemic frame $\mathcal{M}$, $\kappa(\eta(\mathcal{M}))$ is isomorphic to $\mathcal{M}$.*

In light of Theorem 26, chromatic hypergraphs and partial epistemic frames contain exactly the same information. So, in theory, we could have defined the semantics of 2CH in partial epistemic frames. However, this would be quite unnatural to do, since epistemic frames do not have a tangible notion of point of view: we would have to attach atomic propositions, and interpret agent formulas, in the equivalence classes of $\sim_a$.

Instead, we can still embed partial epistemic models (with only world atomic propositions) into a subclass of chromatic hypergraphs models, such that $\mathsf{Ap}_a = \varnothing$ for every agent. From this, we can extend Theorem 26 to work at the level of models:

▶ **Corollary 27.** *The category of partial epistemic models is isomorphic to the category of chromatic hypergraph models with empty sets of atomic propositions for agents.*

4.2 Translation from $\mathsf{KB4_n}$ to 2CH

We can use the equivalence of epistemic frames and hypergraphs for showing how the logics $\mathsf{KB4}_n$ and 2CH are related: we will show that 2CH is a conservative extension of $\mathsf{KB4}_n + \mathsf{NE}$, where axiom NE ensures that there is an alive agent in each world (see [10] for further details). From semantics side, the worlds of epistemic frames are the hyperedges of hypergraphs, thus the formulas of $\mathsf{KB4}_n$ are to be translated to the *world* formulas of 2CH. In particular, when defining the translation, we set the set of world atomic propositions to be the set of atomic propositions of $\mathsf{KB4}_n$. The translation of formulas is defined recursively as follows:

$$\ulcorner p \urcorner := p \qquad \ulcorner \neg\Phi \urcorner := \neg\ulcorner\Phi\urcorner \qquad \ulcorner \Phi \wedge \Psi \urcorner := \ulcorner\Phi\urcorner \wedge \ulcorner\Psi\urcorner \qquad \ulcorner K_a\Phi \urcorner := \mathsf{A}_a\mathsf{K}_a\ulcorner\Phi\urcorner$$

Essentially, this translation interprets the knowledge operator of $\mathsf{KB4}_n$ using the *unsafe knowledge* operator described in Section 2.4. So, if in a given world agent a is dead, $\ulcorner K_a\Phi\urcorner$ will be vacuously true.

▶ **Proposition 28.** *For a partial epistemic frame $\mathcal{M}$ and a formula Φ of $\mathsf{KB4}_n$, $\mathcal{M}, w \models \Phi$ iff $\eta(\mathcal{M}), \eta(w) \models_e \ulcorner\Phi\urcorner$.*

Proof. We show the statement by induction on the structure of Φ. For atomic propositions, as well as boolean connectives, the proof is trivial. For the modality, we have: $\mathcal{M}, w \models K_a\Phi$ if and only if for all $w' \in M$ such that $w \sim_a w'$, $\mathcal{M}, w' \models \Phi$. By induction, this is equivalent to for all $w' \in M$ such that $w \sim_a w'$, $\eta(\mathcal{M}), \eta(w') \models \ulcorner\Phi\urcorner$. By definition of η, it is the same as for all hyperedges $e \in \eta(\mathcal{M})_E$ that share an a vertex with $\eta(w)$, $\eta(\mathcal{M}), e \models \ulcorner\Phi\urcorner$. This is equivalent to $\eta(\mathcal{M}), \eta(w) \models \ulcorner K_a\Phi\urcorner$. ◀

▶ **Corollary 29.** *Φ is valid in a partial epistemic frame $\mathcal{M}$ iff $\ulcorner\Phi\urcorner$ is valid in $\eta(\mathcal{M})$.*

Using this and Theorem 23, we can show that 2CH is a conservative extension of $\mathsf{KB4}_n$:

▶ **Theorem 30.** *For every* $\mathsf{KB4}_n$*-formula* Φ, $\vdash_{\mathsf{KB4}_n+\mathsf{NE}} \Phi$ *if and only if* $\vdash_e \ulcorner\Phi\urcorner$.

Proof. By completeness, for $\mathsf{KB4}_n$ we have $\vdash_{\mathsf{KB4}_n+\mathsf{NE}} \Phi \Leftrightarrow \models \Phi$. By Corollary 29, we have $\models \Phi \Leftrightarrow \models_e \ulcorner\Phi\urcorner$. And by completeness for $\mathsf{2CH}$, we have $\models_e \ulcorner\Phi\urcorner \Leftrightarrow \vdash_e \ulcorner\Phi\urcorner$. ◀

As a corollary, we get that the combined modality $\mathsf{A}_a\mathsf{K}_a$ satisfies axioms K, B, and $\mathsf{4}$.

▶ Remark 31. Similarly, one can wonder which logic we would get if we translate formulas using instead the *safe knowledge* operator $K_a^{\text{safe}}\Phi = \mathsf{E}_a\mathsf{K}_a\Phi$. First thing to note is that this does *not* yield the three-valued logic $\mathsf{S5}_n^{\bowtie}$ of [18][3]. We can show that the safe knowledge modality satisfies axioms K and T. However, this modality is not normal as the necessitation rule is not admissible: true is valid in every world, but it is not the case that $\mathsf{E}_a\mathsf{K}_a\mathsf{true}$ is valid in every world. Thus, we cannot derive $\mathsf{E}_a\mathsf{K}_a\mathsf{true}$ from true.

4.3 Correspondence with neighborhood frames

Chromatic hypergraphs require that every hyperedge contains at most one vertex of each color. So in every world, an agent can have either 0 or 1 point of view. But what happens if we drop this condition and allow agents to have multiple points of view about a given world? Technically, this can be achieved by replacing, in Definition 7, the partial function $\mathsf{pr}_a : E \to V_a$ by a relation $\mathsf{pr}_a \subseteq E \times V_a$, i.e., get rid of the *functionality* requirement.

This leads to an intriguing connection with neighborhood frames [17]. In this subsection we will not prove formal results, but rather give an intuition of the connection between the two notions, applying a similar construction as in Section 4.1.

Neighborhood frames generalize epistemic frames by allowing agents to have multiple points of view on the same world, which on the side of hypergraphs corresponds exactly to the situation when we allow hyperedges to contain multiple vertices of the same color:

▶ **Definition 32** ([17]). *A neighborhood frame is a pair* $\mathsf{M} = (S, \{N_a\}_{a\in\mathcal{A}})$*, where* S *is a set of states, and for every agent* $a \in \mathcal{A}$*,* N_a *is a function that assigns to every state* $s \in S$*, a set* $N_a(s) \subseteq 2^S$ *called the* a*-neighborhoods of* s*.*

An example of a situation where neighborhood frames are required is as follows. Suppose we have two processes, communicating through shared memory. The memory has two cells, and each cell stores a bit of information: 0 or 1. Processes are given access to memory cells arbitrarily, and both can be assigned the same cell. They know which cell is assigned to them, and they know the value that is stored in this cell, that is, they read the value of the cell. Therefore, a process can have two points of view on the same situation, depending on which cell it is given access to. Assume for example that the shared memory stores values $(0, 1)$. Process a, when assigned the first cell, knows that the memory stores 0, and when assigned the second cell, knows that the memory stores 1. So the set of possible states is $S = \{(0, 0), (0, 1), (1, 0), (1, 1)\}$. In state $(0, 1)$, the two possible points of view of process a are described by neighborhoods: $N_a((0, 1)) = \{\{(0, 0), (0, 1)\}, \{(0, 1), (1, 1)\}\}$

We can also make sense of this example using generalized chromatic hypergraphs, where we allow hyperedges that contain multiple points of view of the same agent. In that case, we can model our example as follows. We have two agents, a and b. There are four hyperedges corresponding to four possible states of the memory: $(0, 0)$, $(0, 1)$, $(1, 0)$, and $(1, 1)$. Each agent has four possible points of view, depending on which memory cell is

[3] A translation of $\mathsf{S5}_n^{\bowtie}$ into $\mathsf{2CH}$ is possible, but it is more involved. One must first translate the well-definedness judgment, then set $\ulcorner K_a\Phi\urcorner := K_a^{\text{safe}}(\text{well-defined}(\Phi) \Rightarrow \ulcorner\Phi\urcorner)$.

assigned (left or right), and which bit is read (0 or 1). Let us denote the vertices of agent a by $\{(a, m, b) \mid m \in \{L, R\}, b \in \{0, 1\}\}$, and similarly for agent b. Then, a vertex (a, m, b) belongs to a hyperedge (x_L, x_R) if and only if $x_m = b$.

Recall the duality construction of Figure 3, switching the role of vertices and hyperedges. In our case, the set of hyperedges becomes the set of states, and the set of vertices defines the neighborhood function. In our example, the hyperedge/state $s = (0, a)$ contains two vertices of agent a: $(a, L, 0)$ and $(a, R, 1)$. The first vertex corresponds to the a-neighborhood $\{(0, 0), (0, 1)\}$, which is the set of hyperedges containing this vertex. Similarly, the second vertex corresponds to the neighborhood $\{(0, 1), (1, 1)\}$. This recovers the set $N_a(s)$ of the corresponding neighborhood frame. Note that we do not obtain all neighborhood frames in this way, but only those in which a world belongs to all of its neighborhoods.

5 Conclusion

In this paper, we proposed a many-sorted modal logic for reasoning about knowledge in multi-agent systems that treat as first-class citizens both participating agents and the environment. This allowed us to reconcile the numerous logics and models of the literature, which indeed struggled with expressing coherent general global properties of worlds and local properties of agents. There are two main extensions that we are currently studying based on this work. First, having points of view of agents as first-class citizens, a combination of epistemic logics with temporal modalities allows us to provide a framework with greater emphasis on local action of agents, compared to e.g., DEL [5] or interpreted systems [6]. Secondly, a natural question arises as to whether we can reconcile chromatic hypergraphs, with chromatic (semi-)simplicial sets as studied in e.g., [8]. This would allow us to naturally extend our logic with a distributed knowledge operator.

References

1 Samson Abramsky, Rui Soares Barbosa, Kohei Kishida, Raymond Lal, and Shane Mansfield. Contextuality, cohomology and paradox. In Stephan Kreutzer, editor, *24th EACSL Annual Conference on Computer Science Logic, CSL 2015*, volume 41 of *LIPIcs*, pages 211–228. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.CSL.2015.211.

2 Yifeng Ding, Jixin Liu, and Yanjing Wang. Someone knows that local reasoning on hypergraphs is a weakly aggregative modal logic. *Synthese*, 201(2):1–27, 2023. doi:10.1007/s11229-022-04032-y.

3 Hans van Ditmarsch. Wanted Dead or Alive: Epistemic Logic for Impure Simplicial Complexes. In Alexandra Silva, Renata Wassermann, and Ruy J. G. B. de Queiroz, editors, *Logic, Language, Information, and Computation - 27th International Workshop, WoLLIC 2021, Proceedings*, volume 13038 of *Lecture Notes in Computer Science*, pages 31–46. Springer, 2021. doi:10.1007/978-3-030-88853-4_3.

4 Hans van Ditmarsch, Éric Goubault, Jérémy Ledent, and Sergio Rajsbaum. Knowledge and simplicial complexes. In Björn Lundgren and Nancy Abigail Nuñez Hernández, editors, *Philosophy of Computing*, volume 143, pages 1–50, Cham, 2022. Springer International Publishing. doi:10.1007/978-3-030-75267-5_1.

5 Hans van Ditmarsch, Wiebe van der Hoek, and Barteld Kooi. *Dynamic Epistemic Logic*, volume 337 of *Synthese Library*. Springer, 2007. doi:10.1007/978-1-4020-5839-4.

6 Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning About Knowledge*. MIT Press, Cambridge, MA, USA, 2003.

7 Jelle Gerbrandy and Willem Groeneveld. Reasoning about information change. *Journal of Logic, Language and Information*, 6:147–169, 1997.

8 Éric Goubault, Roman Kniazev, Jérémy Ledent, and Sergio Rajsbaum. Semi-simplicial set models for distributed knowledge. In *38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–13, 2023. doi:10.1109/LICS56636.2023.10175737.

9 Éric Goubault, Jérémy Ledent, and Sergio Rajsbaum. A simplicial complex model for dynamic epistemic logic to study distributed task computability. *Inf. Comput.*, 278:104597, 2021. doi:10.1016/j.ic.2020.104597.

10 Éric Goubault, Jérémy Ledent, and Sergio Rajsbaum. A simplicial model for KB4: Epistemic logic with agents that may die. In *39th International Symposium on Theoretical Aspects of Computer Science, STACS 2022*, pages 33:1–33:20, 2022. doi:10.4230/LIPIcs.STACS.2022.33.

11 Joseph Y. Halpern and Yoram Moses. Knowledge and common knowledge in a distributed environment. *J. ACM*, 37(3):549–587, 1990. doi:10.1145/79147.79161.

12 Joseph Y. Halpern and Moshe Y. Vardi. The complexity of reasoning about knowledge and time. I. lower bounds. *J. Comput. Syst. Sci.*, 38(1):195–237, 1989. doi:10.1016/0022-0000(89)90039-1.

13 Maurice Herlihy, Dmitry Kozlov, and Sergio Rajsbaum. *Distributed Computing Through Combinatorial Topology*. Morgan Kaufmann, San Francisco, CA, USA, 2013.

14 Maurice Herlihy and Sergio Rajsbaum. Algebraic topology and distributed computing: A primer. In Jan van Leeuwen, editor, *Computer Science Today: Recent Trends and Developments*, volume 1000 of *Lecture Notes in Computer Science*, pages 203–217. Springer, 1995. doi:10.1007/BFb0015245.

15 Maurice Herlihy, Sergio Rajsbaum, and Mark R. Tuttle. An overview of synchronous message-passing and topology. *Electronic Notes in Theoretical Computer Science*, 39(2):1–17, 2000. doi:10.1016/S1571-0661(05)01148-5.

16 Jaakko Hintikka. *Knowledge and Belief*. Cornell University Press, 1962.

17 E. Pacuit. *Neighborhood Semantics for Modal Logic*. Short Textbooks in Logic. Springer International Publishing, 2017. URL: https://books.google.co.uk/books?id=WK4-DwAAQBAJ.

18 Rojo Fanamperana Randrianomentsoa, Hans van Ditmarsch, and Roman Kuznets. Impure simplicial complexes: Complete axiomatization. *CoRR*, abs/2211.13543, 2022. doi:10.48550/arXiv.2211.13543.

19 Mehrnoosh Sadrzadeh and Roy Dyckhoff. Positive logic with adjoint modalities: Proof theory, semantics and reasoning about information. *Electronic Notes in Theoretical Computer Science*, 249:451–470, 2009. Proceedings of the 25th Conference on Mathematical Foundations of Programming Semantics (MFPS 2009). doi:10.1016/j.entcs.2009.07.102.

A Proofs

A.1 Proof of Proposition 14

Proof. First, we show that universal modalities distribute over conjunction, that is $\heartsuit(\xi \wedge \eta) \leftrightarrow (\heartsuit\xi \wedge \heartsuit\eta)$. Left-to-right direction: we have that $(\xi \wedge \eta) \to \xi$ and $(\xi \wedge \eta) \to \eta$. Applying the RM rule, we get that $\heartsuit(\xi \wedge \eta) \to \heartsuit\xi$ and $\heartsuit(\xi \wedge \eta) \to \heartsuit\eta$. From this, the left-to-right direction follows. Right-to-left direction: denote the modality adjoint to $\heartsuit$ by $\spadesuit$, that is, if $\heartsuit = \mathsf{A}_a$ then $\spadesuit = \widehat{\mathsf{K}}_a$ and if $\heartsuit = \mathsf{K}_a$ then $\spadesuit = \mathsf{E}_a$. We have that $\spadesuit\heartsuit\xi \to \xi$ from $\heartsuit\xi \to \heartsuit\xi$ and the corresponding adjunction rule, similarly for η. From this we have that $\spadesuit\heartsuit\xi \wedge \spadesuit\heartsuit\eta \to \xi \wedge \eta$. By the same proof as in the left-to-right direction, we have that $\spadesuit(\heartsuit\xi \wedge \heartsuit\eta) \to \spadesuit\heartsuit\xi \wedge \spadesuit\heartsuit\eta$. Thus, we have that $\spadesuit(\heartsuit\xi \wedge \heartsuit\eta) \to \xi \wedge \eta$. Applying the adjunction rule, we get that $\heartsuit(\xi \wedge \eta) \to \heartsuit\xi \wedge \heartsuit\eta$, which is the right-to-left direction.

The fact that K follows from the distribution of universal modalities over conjunction is a standard proof: From $((\xi \to \eta) \wedge \xi) \to \eta$ by RM we have $\heartsuit((\xi \to \eta) \wedge \xi) \to \heartsuit\eta$. By distribution, we have $(\heartsuit(\xi \to \eta) \wedge \heartsuit\xi) \to \heartsuit((\xi \to \eta) \wedge \xi)$. Combining these two, we get $(\heartsuit(\xi \to \eta) \wedge \heartsuit\xi) \to \heartsuit\eta$, and thus $\heartsuit(\xi \to \eta) \to (\heartsuit\xi \to \heartsuit\eta)$. ◀

A.2 Proof of Proposition 15

Proof. Recall the list of formulas:

1. $\mathsf{E}_a\varphi \to \mathsf{A}_a\varphi$;
2. $\mathsf{K}_a\Phi \to \mathsf{K}_a\mathsf{E}_a\mathsf{K}_a\Phi$;
3. $\mathsf{E}_a\mathsf{K}_a\Phi \to \Phi$;
4. $\Phi \to \mathsf{A}_a\widehat{\mathsf{K}}_a\Phi$;
5. $\varphi \to \mathsf{K}_a\mathsf{E}_a\varphi$;
6. $\mathsf{K}_a\Phi \to \widehat{\mathsf{K}}_a\Phi$.

For the first formula, we just apply the adjunction rule to $\widehat{\mathsf{K}}_a\mathsf{E}_a\varphi \to \varphi$, which is an axiom. In order to show the second formula, just apply the adjunction rule to $\mathsf{E}_a\mathsf{K}_a\Phi \to \mathsf{E}_a\mathsf{K}_a\Phi$, which is a tautology. Formulas 3, 4 and 5 are derived from $\mathsf{K}_a\Phi \to \mathsf{K}_a\Phi$, $\widehat{\mathsf{K}}_a\Phi \to \widehat{\mathsf{K}}_a\Phi$, and $\mathsf{E}_a\varphi \to \mathsf{E}_a\varphi$ respectively by applying the adjunction rule. The last formula is shown as follows: from formulas 3 and 4 we have $\mathsf{E}_a\mathsf{K}_a\Phi \to \mathsf{A}_a\widehat{\mathsf{K}}_a\Phi$. Applying the adjunction rule, we get $\widehat{\mathsf{K}}_a\mathsf{E}_a\mathsf{K}_a\Phi \to \widehat{\mathsf{K}}_a\Phi$. We also have $\mathsf{K}_a\Phi \to \widehat{\mathsf{K}}_a\mathsf{E}_a\mathsf{K}_a\Phi$, which is an axiom. From the last two formulas, we get $\mathsf{K}_a\Phi \to \widehat{\mathsf{K}}_a\Phi$. ◀

A.3 Proof of Theorem 26

Proof. We just need to show how η and κ are extended to morphisms. Functoriality is then straightforward, and checking that $\eta(\kappa(H)) \simeq H$ and $\kappa(\eta(\mathcal{M})) \simeq \mathcal{M}$ is also straightforward. Let $f : H \to H'$ be a morphism of chromatic hypergraphs. Then $\eta(f) : \eta(H) \to \eta(H')$ just sends a world e to $f_E(e)$. This is indeed a morphism of partial epistemic frames: suppose two worlds e and e' in $\eta(H)$ are $\sim_a$-equivalent. It means that in H these two hyperedges share a vertex, and thus in H' the two hyperedges $f_E(e)$ and $f_E(e')$ share a vertex, and thus $f_E(e) \sim_a f_E(e')$.

Now let $g : \mathcal{M} \to \mathcal{M}'$ be a morphism of partial epistemic frames. Then $\kappa(g) : \kappa(\mathcal{M}) \to \kappa(\mathcal{M}')$ sends a hyperedge w to $g(w)$, thus $\kappa(g)_E$ is defined. We need to show that it induces a map on vertices, and that the condition for morphsisms is satisfied. As g preserves $\sim_a$, it induces a map on equivalence classes, which is exactly $\kappa(g)_a$. Let w be a hyperedge and v be a vertex in $\kappa(\mathcal{M})$, such that $\mathsf{pr}_a(w) = v$. It means that w belongs to the equivalence class corresponding to v in $\mathcal{M}$. Thus, g(w) belongs to the equivalence class corresponding to $g(v)$ in $\mathcal{M}'$, and thus $\mathsf{pr}_a(g(w)) = g(v)$. ◀

Remarks on Parikh-Recognizable Omega-languages

Mario Grobler
University of Bremen, Germany

Leif Sabellek
University of Bremen, Germany

Sebastian Siebertz
University of Bremen, Germany

Abstract

Several variants of Parikh automata on infinite words were recently introduced by Guha et al. [FSTTCS, 2022]. We show that one of these variants coincides with blind counter machine as introduced by Fernau and Stiebe [Fundamenta Informaticae, 2008]. Fernau and Stiebe showed that every ω-language recognized by a blind counter machine is of the form $\bigcup_i U_i V_i^\omega$ for Parikh recognizable languages U_i, V_i, but blind counter machines fall short of characterizing this class of ω-languages. They posed as an open problem to find a suitable automata-based characterization. We introduce several additional variants of Parikh automata on infinite words that yield automata characterizations of classes of ω-language of the form $\bigcup_i U_i V_i^\omega$ for all combinations of languages U_i, V_i being regular or Parikh-recognizable. When both U_i and V_i are regular, this coincides with Büchi's classical theorem. We study the effect of ε-transitions in all variants of Parikh automata and show that almost all of them admit ε-elimination. Finally we study the classical decision problems with applications to model checking.

2012 ACM Subject Classification Theory of computation → Automata over infinite objects

Keywords and phrases Parikh automata, blind counter machines, infinite words, Büchi's theorem

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.31

Related Version This paper subsumes the unpublished results of [19, 21].
Full Version: `https://arxiv.org/abs/2307.07238` [20]

Acknowledgements We thank Georg Zetzsche for his valuable remarks.

1 Introduction

Finite automata find numerous applications in formal language theory, logic, verification, and many more, in particular due to their good closure properties and algorithmic properties. To enrich this spectrum of applications even more, it has been a fruitful direction to add features to finite automata to capture also situations beyond the regular realm.

One such possible extension of finite automata with counting mechanisms has been introduced by Greibach in her study of blind and partially blind (one-way) multicounter machines [18]. Blind multicounter machines are generalized by weighted automata as introduced in [28]. Parikh automata (PA) were introduced by Klaedtke and Rueß in [26]. A PA is a non-deterministic finite automaton that is additionally equipped with a semi-linear set C, and every transition is equipped with a d-tuple of non-negative integers. Whenever an input word is read, d counters are initialized with the values 0 and every time a transition is used, the counters are incremented by the values in the tuple of the transition accordingly. An input word is accepted if the PA ends in an accepting state and additionally, the resulting d-tuple of counter values lies in C. Klaedtke and Rueß showed that PA are equivalent to weighted automata over the group $(\mathbb{Z}^k, +, \mathbf{0})$, and hence equivalent to Greibach's blind multicounter machines, as well as to reversal bounded multicounter machines [2, 24]. Recently it was shown that these models can be translated into each other

using only logarithmic space [3]. In this work we call the class of languages recognized by any of these models *Parikh recognizable*. Klaedtke and Rueß [26] showed that the class of Parikh recognizable languages is precisely the class of languages definable in weak existential monadic second-order logic of one successor extended with linear cardinality constraints. The class of Parikh-recognizable languages contains all regular languages, but also many more, even languages that are not context-free, e. g., the language $\{a^n b^n c^n \mid n \in \mathbb{N}\}$. On the other hand, the language of palindromes is context-free, but not Parikh-recognizable. On finite words, blind counter automata, Parikh automata and related models have been investigated extensively, extending [18, 26] for example by affine PA and PA on letters [6, 7], bounded PA [8], two-way PA [16], PA with a pushdown stack [25] as well as a combination of both [11], history-deterministic PA [12], automata and grammars with valences [13, 23], and several algorithmic applications, e.g. in the context of path logics for querying graphs [15].

In the well-studied realm of verification of reactive systems, automata-related approaches provide a powerful framework to tackle important problems such as the model checking problem [1, 9, 10]. However, computations of systems are generally represented as infinite objects, as we often expect them to not terminate (but rather interact with the environment). Hence, automata processing infinite words are suited for these tasks. One common approach is the following: assume we are given a system, e.g. represented as a Kripke structure K, and a specification represented as an automaton $\mathcal{A}$ (or any formalism that can be translated into one) accepting all counterexamples. Then we can verify that the system has no bad computations by solving intersection-emptiness of K and $\mathcal{A}$. Yet again, the most basic model of Büchi automata (which recognize ω-regular languages) are quite limited in their expressiveness, although they have nice closure properties.

Let us consider two examples. In a three-user setting in an operating system we would like to ensure that none of the users gets a lot more resources than the other two. A corresponding specification of bad computations can be modeled via the ω-language $\{\alpha \in \{a, b, c\}^\omega \mid$ there are infinitely many prefixes w of α with $|w|_a > |w|_b + |w|_c\}$, stating that one user gets more resources than the other two users combined infinitely often. As another example, consider a classical producer-consumer setting, where a producer continuously produces a good, and a consumer consumes these goods continuously. We can model this setting as an infinite word and ask that at no time the consumer has consumed more than the producer has produced at this time. Bad computations can be modeled via the ω-language $\{\alpha \in \{p, c\}^\omega \mid$ there is a prefix w of α with $|w|_c > |w|_p\}$. Such specifications are not ω-regular, as these require to "count arbitrarily". This motivates the study of blind-counter and Parikh automata on infinite words, which was initiated by Fernau and Stiebe [14]. Independently, Klaedte and Rueß proposed possible extensions of Parikh automata on infinite words. This line of research was recently picked up by Guha et al. [22].

Guha et al. [22] introduced *safety, reachability, Büchi- and co-Büchi Parikh automata*. These models provide natural generalization of studied automata models with Parikh conditions on infinite words. One shortcoming of safety, reachability and co-Büchi Parikh automata is that they do not generalize Büchi automata, that is, they cannot recognize all ω-regular languages. The non-emptiness problem, which is highly relevant for model checking applications, is undecidable for safety and co-Büchi Parikh automata. Furthermore, none of these models has ω-closure, meaning that for every model there is a Parikh-recognizable language (on finite words) L such that L^ω is not recognizable by any of these models. Guha et al. raised the question whether (appropriate variants of) Parikh automata on infinite words have the same expressive power as blind counter automata on infinite words.

Büchi's famous theorem states that ω-regular languages are characterized as languages of the form $\bigcup_i U_i V_i^\omega$, where the U_i and V_i are regular languages [4]. As a consequence of the theorem, many properties of ω-regular languages are inherited from regular languages. For example, the non-emptiness problem for Büchi automata can basically be solved by testing non-emptiness for nondeterministic finite automata. In their systematic study of blind counter automata, Fernau and Stiebe [14] considered the class $\mathcal{K}_*$, the class of ω-languages of the form $\bigcup_i U_i V_i^\omega$ for Parikh-recognizable languages U_i and V_i. They proved that the class of ω-languages recognizable by blind counter machines is a proper subset of the class $\mathcal{K}_*$. They posed as an open problem to provide automata models that capture classes of ω-languages of the form $\bigcup_i U_i V_i^\omega$ where U_i and V_i are described by a certain mechanism.

In this work we propose *reachability-regular Parikh automata*, *limit Parikh automata*, and *reset Parikh automata* as new automata models.

We pick up the question of Fernau and Stiebe [14] to consider classes of ω-languages of the form $\bigcup_i U_i V_i^\omega$ where U_i and V_i are described by a certain mechanism. We define the four classes $\mathcal{L}^\omega_{\mathsf{Reg},\mathsf{Reg}}$, $\mathcal{L}^\omega_{\mathsf{PA},\mathsf{Reg}}$, $\mathcal{L}^\omega_{\mathsf{Reg},\mathsf{PA}}$ and $\mathcal{L}^\omega_{\mathsf{PA},\mathsf{PA}}$ of ω-languages of the form $\bigcup_i U_i V_i^\omega$, where the U_i, V_i are regular or Parikh-recognizable languages of finite words, respectively. By Büchi's theorem the class $\mathcal{L}^\omega_{\mathsf{Reg},\mathsf{Reg}}$ is the class of ω-regular languages.

We show that the newly introduced reachability-regular Parikh automata, which are a small modification of reachability Parikh automata (as introduced by Guha et al. [22]) capture exactly the class $\mathcal{L}^\omega_{\mathsf{PA},\mathsf{Reg}}$. This model turns out to be equivalent to limit Parikh automata. This model was hinted at in the concluding remarks of [26].

Fully resolving the classification of the above mentioned classes we introduce reset Parikh automata. In contrast to all other Parikh models, these are closed under the ω-operation, while maintaining all algorithmic properties of PA (in particular, non-emptiness is NP-complete and hence decidable). We show that the class of Reset-recognizable ω-languages is a strict superclass of $\mathcal{L}^\omega_{\mathsf{PA},\mathsf{PA}}$. We show that appropriate graph-theoretic restrictions of reset Parikh automata exactly capture the classes $\mathcal{L}^\omega_{\mathsf{PA},\mathsf{PA}}$ and $\mathcal{L}^\omega_{\mathsf{Reg},\mathsf{PA}}$, yielding the first automata characterizations for these classes.

The automata models introduced by Guha et al. [22] do not have ε-transitions, while blind counter machines have such transitions. Towards answering the question of Guha et al. we study the effect of ε-transitions in all Parikh automata models. We show that all models except safety and co-Büchi Parikh automata admit ε-elimination. This in particular answers the question of Guha et al. [22] whether blind counter automata and Büchi Parikh automata have the same expressive power over infinite words affirmative. We show that safety and co-Büchi automata with ε-transitions are strictly more powerful than their variants without ε-transitions, and in particular, they give the models enough power to recognize all ω-regular languages.

All lemmas with missing proofs are marked with ($\star$), the full version [20] containing all proofs can be found on arXiv.

2 Preliminaries

2.1 Finite and infinite words

We write $\mathbb{N}$ for the set of non-negative integers including 0, and $\mathbb{Z}$ for the set of all integers. Let Σ be an alphabet, i. e., a finite non-empty set and let Σ^* be the set of all finite words over Σ. For a word $w \in \Sigma^*$, we denote by $|w|$ the length of w, and by $|w|_a$ the number of occurrences of the letter $a \in \Sigma$ in w. We write ε for the empty word of length 0.

An *infinite word* over an alphabet Σ is a function $\alpha : \mathbb{N} \setminus \{0\} \to \Sigma$. We often write α_i instead of $\alpha(i)$. Thus, we can understand an infinite word as an infinite sequence of symbols $\alpha = \alpha_1\alpha_2\alpha_3\ldots$ For $m \leq n$, we abbreviate the finite infix $\alpha_m \ldots \alpha_n$ by $\alpha[m, n]$. We denote by Σ^ω the set of all infinite words over Σ. We call a subset $L \subseteq \Sigma^\omega$ an *ω-language*. Moreover, for $L \subseteq \Sigma^*$, we define $L^\omega = \{w_1w_2\cdots \mid w_i \in L \setminus \{\varepsilon\}\} \subseteq \Sigma^\omega$.

2.2 Regular and ω-regular languages

A *nondeterministic finite automaton* (NFA) is a tuple $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$, where Q is the finite set of states, Σ is the input alphabet, $q_0 \in Q$ is the initial state, $\Delta \subseteq Q \times \Sigma \times Q$ is the set of transitions and $F \subseteq Q$ is the set of accepting states. A *run* of $\mathcal{A}$ on a word $w = w_1 \ldots w_n \in \Sigma^*$ is a (possibly empty) sequence of transitions $r = r_1 \ldots r_n$ with $r_i = (p_{i-1}, w_i, p_i) \in \Delta$ such that $p_0 = q_0$. We say r is *accepting* if $p_n \in F$. The empty run on ε is accepting if $q_0 \in F$. We define the *language recognized by* $\mathcal{A}$ as $L(\mathcal{A}) = \{w \in \Sigma^* \mid$ there is an accepting run of $\mathcal{A}$ on $w\}$. If a language L is recognized by some NFA $\mathcal{A}$, we call L *regular*.

A *Büchi automaton* is an NFA $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$ that takes infinite words as input. A *run* of $\mathcal{A}$ on an infinite word $\alpha_1\alpha_2\alpha_3\ldots$ is an infinite sequence of transitions $r = r_1r_2r_3\ldots$ with $r_i = (p_{i-1}, \alpha_i, p_i) \in \Delta$ such that $p_0 = q_0$. We say r is *accepting* if there are infinitely many i with $p_i \in F$. We define the *ω-language recognized by* $\mathcal{A}$ as $L_\omega(\mathcal{A}) = \{\alpha \in \Sigma^\omega \mid$ there is an accepting run of $\mathcal{A}$ on $\alpha\}$. If an ω-language L is recognized by some Büchi automaton $\mathcal{A}$, we call L *ω-regular*. Büchi's theorem establishes an important connection between regular and ω-regular languages:

▶ **Theorem 1** (Büchi [4]). *A language $L \subseteq \Sigma^\omega$ is ω-regular if and only if there are regular languages $U_1, V_1, \ldots, U_n, V_n \subseteq \Sigma^*$ for some $n \geq 1$ such that $L = U_1V_1^\omega \cup \cdots \cup U_nV_n^\omega$.*

If every state of a Büchi automaton $\mathcal{A}$ is accepting, we call $\mathcal{A}$ a *safety automaton*.

2.3 Semi-linear sets

For some $d \geq 1$, a *linear set* of dimension d is a set of the form $\{b_0 + b_1z_1 + \cdots + b_\ell z_\ell \mid z_1, \ldots, z_\ell \in \mathbb{N}\} \subseteq \mathbb{N}^d$ for $b_0, \ldots, b_\ell \in \mathbb{N}^d$. If $b_0 = \mathbf{0}$, then we call C a *homogeneous* linear set. A *semi-linear set* is a finite union of linear sets. For vectors $\mathbf{u} = (u_1, \ldots, u_c) \in \mathbb{N}^c$ and $\mathbf{v} = (v_1, \ldots, v_d) \in \mathbb{N}^d$, we denote by $\mathbf{u} \cdot \mathbf{v} = (u_1, \ldots, u_c, v_1, \ldots, v_d) \in \mathbb{N}^{c+d}$ the *concatenation of* $\mathbf{u}$ *and* $\mathbf{v}$. We extend this definition to sets of vectors. Let $C \subseteq \mathbb{N}^c$ and $D \subseteq \mathbb{N}^d$. Then $C \cdot D = \{\mathbf{u} \cdot \mathbf{v} \mid \mathbf{u} \in C, \mathbf{v} \in D\} \subseteq \mathbb{N}^{c+d}$. We denote by $\mathbf{0}^d$ (or simply $\mathbf{0}$ if d is clear from the context) the all-zero vector, and by $\mathbf{e}_i^d$ (or simply $\mathbf{e}_i$) the d-dimensional vector where the ith entry is 1 and all other entries are 0. We also consider semi-linear sets over $(\mathbb{N} \cup \{\infty\})^d$, that is semi-linear sets with an additional symbol ∞ for infinity. As usual, addition of vectors and multiplication of a vector with a number is defined component-wise, where $z + \infty = \infty + z = \infty + \infty = \infty$ for all $z \in \mathbb{N}$, $z \cdot \infty = \infty \cdot z = \infty$ for all $z > 0 \in \mathbb{N}$, and $0 \cdot \infty = \infty \cdot 0 = 0$.

2.4 Parikh-recognizable languages

A *Parikh automaton* (PA) is a tuple $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F, C)$ where Q, Σ, q_0, and F are defined as for NFA, $\Delta \subseteq Q \times \Sigma \times \mathbb{N}^d \times Q$ is a finite set of *labeled transitions*, and $C \subseteq \mathbb{N}^d$ is a semi-linear set. We call d the *dimension* of $\mathcal{A}$ and refer to the entries of a vector $\mathbf{v}$ in a transition $(p, a, \mathbf{v}, q)$ as *counters*. Similar to NFA, a *run* of $\mathcal{A}$ on a word $w = x_1 \ldots x_n$ is a (possibly empty) sequence of labeled transitions $r = r_1 \ldots r_n$ with $r_i = (p_{i-1}, x_i, \mathbf{v}_i, p_i) \in \Delta$

such that $p_0 = q_0$. We define the *extended Parikh image* of a run r as $\rho(r) = \sum_{i \leq n} \mathbf{v}_i$ (with the convention that the empty sum equals $\mathbf{0}$). We say r is accepting if $p_n \in F$ and $\rho(r) \in C$, referring to the latter condition as the *Parikh condition*. We define the *language recognized by* $\mathcal{A}$ as $L(\mathcal{A}) = \{w \in \Sigma^* \mid \text{there is an accepting run of } \mathcal{A} \text{ on } w\}$. If a language $L \subseteq \Sigma^*$ is recognized by some PA, then we call L *Parikh-recognizable*.

2.5 Graphs

A *(directed) graph* G consists of its vertex set $V(G)$ and edge set $E(G) \subseteq V(G) \times V(G)$. In particular, a graph G may have loops, that is, edges of the form (u, u). A *(simple) path* from a vertex u to a vertex v in G is a sequence of pairwise distinct vertices $v_1 \dots v_k$ such that $v_1 = u$, $v_k = v$, and $(v_i, v_{i+1}) \in E(G)$ for all $1 \leq i < k$. Similarly, a *(simple) cycle* in G is a sequence of pairwise distinct vertices $v_1 \dots v_k$ such that $(v_i, v_{i+1}) \in E(G)$ for all $1 \leq i < k$, and $(v_k, v_1) \in E(G)$. If G has no cylces, we call G a directed acyclic graph (DAG). For a subset $U \subseteq V(G)$, we denote by $G[U]$ the graph G *induced by* U, i. e., the graph with vertex set U and edge set $\{(u, v) \in E(G) \mid u, v \in U\}$. A *strongly connected component* (SCC) in G is a maximal subset $U \subseteq V(G)$ such that for all $u, v \in U$ there is a path from u to v, i. e., all vertices in U are reachable from each other. We write $SCC(G)$ for the set of all strongly connected components of G (observe that $SCC(G)$ partitions $V(G)$). The *condensation* of G, written $C(G)$, is the DAG obtained from G by contracting each SCC of G into a single vertex, that is $V(C(G)) = SCC(G)$ and $(U, V) \in E(C(G))$ if and only if there is $u \in U$ and $v \in V$ with $(u, v) \in E(G)$. We call the SCCs with no outgoing edges in $C(G)$ leaves. Note that an automaton can be seen as a labeled graph. Hence, all definitions translate to automata by considering the underlying graph (to be precise, an automaton can be seen as a labeled multigraph; however, we simply drop parallel edges).

3 Parikh automata on infinite words

In this section, we recall the acceptance conditions of Parikh automata operating on infinite words that were studied before in the literature and introduce our new models. We make some easy observations and compare the existing with the new automata models. We define only the non-deterministic variants of these automata.

Let $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F, C)$ be a PA. A run of $\mathcal{A}$ on an infinite word $\alpha = \alpha_1\alpha_2\alpha_3 \dots$ is an infinite sequence of labeled transitions $r = r_1r_2r_3 \dots$ with $r_i = (p_{i-1}, \alpha_i, \mathbf{v}_i, p_i) \in \Delta$ such that $p_0 = q_0$. The automata defined below differ only in their acceptance conditions. In the following, whenever we say that an automaton $\mathcal{A}$ accepts an infinite word α, we mean that there is an accepting run of $\mathcal{A}$ on α.

1. The run r satisfies the *safety condition* if for every $i \geq 0$ we have $p_i \in F$ and $\rho(r_1 \dots r_i) \in C$. We call a PA accepting with the safety condition a *safety PA* [22]. We define the ω-language recognized by a safety PA $\mathcal{A}$ as $S_\omega(\mathcal{A}) = \{\alpha \in \Sigma^\omega \mid \mathcal{A} \text{ accepts } \alpha\}$.
2. The run r satisfies the *reachability condition* if there is an $i \geq 1$ such that $p_i \in F$ and $\rho(r_1 \dots r_i) \in C$. We say there is an *accepting hit* in r_i. We call a PA accepting with the reachability condition a *reachability PA* [22]. We define the ω-language recognized by a reachability PA $\mathcal{A}$ as $R_\omega(\mathcal{A}) = \{\alpha \in \Sigma^\omega \mid \mathcal{A} \text{ accepts } \alpha\}$.
3. The run r satisfies the *Büchi condition* if there are infinitely many $i \geq 1$ such that $p_i \in F$ and $\rho(r_1 \dots r_i) \in C$. We call a PA accepting with the Büchi condition a *Büchi PA* [22]. We define the ω-language recognized by a Büchi PA $\mathcal{A}$ as $B_\omega(\mathcal{A}) = \{\alpha \in \Sigma^\omega \mid \mathcal{A} \text{ accepts } \alpha\}$. Hence, a Büchi PA can be seen as a stronger variant of a reachability PA where we require infinitely many accepting hits instead of a single one.

4. The run r satisfies the *co-Büchi condition* if there is i_0 such that for every $i \geq i_0$ we have $p_i \in F$ and $\rho(r_1 \dots r_i) \in C$. We call a PA accepting with the co-Büchi condition a *co-Büchi PA* [22]. We define the ω-language recognized by a co-Büchi PA $\mathcal{A}$ as $CB_\omega(\mathcal{A}) = \{\alpha \in \Sigma^\omega \mid \mathcal{A} \text{ accepts } \alpha\}$.
   Hence, a co-Büchi PA can be seen as a weaker variant of safety PA where the safety condition needs not necessarily be fulfilled from the beginning, but from some point onwards.

Guha et al. [22] assume that reachability PA are complete, i.e., for every $(p, a) \in Q \times \Sigma$ there are $\mathbf{v} \in \mathbb{N}^d$ and $q \in Q$ such that $(p, a, \mathbf{v}, q) \in \Delta$, as incompleteness allows to express additional safety conditions. We also make this assumption in order to study "pure" reachability PA. In fact, we can assume that all models are complete, as the other models can be completed by adding a non-accepting sink. We remark that Guha et al. also considered *asynchronous* reachability and Büchi PA, where the Parikh condition does not necessarily need to be satisfied in accepting states. However, for non-deterministic automata this does not change the expressiveness of the considered models [22].

We now define the models newly introduced in this work. As already observed in [22] among the above considered models only Büchi PA can recognize all ω-regular languages. For example, $\{\alpha \in \{a, b\}^\omega \mid |\alpha|_a = \infty\}$ cannot be recognized by safety PA, reachability PA or co-Büchi PA.

We first extend reachability PA with the classical Büchi condition to obtain *reachability-regular PA*. In Theorem 9 we show that these automata characterize ω-languages of the form $\mathcal{L}^\omega_{\mathsf{PA},\mathsf{Reg}}$, hence, providing a robust and natural model.

5. The run satisfies the *reachability and regularity condition* if there is an $i \geq 1$ such that $p_i \in F$ and $\rho(r_1 \dots r_i) \in C$, and there are infinitely many $j \geq 1$ such that $p_j \in F$. We call a PA accepting with the reachability and regularity condition a *reachability-regular* PA. We define the ω-language recognized by a reachability-regular PA $\mathcal{A}$ as $RR_\omega(\mathcal{A}) = \{\alpha \in \Sigma^\omega \mid \mathcal{A} \text{ accepts } \alpha\}$ and call it reachability-regular.

Note that (in contrast to reachability PA) we may assume that reachability-regular PA are complete without changing their expressiveness. Observe that every ω-regular language is reachability-regular, as we can turn an arbitrary Büchi automaton into an equivalent reachability-regular PA by labeling every transition with 0 and setting $C = \{0\}$.

We next introduce *limit PA*, which were proposed in the concluding remarks of [26]. As we will prove in Theorem 9, this seemingly quite different model is equivalent to reachability-regular PA.

6. The run satisfies the *limit condition* if there are infinitely many $i \geq 1$ such that $p_i \in F$, and if additionally $\rho(r) \in C$, where the jth component of $\rho(r)$ is computed as follows. If there are infinitely many $i \geq 1$ such that the jth component of $\mathbf{v}_i$ has a non-zero value, then the jth component of $\rho(r)$ is ∞. In other words, if the sum of values in a component diverges, then its value is set to ∞. Otherwise, the infinite sum yields a positive integer. We call a PA accepting with the limit condition a *limit PA*. We define the ω-language recognized by a limit PA $\mathcal{A}$ as $L_\omega(\mathcal{A}) = \{\alpha \in \Sigma^\omega \mid \mathcal{A} \text{ accepts } \alpha\}$.

Still, none of the yet introduced models have ω-closure. This shortcoming is addressed with the following two models, which will turn out to be equivalent and form the basis of the automata characterization of $\mathcal{L}^\omega_{\mathsf{Reg},\mathsf{PA}}$ and $\mathcal{L}^\omega_{\mathsf{PA},\mathsf{PA}}$.

7. The run satisfies the *strong reset condition* if the following holds. Let $k_0 = 0$ and denote by $k_1 < k_2 < \dots$ the positions of all accepting states in r. Then r is accepting if $k_1, k_2, \dots$ is an infinite sequence and $\rho(r_{k_{i-1}+1} \dots r_{k_i}) \in C$ for all $i \geq 1$. We call a PA accepting with the strong reset condition a *strong reset PA*. We define the ω-language recognized by a strong reset PA $\mathcal{A}$ as $SR_\omega(\mathcal{A}) = \{\alpha \in \Sigma^\omega \mid \mathcal{A} \text{ accepts } \alpha\}$.
8. The run satisfies the *weak reset condition* if there are infinitely many reset positions $0 = k_0 < k_1 < k_2, \dots$ such that $p_{k_i} \in F$ and $\rho(r_{k_{i-1}+1} \dots r_{k_i}) \in C$ for all $i \geq 1$. We call a PA accepting with the weak reset condition a *weak reset PA*. We define the ω-language recognized by a weak reset PA $\mathcal{A}$ as $WR_\omega(\mathcal{A}) = \{\alpha \in \Sigma^\omega \mid \mathcal{A} \text{ accepts } \alpha\}$.

Intuitively worded, whenever a strong reset PA enters an accepting state, the Parikh condition *must* be satisfied. Then the counters are reset. Similarly, a weak reset PA may reset the counters whenever there is an accepting hit, and they must reset infinitely often, too. In the following we will often just speak of reset PA without explicitly stating whether they are weak or strong. In this case, we mean the strong variant. We will show the equivalence of the two models in Lemma 26 and Lemma 27.

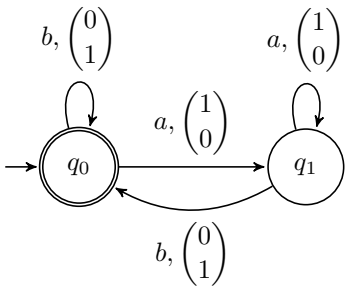


Figure 1 The automaton $\mathcal{A}$ with $C = \{(z, z'), (z, \infty) \mid z' \geq z\}$ from Example 1.

▶ **Example 1.** Let $\mathcal{A}$ be the automaton in Figure 1 with $C = \{(z, z'), (z, \infty) \mid z' \geq z\}$.

- If we interpret $\mathcal{A}$ as a PA (over finite words), then we have $L(\mathcal{A}) = \{w \in \{a, b\}^* \cdot \{b\} \mid |w|_a \leq |w|_b\} \cup \{\varepsilon\}$. The automaton is in the accepting state at the very beginning and every time after reading a b. The first counter counts the occurrences of letter a, the second one counts occurrences of b. By definition of C the automaton only accepts when the second counter value is greater or equal to the first counter value (note that vectors containing an ∞-entry have no additional effect).
- If we interpret $\mathcal{A}$ as a safety PA, then we have $S_\omega(\mathcal{A}) = \{b\}^\omega$. As q_1 is not accepting, only the b-loop on q_0 may be used.
- If we interpret $\mathcal{A}$ as a reachability PA, then we have $R_\omega(\mathcal{A}) = \{\alpha \in \{a, b\}^\omega \mid \alpha \text{ has a prefix in } L(\mathcal{A})\}$. The automaton has satisfied the reachability condition after reading a prefix in $L(\mathcal{A})$ and accepts any continuation after that.
- If we interpret $\mathcal{A}$ as a Büchi PA, then we have $B_\omega(\mathcal{A}) = L(\mathcal{A})^\omega$. The automaton accepts an infinite word if infinitely often the Parikh condition is satisfied in the accepting state. Observe that C is a homogeneous linear set and the initial state as well as the accepting state have the same outgoing transitions.
- If we interpret $\mathcal{A}$ as a co-Büchi PA, then we have $CB_\omega(\mathcal{A}) = L(\mathcal{A}) \cdot \{b\}^\omega$. This is similar to the safety PA, but the accepted words may have a finite "non-safe" prefix from $L(\mathcal{A})$.

- If we interpret $\mathcal{A}$ as a reachability-regular PA, then we have $RR_\omega(\mathcal{A}) = \{\alpha \in \{a,b\}^\omega \mid \alpha$ has a prefix in $L(\mathcal{A})$ and $|\alpha|_b = \infty\}$. After having met the reachability condition the automaton still needs to satisfy the Büchi condition, which enforces infinitely many visits of the accepting state.
- If we interpret $\mathcal{A}$ as a limit PA, then we have $L_\omega(\mathcal{A}) = \{\alpha \in \{a,b\}^\omega \mid |\alpha|_a < \infty\}$. The automaton must visit the accepting state infinitely often. At the same time the extended Parikh image must belong to C, which implies that the infinite word contains only some finite number z of letter a (note that only the vectors of the form (z, ∞) have an effect here, as at least one symbol must be seen infinitely often by the infinite pigeonhole principle).
- If we interpret $\mathcal{A}$ as a weak reset PA, then we have $WR_\omega(\mathcal{A}) = L(\mathcal{A})^\omega$. As a weak reset PA may (but is not forced to) reset the counters upon visiting the accepting state, the automaton may reset every time a (finite) infix in $L(\mathcal{A})$ has been read.
- If we interpret $\mathcal{A}$ as a strong reset PA, then we have $SR_\omega(\mathcal{A}) = \{b^*a\}^\omega \cup \{b^*a\}^* \cdot \{b\}^\omega$. Whenever the automaton reaches an accepting state also the Parikh condition must be satisfied. This implies that the a-loop on q_1 may never be used, as this would increase the first counter value to at least 2, while the second counter value is 1 upon reaching the accepting state q_0 (which resets the counters).

▶ Remark. The automaton $\mathcal{A}$ in the example is deterministic. We note that $L_\omega(\mathcal{A})$ is not deterministic ω-regular but deterministic limit PA-recognizable.

4 Büchi-like characterizations

It was observed in [22] that Büchi PA recognize a strict subset of $\mathcal{L}^\omega_{\mathsf{PA},\mathsf{PA}}$. In this section we first show that the class of reset PA-recognizable ω-languages is a strict superset of $\mathcal{L}^\omega_{\mathsf{PA},\mathsf{PA}}$. Then we provide an automata-based characterization of $\mathcal{L}^\omega_{\mathsf{PA},\mathsf{Reg}}$, $\mathcal{L}^\omega_{\mathsf{PA},\mathsf{PA}}$, and $\mathcal{L}^\omega_{\mathsf{Reg},\mathsf{PA}}$. Towards this goal we first establish some closure properties.

Guha et al. [22] have shown that safety, reachability, Büchi, and co-Büchi PA are closed under union using a modification of the standard construction for PA, i. e., taking the disjoint union of the automata (introducing a fresh initial state), and the disjoint union of the semi-linear sets, where disjointness is achieved by "marking" every vector in the first set by an additional 1 (increasing the dimension by 1), and all vectors in the second set by an additional 2. We observe that the same construction also works for reachability-regular and limit PA, and a small modification is sufficient to make the construction also work for reset PA. We leave the details to the reader.

▶ **Lemma 2.** *The classes of reachability-regular, limit PA-recognizable, and reset PA-recognizable ω-languages are closed under union.*

Furthermore, we show that these classes, as well as the class of Büchi PA-recognizable ω-languages, are closed under left-concatenation with PA-recognizable languages. We provide some details in the next lemma, as we will need to modify the standard construction in such a way that we do not need to keep accepting states of the PA on finite words. This will help to characterize $\mathcal{L}^\omega_{\mathsf{PA},\mathsf{PA}}$ via (restricted) reset PA.

▶ **Lemma 3** ($\star$). *The classes of reachability-regular, limit PA-recognizable, reset PA-recognizable, and Büchi PA-recognizable ω-languages are closed under left-concatenation with PA-recognizable languages.*

Before we continue, we show that we can normalize PA (on finite words) such that the initial state is the only accepting state. This observation simplifies several proofs in this section.

▶ **Lemma 4** (⋆). *Let $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F, C)$ be a PA of dimension d. Then there exists an equivalent PA $\mathcal{A}'$ of dimension $d+1$ with the following properties.*

- *The initial state of $\mathcal{A}'$ is the only accepting state.*
- *$SCC(\mathcal{A}') = \{Q\}$.*

We say that $\mathcal{A}'$ is normalized.

Observe that we have $SR_\omega(\mathcal{A}') = L(\mathcal{A})^\omega$, that is, every normalized PA interpreted as a reset PA recognizes the ω-closure of the language recognized by the PA. As an immediate consequence we obtain the following corollary.

▶ **Corollary 5.** *The class of reset PA-recognizable ω-languages is closed under the ω-operation.*

Combining these results we obtain that every ω-language in $\mathcal{L}^\omega_{\mathsf{PA},\mathsf{PA}}$, i.e. every ω-language of the form $\bigcup_i U_i V_i^\omega$ is reset PA-recognizable. We show that the other direction does not hold, i.e., the inclusion is strict.

▶ **Lemma 6.** *The class $\mathcal{L}^\omega_{\mathsf{PA},\mathsf{PA}}$ is a strict subclass of the class of reset PA-recognizable ω-languages.*

Proof. The inclusion is a direct consequence of Lemma 2, Lemma 3, and Corollary 5. Hence we show that the inclusion is strict.

Consider the ω-language $L = \{a^n b^n \mid n \geq 1\}^\omega \cup \{a^n b^n \mid n \geq 1\}^* \cdot \{a\}^\omega$. This ω-language is reset PA-recognizable, as witnessed by the strong reset PA in Figure 2 with $C = \{(z, z) \mid z \in \mathbb{N}\}$.

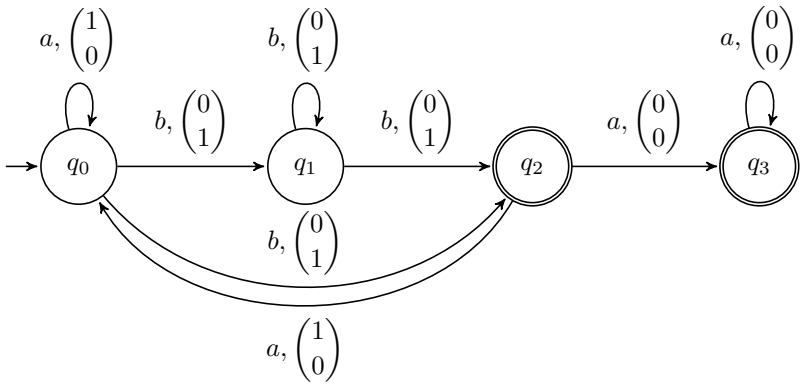


Figure 2 The strong reset PA for $L = \{a^n b^n \mid n \geq 1\}^\omega \cup \{a^n b^n \mid n \geq 1\}^* \cdot \{a\}^\omega$.

We claim that $L \notin \mathcal{L}^\omega_{\mathsf{PA},\mathsf{PA}}$. Assume towards a contraction that $L \in \mathcal{L}^\omega_{\mathsf{PA},\mathsf{PA}}$, i. e., there are Parikh-recognizable languages $U_1, V_1, \ldots, U_n, V_n$ such that $L = U_1 V_1^\omega \cup \cdots \cup U_n V_n^\omega$. Then there is some $i \leq n$ such that for infinitely many $j \geq 1$ the infinite word $\alpha_j = aba^2b^2 \ldots a^j b^j \cdot a^\omega \in U_i V_i^\omega$. Then V_i must contain a word of the form $v = a^k$, $k > 0$. Additionally, there cannot be a word in V_i with infix b. To see this assume for sake of contradiction that there is a word $w \in V_i$ with $\ell = |w|_b > 0$. Let $\beta = (v^{\ell+1} w)^\omega$. Observe that β has an infix that consists of at least $\ell + 1$ many a, followed by at most ℓ, but at least one b, hence, no word of the form $u\beta$ with $u \in U_i$ is in L. This is a contradiction, thus $V_i \subseteq \{a\}^+$.

Since $U_i \in \mathcal{L}_{\mathsf{PA}}$, there is a PA $\mathcal{A}_i$ with $L(\mathcal{A}_i) = U_i$. Let m be the number of states in $\mathcal{A}_i$ and $w' = aba^2b^2 \dots a^{m^4+1}b^{m^4+1}$. Then w' is a prefix of a word accepted by $\mathcal{A}_i$. Now consider the infixes $a^\ell b^\ell$ and the pairs of states q_1, q_2, where we start reading a^ℓ and end reading a^ℓ, and q_3, q_4 where we start to read b^ℓ and end to read b^ℓ, respectively. There are m^2 choices for the first pair and m^2 choices for the second pair, hence m^4 possibilities in total. Hence, as we have more than m^4 such infixes, there must be two with the same associated states q_1, q_2, q_3, q_4. Then we can swap these two infixes and get a word of the form $ab \dots a^r b^s \dots a^s b^r \dots a^{m^4+1}b^{m^4+1}$ that is a prefix of some word in $L(\mathcal{A}_i) = U_i$. But no word in L has such a prefix, a contradiction. Thus, $U_1V_1^\omega \cup \dots \cup U_nV_n^\omega \neq L$. ◀

4.1 Characterization of Büchi Parikh automata

As mentioned in the last section, the class of ω-languages recognized by Büchi PA is a strict subset of $\mathcal{L}^\omega_{\mathsf{PA},\mathsf{PA}}$, i. e., languages of the form $\bigcup_i U_iV_i^\omega$ for Parikh-recognizable U_i and V_i. In this subsection we show that a restriction of the PA recognizing the V_i is sufficient to exactly capture the expressiveness of Büchi PA. To be precise, we show the following.

▶ **Lemma 7.** *The following are equivalent for all ω-languages $L \subseteq \Sigma^\omega$:*

1. *L is Büchi PA-recognizable.*
2. *L is of the form $\bigcup_i U_iV_i^\omega$, where $U_i \in \Sigma^*$ is Parikh-recognizable and $V_i \in \Sigma^*$ is recognized by a normalized PA where C is a homogeneous linear set.*

We note that we can translate every PA (with a linear set C) into an equivalent normalized PA by Lemma 4. However, this construction adds a base vector, as we concatenate $\{1\}$ to C. In fact, this can generally not be avoided without losing expressiveness. It turns out that this loss of expressiveness is exactly what we need to characterize the class of ω-languages recognized by Büchi PA as stated in the previous lemma. The main reason for this is pointed out in the following lemma.

▶ **Lemma 8** ($\star$). *Let L be a language recognized by a (normalized) PA $\mathcal{A} = (Q, \Sigma, q_0, \Delta, \{q_0\}, C)$ where C is a homogeneous linear set. Then we have $B_\omega(\mathcal{A}) = L(\mathcal{A})^\omega$.*

This is the main ingredient to prove Lemma 7.

Proof of Lemma 7. We note that the proof in [22] showing that every ω-language L recognized by a Büchi-PA is of the form $\bigcup_i U_iV_i$ for PA-recognizable U_i and V_i already constructs PA for the V_i of the desired form. This shows the implication (1) $\Rightarrow$ (2).

To show the implication (2) $\Rightarrow$ (1), we use that the ω-closure of languages recognized by PA of the stated form is Büchi PA-recognizable by Lemma 8. As Büchi PA are closed under left-concatenation with PA-recognizable languages (Lemma 3) and union [22], the claim follows. ◀

4.2 Characterization of $\mathcal{L}^\omega_{\mathsf{PA},\mathsf{Reg}}$

In this subsection we characterize $\mathcal{L}^\omega_{\mathsf{PA},\mathsf{Reg}}$ by showing the following equivalences.

▶ **Theorem 9.** *The following are equivalent for all ω-languages $L \subseteq \Sigma^\omega$.*

1. *L is of the form $\bigcup_i U_iV_i^\omega$, where $U_i \in \Sigma^*$ is Parikh-recognizable, and $V_i \subseteq \Sigma^*$ is regular.*
2. *L is limit PA-recognizable.*
3. *L is reachability-regular.*

Observe that in the first item we may assume that L is of the form $\bigcup_i U_i V_i$, where $U_i \in \Sigma^*$ is Parikh-recognizable, and $V_i \subseteq \Sigma^\omega$ is ω-regular. Then, by simple combinatorics and Büchi's theorem we have $\bigcup_i U_i V_i = \bigcup_i U_i(\bigcup_{j_i} X_{j_i} Y_{j_i}^\omega) = \bigcup_{i,j_i} U_i(X_{j_i} Y_{j_i}^\omega) = \bigcup_{i,j_i} (U_i X_{j_i}) Y_{j_i}^\omega$, for regular languages X_{j_i}, Y_{j_i}, where $U_i X_{j_i}$ is Parikh-recognizable, as Parikh-recognizable languages are closed under concatenation [5, Proposition 3].

To simplify the proof, it is convenient to consider the following generalizations of Büchi automata. A *transition-based generalized Büchi automaton* (TGBA) is a tuple $\mathcal{A} = (Q, \Sigma, q_0, \Delta, \mathcal{T})$ where $\mathcal{T} \subseteq 2^\Delta$ is a collection of sets of transitions. Then a run $r_1 r_2 r_3 \dots$ of $\mathcal{A}$ is accepting if for all $T \in \mathcal{T}$ there are infinitely many i such that $r_i \in T$. It is well-known that TGBA have the same expressiveness as Büchi automata [17].

Theorem 9 will be a direct consequence from the following lemmas. The first lemma shows the implication (1) ⇒ (2).

▶ **Lemma 10.** *If $L \in \mathcal{L}^\omega_{\mathsf{PA},\mathsf{Reg}}$, then L is limit PA-recognizable.*

Proof. As the class of limit PA-recognizable ω-languages is closed under union by Lemma 2, it is sufficient to show how to construct a limit PA for an ω-language of the form $L = UV^\omega$, where U is Parikh-recognizable and V is regular.

Let $\mathcal{A}_1 = (Q_1, \Sigma, q_1, \Delta_1, F_1, C)$ be a PA with $L(\mathcal{A}_1) = U$ and $\mathcal{A}_2 = (Q_2, \Sigma, q_2, \Delta_2, F_2)$ be a Büchi automaton with $L_\omega(\mathcal{A}_2) = V^\omega$. We use the following standard construction for concatenation. Let $\mathcal{A} = (Q_1 \cup Q_2, \Sigma, q_1, \Delta, F_2, C)$ be a limit PA where

$$\Delta = \Delta_1 \cup \{(p, a, \mathbf{0}, q) \mid (p, a, q) \in \Delta_2\} \cup \{(f, a, \mathbf{0}, q) \mid (q_2, a, q) \in \Delta_2, f \in F_1\}.$$

We claim that $L_\omega(\mathcal{A}) = L$.

⇒ To show $L_\omega(\mathcal{A}) \subseteq L$, let $\alpha \in L_\omega(\mathcal{A})$ with accepting run $r_1 r_2 r_3 \dots$ where $r_i = (p_{i-1}, \alpha_i, \mathbf{v}_i, p_i)$. As only the states in F_2 are accepting, there is a position j such that $p_{j-1} \in F_1$ and $p_j \in Q_2$. In particular, all transitions of the copy of $\mathcal{A}_2$ are labeled with $\mathbf{0}$, i. e., $\mathbf{v}_i = \mathbf{0}$ for all $i \geq j$. Hence $\rho(r) = \rho(r_1 \dots r_{j-1}) \in C$ (in particular, there is no ∞ value in $\rho(r)$). We observe that $r_1 \dots r_{j-1}$ is an accepting run of $\mathcal{A}_1$ on $\alpha[1, j-1]$, as $p_{j-1} \in F_1$ and $\rho(r_1 \dots r_{j-1}) \in C$. For all $i \geq j$ let $r'_i = (p_{i-1}, \alpha_i, p_i)$. Observe that $(q_2, \alpha_j, p_j) r'_{j+1} r'_{j+2} \dots$ is an accepting run of $\mathcal{A}_2$ on $\alpha_j \alpha_{j+1} \alpha_{j+2} \dots$, hence $\alpha \in L(\mathcal{A}_1) \cdot L_\omega(\mathcal{A}_2) = L$.

⇐ To show $L = UV^\omega \subseteq L_\omega(\mathcal{A})$, let $w \in L(\mathcal{A}_1) = U$ with accepting run s, and $\alpha \in L_\omega(\mathcal{A}_2) = V^\omega$ with accepting run $r = r_1 r_2 r_3 \dots$, where $r_i = (p_{i-1}, \alpha_1, p_i)$. Observe that s is also a partial run of $\mathcal{A}$ on w, ending in an accepting state f. By definition of Δ, we can continue the run s in $\mathcal{A}$ basically as in r. To be precise, let $r'_1 = (f, \alpha_1, \mathbf{0}, p_1)$, and, for all $i > 1$ let $r'_i = (p_{i-1}, \alpha_i, \mathbf{0}, p_i)$. Then $s r'_1 r'_2 r'_3 \dots$ is an accepting run of $\mathcal{A}$ on $w\alpha$, hence $w\alpha \in L_\omega(\mathcal{A})$. ◀

Observe that the construction in the proof of the lemma works the same way when we interpret $\mathcal{A}$ as a reachability-regular PA (every visit of an accepting state has the same good counter value; this argument is even true if we interpret $\mathcal{A}$ as a Büchi PA), showing the implication (1) ⇒ (3).

▶ **Corollary 11.** *If $L \in \mathcal{L}^\omega_{\mathsf{PA},\mathsf{Reg}}$, then L is reachability-regular.*

For the backwards direction we need an auxiliary lemma, essentially stating that semi-linear sets over $C \subseteq (\mathbb{N} \cup \{\infty\})^d$ can be modified such that ∞-entries in vectors in C are replaced by arbitrary integers, and remain semi-linear.

▶ **Lemma 12.** *Let $C \subseteq (\mathbb{N} \cup \{\infty\})^d$ be semi-linear and $D \subseteq \{1, \dots, d\}$. Let $C_D \subseteq \mathbb{N}^d$ be the set obtained from C as follows.*

1. *Remove every vector $\mathbf{v} = (v_1, \dots, v_d)$ where $v_i = \infty$ for an $i \notin D$.*
2. *As long as C_D contains a vector $\mathbf{v} = (v_1, \dots, v_d)$ with $v_i = \infty$ for an $i \leq d$: replace $\mathbf{v}$ by all vectors of the form $(v_1, \dots v_{i-1}, z, v_{i+1}, \dots, v_d)$ for $z \in \mathbb{N}$.*

Then C_D is semi-linear.

Proof. For a vector $\mathbf{v} = (v_1, \dots, v_d) \in (\mathbb{N} \cup \{\infty\})^d$, let $\mathsf{Inf}(\mathbf{v}) = \{i \mid v_i = \infty\}$ denote the positions of ∞-entries in $\mathbf{v}$. Furthermore, let $\bar{\mathbf{v}} = (\bar{v}_1, \dots, \bar{v}_d)$ denote the vector obtained from $\mathbf{v}$ by replacing every ∞-entry by 0, i. e., $\bar{v}_i = 0$ if $v_i = \infty$, and $\bar{v}_i = v_i$ otherwise.

We carry out the following procedure for every linear set of the semi-linear set independently, hence we assume that $C = \{b_0 + b_1 z_1 + \dots + b_\ell z_\ell \mid z_1, \dots, z_\ell \in \mathbb{N}\}$ is linear. We also assume that there is no b_j with $\mathsf{Inf}(b_j) \not\subseteq D$, otherwise, we simply remove it.

Now, if $\mathsf{Inf}(b_0) \not\subseteq D$, then $C_D = \varnothing$, as this implies that every vector in C has an ∞-entry at an unwanted position (the first item of the lemma). Otherwise, $C_D = \{b_0 + \sum_{j \leq \ell} \bar{b}_j z_j + \sum_{i \in \mathsf{Inf}(b_j)} \mathbf{e}_i z_{ij} \mid z_j, z_{ij} \in \mathbb{N}\}$, which is linear by definition. ◀

We are now ready to prove the following lemma, showing the implication (2) ⇒ (1).

▶ **Lemma 13.** *If L is limit PA-recognizable, then $L \in \mathcal{L}^{\omega}_{\mathsf{PA},\mathsf{Reg}}$.*

Proof. Let $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F, C)$ be an limit PA of dimension d. The idea is as follows. We guess a subset $D \subseteq \{1, \dots, d\}$ of counters whose values we expect to be ∞. Observe that every counter not in D has a finite value, hence for every such counter there is a point where all transitions do not increment the counter further. For every subset $D \subseteq \{1, \dots, d\}$ we decompose $\mathcal{A}$ into a PA and a TGBA. In the first step we construct a PA where every counter not in D reaches its final value and is verified. In the second step we construct a TGBA ensuring that for every counter in D at least one transition adding a non-zero value to that counter is used infinitely often. This can be encoded directly into the TGBA. Furthermore we delete all transitions that modify counters not in D.

Fix $D \subseteq \{1, \dots, d\}$ and $f \in F$, and define the PA $\mathcal{A}^D_f = (Q, \Sigma, q_0, \Delta, \{f\}, C_D)$ where C_D is defined as in Lemma 12. Furthermore, we define the TGBA $\mathcal{B}^D_f = (Q, \Sigma, f, \Delta^D, \mathcal{T}^D)$ where Δ^D contains the subset of transitions of Δ where the counters not in D have zero-values (just the transitions without vectors for the counters, as we construct a TGBA). On the other hand, for every counter i in D there is one acceptance component in $\mathcal{T}^D$ that contains exactly those transitions (again without vectors) where the ith counter has a non-zero value. Finally, we encode the condition that at least one accepting state in F needs to by seen infinitely often in $\mathcal{T}^D$ by further adding the component $\{(p, a, q) \in \Delta \mid q \in F\}$ (i. e. now we need to see an incoming transition of a state in F infinitely often).

We claim that $L_\omega(\mathcal{A}) = \bigcup_{D \subseteq \{1,\dots,d\}, f \in F} L(\mathcal{A}^D_f) \cdot L_\omega(\mathcal{B}^D_f)$, which by the comment below Theorem 9 and the equivalence of TGBA and Büchi automata implies the statement of the lemma. The details are presented in the appendix. ◀

The construction in Lemma 10 yields a limit PA whose semi-linear set C contains no vector with an ∞-entry. Hence, by this observation and the construction in the previous lemma we obtain the following corollary.

▶ **Corollary 14.** *For every limit PA there is an equivalent limit PA whose semi-linear set does not contain any ∞-entries.*

Finally we show the implication (3) ⇒ (1).

▶ **Lemma 15.** *If L is reachability-regular, then $L \in \mathcal{L}^{\omega}_{\mathsf{PA,Reg}}$.*

Proof. Let $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F, C)$ be a reachability-regular PA. The intuition is as follows. a reachability-regular PA just needs to verify the counters a single time. Hence, we can recognize the prefixes of infinite words $\alpha \in B_\omega(\mathcal{A})$ that generate the accepting hit with a PA. Further checking that an accepting state is seen infinitely often can be done with a Büchi automaton.

Fix $f \in F$ and let $\mathcal{A}_f = (Q, \Sigma, q_0, \Delta, \{f\}, C)$ be the PA that is, syntactically equal to $\mathcal{A}$ with the only difference that f is the only accepting state. Similarly, let $\mathcal{B}_f = (Q, \Sigma, f, \{(p, a, q) \mid (p, a, \mathbf{v}, q) \in \Delta\}, F)$ be the Büchi automaton obtained from $\mathcal{A}$ by setting f as the initial state and the forgetting the vector labels.

We claim that $RR_\omega(\mathcal{A}) = \bigcup_{f \in F} L(\mathcal{A}_f) \cdot L_\omega(\mathcal{B}_f)$.

$\Rightarrow$ To show $RR_\omega(\mathcal{A}) \subseteq \bigcup_{f \in F} L(\mathcal{A}_f) \cdot L_\omega(\mathcal{B}_f)$, let $\alpha \in B_\omega(\mathcal{A})$ with accepting run $r = r_1 r_2 r_3 \dots$ where $r_i = (p_{i-1}, \alpha_i, \mathbf{v}_i, p_i)$. Let k be arbitrary such that there is an accepting hit in r_k (such a k exists by definition) and consider the prefix $\alpha[1, k]$. Obviously $r_1 \dots r_k$ is an accepting run of $\mathcal{A}_{p_k}$ on $\alpha[1, k]$. Furthermore, there are infinitely many j such that $p_j \in F$ by definition. In particular, there are also infinitely many $j \geq k$ with this property. Let $r'_i = (p_{i-1}, \alpha_i, p_i)$ for all $i > k$. Then $r'_{k+1} r'_{k+2} \dots$ is an accepting run of $\mathcal{B}_{p_k}$ on $\alpha_{k+1}\alpha_{k+2} \dots$ (recall that p_k is the initial state of $\mathcal{B}_{p_k}$). Hence we have $\alpha[1, k] \in L(\mathcal{A}_{p_k})$ and $\alpha_{k+1}\alpha_{k+2} \cdots \in L_\omega(\mathcal{B}_{p_k})$.

$\Leftarrow$ To show $\bigcup_{f \in F} L(\mathcal{A}_f) \cdot L_\omega(\mathcal{B}_f) \subseteq RR_\omega(\mathcal{A})$, let $w \in L(\mathcal{A}_f)$ and $\beta \in L_\omega(\mathcal{B}_f)$ for some $f \in F$. We show $w\beta \in B_\omega(\mathcal{A})$. Let $s = s_1 \dots s_n$ be an accepting run of $\mathcal{A}_f$ on w, which ends in the accepting state f with $\rho(s) \in C$ by definition. Furthermore, let $r = r_1 r_2 r_3 \dots$ be an accepting run of $\mathcal{B}^D_f$ on β which starts in the accepting state f by definition. It is now easily verified that sr' with $r' = r'_1 r'_2 r'_3 \dots$ where $r'_i = (p_{i-1}, \alpha_i, \mathbf{v}_i, p_i)$ (for an arbitrary $\mathbf{v}_i$ such that $r'_i \in \Delta$) is an accepting run of $\mathcal{A}$ on $w\beta$, as there is an accepting hit in s_n, and the (infinitely many) visits of an accepting state in r translate one-to-one, hence $w\beta \in B_\omega(\mathcal{A})$. ◀

As shown in Lemma 7, the class of Büchi PA-recognizable ω-languages is equivalent to the class of ω-languages of the form $\bigcup_i U_i V_i^\omega$ where U_i and V_i are Parikh-recognizable, but the PA for V_i is restricted in such a way that the initial state is the only accepting state and the set is a homogeneous linear set. Observe that for every regular language L there is a Büchi automaton $\mathcal{A}$ where the initial state is the only accepting state with $L_\omega(\mathcal{A}) = L^\omega$ (see e.g. [29, Lemma 1.2]). Hence, $\mathcal{L}^{\omega}_{\mathsf{PA,Reg}}$ is a subset of the class of Büchi PA-recognizable ω-languages. This inclusion is also strict, as witnessed by the Büchi PA in Example 1 which has the mentioned property.

▶ **Corollary 16.** *The class $\mathcal{L}^{\omega}_{\mathsf{PA,Reg}}$ is a strict subclass of the class of Büchi PA-recognizable ω-languages.*

We finish this subsection by observing that (complete) reachability PA capture a subclass of $\mathcal{L}^{\omega}_{\mathsf{PA,Reg}}$ where, due to completeness, all $V_i = \Sigma$.

▶ **Observation 17.** *The following are equivalent for all ω-languages $L \subseteq \Sigma^\omega$.*

1. *L is of the form $\bigcup_i U_i \Sigma^\omega$ where $U_i \subseteq \Sigma^*$ is Parikh-recognizable.*
2. *L is reachability PA-recognizable.*

4.3 Characterization of $\mathcal{L}^{\omega}_{\mathsf{PA,PA}}$ and $\mathcal{L}^{\omega}_{\mathsf{Reg,PA}}$

In this section we give a characterization of $\mathcal{L}^{\omega}_{\mathsf{PA,PA}}$ and a characterization of $\mathcal{L}^{\omega}_{\mathsf{Reg,PA}}$. As mentioned in the beginning of this section, reset PA are too strong to capture this class. However, restrictions of strong reset PA are good candidates to capture $\mathcal{L}^{\omega}_{\mathsf{PA,PA}}$ as well

as $\mathcal{L}^{\omega}_{\mathsf{Reg},\mathsf{PA}}$. In fact we show that it is sufficient to restrict the appearances of accepting states to capture $\mathcal{L}^{\omega}_{\mathsf{PA},\mathsf{PA}}$, as specified by the first theorem of this subsection. Further restricting the vectors yields a model capturing $\mathcal{L}^{\omega}_{\mathsf{Reg},\mathsf{PA}}$, as specified in the second theorem of this subsection. Recall that the condensation of $\mathcal{A}$ is the DAG of strong components of the underlying graph of $\mathcal{A}$.

▶ **Theorem 18.** *The following are equivalent for all ω-languages $L \subseteq \Sigma^{\omega}$.*

1. *L is of the form $\bigcup_i U_i V_i^{\omega}$, where $U_i, V_i \subseteq \Sigma^*$ are Parikh-recognizable.*
2. *L is recognized by a strong reset PA $\mathcal{A}$ with the property that accepting states appear only in the leaves of the condensation of $\mathcal{A}$, and there is at most one accepting state per leaf.*

Proof. (1) ⇒ (2). Let $\mathcal{A}_i = (Q_i, \Sigma, q_i, \Delta_i, F_i)$ for $i \in \{1, 2\}$ be PA and let $L = L(\mathcal{A}_1) \cdot L(\mathcal{A}_2)^{\omega}$. By Lemma 4 we may assume that $\mathcal{A}_2$ is normalized (recall that by Corollary 5 this implies $SR_{\omega}(\mathcal{A}_2) = L(\mathcal{A}_2)^{\omega}$) and hence write $L = L(\mathcal{A}_1) \cdot SR_{\omega}(\mathcal{A}_2)$. As pointed out in the proof of Lemma 3, we can construct a reset PA $\mathcal{A}$ that recognizes L such that only the accepting states of $\mathcal{A}_2$ remain accepting in $\mathcal{A}$. As $\mathcal{A}_2$ is normalized, this means that only q_2 is accepting in $\mathcal{A}$. Hence $\mathcal{A}$ satisfies the property of the theorem. Finally observe that the construction in Lemma 2 maintains this property, implying that the construction presented in Lemma 6 always yields a reset PA of the desired form. ⌟

(2) ⇒ (1). Let $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F, C)$ be a strong reset PA of dimension d with the property of the theorem. Let $f \in F$ and let $\mathcal{A}_f = (Q, \Sigma, q_0, \Delta_f, \{f\}, C \cdot \{1\})$ with $\Delta_f = \{p, a, \mathbf{v} \cdot 0, q) \mid (p, a, \mathbf{v}, q) \in \Delta, q \neq f\} \cup \{(p, a, \mathbf{v} \cdot 1, f) \mid (p, a, \mathbf{v}, f) \in \Delta\}$ be the PA of dimension $d + 1$ obtained from $\mathcal{A}$ by setting f as the only accepting state with an additional counter that is 0 at every transition except the incoming transitions of f, where the counter is set to 1. Additionally all vectors in C are concatenated with 1. Similarly, let $\mathcal{A}_{f,f} = (Q, \Sigma, f, \Delta_f, \{f\}, C \cdot \{1\})$ be the PA of dimension $d + 1$ obtained from $\mathcal{A}_f$ by setting f as the initial state.

⇒ To show $SR_{\omega}(\mathcal{A}) \subseteq \bigcup_{f \in F} L(\mathcal{A}_f) \cdot L(\mathcal{A}_{f,f})^{\omega}$, let $\alpha \in S_{\omega}(\mathcal{A})$ with accepting run $r = r_1 r_2 r_3 \dots$ where $r_i = (p_{i-1}, \alpha_i, \mathbf{v}_i, p_i)$. Let $k_1 < k_2 < \dots$ be the positions of accepting states in r, i. e., $p_{k_i} \in F$ for all $i \geq 1$. First observe that the property in the theorem implies $p_{k_i} = p_{k_j}$ for all $i, j \geq 1$, i. e., no two distinct accepting states appear in r, since accepting states appear only in different leaves of the condensation of $\mathcal{A}$.

For $j \geq 1$ define $r'_j = (p_{j-1}, \alpha_j, \mathbf{v}_j \cdot 0, p_j)$ if $j \neq k_i$ for all $i \geq 1$, and $r'_j = (p_{j-1}, \alpha_j, \mathbf{v}_j \cdot 1, p_j)$ if $j = k_i$ for some $i \geq 1$, i. e., we replace every transition r_j by the corresponding transition in Δ_f.

Now consider the partial run $r_1 \dots r_{k_1}$ and observe that $p_i \neq p_{k_1}$ for all $i < k_1$, and $\rho(r_1 \dots r_{k_1}) \in C$ by the definition of strong reset PA. Hence $r' = r'_1 \dots r'_{k_1}$ is an accepting run of $\mathcal{A}_{p_{k_1}}$ on $\alpha[1, k_1]$, as only a single accepting state appears in r', the newly introduced counter has a value of 1 when entering p_{k_1}, i. e., $\rho(r') \in C \cdot \{1\}$, hence $\alpha[1, k_1] \in L(\mathcal{A}_{p_{k_1}})$.

Finally, we show that $\alpha[k_i + 1, k_{i+1}] \in L(\mathcal{A}_{p_{k_1}, p_{k_1}})$. Observe that $r'_{k_i+1} \dots r'_{k_{i+1}}$ is an accepting run of $\mathcal{A}_{p_{k_1}, p_{k_1}}$ on $\alpha[k_i + 1, k_{i+1}]$: we have $\rho(r_{k_i+1} \dots r_{k_{i+1}}) = \mathbf{v} \in C$ by definition. Again, as only a single accepting state appears in $r'_{k_i+1} \dots r'_{k_{i+1}}$, we have $\rho(r'_{k_i+1} \dots r'_{k_{i+1}}) = \mathbf{v} \cdot 1 \in C \cdot \{1\}$, and hence $\alpha[k_i + 1, k_{i+1}] \in L(\mathcal{A}_{p_{k_1}, p_{k_1}})$. We conclude $\alpha \in L(\mathcal{A}_{p_{k_1}}) \cdot L(\mathcal{A}_{p_{k_1}, p_{k_1}})^{\omega}$.

⇐ To show $\bigcup_{f \in F} L(\mathcal{A}_f) \cdot L(\mathcal{A}_{f,f})^{\omega} \subseteq SR_{\omega}(\mathcal{A})$, let $u \in L(\mathcal{A}_f)$, and $v_1, v_2, \dots \in L(\mathcal{A}_{f,f})$ for some $f \in F$. We show that $u v_1 v_2 \dots \in SR_{\omega}(\mathcal{A})$.

First let $u = u_1 \dots u_n$ and $r' = r'_1 \dots r'_n$ with $r'_i = (p_{i-1}, u_i, \mathbf{v}_i \cdot c_i, p_i)$, where $c_i \in \{0, 1\}$, be an accepting run of $\mathcal{A}_f$ on u. Observe that $\rho(r') \in C \cdot \{1\}$, hence $\sum_{i \leq n} c_i = 1$, i. e., p_n is the only occurrence of an accepting state in r' (if there was another, say p_j, then $c_j = 1$ by the choice of Δ_f, hence $\sum_{i \leq n} c_i > 1$, a contradiction). For all $1 \leq i \leq n$ let $r_i = (p_{i-1}, u_i, \mathbf{v}_i, p_i)$. Then $r_1 \dots r_n$ is a partial run of $\mathcal{A}$ on w with $\rho(r_1 \dots r_n) \in C$ and $p_n = f$.

Similarly, no run of $\mathcal{A}_{f,f}$ on any v_i visits an accepting state before reading the last symbol, hence we continue the run from r_n on $v_1, v_2, \dots$ using the same argument. Hence $uv_1v_2 \cdots \in SR_\omega(\mathcal{A})$, concluding the proof. ◀

As a side product of the proof of Theorem 18 we get the following corollary, which is in general not true for arbitrary reset PA.

▶ **Corollary 19.** *Let $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F, C)$ be a strong reset PA with the property that accepting states appear only in the leaves of the condensation of $\mathcal{A}$, and there is at most one accepting state per leaf. Then we have $SR_\omega(\mathcal{A}) = \bigcup_{f \in F} S_\omega(Q, \Sigma, q_0, \Delta, \{f\}, C)$.*

By even further restricting the power of strong reset PA, we get the following characterization of $\mathcal{L}^\omega_{\mathsf{Reg},\mathsf{PA}}$.

▶ **Theorem 20** ($\star$). *The following are equivalent for all ω-languages $L \subseteq \Sigma^\omega$.*

1. *L is of the form $\bigcup_i U_i V_i^\omega$, where $U_i \subseteq \Sigma^*$ is regular and $V_i \subseteq \Sigma^*$ is Parikh-recognizable.*
2. *L is recognized by a strong reset PA $\mathcal{A}$ with the following properties.*
   - **(a)** *At most one state q per leaf of the condensation of $\mathcal{A}$ may have incoming transitions from outside the leaf, this state q is the only accepting state in the leaf, and there are no accepting states in non-leaves.*
   - **(b)** *only transitions connecting states in a leaf may be labeled with a non-zero vector.*

Observe that property (a) is a stronger property than the one of Theorem 18, hence, strong reset PA with this restriction are at most as powerful as those that characterize $\mathcal{L}^\omega_{\mathsf{PA},\mathsf{PA}}$. However, as a side product of the proof we get that property (a) is equivalent to the property of Theorem 18. Hence, property (b) is mandatory to sufficiently weaken strong reset PA such that they capture $\mathcal{L}^\omega_{\mathsf{Reg},\mathsf{PA}}$. In fact, using the notion of normalization, we can re-use most of the ideas in the proof of Theorem 18.

5 Blind counter machines and ε-elimination

As mentioned in the introduction, blind counter machines as an extension of automata with counting mechanisms were already introduced and studied in the 70s [18]. Over finite words they are equivalent to Parikh automata [26]. Blind counter machines over infinite words were first considered by Fernau and Stiebe [14]. In this section we first recall the definition of blind counter machines as introduced by Fernau and Stiebe [14]. The definition of these automata admits ε-transitions. It is easily observed that Büchi PA with ε-transitions are equivalent to blind counter machines. Therefore, we extend all Parikh automata models studied in this paper with ε-transitions and consider the natural question whether they admit ε-elimination (over infinite words). We show that almost all models allow ε-elimination, the exception being safety and co-Büchi PA. For the latter two models we observe that ε-transitions allow to encode ω-regular conditions, meaning that such transitions give the models enough power such that they can recognize all ω-regular languages.

A *blind k-counter machine* (CM) is a quintuple $\mathcal{M} = (Q, \Sigma, q_0, \Delta, F)$ where Q, Σ, q_0 and F are defined as for NFA, and $\Delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times \mathbb{Z}^k \times Q$ is a finite set of *integer labeled transitions*. In particular, the transitions of Δ are labeled with possibly negative integer vectors. Observe that ε-transitions are allowed.

A *configuration* for an infinite word $\alpha = \alpha_1\alpha_2\alpha_3\ldots$ of $\mathcal{M}$ is a tuple of the form $c = (p, \alpha_1\ldots\alpha_i, \alpha_{i+1}\alpha_{i+2}\ldots, \mathbf{v}) \in Q \times \Sigma^* \times \Sigma^\omega \times \mathbb{Z}^k$ for some $i \geq 0$. A configuration c *derives* into a configuration c', written $c \vdash c'$, if either $c' = (q, \alpha_1\ldots\alpha_{i+1}, \alpha_{i+2}\ldots, \mathbf{v}+\mathbf{u})$ and $(p, \alpha_{i+1}, \mathbf{u}, q) \in \Delta$, or $c' = (q, \alpha_1\ldots\alpha_i, \alpha_{i+1}\alpha_{i+2}\ldots, \mathbf{v}+\mathbf{u})$ and $(p, \varepsilon, \mathbf{u}, q) \in \Delta$. $\mathcal{M}$ *accepts* an infinite word α if there is an infinite sequence of configuration derivations $c_1 \vdash c_2 \vdash c_3 \vdash \ldots$ with $c_1 = (q_0, \varepsilon, \alpha, \mathbf{0})$ such that for infinitely many i we have $c_i = (p_i, \alpha_1\ldots\alpha_j, \alpha_{j+1}\alpha_{j+2}\ldots, \mathbf{0})$ with $p_i \in F$ and for all $j \geq 1$ there is a configuration of the form $(p, \alpha_1\ldots\alpha_j, \alpha_{j+1}\alpha_{j+2}\ldots, \mathbf{v})$ for some $p \in Q$ and $\mathbf{v} \in \mathbb{Z}^k$ in the sequence. That is, a word is accepted if we infinitely often visit an accepting state when the counters are $\mathbf{0}$, and every symbol of α is read at some point. We define the ω-language recognized by $\mathcal{M}$ as $L_\omega(\mathcal{M}) = \{\alpha \in \Sigma^\omega \mid \mathcal{M} \text{ accepts } \alpha\}$.

Parikh automata naturally generalize to Parikh automata with ε-transitions. An ε-PA is a tuple $\mathcal{A} = (Q, \Sigma, q_0, \Delta, \mathcal{E}, F, C)$ where $\mathcal{E} \subseteq Q \times \{\varepsilon\} \times \mathbb{N}^d \times Q$ is a finite set of *labeled ε-transitions*, and all other entries are defined as for PA. A run of $\mathcal{A}$ on an infinite word $\alpha_1\alpha_2\alpha_3\ldots$ is an infinite sequence of transitions $r \in (\mathcal{E}^*\Delta)^\omega$, say $r = r_1r_2r_3\ldots$ with $r_i = (p_{i-1}, \gamma_i, \mathbf{v}_i, p_i)$ such that $p_0 = q_0$, and $\gamma_i = \varepsilon$ if $r_i \in \mathcal{E}$, and $\gamma_i = \alpha_j$ if $r_i \in \Delta$ is the j-th occurrence of a (non-ε) transition in r. The acceptance conditions of the models translate to runs of ε-PA in the obvious way. We use terms like ε-safety PA, ε-reachability PA, etc, to denote an ε-PA with the respective acceptance condition.

Note that we can treat every PA as an ε-PA, that is, a PA $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F, C)$ is equivalent to the ε-PA $\mathcal{A}' = (Q, \Sigma, q_0, \Delta, \varnothing, F, C)$.

5.1 Equivalence of blind counter machines with Büchi PA

We start with the following simple observation.

▶ **Lemma 21 ($\star$).** *CM and ε-Büchi PA are equivalent.*

5.2 ε-elimination for Parikh automata

We now show that almost all PA models admit ε-elimination. We first consider Büchi PA, where ε-elimination implies the equivalence of blind counter machines and Büchi PA by Lemma 21. We provided a direct but quite complicated proof in the manuscript [19]. We thank Georg Zetzsche for outlining a much simpler proof, which we present here.

▶ **Theorem 22.** *ε-Büchi PA admit ε-elimination.*

Proof. Observe that the construction in Lemma 21 translates ε-free CM into ε-free Büchi PA. We can hence translate a given Büchi PA into a CM and eliminate ε-transitions and then translate back into a Büchi PA. Therefore, all we need to show is that CM admit ε-elimination.

To show that CM admit ε-elimination we observe that

$$L \text{ is recognized by a CM} \iff L = \bigcup_i U_i V_i^\omega,$$

where U_i is a language of finite words that is recognized by a CM and V_i is a language of finite words that is recognized by a CM where $F = \{q_0\}$. The proof of this observation is very similar to the proof of Lemma 7 and we leave the details to the reader.

As shown in [18, 27, 30], CM on finite words admit ε-elimination. Furthermore, the proof technique established in [30, Lemma 7.7] it is immediate that the condition that $F = \{q_0\}$ is preserved. We obtain ε-free CM $\mathcal{A}'_i$ and $\mathcal{B}'_i$ for the languages U_i and V_i. Using the

construction of [26], we can translate $\mathcal{A}'_i$ and $\mathcal{B}'_i$ into PA $\mathcal{A}_i$ and $\mathcal{B}_i$, where the $\mathcal{B}_i$ satisfy $F_i = \{q_0\}$ and the sets C_i are homogeneous linear sets (Theorem 32 of [26]). Now the statement follows by Lemma 7. ◀

We continue with ε-reachability, ε-reachability-regular and ε-limit PA, as we show ε-elimination using the same technique for these models. As shown in Observation 17 and Theorem 9, the class of ω-languages recognized by reachability PA coincides with the class of ω-languages of the form $\bigcup_i U_i\Sigma^\omega$ for Parikh-recognizable U_i, and the class of reachability-regular and limit PA-recognizable ω-languages coincides with the class of ω-languages of the form $\bigcup_i U_iV_i^\omega$ for Parikh-recognizable U_i and regular V_i, respectively. It is well-known that NFA and PA on finite words are closed under homomorphisms and hence admit ε-elimination [26] (as a consequence of [27, Proposition II.11], ε-transitions can even be eliminated without changing the semi-linear set). The characterizations allow us to reduce ε-elimination of these infinite word PA to the finite case.

▶ **Lemma 23** (⋆). *ε-reachability, ε-reachability-regular, and ε-limit PA admit ε-elimination.*

Finally we show that safety and co-Büchi PA do not admit ε-elimination.

▶ **Lemma 24.** *ε-safety PA and ε-co-Büchi PA do not admit ε-elimination.*

Proof. Consider the automaton $\mathcal{A}$ in Figure 3 with $C = \{(z, z') \mid z' \geq z\}$.

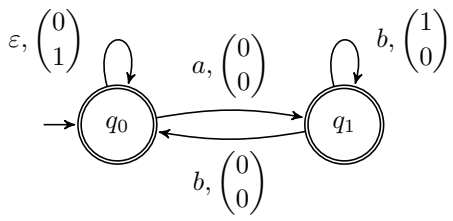


Figure 3 The ε-PA with $C = \{(z, z') \mid z' \geq z\}$ for the proof of Lemma 24.

If we interpret $\mathcal{A}$ as an ε-safety or ε-co-Büchi PA, we have we have $S_\omega(\mathcal{A}) = CB_\omega(\mathcal{A}) = \{ab^+\}^\omega$. This ω-language is neither safety PA nor co-Büchi PA-recognizable (one can easily adapt the proof in [22] showing that $\{\alpha \in \{a, b\}^\omega \mid |\alpha|_a = \infty\}$ is neither safety PA nor co-Büchi PA-recognizable).

Observe how $\mathcal{A}$ utilizes the ε-transition to enforce that q_0 is seen infinitely often: whenever the b-loop on q_1 is used, the first counter increments. The semi-linear set states that at no point the first counter value may be greater than the second counter value which can only be increased using the ε-loop on q_0. Hence, any infinite word accepted by $\mathcal{A}$ may contain arbitrary infixes of the form b^n for $n < \infty$, as the automaton can use the ε-loop on q_0 at least n times before, but not b^ω. ◀

As a consequence of the previous proof we show that ε-safety PA and ε-co-Büchi PA recognize all ω-regular languages, as the presented trick can be used to encode ω-regular conditions, that is ε-transitions can be used to enforce that at least one state of a subset of states needs to be visited infinitely often.

▶ **Lemma 25** (⋆). *Every ω-regular language is ε-safety PA and ε-co-Büchi recognizable.*

Finally we show that strong ε-reset PA and weak ε-reset PA admit ε-elimination. We show that these two models are equivalent. Hence to show this statement we only need to argue that strong ε-reset PA admit ε-elimination.

▶ **Lemma 26** (⋆). *Every strong ε-reset PA $\mathcal{A}$ is equivalent to a weak ε-reset PA $\mathcal{A}'$ that has the same set of states and uses one additional counter. If $\mathcal{A}$ is a strong reset PA, then $\mathcal{A}'$ is a weak reset PA.*

▶ **Lemma 27** (⋆). *Every weak ε-reset PA $\mathcal{A}$ is equivalent to a strong ε-reset PA $\mathcal{A}'$ with at most twice the number of states and the same number of counters. If $\mathcal{A}$ is a strong reset PA, then $\mathcal{A}'$ is a weak reset PA.*

▶ **Lemma 28** (⋆). *Strong ε-reset PA admit ε-elimination.*

6 Decision problems

As shown by Guha et al. [22], the results for common decision problems translate from the finite case to reachability PA and Büchi PA, that is, non-emptiness is NP-complete, and universality (and hence inclusion and equivalence) are undecidable. We show that these results translate to reset PA (which are more expressive), even if we allow ε-transitions (which does not increase their expressiveness but our ε-elimination procedure constructs an equivalent reset PA of super-polynomial size). Hence, (ε-)reset PA are a powerful model that can still be used for algorithmic applications, such as the model checking problem.

The main reason for this is that the ω-languages recognized by reset PA are ultimately periodic, meaning that whenever a reset PA accepts at least one infinite word, then it also accepts an infinite word of the form uv^ω.

▶ **Lemma 29** (⋆). *Let $\mathcal{A}$ be an ε-reset PA. If $SR_\omega(\mathcal{A}) \neq \varnothing$, then $\mathcal{A}$ accepts an infinite word of the form uv^ω.*

As a consequence, we can reduce non-emptiness for reset PA to the finite word case, as clarified in the following lemma.

▶ **Lemma 30** (⋆). *Non-emptiness for ε-reset PA is* NP*-complete.*

Furthermore, we study the following membership problem for automata processing infinite words. Given an automaton $\mathcal{A}$ and finite words u, v, does $\mathcal{A}$ accept uv^ω?

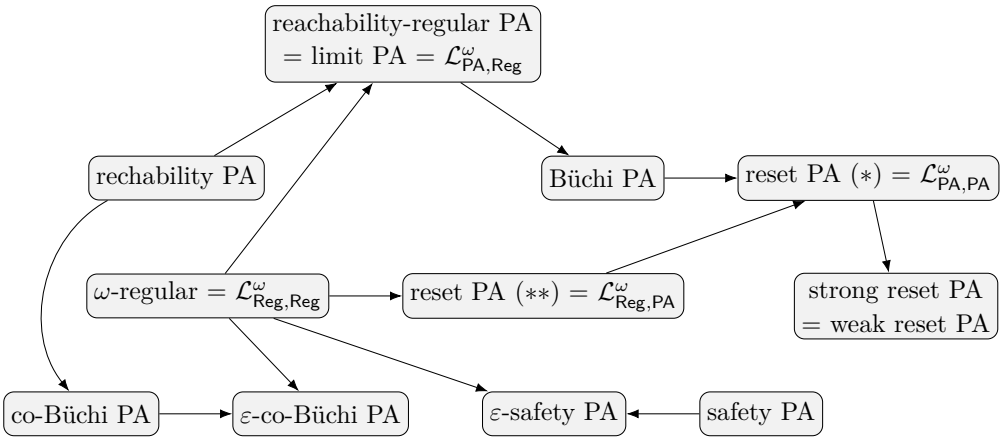
Note that we can always construct a safety automaton that recognizes uv^ω and no other infinite word with $|uv|$ many states. Recall that every state of a safety automaton is accepting. We show that the intersection of a reset PA-recognizable ω-language and a safety automaton-recognizable ω-language remains reset PA-recognizable using a product construction which is computable in polynomial time. Hence, we can reduce the membership problem to the non-emptiness the standard way.

▶ **Lemma 31** (⋆). *The class of reset PA-recognizable ω-languages is closed under intersection with safety automata-recognizable ω-languages.*

As the membership problem for PA (on finite words) is NP-complete [15], and the construction in the previous lemma can be computed efficiently, we obtain the following result.

▶ **Corollary 32.** *Membership for ε-reset PA is* NP*-complete.*

Finally, we observe that universality, inclusion and equivalence remain undecidable for (ε-)reset PA, as these problems are already undecidable for Büchi PA [22] and the constructions showing that the class of Büchi PA-recognizable ω-languages is a subclass of $\mathcal{L}^\omega_{\mathsf{PA},\mathsf{PA}}$, and that $\mathcal{L}^\omega_{\mathsf{PA},\mathsf{PA}}$ is a subclass of the class of reset PA-recognizable ω-languages are effective.



(∗) At most one state q per leaf of $C(\mathcal{A})$ may have incoming transitions from outside the leaf, this state q is the only accepting state in the leaf, and there are no accepting states in non-leaves;
(∗∗) and only transitions connecting states in leaves may be labeled with non-zero vectors.

Figure 4 Overview of our results. Arrows mean strict inclusions. If not explicitly shown otherwise, all models are equivalent to their ε-counterparts.

7 Conclusion

We conclude by giving an overview of all characterizations and inclusions shown in this paper, as depicted in Figure 4.

Recall the ω-languages motivated by the model checking problem from the introduction, namely $\{\alpha \in \{a,b,c\}^\omega \mid$ there are infinitely many prefixes w of α with $|w|_a > |w|_b + |w|_c\}$, representing unfair resource distributions of an operating system, and $\{\alpha \in \{p,c\}^\omega \mid$ there is a prefix w of α with $|w|_c > |w|_p\}$, representing invalid computations in a producer-consumer setting. Both of these ω-languages are Reset PA-recognizable (in fact, the first is Büchi PA-recognizable and the second is even reachability PA-recognizable). As mentioned, in a common approach we are given a system represented as a Kripke structure K, and a specification of counter-examples given as an automaton, e.g. a reset PA $\mathcal{A}$. By moving the labels of the states of K to its transitions, we can see a Kripke structure as a safety automaton $\mathcal{A}_K$ (see [10, Theorem 28] for details). As every state of a safety automaton is accepting, we can easily find a reset automaton recognizing all bad computations of K (that is the intersection of the ω-languages recognized by $\mathcal{A}_K$ and $\mathcal{A}$) by Lemma 31. As (non-)emptiness is decidable for reset PA, we can solve the model-checking problem by computing the product automaton of $\mathcal{A}_K$ and $\mathcal{A}$ and testing for emptiness, which is in coNP by Lemma 30.

References

1 Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. The MIT Press, 2008.

2 Brenda S. Baker and Ronald V. Book. Reversal-bounded multipushdown machines. *Journal of Computer and System Sciences*, 8(3):315–332, 1974.

3 Pascal Baumann, Flavio D'Alessandro, Moses Ganardi, Oscar Ibarra, Ian McQuillan, Lia Schütze, and Georg Zetzsche. Unboundedness problems for machines with reversal-bounded counters. In *Foundations of Software Science and Computation Structures*, pages 240–264, Cham, 2023. Springer Nature Switzerland.

4 J. Richard Büchi. Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly*, 6(1-6):66–92, 1960.

5 Michaël Cadilhac. *Automates à contraintes semilinéaires= Automata with a semilinear constraint*. PhD thesis, University of Montréal, 2013.

6 Michaël Cadilhac, Alain Finkel, and Pierre McKenzie. On the expressiveness of Parikh automata and related models. In *Third Workshop on Non-Classical Models for Automata and Applications - NCMA 2011*, volume 282 of *books@ocg.at*, pages 103–119. Austrian Computer Society, 2011.

7 Michaël Cadilhac, Alain Finkel, and Pierre McKenzie. Affine Parikh automata. *RAIRO Theor. Informatics Appl.*, 46(4):511–545, 2012.

8 Michaël Cadilhac, Alain Finkel, and Pierre McKenzie. Bounded Parikh automata. *Int. J. Found. Comput. Sci.*, 23(8):1691–1710, 2012.

9 Edmund M Clarke, Orna Grumberg, and Doron A. Peled. *Model checking*. The MIT Press, London, Cambridge, 1999.

10 Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem. *Handbook of Model Checking*. Springer Publishing Company, Incorporated, 1st edition, 2018.

11 Luc Dartois, Emmanuel Filiot, and Jean-Marc Talbot. Two-way Parikh automata with a visibly pushdown stack. In *Foundations of Software Science and Computation Structures - 22nd International Conference, FOSSACS 2019*, volume 11425 of *Lecture Notes in Computer Science*, pages 189–206. Springer, 2019.

12 Enzo Erlich, Shibashis Guha, Ismaël Jecker, Karoliina Lehtinen, and Martin Zimmermann. History-deterministic Parikh automata. *arXiv preprint*, 2022. `arXiv:2209.07745`.

13 Henning Fernau and Ralf Stiebe. Sequential grammars and automata with valences. *Theoretical Computer Science*, 276(1):377–405, 2002.

14 Henning Fernau and Ralf Stiebe. Blind counter automata on omega-words. *Fundam. Inform.*, 83:51–64, 2008.

15 Diego Figueira and Leonid Libkin. Path logics for querying graphs: Combining expressiveness and efficiency. In *Proceedings of the 2015 30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, LICS '15, pages 329–340. IEEE, 2015.

16 Emmanuel Filiot, Shibashis Guha, and Nicolas Mazzocchi. Two-way Parikh automata. In *39th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2019*, volume 150 of *LIPIcs*, pages 40:1–40:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019.

17 Dimitra Giannakopoulou and Flavio Lerda. From states to transitions: Improving translation of LTL formulae to Büchi automata. In *Formal Techniques for (Networked and) Distributed Systems*, 2002.

18 Sheila A. Greibach. Remarks on blind and partially blind one-way multicounter machines. *Theoretical Computer Science*, 7(3):311–324, 1978.

19 Mario Grobler, Leif Sabellek, and Sebastian Siebertz. Parikh automata on infinite words. *arXiv preprint*, 2023. `arXiv:2301.08969`.

20 Mario Grobler, Leif Sabellek, and Sebastian Siebertz. Remarks on Parikh-recognizable omega-languages. *arXiv preprint*, 2023. `arXiv:2307.07238`.

21 Mario Grobler and Sebastian Siebertz. Büchi-like characterizations for Parikh-recognizable omega-languages. *arXiv preprint*, 2023. `arXiv:2302.04087`.

22 Shibashis Guha, Ismaël Jecker, Karoliina Lehtinen, and Martin Zimmermann. Parikh Automata over Infinite Words. In *42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2022)*, volume 250 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 40:1–40:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

23 Hendrik Jan Hoogeboom. Context-free valence grammars - revisited. In *Developments in Language Theory*, pages 293–303, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

24 Oscar H. Ibarra. Reversal-bounded multicounter machines and their decision problems. *J. ACM*, 25(1):116–133, 1978.

25 Wong Karianto. Parikh automata with pushdown stack. *Diplomarbeit, RWTH Aachen*, 2004.

26 Felix Klaedtke and Harald Rueß. Monadic second-order logics with cardinalities. In *Automata, Languages and Programming*, pages 681–696, Berlin, Heidelberg, 2003. Springer.

27 Michel Latteux. Cônes rationnels commutatifs. *Journal of Computer and System Sciences*, 18(3):307–333, 1979.

28 Victor Mitrana and Ralf Stiebe. Extended finite automata over groups. *Discrete Applied Mathematics*, 108(3):287–300, 2001.

29 Wolfgang Thomas. *Automata on Infinite Objects*, pages 133–191. MIT Press, Cambridge, MA, USA, 1991.

30 Georg Zetzsche. Silent transitions in automata with storage. In *Automata, Languages, and Programming*, pages 434–445, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

Characterising and Verifying the Core in Concurrent Multi-Player Mean-Payoff Games

Julian Gutierrez
Monash University, Clayton, Australia

Anthony W. Lin
University of Kaiserslautern-Landau, Germany
Max-Planck Institute for Software Systems, Kaiserslautern, Germany

Muhammad Najib
Heriot-Watt University, Edinburgh, UK

Thomas Steeples
University of Oxford, UK

Michael Wooldridge
University of Oxford, UK

Abstract

Concurrent multi-player mean-payoff games are important models for systems of agents with individual, non-dichotomous preferences. Whilst these games have been extensively studied in terms of their equilibria in non-cooperative settings, this paper explores an alternative solution concept: the core from cooperative game theory. This concept is particularly relevant for cooperative AI systems, as it enables the modelling of cooperation among agents, even when their goals are not fully aligned. Our contribution is twofold. First, we provide a characterisation of the core using discrete geometry techniques and establish a necessary and sufficient condition for its non-emptiness. We then use the characterisation to prove the existence of polynomial witnesses in the core. Second, we use the existence of such witnesses to solve key decision problems in rational verification and provide tight complexity bounds for the problem of checking whether some/every equilibrium in a game satisfies a given LTL or GR(1) specification. Our approach is general and can be adapted to handle other specifications expressed in various fragments of LTL without incurring additional computational costs.

2012 ACM Subject Classification Theory of computation → Logic and verification; Theory of computation → Verification by model checking; Theory of computation → Solution concepts in game theory

Keywords and phrases Concurrent games, multi-agent systems, temporal logic, game theory

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.32

Related Version *Full Version*: https://doi.org/10.48550/arXiv.2311.15883

Funding *Julian Gutierrez*: Research partially sponsored by the DARPA Assured Neuro Symbolic Learning and Reasoning (ANSR) program under award number FA8750-23-2-1016.
Anthony W. Lin: supported by European Research Council under European Union's Horizon research and innovation programme (grant agreement no 501100000781)
Michael Wooldridge: Supported by a UKRI Turing AI World Leading Researcher Fellowship (EP/W002949/1).

Acknowledgements We wish to thank anonymous reviewers for their useful feedback.

1 Introduction

Concurrent games, where agents interact over an infinite sequence of rounds by choosing actions simultaneously, are one of the most important tools for modelling multi-agent systems. This model has received considerable attention in the research community (see,

e.g., [4, 11, 38, 32, 45]). In these games, the system evolves based on the agents' choices, and their preferences are typically captured by associating them with a *Boolean objective* (e.g., a *temporal logic formula*) representing their goal. Strategic issues arise as players seek to satisfy their own goals while taking into account the goals and rational behaviour of other players. Note that the preferences induced by such goals are *dichotomous*: a player will either be satisfied or unsatisfied. However, many systems require richer models of preferences that capture issues such as resource consumption, cost, or system performance [19, 18, 22].

Mean-payoff games [29, 62] are widely used to model the quantitative aspects of systems. Whilst much research has been conducted on *non-cooperative* mean-payoff games and solution concepts such as *Nash equilibrium* (NE) and *subgame perfect equilibrium* (e.g., [58, 14, 15]), this paper focuses on a *cooperative* setting. In this setting, players can reach binding agreements and form coalitions to collectively achieve better payoffs or eliminate undesirable outcomes[1]. As a result, NE and its variants may not be suitable for examining the stable behaviours that arise in these types of games. For example, in the Prisoner's Dilemma game, players can avoid mutual defection, which is the unique NE, by establishing binding agreements [53]. Thus, analysing games through the lens of cooperative game theory poses distinct challenges and is important in and of itself. This paradigm is particularly relevant for modelling and analysing *cooperative AI* systems, which have recently emerged as a prominent topic [26, 25, 24, 7]. In these systems, agents are able to communicate and benefit from cooperation, even when their goals are not fully aligned. We illustrate that this is also the case in the context of mean-payoff games in Example 1.

We focus on a solution concept from cooperative game theory known as the *core* [5, 54, 40], which is the most widely-studied solution concept for cooperative games. Particularly, we study the core of mean-payoff games where players have access to *finite but unbounded memory strategies*. The motivation is clear, as finite-memory strategies are sufficiently powerful for implementing LTL objectives while being realisable in practice. Our main contribution is twofold: First, we provide a characterisation of the core using techniques from discrete geometry[2] (cf. logical characterisation in [40, 39]) and establish a necessary and sufficient condition for its non-emptiness. We believe that our characterisation holds value in its own right, as it connects to established techniques used in game theory and economics. This has the potential to enable the application of more sophisticated methods and computational tools (e.g., linear programming solvers) in the area of *rational verification* [1, 38]. Second, we provide tight complexity bounds for key decision problems in rational verification with LTL and $\mathsf{GR}(1)$ [8] specifications (see Table 1). $\mathsf{GR}(1)$ is a LTL fragment that has been used in various domains [35, 17, 31, 47] and covers a wide class of common LTL specification patterns [46]. Our approach to solving rational verification problems is very general and can be easily adapted for different LTL fragments beyond $\mathsf{GR}(1)$. This is the first work to study the core of mean-payoff games with *finite but unbounded memory strategies*, and to explore the complexity of problems related to the rational verification of such games in this setting.

Related Work. The game-theoretical analysis of temporal logic properties in multi-agent systems has been studied for over a decade (see e.g., [32, 23, 49, 45, 37, 61]). However, most of the work has focused on a non-cooperative setting. Recently, there has been

[1] We emphasise that this paper concerns the *outcome* of games *when* such agreements can be reached. The mechanism for agreements is assumed to be exogenous and beyond the scope of this paper.

[2] We note that linear programming and convex analysis are well-established tools for studying the core in traditional economics (see e.g., [33, 10, 54, 59]). However, the settings and contexts (e.g., the game models) of these previous works differ from those of the present work, and their results do not automatically carry over.

an increased interest in the analysis of concurrent games in a cooperative setting. The core has been studied in the context of deterministic games with dichotomous preferences by [40, 39] using the logics ATL* [4] and SL [49]. However, as far as we are aware, there are no extensions of these logics that adopt mean-payoff semantics. Quantitative extensions exist [16, 12], and the core is studied in [12] using the logic SL[$\mathcal{F}$] that extends SL with quantitative satisfaction value, but the semantics of these logics are not defined on mean-payoff conditions and thus cannot be used to reason about the core of mean-payoff games. In the stochastic setting, [36] examines the core in stochastic games with LTL objectives under the *almost-sure* satisfaction condition. The approach relies on *qualitative parity logic* [6] and is not applicable to mean-payoff objectives. Closer to our work is [57] which studies the core of multi-player mean-payoff games with Emerson-Lei condition [30] in the *memoryless* setting. Whilst memoryless strategies are easy to implement, finite-memory and arbitrary mathematical strategies offer greater richness. For instance, players can achieve higher payoffs and implement LTL properties with finite-memory strategies, which may not be possible with memoryless ones (see Example 1). The approach proposed in [57], which involves using a non-deterministic Turing machine to guess the correct strategies, is not applicable in the present work's setting. This is because players may have finite but unbounded memory strategies, and as such, strategies may be arbitrarily large. To address this limitation, we propose a new approach that can handle such scenarios.

Organisation. The rest of the paper is structured as follows. Section 2 provides an overview of temporal logics, multi-player mean-payoff games, relevant game-theoretic concepts, and key mathematical concepts. Section 3 develops a method to characterise the core using discrete geometry techniques, leading to a crucial result for Section 4, where we determine the complexity of several decision problems. Finally, Section 5 offers concluding remarks. Due to space constraints, all the missing proofs and some technical details are omitted from this version and can be found in the full version [41].

2 Preliminaries

Given any set X, we use X^*, X^ω and X^+ for, respectively, the sets of finite, infinite, and non-empty finite sequences of elements in X. For $Y \subseteq X$, we write X_{-Y} for $X \setminus Y$ and X_{-i} if $Y = \{i\}$. We extend this notation to tuples $w = (x_1, ..., x_k, ..., x_n) \in X_1 \times \cdots \times X_n$, and write w_{-k} for $(x_1, ..., x_{k-1}, x_{k+1}, ..., x_n)$. Similarly, for sets of elements, we write w_{-Y} to denote w without each x_k, for $k \in Y$. For a sequence v, we write $v[t]$ or v^t for the element in position $t+1$ in the sequence; for example, $v[0] = v^0$ is the first element of v.

Mean-Payoff. For an infinite sequence of real numbers, $r^0 r^1 r^2 \cdots \in \mathbb{R}^\omega$, we define the *mean-payoff* value of r, denoted $\mathsf{mp}(r)$, to be the quantity, $\mathsf{mp}(r) = \liminf_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} r^i$.

Temporal Logics. We use LTL [52] with the usual temporal operators, $\mathbf{X}$ ("next") and $\mathbf{U}$ ("until"), and the derived operators $\mathbf{G}$ ("always") and $\mathbf{F}$ ("eventually"). We also use $\mathsf{GR}(1)$ [8], a fragment of LTL given by formulae written in the following form:

$$(\mathbf{GF}\psi_1 \wedge \cdots \wedge \mathbf{GF}\psi_m) \rightarrow (\mathbf{GF}\varphi_1 \wedge \cdots \wedge \mathbf{GF}\varphi_n),$$

where each subformula ψ_i and φ_i is a Boolean combination of atomic propositions. Additionally, we also utilise an extension of LTL known as LTL$^{lim\Sigma}$ [9] that allows mean-payoff assertion such as $\mathsf{mp}(v) \geq c$ for a numeric variable v and a constant number c, which asserts

that the mean-payoff of v is greater than or equal to c along an entire path. The satisfaction of temporal logic formulae is defined using standard semantics. We use the notation $\alpha \models \varphi$ to indicate that the formula φ is satisfied by the infinite sequence α.

Arenas. An *arena* is a tuple $A = \langle \mathrm{N}, \{\mathrm{Ac}_i\}_{i\in\mathrm{N}}, \mathrm{St}, s_{init}, \mathsf{tr}, \mathsf{lab}\rangle$ where N, Ac_i, and St are finite non-empty sets of *players*, *actions* for player i, and *states*, respectively; $s_{init} \in \mathrm{St}$ is the *initial state*; $\mathsf{tr} : \mathrm{St} \times \vec{\mathrm{Ac}} \to \mathrm{St}$ is a *transition function* mapping each pair consisting of a state $s \in \mathrm{St}$ and an *action profile* $\vec{ac} \in \vec{\mathrm{Ac}} = \mathrm{Ac}_1 \times \cdots \times \mathrm{Ac}_n$, with one action for each player, to a successor state; and $\mathsf{lab} : \mathrm{St} \to 2^{\mathrm{AP}}$ is a labelling function, mapping every state to a subset of *atomic propositions*.

A *run* $\rho = (s^0, \vec{ac}^0), (s^1, \vec{ac}^1) \cdots$ is an infinite sequence in $(\mathrm{St} \times \vec{\mathrm{Ac}})^\omega$ such that $\mathsf{tr}(s^k, \vec{ac}^k) = s^{k+1}$ for all k. Runs are generated in the arena by each player i selecting a *strategy* σ_i that will define how to make choices over time. A strategy for i can be understood abstractly as a function $\sigma_i : \mathrm{St}^+ \to \mathrm{Ac}_i$ which maps sequences (or histories) of states into a chosen action for player i. A strategy σ_i is a *finite-memory* strategy if it can be represented by a finite state machine $\sigma_i = (Q_i, q_i^0, \delta_i, \tau_i)$, where Q_i is a finite and non-empty set of *internal states*, q_i^0 is the *initial state*, $\delta_i : Q_i \times \mathrm{St} \to Q_i$ is a deterministic *internal transition function*, and $\tau_i : Q_i \to \mathrm{Ac}_i$ an *action function*. A *memoryless* strategy $\sigma_i : \mathrm{St} \to \mathrm{Ac}_i$ chooses an action based only on the current state of the environment. We write Σ_i for the set of strategies for player i.

A *strategy profile* $\vec{\sigma} = (\sigma_1, \ldots, \sigma_n)$ is a vector of strategies, one for each player. Once a state s and profile $\vec{\sigma}$ are fixed, the game has an *outcome*, i.e., a path in A, denoted by $\pi(\vec{\sigma}, s)$. In this paper, we assume that players' strategies are *finite-memory* and *deterministic*, as such, $\pi(\vec{\sigma}, s)$ is the *unique* path induced by $\vec{\sigma}$, that is, the sequence $s^0 s^1 s^2 \ldots$ such that $s^0 = s$, $s^{k+1} = \mathsf{tr}(s^k, (\tau_1(q_1^k), \ldots, \tau_n(q_n^k)))$, and $q_i^{k+1} = \delta_i(q_i^k, s^k)$, for all $k \geq 0$. Note that such a path is *ultimately periodic* (i.e., a lasso). We simply write $\pi(\vec{\sigma})$ for $\pi(\vec{\sigma}, s_{init})$. We extend this to run induced by $\vec{\sigma}$ in a similar way, i.e., $\rho(\vec{\sigma}) = (s^0, \vec{ac}^0), (s^1, \vec{ac}^1), \ldots$. For an element of a run $\rho(\vec{\sigma})[k] = (s^k, \vec{ac}^k)$, we associate the configuration $\mathsf{cfg}(\vec{\sigma}, k) = (s^k, q_1^k, \ldots, q_n^k)$ with $\tau_i(q_i^k) = \mathrm{ac}_i^k$ for each i.

Multi-Player Games. A *multi-player game* is obtained from an arena A by associating each player with a goal. We consider multi-player games with mean-payoff goals. A *multi-player mean-payoff game* (or simply a *game*) is a tuple $\mathcal{G} = \langle A, (\mathsf{w}_i)_{i\in\mathrm{N}}\rangle$, where A is an arena and $\mathsf{w}_i : \mathrm{St} \to \mathbb{Z}$ is a function mapping, for every player i, every state of the arena into an integer number. Given a game $\mathcal{G} = \langle A, (\mathsf{w}_i)_{i\in\mathrm{N}}\rangle$ and a strategy profile $\vec{\sigma}$, an outcome $\pi(\vec{\sigma})$ in A induces a sequence $\mathsf{lab}(\pi(\vec{\sigma})) = \mathsf{lab}(s^0)\mathsf{lab}(s^1)\cdots$ of sets of atomic propositions, and for each player i, the sequence $\mathsf{w}_i(\pi(\vec{\sigma})) = \mathsf{w}_i(s^0)\mathsf{w}_i(s^1)\cdots$ of weights. The *payoff* of player i is $\mathsf{pay}_i(\pi(\vec{\sigma})) = \mathsf{mp}(\mathsf{w}_i(\pi(\vec{\sigma})))$. By a slight abuse of notation, we write $\mathsf{pay}_i(\vec{\sigma})$ for $\mathsf{pay}_i(\pi(\vec{\sigma}))$, and $\pi(\vec{\sigma}) \models \varphi$ or $\vec{\sigma} \models \varphi$ for $\mathsf{lab}(\pi(\vec{\sigma})) \models \varphi$ for some temporal logic formula φ.

Solution Concept. We focus on a solution concept known as the *core* [5, 54, 40]. To understand the concept of the core, it might be helpful to compare it with the NE and how each can be characterised by *deviations*. Informally, a NE is a strategy profile from which no player has any incentive to *unilaterally* deviate. On the other hand, the core comprises strategy profiles from which no *coalitions* of agents can deviate such that *every* agent in the coalition is strictly better off, regardless of the actions of the remaining players.

Formally, we say that a strategy profile $\vec{\sigma}$ is in the core if for all coalitions $C \subseteq \mathrm{N}$, and strategy profiles $\vec{\sigma}'_C$, there is some counter-strategy profile $\vec{\sigma}'_{-C}$ such that $\mathsf{pay}_i(\vec{\sigma}) \geq \mathsf{pay}_i(\vec{\sigma}'_C, \vec{\sigma}'_{-C})$, for some $i \in C$. Alternatively, as we already discussed above, we can

characterise the core by using the notion of *beneficial deviations*: Given a strategy profile $\vec{\sigma}$ and a coalition $C \subseteq \mathrm{N}, C \neq \varnothing$, we say that the strategy profile $\vec{\sigma}'_C$ is a *beneficial deviation* if for all counter-strategies $\vec{\sigma}'_{-C}$, we have $\mathsf{pay}_i((\vec{\sigma}'_C, \vec{\sigma}'_{-C})) > \mathsf{pay}_i(\vec{\sigma})$ for all $i \in C$. The core then consists of those strategy profiles which admit no beneficial deviations; note that these two definitions are equivalent. For a given game $\mathcal{G}$, let $\mathrm{Core}(\mathcal{G})$ denote the set of strategy profiles in the core of $\mathcal{G}$.

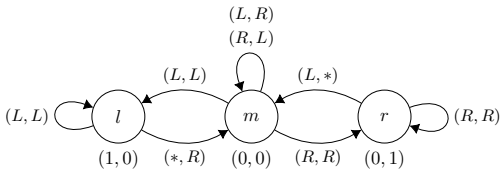


Figure 1 Arena for Example 1. The symbol $*$ is a wildcard that matches all possible actions.

▶ **Example 1.** We illustrate how the core differs from NE, and how cooperation and memory affect the outcome of a game. Consider a game consisting of two players $\{1, 2\}$. The arena is depicted in Figure 1, and the players are initially in m. Each player has two actions: L and R. Player 1 (resp. 2) gets 1 when the play visits l (resp. r) – e.g., tasks assigned to the players, for which they are rewarded upon completion. However, these states can only be visited by agreeing on the actions (e.g., tasks that must be carried out by multiple robots). Observe that player 1 (resp. 2) always choosing L (resp. R) is a NE, and a "bad" one since each player receives a payoff of 0. On the other hand, this bad equilibrium is not included in the core: the players can coordinate/cooperate to alternately visit l and r and obtain higher payoffs (i.e., each receives $\frac{1}{4}$). Furthermore, observe that to execute this plan, the players must remember previously visited states (i.e., finite-memory strategies are necessary). This outcome also corresponds to the liveness property $\mathbf{GF}l \wedge \mathbf{GF}r$ ("the tasks will be completed infinitely often"), which cannot be realised using memoryless strategies.

Vectors and Inequations. Given two vectors $\vec{a}, \vec{b} \in \mathbb{Q}^d$ the notation $\vec{a} \geq \vec{b}$ corresponds to the *component-wise inequality*, and let $||\vec{a}|| = d + \sum_{i \in [\![1,d]\!]} ||a_i||$, a_i is represented using the usual binary encoding of numerators/denominators. The *linear function* $f_{\vec{a}} : \mathbb{R}^d \to \mathbb{R}$ is the function $f_{\vec{a}}(\vec{x}) = \sum_{i \in [\![1,d]\!]} a_i \cdot x_i$. A *linear inequation* is a pair $(\vec{a}, b)$ where $\vec{a} \in \mathbb{Q}^d \setminus \{\vec{0}\}$ and $b \in \mathbb{Q}$. The size of $(\vec{a}, b)$ is $||(\vec{a}, b)|| = ||\vec{a}|| + ||b||$. The *half-space* corresponding to $(\vec{a}, b)$ is the set $\mathsf{hspace}(\vec{a}, b) = \{\vec{x} \in \mathbb{R}^d \mid f_{\vec{a}}(\vec{x}) \leq b\}$. A *linear inequality system* is a set $\lambda = \{(\vec{a}_1, b_1), \ldots, (\vec{a}_l, b_l)\}$ of linear inequations. A *polyhedron* generated by λ is denoted by $\mathsf{poly}(\lambda) = \bigcap_{(\vec{a},b) \in \lambda} \mathsf{hspace}(\vec{a}, b)$. Let P be a polyhedron in $\mathbb{R}^d$ and $C \subseteq D = \{1, \ldots, d\}$, and let $c = |C|$. The *projection* of $P \subseteq \mathbb{R}^d$ on variables with indices in C is the set $\mathsf{proj}_C(P) = \{\vec{x} \in \mathbb{R}^c \mid \exists \vec{y} \in P \wedge \forall i \in C, y_i = x_i\}$.

3 Characterising the core

In this section, we provide a characterisation of the core and other important concepts which we will use to prove our complexity results.

Multi-Mean-Payoff Games. *Multi-mean-payoff games* (MMPGs) [21, 60] are similar to two-player, turn-based, zero-sum mean-payoff games, except the states of the game graph are labelled with k-dimensional integer vectors representing the weights. Player 1's objective is to maximise the mean-payoff of the k-dimensional weight function. Note that since the weights are multidimensional, there is not a unique maximal value in general.

Formally, a multi-mean-payoff game G is a tuple, $G = (V_1, V_2, E, w)$, where V_1, V_2 are the *states* controlled by player 1 and 2 respectively, with $V \coloneqq V_1 \cup V_2$ and $V_1 \cap V_2 = \varnothing$; $E \subseteq V \times V$ is a set of *edges*; $w : V \to \mathbb{Z}^k$ is a *weight function* with $k \in \mathbb{N}$. Given a start state $v^0 \in V_i$, player i chooses an edge $(v^0, v^1) \in E$, and the game moves to state $v^1 \in V_j$. Then player j chooses an edge and the game moves to the specified state, and this continues forever. Paths are defined in the usual way and for a path π, the payoff $\mathsf{pay}(\pi)$ is the vector $(\mathsf{mp}(w_1(\pi)), \ldots, \mathsf{mp}(w_k(\pi)))$. It is shown in [60] that memoryless strategies suffice for player 2 to act optimally, and that the decision problem which asks if player 1 has a strategy that ensures $\mathsf{pay}(\pi) \geq \vec{x}$ from a given state and for some $\vec{x} \in \mathbb{R}^k$ is coNP-complete.

We consider a *sequentialisation* of a game where players are partitioned into two coalitions, $C \subseteq \mathrm{N}$ and $-C = \mathrm{N} \setminus C$. This game is modelled by a MMPG where coalition C acts as player 1 and $-C$ as player 2. The k-dimensional vectors represent the weight functions of players in C. In the case $C = \mathrm{N}$, player 2 is a "dummy" player with no influence in the game.

▶ **Definition 2.** *Let $\mathcal{G} = (A, (\mathsf{w}_i)_{i \in \mathrm{N}})$ be a game with $A = (\mathrm{N}, \mathrm{Ac}, \mathrm{St}, s_{init}, \mathsf{tr}, \mathsf{lab})$ and let $C \subseteq \mathrm{N}$. The sequentialisation of $\mathcal{G}$ with respect to C is the (turn-based two-player) MMPG $G^C = (V_1, V_2, E, w)$ where $V_1 = St$, $V_2 = \mathrm{St} \times \vec{\mathrm{Ac}}_C$; $w : V_1 \cup V_2 \to \mathbb{Z}^c$ is such that $w_i(s) = w_i(s, \vec{ac}_C) = \mathsf{w}_i(s)$; and $E = \{(s, (s, \vec{ac}_C)) \in \mathrm{St} \times (\mathrm{St} \times \vec{\mathrm{Ac}}_C)\} \cup \{((s, \vec{ac}_C), s') \in (\mathrm{St} \times \vec{\mathrm{Ac}}_C) \times \mathrm{St} : \exists \vec{ac}_{-C} \in \vec{\mathrm{Ac}}_{-C}. s' = \mathsf{tr}(s, (\vec{ac}_C, \vec{ac}_{-C}))\}$.*[3]

The construction above is clearly polynomial in the size of the original game $\mathcal{G}$.

Let Σ_2^M be the set of memoryless strategies[4] for player 2. For a strategy $\sigma_2 \in \Sigma_2^M$, the game induced by applying such strategy is given by $G^C[\sigma_2] = (V_1, V_2, E', (w_i)_{i \in C})$ where $E' = \{(s, s') \in E \mid s \in V_1 \vee (s \in V_2 \wedge \sigma_2(s) = s')\}$. That is, a subgame in which player 2 plays according to the memoryless strategy σ_2.

Enforceable Values and Pareto Optimality. We present the definitions of *enforceable values* and *Pareto optimality* in MMPGs [13] below, which we will use for our characterisation of the core.

▶ **Definition 3.** *For a MMPG G^C and a state $s \in V_1 \cup V_2$, define the set of enforceable values that player 1 can guarantee from state s as:*

$$\mathsf{val}(G^C, s) = \{\vec{x} \in \mathbb{R}^c \mid \exists \sigma_1 \forall \sigma_2 \forall j \in C : x_j \leq \mathsf{mp}_j(w_j(\pi((\sigma_1, \sigma_2), s)))\}.$$

A vector $\vec{x} \in \mathbb{R}^c$ is *C-Pareto optimal from s* (or simply *Pareto optimal* when $C = \mathrm{N}$ or C is clear from the context, and $s = s_{init}$) if it is maximal in the set $\mathsf{val}(G^C, s)$. The set of Pareto optimal values is called *Pareto set*, formally defined as:

$$\mathrm{PO}(G^C, s) = \{\vec{x} \in \mathsf{val}(G^C, s) \mid \neg \exists \vec{x}' \in \mathsf{val}(G^C, s) : \vec{x}' \geq \vec{x} \wedge \exists i \,\mathrm{s.t.}\, x'_i > x_i\}.$$

When $s = s_{init}$, we simply write $\mathsf{val}(G^C)$ and $\mathrm{PO}(G^C)$. We naturally extend Pareto optimality to strategy profiles: a strategy profile $\vec{\sigma}$ is C-Pareto optimal if $(\mathsf{pay}_i(\vec{\sigma}))_{i \in C} \in \mathrm{PO}(G^C)$.

A notable aspect of the core in mean-payoff games is that it generally does not coincide with Pareto optimality, as shown in Propositions 4 and 5 below (the proofs are provided in [41]). This stands in sharp contrast to conventional cooperative (transferable utility, superadditive) games in which the core is always included in the Pareto set [20, pp. 24–25].

[3] For $C = \mathrm{N}$, the set $\vec{\mathrm{Ac}}_{-C}$ is empty, and the transition is fully characterised by $\vec{\mathrm{Ac}}_C$. We keep the current notation to avoid clutters.

[4] Here we define a strategy as a mapping from sequences of states to a successor state $\sigma_i : V^* V_i \to V$ for $i \in \{1, 2\}$. A strategy is memoryless when it chooses a successor based on the current state $\sigma_i : V_i \to V$.

▶ **Proposition 4.** *There exist games $\mathcal{G}$ such that $\vec{\sigma} \in \mathrm{Core}(\mathcal{G})$ and $\vec{\sigma}$ is not Pareto optimal.*

▶ **Proposition 5.** *There exist games $\mathcal{G}$ such that $\vec{\sigma}$ is Pareto optimal and $\vec{\sigma} \notin \mathrm{Core}(\mathcal{G})$.*

Discrete Geometry and Values. To characterise the core, we utilise techniques from discrete geometry. First, we provide the definitions of two concepts: *convex hull* and *downward closure*. The *convex hull* of a set $X \subseteq \mathbb{R}^d$ is the set $\mathsf{conv}(X) = \{\sum_{\vec{x}\in X} a_{\vec{x}} \cdot \vec{x} \mid \forall \vec{x} \in X, a_{\vec{x}} \in [0,1] \wedge \sum_{\vec{x}\in X} a_{\vec{x}} = 1\}$. The *downward closure* of a set $X \subseteq \mathbb{R}^d$ is the set $\downarrow X = \{\vec{x} \in \mathbb{Q}^d \mid \exists \vec{x}' \in X, \forall i \in [\![1,d]\!], x_i \leq x'_i\}$. Note that if the set X is finite, then $\mathsf{conv}(X)$ and $\downarrow \mathsf{conv}(X)$ are convex polyhedra, thus can be represented by intersections of some finite number of half-spaces [34, Theorem 3.1.1].

Now, observe that the downward closure of the Pareto set is equal to the set of values that player 1 can enforce, that is, $\downarrow \mathrm{PO}(G^C, s) = \mathsf{val}(G^C, s)$. The set $\mathsf{val}(G^C)$ can also be characterised by the set of *simple cycles* and *strongly connected components* (SCCs) in the arena of G^C [3]. A *simple cycle* within $S \subseteq (V_1 \cup V_2)$ is a finite sequence of states $o = s^0 s^1 \cdots s^k \in S^*$ with $s^0 = s^k$ and for all i and $j, 0 \leq i < j < k, s^i \neq s^j$. Let $\mathbb{C}(S)$ be the set of simple cycles in S, and $\mathrm{SCC}(G^C[\sigma_2])$ the set of SCCs reachable from s_{init} in $G^C[\sigma_2]$. The set of values that player 1 can enforce is characterised by the intersection of all sets of values that it can achieve against memoryless strategies of player 2. Formally, we have the following [13, Theorem 4]:

$$\mathsf{val}(G^C) = \bigcap_{\sigma_2 \in \Sigma_2^M} \bigcup_{S \in \mathrm{SCC}(G^C[\sigma_2])} \downarrow \mathsf{conv}\left(\left\{ \left(\frac{\sum_{j=0}^{k} w_i(o^j)}{|o|} \right)_{i \in C} \,\middle|\, o \in \mathbb{C}(S) \right\}\right).$$

With these definitions in place, we first obtain the following lemma, which shows that the set of enforceable values has polynomial representation.

▶ **Lemma 6.** *The set $\mathsf{val}(G^C)$ can be represented by a finite union of polyhedra $P_1^C, \ldots, P_k^C$, each of them defineable by a system of linear inequations λ_j^C. Moreover, each linear inequation $(\vec{a}, b) \in \lambda_j^C$ can be represented polynomially in the size of G^C.*

Proof. Let $X = \{\vec{x}_1, \ldots, \vec{x}_m\}$ be the set of extreme points of $\mathsf{conv}(\{(\frac{\sum_{j=0}^{k} w_i(o_j)}{|o|})_{i\in C} \mid o \in \mathbb{C}(S)\})$ for a given $S \in \mathrm{SCC}(G^C[\sigma_2])$. Observe that X corresponds to the set of simple cycles in S, as such, for each $\vec{x} \in X$ we have $||\vec{x}||$ that is of polynomial in the size of G^C. As shown in [13, Theorem 3], $\downarrow \mathsf{conv}(X)$ has a system of inequations λ whose each inequation has representation polynomial in c and $\log_2(\max\{||\vec{x}|| \mid \vec{x} \in X\})$. Since this holds for each $\sigma_2 \in \Sigma_2^M$ and for each SCC in $G^C[\sigma_2]$, we obtain the lemma. ◀

Let $\mathrm{PS}(G^C)$ denote the set of polyhedra whose union represents $\mathsf{val}(G^C)$, and for a polyhedron $P_j^C \in \mathrm{PS}(G^C)$, we denote by $\mathcal{H}_j^C$ the set of half-spaces whose intersection corresponds to P_j^C.

Polynomial Witness in the Core. A *polynomial witness* in the core of $\mathcal{G}$ is a vector $\vec{x} \in \mathbb{Q}^n$ such that there exists $\vec{\sigma} \in \mathrm{Core}(\mathcal{G})$ where $(\mathsf{pay}_i(\vec{\sigma}))_{i\in \mathrm{N}} = \vec{x}$ and $\vec{x}$ has a polynomial representation with respect to $\mathcal{G}$. The rest of this section focuses on characterising the core (Theorem 12) and showing the existence of a polynomial witness in a non-empty core (Theorem 13). We start by introducing some concepts and proving a couple of lemmas.

▶ **Definition 7.** *Given a set of player* N *and a coalition $C \subseteq \mathrm{N}$. The* inclusion mapping *of $X \subseteq \mathbb{R}^c$ to subsets of $\mathbb{R}^n$ is the set $\mathcal{F}(X) = \{\vec{y} \in \mathbb{R}^n \mid \exists \vec{x} \in X, \forall j \in C, x_j = y_j\}$.*

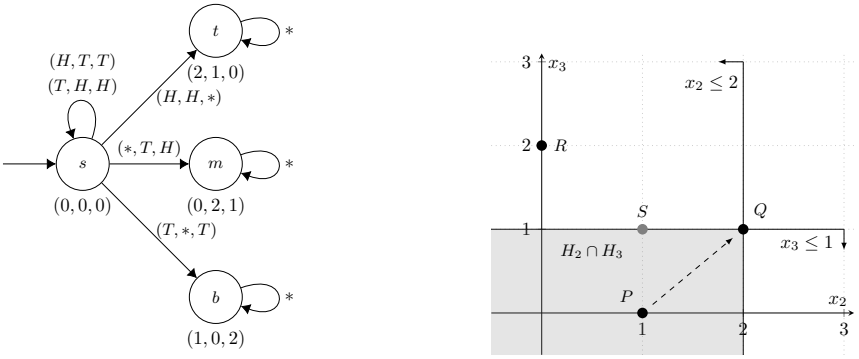


Figure 2 Left: Arena for Example 11. Right: Graphical representation of $\mathsf{val}(G^{\{2,3\}})$. Coordinates P, Q, R, S corresponds to the set $\mathrm{PO}(G'^{\mathrm{N}})$. There is a beneficial deviation by $\{2,3\}$ (dashed arrow) from P (the $\{1,2\}$-Pareto optimal value) to Q (the $\{2,3\}$-Pareto optimal value), but there is no such a deviation from S.

▶ **Definition 8.** *Let* $H = \mathsf{hspace}(\vec{a}, b)$ *be a half-space, the* closed complementary half-space $\overline{H}$ *is given by* $\overline{H} = \{\vec{x} \in \mathbb{R}^d \mid f_{\vec{a}}(\vec{x}) \geq b\}$.

▶ **Lemma 9.** *If* $\vec{\sigma} \in \mathrm{Core}(\mathcal{G})$ *then for all coalitions* $C \subseteq \mathrm{N}$ *and for all polyhedra* $P_j^C \in \mathrm{PS}(G^C)$, *there is a half-space* $H \in \mathcal{H}_j^C$ *such that* $\mathcal{F}((\mathsf{pay}_i(\vec{\sigma}))_{i \in C}) \subseteq \mathcal{F}(\overline{H})$.

Proof. Suppose, for the sake of contradiction, that there is a strategy profile $\vec{\sigma} \in \mathrm{Core}(\mathcal{G})$, coalition $C \subseteq \mathrm{N}$, and polyhedron P_j^C such that for every half-space $H \in \mathcal{H}_j^C$ we have $\mathcal{F}((\mathsf{pay}_i(\vec{\sigma}))_{i \in C}) \not\subseteq \mathcal{F}(\overline{H})$. Thus, it follows that $\mathcal{F}((\mathsf{pay}_i(\vec{\sigma}))_{i \in C}) \subseteq \mathcal{F}(\mathsf{val}(G^C))$ and there exists a vector $\vec{x} \in \mathcal{F}(\mathsf{val}(G^C))$ such that for every player $i \in C$, we have $x_i > \mathsf{pay}_i(\vec{\sigma})$. This implies that there exists a strategy profile $\vec{\sigma}_C$ such that for all counter-strategies $\vec{\sigma}_{-C}$ and players $i \in C$, we have $\mathsf{pay}_i((\vec{\sigma}_C, \vec{\sigma}_{-C})) > \mathsf{pay}_i(\vec{\sigma})$. In other words, there is a beneficial deviation by the coalition C. Therefore, $\vec{\sigma}$ cannot be in the core, leading to a contradiction. ◀

In essence, Lemma 9 states that the absence of a beneficial deviation from a strategy profile $\vec{\sigma}$ can be expressed in terms of polyhedral representations and closed complementary half-spaces. The next lemma, asserts that any value $\vec{x} \in \mathbb{R}^c$ enforceable by a coalition C can also be achieved by the grand coalition N.

▶ **Lemma 10.** *For all coalitions* $C \subseteq \mathrm{N}$, *it holds that* $\mathsf{val}(G^C) \subseteq \mathsf{proj}_C(\mathsf{val}(G^{\mathrm{N}}))$.

Proof. Suppose, for the sake of contradiction, that there is a vector $\vec{x} = (x_1, \ldots, x_c) \in \mathsf{val}(G^C)$ such that $\vec{x} \notin \mathsf{proj}_C(\mathsf{val}(G^{\mathrm{N}}))$. This means that there is some strategy profile $(\vec{\sigma}_C, \vec{\sigma}_{-C})$ and a player $i \in C$ with $\mathsf{pay}_i((\vec{\sigma}_C, \vec{\sigma}_{-C})) > \mathsf{pay}_i(\vec{\sigma})$ for all $\vec{\sigma} \in \Sigma_{C \cup -C}$. This implies that there is a strategy profile $(\vec{\sigma}_C, \vec{\sigma}_{-C}) \notin \Sigma_{C \cup -C}$, i.e., there is a strategy profile that is not included in the set of all strategy profiles, which is a contradiction. ◀

▶ **Example 11.** Consider a game with $\mathrm{N} = \{1,2,3\}$. The arena is depicted in Figure 2. Observe that the game has an empty core: if the players stay in s forever, then $\{1,2\}$ can beneficially deviate to t. If the play goes to t, then $\{2,3\}$ can beneficially deviate to m. Similar arguments can be used for m and b; thus, no strategy profile lies in the core. We can show this using (the contrapositive of) Lemma 9: for instance, take a strategy profile $\vec{\sigma}$ that goes to t, and let $C = \{2,3\}$. Then $\mathsf{val}(G^C)$ can be represented by the intersection of half-spaces

$H_2 = \{\vec{x} \in \mathbb{R}^2 \mid x_2 \leq 2\}$ and $H_3 = \{\vec{x} \in \mathbb{R}^2 \mid x_3 \leq 1\}$ (see Figure 2 right). Coordinate P corresponds to $\vec{\sigma}$, and $\mathcal{F}((\mathsf{pay}_i(\vec{\sigma}))_{i\in\{2,3\}}) \not\subseteq \mathcal{F}(\overline{H}_2)$ and $\mathcal{F}((\mathsf{pay}_i(\vec{\sigma}))_{i\in\{2,3\}}) \not\subseteq \mathcal{F}(\overline{H}_3)$. Thus, $\vec{\sigma}$ is not in the core. Now, suppose we modify the game such that $(\mathsf{w}_i(s))_{i\in\mathrm{N}} = (1,1,1)$; we obtain $\mathrm{PO}(G'^{\mathrm{N}}) = \{(2,1,0),(0,2,1),(1,0,2),(1,1,1)\}$. Let $\vec{\sigma}'$ be a strategy profile that stays in s forever (corresponding to S in Figure 2 right); $\vec{\sigma}'$ is in the core of the modified game, and $\mathcal{F}((\mathsf{pay}_i(\vec{\sigma}'))_{i\in\{2,3\}}) \subseteq \mathcal{F}(\overline{H}_3)$. Indeed for all $C \subseteq \mathrm{N}$ there exists such a half-space. Now if we take the intersection of such half-spaces and the set $\mathsf{val}(G'^{\mathrm{N}}) = {\downarrow}\,\mathrm{PO}(G'^{\mathrm{N}})$, we obtain a non-empty set namely $\{(1,1,1)\}$ which corresponds to a member of the core $\vec{\sigma}'$.

From Example 11, we observe that a member of the core can be found in the intersection of some set of half-spaces and the set of values enforceable by the grand coalition. We formalise this observation in Theorem 12, which provides a necessary and sufficient condition for the non-emptiness of the core.

▶ **Theorem 12.** *The core of a game $\mathcal{G}$ is non-empty if and only if there exists a set of half-spaces I such that*

1. *for all coalitions $C \subseteq \mathrm{N}$ and for all polyhedra $P_j^C \in \mathrm{PS}(G^C)$, $I \cap \mathcal{H}_j^C \neq \varnothing$, and*
2. *there exists a polyhedron $P^{\mathrm{N}} \in \mathrm{PS}(G^N)$ such that $R = \bigcap_{H\in I} \mathcal{F}(\overline{H}) \cap P^{\mathrm{N}} \neq \varnothing$.*

Proof. From left to right. Suppose that $\mathrm{Core}(\mathcal{G}) \neq \varnothing$, then there is a strategy profile $\vec{\sigma} \in \mathrm{Core}(\mathcal{G})$. It follows from Lemma 9 that for each coalition $C \subseteq \mathrm{N}$ and for each polyhedron $P_j^C \in \mathrm{PS}(G^C)$, there exists a half-space $H \in \mathcal{H}_j^C$ such that $\mathcal{F}((\mathsf{pay}_i(\vec{\sigma}))_{i\in C}) \subseteq \mathcal{F}(\overline{H})$. Since this holds for each coalition $C \subseteq \mathrm{N}$ and for each polyhedron $P_j^C \in \mathrm{PS}(G^C)$, then it is the case that there exists a set of half-spaces I such that for all coalitions $C \subseteq \mathrm{N}$ and for all polyhedra $P_j^C \in \mathrm{PS}(G^C)$ there is a half-space $H \in I \cap \mathcal{H}_j^C$, and $(\mathsf{pay}_i(\vec{\sigma}))_{i\in\mathrm{N}} \in \bigcap_{H\in I} \mathcal{F}(\overline{H})$. Furthermore, for each coalition $C \subseteq \mathrm{N}$, it is the case that $\mathcal{F}((\mathsf{pay}_i(\vec{\sigma}))_{i\in C}) \subseteq \mathcal{F}(\mathsf{val}(G^C))$ and by Lemma 10, we have $(\mathsf{pay}_i(\vec{\sigma}))_{i\in C} \in \mathsf{proj}_C(\mathsf{val}(G^{\mathrm{N}}))$. Thus, it is also the case that there exists a polyhedron $P^{\mathrm{N}} \in \mathrm{PS}(G^{\mathrm{N}})$, such that $(\mathsf{pay}_i(\vec{\sigma}))_{i\in\mathrm{N}} \in P^{\mathrm{N}}$. Thus, it follows that $(\mathsf{pay}_i(\vec{\sigma}))_{i\in\mathrm{N}} \in \bigcap_{H\in I} \mathcal{F}(\overline{H}) \cap P^{\mathrm{N}}$ and consequently $\bigcap_{H\in I} \mathcal{F}(\overline{H}) \cap P^{\mathrm{N}} \neq \varnothing$.

From right to left. Suppose $R \neq \varnothing$. Take a vector $\vec{x} \in R$. Since $\vec{x} \in \bigcap_{H\in I} \mathcal{F}(\overline{H})$, then for all coalitions $C \subseteq \mathrm{N}$ there is a player $i \in C$ where $x_i \geq x'_i$ for some vector $\vec{x}' \in \mathcal{F}(\mathrm{PO}(G^C))$. Thus, by the definition of C-Pareto optimality, there exists a player $i \in C$ that cannot strictly improve its payoff without making other player $j \in C, j \neq i$, worse off. Thus, for each coalition $C \subseteq \mathrm{N}$ there is no (partial) strategy profile $\vec{\sigma}_C$ such that for all counter-strategy profiles $\vec{\sigma}_{-C}$ we have $\mathsf{pay}_i((\vec{\sigma}_C, \vec{\sigma}_{-C})) > x_i$ for every player $i \in C$. In other words, for each coalition C and (partial) strategy profile $\vec{\sigma}_C$, there is a counter-strategy profile $\vec{\sigma}_{-C}$ that ensures $\mathsf{pay}_i((\vec{\sigma}_C, \vec{\sigma}_{-C})) \leq x_i$. This means that there is no beneficial deviation by the coalition C. Moreover, since $\vec{x} \in P^{\mathrm{N}}$, then we have $\vec{x} \in \mathsf{val}(G^{\mathrm{N}})$. As such, there exists a strategy profile $\vec{\sigma} \in \Sigma_{\mathrm{N}}$ with $(\mathsf{pay}_i(\vec{\sigma}))_{i\in\mathrm{N}} \geq \vec{x}$ and $\vec{\sigma} \in \mathrm{Core}(\mathcal{G})$. ◀

Using the characterisation of the core from Theorem 12 above, it follows that if the core is non-empty, then the set R is a polyhedron $\mathsf{poly}(\lambda)$ for some system of inequations λ. As such, there exists a vector $\vec{x} \in R$ whose representation is polynomial in n and $\max\{||(\vec{a}, b)|| \mid (\vec{a}, b) \in \lambda\}$ [13, Theorem 2]. By Lemma 6, it is also the case that $\max\{||(\vec{a}, b)|| \mid (\vec{a}, b) \in \lambda\}$ is polynomial in the size of the game. Therefore, we obtain the following.

▶ **Theorem 13.** *Given a game $\mathcal{G}$, if the core is non-empty, then there is $\vec{\sigma} \in \mathrm{Core}(\mathcal{G})$ such that $(\mathsf{pay}_i(\vec{\sigma}))_{i\in\mathrm{N}}$ can be represented polynomially in the size of $\mathcal{G}$.*

Theorem 13 plays a crucial role in our approach to solving Non-Emptiness and E-Core problems discussed in the next section. It guarantees the existence of a polynomial witness if the core is non-empty, allowing it to be guessed and verified in polynomial time.

Algorithm 1 Algorithm for DOMINATED.

input: $\mathcal{G}, s, \vec{x}$
1: guess a coalition $C \subseteq \mathrm{N}$ and a vector $(x'_i)_{i \in C} \in \mathbb{Q}^c$
2: $G^C \leftarrow$ SEQUENTIALISE$(\mathcal{G}, C)$
3: **if** $\forall i \in C, x'_i > x_i$ and $(x'_i)_{i \in C} \in \mathsf{val}(G^C, s)$ **then**
4: **return** YES
5: **return** NO

4 Decision problems

We are now in a position to study each of our decision problems in turn, and establish their complexities. We write $d \in D$ to denote "d is a yes-instance of decision problem D". Our first problem, called DOMINATED, serves as an important foundation for studying the other problems. It is formally defined as follows.

Given: Game $\mathcal{G}$, state s, and vector $\vec{x} \in \mathbb{Q}^n$.
DOMINATED: Is there a coalition $C \subseteq \mathrm{N}$, and a strategy profile $\vec{\sigma}_C$, such that for all counter-strategy profile $\vec{\sigma}_{-C}$, we have $\mathsf{pay}_i(\pi((\vec{\sigma}_C, \vec{\sigma}_{-C}), s)) > x_i$ for each $i \in C$?

▶ **Theorem 14.** *DOMINATED is Σ_2^{P}-complete.*

Proof. Observe that an instance $(\mathcal{G}, s, \vec{x}) \in$ DOMINATED has a witness vector $(x'_i)_{i \in C}$ that lies in the intersection of a polyhedron $P^C \in \mathrm{PS}(G^C, s)$ and the set $\{\vec{y} \in \mathbb{R}^c \mid \forall i \in C : y_i \geq x_i\}$. Such an intersection forms a polyhedron $\mathsf{poly}(\lambda)$, definable by a system of linear inequalities λ. By Lemma 6, each $(\vec{a}, b) \in \lambda$ has polynomial representation in the size of G^C. Therefore, $(x'_i)_{i \in C}$ has a representation that is polynomial in the size of $\mathcal{G}$. To solve DOMINATED, we provide Algorithm 1. The correctness follows directly from the definition of DOMINATED. For the upper bound: since $(x'_i)_{i \in C}$ is of polynomial size, line 1 can be done in NP. In line 2, we have subprocedure SEQUENTIALISE that builds and returns sequentialisation of $\mathcal{G}$ w.r.t. coalition C; this can be done in polynomial time. Finally, line 3 is in coNP [60, Theorem 3, Lemma 6]. Therefore, the algorithm runs in $\mathsf{NP}^{\mathsf{coNP}} = \Sigma_2^{\mathsf{P}}$.

For the lower bound, we reduce from QSAT_2(3DNF) (satisfiability of quantified Boolean formulae with 2 alternations and 3DNF clauses). The complete hardness proof can be found in the appendix.

To illustrate the reduction, consider the formula

$$\Phi = \exists x_1 \exists x_2 \forall y_1 \forall y_2 (x_1 \wedge x_2 \wedge y_1) \vee (x_1 \wedge \neg x_2 \wedge \neg y_2) \vee (x_1 \wedge x_2 \wedge \neg y_1).$$

We build a corresponding game $\mathcal{G}^\Phi$ such that $(\mathcal{G}^\Phi, s_{init}, (-1, -1, -1, -1, -1, 0)) = \chi \in$ DOMINATED if and only if Φ is satisfiable. To this end, we construct the game $\mathcal{G}^\Phi$ in Figure 3 with $\mathrm{N} = \{1, 2, 3, 4, E, A\}$ and the weight function given as vectors, such that for a given vector $(w_1, ..., w_6)$ in state s, $\mathsf{w}_i(s) = w_i, i \in \{1, 2, 3, 4\}$ and $\mathsf{w}_E(s) = w_5, \mathsf{w}_A(s) = w_6$. The s_{sink} (not shown) only has transition to itself and its weights is given by the vector $(-1, -1, -1, -1, -1, 0)$. The intuition is that if Φ is satisfiable, then there is a joint strategy $\vec{\sigma}_C$ by $C = \mathrm{N} \setminus \{A\}$ that guarantees a payoff of 0 for each $i \in C$. If Φ is not satisfiable, then A has a strategy that visits some state y_k (resp. $\neg y_k$) infinitely often and player $2k - 1$ (resp. $2k$) gets payoff < -1. Since y_k (resp. $\neg y_k$) is controlled by $2k - 1$ (resp. $2k$), then the player will deviate to s_{sink}, and $\chi \notin$ DOMINATED. On the other hand, if $\chi \in$ DOMINATED, then

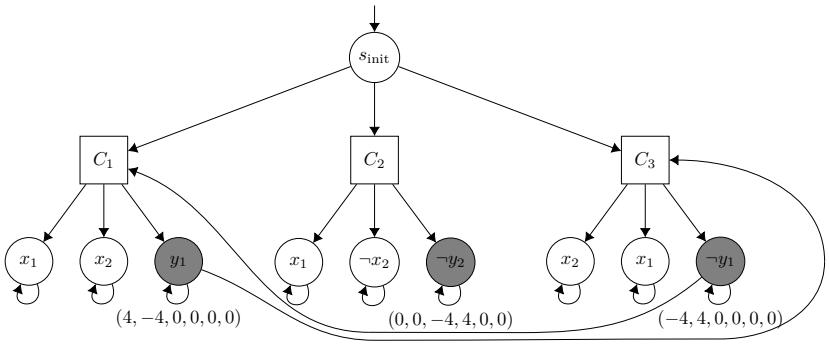


Figure 3 The game arena of $\mathcal{G}^{\Phi}$. White circle states are controlled by E, square by A, grey circles $y_1, \neg y_1, \neg y_2$ by players $1, 2, 4$, respectively. The weight function is given as vectors shown below the states, and states without vectors have $\vec{0}$. Furthermore, each grey circle state also has a transition to s_{sink}, which is not shown in the figure.

there exists a strategy $\vec{\sigma}_C$ which guarantees that the play: (a) ends up in some state x_k or $\neg x_k$, or (b) visits both y_k and $\neg y_k$ infinitely often. For the former, it means that there is a clause with only x-literals, and the latter implies that for all (valid) assignments of y-literals, there is an assignment for x-literal that makes at least one clause evaluate to true. Both cases show that Φ is satisfiable. Now, notice that the formula Φ is satisfiable: take the assignment that set x_1 and x_2 to be both true. Indeed, $\chi \in$ Dominated: the coalition $\{1, 2, 3, 4, E\}$ have a strategy that results in payoff vector $\vec{0}$, e.g., take a strategy profile that corresponds to the cycle $(C_1 y_1 C_3 \neg y_1)^{\omega}$. ◀

Our next problem ∃-Ben-Dev simply asks if a given game has a beneficial deviation from a provided strategy profile:

Given: Game $\mathcal{G}$, strategy profile $\vec{\sigma}$.
∃-Ben-Dev: Does there exist some coalition $C \subseteq$ N such that C has a beneficial deviation from $\vec{\sigma}$?

Notice that ∃-Ben-Dev is closely related to Dominated. Firstly, we fix s to be the initial state. Secondly, instead of a vector, we are given a strategy profile. If we can compute the payoff induced by the strategy profile, then we can immediately reduce ∃-Ben-Dev to Dominated. [57] studies this problem in the memoryless setting, but the approach presented there (i.e., by "running" the strategy profile and calculating the payoff vectors) does not generalise to finite-state strategies $\vec{\sigma}$ as the lasso $\pi(\vec{\sigma})$ may be of exponential size. To illustrate this, consider a profile $\vec{\sigma}$ that acts like a binary counter. We have $|\vec{\sigma}|$ that is of polynomial size, but when we run the profile, we obtain an exponential number of step before we encounter the same configuration of game and strategies states. However, in order to compute the payoff vector of a finite-state strategy profile $\vec{\sigma}$, we only need polynomial space. First, we recall that for deterministic, finite-state strategies, the path $\pi(\vec{\sigma})$ is ultimately periodic (i.e., a lasso-path). As such, there exist $(s^k, \vec{ac}^k)$ and $(s^l, \vec{ac}^l)$ with $l > k$ and $\mathsf{cfg}(\vec{\sigma}, k) = \mathsf{cfg}(\vec{\sigma}, l)$. With this observation, computing the payoff vector can be done by Algorithm 2.

Line 1 can be done non-deterministically in polynomial space. In line 2, we have ComputeIndex subprocedure that computes and returns k, l. This procedure is also in polynomial space: we run the profile $\vec{\sigma}$ from s_{init} and in each step only store the current configuration; for the first time we have $\mathsf{cfg}(\vec{\sigma}, t) = (s^j, q_1^j, \ldots, q_n^j)$, assign $k = t$, and the

Algorithm 2 Algorithm for computing payoff.

input: $\mathcal{G}, \vec{\sigma}$
1: guess s^j and a vector $(q_1^j, \dots, q_n^j) \in \prod_{i \in \mathrm{N}} Q_i$
2: $k, l \leftarrow \textsc{ComputeIndex}\Big(\mathcal{G}, \vec{\sigma}, (s^j, q_1^j, \dots, q_n^j)\Big)$
3: **return** $\left(\frac{\sum_{t=k}^{l} \mathsf{w}_i(\pi(\vec{\sigma})[t])}{l-k}\right)_{i \in \mathrm{N}}$

second time $\mathsf{cfg}(\vec{\sigma}, t') = (s^j, q_1^j, \dots, q_n^j)$, assign $l = t'$, and we are done. Note that this subprocedure returns the smallest pair of k, l. Line 3 is in polynomial time. So, overall we have a function problem that can be solved in NPSPACE, and by Savitch's theorem we obtain the following.

▶ **Lemma 15.** *For a given $\mathcal{G}$ and $\vec{\sigma}$, the payoff vector $(\mathsf{pay}_i(\vec{\sigma}))_{i \in \mathrm{N}}$ can be computed in* PSPACE.

This puts us in position to determine the complexity of ∃-Ben-Dev as follows.

▶ **Theorem 16.** *∃-Ben-Dev is* PSPACE*-complete.*

Proof. To solve ∃-Ben-Dev, we reduce it to Dominated as follows. First, using Algorithm 2 we compute $(\mathsf{pay}_i(\vec{\sigma}))_{i \in \mathrm{N}}$ in PSPACE (Lemma 15). Then, using Algorithm 1 we can check whether $(\mathcal{G}, s_{init}, (\mathsf{pay}_i(\vec{\sigma}))_{i \in \mathrm{N}}) \in$ Dominated. Since $\Sigma_2^{\mathsf{P}} \subseteq$ PSPACE, ∃-Ben-Dev can be solved in PSPACE. For the lower bound, we reduce from the non-emptiness problem of intersection of automata that is known to be PSPACE-hard [44]. The full proof is provided in the appendix. ◀

Another decision problem that is naturally related to the core is asking whether a given strategy profile $\vec{\sigma}$ is in the core of a given game. The problem is formally stated as follows.

Given: Game $\mathcal{G}$ and strategy profile $\vec{\sigma}$.
Membership: Is it the case that $\vec{\sigma} \in \mathrm{Core}(\mathcal{G})$?

Observe that we can immediately see the connection between ∃-Ben-Dev and Membership: they are essentially dual to each other. Therefore, we immediately obtain the following lemma.

▶ **Lemma 17.** *For a given game $\mathcal{G}$ and strategy profile $\vec{\sigma}$, it holds that $\vec{\sigma} \in \mathrm{Core}(\mathcal{G})$ if and only if $(\mathcal{G}, \vec{\sigma}) \notin$ ∃-Ben-Dev.*

Using Lemma 17 and the fact that co-PSPACE = PSPACE, we obtain the following theorem.

▶ **Theorem 18.** *Membership is* PSPACE*-complete.*

In rational verification, we check which temporal logic properties are satisfied by a game's stable outcomes. Two key decision problems are formally defined as follows.

Given: Game $\mathcal{G}$, formula φ.
E-Core: Is it the case that there exists some $\vec{\sigma} \in \mathrm{Core}(\mathcal{G})$ such that $\vec{\sigma} \models \varphi$?
A-Core: Is it the case that for all $\vec{\sigma} \in \mathrm{Core}(\mathcal{G})$, we have $\vec{\sigma} \models \varphi$?

Algorithm 3 Algorithm for Non-Emptiness.

input: $\mathcal{G}$
1: $G^{\mathrm{N}} \leftarrow$ Sequentialise$(\mathcal{G}, \mathrm{N})$
2: guess a vector $\vec{x} \in \mathbb{Q}^n$ s.t. $\vec{x} \in \mathsf{val}(G^{\mathrm{N}})$
3: **if** $(\mathcal{G}, s_{init}, \vec{x}) \in$ Dominated (Alg. 1) **then**
4: **return** NO
5: **return** YES

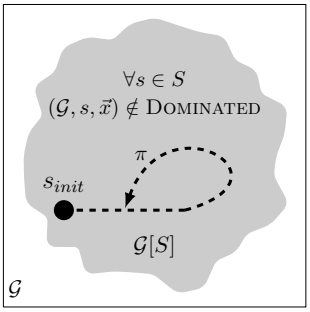


Figure 4 Illustration for solving E-Core.

Algorithm 4 Algorithm for E-Core.

input: $\mathcal{G}, \varphi$
1: $G^{\mathrm{N}} \leftarrow$ Sequentialise$(\mathcal{G}, \mathrm{N})$
2: guess a vector $\vec{x} \in \mathbb{Q}^n$ s.t. $\vec{x} \in \mathsf{val}(G^{\mathrm{N}})$ and set of states $S \subseteq \mathrm{St}$
3: **if** there is no $s \in S$ s.t. $(\mathcal{G}, s, \vec{x}) \in$ Dominated (Algorithm 1) **then**
4: $\mathcal{G}[S] \leftarrow$ UpdateArena$(\mathcal{G}, S)$
5: **if** $\pi \models \psi$ for some $\pi \in \mathcal{G}[S]$ **then**
6: **return** YES
7: **return** NO

To illustrate the decision problems above, let us revisit Example 1. Consider a query of A-Core for Example 1 with property $\varphi = \mathbf{GF}l \wedge \mathbf{GF}r$. Such a query will return a positive answer, i.e., *every* strategy profile that lies in the core satisfies φ.

Another key decision problem in rational verification is determining whether a given game has any stable outcomes. This involves checking if the game has a non-empty core.

Given: Game $\mathcal{G}$.
Non-Emptiness: Is it the case that $\mathrm{Core}(\mathcal{G}) \neq \varnothing$?

As demonstrated in Example 11, there exist mean-payoff games with an empty core – this is in stark contrast to the dichotomous preferences setting (cf. [40, 36]). As such, Non-Emptiness problem is non-trivial in mean-payoff games.

To solve Non-Emptiness, it is important to recall the following two results. Firstly, if a game $\mathcal{G}$ has a non-empty core, then there is a payoff vector $\vec{x}$ resulting from $\vec{\sigma} \in \mathrm{Core}(\mathcal{G})$ whose representation is polynomial (Theorem 13). Secondly, if $\vec{x}$ is a witness for the core, then $(\mathcal{G}, s_{init}, \vec{x}) \notin$ Dominated. With these observations, solving Non-Emptiness can be done by Algorithm 3. The subprocedure in line 1 is polynomial. Line 2 is in NP (Theorem 13) and we call Σ_2^{P} oracle for line 3. Thus, Algorithm 3 runs in Σ_3^{P}. For hardness, we reduce from $\mathrm{QSAT}_3(3\mathrm{CNF})$ (satisfiability of quantified Boolean formulae with 3 alternations and 3CNF clauses). The reduction has a similar flavour to the one used in Theorem 14, albeit a bit more involved. The complete hardness proof is included in the appendix.

▶ **Theorem 19.** *Non-Emptiness is Σ_3^{P}-complete.*

Now we turn our attention to E-Core. Observe that for a game $\mathcal{G}$ and a LTL specification φ, a witness to E-Core would be a path π such that $(\mathsf{pay}_i(\pi))_{i \in \mathrm{N}} \geq (\mathsf{pay}_i(\vec{\sigma}))$ for some $\vec{\sigma} \in \mathrm{Core}(\mathcal{G})$, and $\pi \models \varphi$. Furthermore, a (satisfiable) LTL formula φ has an ultimately

periodic model of size $2^{O(|\varphi|)}$ [56]. Thus, the size of representation of $\mathsf{pay}_i(\pi)$ is at most $\log_2(|W| \cdot 2^{O|\varphi|})$, where W is the maximal absolute value appearing in the weights in $\mathcal{G}$, i.e., $W = \max\{|\mathsf{w}_i(s)| \mid i \in \mathrm{N}, s \in \mathrm{St}\}$. To solve E-Core with a LTL specification φ we use Algorithm 4. An intuitive illustration is provided in Figure 4. We begin by guessing a vector $\vec{x} \in \mathbb{Q}^n$ and a set of states $S \subseteq \mathrm{St}$, such that for every $s \in S, (\mathcal{G}, s, \vec{x}) \notin$ Dominated. Next, we obtain a (sub-)game $\mathcal{G}[S]$ (shaded area) by removing all states $s \notin S$ and edges leading to those removed states. In this new game $\mathcal{G}[S]$, we identify the lasso path π with $\mathsf{pay}_i(\pi) \geq x_i$ for all $i \in \mathrm{N}$ and $\pi \models \varphi$. This path corresponds to a strategy profile in the core since there is no beneficial deviation by any $C \subseteq \mathrm{N}$ in any state in it.

Line 1 is in polynomial time. Line 2 can be done in NP (Theorem 13). In line 3, we can use Algorithm 1 with a slight modification: the state s is *not* given as part of the input, but *included in the first guess* in the algorithm. Clearly, the modified algorithm still runs in Σ_2^{P}. In line 4, we have the subprocedure UpdateArena that returns $\mathcal{G}[S]$; this can be done in polynomial time. For line 5, consider the LTL$^{lim\Sigma}$ formula $\psi := \varphi \wedge \bigwedge_{i \in \mathrm{N}}(\mathsf{mp}(\mathsf{w}_i) \geq x_i)$. Observe that a path in $\mathcal{G}[S]$ satisfying the formula ψ corresponds to a strategy profile $\vec{\sigma}$ such that in every state s in $\pi(\vec{\sigma})$, $(\mathcal{G}, s, (\mathsf{pay}_i(\vec{\sigma}))_{i \in \mathrm{N}}) \notin$ Dominated. Thus, it follows that $\vec{\sigma} \in \mathrm{Core}(\mathcal{G})$, and additionally, $\pi(\vec{\sigma}) \models \varphi$. Finding such a path corresponds to (existential) model checking ψ against the underlying arena of $\mathcal{G}[S]$ – this can be done in PSPACE [9]. Hardness directly follows from setting $\mathsf{w}_i(s) = 0$ for all $i \in \mathrm{N}$ and $s \in \mathrm{St}$. For A-Core, observe that the problem is exactly the dual of E-Core, and since co-PSPACE = PSPACE, we have the following theorem.

▶ **Theorem 20.** *The E-Core and A-Core problems with LTL specifications are* PSPACE*-complete.*

E/A-Core with GR(1) Specifications. The main bottleneck in Algorithm 4 for LTL specifications is in line 5, where the model checking of LTL$^{lim\Sigma}$ formula occurs. This can be avoided by considering classes of properties with easier model checking problem. In this section, we address E/A-Core with GR(1) specifications[5]. The approach is similar to that in [58, Theorem 18]. The main idea is to define a linear program $\mathcal{L}$ such that it has a feasible solution if and only if the condition in line 5 of Algorithm 4 is met.

To this end, first recall that a GR(1) formula φ has the following form

$$\varphi = \bigwedge_{l=1}^{m} \mathbf{GF}\psi_l \rightarrow \bigwedge_{r=1}^{n} \mathbf{GF}\theta_r,$$

and let $V(\psi_l)$ and $V(\theta_r)$ be the subset of states in $\mathcal{G}$ that satisfy the Boolean combinations ψ_l and θ_r, respectively. Observe that property φ is satisfied over a path π if, and only if, either π visits every $V(\theta_r)$ infinitely many times or visits some of the $V(\psi_l)$ only a finite number of times. To check the satisfaction of $\bigwedge_{l=1}^{m} \mathbf{GF}\psi_l$ we define linear programs $\mathcal{L}(\psi_l)$ such that it admits a solution if and only if there is a path π in $\mathcal{G}[S]$ such that $\mathsf{pay}_i(\pi) \geq x_i$ for every player i and visits $V(\psi_l)$ only *finitely many times*. Similarly, for $\bigwedge_{r=1}^{n} \mathbf{GF}\theta_r$, define a linear program $\mathcal{L}(\theta_1, \ldots, \theta_n)$ that admits a solution if and only if there exists a path π in $\mathcal{G}[S]$ such that $\mathsf{pay}_i(\pi) \geq x_i$ for every player i and visits every $V(\theta_r)$ *infinitely many times*. Both linear programs are polynomial in the size of $\mathcal{G}$ and φ, and at least one of them admits a solution if and only if φ is satisfied in some path in $\mathcal{G}[S]$. Therefore, given $\mathcal{G}[S]$ and GR(1) formula φ it is possible to check in polynomial time whether φ is satisfied by a suitable path π in $\mathcal{G}[S]$. The detailed construction is provided in the full version [41].

[5] We could use any other "easy" fragment of LTL to avoid this bottleneck, as we will discuss later.

Table 1 Summary of complexity results. The NE column shows complexity results for the corresponding decision problems with NE. Complexity results for decision problems related to the core in the memoryless setting can be found in [57], whereas for NE in [58, 43].

Problem	Finite Memory	Memoryless	NE
Dominated	Σ_2^{P}-c (Thm. 14)		
∃-Ben-Dev	PSPACE-c (Thm. 16)	NP-c	
Membership	PSPACE-c (Thm. 18)	coNP-c	
Non-Emptiness	Σ_3^{P}-c (Thm. 19)	Σ_2^{P}	NP-c
E-Core with LTL spec.	PSPACE-c (Thm. 20)		PSPACE-c
A-Core with LTL spec.	PSPACE-c (Thm. 20)		PSPACE-c
E-Core with GR(1) spec.	Σ_3^{P}-c (Thm. 21)	Σ_2^{P}	NP-c
A-Core with GR(1) spec.	Π_3^{P}-c (Thm. 21)	Π_2^{P}	coNP-c

Therefore, to solve E-Core with GR(1) specifications, we can use Algorithm 4 with polynomial time check for line 5. Thus, it follows that E-Core with GR(1) specifications can be solved in Σ_3^{P}. The lower bound follows directly from hardness result of Non-Emptiness by setting $\varphi = \top$. Moreover, since A-Core is the dual of E-Core, we obtain the following theorem.

▶ **Theorem 21.** *The E-Core and A-Core problems with* GR(1) *specifications are* Σ_3^{P}*-complete and* Π_3^{P}*-complete, respectively.*

E/A-Core with Other Specifications. We conclude this section by noting that the approach presented here for solving E/A-Core problem is easily generalisable to different types of specification languages without incurring additional computational costs. For instance, the approach for GR(1) is directly applicable to the ω-regular specifications considered in [57]. Furthermore, Algorithm 4 can also be easily adapted for LTL fragments whose witnesses are of polynomial size w.r.t. $\mathcal{G}$ and φ [28, 48]. This can be done by (1) guessing a witness π in line 2 and (2) checking whether $\pi \models \varphi$ and $\mathsf{pay}_i(\pi) \geq x_i$ for all $i \in \mathrm{N}$ in line 5, resulting in the same complexity classes as stated in Theorem 21.

5 Concluding remarks

In this paper, we present a novel characterisation of the core of cooperative concurrent mean-payoff games using discrete geometry techniques which differs from previous methods that relied on logical characterisation and punishment/security values [57, 39]. We have also determined the exact complexity of several related decision problems in rational verification. Our results and other related results from previous work are summarised in Table 1.

It is interesting to note that Non-Emptiness of the core is two rungs higher up the polynomial hierarchy from its NE counterpart. This seems to be induced by the fact that for a given deviation, the punishment/counter-strategy is not static as in NE. It is also worth mentioning that generalising to finite-memory strategies (second column) results in an increase in complexity classes compared to the memoryless setting (third column). In particular, ∃-Ben-Dev and Membership jump significantly from NP-complete and coNP-complete, respectively, to PSPACE-complete. Furthermore, and rather surprisingly, in the finite memory setting, ∃-Ben-Dev and Membership are harder than Non-Emptiness, which sharply contrasts with the memoryless setting. This seems to be caused by the

following: Algorithm 3 for Non-Emptiness is "non-constructive", in the sense that we only care about the existence of a strategy profile that lies in the core without having to explicitly construct one. On the other hand, with Membership, we have to calculate the payoff from a compact representation of a given strategy profile, which requires us to "unpack" the profile.

Our characterisation of the non-emptiness of the core (Theorem 12) provides a way to ensure that the core always has a polynomially representable witness. However, it would be interesting to establish the sufficient and necessary conditions in a broader sense. Previous work has addressed the sufficient and necessary conditions for the non-emptiness of the core in *non-transferable utility* (NTU) games. For example, [55] showed that the core of an NTU game is non-empty when the players have *continuous* and *quasi-concave utility functions*. [59] relaxed the continuity assumption (which aligns more closely with the setting in this paper) and achieved a result similar to [55]. However, their game models differ from ours, and the results do not directly apply to our setting. We conjecture that a similar condition, namely the quasi-concavity of utility functions, plays a vital role in the non-emptiness of the core in concurrent multi-player mean-payoff games. Nevertheless, this still needs to be formally proven and would make for interesting future work.

As previously mentioned, a key difference between the core of concurrent multi-player mean-payoff games and games with dichotomous preferences is that the former may have an empty core. This raises the question: what can we do when the core is empty? One might want to introduce stability, thereby making the core non-empty. One approach, which relates to the above conjecture, involves modifying the utility functions, for instance, through subsidies or rewards [42, 2]. Another approach is to introduce norms [51]. This is an area for future exploration.

It would also be interesting to generalise the current work to decidable classes of imperfect information mean-payoff games [27]. Another potential avenue is to relax the concurrency, for instance, by making agents loosely coupled. A different but intriguing direction would be to investigate the possibility of using our construction and characterisation here to extend ATL* with mean-payoff semantics.

References

1 Alessandro Abate, Julian Gutierrez, Lewis Hammond, Paul Harrenstein, Marta Kwiatkowska, Muhammad Najib, Giuseppe Perelli, Thomas Steeples, and Michael Wooldridge. Rational verification: game-theoretic verification of multi-agent systems. *Applied Intelligence*, 51(9):6569–6584, 2021.

2 S. Almagor, G. Avni, and O. Kupferman. Repairing Multi-Player Games. In *CONCUR*, volume 42 of *LIPIcs*, pages 325–339. Schloss Dagstuhl, 2015.

3 Rajeev Alur, Aldric Degorre, Oded Maler, and Gera Weiss. On omega-languages defined by mean-payoff conditions. In Luca de Alfaro, editor, *Foundations of Software Science and Computational Structures*, pages 333–347, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

4 Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, September 2002. doi:10.1145/585265.585270.

5 Robert J Aumann. The core of a cooperative game without side payments. *Transactions of the American Mathematical Society*, 98(3):539–552, 1961.

6 Raphaël Berthon, Shibashis Guha, and Jean-François Raskin. Mixing probabilistic and non-probabilistic objectives in markov decision processes. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 195–208, 2020.

7 Elisa Bertino, Finale Doshi-Velez, Maria Gini, Daniel Lopresti, and David Parkes. Artificial intelligence and cooperation. Technical Report White Paper 4, Computing Community Consortium, Washington, D.C., October 2020. URL: https://cra.org/ccc/wp-content/uploads/sites/2/2020/11/AI-and-Cooperation.pdf.

8 Roderick Bloem, Barbara Jobstmann, Nir Piterman, Amir Pnueli, and Yaniv Sa'ar. Synthesis of reactive(1) designs. *J. Comput. Syst. Sci.*, 78(3):911–938, May 2012. doi:10.1016/j.jcss.2011.08.007.

9 Udi Boker, Krishnendu Chatterjee, Thomas A. Henzinger, and Orna Kupferman. Temporal specifications with accumulative values. *ACM Trans. Comput. Log.*, 15(4):1–25, August 2014. doi:10.1145/2629686.

10 Olga N Bondareva. Some applications of linear programming methods to the theory of cooperative games. *Problemy kibernetiki*, 10(119):139, 1963.

11 Patricia Bouyer, Romain Brenguier, Nicolas Markey, and Michael Ummels. Pure Nash equilibria in concurrent deterministic games. *Log. Meth. Comput. Sci.*, 11(2), June 2015. doi:10.2168/lmcs-11(2:9)2015.

12 Patricia Bouyer, Orna Kupferman, Nicolas Markey, Bastien Maubert, Aniello Murano, and Giuseppe Perelli. Reasoning about quality and fuzziness of strategic behaviors. *ACM Transactions on Computational Logic*, 24(3):1–38, 2023.

13 Romain Brenguier and Jean-François Raskin. Pareto curves of multidimensional mean-payoff games. In Daniel Kroening and Corina S. Păsăreanu, editors, *Computer Aided Verification*, pages 251–267. Springer International Publishing, 2015.

14 Léonard Brice, Jean-François Raskin, and Marie van den Bogaard. Subgame-perfect equilibria in mean-payoff games. In Serge Haddad and Daniele Varacca, editors, *32nd International Conference on Concurrency Theory, CONCUR 2021, August 24-27, 2021, Virtual Conference*, volume 203 of *LIPIcs*, pages 8:1–8:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.CONCUR.2021.8.

15 Thomas Brihaye, Julie De Pril, and Sven Schewe. Multiplayer cost games with simple nash equilibria. In Sergei Artemov and Anil Nerode, editors, *Logical Foundations of Computer Science*, pages 59–73, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

16 Nils Bulling and Valentin Goranko. Combining quantitative and qualitative reasoning in concurrent multi-player games. *Autonomous Agents and Multi-Agent Systems*, 36:1–33, 2022.

17 Alberto Camacho, Meghyn Bienvenu, and Sheila A McIlraith. Towards a unified view of ai planning and reactive synthesis. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, pages 58–67, 2019.

18 Arindam Chakrabarti, Krishnendu Chatterjee, Thomas A. Henzinger, Orna Kupferman, and Rupak Majumdar. Verifying quantitative properties using bound functions. In Dominique Borrione and Wolfgang Paul, editors, *Correct Hardware Design and Verification Methods*, pages 50–64, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

19 Arindam Chakrabarti, Luca de Alfaro, Thomas A. Henzinger, and Mariëlle Stoelinga. Resource interfaces. In Rajeev Alur and Insup Lee, editors, *Embedded Software*, pages 117–133, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

20 Georgios Chalkiadakis, Edith Elkind, and Michael J. Wooldridge. *Computational Aspects of Cooperative Game Theory*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2011. doi:10.2200/S00355ED1V01Y201107AIM016.

21 Krishnendu Chatterjee, Laurent Doyen, Thomas A. Henzinger, and Jean-François Raskin. Generalized Mean-payoff and Energy Games. In *FSTTCS*, pages 505–516, 2010. doi:10.4230/LIPIcs.FSTTCS.2010.505.

22 Krishnendu Chatterjee, Arkadeb Ghosal, Thomas A. Henzinger, Daniel Iercan, Christoph M. Kirsch, Claudio Pinello, and Alberto Sangiovanni-Vincentelli. Logical reliability of interacting real-time tasks. In *2008 Design, Automation and Test in Europe*, pages 909–914, 2008. doi:10.1109/DATE.2008.4484790.

23 Krishnendu Chatterjee, Thomas A Henzinger, and Nir Piterman. Strategy logic. *Information and Computation*, 208(6):677–693, 2010.

24 Vincent Conitzer and Caspar Oesterheld. Foundations of cooperative AI. In Brian Williams, Yiling Chen, and Jennifer Neville, editors, *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, pages 15359–15367. AAAI Press, 2023. doi:10.1609/AAAI.V37I13.26791.

25 Allan Dafoe, Yoram Bachrach, Gillian Hadfield, Eric Horvitz, Kate Larson, and Thore Graepel. Cooperative ai: machines must learn to find common ground. *Nature*, 593(7857):33–36, 2021.

26 Allan Dafoe, Edward Hughes, Yoram Bachrach, Tantum Collins, Kevin R McKee, Joel Z Leibo, Kate Larson, and Thore Graepel. Open problems in cooperative ai. *arXiv preprint arXiv:2012.08630*, 2020.

27 Aldric Degorre, Laurent Doyen, Raffaella Gentilini, Jean-François Raskin, and Szymon Toruńczyk. Energy and mean-payoff games with imperfect information. In Anuj Dawar and Helmut Veith, editors, *Computer Science Logic*, pages 260–274, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

28 Stéphane Demri and Philippe Schnoebelen. The complexity of propositional linear temporal logics in simple cases. *Information and Computation*, 174(1):84–103, 2002.

29 A. Ehrenfeucht and J. Mycielski. Positional strategies for mean payoff games. *Int. J. Game Theory*, 8(2):109–113, June 1979. doi:10.1007/BF01768705.

30 E. Allen Emerson and Chin-Laung Lei. Modalities for model checking: Branching time logic strikes back. *Sci. Comput. Program.*, 8(3):275–306, June 1987. doi:10.1016/0167-6423(87)90036-0.

31 Ioannis Filippidis, Sumanth Dathathri, Scott C Livingston, Necmiye Ozay, and Richard M Murray. Control design for hybrid systems with tulip: The temporal logic planning toolbox. In *2016 IEEE Conference on Control Applications (CCA)*, pages 1030–1041. IEEE, 2016.

32 Dana Fisman, Orna Kupferman, and Yoad Lustig. Rational synthesis. In *Tools and Algorithms for the Construction and Analysis of Systems: 16th International Conference, TACAS 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2010, Paphos, Cyprus, March 20-28, 2010. Proceedings 16*, pages 190–204. Springer, 2010.

33 Donald B Gillies. Solutions to general non-zero-sum games. *Contributions to the Theory of Games*, 4(40):47–85, 1959.

34 Branko Grünbaum, Volker Kaibel, Victor Klee, and Günter M. Ziegler. *Convex polytopes*. Springer, New York, 2003. URL: http://www.springer.com/mathematics/geometry/book/978-0-387-00424-2.

35 Zhaoyuan Gu, Nathan Boyd, and Ye Zhao. Reactive locomotion decision-making and robust motion planning for real-time perturbation recovery. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 1896–1902, 2022. doi:10.1109/ICRA46639.2022.9812068.

36 Julian Gutierrez, Lewis Hammond, Anthony W. Lin, Muhammad Najib, and Michael J. Wooldridge. Rational verification for probabilistic systems. In Meghyn Bienvenu, Gerhard Lakemeyer, and Esra Erdem, editors, *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021, Online event, November 3-12, 2021*, pages 312–322, 2021. doi:10.24963/kr.2021/30.

37 Julian Gutierrez, Paul Harrenstein, and Michael Wooldridge. Iterated boolean games. *Inform. Comput.*, 242:53–79, June 2015. doi:10.1016/j.ic.2015.03.011.

38 Julian Gutierrez, Paul Harrenstein, and Michael J. Wooldridge. From model checking to equilibrium checking: Reactive modules for rational verification. *Artif. Intell.*, 248:123–157, 2017. doi:10.1016/j.artint.2017.04.003.

39 Julian Gutierrez, Szymon Kowara, Sarit Kraus, Thomas Steeples, and Michael Wooldridge. Cooperative concurrent games. *Artificial Intelligence*, 314:103806, 2023.

40 Julian Gutierrez, Sarit Kraus, and Michael J. Wooldridge. Cooperative concurrent games. In Edith Elkind, Manuela Veloso, Noa Agmon, and Matthew E. Taylor, editors, *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, {AAMAS} '19, Montreal, QC, Canada, May 13-17, 2019*, pages 1198–1206. International Foundation for Autonomous Agents and Multiagent Systems, 2019. URL: `http://dl.acm.org/citation.cfm?id=3331822`.

41 Julian Gutierrez, Anthony W. Lin, Muhammad Najib, Thomas Steeples, and Michael Wooldridge. Characterising and verifying the core in concurrent multi-player mean-payoff games (full version), 2023. `arXiv:2311.15883`.

42 Julian Gutierrez, Muhammad Najib, Giuseppe Perelli, and Michael Wooldridge. Equilibrium Design for Concurrent Games. In Wan Fokkink and Rob van Glabbeek, editors, *30th International Conference on Concurrency Theory (CONCUR 2019)*, volume 140 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 22:1–22:16, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. `doi:10.4230/LIPIcs.CONCUR.2019.22`.

43 Julian Gutierrez, Muhammad Najib, Giuseppe Perelli, and Michael Wooldridge. On the complexity of rational verification. *Annals of Mathematics and Artificial Intelligence*, 91(4):409–430, 2023.

44 Dexter Kozen. Lower bounds for natural proof systems. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 254–266, 1977. `doi:10.1109/SFCS.1977.16`.

45 Orna Kupferman, Giuseppe Perelli, and Moshe Y. Vardi. Synthesis with rational environments. *Ann. Math. Artif. Intel.*, 78(1):3–20, June 2016. `doi:10.1007/s10472-016-9508-8`.

46 Shahar Maoz and Jan Oliver Ringert. Gr (1) synthesis for ltl specification patterns. In *Proceedings of the 2015 10th joint meeting on foundations of software engineering*, pages 96–106, 2015.

47 Shahar Maoz and Yaniv Sa'ar. Assume-guarantee scenarios: Semantics and synthesis. In Robert B. France, Jürgen Kazmeier, Ruth Breu, and Colin Atkinson, editors, *Model Driven Engineering Languages and Systems*, pages 335–351, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

48 Nicolas Markey. Past is for free: on the complexity of verifying linear temporal properties with past. *Acta Informatica*, 40:431–458, 2004.

49 Fabio Mogavero, Aniello Murano, Giuseppe Perelli, and Moshe Y Vardi. Reasoning about strategies: On the model-checking problem. *ACM Transactions on Computational Logic (TOCL)*, 15(4):1–47, 2014.

50 C. Papadimitriou. *Computational complexity*. Addison-Wesley, Reading, Massachusetts, 1994.

51 G. Perelli. Enforcing equilibria in multi-agent systems. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '19, pages 188–196, 2019. URL: `http://dl.acm.org/citation.cfm?id=3306127.3331692`.

52 Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 46–57. IEEE, September 1977. `doi:10.1109/sfcs.1977.32`.

53 Debraj Ray and Rajiv Vohra. Equilibrium binding agreements. *Journal of Economic theory*, 73(1):30–78, 1997.

54 Herbert E Scarf. The core of an n person game. *Econometrica: Journal of the Econometric Society*, pages 50–69, 1967.

55 Herbert E Scarf. On the existence of a coopertive solution for a general class of n-person games. *Journal of Economic Theory*, 3(2):169–181, 1971.

56 A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *J. ACM*, 32(3):733–749, July 1985. `doi:10.1145/3828.3837`.

57 Thomas Steeples, Julian Gutierrez, and Michael Wooldridge. Mean-payoff games with ω-regular specifications. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1272–1280, 2021.

58 M. Ummels and D. Wojtczak. The Complexity of Nash Equilibria in Limit-Average Games. In *CONCUR*, pages 482–496, 2011. doi:10.1007/978-3-642-23217-6_32.

59 Metin Uyanık. On the nonemptiness of the α-core of discontinuous games: Transferable and nontransferable utilities. *Journal of Economic Theory*, 158:213–231, 2015.

60 Yaron Velner, Krishnendu Chatterjee, Laurent Doyen, Thomas A. Henzinger, Alexander Rabinovich, and Jean-François Raskin. The complexity of multi-mean-payoff and multi-energy games. *Inform. Comput.*, 241:177–196, April 2015. doi:10.1016/j.ic.2015.03.001.

61 M. Wooldridge, J. Gutierrez, P. Harrenstein, E. Marchioni, G. Perelli, and A. Toumi. Rational Verification: From Model Checking to Equilibrium Checking. In *{AAAI}*, pages 4184–4191. {AAAI} Press, 2016.

62 Uri Zwick and Mike Paterson. The complexity of mean payoff games on graphs. *Theor. Comput. Sci.*, 158(1-2):343–359, May 1996. doi:10.1016/0304-3975(95)00188-3.

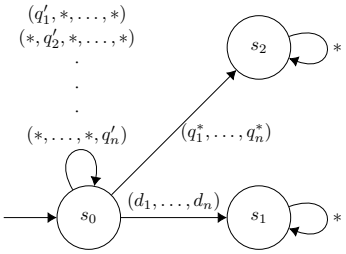
A Appendix: Proofs

A.1 Proof of Theorem 16

▶ **Theorem 16.** $\exists$-*Ben-Dev is* PSPACE-*complete.*

Proof. To solve $\exists$-Ben-Dev, we reduce it to Dominated as follows. First we compute $(\mathsf{pay}_i(\vec{\sigma}))_{i\in \mathrm{N}}$ in PSPACE (Lemma 15). Then we can query whether $(\mathcal{G}, s_{init}, (\mathsf{pay}_i(\vec{\sigma}))_{i\in \mathrm{N}}) \in$ Dominated. Since $\Sigma_2^{\mathsf{P}} \subseteq$ PSPACE, $\exists$-Ben-Dev can be solved in PSPACE.

For the lower bound, we reduce from the non-emptiness problem of the intersection of deterministic finite automata (DFA) that is known to be PSPACE-hard [44]. Let $A_1, \ldots, A_n$ be a set of deterministic finite automata (DFAs), and let $F_i = \{q_i^*\}$ be the set of accepting state of A_i. Note that we can always assume that F_i only has one state; otherwise, we can simply introduce a new symbol in the alphabet (call it a), a new state f_i for A_i, and define the final state of A_i to be f_i, as well as defining $\Delta_i(q, a) := f_i$, for each $q \in F_i$, where Δ_i is the transition function of F_i. We construct from each $A_i = (Q_i, \Sigma_i, \delta_i, q_i^0, F_i)$ a strategy $\sigma_i = (Q_i, q_i^0, \delta_i, \tau_i)$ where $\tau_i(q_i) = q_i$. We build a game with $\mathrm{N} = \{1, \ldots, n\}$ and arena with 3 states $\mathrm{St} = \{s_0, s_1, s_2\}$. For each $i \in \mathrm{N}, \mathrm{Ac}_i = Q_i \cup \{d_i\}$, where d_i is a fresh variable. The transition function is defined by Figure 5 left, and the weight function by Figure 5 right.



$s \in \mathrm{St}$	$(\mathsf{w}_i(s))_{i\in \mathrm{N}}$
s_0	$(0, \ldots, 0)$
s_1	$(1, \ldots, 1)$
s_2	$(1, \ldots, 1)$

Figure 5 Left: The game arena where $q_i' \neq q_i^*$. Right: The weight function of the game.

Given $(\mathcal{G}, \vec{\sigma})$ where $\vec{\sigma} = (\sigma_1, \ldots, \sigma_n)$, we claim that $(\mathcal{G}, \vec{\sigma}) \notin \exists$-Ben-Dev if and only if the intersection of $A_1, \ldots, A_n$ has non-empty language. From left to right: it is easy to see that in order for $\vec{\sigma}$ to admit no beneficial deviation, the game has to eventually enter s_2, because otherwise the grand coalition can deviate to s_1 and obtain better payoffs. The only possible way to enter s_2 is when each of A_i arrives at the accepting state, and thus the intersection has non-empty language. From right to left, we argue in a similar way. ◀

A.2 Proof of Theorem 14

▶ **Theorem 14.** *DOMINATED is Σ_2^{P}-complete.*

Proof. For the lower bound, we reduce from $\mathrm{QSAT}_2(3\mathrm{DNF})$ (satisfiability of quantified Boolean formulae with 2 alternations and 3DNF clauses), which is known to be Σ_2^{P}-hard [50]. Consider a formula of the form $\Phi := \exists x_1 \cdots \exists x_p \forall y_1 \cdots \forall y_q C_1 \vee \cdots \vee C_r$ where each C_i is the conjunction of three literals $C_i = l_{i,1} \wedge l_{i,2} \wedge l_{i,3}$, and the literals are of the form $x_k, \neg x_k, y_k$, or $\neg y_k$. For clauses C and C', we say that they are not *clashing* if there is no literal x_k appears in C and $\neg x_k$ in C'.

For a given formula Φ we build a corresponding game $\mathcal{G}^\Phi$ such that $(\mathcal{G}^\Phi, s_{init}, (-1, \ldots, -1, 0)) \in$ DOMINATED if and only if Φ is satisfiable, as follows.

- $\mathrm{N} = \{1, \ldots, 2q, E, A\}$;
- $\mathrm{St} = \{s_{init}, C_1, \ldots, C_r, l_{1,1}, \ldots, l_{r,3}, s_{sink}\}$ where
  - the states s_{init} and x-literal states are controlled by player E,
  - each state $l_{i,j}$ of the from y_k (resp. $\neg y_k$) is controlled by player $2k$ (resp. $2k-1$), and
  - $\{C_1, \ldots, C_r\}$ (i.e., the clause states) by player A;
- the transition function is given as:
  - from s_{init}, player E can decide to which state in $\{C_1, \ldots, C_r\}$ the play will proceed – she picks the clause;
  - from each state C_i, player A can decide to which state in $\{l_{i,1}, \ldots, l_{i,3}\}$ the play will proceed – he picks the literal;
  - from each $l_{i,j}$, there is a self-loop transition,
  - from each $l_{i,j}$ of the form y_k (resp. $\neg y_k$), the transitions are controlled by player $2k$ (resp. $2k-1$), and defined as follows:
    - there is a transition from $l_{i,j}$ to every $C_h, i \neq h$, where y_k or $\neg y_k$ occurs in C_h, and C_i, C_h are not clashing, and
    - there is also a transition to s_{sink}.
  - s_{sink} has only self-loop transition.
- the weight function is given as:
  - for a literal state $l_{i,j}$
    - if $l_{i,j}$ is of the form y_k, then $\mathsf{w}_{2k-1}(l_{i,j}) = 2q$ and $\mathsf{w}_{2k}(l_{i,j}) = -2q$, and for each $a \in \mathrm{N} \setminus \{2k-1, 2k\}$, $\mathsf{w}_a(l_{i,j}) = 0$;
    - if $l_{i,j}$ is of the form $\neg y_k$, then $\mathsf{w}_{2k-1}(l_{i,j}) = -2q$ and $\mathsf{w}_{2k}(l_{i,j}) = 2q$ and for each $a \in \mathrm{N} \setminus \{2k-1, 2k\}$, $\mathsf{w}_a(l_{i,j}) = 0$;
    - if $l_{i,j}$ is of the form x_k or $\neg x_k$, $(\mathsf{w}_a(l_{i,j}))_{a \in \mathrm{N}} = \vec{0}$.
  - for each non-literal state $s \in \{s_{init}, C_1, \ldots, C_r\}$, we have $(\mathsf{w}_i(s))_{i \in \mathrm{N}} = \vec{0}$.
  - for each $i \in \mathrm{N} \setminus \{A\}, \mathsf{w}_i(s_{sink}) = -1$ and $\mathsf{w}_A(s_{sink}) = 0$.

We show that $(\mathcal{G}^\Phi, s_{init}, (-1, \ldots, -1, 0)) \in$ DOMINATED if and only if the formula Φ is satisfiable.

($\Leftarrow$) Assume that Φ is satisfiable, then there is a (partial) assignment $v(x_1, \ldots, x_p)$ such that the formula $\forall y_1 \cdots \forall y_q C_1 \vee \cdots \vee C_r$ is valid. Let $\sigma_{\mathcal{K}}$ and σ_A denote strategies of coalition $\mathcal{K} = \mathrm{N} \setminus \{A\}$ and player A, respectively. According to [60], it is enough to only consider memoryless strategies σ_A. The strategies correspond to some assignments of variables, that is, by choosing the literal y_k or $\neg y_k$, player A sets the assignment of the literal such that it evaluates to false. Similarly, by choosing the clause C_i, $\mathcal{K}$ pick the correct assignments for literals x_k or $\neg x_k$ in C_i. We distinguish between strategies that are *admissible* and those that are not. A non-admissible strategy is a strategy that chooses two contradictory literals y_k in C and $\neg y_k$ in C'. If σ_A is non-admissible, then $\mathcal{K}$ can achieve $\vec{0}$ by choosing the strategy that alternates between C and C', and thus we have a yes-instance of DOMINATED.

Now suppose that A chooses an admissible strategy σ_A. Then it corresponds to a valid assignment $v(y_1, \ldots, y_q)$. Since for $v(x_1, \ldots, x_p)$ the formula $\forall y_1 \cdots \forall y_q C_1 \vee \cdots \vee C_r$ is valid, the (full) assignment $v(x_1, \ldots, x_p, y_1, \ldots, y_q)$ makes the formula $C_1 \vee \cdots \vee C_r$ evaluate to true. Thus, $\mathcal{K}$ can pick a clause state C_i that is true under $v(x_1, \ldots, x_p, y_1, \ldots, y_q)$ and A picks a literal state of the form x_k or $\neg x_k$ in clause C_i, and not y_k or $\neg y_k$ since it will contradict the assumption that C_i evaluates to true. Therefore, the strategy profile $(\sigma_{\mathcal{K}}, \sigma_A)$ induces the payoff $\vec{0}$, and we have a yes-instance of Dominated.

($\Rightarrow$) Assume that the strategy profile $(\sigma_{\mathcal{K}}, \sigma_A)$ induces a payoff $\mathsf{pay}_j((\sigma_{\mathcal{K}}, \sigma_A)) > -1$ for each $j \in \mathcal{K}$. Let $\mathcal{C}$ and $\overline{\mathcal{C}}$ be the set of clauses that are chosen and not chosen in $(\sigma_{\mathcal{K}}, \sigma_A)$, respectively. We define the (partial) assignment of $v(x_1, \ldots, x_p)$ as follows:

1. for each $C_i \in \mathcal{C}$ and for each literal x_k or $\neg x_k$ in C_i
   a. $v(x_k)$ is true;
   b. $v(\neg x_k)$ is false;
2. for each $C_h \in \overline{\mathcal{C}}$ and for each literal x_k or $\neg x_k$ in C_h, if it does not appear in $C_i \in \mathcal{C}$, then $v(x_k)$ or $v(\neg x_k)$ is true.

Let v' be an (extended) arbitrary assignment of $x_1, \ldots, x_p, y_1, \ldots, y_q$ compatible with $v(x_1, \ldots, x_p)$. Assume towards a contradiction that v' does not make any of the clauses evaluate to true. Then in each $C_i \in \mathcal{C}$, A can choose a literal that makes C_i false. Either (i) A chooses a literal y_k or $\neg y_k$ and there is only a self-loop from the sate y_k or $\neg y_k$, or (ii) we visit some clauses infinitely often. We distinguish between these two cases:

(i) If the run arrives in literal y_k or $\neg y_k$ and there is only a self-loop from the sate y_k or $\neg y_k$, then player $2k$ or $2k-1$ will choose to move into the sink state and the players get payoff $(-1, \ldots, -1, 0)$. This contradicts our previous assumption that $\mathsf{pay}_j((\sigma_{\mathcal{K}}, \sigma_A)) > -1$ for each $j \in \mathcal{K}$;

(ii) If the play visits some clauses infinitely often, then by the construction of the game graph there exists a literal state y_k (resp. $\neg y_k$) visited infinitely often with $\mathsf{w}_{2k-1}(y_k) = -2q$ (resp. $\mathsf{w}_{2k}(\neg y_k) = -2q$) and the state $\neg y_k$ (resp. y_k) is never visited. This means that either $\mathsf{pay}_{2k}((\sigma_{\mathcal{K}}, \sigma_A)) < -1$ or $\mathsf{pay}_{2k-1}((\sigma_{\mathcal{K}}, \sigma_A)) < -1$, and player $2k$ or $2k-1$ will choose to go to s_{sink} and the players get $(-1, \ldots, -1, 0)$. This contradicts our previous assumption that $\mathsf{pay}_j((\sigma_{\mathcal{K}}, \sigma_A)) > -1$ for each $j \in \mathcal{K}$;

This implies that assignment v' makes at least one clause evaluate to true. Furthermore, since this holds for any arbitrary v' compatible with $v(x_1, \ldots, x_p)$, we conclude that $\Phi \in \mathrm{QSAT}_2$. ◀

A.3 Proof of Theorem 19

▶ **Theorem 19.** *Non-Emptiness is Σ_3^{P}-complete.*

Proof. For hardness, we reduce from QSAT_3(3CNF) (satisfiability of quantified Boolean formulae with 3 alternations and 3CNF clauses). Consider a formula of the form

$$\Psi \coloneqq \exists x_1 \cdots \exists x_p \forall y_1 \cdots \forall y_q \exists z_1 \cdots \exists z_t C_1 \wedge \cdots \wedge C_r.$$

where each C_i is the disjunction of three literals $C_i = l_{i,1} \vee l_{i,2} \vee l_{i,3}$, and the literals are of the form $x_k, \neg x_k, y_k, \neg y_k, z_k$, or $\neg z_k$. For clauses C and C', we say that they are not *y-clashing* if there is no literal y_k (resp. $\neg y_k$) appears in C and $\neg y_k$ (resp. y_k) in C'.

For a given formula Ψ we build a corresponding game $\mathcal{G}^{\Psi}$ such that the core of $\mathcal{G}^{\Psi}$ is not empty if and only if Ψ is satisfiable, as follows.

- $\mathrm{N} = \{1, \ldots, 2p, 2p+1, \ldots, 2p+2t, E, A, P, Q, R\}$
- $\mathrm{St} = \{s_{init}, s_{sink}\} \cup \{C_v | 1 \leq v \leq r\} \cup \{l_{1,1}, \ldots, l_{r,3}\}$, where
  - state s_{init} is controlled by player A
  - states $C_1, \ldots, C_r$ are controlled by player E
  - each state $l_{i,j}$ of the form x_k (resp.$\neg x_k$) is controlled by player $2k-1$ (resp. $2k$)
  - each state $l_{i,j}$ of the form z_k (resp.$\neg z_k$) is controlled by player $2(p+k)-1$ (resp. $2(p+k)$) and player A, where player $2(p+k)-1/2(p+k)$ has a "veto" power to either follow player A's decision or, instead, unilaterally choose to go to s_{sink}
  - each state $l_{i,j}$ of the form y_k or $\neg y_k$ is controlled by player A [6]
  - the state s_{sink} is a sink state, and implemented by a gadget that will be explained later.
- the transition function is given as:
  - from s_{init} player A can choose to move to a clause state $C_v, 1 \leq v \leq r$
  - from a state C_v player E can choose to move to a literal state $l_{v,j}$
  - from a literal state $l_{i,j}$ of the form x_k (resp. $\neg x_k$), player $2k-1$ (resp. $2k$) can choose to move to s_{init} or s_{sink}
  - from a literal state $l_{i,j}$ of the form z_k or $\neg z_k$, player E can choose to stay in the current state or to move to any clause state C' that is not y-clashing with C_i.
  - from a literal state $l_{i,j}$ of the form z_k (resp. $\neg z_k$) player $2(p+k)-1$ (resp. $2(p+k)$) can overrule player A's decision, and move to s_{sink}.
- the weight function is given as:
  - for each literal state $l_{i,j}$
    - if it is of the form x_k, then $\mathsf{w}_{2k-1}(l_{i,j}) = 3r, \mathsf{w}_{2k}(l_{i,j}) = -3r$ and for each $a \in \mathrm{N} \setminus \{2k-1, 2k\}, \mathsf{w}_a(l_{i,j}) = 0$
    - if it is of the form $\neg x_k$, then $\mathsf{w}_{2k}(l_{i,j}) = 3r, \mathsf{w}_{2k-1}(l_{i,j}) = -3r$ and for each $a \in \mathrm{N} \setminus \{2k-1, 2k\}, \mathsf{w}_a(l_{i,j}) = 0$
    - if it is of the form z_k, then $\mathsf{w}_{2(p+k)-1}(l_{i,j}) = 3r, \mathsf{w}_{2(p+k)}(l_{i,j}) = -3r$ and for each $a \in \mathrm{N} \setminus \{2(p+k)-1, 2(p+k)\}, \mathsf{w}_a(l_{i,j}) = 0$
    - if it is of the form $\neg z_k$, then $\mathsf{w}_{2(p+k)}(l_{i,j}) = 3r, \mathsf{w}_{2(p+k)-1}(l_{i,j}) = -3r$ and for each $a \in \mathrm{N} \setminus \{2(p+k)-1, 2(p+k)\}, \mathsf{w}_a(l_{i,j}) = 0$
    - otherwise, $\mathsf{w}_a(l_{i,j}) = 0$ for each $a \in \mathrm{N}$.
  - $\mathsf{w}_a(s_{init}) = \mathsf{w}_a(s_\forall) = \mathsf{w}_a(C_i) = 0$ for each $a \in \mathrm{N}$ and $1 \leq i \leq r$.

Now we explain the construction of s_{sink} gadget which is a small variation of a game with an empty core provided in the proof of Proposition 5. Consider a graph arena with four states I, U, M, B in which the players P, Q, R each has two actions: H, T, and only the actions of those players matter in these states (i.e., the rest of the players are dummy players.) The weight function and the transition function are given below – we only specify the transitions for the state I as the other states only have self-loops.

[6] Note that the controller of these states is ultimately not important because, as later defined, from these states we can only go to s_{sink}.

$\mathsf{w}_a(s)$	P	Q	R	E	$a \in \mathrm{N} \setminus \{P, Q, R, E\}$
I	-1	-1	-1	0	1
U	2	1	0	0	1
M	0	2	1	0	1
B	1	0	2	0	1

(a_P, a_Q, a_R)	St
(H, H, H)	U
(H, H, T)	U
(H, T, H)	M
(H, T, T)	I
(T, H, H)	I
(T, H, T)	B
(T, T, H)	M
(T, T, T)	B

Observe that once we enter s_{sink}, we cannot get out. Furthermore, every strategy profile that starts at state I admits beneficial deviations. If the run stays at I forever, the players can beneficially deviate by moving to one of U, M, B. However, if the game ends up at either of those states, then there will always be a coalition (of 2 players) that can beneficially deviate.

We now show that the core of $\mathcal{G}^\Psi$ is not empty if and only if Ψ is satisfiable.

($\Rightarrow$) Suppose $\vec{\sigma} \in \mathrm{Core}(\mathcal{G}^\Psi)$. By the construction of the game, there are three cases:

(a) $\pi(\vec{\sigma})$ visits some literal state of the form x_k (resp. $\neg x_k$) infinitely often, and $\mathsf{pay}_{2k-1}(\vec{\sigma}) \geq 1$ (resp. $\mathsf{pay}_{2k}(\vec{\sigma}) \geq 1$)

(b) $\pi(\vec{\sigma})$ visits some literal state of the form z_k (resp. $\neg z_k$) infinitely often, and $\mathsf{pay}_{2(p+k)-1}(\vec{\sigma}) \geq 1$ (resp. $\mathsf{pay}_{2(p+k)}(\vec{\sigma}) \geq 1$)

(c) both (a) and (b).

The condition $\mathsf{pay}_i(\vec{\sigma}) \geq 1$ is necessary, because otherwise player i can deviate to s_{sink} and gets a payoff of 1 which contradicts $\vec{\sigma}$ being in the core.

We start with (a). This implies that for each clause $C_i, 1 \leq i \leq r$, there is a strategy σ_E for player E that agrees with $\vec{\sigma}$ for choosing a literal state $l_{i,j}$ such that for a literal of the form x_k (resp. $\neg x_k$) we have $\mathsf{w}_{2k-1}(l_{i,j}) \geq 3$ (resp. $\mathsf{w}_{2k}(l_{i,j}) \geq 3$). Moreover, if such a strategy exists, then it is a valid assignment for $x_1, \ldots, x_p$ (i.e., contains no contradictions), since otherwise player A can alternate between the two contradictory choices and gets $\mathsf{pay}_{2k}(\vec{\sigma}) = 0$ or $\mathsf{pay}_{2k-1}(\vec{\sigma}) = 0$, which implies that there is a beneficial deviation by player $2k$ or $2k-1$–contradicting our assumption that $\vec{\sigma}$ being in the core. Since this assignment is valid and makes all clauses evaluate to true, then it is the case that Ψ is satisfiable.

For case (b), the argument is similar to (a). The main difference is that from a literal state $l_{i,j}$ of the form z_k or $\neg z_k$, player A can choose to go to state a C' that is not y-clashing with C_i. This assures that player A can only choose a valid assignment for $y_1, \ldots, y_q$. Moreover, since we have $\mathsf{pay}_{2(p+k)-1}(\vec{\sigma}) \geq 1$ or $\mathsf{pay}_{2(p+k)}(\vec{\sigma}) \geq 1$, then for each clause visited, there exists an assignment of $z_1, \ldots, z_t$ that makes the clause evaluates to true. This assignment is a satisfying assignment for Ψ. For case (c), we combine the arguments from (a) and (b), and obtain a similar conclusion.

($\Leftarrow$) Now, suppose that Ψ is satisfiable, then we have the following cases:

(1) there exists an assignment $v(x_1, \ldots, x_p)$ such that $\Psi(v)$ is a tautology, where $\Psi(v)$ is the resulting formula after applying the assignment $v(x_1, \ldots, x_p)$.

(2) there exists an assignment $v(x_1, \ldots, x_p)$ such that for each assignment $w(y_1, \ldots, y_q)$, there is an assignment $u(z_1, \ldots, z_t)$ that makes $\Psi(v, w, u)$ evaluates to true.

For case (1), we start by turning the assignment $v(x_1, \ldots, x_p)$ into a strategy σ_E that prescribes to which x-literal state $l_{i,j}$ from each clause state C_i the play must proceed. For instance, if $v(x_k)$ is true and x_k is a literal in C_i, then player E will choose to go to x_k from C_i. Notice that it may be the case that there are more than one possible ways to choose a literal according to a given assignment, in which we can just arbitrarily choose one. Observe

that by following σ_E, for all strategy of player A σ_A, corresponding to the assignments of $y_1, \ldots, y_q$, and for all literal state x_k (resp. $\neg x_k$) visited infinitely often in $\pi((\sigma_E, \sigma_A))$ we have $\mathsf{pay}_{2k-1}((\sigma_E, \sigma_A)) \geq 1$ (resp. $\mathsf{pay}_{2k}((\sigma_E, \sigma_A)) \geq 1$). This means that (σ_E, σ_A) admits no beneficial deviation and thus it is in the core.

For case (2), we perform a similar strategy construction as in (1). First, observe that the resulting formula $\Psi(v)$ may contain clauses that evaluate to true. We denote this by $\chi(\Psi(v))$. Notice that if $\chi(\Psi(v)) = \{C_v | 1 \leq v \leq r\}$, then $\Psi(v)$ is a tautology – the same as case (1), and we are done. Otherwise, there is $C_i \notin \chi(\Psi(v))$ and C_i contains some z-literals. Now, using $u(z_1, \ldots, z_t)$ we construct a strategy σ'_E that prescribes which x-literal and z-literal to choose from each clause C_i. Since $\Psi(v, w, u)$ evaluates to true, then for each C_i it is the case that $C_i \in \chi(\Psi(v, w, u))$. This means that for any $C_i, C_j \notin \chi(\Psi(v))$ that are visited infinitely often in a play resulting from (σ_A, σ'_E), there exist no clashing z-literals in C_i, C_j visited infinitely often. That is, for any $C_i, C_j \notin \chi(\Psi(v))$ we have only z_k (resp. $\neg z_k$) visited infinitely often, and by the weight function of the game, we have $\mathsf{pay}_{2(p+k)-1}((\sigma_A, \sigma'_E)) \geq 1$ (resp. $\mathsf{pay}_{2(p+k)}((\sigma_A, \sigma'_E)) \geq 1$). Thus, it is the case that $(\sigma_A, \sigma'_E) \in \mathrm{Core}(\mathcal{G}^\Psi)$. ◀

Decidable (Ac)counting with Parikh and Muller: Adding Presburger Arithmetic to Monadic Second-Order Logic over Tree-Interpretable Structures

Luisa Herrmann
Computational Logic Group, TU Dresden, Germany
Center for Scalable Data Analytics and Artificial Intelligence Dresden/Leipzig, Germany

Vincent Peth
Département d'informatique de l'ÉNS, École normale supérieure, CNRS, PSL University, Paris, France

Sebastian Rudolph
Computational Logic Group, TU Dresden, Germany
Center for Scalable Data Analytics and Artificial Intelligence Dresden/Leipzig, Germany

Abstract

We propose ωMSO⋈BAPA, an expressive logic for describing countable structures, which subsumes and transcends both Counting Monadic Second-Order Logic (CMSO) and Boolean Algebra with Presburger Arithmetic (BAPA). We show that satisfiability of ωMSO⋈BAPA is decidable over the class of labeled infinite binary trees, whereas it becomes undecidable even for a rather mild relaxations. The decidability result is established by an elaborate multi-step transformation into a particular normal form, followed by the deployment of *Parikh-Muller Tree Automata*, a novel kind of automaton for infinite labeled binary trees, integrating and generalizing both Muller and Parikh automata while still exhibiting a decidable (in fact PSPACE-complete) emptiness problem. By means of MSO-interpretations, we lift the decidability result to all tree-interpretable classes of structures, including the classes of finite/countable structures of bounded treewidth/cliquewidth/partitionwidth. We generalize the result further by showing that decidability is even preserved when coupling width-restricted ωMSO⋈BAPA with width-unrestricted two-variable logic with advanced counting. A final showcase demonstrates how our results can be leveraged to harvest decidability results for expressive μ-calculi extended by global Presburger constraints.

2012 ACM Subject Classification Theory of computation → Logic; Theory of computation → Tree languages; Theory of computation → Automata over infinite objects; Theory of computation → Automated reasoning

Keywords and phrases MSO, BAPA, Parikh-Muller tree automata, decidability, MSO-interpretations

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.33

Related Version *An extended version of the paper including more details and full proofs is available at*: `https://arxiv.org/abs/2305.01962`

Funding BMBF (SCADS22B) and SMWK by funding ScaDS.AI Dresden/Leipzig.
Sebastian Rudolph: European Research Council, Consolidator Grant DeciGUT (771779).

1 Introduction

Monadic second-order logic (MSO) is a popular, expressive, yet computationally reasonably well-behaved logical formalism to deal with various classes of finite or countable structures. It allows for expressing "mildly recursive" structural properties like connectedness or reachability, which go beyond first-order logic yet meet crucial modeling demands in verification, database theory, knowledge representation, and other fields of computational logic. The well-understood

link between MSO and automata theory has been very fertile in theory and practice. In particular, the MSO theory of infinite binary trees is decidable by Rabin's famous result [50], and the same holds for structures of bounded treewidth, cliquewidth, and partitionwidth.

Unfortunately, MSO's native capabilities to express cardinality relationships are very limited; they are essentially restricted to fixed thresholds (e.g. "there are at least 10 leaves"). *Counting MSO* [18, 17], denoted CMSO, extends MSO by modulo counting and a finiteness test over sets (e.g. "there is an even number of nodes"), which increases expressiveness in general, while over finite and infinite words or trees, CMSO can be simulated in plain MSO. In contrast, enriching MSO with *cardinality constraints* [40, 41] (as in "all nodes have as many incoming as outgoing edges") increases the expressivity drastically, but causes satisfiability to become undecidable even over finite words. Decidability (over finite words, trees, or graphs of bounded treewidth [42]) can be recovered when confining set variables occurring in cardinality constraints to those existentially quantified in front ($\mathrm{MSO}^{\exists\mathrm{Card}}$). One way to show this is through *Parikh automata* extending finite automata by adding finitely many counters and exploiting the relationship of Presburger arithmetic and semilinear sets [31].

Very recent work [37, 33, 35] extended Parikh word automata to infinite words and investigated the impact of various acceptance conditions, but left a logical characterization as open question. As with the original Parikh automata, one central motivation behind these works is to provide automata-based approaches for specifying and verifying systems beyond regular languages. The study of ω-Parikh automata is motivated by reactive systems, whose behaviors are typically represented by infinite words. Then, the plethora of branching-time approaches in verification should call for a further generalization to ω-tree-automata. Yet, to our knowledge, Parikh automata have not been studied in the context of infinite trees so far.

Another, orthogonal logical approach for describing sets and their cardinalities, motivated by tasks from program analysis and verification, combines the first-order theory of *Boolean algebras* (BA) with *Presburger arithmetic* (PA), resulting in the theory of *BAPA* [44, 45]. As opposed to computationally benign extensions of MSO, BAPA provides stronger support for arithmetic (so one can talk about "all selections with the same number of blue and red nodes" or even "all selections with a share of $70\% - 80\%$ red nodes", modeling statistical information). BAPA usually assumes a finite universe, but can be extended to the countable setting [46]; satisfiability is decidable in either case. However, very regrettably, BAPA lacks non-unary relations, which is outright fatal when it comes to expressing structural properties.

Combining both worlds, we introduce $\omega\mathrm{MSO}{\bowtie}\mathrm{BAPA}$ ['oːmzoǁˌbapa],[1] a logic for countable structures, which extends CMSO by BAPA's set operations and Presburger statements, strictly contains $\mathrm{MSO}^{\exists\mathrm{Card}}$, and allows for sophisticated structural-arithmetic statements (Section 3). We warrant computational manageability by gently controlling the usage of variables, noting that satisfiability turns undecidable otherwise (Section 4). Exhibiting an elaborate transformation (Section 5), we prove that $\omega\mathrm{MSO}{\bowtie}\mathrm{BAPA}$ formulae over trees can be brought into a very restricted *tree normal form* (TNF). We then provide a characterization showing that the sets of ω-trees satisfying TNF formulae coincide with the sets of trees recognized by *Parikh-Muller Tree Automata* (PMTA), a novel automata model designed by us – and the first-ever automaton model on infinite trees capable of testing Parikh conditions (Section 6). PMTA generalize both Muller and Parikh automata and their emptiness is decidable. The decidability of $\omega\mathrm{MSO}{\bowtie}\mathrm{BAPA}$ over the class of labeled infinite binary trees thereby obtained is then lifted to all *tree-interpretable classes*, including vast and practically

[1] Note that the "$\bowtie$" inside the name is meant to be pronounced as lateral click, commonly used by riders and coachmen to urge on their horses, and present in several African languages as a consonant.

relevant classes of finite or countable structures that are bounded in terms of certain width measures (Section 7). Such width-bounded ωMSO$\bowtie$BAPA can be decidably coupled with width-unbounded two-variable logics with advanced counting (Section 8). We demonstrate how to leverage our results to gain decidability results for statistics-enhanced formalisms of the μ-calculus family, which subsumes branching-time logics such as CTL* (Section 9).

2 Preliminaries

As usual, for any $n \in \mathbb{N}$, let $[n] \coloneqq \{1, \dots, n\}$. In order to **count to infinity**, we use $\mathbb{N}$ extended by (countable) infinity ∞, with arithmetics lifted in the usual way; in particular, $\infty + n = \infty + \infty = (n+1) \cdot \infty = \infty$ and $0 \cdot \infty = 0$ as well as $n \leq \infty$ and $\infty \leq \infty$. For countable sets A, let $|A|$ denote the element of $\mathbb{N} \cup \{\infty\}$ that corresponds to their cardinality.

To define **countable structures**, assume the following countable, pairwise disjoint sets: a set $\mathbf{C}$ of *(individual) constants*, denoted by $\mathtt{a}, \mathtt{b}, \mathtt{c}, \dots$, and, for every $n \in \mathbb{N}$, a set $\mathbf{P}_n$ of *n-ary predicates*, denoted by $\mathtt{P}, \mathtt{R}, \mathtt{Q}, \dots$. The set of all predicates will be denoted by $\mathbf{P} \coloneqq \bigcup_{i \in \mathbb{N}} \mathbf{P}_n$, and we let $\mathrm{ar} : \mathbf{P} \to \mathbb{N}$ such that $\mathrm{ar}(\mathtt{Q}) = n$ iff $\mathtt{Q} \in \mathbf{P}_n$. A *(relational) signature* $\mathbb{S}$ is a union $\mathbb{S}_\mathbf{C} \cup \mathbb{S}_\mathbf{P}$ of finite subsets of $\mathbf{C}$ and $\mathbf{P}$, respectively. An $\mathbb{S}$-*structure* is a pair $\mathfrak{A} = (A, \cdot^\mathfrak{A})$, where A is a countable, nonempty set, called the *domain* of $\mathfrak{A}$ and $\cdot^\mathfrak{A}$ is a function that maps every $\mathtt{a} \in \mathbb{S}_\mathbf{C}$ to a domain element $\mathtt{a}^\mathfrak{A} \in A$, and every $\mathtt{Q} \in \mathbb{S}_\mathbf{P}$ to an $\mathrm{ar}(\mathtt{Q})$-ary relation $\mathtt{Q}^\mathfrak{A} \subseteq A^{\mathrm{ar}(\mathtt{Q})}$.

We define **infinite trees** starting from a finite, non-empty set Σ, called *alphabet*. A *(full) infinite binary tree* (often simply called a *tree*) labeled by some alphabet Σ is a mapping $\xi \colon \{0,1\}^* \to \Sigma$. We denote the set of all trees labeled by Σ by T_Σ^ω. A *finite tree* is a mapping $\xi \colon X \to \Sigma$ where X is a finite, prefix-closed subset of $\{0,1\}^*$. The set of all finite trees over Σ will be denoted by T_Σ. We sometimes refer to the domain X of ξ by $\mathrm{pos}(\xi)$, whose elements we call *positions* or *nodes* of ξ. Given a tree $\xi \in T_\Sigma^\omega$ and a finite, prefix-closed set $X \subseteq \{0,1\}^*$, we denote by $\xi_{|X}$ the finite tree in T_Σ that has X as domain and coincides with ξ on X.

An *(infinite) path* π is an infinite sequence $\pi = \pi_1 \pi_2 \dots$ of positions from $\{0,1\}^*$ such that $\pi_1 = \varepsilon$ and, for each $i \geq 1$, $\pi_{i+1} \in (\pi_i \cdot \{0,1\})$. Given a tree $\xi \in T_\Sigma^\omega$ and a path π, we denote by $\xi(\pi)$ the infinite word $\xi(\pi_1)\xi(\pi_2)\dots$ obtained by concatenating the labels of ξ along π. We denote by $\inf(\xi(\pi))$ the set of all labels occurring infinitely often in $\xi(\pi)$.

We will also find it convenient to represent trees over some given alphabet $\Sigma = \{a_1, \dots, a_n\}$ as structures over the signature $\mathbb{S} = \mathbb{S}_\mathbf{P} = \{\succ_0, \succ_1, \mathtt{P}_{a_1}, \dots, \mathtt{P}_{a_n}\}$: Thereby, a tree $\xi \in T_\Sigma^\omega$ will be represented by the structure $\mathfrak{A}_\xi$ with $A_\xi = \{0,1\}^*$, where $\succ_0^{\mathfrak{A}_\xi} = \{(w, w0) \mid w \in \{0,1\}^*\}$ and $\succ_1^{\mathfrak{A}_\xi} = \{(w, w1) \mid w \in \{0,1\}^*\}$ while $\mathtt{P}_{a_i}^{\mathfrak{A}_\xi} = \{u \in \{0,1\}^* \mid \xi(u) = a_i\}$ for each $i \in [n]$. When there is no danger of confusion, we will simply write ξ instead of $\mathfrak{A}_\xi$.

3 Syntax and Semantics of ωMSO$\bowtie$BAPA

We now introduce the logic ωMSO$\bowtie$BAPA. The underlying "design principles" for this logical formalism are to have a language that syntactically subsumes and tightly integrates CMSO and BAPA, while still exhibiting favorable computational properties, even over countably infinite structures. To this end, we will first define the language ωMSO$\cdot$BAPA and then obtain ωMSO$\bowtie$BAPA by imposing some syntactic restrictions on the usage of variables.

▶ **Definition 1** (Syntax of ωMSO$\cdot$BAPA). *Given a signature $\mathbb{S} = \mathbb{S}_\mathbf{C} \cup \mathbb{S}_\mathbf{P}$, together with three countable and pairwise disjoint sets $\mathbf{V}_\mathrm{ind}$ of* individual variables *(denoted $x, y, z, \dots$), $\mathbf{V}_\mathrm{set}$ of* set variables *(denoted $X, Y, Z, \dots$), and $\mathbf{V}_\mathrm{num}$ of* number variables *(denoted $\Bbbk, \ell, m, n \dots$), we define the following sets of expressions by mutual induction:*

- *the set* $\mathbf{I}$ *of* individual terms*:* $\iota ::= \mathtt{a} \mid x$
- *the set* $\mathbf{S}$ *of* set terms *(*$\mathtt{P}$ *being a unary predicate):* $S ::= \{\mathtt{a}\} \mid \mathtt{P} \mid X \mid S^c \mid S_1 \cap S_2 \mid S_1 \cup S_2$
- *the set* $\mathbf{N}$ *of* number terms*:*[2] $t ::= \mathtt{n} \mid \infty \mid \mathtt{k} \mid \#S \mid \mathtt{m}\, t \mid t_1 + t_2$
  (with $n \in \mathbb{N}$ *and* $m \in \mathbb{N} \setminus \{0\}$*; we use typewriter font to indicate that we mean an explicit representation of a constant natural number* n *or* m *rather than the symbol "n" or "m")*
- *the set* $\mathbf{F}$ *of (unrestricted)* formulae*:*

$$\varphi ::= \mathtt{Q}(\iota_1, \ldots, \iota_n) \mid S(\iota) \mid t \leq_{\mathtt{fin}} t' \mid t \leq t' \mid \#S \equiv_{\mathtt{n}} \mathtt{m} \mid \mathrm{Fin}(S) \mid \mathbf{true} \mid \mathbf{false} \mid \neg\varphi \mid \varphi \wedge \varphi' \mid \varphi \vee \varphi' \mid \exists x.\varphi \mid \forall x.\varphi \mid \exists X.\varphi \mid \forall X.\varphi \mid \exists \mathtt{k}.\varphi \mid \forall \mathtt{k}.\varphi$$

The first six types of atomic formulae will be referred to as predicate atoms*,* set atoms*,* classical Presburger atoms*,* modern Presburger atoms*,* modulo atoms*, and* finiteness atoms*, respectively. We use* Presburger atoms *and write* $t \leq_{(\mathtt{fin})} t'$ *to jointly refer to the classical and modern variants. A Presburger atom* $t \leq_{(\mathtt{fin})} t'$ *is called* simple*, if it contains at most one occurrence of a term of the shape* $\#S$ *and no occurrences of number variables.*

▶ **Definition 2** (Semantics of ωMSO·BAPA)**.** *A* variable assignment *(for a structure* $\mathfrak{A}$*) is a function* ν *that maps*

- *every individual variable* $x \in \mathbf{V}_{\mathrm{ind}}$ *to a domain element* $\nu(x) \in A$*,*
- *every set variable* $X \in \mathbf{V}_{\mathrm{set}}$ *to a subset* $\nu(X) \subseteq A$ *of the domain, and*
- *every number variable* $\mathtt{k} \in \mathbf{V}_{\mathrm{num}}$ *to a number* $\nu(\mathtt{k}) \in \mathbb{N} \cup \{\infty\}$*.*

We write $\nu_{x \mapsto a}$*,* $\nu_{X \mapsto A'}$*, and* $\nu_{\mathtt{k} \mapsto n}$ *to denote* ν *updated in the way indicated in the subscript.*
Given an interpretation $\mathfrak{A}$ *and a variable assignment* ν*, we let the function* $\cdot^{\mathfrak{A},\nu}$ *map*

- $\mathbf{I}$ *to* A *by letting* $\mathtt{a}^{\mathfrak{A},\nu} = \mathtt{a}^{\mathfrak{A}}$ *and* $x^{\mathfrak{A},\nu} = \nu(x)$*,*
- $\mathbf{S}$ *to* 2^A *by letting*

$$\{\mathtt{a}\}^{\mathfrak{A},\nu} = \{\mathtt{a}^{\mathfrak{A},\nu}\} \qquad X^{\mathfrak{A},\nu} = \nu(X) \qquad (S_1 \cap S_2)^{\mathfrak{A},\nu} = S_1^{\mathfrak{A},\nu} \cap S_2^{\mathfrak{A},\nu}$$
$$\mathtt{P}^{\mathfrak{A},\nu} = \mathtt{P}^{\mathfrak{A}} \qquad (S^c)^{\mathfrak{A},\nu} = A \setminus S^{\mathfrak{A},\nu} \qquad (S_1 \cup S_2)^{\mathfrak{A},\nu} = S_1^{\mathfrak{A},\nu} \cup S_2^{\mathfrak{A},\nu}$$

- $\mathbf{N}$ *to* $\mathbb{N} \cup \{\infty\}$ *by letting*

$$\mathtt{n}^{\mathfrak{A},\nu} = n \qquad \mathtt{k}^{\mathfrak{A},\nu} = \nu(\mathtt{k}) \qquad (\mathtt{n}\, t)^{\mathfrak{A},\nu} = n \cdot t^{\mathfrak{A},\nu}$$
$$\infty^{\mathfrak{A},\nu} = \infty \qquad (\#S)^{\mathfrak{A},\nu} = |S^{\mathfrak{A},\nu}| \qquad (t_1 + t_2)^{\mathfrak{A},\nu} = t_1^{\mathfrak{A},\nu} + t_2^{\mathfrak{A},\nu}$$

Finally we define satisfaction of formulae from $\mathbf{F}$ *as follows:* $\mathfrak{A}, \nu$ *satisfies*

$\mathtt{Q}(\iota_1,\ldots,\iota_n)$	*iff* $((\iota_1)^{\mathfrak{A},\nu},\ldots,(\iota_n)^{\mathfrak{A},\nu}) \in \mathtt{Q}^{\mathfrak{A}}$	$\varphi_1 \wedge \varphi_2$	*iff* $\mathfrak{A}, \nu \models \varphi_1$ *and* $\mathfrak{A}, \nu \models \varphi_2$
$S(\iota)$	*iff* $\iota^{\mathfrak{A},\nu} \in S^{\mathfrak{A},\nu}$	$\varphi_1 \vee \varphi_2$	*iff* $\mathfrak{A}, \nu \models \varphi_1$ *or* $\mathfrak{A}, \nu \models \varphi_2$
$t_1 \leq t_2$	*iff* $t_1^{\mathfrak{A},\nu} \leq t_2^{\mathfrak{A},\nu}$	$\exists x.\varphi$	*iff* $\mathfrak{A}, \nu_{x \mapsto a} \models \varphi$ *for some* $a \in A$
$t_1 \leq_{\mathtt{fin}} t_2$	*iff* $t_1^{\mathfrak{A},\nu} \leq t_2^{\mathfrak{A},\nu} < \infty$	$\forall x.\varphi$	*iff* $\mathfrak{A}, \nu_{x \mapsto a} \models \varphi$ *for all* $a \in A$
$\#S \equiv_{\mathtt{n}} \mathtt{m}$	*iff* $(\#S)^{\mathfrak{A},\nu} = m \mod n$	$\exists X.\varphi$	*iff* $\mathfrak{A}, \nu_{X \mapsto A'} \models \varphi$ *for some* $A' \subseteq A$
	and $(\#S)^{\mathfrak{A},\nu} < \infty$	$\forall X.\varphi$	*iff* $\mathfrak{A}, \nu_{X \mapsto A'} \models \varphi$ *for all* $A' \subseteq A$
$\mathrm{Fin}(S)$	*iff* $\lvert S^{\mathfrak{A},\nu}\rvert < \infty$	$\exists \mathtt{k}.\varphi$	*iff* $\mathfrak{A}, \nu_{\mathtt{k} \mapsto n} \models \varphi$ *for some* $n \in \mathbb{N} \cup \{\infty\}$
$\neg\varphi$	*iff* $\mathfrak{A}, \nu \not\models \varphi$	$\forall \mathtt{k}.\varphi$	*iff* $\mathfrak{A}, \nu_{\mathtt{k} \mapsto n} \models \varphi$ *for all* $n \in \mathbb{N} \cup \{\infty\}$

Plus, we always let $\mathfrak{A}, \nu \models \mathbf{true}$ *and* $\mathfrak{A}, \nu \not\models \mathbf{false}$*. For a formula* φ*, its* free variables *(denoted* $\mathrm{free}(\varphi)$*) are defined as usual;* φ *is a* sentence *if* $\mathrm{free}(\varphi) = \emptyset$*. For sentences,* ν *does not influence satisfaction, which allows us to write* $\mathfrak{A} \models \varphi$ *and call* $\mathfrak{A}$ *a* model *of* φ *in case* $\mathfrak{A}, \nu \models \varphi$ *holds for any* ν*. We call* φ satisfiable *if it has a model.*

[2] We will consider number terms obtainable from each other through basic transformations (reordering, factoring, summarizing, rules for ∞) as syntactically equal, allowing us to focus on simplified expressions.

Note that, for notational homogeneity, we choose to write $X(\iota)$ instead of $\iota \in X$. Where convenient, we will also make use of the Boolean connectives $\Rightarrow$ and $\Leftrightarrow$ as abbreviations with the usual meaning. While the original syntax of ωMSO·BAPA does not provide an explicit equality predicate, both individual and set equality can be expressed (see further below).

▶ **Definition 3** (Syntax of ωMSO⋈BAPA)**.** *From now on, we will make the following assumption (which is easily obtainable via renaming): In every formula, all quantifications use different variable names and these are disjoint from the names of free variables. Given an ωMSO·BAPA formula φ satisfying this assumption, we analyze its constituents as follows:*

- *A (set or individual) variable is called* assertive*, if it is free, or it is existentially quantified and the quantification does not occur inside the scope of a negation or of a universal (set, individual, or number) quantifier.*
- *The set of* delicate *individual and set variables is the smallest set of (non-assertive) variables satisfying the following:*
  - *Every non-assertive set variable occurring in a non-simple Presburger atom is delicate.*
  - *If some atom contains a delicate (individual or set) variable, then all of this atom's non-assertive (individual or set) variables are delicate.*

Then, φ is an ωMSO⋈BAPA formula *iff each of its predicate atoms $\mathtt{Q}(\cdots)$ contains at most one delicate variable (possibly in multiple occurrences).*

It is easy to see that, despite the above restrictions, ωMSO⋈BAPA entirely encompasses CMSO and $\mathrm{MSO}^{\exists \mathrm{Card}}$ (no delicate variables) as well as BAPA (no predicates of arity >1). For convenience and better readability, we will make use of the following abbreviations.

$$
\begin{array}{lll@{\qquad}lll}
x = y & \coloneqq \forall Z.Z(x) \Leftrightarrow Z(y) & & \exists x \in S.\varphi & \coloneqq \exists x.S(x) \wedge \varphi \\
S \neq \emptyset & \coloneqq \exists z.S(z) & & \forall x \in S.\varphi & \coloneqq \forall x.S(x) \Rightarrow \varphi \\
S \subseteq S' & \coloneqq \forall z.S(z) \Rightarrow S'(z) & & t = t' & \coloneqq (t \leq t') \wedge (t' \leq t) \\
S = S' & \coloneqq (S \subseteq S') \wedge (S' \subseteq S) & & t =_{\mathtt{fin}} t' & \coloneqq (t \leq_{\mathtt{fin}} t') \wedge (t' \leq_{\mathtt{fin}} t)
\end{array}
$$

An analysis of these abbreviations reveals that ωMSO⋈BAPA allows for the variables x, y and set variables in S, S' in these abbreviations to be delicate. We will also employ shortcuts specific to the signature $\{\succ_0, \succ_1, \mathtt{P}_a \mid a \in \Sigma\}$ for Σ-labeled trees. Contrary to above, in these shortcuts, x, y, X, Y must not be delicate to warrant inclusion in ωMSO⋈BAPA (Obs. †).

$$
\begin{aligned}
X(x.i) &\coloneqq \exists y.x \succ_i y \wedge X(y) \\
x \succ y &\coloneqq (x \succ_0 y) \vee (x \succ_1 y) \\
\varphi_{\mathrm{root}}(x) &\coloneqq \neg\exists z.(z \succ x) \\
\varphi_{\uparrow\mathrm{clsd}}(X) &\coloneqq \forall z.X(z.0) \vee X(z.1) \Rightarrow X(z) \\
x \succ^* y &\coloneqq \forall Z.Z(y) \wedge \varphi_{\uparrow\mathrm{clsd}}(Z) \Rightarrow Z(x) \\
x \succ^+ y &\coloneqq (x \succ^* y) \wedge (x \neq y) \\
\varphi_{\downarrow}(x, X) &\coloneqq \forall z.\big(X(z) \Leftrightarrow x \succ^* z\big) \\
\varphi_{\mathrm{path}}(X) &\coloneqq X \neq \emptyset \wedge \varphi_{\uparrow\mathrm{clsd}}(X) \wedge \forall z \in X.\big(X(z.0) \Leftrightarrow \neg X(z.1)\big) \\
\varphi_{\mathrm{inf}}(X) &\coloneqq \exists Z.\varphi_{\mathrm{path}}(Z) \wedge \forall z \in Z.\exists z' \in X.(z \succ^+ z') \\
\varphi^{\cap}_{\mathrm{inf}}(X, Y) &\coloneqq \exists Z.Z \subseteq X \wedge Z \subseteq Y \wedge \varphi_{\mathrm{inf}}(Z)
\end{aligned}
$$

▶ **Example 4.** We use ωMSO$\bowtie$BAPA to specify the class of all labeled infinite binary trees over the alphabet $\Sigma = \{\text{blue}, \text{red}, \text{green}, \text{yellow}, \text{black}\}$ satisfying the following property:

"There is a path X and some node x on X such that the following hold:

1. *For every infinite selection Y of blue nodes from the x-descendants on the path X, there is a selection Y' of red nodes from the whole tree, such that*
   a. *Y and Y' contain the same number of nodes with infinitely many green descendants,*
   b. *Y contains twice as many nodes as Y' having less than 10 yellow descendants.*
2. *For every finite selection Z of blue x-descendants, the total number of nodes lying on paths from x to nodes of Z is even."*

$$
\begin{aligned}
&\exists X.\exists x.\varphi_{\text{path}}(X) \wedge X(x) \wedge \exists V_0.\varphi_{\downarrow}(x, V_0) \wedge \\
&\quad \Big(\exists V_1.\big(\forall v_1.V_1(v_1) \Leftrightarrow \exists V_{\downarrow}^{v_1}.\varphi_{\downarrow}(v_1, V_{\downarrow}^{v_1}) \wedge \neg\text{Fin}(V_{\downarrow}^{v_1} \cap \mathtt{P}_{\text{green}})\big) \wedge \\
&\qquad \exists V_2.\big(\forall v_2.V_2(v_2) \Leftrightarrow \exists V_{\downarrow}^{v_2}.\varphi_{\downarrow}(v_2, V_{\downarrow}^{v_2}) \wedge \#(V_{\downarrow}^{v_2} \cap \mathtt{P}_{\text{yellow}}) \leq 10\big) \wedge \\
&\qquad\quad \Big(\forall Y.\big(\neg\text{Fin}(Y) \wedge Y \subseteq X \cap V_0 \cap \mathtt{P}_{\text{blue}}\big) \Rightarrow \\
&\qquad\qquad \exists Y'.Y' \subseteq \mathtt{P}_{\text{red}} \wedge \#(Y \cap V_1) = \#(Y' \cap V_1) \wedge \#(Y \cap V_2) = 2\,\#(Y' \cap V_2)\Big)\Big) \wedge \\
&\quad \Big(\forall Z.\big(\text{Fin}(Z) \wedge Z \subseteq V_0 \cap \mathtt{P}_{\text{blue}}\big) \Rightarrow \\
&\qquad\quad \exists V_3.\big(\forall v_3.V_3(v_3) \Leftrightarrow (x \succ^{+} v_3 \wedge \exists z \in Z.v_3 \succ^{*} z')\big) \wedge \#V_3 \equiv_2 0\Big)
\end{aligned}
$$

Therein, we use set variables capturing all descendants of x (V_0); all nodes with infinitely many green descendants (V_1); all nodes with less than 10 yellow descendants (V_2); and all nodes between x and elements of Z (V_3). Analysing the variables yields that X, x, V_0, V_1, and V_2 are assertive, while Y and Y' are delicate due to their occurrence in the non-simple Presburger atoms in the fifth line. Delicacy is not inherited further, thus no two delicate variables occur in any predicate atom. Therefore the formula is indeed in ωMSO$\bowtie$BAPA. Note that it is crucial that V_1 and V_2 are defined "prematurely" outside the scope of $\forall Y$, so they become assertive and thus their occurrence in the (non-simple) Presburger atoms does not turn them delicate. This technique of "encapsulating" unary descriptions into assertive set variables unveils significant additional expressiveness of ωMSO$\bowtie$BAPA. See also Section 10 for a discussion on a handier syntax for this.

4 Mildly Extending ωMSO$\bowtie$BAPA Leads to Undecidability

Just slightly relaxing the syntax of ωMSO$\bowtie$BAPA allows us to express Hilbert's 10th Problem.

▶ **Definition 5** (Positive Diophantine Equation). *A* positive Diophantine equation $\mathcal{D}$ *is a tuple* $(NV, M, (n_w)_{w\in M}, (m_w)_{w\in M})$ *where NV is a non-empty, ordered set $\{\varkappa_1, \ldots, \varkappa_k\}$ of number variables; M (the variable products or* monomials*) is a finite and non-empty, prefix-closed set of sorted variable sequences, i.e.,*

$$M \subseteq \{\underbrace{\varkappa_1 \ldots \varkappa_1}_{i_1} \ldots \underbrace{\varkappa_k \ldots \varkappa_k}_{i_k} \mid i_1, \ldots, i_k \in \mathbb{N}\};$$

and all n_w and m_w are from $\mathbb{N}$ and encode the monomial coefficients on either side of the equation. A positive Diophantine equation is solvable *if it admits a solution, where a* solution *for $\mathcal{D} = (NV, M, (n_w)_{w\in M}, (m_w)_{w\in M})$ is a variable assignment $\nu : NV \to \mathbb{N}$ satisfying*

$$\sum\nolimits_{w=\varkappa_1^{i_1}\ldots\varkappa_k^{i_k}\in M} n_w \cdot \nu(\varkappa_1)^{i_1} \cdot \ldots \cdot \nu(\varkappa_k)^{i_k} = \sum\nolimits_{w=\varkappa_1^{i_1}\ldots\varkappa_k^{i_k}\in M} m_w \cdot \nu(\varkappa_1)^{i_1} \cdot \ldots \cdot \nu(\varkappa_k)^{i_k}\,.$$

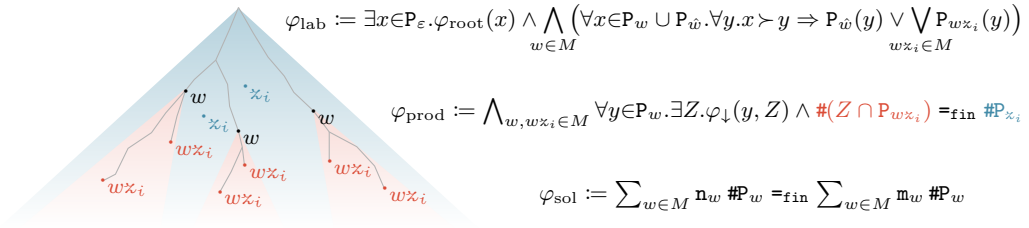


Figure 1 Illustration of the intended model structure and definition of $\varphi_{\mathcal{D}} \coloneqq \varphi_{\text{lab}} \wedge \varphi_{\text{prod}} \wedge \varphi_{\text{sol}}$.

Solvability of positive Diophantine equations is undecidable, which is a straightforward consequence of the undecidability of arbitrary Diophantine equations over integers [48].

We will show that for any $\mathcal{D}$, we can compute an ωMSO·BAPA sentence $\varphi_{\mathcal{D}}$ whose satisfiability over labeled trees coincides with solvability of $\mathcal{D}$, despite $\varphi_{\mathcal{D}}$ being only "minimally outside" ωMSO⋈BAPA — also contrasting the fact that sentences of this shape still warrant decidable satisfiability over finite words [39, Thm. 8.13].

As detailed in Figure 1, we let $\varphi_{\mathcal{D}} \coloneqq \varphi_{\text{lab}} \wedge \varphi_{\text{prod}} \wedge \varphi_{\text{sol}}$ characterize trees labeled by w and $\hat{w}$, for $w \in M$, such that each model ξ of $\varphi_{\mathcal{D}}$ corresponds to a solution ν of $\mathcal{D}$ as follows: for each $\varkappa \in NV$, the number of nodes in ξ labeled with $\varkappa$ (i.e., $\#\mathtt{P}_\varkappa$) equals the number that ν assigns to $\varkappa$. Likewise, for each variable product $w\varkappa_i \in M$, we ensure that $\#\mathtt{P}_{w\varkappa_i} = \#\mathtt{P}_w \cdot \#\mathtt{P}_{\varkappa_i}$. To this end, we stipulate via φ_{lab} that for any w, all w-labeled nodes are pairwise $\succ^*$-incomparable, and every $w\varkappa$-labeled node has exactly one w-labeled ancestor (using the label $\hat{w}$ for "padding" between w and $w\varkappa_i$), and we enforce via φ_{prod} that for any $w, w\varkappa_i \in M$, each subtree rooted in a w-labeled node contains precisely as many $w\varkappa_i$-labeled nodes as there are $\varkappa_i$-labeled nodes in the whole tree. Finally, under the conditions enforced by φ_{lab} and φ_{prod}, φ_{sol} implements that the model indeed encodes a solution of the given $\mathcal{D}$.

While the first conjunct is pure MSO and the third is a variable-free Presburger atom, the second is not in ωMSO⋈BAPA: $\exists Z$ occurs inside the scope of $\forall y$, thus Z is not assertive. Yet, as discussed in Section 3 (Obs. †), this is at odds with Z occurring in $\varphi_{\downarrow}(y, Z)$.

▶ **Proposition 6.** *For any positive Diophantine equation $\mathcal{D}$, satisfaction of $\varphi_{\mathcal{D}}$ over (finite or infinite) labeled trees coincides with solvability of $\mathcal{D}$. Consequently, satisfiability of the class of ωMSO·BAPA sentences of the shape $\varphi_{\mathcal{D}}$ is undecidable.*

5 Transformation into Normal Form

Toward establishing our decidability result, we show that ωMSO⋈BAPA formulae can be transformed into a specific, very restricted normal form. To this end, we use a variety of techniques, mostly known from the literature, but with some adjustments to our setting; thus, due to space, we will restrict ourselves to a high-level description and examples. The normalization procedure is subdivided into two phases: The first phase, establishing the *general normal form* (GNF), is valid independently of the underlying class of structures. The second phase, yielding the *tree normal form* (TNF), is specific to the class of labeled trees.

Given an ωMSO⋈BAPA formula, substitute complex set expressions in modulo and finiteness atoms by new set variables (e.g. $\text{Fin}(\mathtt{P} \cap X)$ becomes $\exists Y.(Y = \mathtt{P} \cap X) \wedge \text{Fin}(Y)$), remove set operations from set atoms (e.g. turning $(\mathtt{P}^c \cap X)(y)$ into $\neg\mathtt{P}(y) \wedge X(y)$), and rewrite all simple Presburger atoms into plain MSO (e.g. $2\,\#\mathtt{P} \leq 3$ becomes $\forall xy.\mathtt{P}(x) \wedge \mathtt{P}(y) \Rightarrow x{=}y$). Then, *skolemize* all assertive variables (e.g. $\exists x.\exists X.\forall y.\mathtt{R}(x,y) \Rightarrow X(y)$ becomes $\forall y.\mathtt{R}(\mathtt{c}_x, y) \Rightarrow \mathtt{P}_X(y)$). Next *"presburgerize"* all non-Presburger atoms containing (only) delicate variables (e.g. replacing $\#X \equiv_3 1$ with $\exists k.\#X =_{\mathtt{fin}} 3k + 1$), which may require to turn delicate individual

into set variables (e.g. $\forall y.\mathtt{P}(y) \Rightarrow X(y)$ becomes $\forall Y.(\#Y = 1) \wedge 1 \leq \#(\mathtt{P} \cap Y) \Rightarrow 1 \leq \#(X \cap Y)$). The resulting formula exhibits a clear separation of variable usage: Presburger atoms use delicate and number variables, all other atoms use non-delicate variables. In a subsequent step, we *"disentangle"* the quantifiers, such that the scopes of quantified number or delicate variables are strictly separated from those of non-delicate variables.[3]

We next apply *"vennification"*: a technique known from BAPA. In essence, we introduce new number variables to count the number of elements contained in every *Venn region*, that is, every possible combination of set (non-)memberships (with this, $\#(\mathtt{P} \cup X) \leq \#\mathtt{P}^c$ becomes $k_{\mathtt{P}\cap X} + k_{\mathtt{P}^c\cap X} + k_{\mathtt{P}\cap X^c} \leq k_{\mathtt{P}^c\cap X} + k_{\mathtt{P}^c\cap X^c}$). This allows us to remove all delicate set variables from our formula. We are now in the setting where we can apply the well-known *quantifier elimination* for Presburger Arithmetic over the "purely arithmetic" subformulae (which may produce new modulo atoms) – since the latter is classically defined for $\mathbb{N}$ instead of $\mathbb{N} \cup \{\infty\}$, we require a pre-processing step implementing a vast case-distinction as to which of the Venn regions are infinite. As a consequence, we obtain a formula free of number variables, with all Presburger atoms being classic and outside any quantifier scope.

Finally, we *"de-skolemize"*: all constants and unary predicates introduced via the initial skolemization, but also by the intermediate transformation steps, are projected away from the signature, re-interpreting them as existentially quantified individual and set variables. We thus recover "proper" equivalence with the initial formula. Last, we bring the formula in disjunctive normal form and pull the trailing existential individual quantifiers inside.

▶ **Definition 7** (General Normal Form). *A* Parikh constraint *is a classical Presburger atom without number variables and where all occurring set terms are set variables. An* ωMSO⋈BAPA *formula is in* general normal form *(GNF), if it is of the shape*

$$\exists X_1. \cdots \exists X_n. \bigvee_{i=1}^{k} \left(\varphi_i \wedge \bigwedge_{j=1}^{l_i} \chi_{i,j}\right),$$

where the φ_i *are CMSO formulae,*[4] *whereas the* $\chi_{i,j}$ *are (unnegated) Parikh constraints.*

▶ **Theorem 8.** *For every* ωMSO⋈BAPA *formula* φ*, it is possible to compute an equivalent formula* φ' *in general normal form.*

We now focus on the case of labeled trees. Very similar to the case of CMSO, under this assumption, we can equivalently transform the GNF formula into one without occurrences of modulo and finiteness atoms. We rewrite $\#X \equiv_{\mathtt{n}} \mathtt{m}$ into the formula

$$\mathrm{Fin}(X) \wedge \exists X_0 ... \exists X_{n-1}. \Big(\exists x. \big(\varphi_{\mathrm{root}}(x) \wedge \bigwedge\nolimits_{\substack{0 \leq i < n \\ i \neq m}} \neg X_i(x)\big) \wedge \forall x. \big((\exists y \in X. x \succ^* y) \vee X_0(x)\big) \wedge \bigwedge\nolimits_{i,j \in \{0,\ldots,n-1\}} \forall z. \big(X_i(z.0) \wedge X_j(z.1) \Rightarrow (\neg X(z) \Rightarrow X_{i \oplus j}(z)) \wedge (X(z) \Rightarrow X_{i \oplus j \oplus 1}(z))\big)\Big),$$

where $\oplus$ denotes addition modulo n. Finally, we replace all occurrences of $\mathrm{Fin}(X)$ by $\varphi_{\mathrm{fin}}(X)$, as defined in Section 3. Thus, when employing ωMSO⋈BAPA to describe labeled trees, we can confine ourselves to an even more restrictive normal form.

▶ **Definition 9** (Tree Normal Form). *An* ωMSO⋈BAPA *formula is in* tree normal form *(TNF), if it is of the shape*

$$\exists X_1. \cdots \exists X_n. \bigvee_{i=1}^{k} \left(\varphi_i \wedge \bigwedge_{j=1}^{l_i} \chi_{i,j}\right),$$

where the φ_i *are plain MSO formulae and the* $\chi_{i,j}$ *are (unnegated) Parikh constraints.*

▶ **Theorem 10.** *For every* ωMSO⋈BAPA *formula* φ*, it is possible to compute a formula* φ' *in tree normal form that is equivalent to* φ *over all labeled infinite binary trees.*

---

[3] While this transformation is not very complicated technically, it may incur nonelementary blowup.
[4] Recall that CMSO is MSO with modulo and finiteness atoms over set variables.

6 Parikh-Muller Tree Automata

In this section, we introduce a novel type of automata, combining and generalizing Parikh tree automata and Muller tree automata. We prove that the tree languages recognized by this automaton type coincide with those definable by TNF formulae. Moreover, we show that the emptiness problem of this automaton model is decidable. In combination, this yields us decidable satisfiability of ωMSO⋈BAPA over labeled infinite binary trees.

Variable-adorned Trees, Semilinear Sets, and Extended Parikh Maps

Given a finite set $\mathbf{V} \subseteq (\mathbf{V}_{\text{ind}} \cup \mathbf{V}_{\text{set}})$, we denote by $\Phi_{\mathbf{V}}$ the set of all variable assignments of variables from $\mathbf{V}$ to elements/subsets of $\{0,1\}^*$. The *set of* $\mathbf{V}$*-models* of a formula φ is the set $\mathcal{L}_{\mathbf{V}}(\varphi) \coloneqq \{(\xi,\nu) \mid \xi \in T^\omega_\Sigma, \nu \in \Phi_{\mathbf{V}}, \xi,\nu \models \varphi\}$ and by $\mathcal{L}(\varphi)$ we mean $\mathcal{L}_{\text{free}(\varphi)}(\varphi)$. To represent $\mathbf{V}$-models, it is convenient to encode variable assignments $\nu \in \Phi_{\mathbf{V}}$ into the alphabet. For this, we let $\Sigma_{\mathbf{V}} = \Sigma \times 2^{\mathbf{V}}$ be a new alphabet and identify $\Sigma_\emptyset$ with Σ. We say that a tree $\xi \in T^\omega_{\Sigma_{\mathbf{V}}}$ is *valid* (i.e., it encodes a variable assignment) if for each individual variable x in $\mathbf{V}$ there is exactly one position in ξ where x occurs. As there is a bijection between $T^\omega_\Sigma \times \Phi_{\mathbf{V}}$ and the set of all valid trees in $T^\omega_{\Sigma_{\mathbf{V}}}$, we use these two views interchangeably.

A set $C \subseteq \mathbb{N}^s$, $s \geq 1$, is *linear* if it is of the form $C = \{\vec{v}_0 + \sum_{i\in[l]} m_i \vec{v}_i \mid m_1,\dots,m_l \in \mathbb{N}\}$ for some $l \in \mathbb{N}$ and vectors $\vec{v}_0,\dots,\vec{v}_l \in \mathbb{N}^s$. Any finite union of linear sets is called *semilinear*.

Given two vectors $\vec{v} = (v_1,\dots,v_s) \in \mathbb{N}^s$ and $\vec{v}' = (v'_1,\dots,v'_{s'}) \in \mathbb{N}^{s'}$, we define their *concatenation* $\vec{v}\cdot\vec{v}'$ as the vector $(v_1,\dots,v_s,v'_1,\dots,v'_{s'}) \in \mathbb{N}^{s+s'}$. This definition is lifted to sets by letting $C \cdot C' = \{\vec{v}\cdot\vec{v}' \mid \vec{v}\in C,\ \vec{v}' \in C'\} \subseteq \mathbb{N}^{s+s'}$ for $C \subseteq \mathbb{N}^s, C' \subseteq \mathbb{N}^{s'}$.

▶ **Lemma 11** ([30, 31]). *The family of semilinear sets of $\mathbb{N}^s$ coincides with the family of Presburger sets of $\mathbb{N}^s$ (i.e., sets of the form $\{(x_1,\dots,x_s) \mid \varphi(x_1,\dots,x_s)\}$ for a Presburger formula φ). Semilinear sets are closed under union, intersection, complement, and concatenation.*

Given an alphabet Σ and some finite $D \subseteq \mathbb{N}^s$ for $s \geq 1$, our automaton model works with symbols from $\Sigma \times D$. Thus we use the *projections* $\cdot_\Sigma : \Sigma \times D \to \Sigma$ with $(a,d)_\Sigma = a$ and $\cdot_D : \Sigma \times D \to D$ with $(a,d)_D = d$, which we will also apply to finite and infinite trees, resulting in a pointwise substitution of labels. Moreover, the *extended Parikh map* $\Psi\colon T_{\Sigma\times D} \to \mathbb{N}^s$ is defined for each finite, non-empty tree $\xi \in T_{\Sigma\times D}$ by $\Psi(\xi) = \sum_{i\in\text{pos}(\xi)} (\xi(i))_D$.

Automaton Model

We now formally introduce our notion of a *Parikh-Muller Tree Automaton (PMTA)*, which recognizes infinite trees employing a Muller acceptance condition while also testing some finite initial tree part for an arithmetic property related to Parikh's commutative image [49]. This is implemented by utilizing a finite number of global counters, which are "blindly" increased throughout the run, but are read off only once a posteriori – when it is verified whether the tuple of the final counter values belongs to a given semilinear set.

▶ **Definition 12** (Parikh-Muller Tree Automaton). *Let Σ be an alphabet, let $s \in \mathbb{N}\setminus\{0\}$, let $D \subseteq \mathbb{N}^s$ be finite, and denote $(\Sigma\times D)\cup\Sigma$ by Ξ. A PMTA (of* dimension *s) is a tuple $\mathcal{A} = (Q,\Xi,q_I,\Delta,\mathcal{F},C)$ where $Q = Q_P \cup Q_\omega \cup \{q_I\}$ is a finite set of* states *with Q_P,Q_ω disjoint and q_I being the* initial state, *$\Delta = \Delta_P \cup \Delta_\omega$ is the* transition relation *with*

$$\Delta_P \subseteq (Q_P \cup \{q_I\}) \times (\Sigma\times D) \times Q \times Q \quad \text{and} \quad \Delta_\omega \subseteq (Q_\omega \cup \{q_I\}) \times \Sigma \times Q_\omega \times Q_\omega,$$

$\mathcal{F} \subseteq 2^{Q_\omega}$ is a set of final state sets, *and $C \subseteq \mathbb{N}^s$ is a semilinear set named* final constraint.

▶ **Definition 13** (Semantics of PMTA). *A* run *of $\mathcal{A}$ on a tree $\zeta \in T_\Xi^\omega$ is a tree $\kappa_\zeta \in T_Q^\omega$ whose root is labeled with q_I and which respects Δ jointly with ζ. By definition of Δ, if a run exists, then $\zeta^{-1}(\Sigma \times D)$ is prefix-closed; we denote $\zeta_{|\zeta^{-1}(\Sigma\times D)}$ by ζ_{cnt}. A run κ_ζ is* accepting *if*
1. *for each path π, we have $\inf(\kappa_\zeta(\pi)) \in \mathcal{F}$, and*
2. *if $\mathrm{pos}(\zeta_{\mathrm{cnt}}) \neq \emptyset$, then $\Psi(\zeta_{\mathrm{cnt}}) \in C$.*

Note that, by the first condition, κ_ζ being accepting implies finiteness of ζ_{cnt} and, thus, well-definedness of the sum in $\Psi(\zeta_{\mathrm{cnt}})$. The set of all accepting runs of $\mathcal{A}$ on ζ will be denoted by $\mathrm{Run}_\mathcal{A}(\zeta)$. Then, the tree language of $\mathcal{A}$*, denoted by $\mathcal{L}(\mathcal{A})$, is the set*

$$\mathcal{L}(\mathcal{A}) \coloneqq \{\xi \in T_\Sigma^\omega \mid \exists \zeta \in T_\Xi^\omega \text{ with } \mathrm{Run}_\mathcal{A}(\zeta) \neq \emptyset \text{ and } (\zeta)_\Sigma = \xi\}\,.$$

We highlight that, by choosing $\Delta_P = \emptyset$, we reobtain the well-known concept of a *Muller tree automaton (MTA)*. In this case, we can drop Q_P, D, Δ_P, and C from $\mathcal{A}$'s specification without affecting its semantics. Thus, we define an MTA $\mathcal{A}$ by the tuple $(Q_\omega, \Sigma, q_I, \Delta_\omega, \mathcal{F})$.

For alphabets Σ, Γ, a *relabeling* (from Σ to Γ) is a mapping $\tau\colon \Sigma \to \mathcal{P}(\Gamma)$. We extend it to a mapping $\tau\colon T_\Sigma^\omega \to \mathcal{P}(T_\Gamma^\omega)$ by letting $\xi' \in \tau(\xi)$ if and only if for each position $\varrho \in \{0,1\}^*$, we have $\xi'(\varrho) \in \tau(\xi(\varrho))$. Note that the reverse τ^{-1} of a relabeling τ is again a relabeling.

▶ **Proposition 14.** *The set of tree languages recognized by Parikh-Muller tree automata is closed under union, intersection, and relabeling.*

Proof (sketch). As the proof techniques are rather standard and some of them were already presented in earlier work [37], we only sketch the main ideas here. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be PMTA.

For the *union*, we construct a PMTA that starts in a fresh initial state. From there, it can either enter the transitions of $\mathcal{A}_1$ or of $\mathcal{A}_2$; we keep apart the final constraints of $\mathcal{A}_1$ and $\mathcal{A}_2$ by using one additional dimension. The *intersection* PMTA is constructed as the Cartesian product of $\mathcal{A}_1$ and $\mathcal{A}_2$; it uses the concatenation of final constraints of both given PMTA and, as $\mathcal{A}_1$ and $\mathcal{A}_2$ might not "arithmetically test" the same initial tree part, it can nondeterministically freeze parts of its counters on different paths. *Relabeling* is trivial. ◀

Correspondence of PMTA and ωMSO⋈BAPA

We now provide a logical characterization of PMTAs, by showing that a tree language is recognized by a PMTA precisely if it is the set of tree models of some ωMSO⋈BAPA sentence. The "only if" part is established by Proposition 15 and the "if" part by Proposition 17.

▶ **Proposition 15.** *For any PMTA $\mathcal{A}$, there is an* ωMSO⋈BAPA *sentence φ with $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\varphi)$.*

Proof. Given a PMTA $\mathcal{A} = (Q, \Xi, q_I, \Delta, \mathcal{F}, C)$, we adopt (and slightly simplify) the idea from [41, Thm. 10] of how to encode counter values and the semilinear set C, and combine it with the usual construction to define the behavior of an MTA by means of an MSO formula: The existence of a run is defined by a sequence of existential set quantifiers representing the states of $\mathcal{A}$; one additional universal set quantifier ranging over paths is used to encode the Muller acceptance condition. Furthermore, we (outermost) existentially quantify over "counter contributions" using set quantifiers $Z_1^0, \dots, Z_1^K, \dots, Z_s^0, \dots, Z_s^K$ (with s being the number of counters and K the greatest counter increment occurring in $\mathcal{A}$'s transitions) – the presence of a variable $Z_i^{d_i}$ at a position indicates that d_i has to be added to the ith counter to simulate the extended Parikh map. Then we enforce satisfaction of the final constraint C by adding the conjunct φ_C defined as follows: By definition of C, there are $k, l \in \mathbb{N} \setminus \{0\}$ and linear polynomials $p_1, \dots, p_k\colon \mathbb{N}^l \to \mathbb{N}^s$ such that C is the union of the images of $p_1, \dots, p_k$.

Assume $p_g(m_1, \dots, m_l) = \vec{v}_0 + m_1\vec{v}_1 + \dots + m_l\vec{v}_l$ with $\vec{v}_j = (v_{j,1}, \dots, v_{j,s})$. Then, using number variables $m_1, \dots, m_l$, we encode p_g by

$$\varphi_{p_g} \coloneqq \exists m_1 \dots \exists m_l. \bigwedge_{i=1}^{s} \Big(\sum_{d=0}^{K} \mathtt{d}\#Z_i^d \mathrel{=_{\mathtt{fin}}} \mathtt{v}_{0,i} + \mathtt{v}_{1,i} m_1 + \dots + \mathtt{v}_{l,i} m_l \Big),$$

and let $\varphi_C \coloneqq \big(\bigwedge_{i=1}^{s} \bigwedge_{d=0}^{K} \forall x. \neg Z_i^d(x) \big) \vee \varphi_{p_1} \vee \dots \vee \varphi_{p_k}$. This finishes the construction of the overall sentence specifying $\mathcal{L}(\mathcal{A})$, which can be easily shown to be in ωMSO⋈BAPA. ◀

The other direction is proved by an induction on the structure of TNF formulae involving the closure properties of PMTA. The last piece that needs to be shown for this is the recognizability of the models of a Parikh constraint.

▶ **Lemma 16.** *For each Parikh constraint χ there is a PMTA $\mathcal{A}$ with $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\chi)$.*

Proof. We assume w.l.o.g. that χ is of the form $\mathtt{c} + \sum_{i\in[r]} \mathtt{c_i}\,\#X_i \mathrel{\leq_{\mathtt{fin}}} \mathtt{d} + \sum_{j\in[k]} \mathtt{d_j}\,\#Y_j$ where all X_i are pairwise distinct, and all Y_j likewise. Given a subset $\theta \subseteq \mathrm{free}(\chi)$, we denote by $|\theta|_X$ the number $\sum_{X_i\in\theta} c_i$ (and similar for $|\theta|_Y$). Then, assuming $\xi(\varrho) = (\sigma^\xi_\varrho, \theta^\xi_\varrho)$, we get

$$\mathcal{L}(\chi) = \{\xi \in T^\omega_{\Sigma_{\mathrm{free}(\chi)}} \mid c + \textstyle\sum_{\varrho\in\mathrm{pos}(\xi)} |\theta^\xi_\varrho|_X \leq d + \sum_{\varrho\in\mathrm{pos}(\xi)} |\theta^\xi_\varrho|_Y < \infty\}$$

and, by the condition $< \infty$, both sums can add up only finitely many non-zero elements. Therefore, $\xi \in \mathcal{L}(\chi)$ holds exactly if there is a non-empty, finite, prefix-closed $Z \subset \{0,1\}^*$ that comprises all positions holding variable assignments and for which $\xi|_Z$ satisfies χ. This condition can be verified by a PMTA defined in the following.

Let $D = \{(i,j) \mid 0 \leq i \leq \sum_{l\in[r]} c_l, 0 \leq j \leq \sum_{l\in[k]} d_l\}$. We construct the PMTA $\mathcal{A} = (\{q_I, q_f\}, \Xi, q_I, \Delta, \{\{q_f\}\}, C)$ with $\Xi = (\Sigma_{\mathrm{free}(\chi)} \times D) \cup \Sigma_{\mathrm{free}(\chi)}$, $\Delta = \Delta_P \cup \Delta_\omega$ where

- $\Delta_P = \{(q_I, \big((\sigma,\theta), (|\theta|_X, |\theta|_Y)\big), q', q') \mid (\sigma,\theta) \in \Sigma_{\mathrm{free}(\chi)}, q' \in \{q_I, q_f\}\}$ and
- $\Delta_\omega = \{(q_f, (\sigma, \emptyset), q_f, q_f) \mid \sigma \in \Sigma\}$

and $C = \{(z_1, z_2) \mid \mathtt{c} + z_1 \mathrel{\leq_{\mathtt{fin}}} \mathtt{d} + z_2\}$.[5] Then, one can easily show that $\mathcal{L}(\chi) = \mathcal{L}(\mathcal{A})$. ◀

▶ **Proposition 17.** *For every ω*MSO⋈BAPA *formula φ there is a PMTA $\mathcal{A}$ with $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\varphi)$.*

Proof. Let φ be an ωMSO⋈BAPA formula. By Theorem 10, we can assume that φ is in tree normal form, i.e., of the form $\exists X_1. \cdots \exists X_n. \bigvee_{i=1}^{k} \big(\varphi_i \wedge \bigwedge_{j=1}^{l_i} \chi_{i,j}\big)$, where φ_i are plain MSO sentences and the $\chi_{i,j}$ are (unnegated) Parikh constraints. The proof of the statement is an induction on the (now restricted) structure of φ using the well-known recognizability of MSO sentences [50], Lemma 16, and Proposition 14. ◀

The characterization obtained through Proposition 15 and Proposition 17 also provides an answer to the open problem posed by the authors in [35, 34] to find a logical characterization for their *reachability-regular Parikh automata* (RRPA) on words: in the usual way, our tree automata can simulate word automata (by embedding words in particular trees) and it is not too hard to see that the word version of PMTA is expressively equivalent to RRPA (details can be found in the appendix). Finally, by a routine inspection of the corresponding proofs we easily observe that our logical characterization also applies to the word setting.

[5] Note that by Lemma 11 we can use this description for a semilinear set.

Deciding Emptiness of Parikh-Muller Tree Automata

Our proof of decidability (and complexity) of the emptiness problem of PMTA rests on the respective results for the two components it combines, MTA and PTA. Thus, let us first recall the definition of Parikh tree automata [40, 39], slightly adjusted to our setting.

▶ **Definition 18** (Parikh tree automaton [41]). *Let Σ be an alphabet, let $s \geq 1$, and let $D \subseteq \mathbb{N}^s$ be finite. A* Parikh tree automaton (PTA) *is a tuple $\mathcal{A} = (Q, \Sigma \times D, \delta, q_I, F, C)$ where Q is a finite set of* states*, $\delta \subseteq Q \times (\Sigma \times D) \times Q \times Q$ is the* transition relation*, q_I is the* initial state*, $F \subseteq Q$ is a set of* final states*, and $C \subseteq \mathbb{N}^s$ is a semilinear set.*[6] *Given a finite tree $\xi \in T_{\Sigma \times D}$, a* run *of $\mathcal{A}$ on ξ is a tree $\kappa_\xi \in T_Q$ with $\mathrm{pos}(\kappa_\xi) = \{\varepsilon\} \cup \{ui \mid u \in \mathrm{pos}(\xi), i \in \{0,1\}\}$ and $\kappa(\varepsilon) = q_I$ that respects the transition relation of $\mathcal{A}$. The run κ_ξ is said to be* accepting *if $\Psi(\xi) \in C$ and $\kappa_\xi(u) \in F$ for each leaf $u \in \mathrm{pos}(\kappa_\xi) \setminus \mathrm{pos}(\xi)$; we denote the set of all accepting runs of $\mathcal{A}$ on ξ by $\mathrm{Run}_\mathcal{A}(\xi)$. Finally, the* tree language of $\mathcal{A}$*, denoted $\mathcal{L}(\mathcal{A})$, is the set*

$$\mathcal{L}(\mathcal{A}) \coloneqq \{\xi \in T_\Sigma \mid \exists \xi' \in T_{\Sigma \times D} \text{ with } \mathrm{Run}_\mathcal{A}(\xi') \neq \emptyset \text{ and } (\xi')_\Sigma = \xi\}\,.$$

It was shown in [39] that non-emptiness is decidable for PTA. The exact complexity can be obtained by adopting [28, Proposition III.2.] to the tree setting. This ultimately enables us to establish the desired result for our automaton model.

▶ **Proposition 19** (based on [39, 28]). *Given a PTA $\mathcal{A}$, deciding $\mathcal{L}(\mathcal{A}) \neq \emptyset$ is NP-complete.*

▶ **Theorem 20.** *Given a PMTA $\mathcal{A}$, deciding $\mathcal{L}(\mathcal{A}) \neq \emptyset$ is PSpace-complete.*

Proof (sketch). Let $\mathcal{A} = (Q, \Xi, q_I, \Delta, \mathcal{F}, C)$ be a PMTA with $Q = Q_P \cup Q_\omega \cup \{q_I\}$, $\Xi = (\Sigma \times D) \cup \Sigma$, and $\Delta = \Delta_P \cup \Delta_\omega$. We observe that each tree in the language of $\mathcal{A}$ can be seen as some finite tree over $\Sigma \times D$ (on which the Parikh constraint is tested), having infinite trees from T_Σ attached to all its leafs. This allows us to reduce PMTA non-emptiness testing to deciding non-emptiness of Muller tree automata and Parikh tree automata. To this end, consider

- the Muller tree automaton $\mathcal{A}_{q_I} = (Q_\omega \cup \{q_I\}, \Sigma, q_I, \Delta_\omega, \mathcal{F})$,
- the Muller tree automata $\mathcal{A}_q = (Q_\omega, \Sigma, q, \Delta_\omega, \mathcal{F})$ for all $q \in Q_\omega$, and
- the Parikh tree automaton $\mathcal{A}_P = (Q, \Sigma \times D, q_I, \Delta_P, F_P, C)$ with $F_P = \{q \in Q_\omega \mid \mathcal{L}(\mathcal{A}_q) \neq \emptyset\}$.

As deciding $\mathcal{L}(\mathcal{A}_q) \neq \emptyset$ is PSpace-complete [50, 38], $\mathcal{A}_P$ can be constructed in PSpace and, by Proposition 19, its non-emptiness can be decided in NP. Thus, the overall PSpace complexity follows from the observation that $\mathcal{L}(\mathcal{A}) \neq \emptyset$ iff $\mathcal{L}(\mathcal{A}_{q_I}) \neq \emptyset$ or $\mathcal{L}(\mathcal{A}_P) \neq \emptyset$. ◀

▶ **Corollary 21.** *Satisfiability of ωMSO$\bowtie$BAPA over labeled infinite binary trees is decidable.*

7 Decidability over Tree-Interpretable Classes of Structures

Finally, we lift the obtained decidability result for labeled trees to much more general classes of structures, leveraging the well-known technique of *MSO-interpretations* (also referred to as MSO-transductions or MSO-definable functions in the literature [1, 19, 25, 20, 22]).

[6] We note that the PTAs defined in [41] were total, i.e., δ is a function of type $Q \times (\Sigma \times D) \to \mathcal{P}(Q \times Q)$. Each PTA as defined here can be made total by using an additional sink state.

▶ **Definition 22** (MSO-Interpretation). *Given two signatures* $\mathbb{S}$ *and* $\mathbb{S}'$*, an* MSO-interpretation *is a sequence* $\mathcal{I} = (\varphi_{\mathrm{Dom}}(x), (\varphi_{\mathsf{c}}(x))_{\mathsf{c}\in\mathbb{S}_{\mathbf{C}}}, (\varphi_{\mathsf{Q}}(x_1,\ldots,x_{ar(\mathsf{Q})}))_{\mathsf{Q}\in\mathbb{S}_{\mathbf{P}}})$ *of MSO-formulae over* $\mathbb{S}'$ *(with free variables as indicated). We identify* $\mathcal{I}$ *with the partial function satisfying* $\mathcal{I}(\mathfrak{A}) = \mathfrak{B}$ *for an* $\mathbb{S}'$*-structure* $\mathfrak{A}$ *and an* $\mathbb{S}$*-structure* $\mathfrak{B}$ *if* $\{a \in A \mid \mathfrak{A}, \{x \mapsto a\} \models \varphi_{\mathrm{Dom}}(x)\} = B$ *as well as* $\{a \in B \mid \mathfrak{A}, \{x \mapsto a\} \models \varphi_{\mathsf{c}}(x)\} = \{\mathsf{c}^{\mathfrak{B}}\}$ *for every* $\mathsf{c} \in \mathbb{S}_{\mathbf{C}}$*, and, for every* $\mathsf{Q} \in \mathbb{S}_{\mathbf{P}}$*, we have* $\mathsf{Q}^{\mathfrak{B}} = \{(a_1,\ldots,a_{ar(\mathsf{Q})}) \in B^{ar(\mathsf{Q})} \mid \mathfrak{A}, \{x_i \mapsto a_i\}_{1\le i\le ar(\mathsf{Q})} \models \varphi_{\mathsf{Q}}(x_1,\ldots,x_{ar(\mathsf{Q})})\}$*. For a class* $\mathcal{S}$ *of* $\mathbb{S}'$*-structures, let* $\mathcal{I}(\mathcal{S}) \coloneqq \{\mathfrak{B} \mid \mathcal{I}(\mathfrak{A}) = \mathfrak{B}, \mathfrak{A} \in \mathcal{S}\}$*. A class* $\mathcal{T}$ *of* $\mathbb{S}$*-structures is* tree-interpretable*, if it coincides with* $\mathcal{I}(T^{\omega}_{\Sigma})$ *for some* Σ *and corresponding MSO-interpretation* $\mathcal{I}$*.*

The key insight for our result is that the well-known rewritability of MSO formulae under MSO-interpretations can be lifted to ωMSO⋈BAPA without much effort.

▶ **Lemma 23.** *Let* $\mathcal{I}$ *be an MSO-interpretation. Then, for every* ωMSO⋈BAPA *sentence* φ *over* $\mathbb{S}$ *one can compute an* ωMSO⋈BAPA *sentence* $\varphi^{\mathcal{I}}$ *over* $\mathbb{S}'$ *satisfying* $\mathfrak{A} \models \varphi^{\mathcal{I}} \iff \mathfrak{B} \models \varphi$ *for every* $\mathbb{S}'$*-structure* $\mathfrak{A}$ *and* $\mathbb{S}$*-structure* $\mathfrak{B}$ *with* $\mathcal{I}(\mathfrak{A}) \cong \mathfrak{B}$*.*

This insight can be used to show that decidability is propagated through MSO-interpretations, and thus can be guaranteed for all tree-interpretable classes, thanks to Corollary 21.

▶ **Theorem 24.** *Let* $\mathcal{S}$ *be a class of structures over which satisfiability of* ωMSO⋈BAPA *is decidable, let* $\mathcal{I}$ *be an MSO-interpretation. Then satisfiability of* ωMSO⋈BAPA *over* $\mathcal{I}(\mathcal{S})$ *is decidable as well. In particular,* ωMSO⋈BAPA *is decidable over any tree-interpretable class.*

This result allows us, in one go, to harvest several decidability results, as tree-interpretability is able to capture classes of (finite or countable) structures whose treewidth [51], cliquewidth [27, 22, 21, 36], or partitionwidth [10, 11, 26] is bounded by some value $k \in \mathbb{N}$.

▶ **Corollary 25.** *Given a signature* $\mathbb{S}$*, satisfiability of* ωMSO⋈BAPA *is decidable over the classes of finite or countable* $\mathbb{S}$*-structures of bounded treewidth, cliquewidth, and partitionwidth.*

8 Incorporating Two-Variable-Logics without Width Restrictions

Corollary 25 constitutes a strong decidability result, also in view of the fact that lifting the width restriction immediately leads to undecidability even for much weaker logics like FO. A feasible way to nevertheless relax this restriction without putting decidability at risk and yet maintaining all the expressive power of ωMSO⋈BAPA is to "couple" it with another logic $\mathbb{L}$ whose satisfiability problem is decidable over arbitrary structures. Then, one considers sentences $\varphi \wedge \psi$, where φ is an ωMSO⋈BAPA sentence while ψ is an $\mathbb{L}$-sentence, and asks for models whose reduct to the signature of φ adheres to the width restriction. That way, signature elements of ψ not occurring in φ can "behave freely" and are not subject to the imposed width constraint.[7] Such a "coupling" of ωMSO⋈BAPA and $\mathbb{L}$ can be made more or less "tight" depending on the arity of the predicates allowed to be shared between φ and ψ.

We can show that a decidable coupling with shared unary predicates can be done for $\mathbb{L} = \mathrm{FO}^2_{\mathrm{Pres}}$ [7], an expressive extension of 2-variable first-order logic by Presburger-like counting quantifiers of the form $\exists^S$, where $S \subseteq \mathbb{N} \cup \{\infty\}$ is an ultimately periodic set from $\mathbb{N} \cup \{\infty\}$ with the semantics defined by $\mathfrak{A}, \nu \models \exists^S x.\varphi$ iff $|\{a \in A \mid \mathfrak{A}, \nu_{x\mapsto a} \models \varphi\}| \in S$. $\mathrm{FO}^2_{\mathrm{Pres}}$ subsumes the prominent counting 2-variable first-order fragment C^2 [32], but goes beyond first-order logic. Its satisfiability problem was shown to be decidable only recently [7].

[7] We refer to Kotek et al. [42] for a result that is similar in spirit, establishing decidability of finite satisfiability of treewidth-bounded MSO_2 coupled with C^2.

▶ **Theorem 26.** *Let w be any of treewidth, cliquewidth, or partitionwidth, and let $n \in \mathbb{N}$. Let $\mathbb{S}_a$ and $\mathbb{S}_b$ be signatures whose only joint elements are unary predicates. Then the following problem is decidable: Given a ωMSO$\bowtie$BAPA sentence φ over $\mathbb{S}_a$ and a $\mathrm{FO}^2_{\mathrm{Pres}}$ sentence ψ over $\mathbb{S}_b$, does there exist a countable $\mathbb{S}_a \cup \mathbb{S}_b$-structure $\mathfrak{C}$ satisfying $w(\mathfrak{C}|_{\mathbb{S}_a}) \leq n$ and $\mathfrak{C} \models \varphi \wedge \psi$.*

In a nutshell, this result is obtained by exploiting the fact that, for every $\mathrm{FO}^2_{\mathrm{Pres}}$ formula ψ over $\mathbb{S}_b$, one can construct a ωMSO$\bowtie$BAPA formula ψ' over the purely unary signature $\mathbb{S}_a \cap \mathbb{S}_b$ that is satisfied by precisely those $\mathbb{S}_a$-structures that are "$\mathbb{S}_a \cap \mathbb{S}_b$-compatible" with some model of ψ. Consequently, the ωMSO$\bowtie$BAPA formula $\varphi \wedge \psi'$ over $\mathbb{S}_a$ is such that for any of its models $\mathfrak{A}$ one finds a "$\mathbb{S}_a \cap \mathbb{S}_b$-compatible" model $\mathfrak{B}$ of ψ. Then, superimposing $\mathfrak{A}$ and $\mathfrak{B}$ would yield a model $\mathfrak{C}$ of $\varphi \wedge \psi$, which by construction satisfies $w(\mathfrak{C}|_{\mathbb{S}_a}) = w(\mathfrak{A})$. Consequently, to solve the decision problem of Theorem 26, it suffices to check if the ωMSO$\bowtie$BAPA formula $\varphi \wedge \psi'$ has a model $\mathfrak{A}$ satisfying $w(\mathfrak{A}|_{\mathbb{S}_a}) \leq n$ which is decidable by Corollary 25. We note that the extended arithmetic capabilities of ωMSO$\bowtie$BAPA are essential for this result, as ψ' needs to encode linear inequalities over counts of realized atomic 1-types.

9 Showcase: Decidability of Tame Satisfiability of the Fully Enriched μ-Calculus with Global Presburger Counting

An important and practically relevant class of expressive logical formalisms, which play a pivotal role in logic-based knowledge representation and verification, is obtained from variations and extensions of propositional modal logics [8, 9] and description logics [4, 53]. This class contains most ontology languages as well as PDL [29], CTL* [24], the propositional modal μ-calculus [43] and their extensions. Modulo some representational variations, all these logics' model-theoretic semantics rest on structures over unary and binary predicates (often interpreted as a transition system's state space). While the simpler variants of this family can be seen as fragments of first-order logic, the more expressive ones cannot, as they feature fixed-point capabilities (through regular path expressions or explicit fixed-point operators). Typically, decidability of the satisfiability problem in these logics follows from some sort of tree-model property. Many of these logics exhibit some limited local counting capabilities [54], but recently, there has been an increased interest in accommodating more advanced arithmetic constraints [23, 47, 2, 5], including *global constraints* [3, 52] expressing statistical information such as "more than 50% of the state space's final states are successful".

We will demonstrate the usefulness of ωMSO$\bowtie$BAPA for establishing decidability results at the example of adding global Presburger constraints to the *fully enriched μ-calculus*, a very powerful formalism used in verification. We first introduce syntax and semantics.[8]

▶ **Definition 27.** *Given a signature $\mathbb{S} = \mathbb{S}_{\mathbf{C}} \cup \mathbb{S}_{\mathbf{P},1} \cup \mathbb{S}_{\mathbf{P},2}$ of constants, unary predicates and binary predicates, the formulas of the fully enriched μ-calculus (FEμ) are defined by*

$$\varphi ::= \mathbf{true} \mid \mathbf{false} \mid X \mid \mathtt{c} \mid \neg\mathtt{c} \mid \mathtt{P} \mid \neg\mathtt{P} \mid \varphi \wedge \varphi' \mid \varphi \vee \varphi' \mid \langle n, \alpha\rangle\varphi \mid [n, \alpha]\varphi \mid \mu X.\varphi \mid \nu X.\varphi$$

where X is a set variable from some countable set $\mathbf{V}_{\mathrm{set}}$, $\mathtt{P} \in \mathbb{S}_{\mathbf{P},1}$, $n \in \mathbb{N}$ and α has the form $\mathtt{R}$ or $\mathtt{R}^-$ for some $\mathtt{R} \in \mathbb{S}_{\mathbf{P},2}$. For ease of presentation, we assume positive normal form.

[8] For brevity and coherence, we slightly adjust the syntax and use classical model-theoretic semantics (structures with unary and binary predicates) instead of the original one of modal logic (Kripke structures with propositional variables and programs), as the two are well known to be equivalent.

Given a structure $\mathfrak{A}$ *and a set variable assignment* $\nu : \mathbf{V}_{\text{set}} \to 2^A$*, the semantics* $[\![\varphi]\!]^{\mathfrak{A}}_{\nu} \subseteq A$ *of formulae* φ *is defined by the following function (stipulating* $(\mathtt{R}^-)^{\mathfrak{A}} = \{(a, a') \mid (a', a) \in \mathtt{R}^{\mathfrak{A}}\}$*):*

$$\begin{array}{llll} \mathbf{true} \mapsto A & X \mapsto \nu(X) & \mathtt{c} \mapsto \{\mathtt{c}^{\mathfrak{A}}\} & \mathtt{P} \mapsto \mathtt{P}^{\mathfrak{A}} & \varphi \wedge \varphi' \mapsto [\![\varphi]\!]^{\mathfrak{A}}_{\nu} \cap [\![\varphi']\!]^{\mathfrak{A}}_{\nu} \\ \mathbf{false} \mapsto \emptyset & & \neg\mathtt{c} \mapsto A \setminus \{\mathtt{c}^{\mathfrak{A}}\} & \neg\mathtt{P} \mapsto A \setminus \mathtt{P}^{\mathfrak{A}} & \varphi \vee \varphi' \mapsto [\![\varphi]\!]^{\mathfrak{A}}_{\nu} \cup [\![\varphi']\!]^{\mathfrak{A}}_{\nu} \\ \langle n, \alpha\rangle\varphi \mapsto \{a \mid |\{\alpha^{\mathfrak{A}} \cap (\{a\}\times[\![\varphi]\!]^{\mathfrak{A}}_{\nu})\}| \geq n\} & & & \mu X.\varphi \mapsto \bigcap\{A' \subseteq A \mid [\![\varphi]\!]^{\mathfrak{A}}_{\nu_{X\mapsto A'}} \subseteq A'\} & \\ [n, \alpha]\varphi \mapsto \{a \mid |\{\alpha^{\mathfrak{A}} \cap (\{a\}\times(A \setminus [\![\varphi]\!]^{\mathfrak{A}}_{\nu}))\}| \leq n\} & & & \nu X.\varphi \mapsto \bigcup\{A' \subseteq A \mid A' \subseteq [\![\varphi]\!]^{\mathfrak{A}}_{\nu_{X\mapsto A'}}\} & \end{array}$$

A FEμ formula is closed *if all occurrences of set variables are in the scope of some* μ *or* ν*. A* global FEμ Presburger constraint *is a Parikh constraint (cf. Definition 7), where all set variables have been replaced by closed FEμ formulae. Given a set* Π *of global FEμ Presburger constraints, we let* $\mathfrak{A} \models \Pi$ *if for every element of* Π*, replacing each of its closed FEμ formulae* ψ *by* $[\![\psi]\!]^{\mathfrak{A}}_{\emptyset}$ *produces a statement valid in* $\mathfrak{A}$*. A closed FEμ formula* φ *is satisfiable wrt.* Π *if there is some structure* $\mathfrak{A} \models \Pi$ *with* $[\![\varphi]\!]^{\mathfrak{A}}_{\emptyset} \neq \emptyset$*, in which case we call* $\mathfrak{A}$ *a model of* (φ, Π)*.*

In fact, unrestricted satisfiability in FEμ (even without Presburger constraints) is undecidable [13]. Decidability can be regained, however, when restricting to *tame structures*, also commonly known as "quasi-forests" [15, 12, 16, 6].

▶ **Definition 28** (tame structures). *Let* $\mathbb{S} = \mathbb{S}_{\mathbf{C}} \cup \mathbb{S}_{\mathbf{P},1} \cup \mathbb{S}_{\mathbf{P},2}$ *be a signature as above. A* tame structure $\mathfrak{A}$ *over* $\mathbb{S}$ *is a countable structure such that, for some finite set Roots,*

- *the domain* A *of* $\mathfrak{A}$ *is a forest, i.e., a prefix-closed subset of* $\{rw \mid r \in Roots, w \in \mathbb{N}^*\}$*,*
- *the roots coincide with the named elements, i.e.,* $Roots = \{\mathtt{a}^{\mathfrak{A}} \mid \mathtt{a} \in \mathbb{S}_{\mathbf{C}}\}$*, and*
- *for every* $a, a' \in A$ *with* $(a, a') \in \mathtt{R}^{\mathfrak{A}}$ *for some* $\mathtt{R} \in \mathbb{S}_{\mathbf{P},2}$*, either (i)* $\{a, a'\} \cap Roots \neq \emptyset$*, or (ii)* $a = a'$*, or (iii)* a *is a child of* a'*, or (iv)* a' *is a child of* a*.*

A logic has the tame model property *if every satisfiable formula* φ *has a model that is tame over the signature used by* φ*. The* tame satisfiability *problem consists in deciding if a given formula has a tame model.*

While the restriction to tame structures may seem somewhat arbitrary at first, it is well justified: three maximal decidable sublogics of FEμ have the tame-model-property [12], in which case satisfiability over arbitrary structures and tame structures coincide. Also, the structural restriction has some plausibility from a transition system perspective in that one distinguishes between a finite set of "named" states with arbitrary transitions between them and potentially infinitely many "anonymous" states with more restricted access. It is easy to see that all tame structures over $\mathbb{S} = \mathbb{S}_{\mathbf{C}} \cup \mathbb{S}_{\mathbf{P}}$ have a treewidth not larger than $|\mathbb{S}_{\mathbf{C}}| + 1$.

▶ **Theorem 29.** *The tame satisfiability problem of the fully enriched* μ*-calculus with global Presburger constraints is decidable.*

Proof (sketch). Let $\mathbb{S}$ be a finite signature, φ a closed FEμ formula over $\mathbb{S}$, and Π a finite set of global FEμ Presburger constraints. Being a tame structure over $\mathbb{S}$ can be expressed by an MSO sentence ψ_{tame}. We define a translation trans_x mapping closed FEμ formulae to ωMSO⋈BAPA formulae with free variable x such that $\mathfrak{A}, \{x \mapsto a\} \models \text{trans}_x(\varphi)$ iff $a \in [\![\varphi]\!]^{\mathfrak{A}}_{\emptyset}$. Based on this, we exhibit another translation trans, which maps global FEμ Presburger constraints to equivalent ωMSO⋈BAPA sentences. Then, tame satisfiability of (φ, Π) corresponds to satisfiability of the ωMSO⋈BAPA sentence $\psi_{\text{tame}} \wedge \exists x.\text{trans}_x(\varphi) \wedge \bigwedge \text{trans}(\Pi)$ over all countable structures of treewidth $\leq |\mathbb{S}_{\mathbf{C}}| + 1$, which is decidable by Corollary 25. ◀

Thanks to the expressive power of FEμ, the above result transfers to numerous other prominent logics (and their fragments), including PDL and CTL* as well as the description logics $\mu\mathcal{ALCOIQ}$ and $\mathcal{ALCOIQ}^{\text{reg}}$ [14], for all of which tame satisfiability is thus decidable even in the presence of global Presburger constraints. The argument easily extends to the description logic $\mathcal{ZOIQ}$ [16], adding Boolean combinations of binary predicates (programs).

10 Conclusion

We have proposed ωMSO⋈BAPA, a logic with a high combined structural and arithmetic expressivity, subsuming and properly extending existing popular formalisms for either purpose. We have established decidability of the satisfiability of ωMSO⋈BAPA formulae over arbitrary tree-interpretable classes of structures. A key role is played by Parikh-Muller Tree Automata, a novel type of automaton over labeled infinite binary trees with decidable emptiness.

For improving readability and succinctness, the syntax of our formalism could be extended by "comprehension expressions": set terms of the form $\{x \mid \psi\}$ with $x \in \mathbf{V}_{\text{ind}}$ and $\psi \in \mathbf{F}$, whose semantics is straightforwardly defined by $\{x \mid \psi\}^{\mathfrak{A},\nu} = \{a \in A \mid \mathfrak{A}, \nu_{x \mapsto a} \models \psi\}$. E.g., this allows us to write $2 \,\#\{x \mid \exists y.\mathtt{R}(x,y)\} = 3 \,\#\{y \mid \exists x.\mathtt{R}(x,y)\}$ rather than the more unwieldy

$$\exists V_1.(\forall x.V_1(x) \Leftrightarrow \exists y.\mathtt{R}(x,y)) \wedge \exists V_2.(\forall y.V_2(y) \Leftrightarrow \exists x.\mathtt{R}(x,y)) \wedge 2 \,\#V_1 = 3 \,\#V_2.$$

Note that comprehension expressions do not increase expressivity; they can be removed from a formula φ yielding an equivalent formula φ' as follows: Let χ be the largest subformula of φ that contains the expression $\{x \mid \psi\}$ but no quantifiers binding any of the free variables of ψ. Then, obtain φ' from φ by replacing χ by χ', where $\chi' := \exists Z.(\forall x.Z(x) \Leftrightarrow \psi) \wedge \chi[\{x \mid \psi\} \mapsto Z]$. ωMSO⋈BAPA membership of such extended formulae can then be decided based on their "purified" variant,[9] or by means of an elaborately refined analysis of variable interactions.

Concluding, we are quite confident that this paper's findings and techniques will prove useful as a generic tool for establishing decidability results for formalisms from various areas of computer science such as knowledge representation or verification. That said, in view of the non-elementary blow-ups abounding in our methods, we concede that they are unlikely to be helpful in more fine-grained complexity analyses, once decidability is established.

References

1 Stefan Arnborg, Jens Lagergren, and Detlef Seese. Problems easy for tree-decomposable graphs (extended abstract). In Timo Lepistö and Arto Salomaa, editors, *15th International Colloquium on Automata, Languages and Programming (ICALP 1988)*, volume 317 of *LNCS*, pages 38–51. Springer, 1988. doi:10.1007/3-540-19488-6_105.

2 Franz Baader, Bartosz Bednarczyk, and Sebastian Rudolph. Satisfiability and query answering in description logics with global and local cardinality constraints. In Giuseppe De Giacomo, Alejandro Catalá, Bistra Dilkina, Michela Milano, Senén Barro, Alberto Bugarín, and Jérôme Lang, editors, *24th European Conference on Artificial Intelligence, (ECAI 2020) – Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020)*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pages 616–623. IOS Press, 2020. doi:10.3233/FAIA200146.

3 Franz Baader, Martin Buchheit, and Bernhard Hollunder. Cardinality restrictions on concepts. *Artif. Intell.*, 88(1-2):195–213, 1996. doi:10.1016/S0004-3702(96)00010-0.

4 Franz Baader, Ian Horrocks, Carsten Lutz, and Uli Sattler. *An Introduction to Description Logic*. Cambridge University Press, 2017. doi:10.1017/9781139025355.

5 Bartosz Bednarczyk, Maja Orlowska, Anna Pacanowska, and Tony Tan. On classical decidable logics extended with percentage quantifiers and arithmetics. In Mikolaj Bojanczyk and Chandra Chekuri, editors, *41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2021)*, volume 213 of *LIPIcs*, pages 36:1–36:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.FSTTCS.2021.36.

[9] The described removal technique is optimized toward producing formulae in ωMSO⋈BAPA.

6 Bartosz Bednarczyk and Sebastian Rudolph. Worst-case optimal querying of very expressive description logics with path expressions and succinct counting. In Sarit Kraus, editor, *28st International Joint Conference on Artificial Intelligence (IJCAI 2019)*, pages 1530–1536. ijcai.org, 2019. doi:10.24963/ijcai.2019/212.

7 Michael Benedikt, Egor V. Kostylev, and Tony Tan. Two variable logic with ultimately periodic counting. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*, volume 168 of *LIPIcs*, pages 112:1–112:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.112.

8 Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2001. doi:10.1017/CBO9781107050884.

9 Patrick Blackburn, Johan F.A.K. van Benthem, and Frank Wolter, editors. *Handbook of Modal Logic*, volume 3 of *Studies in logic and practical reasoning*. North-Holland, 2007.

10 Achim Blumensath. *Structures of bounded partition width*. PhD thesis, RWTH Aachen University, Germany, 2003. URL: http://sylvester.bth.rwth-aachen.de/dissertationen/2003/256/index.htm.

11 Achim Blumensath. A model-theoretic characterisation of clique width. *Annals of Pure and Applied Logic*, 142(1-3):321–350, 2006. doi:10.1016/j.apal.2006.02.004.

12 Piero A. Bonatti, Carsten Lutz, Aniello Murano, and Moshe Y. Vardi. The complexity of enriched mu-calculi. *Log. Methods Comput. Sci.*, 4(3), 2008. doi:10.2168/LMCS-4(3:11)2008.

13 Piero A. Bonatti and Adriano Peron. On the undecidability of logics with converse, nominals, recursion and counting. *Artif. Intell.*, 158(1):75–96, 2004. doi:10.1016/j.artint.2004.04.012.

14 Diego Calvanese and Giuseppe De Giacomo. *Expressive Description Logics*, pages 193–236. Cambridge University Press, 2 edition, 2007. doi:10.1017/CBO9780511711787.007.

15 Diego Calvanese, Thomas Eiter, and Magdalena Ortiz. Answering regular path queries in expressive description logics: An automata-theoretic approach. In *22nd Conference on Artificial Intelligence (AAAI 2007)*, pages 391–396. AAAI Press, 2007. URL: http://www.aaai.org/Library/AAAI/2007/aaai07-061.php.

16 Diego Calvanese, Thomas Eiter, and Magdalena Ortiz. Regular path queries in expressive description logics with nominals. In Craig Boutilier, editor, *21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, pages 714–720, 2009. URL: http://ijcai.org/Proceedings/09/Papers/124.pdf.

17 Bruno Courcelle. The monadic second-order logic of graphs, II: Infinite graphs of bounded width. *Mathematical Systems Theory*, 21(1):187–221, 1988. doi:10.1007/BF02088013.

18 Bruno Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75, 1990. doi:10.1016/0890-5401(90)90043-H.

19 Bruno Courcelle. The monadic second-order logic of graphs V. On closing the gap between definability and recognizability. *Theoretical Computer Science*, 80(2):153–202, 1991. doi:10.1016/0304-3975(91)90387-H.

20 Bruno Courcelle. Monadic second-order definable graph transductions: A survey. *Theoretical Computer Science*, 126(1):53–75, 1994. doi:10.1016/0304-3975(94)90268-2.

21 Bruno Courcelle. Clique-width of countable graphs: A compactness property. *Discrete Mathematics*, 276(1-3):127–148, 2004. doi:10.1016/S0012-365X(03)00303-0.

22 Bruno Courcelle and Joost Engelfriet. *Graph Structure and Monadic Second-Order Logic – A Language-Theoretic Approach*, volume 138 of *Encyclopedia of mathematics and its applications*. Cambridge University Press, 2012. doi:10.1017/CBO9780511977619.

23 Stéphane Demri and Denis Lugiez. Complexity of modal logics with presburger constraints. *J. Appl. Log.*, 8(3):233–252, 2010. doi:10.1016/j.jal.2010.03.001.

24 E. Allen Emerson and Joseph Y. Halpern. "sometimes" and "not never" revisited: On branching versus linear time. In John R. Wright, Larry Landweber, Alan J. Demers, and Tim Teitelbaum, editors, *10th Annual ACM Symposium on Principles of Programming Languages (POPL 1983)*, pages 127–140. ACM Press, 1983. doi:10.1145/567067.567081.

25 Joost Engelfriet. A characterization of context-free NCE graph languages by monadic second-order logic on trees. In Hartmut Ehrig, Hans-Jörg Kreowski, and Grzegorz Rozenberg, editors, *4th International Workshop on Graph-Grammars and Their Application to Computer Science (Graph Grammars 1990)*, volume 532 of *LNCS*, pages 311–327. Springer, 1990. doi:10.1007/BFb0017397.

26 Thomas Feller, Tim S. Lyon, Piotr Ostropolski-Nalewaja, and Sebastian Rudolph. Decidability of querying first-order theories via countermodels of finite width, 2023. arXiv:2304.06348.

27 Thomas Feller, Tim S. Lyon, Piotr Ostropolski-Nalewaja, and Sebastian Rudolph. Finite-cliquewidth sets of existential rules: Toward a general criterion for decidable yet highly expressive querying. In *26th International Conference on Database Theory (ICDT 2023)*, LIPIcs. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023.

28 Diego Figueira and Leonid Libkin. Path Logics for Querying Graphs: Combining Expressiveness and Efficiency. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2015)*, pages 329–340, 2015. doi:10.1109/LICS.2015.39.

29 Michael J. Fischer and Richard E. Ladner. Propositional dynamic logic of regular programs. *J. Comput. Syst. Sci.*, 18(2):194–211, 1979. doi:10.1016/0022-0000(79)90046-1.

30 Seymour Ginsburg and Edwin H. Spanier. Bounded Algol-Like Languages. *Transactions of the American Mathematical Society*, 113(2):333, 1964. doi:10.2307/1994067.

31 Seymour Ginsburg and Edwin H. Spanier. Semigroups, Presburger formulas, and languages. *Pacific Journal of Mathematics*, 16(2):285–296, 1966. doi:10.2140/pjm.1966.16.285.

32 Erich Grädel, Martin Otto, and Eric Rosen. Two-variable logic with counting is decidable. In *12th Annual IEEE Symposium on Logic in Computer Science (LICS'97)*, pages 306–317. IEEE Computer Society, 1997. doi:10.1109/LICS.1997.614957.

33 Mario Grobler, Leif Sabellek, and Sebastian Siebertz. Parikh Automata on Infinite Words, 2023. arXiv:2301.08969.

34 Mario Grobler, Leif Sabellek, and Sebastian Siebertz. Remarks on Parikh-recognizable omega-languages, 2023. arXiv:2307.07238.

35 Mario Grobler and Sebastian Siebertz. Büchi-like characterizations for Parikh-recognizable omega-languages, 2023. arXiv:2302.04087.

36 Martin Grohe and György Turán. Learnability and definability in trees and similar structures. *Theory of Computing Systems*, 37(1):193–220, 2004. doi:10.1007/s00224-003-1112-8.

37 Shibashis Guha, Ismaël Jecker, Karoliina Lehtinen, and Martin Zimmermann. Parikh automata over infinite words. In Anuj Dawar and Venkatesan Guruswami, editors, *42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2022)*, volume 250 of *LIPIcs*, pages 40:1–40:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.FSTTCS.2022.40.

38 Paul Hunter and Anuj Dawar. Complexity Bounds for Regular Games. In Joanna Jędrzejowicz and Andrzej Szepietowski, editors, *Mathematical Foundations of Computer Science (MFCS 2005)*, LNCS, pages 495–506. Springer, 2005. doi:10.1007/11549345_43.

39 Felix Klaedtke. *Automata-based decision procedures for weak arithmetics*. PhD thesis, University of Freiburg, Freiburg im Breisgau, Germany, 2004. URL: http://freidok.ub.uni-freiburg.de/volltexte/1439/index.html.

40 Felix Klaedtke and Harald Rueß. Parikh automata and monadic second-order logics with linear cardinality constraints. Technical Report 177, Albert-Ludwigs-Universität Freiburg, 2002. (revised version).

41 Felix Klaedtke and Harald Rueß. Monadic second-order logics with cardinalities. In Jos C. M. Baeten, Jan Karel Lenstra, Joachim Parrow, and Gerhard J. Woeginger, editors, *Automata, Languages and Programming, 30th International Colloquium (ICALP 2003)*, volume 2719 of *LNCS*, pages 681–696. Springer, 2003. doi:10.1007/3-540-45061-0_54.

42 Tomer Kotek, Helmut Veith, and Florian Zuleger. Monadic Second Order Finite Satisfiability and Unbounded Tree-Width. In Jean-Marc Talbot and Laurent Regnier, editors, *25th EACSL Annual Conference on Computer Science Logic (CSL 2016)*, volume 62 of *LIPIcs*, pages 13:1–13:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.CSL.2016.13.

43 Dexter Kozen. Results on the propositional μ-calculus. In Mogens Nielsen and Erik Meineche Schmidt, editors, *Automata, Languages and Programming, 9th Colloquium (ICALP 1982)*, volume 140 of *LNCS*, pages 348–359. Springer, 1982. doi:10.1007/BFb0012782.

44 Viktor Kuncak, Huu Hai Nguyen, and Martin Rinard. An Algorithm for Deciding BAPA: Boolean Algebra with Presburger Arithmetic. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Dough Tygar, Moshe Y. Vardi, Gerhard Weikum, and Robert Nieuwenhuis, editors, *Automated Deduction (CADE 2005)*, volume 3632 of *LNCS*, pages 260–277. Springer, 2005. doi:10.1007/11532231_20.

45 Viktor Kuncak, Huu Hai Nguyen, and Martin Rinard. Deciding Boolean Algebra with Presburger Arithmetic. *Journal of Automated Reasoning*, 36(3):213–239, 2006. doi:10.1007/s10817-006-9042-1.

46 Aless Lasaruk and Thomas Sturm. Effective Quantifier Elimination for Presburger Arithmetic with Infinity. In Vladimir P. Gerdt, Ernst W. Mayr, and Evgenii V. Vorozhtsov, editors, *Computer Algebra in Scientific Computing (CASC 2009)*, volume 5743 of *LNCS*, pages 195–212. Springer, 2009. doi:10.1007/978-3-642-04103-7_18.

47 Yensen Limón, Edgard Benítez-Guerrero, Everardo Bárcenas, Guillermo Molero-Castillo, and Alejandro Velázquez-Mena. A satisfiability algorithm for the mu-calculus for trees with presburger constraints. In *7th International Conference in Software Engineering Research and Innovation (CONISOFT 2019)*, pages 72–79, 2019. doi:10.1109/CONISOFT.2019.00020.

48 Yuri V. Matiyasevich. *Hilbert's Tenth Problem*. Foundations of Computing. MIT Press, 1993.

49 Rohit J. Parikh. On Context-Free Languages. *Journal of the ACM*, 13(4):570–581, 1966. doi:10.1145/321356.321364.

50 Michael O. Rabin. Decidability of Second-Order Theories and Automata on Infinite Trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969. doi:10.2307/1995086.

51 Neil Robertson and P.D Seymour. Graph minors. III. Planar tree-width. *Journal of Combinatorial Theory, Series B*, 36(1):49–64, 1984. doi:10.1016/0095-8956(84)90013-3.

52 "Johann" Sebastian Rudolph. Presburger concept cardinality constraints in very expressive description logics – Allegro sexagenarioso ma non ritardando. In Carsten Lutz, Uli Sattler, Cesare Tinelli, Anni-Yasmin Turhan, and Frank Wolter, editors, *Description Logic, Theory Combination, and All That – Essays Dedicated to Franz Baader on the Occasion of His 60th Birthday*, volume 11560 of *LNCS*, pages 542–561. Springer, 2019. doi:10.1007/978-3-030-22102-7_25.

53 Sebastian Rudolph. Foundations of description logics. In Axel Polleres, Claudia d'Amato, Marcelo Arenas, Siegfried Handschuh, Paula Kroner, Sascha Ossowski, and Peter F. Patel-Schneider, editors, *Lecture Notes of the 7th International Reasoning Web Summer School (RW'11)*, volume 6848 of *LNCS*, pages 76–136. Springer, 2011. doi:10.1007/978-3-642-23032-5_2.

54 Stephan Tobies. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *J. Artif. Intell. Res.*, 12:199–217, 2000. doi:10.1613/jair.705.

Energy Games over Totally Ordered Groups

Alexander Kozachinskiy
IMFD Chile & CENIA Chile, Santiago, Chile

Abstract

Kopczyński (ICALP 2006) conjectured that prefix-independent half-positional winning conditions are closed under finite unions. We refute this conjecture over finite arenas. For that, we introduce a new class of prefix-independent bi-positional winning conditions called energy conditions over totally ordered groups. We give an example of two such conditions whose union is not half-positional. We also conjecture that every prefix-independent bi-positional winning condition coincides with some energy condition over a totally ordered group on periodic sequences.

2012 ACM Subject Classification Theory of computation → Logic

Keywords and phrases Games on graphs, half-positionality, ordered groups

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.34

Related Version *Previous Version*: https://doi.org/10.48550/arXiv.2205.04508

Funding *Alexander Kozachinskiy*: the author is funded by ANID – Millennium Science Initiative Program – Code ICN17002, and the National Center for Artificial Intelligence CENIA FB210017, Basal ANID.

1 Introduction

This paper is devoted to positional determinacy in turn-based infinite-duration games. An arena is a (possibly, infinite) directed graph whose edges are colored into elements of some set of colors C and whose nodes are partitioned between two players called Eve and Adam. They play by traveling over the nodes of the arena. In each turn, one of the players chooses an edge from the current node, and the players move toward the endpoint of this edge. Whether it is an Eve's or an Adam's turn to choose depends on whether the current node is an Eve's node or an Adam's node. This continues for infinitely many turns. As a result, the players obtain an infinite word over C (by concatenating the colors of edges that appear in the play). A *winning condition* W, which is a set of infinite words over C, defines the aims of the players. Eve wants to obtain an infinite word that belongs to W, while Adam wants it to be outside W.

A vast amount of literature in this area is devoted to *positional strategies*. A strategy of Eve or Adam is positional if, for every node controlled by the player in question, there exists an out-going edge which is always played by this strategy at this node. Implementing such strategies is easy because we only have to specify one edge for each node of the corresponding player. This makes these strategies relevant for such areas as controller synthesis [2], where an implementation of a controller can be seen as its strategy against an environment.

Correspondingly, of great interest are winning conditions for which positional strategies are always sufficient to play optimally (for one of the players or even for both of them). This area has the following terminology. A winning condition W is *half-positional* if in every arena Eve has a positional strategy σ such that for every node of the arena the following holds: if σ is not winning w.r.t. W if the game starts at this node, then Adam has a winning strategy w.r.t. W (not necessarily positional) from this node. To put it simply, σ must be winning everywhere where Adam does not have a winning strategy. If this condition holds in all finite arenas (but possibly does not hold in some infinite arenas), then W is called half-positional over finite arenas. A winning condition W is *bi-positional* (over all or over

finite arenas) if additionally the same requirement as for Eve holds for Adam (or, in other words, if both W and its complement are half-positional). The family of parity condition, which is of great interest due to its applications in logic [20, 14], is known to be bi-positional over finite and infinite arenas [9]. There exist winning conditions that are bi-positional over finite arenas, but not even half-positional over infinite arenas, for instance, certain variants of the mean-payoff condition [18].

Winning conditions, bi-positional over infinite arenas, are understood quite well. For instance, it is known that all of them are ω-regular [4]. In turn, parity conditions are exactly those winning conditions that are bi-positional over infinite arenas and are prefix-independent, that is, closed under adding or removing finite prefixes [7].

Bi-positionality over finite arenas was studied by Gimbert and Zielonka [11, 12]. In [11], they gave a simple sufficient condition for bi-positionality over finite arenas, suitable for the majority of the applications, and in [12], they gave a condition which is sufficient and necessary (but far more complex). Their sufficient and necessary condition has a corollary called *1-to-2-player lifting*, which is of great interest in practice. It states that as long as a winning condition is half-positional for Eve in finite arenas without Adam and half-positional for Adam in finite arenas without Eve, it is bi-positional in all finite arenas.

Recently, Ohlmann [18] obtained a sufficient and necessary condition for half-positionality over infinite arenas. As for finite arenas, several sufficient conditions for half-positionality were obtained in the literature [15, 1], but none of them is necessary.

Are there set operations under which bi-positional and half-positional winning conditions are closed? Bi-positional winning conditions are closed under complement by definition. At the same time, bi-positional winning conditions are not closed under intersection (and hence under union) [17].

Closure properties of half-positional winning conditions were first addressed by Kopczyński [15]. He conjectured that prefix-independent half-positional winning conditions are closed under union. This conjecture has many variants, depending on whether we mean half-positionality over finite or infinite arenas, and whether we consider arbitrary unions or only finite ones. Kopczyński himself refuted a variant for infinite arenas and uncountable unions. He also noticed that dropping the prefix-independence assumption or changing union to intersection immediately makes the conjecture false.

No counter-example, refuting it for finite or even countable unions, had been found. On the positive side, several classes of prefix-independent half-positional winning conditions that are closed under union were identified in the literature, including concave conditions (over finite arenas and arbitrary unions) of Kopczyński [15] and conditions "excluding healing" of Ohlmann [18] (for infinite arenas but at most countable unions).

In this paper, we refute the Kopczyński's conjecture for *finite* arenas and *finite* unions. Moreover, we present two winning conditions that are *bi-positional* over finite arenas and whose union is not half-positional over finite arenas.

Kopczyński's conjecture over infinite arenas for finite/countable unions remains open. Additionally, there has been an interest in whether some variant of the Kopczyński's conjecture holds in a restriction to ω-regular condition. Bouyer et al. [3] obtained that prefix-independent ω-regular conditions, recognizable by deterministic Büchi automata (DBA), are closed under finite union. In fact, they simply show that every prefix-independent DBA-recognizable ω-regular condition can be given as a set of sequences, having infinitely many occurrences of some fixed subset of colors. Such conditions are trivially closed under finite unions.

Our technique. We introduce a new class of bi-positional winning conditions called *energy conditions over totally ordered groups*, or ETOG conditions for short. They are defined as follows (see more details in Section 3). We consider elements of some totally ordered group (we stress that it should be bi-ordered) as colors of edges. Given an infinite sequence of these elements, we arrange them into a formal series. Eve wants the sequence of its partial sums to have an infinite decreasing subsequence. Canonical energy conditions [5] can be defined in this way over $\mathbb{Z}$ with the standard ordering.

In Section 4, we establish the bi-positionality (over finite arenas) of the ETOG conditions using a sufficient condition of Gimbert and Zielonka. In Section 5, we refute the Kopczyński's conjecture over finite arenas and for finite unions. A key factor allowing us to do this is that free groups can be totally ordered. We construct two energy conditions over a free group with 2 generators whose union is not half-positional. We also observe in Section 5 that energy conditions over free groups are non-permuting, and that they can be used to refute 1-to-2-player lifting for half-positionality.

We believe that the class of energy conditions over totally ordered groups is interesting on its own. Namely, we find this class suitable for the following conjecture.

▶ **Conjecture 1.** *Every prefix-independent winning condition, bi-positional over finite arenas, coincides on periodic sequences with some energy condition over a totally ordered group.*

We cannot expect it to hold for all sequences, but periodic ones are sufficient, say, for algorithmic applications. If our conjecture is true, it gives an explicit description of the class of bi-positional prefix-independent winning condition. This would be in line with an explicit description of the class of *continuous* bi-positional payoffs from [16]. We discuss our conjecture in more detail in Section 6, where we reduce it to a problem about free groups.

2 Preliminaries

From now on, we restrict ourselves to finite arenas and to bi(half)-positionality over finite arenas.

If C is a set, we denote by C^* (resp., by C^ω) the set of all finite (resp., infinite) words over C. For $x \in C^*$, by $|x|$ we denote the length of x. Additionally, by C^+ we denote the set of all finite non-empty words over C. If $x \in C^+$, then by x^ω we denote an infinite word obtained by repeating x infinitely many times. The free group over C is denoted by F_C.

An arena $\mathcal{A}$ over a non-empty finite set (of colors) C is a tuple $\langle V_A, V_B, E\rangle$, where V_A and V_B are disjoint finite sets and $E \subseteq (V_A \cup V_E) \times C \times (V_A \cup V_B)$ is such that for every $s \in V_A \cup V_B$ there exist $c \in C$ and $t \in V_A \cup V_B$ for which $(s, c, t) \in E$. Elements of V_A are called Eve's nodes, and elements of V_B are called Adam's nodes. Elements of E are called edges of $\mathcal{A}$. An edge $e = (s, c, t) \in E$ is represented as a c-colored arrow from s to t. We use the notation $\mathsf{source}((s, c, t)) = s, \mathsf{col}((s, c, t)) = c$ and $\mathsf{target}((s, c, t)) = t$. Our definition guarantees that every node $v \in V_A \cup V_B$ has an out-going edge, that is, an edge e such that $\mathsf{source}(e) = v$.

An infinite-duration game over $\mathcal{A}$ from a node $s \in V_A \cup V_B$ is played as follows. At the beginning, one of the players chooses an edge $e_1 \in E$ with $\mathsf{source}(e_1) = s$. Namely, if $s \in V_A$, then Eve chooses e_1, and if $s \in V_B$, then Adam chooses e_1. More generally, in the first n turns players choose n edges $e_1, e_2, \dots, e_n \in E$, one edge per turn. These edges always form a *path* in $\mathcal{A}$, that is, we have $\mathsf{target}(e_1) = \mathsf{source}(e_2), \dots, \mathsf{target}(e_{n-1}) = \mathsf{source}(e_n)$. Then the $(n+1)$st turn is played as follows. Players consider the endpoint node of the current path, which is $\mathsf{target}(e_n)$. One of the players chooses an edge e_{n+1} with $\mathsf{source}(e_{n+1}) = \mathsf{target}(e_n)$.

Namely, if $\mathsf{target}(e_n) \in V_A$, then Eve chooses e_{n+1}, and if $\mathsf{target}(e_n) \in V_B$, then Adam chooses e_{n+1}. After infinitely many turns, players get an infinite sequence of edges $p = (e_1, e_2, e_3, \ldots)$ called a play (it forms an infinite path in $\mathcal{A}$).

A winning condition over a set of colors C is a subset $W \subseteq C^\omega$. A strategy of Eve is winning from $s \in V_A \cup V_B$ w.r.t. W if any play $p = (e_1, e_2, e_3, \ldots)$ with this strategy in the infinite-duration game over $\mathcal{A}$ from s is such that its sequence of colors $\mathsf{col}(e_1)\mathsf{col}(e_2)\mathsf{col}(e_3)\ldots$ belongs to W. Similarly, a strategy of Adam is winning from $s \in V_A \cup V_B$ w.r.t. W if any play $p = (e_1, e_2, e_3, \ldots)$ with this strategy in the infinite-duration game over $\mathcal{A}$ from s is such that $\mathsf{col}(e_1)\mathsf{col}(e_2)\mathsf{col}(e_3)\ldots \notin W$.

A positional strategy of Eve is a function $\sigma\colon V_A \to E$ such that $\mathsf{source}(\sigma(u)) = u$ for any $u \in V_A$. It is interpreted as follows: for any $u \in V_A$, whenever Eve has to choose an edge from u, she chooses $\sigma(u)$. Similarly, a positional strategy of Adam is a function $\tau\colon V_B \to E$ such that $\mathsf{source}(\tau(u)) = u$ for any $u \in V_B$. It is interpreted analogously.

A winning condition $W \subseteq C^\omega$ is *half-positional* if for every finite arena $\mathcal{A}$ over C there exists a positional strategy σ of Eve such that for every node s of $\mathcal{A}$ the following holds: if σ is not winning w.r.t. W from s, then Adam has a winning strategy w.r.t. W from s. A winning condition W is *bi-positional* if both W and its complement $C^\omega \setminus W$ are half-positional.

A winning condition $W \subseteq C^\omega$ is *prefix-independent* if for all $x \in C^*$ and $\alpha \in C^\omega$ we have $\alpha \in W \iff x\alpha \in W$.

We state the following sufficient condition for bi-positionality due to Gimbert and Zielonka.

▶ **Definition 2.** *Let $W \subseteq C^\omega$ be a winning condition over a finite set of colors C. We call W **fairly mixing** if the following 3 conditions hold:*

- *A) For every $x \in C^*$ and $\alpha, \beta \in C^\omega$ we have that*

$$(x\alpha \notin W \wedge x\beta \in W) \implies (\alpha \notin W \wedge \beta \in W).$$

- *B) For every $S \in \{W, C^\omega \setminus W\}$, for every $x \in C^+$ and for every $\alpha \in C^\omega$ we have that*

$$(x^\omega \in S, \alpha \in S) \implies (x\alpha \in S).$$

- *C) For every $S \in \{W, C^\omega \setminus W\}$ and for every infinite sequence $x_1, x_2, x_3, \ldots \in C^+$ it holds that:*

$$\Big[(x_1x_3x_5\ldots \in S) \wedge (x_2x_4x_6\ldots \in S) \wedge (\forall n \geq 1\ x_n^\omega \in S)\Big] \implies x_1x_2x_3\ldots \in S.$$

▶ **Theorem 3** ([11])**.** *Any fairly mixing winning condition is bi-positional over finite arenas.*

3 Definition of Energy Games over Totally Ordered Groups

A totally ordered group [8] is a triple $(G, +, \leq)$, where $(G, +)$ is a group and $\leq$ is a total order on G such that

$$a \leq b \implies x + a + y \leq x + b + y \qquad \text{for all } a, b, x, y \in G.$$

The neutral element of G is denoted by 0. We do not assume that $+$ is commutative[1].

[1] Possibly, more common is to use the multiplicative notation for non-Abelian groups. However, we find additive notation more suitable due to the intuition that comes with the standard energy games.

Consider any finite set C of colors and any totally ordered group $(G, +, \leq)$. By a *valuation of colors* over $(G, +, \leq)$ we mean any function $\mathbf{val}\colon C \to G$. It can be extended to a homomorphism $\mathbf{val}\colon C^* \to G$ by setting

$$\mathbf{val}(\text{empty word}) = 0, \qquad \mathbf{val}(c_1 c_2 \dots c_n) = \mathbf{val}(c_1) + \mathbf{val}(c_2) + \dots + \mathbf{val}(c_n).$$

Additionally, for every infinite sequence of colors $c_1 c_2 c_3 \dots \in C^\omega$, we denote by $\overline{\mathbf{val}}(c_1 c_2 c_3 \dots)$ the sequences of valuations of its finite prefixes:

$$\overline{\mathbf{val}}(c_1 c_2 c_3 \dots) = \{\mathbf{val}(c_1 \dots c_n)\}_{n=1}^{\infty}.$$

In other words, $\overline{\mathbf{val}}(c_1 c_2 c_3 \dots)$ is the sequence of partial sums of the formal series $\sum_{n=1}^{\infty} \mathbf{val}(c_n)$.

An energy condition over $(G, +, \leq)$, defined by a valuation of colors $\mathbf{val}\colon C \to G$, is the set $W \subseteq C^\omega$ of all $\alpha \in C^\omega$ such that $\overline{\mathbf{val}}(\alpha)$ has an infinite decreasing subsequence. It is immediate that any energy condition over a totally ordered group is prefix-independent.

As an illustration, we show that parity conditions fall into this definition. The parity condition over d priorities is a winning condition $W^d_{par} \subseteq \{1, 2, \dots, d\}^\omega$,

$$W^d_{par} = \{c_1 c_2 c_3 \dots \in \{1, 2, \dots, d\}^\omega \mid \limsup_{n \to \infty} c_i \text{ is odd}\}.$$

Observe that W^d_{par} is an energy condition over $\mathbb{Z}^d$ with the lexicographic ordering, defined by the following valuation:

$$\begin{aligned} \mathbf{val}(d) &= ((-1)^d, 0, \dots 0) \\ \mathbf{val}(d-1) &= (0, (-1)^{d-1}, \dots 0) \\ &\vdots \\ \mathbf{val}(1) &= (0, 0, \dots, -1). \end{aligned}$$

As far as we know, the most general class of bi-positional prefix-independent winning conditions that were previously considered are *priority mean payoff conditions* [13]. They can also be defined as energy conditions over $\mathbb{Z}^d$. Moreover, to define them, it is sufficient to consider only valuations that map each color to a vector with at most 1 non-zero coordinate, as in the case of parity conditions.

4 Bi-positionality of Energy Conditions over Totally Ordered Groups

In this section, we establish

▶ **Theorem 4.** *Every ETOG condition is bi-positional over finite arenas.*

We derive it from the following technical result (which will also be useful in Section 6). If C is a non-empty finite set and $W \subseteq C^\omega$, define $\mathsf{per}(W) = \{x \in C^+ \mid x^\omega \in W\}$ to be the set of periods of periodic words from W.

▶ **Proposition 5.** *Let C be a non-empty finite set. Consider any set $P \subseteq C^+$ such that both P and $C^+ \setminus P$ are closed under concatenations and cyclic shifts. Define a winning condition $W_P \subseteq C^\omega$ as follows:*

$$W_P = \{x y_1 y_2 y_3 \dots \mid x \in C^*, y_1, y_2, y_3, \dots \in P\}.$$

Then W_P is a prefix-independent fairly mixing winning condition with $P = \mathsf{per}(W_P)$.

Let us start with a derivation of Theorem 4.

Proof of Theorem 4 (modulo Proposition 5). Assume that $W \subseteq C^\omega$ is an energy condition over a totally ordered group $(G, +, \leq)$, defined by a valuation of colors $\mathbf{val}\colon C \to G$. Set $P = \{y \in C^+ \mid \mathbf{val}(y) < 0\}$. We claim that $W = W_P$. Indeed, W consists of all $\alpha = c_1c_2c_3\ldots \in C^\omega$ such that

$$\overline{\mathbf{val}}(\alpha) = (\mathbf{val}(c_1), \mathbf{val}(c_1c_2), \mathbf{val}(c_1c_2c_3), \ldots)$$

has an infinite decreasing subsequence. Consider any $i < j$. Observe that the jth element of $\overline{\mathbf{val}}(\alpha)$ is smaller than the ith element of $\overline{\mathbf{val}}(\alpha)$ if and only if

$$-\mathbf{val}(c_1\ldots c_i) + \mathbf{val}(c_1\ldots c_j) = \mathbf{val}(c_{i+1}\ldots c_j) < 0.$$

In other words, $\overline{\mathbf{val}}(\alpha)$ has an infinite decreasing subsequence if and only if $\alpha = c_1c_2c_3\ldots$ can be represented, except for some finite prefix, as a as a sequence of words with negative valuations. This means that $W = W_P$.

We now show that both P and $C^+ \setminus P$ are closed under concatenations and cyclic shifts. By Proposition 5, this would imply that $W = W_P$ is fairly mixing. In turn, by Theorem 3, this implies that W is bi-positional.

Consider any two words $x, y \in C^+$. Obviously:

$$\mathbf{val}(x) < 0, \mathbf{val}(y) < 0 \implies \mathbf{val}(xy) = \mathbf{val}(x) + \mathbf{val}(y) < 0,$$
$$\mathbf{val}(x) \geq 0, \mathbf{val}(y) \geq 0 \implies \mathbf{val}(xy) = \mathbf{val}(x) + \mathbf{val}(y) \geq 0.$$

This demonstrates that both P and $C^+ \setminus P$ are closed under concatenations. Now, we claim that $\mathbf{val}(c_1c_2\ldots c_n) < 0 \iff \mathbf{val}(c_2\ldots c_nc_1) < 0$ for any word $c_1c_2\ldots c_n \in C^+$ (this implies that both P and $C^+ \setminus P$ are closed under cyclic shifts). Indeed,

$$\begin{aligned}
&\mathbf{val}(c_1) + \mathbf{val}(c_2) + \ldots + \mathbf{val}(c_n) < 0\\
&\iff -\mathbf{val}(c_1) + (\mathbf{val}(c_1) + \mathbf{val}(c_2) + \ldots + \mathbf{val}(c_n)) + \mathbf{val}(c_1) < -\mathbf{val}(c_1) + 0 + \mathbf{val}(c_1)\\
&\iff \mathbf{val}(c_2) + \ldots + \mathbf{val}(c_n) + \mathbf{val}(c_1) < 0.
\end{aligned}$$

Proof of Proposition 5. Prefix-independence of W_P is immediate. We now show that $P = \mathsf{per}(W_P)$. We have $z^\omega \in W_P$ for any $z \in P$ by definition. Hence, $P \subseteq \mathsf{per}(W_P)$. Now, take any $z \in \mathsf{per}(W_P)$. We show that $z \in P$. By definition of $\mathsf{per}(W_P)$, we have $z^\omega = xy_1y_2y_3\ldots$ for some $x \in C^*$ and $y_1, y_2, y_3\ldots \in P$. There exist $i < j$ such that $|xy_1\ldots y_i|$ and $|xy_1\ldots y_j|$ are equal modulo $|z|$. This means that $y_{j+1}\ldots y_j$ must be a multiple of some cyclic shift of z. We have that $y_{j+1}\ldots y_j \in P$ because P is closed under concatenations. This means that this cyclic shift of z also belongs to P. Indeed, otherwise, we could write $y_{j+1}\ldots y_j$ as a multiple of some word from $C^+ \setminus P$, and this is impossible because $C^+ \setminus P$ is closed under concatenations. Since P is closed under cyclic shifts, we obtain $z \in P$.

Finally, we show that W_P is fairly mixing. Since W_P is prefix-independent, we should care only about the third item of Definition 2. That is, we only have to show the following two claims:

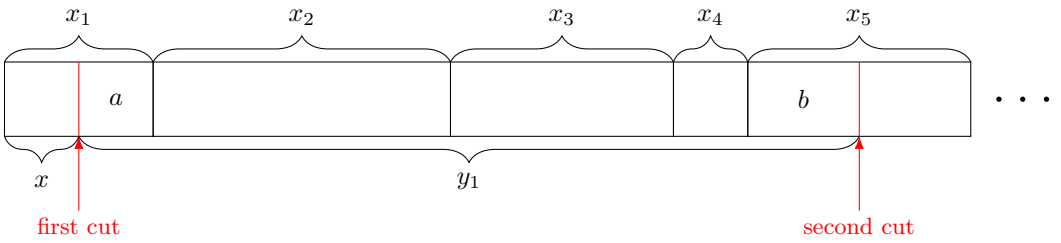
$$\left[(x_1x_3x_5\ldots \in W_P) \wedge (x_2x_4x_6\ldots \in W_P) \wedge (\forall n \geq 1\ x_n^\omega \in W_P)\right] \implies x_1x_2x_3\ldots \in W_P, \tag{1}$$

$$\left[(x_1x_3x_5\ldots \in \overline{W_P}) \wedge (x_2x_4x_6\ldots \in \overline{W_P}) \wedge (\forall n \geq 1\ x_n^\omega \in \overline{W_P})\right] \implies x_1x_2x_3\ldots \in \overline{W_P}, \tag{2}$$

for every infinite sequence of words $x_1, x_2, x_3, \ldots \in C^+$. Here, for brevity, by $\overline{W_P}$ we denote $C^\omega \setminus W_P$.

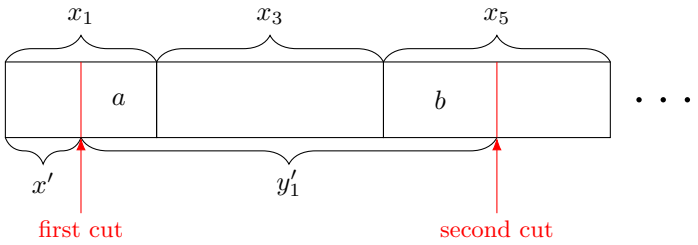
We first show (1). If $x_n^\omega \in W_P$ for every n, then $x_n \in \mathsf{per}(W_P) = P$ for every n, and hence $x_1x_2x_3\ldots \in W_P$ by definition.

A proof of (2) is more elaborate. Assume for contradiction that $x_1x_2x_3\ldots \in W_P$. Then we can write $x_1x_2x_3\ldots = xy_1y_2y_3\ldots$ for some $x \in C^*$ and $y_1, y_2, y_3, \ldots \in P$. One can represent the equality as a sequence of "cuts" inside $x_1x_2x_3\ldots$, as on the following picture:



Either there are infinitely many cuts inside x_n with odd indices, or there are infinitely many cuts inside x_n with even indices. Without loss of generality, we may assume that we only have cuts inside x_n with odd indices, and at most one for each n. Indeed, if necessary, we can join several successive y_i's into one word (this is legal because P is closed under concatenations).

We can now write each y_i as $y_i = ax_{2k}x_{2k+1}\ldots x_{2m}b$ for some $a, b \in C^*$ and $1 \le k \le m$. Now, let $y_i' = ax_{2k+1}x_{2k+3}\ldots x_{2m-1}b$ be a word which can be obtained from y_i by removing x_n with even indices. Additionally, we let $x' \in C^*$ be a word which can be obtained from x in the same way. Since each x_n with an even index lies entirely in some y_i or in x, we have that $x_1x_3x_5\ldots = x'y_1'y_2'y_3'\ldots$, as the following picture illustrates:



We will show that $y_i' \in P$ for every P. This would contradict the fact that $x_1x_3x_5\ldots \in \overline{W_P}$.

First, observe that $x_n \notin P$ for every n. Indeed, we are given that $x_n^\omega \in \overline{W_P}$ for every n. Hence, $x_n \notin \mathsf{per}(W_P) = P$, as required.

Assume for contradiction that $y_i' = ax_{2k+1}x_{2k+3}\ldots x_{2m-1}b \notin P$. Using the fact that $C^+ \setminus P$ is closed under concatenations and cyclic shifts, we obtain:

$$
\begin{aligned}
& y_i' = ax_{2k+1}x_{2k+3}\ldots x_{2m-1}b \notin P \\
\implies & x_{2k+1}x_{2k+3}\ldots x_{2m-1}ba \notin P \\
\implies & x_{2k}x_{2k+1}x_{2k+3}\ldots x_{2m-1}ba \notin P && \text{because } x_{2k} \notin P \\
\implies & x_{2k+3}\ldots x_{2m-1}bax_{2k}x_{2k+1} \notin P \\
\implies & x_{2k+2}x_{2k+3}\ldots x_{2m-1}bax_{2k}x_{2k+1} \notin P && \text{because } x_{2k+2} \notin P \\
& \vdots \\
\implies & x_{2m}bax_{2k}x_{2k+1}\ldots x_{2m-1} \notin P && \text{because } x_{2m} \notin P \\
\implies & y_i = ax_{2k}x_{2k+1}\ldots x_{2m}b \notin P,
\end{aligned}
$$

contradiction. ◀

5 Refuting Kopczyński's conjecture

▶ **Theorem 6.** *There exist two ETOG conditions whose union is not half-positional over finite arenas.*

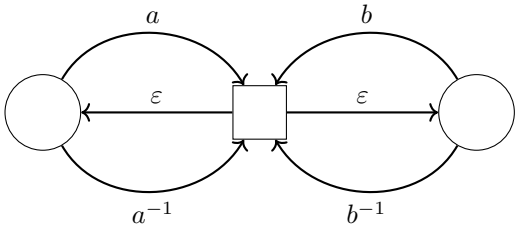
This theorem, together with the result that ETOG conditions are bi-positional over finite arenas (Theorem 4), refutes the Kopczyński's conjecture over finite arenas for finite unions.

Proof of Theorem 6. Consider the free group $F_{\{a,b\}}$ with 2 generators a, b. As was proved by Shimbireva [19], see also [8, Page 18], free groups can be totally ordered. We take an arbitrary total ordering $\leq$ of $F_{\{a,b\}}$. We also consider its inverse $\leq^{-1}$, which is also a total ordering of $F_{\{a,b\}}$. Define a set of colors $C = \{a, a^{-1}, b, b^{-1}, \varepsilon\}$. Here a^{-1}, b^{-1} are inverses of a, b in $F_{\{a,b\}}$, and ε is the identity element of $F_{\{a,b\}}$.

Let $W_1 \subseteq C^\omega$ be an energy condition over $(F_{\{a,b\}}, \leq)$, defined by a (suggestive) valuation of colors which interprets elements of C as corresponding elements of $F_{\{a,b\}}$. Similarly, we let $W_2 \subseteq C^\omega$ be an energy condition over $(F_{\{a,b\}}, \leq^{-1})$, defined by the same valuation. The only difference between W_1 and W_2 is that they are defined w.r.t. different total orderings of $F_{\{a,b\}}$ (one ordering is the inverse of the other one).

We show that the union $W_1 \cup W_2$ is not half-positional. It consists of all $\alpha \in C^\omega$ such that $\overline{\mathbf{val}}(\alpha)$ contains either an infinite decreasing subsequence w.r.t. $\leq$ or an infinite decreasing subsequence w.r.t. $\leq^{-1}$. In other words, it consists of all $\alpha \in C^\omega$ such that $\overline{\mathbf{val}}(\alpha)$ contains either an infinite decreasing subsequence or an infinite *increasing* subsequence w.r.t. $\leq$.

We show that $W_1 \cup W_2$ is not half-positional in the following finite arena.



Here, Eve controls the square and Adam controls the two circles. Assume that the game starts in the square. We show that Eve has a winning strategy w.r.t. $W_1 \cup W_2$, but not a positional one.

Eve has two positional strategies in this arena: always go to the left and always go to the right. Consider, for example, the first one. Adam has the following counter-strategy which wins against it: alternate the a-edge with the a^{-1}-edge. We get the following sequence of colors in the play of these two strategies:

$$\varepsilon a \varepsilon a^{-1} \varepsilon a \varepsilon a^{-1} \dots$$

This sequence does not belong to $W_1 \cup W_2$ because

$$\overline{\mathbf{val}}(\varepsilon a \varepsilon a^{-1} \varepsilon a \varepsilon a^{-1} \dots) = \varepsilon, a, a, \varepsilon, \varepsilon, a, a, \varepsilon, \dots$$

There are only two distinct elements of $F_{\{a,b\}}$ occurring in $\overline{\mathbf{val}}(\varepsilon a \varepsilon a^{-1} \varepsilon a \varepsilon a^{-1} \dots)$. Hence, it neither has an infinite decreasing subsequence nor an infinite increasing subsequence. By the same argument, the second positional strategy of Eve (always go to the right) is not winning w.r.t. $W_1 \cup W_2$ either.

On the other hand, Eve has the following winning strategy: alternate the edge to the left circle with the edge to the right circle. Consider any play with this strategy. Its sequence of colors looks as follows:

$$\varepsilon a^{\pm 1}\varepsilon b^{\pm 1}\varepsilon a^{\pm 1}\varepsilon b^{\pm 1}\dots$$

We show that this sequence belongs to $W_1 \cup W_2$. A restriction of $\overline{\mathbf{val}}(\varepsilon a^{\pm 1}\varepsilon b^{\pm 1}\varepsilon a^{\pm 1}\varepsilon b^{\pm 1})$ to elements with even indices looks like this:

$$a^{\pm 1},\ a^{\pm 1}b^{\pm 1},\ a^{\pm 1}b^{\pm 1}a^{\pm 1},\ a^{\pm 1}b^{\pm 1}a^{\pm 1}b^{\pm 1}\dots \tag{3}$$

All elements of (3) are distinct. Hence, by the Infinite Ramsey Theorem, it either has an infinite decreasing subsequence or an infinite increasing subsequence w.r.t. $\leq$. Indeed, consider an infinite complete graph over $\{1, 2, 3, \dots\}$, whose edges are colored green and red as follows. Pick any $i, j \in \{1, 2, 3, \dots\}$, $i < j$. If the ith element of (3) is bigger than the jth element of (3), then color the edge between i and j into green. Otherwise, color this edge red (in this case, the ith element of (3) is smaller than the jth element of (3)). Our graph has an infinite induced subgraph in which all edges are of the same color. If they are all green (resp., red), then this subgraph defines an infinite decreasing (resp., increasing) subsequence of (3). ◀

Additional remarks. Energy conditions over free groups are interesting because they are *non-permuting* (if there is more than one generator). A prefix-independent winning condition is permuting if it is closed under permuting periods of periodic sequences. All previously known prefix-independent bi-positional winning conditions were permuting. This is because they can be seen as energy conditions over Abelian groups (on periodic sequences). In a talk of Colcombet and Niwiński [6] it was asked whether there exists a non-permuting bi-positional prefix-independent winning condition. The answer is "yes". For example, take W_1 as above in this section. Without loss of generality, we may assume that $aba^{-1}b^{-1}$ is negative w.r.t. $\leq$ (otherwise we can consider its inverse). Then $(aba^{-1}b^{-1})^\omega \in W_1$, but $(aa^{-1}bb^{-1})^\omega \notin W_1$.

Additionally, the winning condition $W_1 \cup W_2$ is interesting because it refutes 1-to-2-player lifting for half-positionality. Namely, it is easy to see that $W_1 \cup W_2$ is positional for Eve in all arenas, where there are no nodes of Adam. This is because she can win in such arenas if and only if there is a reachable non-zero simple cycle. But as we have shown, $W_1 \cup W_2$ is not positional for Eve in the presence of Adam. Previously, there were examples that refute 1-to-2-player lifting for half-positionality in stochastic games [10].

6 Discussing Conjecture 1

First, it is useful to understand how prefix-independent bi-positional winning conditions are arranged on periodic sequences. Luckily, Proposition 5 gives an answer.

▶ **Proposition 7.** *Let C be a finite non-empty set. Then for any $P \subseteq C^+$ the following two conditions are equivalent:*

- *A) $P = \mathsf{per}(W)$ for some prefix-independent bi-positional (over finite arenas) winning condition $W \subseteq C^\omega$;*
- *B) P and $C^+ \setminus P$ are closed under concatenations and cyclic shifts;*

Proof. The fact that the second item implies the first item follows from Proposition 5. Indeed, if P and $C^+ \setminus P$ are closed under concatenations and cyclic shifts, then $P = \mathsf{per}(W_P)$ for a prefix-independent fairly mixing winning condition W_P, which is bi-positional by Theorem 3.

We now show that the first item implies the second item. The fact that P and $C^+ \setminus P$ are closed under cyclic shifts is a consequence of the prefix-independence of W:

$$c_1c_2\dots c_n \in P \iff (c_1c_2\dots c_n)^\omega \in W \iff c_n(c_1c_2\dots c_n)^\omega = (c_nc_1\dots c_{n-1})^\omega \in W$$
$$\iff c_nc_1\dots c_{n-1} \in P.$$

We now show that P is closed under concatenations (there is a similar argument for $C^+ \setminus P$). Take any $x, y \in P$. Consider the following arena.

It has a central circle node that lies on two simple cycles, one of which is colored by x and the other one by y. All nodes are controlled by Adam. Since, $x, y \in P$, we have that $x^\omega, y^\omega \in W$. Hence, Adam does not have a positional winning strategy w.r.t. W from the central circle. Since W is bi-positional, Adam has no winning strategy from the central circle w.r.t. W. Now, assume that Adam alternates the x-cycle with the y-cycle. He obtains $(xy)^\omega$ as a sequence of colors. Since this strategy is not winning, we have $xy \in P$. ◀

In turn, periods of periodic sequences of ETOG conditions are arranged as follows.

▶ **Proposition 8.** *Let C be a non-empty finite set and $W \subseteq C^\omega$ be an energy condition over a totally ordered group $(G, +, \leq)$, defined by a valuation of colors $\mathbf{val}\colon C \to G$. Then $\mathsf{per}(W) = \{x \in C^+ \mid \mathbf{val}(x) < 0\}$.*

Proof. Define $P = \{x \in C^+ \mid \mathbf{val}(x) < 0\}$. By the argument from the derivation of Theorem 4, we have $W = W_P$. Moreover, it was shown there that P and $C^+ \setminus P$ are closed under concatenations and cyclic shifts. Finally, by Proposition 5, we have that $P = \mathsf{per}(W_P) = \mathsf{per}(W)$. ◀

Thus, Conjecture 1 is equivalent to the following conjecture.

▶ **Conjecture 9.** *Let C be any non-empty finite set. Then for any $P \subseteq C^+$ such that P and $C^+ \setminus P$ are closed under concatenations and cyclic shifts there exists a totally ordered group $(G, +, \leq)$ and a valuation of colors $\mathbf{val}\colon C \to G$ such that $P = \{x \in C^+ \mid \mathbf{val}(x) < 0\}$.*

It might be concerning that P and $C^+ \setminus P$ are interchangeable in Conjecture 9, while **val** treats them asymmetrically. Namely, we require it to be negative on P and *non-negative* on $C^+ \setminus P$. However, **val** can always be made strictly positive on $C^+ \setminus P$. Namely, instead of G, consider the direct product $G \times \mathbb{Z}$ with the lexicographic order, and define a new valuation of colors $\mathbf{val}'\colon C \to G \times \mathbb{Z}$, $\mathbf{val}'(c) = (\mathbf{val}(c), 1)$.

Finally, we notice that our conjecture can be reduced to a reasoning about free groups.

▶ **Definition 10.** *A subset S of a group G is called an **invariant sub-semigroup** of G if the following two conditions hold:*

- *A) $xy \in S$ for all $x, y \in S$ (closure under multiplications);*
- *B) $gxg^{-1} \in S$ for all $g \in G, x \in S$ (closure under conjugations with elements of G).*

▶ **Conjecture 11.** *Consider an arbitrary non-empty finite set C and any $P \subseteq C^+$ such that P and $C^+ \setminus P$ are closed under concatenations and cyclic shifts. Then there exists an invariant sub-semigroup S of the free group F_C such that, first, $C^+ \setminus P$ is a subset of S, second, P is disjoint with S, and third, for every $g \in F_C$ either $g \in S$ or $g^{-1} \in S$ (in particular, S must have the neutral element).*

▶ **Proposition 12.** *Conjecture 9 is equivalent to Conjecture 11.*

Proof. Consider an arbitrary non-empty finite set C. It is sufficient to show that for any $P \subseteq C^+$ the following two conditions are equivalent:

- A) there exist a totally ordered group $(G, +, \leq)$ and a valuation of colors $\mathbf{val}\colon C \to G$ such that $P = \{x \in C^+ \mid \mathbf{val}(x) < 0\}$.
- B) there exists an invariant sub-semigroup S of the free group F_C such that, first, $C^+ \setminus P$ is a subset of S, second, P is disjoint with S, and third, for every $g \in F_C$ either $g \in S$ or $g^{-1} \in S$.

We first establish A) $\implies$ B). Extend $\mathbf{val}$ to a homomorphism from F_C to G by setting $\mathbf{val}(c^{-1}) = -\mathbf{val}(c)$ for $c \in C$. Set $S = \{g \in F_C \mid \mathbf{val}(g) \geq 0\}$. It is easy to check that all conditions on S are satisfied.

Now we establish B) $\implies$ A). Let S be as in B). Consider a binary relation $\sim$ on F_C, defined by $f \sim g \iff fg^{-1}, gf^{-1} \in S$ for $f, g \in F_C$. The fact that S is an invariant sub-semigroup with the neutral element implies that $\sim$ is a congruence on the group F_C. Let $G = F_C/\sim$ be the corresponding quotient group. Now, consider a binary relation $\preceq$ on F_C, defined by $f \preceq g \iff gf^{-1} \in S$ for $f, g \in F_C$ (observe that $f \sim g \iff f \preceq g, g \preceq f$). It is easy to see that $\preceq$ is correctly defined over $F_C/\sim$, whose elements are equivalence classes of $\sim$. More formally, it holds that if $a \sim b, x \sim y$, then $a \preceq x \iff b \preceq y$ (it can again be derived from the fact that S is an invariant sub-semigroup). It is also routine to check that $\preceq$ defines a total ordering on G. We need a condition that either $g \in S$ or $g^{-1} \in S$ for every $g \in F_C$ only to show the totality of our order. Namely, to show that there are no $f, g \in F_C$ with $f \not\preceq g$ and $g \not\preceq f$, we notice that otherwise neither gf^{-1} nor $fg^{-1} = (gf^{-1})^{-1}$ are in S. Observe that the equivalence class of $g \in F_C$ w.r.t. $\sim$ is non-negative in $(G, \preceq)$ if and only if $g \in S$. Now, recall that $C^+ \setminus P$ is a subset of S and P is disjoint with P. Hence, if we consider a valuation of colors $\mathbf{val}\colon C \to G$, which maps $c \in C$ to its equivalence class w.r.t. $\sim$, then P would be the set of words from C^+ whose valuation is negative w.r.t. $\preceq$. ◀

References

1. Alessandro Bianco, Marco Faella, Fabio Mogavero, and Aniello Murano. Exploring the boundary of half-positionality. *Annals of Mathematics and Artificial Intelligence*, 62(1):55–77, 2011.
2. Roderick Bloem, Krishnendu Chatterjee, and Barbara Jobstmann. Graph games and reactive synthesis. In *Handbook of Model Checking*, pages 921–962. Springer, 2018.
3. Patricia Bouyer, Antonio Casares, Mickael Randour, and Pierre Vandenhove. Half-positional objectives recognized by deterministic büchi automata. In *33rd International Conference on Concurrency Theory (CONCUR 2022)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.
4. Patricia Bouyer, Mickael Randour, and Pierre Vandenhove. Characterizing omega-regularity through finite-memory determinacy of games on infinite graphs. *TheoretiCS*, 2(1):1–48, 2023.
5. Arindam Chakrabarti, Luca de Alfaro, Thomas A Henzinger, and Mariëlle Stoelinga. Resource interfaces. In *International Workshop on Embedded Software*, pages 117–133. Springer, 2003.

6 Thomas Colcombet and Damian Niwiński. Positional determinacy over finite games. Slides, available at https://www.irif.fr/~colcombe/Talks/talk_posdet_rennes_13.10.05.pdf, 2005.

7 Thomas Colcombet and Damian Niwiński. On the positional determinacy of edge-labeled games. *Theoretical Computer Science*, 352(1-3):190–196, 2006.

8 Bertrand Deroin, Andrés Navas, and Cristóbal Rivas. Groups, orders, and dynamics. *arXiv preprint*, 2014. arXiv:1408.5805.

9 EA Emerson and CS Jutla. Tree automata, mu-calculus and determinacy. In *[1991] Proceedings 32nd Annual Symposium of Foundations of Computer Science*, pages 368–377. IEEE, 1991.

10 H Gimbert and E Kelmendi. Two-player perfect-information shift-invariant submixing stochastic games are half-positional. corr, abs/1401.6575. *arXiv preprint*, 2014. arXiv:1401.6575.

11 Hugo Gimbert and Wiesław Zielonka. When can you play positionally? In *International Symposium on Mathematical Foundations of Computer Science*, pages 686–697. Springer, 2004.

12 Hugo Gimbert and Wiesław Zielonka. Games where you can play optimally without any memory. In *International Conference on Concurrency Theory*, pages 428–442. Springer, 2005.

13 Hugo Gimbert and Wiesław Zielonka. Deterministic priority mean-payoff games as limits of discounted games. In *International Colloquium on Automata, Languages, and Programming*, pages 312–323. Springer, 2006.

14 Erich Grädel, Wolfgang Thomas, and Thomas Wilke. *Automata, logics, and infinite games: a guide to current research*, volume 2500. Springer, 2003.

15 Eryk Kopczyński. Half-positional determinacy of infinite games. In *International Colloquium on Automata, Languages, and Programming*, pages 336–347. Springer, 2006.

16 Alexander Kozachinskiy. Continuous Positional Payoffs. In Serge Haddad and Daniele Varacca, editors, *32nd International Conference on Concurrency Theory (CONCUR 2021)*, volume 203 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 10:1–10:17, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.CONCUR.2021.10.

17 Pierre Ohlmann. *Monotonic graphs for parity and mean-payoff games*. PhD thesis, IRIF – Research Institute on the Foundations of Computer Science, 2021.

18 Pierre Ohlmann. Characterizing positionality in games of infinite duration over infinite graphs. *TheoretiCS*, 2(3):1–51, 2023.

19 H Shimbireva. On the theory of partially ordered groups. *Matematicheskii Sbornik*, 62(1):145–178, 1947.

20 Wieslaw Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1-2):135–183, 1998.

QLTL Model-Checking

François Laroussinie
IRIF, Université Paris Cité, France

Loriane Leclercq
Ecole Centrale de Nantes, CNRS, LS2N, Nantes, France

Arnaud Sangnier
DIBRIS, Università di Genova, Italy

Abstract

Quantified LTL (QLTL) extends the temporal logic LTL with quantifications over atomic propositions. Several semantics exist to handle these quantifications, depending on the definition of executions over which formulas are interpreted: either infinite sequences of subsets of atomic propositions (aka the "tree semantics") or infinite sequences of control states combined with a labelling function that associates atomic propositions to the control states (aka the "structure semantics"). The main difference being that in the latter different occurrences of a control state should be labelled similarly. The tree semantics has been intensively studied from the complexity and expressivity point of view (especially in the work of Sistla [21, 22]) for which the satisfiability and model-checking problems are known to be TOWER-complete. For the structure semantics, French has shown that the satisfiability problem is undecidable [8]. We study here the model-checking problem for QLTL under this semantics and prove that it is EXPSPACE-complete. We also show that the complexity drops down to PSPACE-complete for two specific cases of structures, namely path and flat ones.

2012 ACM Subject Classification Theory of Computation → Logic

Keywords and phrases Quantified Linear-time temporal logic, Model-cheking, Verification, Automata theory

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.35

Funding *Loriane Leclercq*: ANR project ProMiS ANR-19-CE25-0015

1 Introduction

Temporal logics (TLs) have been introduced in computer science in the late 1970's by Pnueli [19]; they provide a powerful formalism for specifying correctness properties of evolving systems. Various kinds of temporal logics have been defined, with different expressive power and algorithmic properties. For instance, the *Computation Tree Logic* (CTL) expresses properties of the computation tree of the system under study (time is branching: a state may have several successors), and the *Linear-time Temporal Logic* (LTL) expresses properties of one execution at a time (a system is viewed as a set of executions).

In verification, we are mainly interested in two decision problems: the satisfiability problem (given a formula, decide whether there exists a model for it) and the model-checking problem (given a potential model and a formula, decide whether the formula holds true or not over the model). For the temporal logic LTL, it is well known that both these problems are PSPACE-complete. The key argument is the construction of an automaton that recognises the models (a set of infinite words) of a fixed formula. For CTL, the model-checking can be done in polynomial time while satisfiability is EXPTIME-complete (these decision procedures can also be based on automata techniques, but this time we need to use tree automata).

In terms of expressiveness, classical temporal logics like CTL or LTL still have some limitations: in particular, they lack the ability of *counting*. For instance, they cannot express that an event occurs (at least) at every even position along a path, or that a state has

exactly two successors. In order to cope with this, temporal logics have been extended with *propositional quantifiers* [21, 22]: in this framework, a formula $\exists p.\varphi$ is verified when there is a way of labelling the current execution by p in such a way that φ holds true.

Different semantics for quantified TLs have been studied in the literature depending on the definition of the quantifiers. For example, consider a formula Φ of the linear-time temporal logic QLTL (i.e., LTL extended with propositions quantifiers). We can interpret Φ either over an infinite word over the alphabet 2^{AP} (where AP denotes the set of atomic propositions used in Φ), or over a *labelled execution* composed by an infinite sequence of control states (some of them may have several occurrences along the execution) and a labelling function which maps each control state to a set of atomic propositions. These two points of view coincide for LTL formulas. But as soon as quantifiers are added, the two semantics are quite different: in the first one (called the *tree semantics*), each position along the execution corresponds to a new (control) state whose labelling is independent of the others. In the second one (called the *structure semantics*), two occurrences of the same control state are always labelled in the same manner.

The tree semantics has been studied extensively, especially in [22] where QLTL is proven to be as expressive as S1S (the monadic second-order logic with one successor), and the satisfiability and model-checking problems for the k-alternation fragment are shown k-EXPSPACE-complete.

For QLTL with the structure semantics, the main result concerns the satisfiability problem. In this setting, the satisfiability problem is stated as follows: given a QLTL formula Φ, does there exist an infinite word $\rho \in Q^\omega$ where Q is an alphabet describing the control states, and a labelling $\ell : Q \mapsto 2^{\mathsf{AP}}$ (where AP is the set of atomic propositions occuring in Φ) such that the labelled execution (ρ, ℓ) satisfies Φ? French has shown that this problem is undecidable even when Q is assumed to be finite [8].

In this paper, we focus on the model-checking problem for QLTL under the structure semantics. We first explain the types of properties we can express with this logic, and then we show that the problem is EXPSPACE-complete. We also consider two other special model-checking problems: when the structure is a path of the form $\rho_1 \cdot \rho_2^\omega$ and when the structure is *flat*. In both cases, we prove that the problem is then PSPACE-complete.

Related results. Adding quantifications over atomic propositions in LTL has been first considered in [21, 22] for the tree semantics: both the expressiveness and the complexity have been studied. The stutter-invariant fragment of QLTL, with a restricted notion of propositional quantification, was developed in [6]. Proof systems for QLTL were developed (still with the tree semantics), both with and without past-time modalities [11, 9].

For QLTL under the structure semantics, the main contribution is [8] where French shows that satisfiability is undecidable.

Propositional quantifications have also been studied for branching-time temporal logics. The extension of CTL^* with external existential quantification was compared with tree automata over binary trees [5]. In [13], restricted quantifications are added to CTL. In [7], the satisfiability of QCTL^* was proven undecidable in the structure semantics, and decidable in the tree semantics. In [14], QCTL and QCTL^* are considered (both for the structure semantics and the tree semantics): the expressive power (relationship with the monadic second-order logic MSO) and complexity (for model-checking and satisfiability) are studied. In [1], the satisfiability problem of several fragments of QCTL (e.g. with only the **EX** modality) under the tree semantics is shown to be TOWER-complete, i.e. exactly as it is for the full QCTL logic. In [10], a model-checking algorithm is proposed for QCTL with the structure semantics based on a reduction to QBF and experimental results are discussed (with several QBF solvers).

The model-checking problem restricted to a path has been studied in [16, 18]. The (existential) model-checking of LTL over flat structures has been first studied in [12] where it is shown to be in NP and later on, in [4], the problem has been shown to be NP-complete even when adding past operators.

2 Model and Logic

2.1 Kripke Structures and Labelled Executions

In formal verification, systems are usually modelled with *labelled transition systems* (or *Kripke structures*): we have a finite set of control states, transitions to move from a state to another one and a labelling function which associates a set of atomic propositions with every control state. In this paper, we consider a countable set of atomic propositions denoted by AP. Kripke structures are then formally defined as follows.

▶ **Definition 1.** *A* Kripke structure (KS) *is a tuple* $\mathcal{K} = \langle Q, R, q_{in}, \ell \rangle$ *where* Q *is a finite set of states,* $R \subseteq Q \times Q$ *is a set of transitions (we assume that for any* $q \in Q$*, there exists* $q' \in Q$ *s.t.* $(q, q') \in R$*),* $q_{in} \in Q$ *is the initial control state and* $\ell \colon Q \to 2^{\mathsf{AP}}$ *is a labelling function.*

The semantics of such a transition system is either the structure itself or its unfolding (an infinite labelled tree), and an execution can be seen either as an infinite sequence of labelled control states or an infinite word over the alphabet 2^{AP} (the underlying control states are forgotten). These two points of view coincide when considering fragments of CTL^* logic: the truth value of a formula does not depend on this choice of semantics. There is no way to distinguish two different control states if their behaviours defined in terms of atomic proposition and transitions are "equivalent" (i.e. bisimilar) w.r.t. the considered logic. Classically the infinite 2^{AP}-labelled tree unfolding is used to base decision procedures over automata theory (word automata for linear-time temporal logic, or tree automata for branching-time temporal logic). And the "structure" view is used for example to develop classical algorithms for CTL-like logics (corresponding to basic graph analysis).

Adding quantifications over atomic propositions makes a big difference (see Section 3) between these two approaches. Indeed labelling control states implies that every occurrence of a state will carry the same labels. It can be seen as a second order quantification over the control states of the structure. For example it becomes possible to specify that there exists a self-loop from a given state (we can mark a single control state in order to distinguish it from other ones). With the "labelled tree" semantics, every position can be labelled independently. As in previous works on this topic, we will refer to the former semantics as the *structure semantics* and the latter as the *tree semantics*. For the linear-time logic QLTL, the most popular approach is the tree semantics: in [22], Sistla *et al.* showed important properties about its complexity and its relation with Büchi automata. Here we consider the structure semantics where the model for QLTL formulas are labelled executions.

An *execution* ρ of a Kripke structure $\mathcal{K} = \langle Q, R, q_{in}, \ell \rangle$ is an infinite sequence of states $q_0 q_1 q_2 \dots$ such that $(q_i, q_{i+1}) \in R$ for all $i \in \mathbb{N}$. Given $i \in \mathbb{N}$, we use $\rho(i)$ to denote q_i the i-th (control) state of ρ, and $\rho_{\geq i}$ to denote $q_i q_{i+1} q_{i+2} \dots$ its i-th suffix. Finally a *labelled execution* is a pair (ρ, λ) where ρ is an execution and λ is a labelling function from Q to 2^{AP}. We use $\mathsf{Exec}_{\mathcal{K}}(q)$ [resp. $\mathsf{Exec}^{\mathsf{lab}}{}_{\mathcal{K}}(q)$] to denote the set of executions ρ [resp. labelled executions (ρ, λ)] in $\mathcal{K}$ starting from q, i.e. such that $\rho(0) = q$.

2.2 Syntax and (Structure) Semantics of QLTL

We present now the definition of the logic QLTL, which extends the classical linear-time temporal logic LTL with quantifications over atomic propositions. The syntax of QLTL is given by the following grammar:

$$\varphi ::= q \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathbf{X}\varphi \mid \varphi \mathbf{U} \varphi \mid \exists p.\ \varphi$$

where q and p range over AP.

In the structure semantics, QLTL formulas are evaluated over *labelled executions* of a Kripke structure $\mathcal{K} = \langle Q, R, q_{in}, \ell \rangle$. Before providing the formal semantics, we need to introduce another notion. Given a set $P \subseteq \mathsf{AP}$, two labellings λ and λ' from Q to 2^{AP} are said to be P-equivalent (denoted by $\lambda \equiv_P \lambda'$) iff $\lambda(q) \cap P = \lambda'(q) \cap P$ for every $q \in Q$. The semantics of QLTL formulas is then provided by the satisfaction relation $\models$ between a labelled execution (ρ, λ) and a formula φ which is defined inductively as follows:

$(\rho, \lambda) \models p$ iff $p \in \lambda(\rho(0))$
$(\rho, \lambda) \models \neg\varphi$ iff $(\rho, \lambda) \not\models \varphi$
$(\rho, \lambda) \models \varphi \vee \psi$ iff $(\rho, \lambda) \models \varphi$ or $(\rho, \lambda) \models \psi$
$(\rho, \lambda) \models \mathbf{X}\varphi$ iff $(\rho_{\geq 1}, \lambda) \models \varphi$
$(\rho, \lambda) \models \varphi \mathbf{U} \psi$ iff there exists $i \geq 0$ s.t. $(\rho_{\geq i}, \lambda) \models \psi$ and $(\rho_{\geq j}, \lambda) \models \varphi$ for all $i > j \geq 0$
$(\rho, \lambda) \models \exists p.\ \varphi$ iff there exists a labelling λ' s.t. $\lambda' \equiv_{\mathsf{AP} \setminus \{p\}} \lambda$ and $(\rho, \lambda') \models \varphi$

In the sequel, we use standard abbreviations such as $\top$, $\bot$, $\wedge$, $\Rightarrow$ and $\Leftrightarrow$. We also use the additional (classical) temporal modalities of LTL : $\mathbf{F}\varphi = \top \mathbf{U} \varphi$, $\mathbf{G}\varphi = \neg\mathbf{F}\neg\varphi$ and $\varphi \mathbf{R} \psi = \neg((\neg\varphi)\mathbf{U}(\neg\psi))$. Moreover, we use the following abbreviations related to quantifiers over atomic propositions : $\forall p.\ \varphi = \neg\exists p.\ \neg\varphi$, and for a set $P = \{p_1, \ldots, p_k\} \subseteq \mathsf{AP}$, we write $\exists P.\varphi$ for $\exists p_1 \ldots \exists p_k.\varphi$ and $\forall P.\varphi$ for $\forall p_1 \ldots \forall p_k.\varphi$.

Given a formula φ, we denote by $\mathsf{SubF}(\varphi)$ the set of its subformulas and $\mathsf{Prop}(\varphi)$ the set of the atomic propositions. A proposition p is said to be free in φ if it appears out of scope of some operator $\exists p. \ldots$ or $\forall p. \ldots$.

The size of a formula $\varphi \in$ QLTL, denoted $|\varphi|$, is defined inductively by : $|q| = 1$, $|\neg\varphi| = |\exists p.\varphi| = |\forall p.\varphi| = |\mathbf{X}\varphi| = 1 + |\varphi|$, $|\varphi \vee \psi| = |\varphi \mathbf{U} \psi| = |\varphi \mathbf{R} \psi| = 1 + |\varphi| + |\psi|$. Moreover, we use $\mathsf{th}(\varphi)$ to denote the *temporal height* of φ, i.e. the maximal number of nested temporal modalities in φ.

A formula is said to be in negated normal form (NNF) if negations apply only to atomic propositions. Any QLTL formula φ can be transformed into an equivalent NNF formula ψ s.t. $|\psi| = O(|\varphi|)$. Note that ψ is built from boolean operators $\wedge$ and $\vee$, Temporal modalities $\mathbf{X}$, $\mathbf{U}$ and $\mathbf{R}$, atomic propositions and their negations, and quantifiers $\exists$ and $\forall$.

Two QLTL formulas φ and ψ are said to be *equivalent* (written $\varphi \equiv \psi$) iff for any labelled execution (ρ, λ), we have $(\rho, \lambda) \models \varphi$ iff $(\rho, \lambda) \models \psi$. This equivalence is substitutive.

We write $\mathcal{K} \models_\exists \varphi$ when φ is satisfied by a labelled execution (ρ, ℓ) in $\mathcal{K}$ rooted at the initial state q_{in}, and $\mathcal{K} \models_\forall \varphi$ when every such labelled execution in $\mathcal{K}$ satisfy φ.

3 What can we express with QLTL?

To illustrate the kind of properties that can be expressed with QLTL with the structure semantics, we introduce the following abbreviation:

$$\exists^1 p.\varphi \quad = \quad \exists p.\Big(\mathbf{F}p \wedge \big(\forall p'.(\mathbf{F}(p \wedge p') \ \Rightarrow\ \mathbf{G}(p \Rightarrow p'))\big) \wedge \varphi\Big)$$

and its dual $\forall^1 p.\varphi = \neg\exists^1 p.\neg\varphi$. Informally $\exists^1 p.\varphi$ is satisfied if one can label exactly one control state (reachable from the current position) by p in order to make φ to be satisfied by the execution.

Now we can express the fact that a labelled execution (ρ, λ) is built from a *finite* number of control states (which is of course always the case when considering finite KS). The following formula expresses this property:

$$\Phi_{\text{finite}} \quad = \quad \exists^1 q.\forall p.\Big((p\mathbf{U}(q \wedge p)) \Rightarrow (\mathbf{G}p)\Big)$$

Assume that the property holds true for (ρ, λ). Then there is a *last occurring* control state: the control state whose first occurrence is located after the first occurrence of every other control state. Clearly if we label this state by q, any p-labelling of all states occurring before q-state labels all states of ρ. Conversely if the formula is satisfied by some execution (ρ, λ), it implies that the p-labelling limited to control states occurring over a finite prefix allows us to label all ρ-states.

One can also ensure that at least k distinct control states occur infinitely often along an execution with the following formula:

$$\Phi_{\geq \mathsf{k}} \quad = \quad \exists^1 q_1 \dots q_k. \bigwedge_{1\leq i\neq j\leq k} \mathbf{G}(\neg q_i \vee \neg q_j) \wedge \bigwedge_{1\leq i\leq k} \Big(\mathbf{GF}(\, q_i)\Big)$$

We can also specify that any *position* satisfying the proposition a is always followed by the same control state with:

$$\Phi_{\text{succ}} \quad = \quad \exists^1 q.\Big(\mathbf{G}\,(a \Rightarrow \mathbf{X}\,q)\Big)$$

An execution (ρ, λ) is deterministic if every control state occurring along ρ is always followed by the same control state. This property can easily be expressed with QLTL:

$$\Phi_{\text{deter}} \quad = \quad \forall^1 q.\forall^1 q'.\Big(\mathbf{F}(q \wedge \mathbf{X}q') \Rightarrow \mathbf{G}(q \Rightarrow \mathbf{X}q')\Big)$$

Indeed, assume we label two control states by q and q' respectively, and the formula $(q \wedge \mathbf{X}q')$ holds true at a position along ρ, then we require that every occurrence of q is followed by q'. This is precisely the definition of deterministic execution.

When the set of control states Q is fixed and finite, any quantification $\exists p.\varphi$ is equivalent to some disjunction of all possible subsets associated with the proposition p. And then any QLTL formula is equivalent *for executions built with exactly $|Q|$-control states* to some QLTL formula with a unique existential quantification:

▶ **Proposition 2.** *Let $k \geq 1$. Any* QLTL *formula Φ is equivalent* for labelled executions containing exactly k distinct control states *to some* QLTL *formula $\widetilde{\Phi} = \exists p_0 \dots p_{2^k-1}.\widehat{\Phi}$ where $\widehat{\Phi}$ belongs to* LTL*.*

Proof. Consider a set of control states Q s.t. $|Q| = k$. We use 2^k new atomic propositions in order to label every possible subset over Q. The formula $\widehat{\Phi}$ combines two parts: the first one ensures this labelling of subsets and the second one is just Φ where every existential (resp. universal) quantification is replaced by a disjunction (resp. conjunction). Formally given a QLTL formula φ, we define $\widetilde{\varphi}^k$ inductively as follows:

$$\widetilde{p}^k = p \quad \widetilde{\varphi_1 \wedge \varphi_2}^k = \widetilde{\varphi_1}^k \wedge \widetilde{\varphi_2}^k \quad \widetilde{\neg\varphi_1}^k = \neg\widetilde{\varphi_1}^k \quad \widetilde{\varphi_1\mathbf{U}\varphi_2}^k = \widetilde{\varphi_1}^k\mathbf{U}\widetilde{\varphi_2}^k$$
$$\widetilde{\mathbf{X}\varphi_1}^k = \mathbf{X}\widetilde{\varphi_1}^k \quad \widetilde{\exists p.\alpha}^k = \bigvee_{0\leq i<2^k} \widetilde{\alpha[p \mapsto p_i]}^k \quad \widetilde{\forall p.\alpha}^k = \bigwedge_{0\leq i<2^k} \widetilde{\alpha[p \mapsto p_i]}^k$$

And then it remains to show that for any execution (ρ, ℓ) and for any QLTL formula Φ, if the number of distinct control states occurring along ρ equals k, we have:

$$(\rho,\ell) \models \Phi \quad \text{iff} \quad (\rho,\ell) \models \exists p_0 \dots p_{2^k-1}.\Big(\bigwedge_{0\le i<j<2^k} \mathbf{F}(p_i \Leftrightarrow \neg p_j)\Big) \wedge \widetilde{\Phi}^k$$

The proof is based on the fact that the labelling of every p_i labels a specific subset of control states. This is ensured by the first part of the formula: there are 2^k labellings and all are different. Property 2 is then proved. ◀

We can observe several important differences with the tree semantics [1]. For example, under the tree semantics, the formula Φ_{deter} is valid (as every position corresponds to a new control state). It is also well known that the property $\mathsf{Even}^{\mathsf{c}}(p)$ defined by "a control state is labelled by p iff at least one of its occurrences is located at an even position along the execution" can easily be expressed with the tree semantics, but this is not true anymore with the structure semantics:

▶ **Proposition 3.** *With the structure semantics, there is no* QLTL *formula equivalent to the property* $\mathsf{Even}^{\mathsf{c}}(p)$.

Proof. We reuse a construction of [24] for proving that LTL cannot express that a proposition holds for every even state. Consider the set of control states $Q = \{q, q'\}$ and the execution $\rho_k = q^k \cdot q' \cdot q^\omega$ for $k > 0$. We can easily observe that the labelled execution (ρ_k, ℓ) satisfies $\mathsf{Even}^{\mathsf{c}}(p)$ if and only if (1) k is even and both q and q' are labelled by p, or (2) k is odd and only q is labelled by p.

Now we can show that given $k, k' \geq 1$ and any LTL formula ψ s.t. $|\psi| \leq \min(k, k')$, we have: $(\rho_k, \ell) \models \psi \;\Leftrightarrow\; (\rho_{k'}, \ell) \models \psi$. The proof of this result is done by induction over $|\psi|$:

- $|\psi| = 1$: ψ is an atomic proposition, and $\ell(\rho_k(0)) = \ell(\rho_{k'}(0))$ as $k, k' \geq 1$.
- $|\psi| = n + 1$: We distinguish several cases (and omit the case of Boolean combinators):
  - $\psi = \mathbf{X}\psi_1$: consider $k, k' \geq n + 1$ and assume $(\rho_k, \ell) \models \mathbf{X}\psi_1$. Then $(\rho_{k-1}, \ell) \models \psi_1$. As both $k - 1$ and $k' - 1$ are greater or equal to $n = |\psi_1|$, we have by i.h. $(\rho_{k'-1}, \ell) \models \psi_1$ and then $(\rho_{k'}, \ell) \models \mathbf{X}\psi_1$.
  - $\psi = \psi_1 \mathbf{U} \psi_2$: consider $k, k' \geq n + 1$ and assume $(\rho_k, \ell) \models \psi_1 \mathbf{U} \psi_2$. Then there exists $i \geq 0$ s.t. $((\rho_k)_{\geq i}, \ell) \models \psi_2$ and for any $0 \leq j <$ we have $((\rho_k)_{\geq j}, \ell) \models \psi_1$. We distinguish several cases:
    - $k > k'$ and $k - i \leq k'$: then ψ_2 holds true for $((\rho_{k'})_{\geq i-(k-k')}, \ell)$ and ψ_1 is verified for any $((\rho_{k'})_{\geq j}, \ell)$ for $0 \leq j < i - (k - k')$, and this provides the result $((\rho_{k'}), \ell) \models \psi$.
    - $k > k'$ and $k - i > k'$: In this case, we know that $((\rho_k)_{\geq i}, \ell) \models \psi_2$ is equivalent to $((\rho_{k-i}), \ell) \models \psi_2$ and $k - i$ and k' are both greater than $|\psi_2|$, which allows us to use the i.h. and deduce $((\rho_{k'}), \ell) \models \psi_2$, and then $((\rho_{k'}), \ell) \models \psi$.
    - $k < k'$: if $(\rho_k, \ell) \models \psi_1$, then we can use the i.h. as in the previous case to deduce that for all j s.t. $(\rho_{k'-j}, \ell) \models \psi_1$ for any $0 \leq j \leq k' - k$. This ensures the result. And if $(\rho_k, \ell) \models \psi_2$, we deduce directly $(\rho_{k'}, \ell) \models \psi_2$ by the i.h. In both case, we obtain $(\rho_{k'}, \ell) \models \psi$

Now assume that there exists some QLTL formula Ψ equivalent to $\mathsf{Even}^{\mathsf{c}}(p)$. From Proposition 2, there exists a formula of the form $\Psi' = \exists P.\psi$ with $\psi \in \mathsf{LTL}$ such that Ψ and Ψ' are equivalent over all the executions ρ_k (and more generally over any execution containing two control states). Let K be the size of ψ.

[1] The tree semantics has not been formally defined but it just consists in interpreting formulas over words in $(2^{\mathsf{AP}})^\omega$ or equivalently over labelled executions where every control state occurs exactly once.

Consider the odd integer $\lambda = 2 \cdot K + 1$ and the labelling ℓ defined by $\ell(q) = \{p\}$ and $\ell(q') = \emptyset$. We know that $(\rho_\lambda, \ell) \models \mathsf{Even}^{\mathsf{c}}(p)$ and $(\rho_{\lambda+1}, \ell) \not\models \mathsf{Even}^{\mathsf{c}}(p)$, and therefore $(\rho_\lambda, \ell) \models \Psi'$ and $(\rho_{\lambda+1}, \ell) \not\models \Psi'$. Then there exists an extended labelling ℓ' of ℓ such that $(\rho_\lambda, \ell') \models \psi$. And from the previous result, we can deduce $(\rho_{\lambda+1}, \ell') \models \psi$ as λ and $\lambda + 1$ are greater than $|\psi|$, and this contradicts the fact $(\rho_{\lambda+1}, \ell) \not\models \Psi'$. ◀

Note that $\mathsf{Even}^{\mathsf{c}}(p)$ is expressible when every control state always occurs at positions of the same parity (and this explains why this property is easily expressed with the tree semantics where every position corresponds to a unique control state).

Finally we can also notice that the $\mathbf{U}$ modality cannot be expressed with $\mathbf{F}$, $\mathbf{X}$ and quantifications (contrary to what happens in tree semantics). Let $\mathsf{L}_{\mathsf{Q}}(\mathbf{F}, \mathbf{X})$ (resp. $\mathsf{L}(\mathbf{F}, \mathbf{X})$) be the fragment of QLTL (resp. LTL) where the Until modality is replaced by its weak form $\mathbf{F}$. We have:

▶ **Proposition 4.** *With the structure semantics, there is no* $\mathsf{L}_{\mathsf{Q}}(\mathbf{F}, \mathbf{X})$ *formula equivalent to the formula* $a\mathbf{U}b$.

Proof. Consider the set $Q = \{q_0, q_1, q_2\}$ and an integer $k > 0$. Let ρ_k be the finite sequence $(q_0)^k \cdot q_1$ and ρ'_k be the finite sequence $(q_0)^k \cdot q_2$. Consider the labelling function ℓ over Q defined as follows: $\ell(q_0) = \{a\}$, $\ell(q_1) = \{b\}$ and $\ell(q_2) = \emptyset$. Let π_k (resp. π'_k) be the infinite execution $(\rho_k \cdot \rho'_k)^\omega$ (resp. $(\rho'_k \cdot \rho_k)^\omega$). We clearly have $(\pi_k, \ell) \models a\mathbf{U}b$ and $(\pi'_k, \ell) \not\models a\mathbf{U}b$. Now assume that there exists some $\mathsf{L}_{\mathsf{Q}}(\mathbf{F}, \mathbf{X})$ formula Ψ equivalent to $a\mathbf{U}b$. Over the executions π_k and π'_k, such a formula would be equivalent to some formula of the form $\exists P.\psi$ with ψ an $\mathsf{L}(\mathbf{F}, \mathbf{X})$ formula (indeed Proposition 2 does not introduce any $\mathbf{U}$ modality). And then we would have $(\pi_k, \ell) \models \exists P.\psi$ and $(\pi'_k, \ell) \not\models \exists P.\psi$. Now let ℓ' be the extended labelling ℓ' with 8 atomic propositions $p_0, \ldots, p_7$ that label each subset of Q such that $(\pi_k, \ell') \models \psi$. Note that the $\mathsf{L}(\mathbf{F}, \mathbf{X})$ formula ψ does not depend on k: its construction is only based on the number of control states in the structures.

It remains to show that if k is large enough (i.e. larger than the number of nested $\mathbf{X}$ on top of the formula), then we have $(\pi'_k, \ell') \models \psi$, which contradicts the hypothesis. Let $h_{\mathbf{X}}(\varphi)$ be the height of Next modalities on the top of φ. We can prove the following result: for any $\psi \in \mathsf{L}(\mathbf{F}, \mathbf{X})$ such that $h_{\mathbf{X}}(\psi) < k - i$, we have $((\pi_k)_{\geq i}, \ell') \models \psi \;\Leftrightarrow\; ((\pi'_k)_{\geq i}, \ell') \models \psi$. The proof is done by induction over $h_{\mathbf{X}}(\psi)$:

- $h_{\mathbf{X}}(\psi) = 0$: If ψ is an atomic proposition, it is true because $\ell'((\pi_k)(i)) = \ell'((\pi'_k)(i))$ if $i < k$. If $\psi = \mathbf{F}\psi_1$ and $((\pi_k)_{\geq i}, \ell') \models \psi$, then there exists $i' \geq i$ such that $((\pi_k)_{\geq i'}, \ell') \models \psi_1$ and given the definition of executions π_k and π'_k, there exists $i'' \geq i$ such that $((\pi_k)_{\geq i'}, \ell') = ((\pi'_k)_{\geq i''}, \ell')$. The same holds when $((\pi'_k)_{\geq i}, \ell') \models \psi$.
- $h_{\mathbf{X}}(\psi) > 0$: in that case, ψ is a Boolean combination of atomic propositions, $\mathbf{F}$-formulas or $\mathbf{X}$-formulas. The Boolean part is easily handled by a induction over the size of the formula, and the last case is when $\psi = \mathbf{X}\psi_1$. Assume $((\pi_k)_{\geq i}, \ell') \models \mathbf{X}\psi_1$, then $((\pi_k)_{\geq i+1}, \ell') \models \psi_1$, then by i.h. we get $((\pi'_k)_{\geq i+1}, \ell') \models \psi_1$ since $h_{\mathbf{X}}(\psi_1) < k - i - 1$, and then $((\pi'_k)_{\geq i}, \ell') \models \mathbf{X}\psi_1$. The other direction is done in a similar way.

In conclusion, the formula $\exists P.\psi$ obtained from the hypothetical Ψ is fixed, as is $h_{\mathbf{X}}(\psi)$. For any $k > h_{\mathbf{X}}(\psi)$, we know that $((\pi_k), \ell') \models \psi$ if and only if $((\pi'_k), \ell') \models \psi$ and this contradicts the fact that Ψ is equivalent to $a\mathbf{U}b$. ◀

4 Model checking

4.1 The model-checking problem

The model-checking problem consists in verifying that a given formula is satisfied by a given model. In our framework, we can consider several variants of this problem. First, we distinguish between the existential and the universal model-checking problem:

- $\mathsf{MC}_\exists(\mathsf{QLTL})$: Given a KS $\mathcal{K} = \langle Q, R, q_{in}, \ell \rangle$ and a formula $\varphi \in \mathsf{QLTL}$, does there exist a labelled execution $(\rho, \ell) \in \mathsf{Exec}^{\mathsf{lab}}{}_{\mathcal{K}}(q_{in})$ satisfying φ, i.e. $\mathcal{K} \models_\exists \varphi$?
- $\mathsf{MC}_\forall(\mathsf{QLTL})$: Given a KS $\mathcal{K} = \langle Q, R, q_{in}, \ell \rangle$ and a formula $\varphi \in \mathsf{QLTL}$, do all labelled executions $(\rho, \ell) \in \mathsf{Exec}^{\mathsf{lab}}{}_{\mathcal{K}}(q_{in})$ satisfy φ, i.e. $\mathcal{K} \models_\forall \varphi$?

Note that in the statement of these two problems, we assume that the initial labelling associated with the executions is the labelling ℓ of the Kripke structure $\mathcal{K}$. We point out the fact that these two problems are strongly related, indeed we can use for instance an algorithm for $\mathsf{MC}_\exists(\mathsf{QLTL})$ to solve $\mathsf{MC}_\forall(\mathsf{QLTL})$ since all labelled executions $(\rho, \ell) \in \mathsf{Exec}^{\mathsf{lab}}{}_{\mathcal{K}}(q_{in})$ satisfy φ iff there does not exist a labelled execution $(\rho, \ell) \in \mathsf{Exec}^{\mathsf{lab}}{}_{\mathcal{K}}(q_{in})$ satisfying $\neg\varphi$.

▶ **Example 5.** Let $\mathcal{K}$ be the Kripke structure of Figure 1 rooted at q_{in} where atomic propositions are defined next to nodes. Now, we can consider the two following formulas:

- $\Psi_0 = \Phi_{\mathsf{deter}} \Rightarrow \Big(\mathbf{F}\, a \Rightarrow \mathbf{G}\, \neg c\Big)$, and
- $\Psi_1 = \Big(\mathbf{F}\, \mathbf{G}\, c\Big) \Rightarrow \Phi_{\mathsf{deter}}$.

where Φ_{deter} is the formula we defined previously. We can observe that Ψ_0 holds true for every execution starting from q_{in} (if the execution is deterministic, q_2 will be followed by q_4 or always by q_3), but this is not true for Ψ_1 (an execution ending with a loop in q_4 may contain both the edges $q_2 \to q_3$ and $q_2 \to q_4$). Therefore we clearly have $\mathcal{K} \models_\forall \Psi_0$ and $\mathcal{K} \not\models_\forall \Psi_1$.

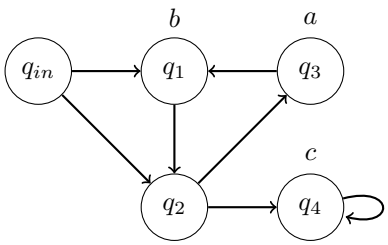


Figure 1 Example of model-checking problem.

4.2 Upper bound for the QLTL model-checking problems

This section is devoted to show the EXPSPACE-membership of $\mathsf{MC}_\exists(\mathsf{QLTL})$ and $\mathsf{MC}_\forall(\mathsf{QLTL})$. For this matter, we rely on alternating Büchi automata. Note that for what concerns the LTL (existential) model-checking, one technique consists in translating an LTL formula into an alternating Büchi automaton, which recognises all the models of the LTL formula, and in performing a cross-product with the Kripke structure to check whether an execution of the structure is accepted by the automaton (see, e.g. [23]). Such a method can as well be used to decide the satisfiability of LTL formulas by checking the emptiness of the associated automata. As the satisfiability problem is undecidable for QLTL [8], such a translation to a class of automata with a decidable emptiness problem will not exist. However we shall see that dealing with the model-checking problem allows us to rely on the Kripke structure to build an alternating Büchi automaton which will recognise the executions satisfying the QLTL formula. We now pursue with the formal explanation.

▶ **Definition 6.** *An* Alternating Büchi Automaton *(ABA) on infinite words is a tuple* $\mathcal{A} = (S, \Sigma, s_{in}, \delta, F)$ *where* S *is a finite set of states,* Σ *is a finite alphabet,* s_{in} *is the initial state,* $\delta : S \times \Sigma \to \mathcal{B}^+(S)$ *is the transition function assigning a positive Boolean formula over* S *including false* ($\bot$) *and true* ($\top$)*, to every pair* (s, σ)*, and* $F \subseteq S$ *is the accepting set of states.*

In order to define how ABAs recognise infinite words over the alphabet Σ, we first need to introduce labelled trees. Given a set Γ, a Γ-labelled tree is a pair (T, σ) where:

- $T \subseteq \mathbb{N}^*$ is a tree, that is, a set of finite words in $\mathbb{N}^*$, called the nodes of T, such that for any $t \in T$ and $n \in \mathbb{N}$, if $t \cdot n \in T$, then $t \in T$ and $t \cdot k \in T$ for any $0 \leq k < n$. The empty word ε represents the root of T. A node $t \in T$ is said to be at depth $i \geq 0$ if its length is equal to i (the root node is hence at depth 0).
- $\sigma : T \mapsto \Gamma$ assigns an element in Γ to every node of T.

Given an automaton $\mathcal{A}$, a run of $\mathcal{A}$ over an infinite word $w = a_0 a_1 \cdots \in \Sigma^\omega$ is a S-labelled tree $\mathcal{T} = (T, \sigma)$ such that: $\sigma(\varepsilon) = s_{in}$ and every node t at depth i (written $|t| = i$) has k ($k \geq 0$) children $t_1, \ldots, t_k$ such that the formula $\delta(\sigma(t), a_i)$ is interpreted to true when one assigns $\top$ to every state in $\{\sigma(t_1), \ldots, \sigma(t_k)\}$ and $\bot$ to other states. Such a run is accepting when every infinite branch of $\mathcal{T}$ contains infinitely often nodes labelled by states in F and every finite branch ends in a node t such that $\delta(\sigma(t), a_{|t|}) = \top$. We use $\mathcal{L}(\mathcal{A})$ to denote the set of words accepted by $\mathcal{A}$. Deciding whether $\mathcal{L}(\mathcal{A}) = \emptyset$ is a PSPACE-complete problem [2].

We will now show how to build an ABA accepting the executions of a Kripke structure satisfying a QLTL formula. Let $\mathcal{K} = \langle Q, R, q_{in}, \ell \rangle$ be a Kripke structure and φ a QLTL formula in negated normal form. We define the ABA $\mathcal{A}_{\varphi,\mathcal{K}} = (S, Q, s_{in}, \delta, F)$ over the alphabet Q with:

- $S = \{s_{in}\} \cup Q \cup \{(\psi, \lambda) \mid \psi \in \mathsf{SubF}(\varphi) \text{ and } \lambda : Q \mapsto 2^{\mathsf{Prop}(\varphi)}\}$;
- F contains the elements (ψ, λ) such that ψ is not of the form $\psi_1 \mathbf{U} \psi_2$;
- and the transition function δ is defined in Figure 2.

Note that the size of S (i.e. $|S|$) is in $O(|Q| + |\varphi| \cdot 2^{|\mathsf{Prop}(\varphi)| \cdot |Q|})$. The intuition behind this construction is that we distinguish two parts in the automaton. First, the states in Q are used to ensure that the accepting word in Q^ω corresponds to some execution in $\mathcal{K}$ (see the definition of $\delta(q, q)$, in equations 2 to 5). The other part (the states (ψ, λ) from 6 to 14) ensures that the subformula ψ holds for true along the forthcoming execution labelled with λ. The key to the correct translation to solve the model-checking problem is that, in this ABA, the same word over Q is recognised along every branch of the execution tree. The equation 1 ensures that the two parts of the automaton are visited to check whether the word corresponds to a correct execution from the structure and that it satisfies the formula.

To state the correctness of the construction, we will use $\mathcal{A}_{\varphi,\mathcal{K}}[s]$ to denote the automaton where s is used as the initial state. We have then:

▶ **Lemma 7.** *Let* ψ *be a subformula of* φ*,* q *be a control state,* ρ *be an infinite sequence in* Q^ω *and* $\lambda : Q \mapsto 2^{\mathsf{Prop}(\varphi)}$ *be a labelling function (defined on the free variables in* ψ*). The two following properties hold:*

1. $\rho \in \mathcal{L}(\mathcal{A}_{\varphi,\mathcal{K}}[q]) \quad \Leftrightarrow \quad \rho \in \mathsf{Exec}_{\mathcal{K}}(q)$,
2. $\rho \in \mathcal{L}(\mathcal{A}_{\varphi,\mathcal{K}}[(\psi, \lambda)]) \quad \Leftrightarrow \quad (\rho, \lambda) \models \psi$.

Proof. The first result comes directly from the definition of $\delta(q, -)$ when $q \in Q$.
The second point is proved by induction over ψ. We only focus on the main cases and note that we omit argument when the reverse direction is similar to the described one:

- $\psi = p$: If $\rho \in \mathcal{L}(\mathcal{A}_{\varphi,\mathcal{K}}[(p, \lambda])$, then by def. $\delta((p, \lambda), \rho(0)) = \top$ and we have $p \in \lambda(\rho(0))$ and then $(\rho, \lambda) \models p$. The reverse direction is similar.

$$\delta(s_{in}, q) = \delta((\varphi, \ell), q) \wedge \delta(q_{in}, q) \tag{1}$$

$$\delta(q, q) = \bigvee_{(q,q') \in R} q' \tag{2}$$

$$\delta(q, q') = \bot \quad \text{if } q \neq q' \tag{3}$$

$$\delta((\top, \lambda), q) = \top \tag{4}$$

$$\delta((\bot, \lambda), q) = \bot \tag{5}$$

$$\delta((p, \lambda), q) = \begin{cases} \top & \text{if } p \in \lambda(q) \\ \bot & \text{otherwise} \end{cases} \tag{6}$$

$$\delta((\neg p, \lambda), q) = \begin{cases} \bot & \text{if } p \in \lambda(q) \\ \top & \text{otherwise} \end{cases} \tag{7}$$

$$\delta((\psi_1 \vee \psi_2, \lambda), q) = \delta((\psi_1, \lambda), q) \vee \delta((\psi_2, \lambda), q) \tag{8}$$

$$\delta((\psi_1 \wedge \psi_2, \lambda), q) = \delta((\psi_1, \lambda), q) \wedge \delta((\psi_2, \lambda), q) \tag{9}$$

$$\delta((\mathbf{X}\psi, \lambda), q) = (\psi, \lambda) \tag{10}$$

$$\delta((\psi_1 \mathbf{U} \psi_2, \lambda), q) = \delta((\psi_2, \lambda), q) \vee (\delta((\psi_1, \lambda), q) \wedge (\psi_1 \mathbf{U} \psi_2, \lambda)) \tag{11}$$

$$\delta((\psi_1 \mathbf{R} \psi_2, \lambda), q) = \delta((\psi_2, \lambda), q) \wedge (\delta((\psi_1, \lambda), q) \vee (\psi_1 \mathbf{R} \psi_2, \lambda)) \tag{12}$$

$$\delta((\exists p.\psi, \lambda), q) = \bigvee_{P \subseteq Q} \delta((\psi, \lambda[p \rightsquigarrow P]), q) \tag{13}$$

$$\delta((\forall p.\psi, \lambda), q) = \bigwedge_{P \subseteq Q} \delta((\psi, \lambda[p \rightsquigarrow P]), q) \tag{14}$$

where $\lambda[p \rightsquigarrow P]$ denotes the labelling function λ' defined by: $\lambda'(q) = \lambda(q) \cup \{p\}$ if $q \in P$ and $\lambda'(q) = \lambda(q)$ otherwise.

Figure 2 Definition of δ.

- $\psi = \psi_1 \wedge \psi_2$: If $\rho \in \mathcal{L}(\mathcal{A}_{\varphi,\mathcal{K}}[(\psi_1 \wedge \psi_2, \lambda)]$, then by def. of δ we have $\rho \in \mathcal{L}(\mathcal{A}_{\varphi,\mathcal{K}}[(\psi_1, \lambda)])$ and $\rho \in \mathcal{L}(\mathcal{A}_{\varphi,\mathcal{K}}[(\psi_2, \lambda)])$. By ind. hyp. we can deduce $(\rho, \lambda) \models \psi_1$ and $(\rho, \lambda) \models \psi_2$, and therefore $(\rho, \lambda) \models \psi$. The reverse direction is similar.
- $\psi = \mathbf{X}\psi_1$: If $\rho \in \mathcal{L}(\mathcal{A}_{\varphi,\mathcal{K}}[(\mathbf{X}\psi_1, \lambda)])$, then by def. of δ we have $\rho_{\geq 1} \in \mathcal{L}(\mathcal{A}_{\varphi,\mathcal{K}}[(\psi_1, \lambda)]$ and by i.h. we have $(\rho_{\geq 1}, \lambda) \models \psi_1$, from which we get $(\rho, \lambda) \models \mathbf{X}\psi_1$ by the semantics of $\mathbf{X}$.
- $\psi = \psi_1 \mathbf{U} \psi_2$: If $\rho \in \mathcal{L}(\mathcal{A}_{\varphi,\mathcal{K}}[(\psi, \lambda)])$, then by def. of δ we have either (1) $\delta((\psi_2, \lambda), \rho(0))$ holds true for the run (that is $\rho \in \mathcal{L}(\mathcal{A}_{\varphi,\mathcal{K}}[(\psi_2, \lambda)])$ or equivalently $(\rho, \lambda) \models \psi_2$), or $\delta((\psi_1, \lambda), \rho(0))$ holds true and one successor node (in the execution tree) is recognised by the state $(\psi_1 \mathbf{U} \psi_2, \lambda)$, and so on. As any infinite branch cannot be labelled infinitely often by some $(\psi_1 \mathbf{U} \psi_2, -)$-state, this stops at some level k with the satisfaction of $\delta((\psi_2, \lambda), \rho(0))$. From this we can deduce by i.h. that $(\rho_{\geq k}, \lambda) \models \psi_2$ and $(\rho_{\geq i}, \lambda) \models \psi_1$ for $0 \leq i < k$. This is precisely the definition of $\psi_1 \mathbf{U} \psi_2$.
  Now assume $(\rho, \lambda) \models \psi_1 \mathbf{U} \psi_2$. By definition, there exists $k \geq 0$ s.t. $(\rho_{\geq k}, \lambda) \models \psi_2$ and $(\rho_{\geq i}, \lambda) \models \psi_1$ for $0 \leq i < k$. By i.h. we get $\rho_{\geq k} \in \mathcal{L}(\mathcal{A}_{\varphi,\mathcal{K}}[(\psi_2, \lambda)])$ and $\rho_{\geq i} \in \mathcal{L}(\mathcal{A}_{\varphi,\mathcal{K}}[(\psi_1, \lambda])$ for $0 \leq i < k$. We can deduce that $\rho \in \mathcal{L}(\mathcal{A}_{\varphi,\mathcal{K}}[(\psi_1 \mathbf{U} \psi_2, \lambda)])$.
- $\psi = \exists p.\psi_1$. If $\rho \in \mathcal{L}(\mathcal{A}_{\varphi,\mathcal{K}}[(\psi, \lambda)])$, then by def. of δ, there exists some $P \subseteq Q$ s.t. $\delta((\psi_1, \lambda[p \rightsquigarrow P]), q)$ holds true for the run. By i.h. we get $(\rho, \lambda[p \rightsquigarrow P]) \models \psi_1$, which implies $(\rho, \lambda) \models \exists p.\psi_1$. ◀

Note that if we apply the previous lemma with the formula φ and the labelling ℓ of $\mathcal{K}$ and since $\delta(s_{in}, q) = \delta((\varphi, \ell), q) \wedge \delta(q_{in}, q)$ for all $q \in Q$, we deduce that $\mathcal{L}(\mathcal{A}_{\varphi,\mathcal{K}}) \neq \emptyset$ if and only if there exists a labelled execution $(\rho, \ell) \in \mathsf{Exec}^{\mathsf{lab}}{}_{\mathcal{K}}(q_{in})$ satisfying φ. Furthermore since the number of states of $\mathcal{A}_{\varphi,\mathcal{K}}$ is in $O(|Q| + |\varphi| \cdot 2^{|\mathsf{Prop}(\varphi)|\cdot|Q|})$ and the emptiness problem for ABA is PSPACE-complete [2], we deduce the following result.

▶ **Proposition 8.** $\mathsf{MC}_\exists(\mathsf{QLTL})$ *and* $\mathsf{MC}_\forall(\mathsf{QLTL})$ *are in EXPSPACE.*

4.3 Lower bound for the QLTL model-checking problems

To prove the complexity lower bound (EXPSPACE-hardness), we use a domino tiling problem that we shall now define. Let C be a finite set of colours. A tile's type is then a tuple $(c_{down}, c_{left}, c_{up}, c_{right})$ in C^4. Let T be a finite set of *tile's type*. Given two integers a and b, a T-tiling function for the $a \times b$-grid (with a rows and b columns) is a function $f : [0, a-1] \times [0, b-1] \to T$ such that for all $0 \leq i < a$ and $0 \leq j < b$, we have: (1) $f(i,j)_{up} = f(i+1,j)_{down}$ if $i < a-1$, and (2) $f(i,j)_{right} = f(i,j+1)_{left}$ if $j < b-1$. As a matter of fact, the tiling function ensures that adjacent tiles share the same colour.

When such a function exists, we say that the grid can be tiled. We now define the following tiling decision problem $\mathcal{T}$:
Input: a set of colours C, a set of tile's types T, an integer m (encoded in unary) and $t_{init}, t_{final} \in T$
Output: yes iff there exists an integer n and a T-tiling function f for the $n \times 2^m$-grid such that $f(0,0) = t_{init}$ and $f(n-1, 2^m - 1) = t_{final}$.
This problem is EXPSPACE-complete (see e.g. [20]) and has already been used to prove complexity lower bound for variant of LTL as in [15]. We adapt here the reduction proposed in this latter work to our context.

▶ **Proposition 9.** $\mathsf{MC}_\exists(\mathsf{QLTL})$ *and* $\mathsf{MC}_\forall(\mathsf{QLTL})$ *are EXPSPACE-hard.*

Proof. Let $\mathcal{T} = (C, T, t_{init}, t_{final}, m)$ be an instance of the tiling problem with $T = \{t_1, \ldots, t_p\}$. We build a Kripke structure $\mathcal{K}_\mathcal{T}$ and a QLTL formula $\Phi_\mathcal{T}$ such that there exist a labelled execution $(\rho, \ell) \in \mathsf{Exec}^{\mathsf{lab}}{}_{\mathcal{K}_\mathcal{T}}(q_{in})$ satisfying φ if and only if there is some n such that the $n \times 2^m$-grid can be tiled. The path witnessing the existence of the tiling function for the grid $n \times 2^m$ is of the form $q_{in} \cdot \Big(((b_1 \cdots b_m) \cdot t)^{2^m}\Big)^n \cdot E^\omega$ (with $b_j \in \{b_j^-, b_j^+\}$ for all $j \in \{1, \ldots, m\}$ and $t \in \{t_1, \ldots, t_p\}$). This word represents the sequence of the lines of the grid. The i-th line is listed from the cell $(i, 0)$ to cell $(i, 2^m - 1)$: each cell description starts with a sequence of m bits which encodes the cell's column number, followed by the type of tile associated with it. Such a path clearly belongs to the generic Kripke structure depicted in Figure 3 where we suppose that control states and their associated labels are the same.

We now build a QLTL formula $\Phi_\mathcal{T}$ to specify what is a "tiling function" path in $\mathcal{K}_\mathcal{T}$. First we use $b_j^\pm$ as a shorthand for $b_j^+ \vee b_j^-$. The formula $\Phi_\mathcal{T}$ is a conjunction of several properties:

- The tile t_{init} (resp. t_{final}) occurs at the first (resp. last) cell:

$$\Phi_1 \;=\; \mathbf{X}\Big[\Big(\bigwedge_{j=1}^{m} \mathbf{X}^{j-1}\, b_j^-\Big) \wedge \mathbf{X}^m\, t_{init}\Big] \wedge \mathbf{F}\,\Big[\Big(\bigwedge_{j=1}^{m} \mathbf{X}^{j-1} b_j^+\Big) \wedge \mathbf{X}^m\,(t_{final} \wedge \mathbf{X}E)\Big)\Big]$$

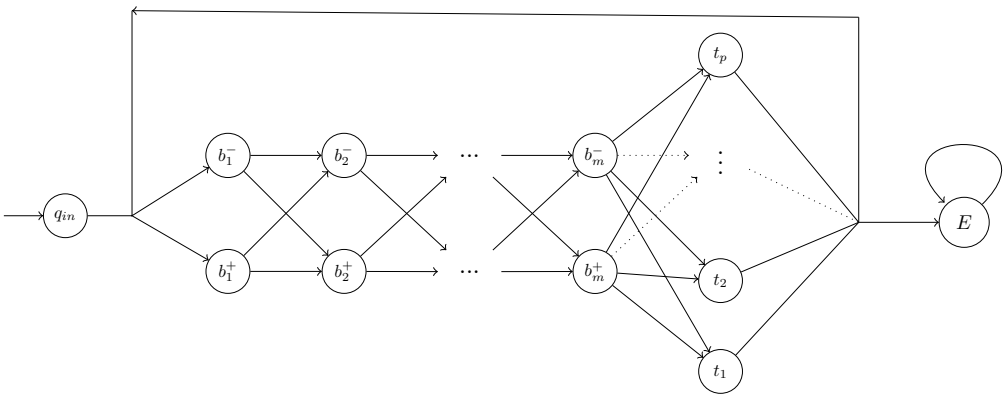


Figure 3 Kripke structure $\mathcal{K}_\mathcal{T}$ for the tiling problem.

- The ordering of the cells is correct (w.r.t. the counter encoded by the b_js):

$$\Phi_2 = \mathbf{G}\Big[\Big(b_1^{\pm} \wedge \neg \mathbf{X}^{m+1} E\Big) \Rightarrow \Big(b_1^+ \Leftrightarrow \mathbf{X}^{m+1} b_1^-\Big)\Big] \\ \wedge \mathbf{G} \bigwedge_{j=1}^{m-1} \Big[\Big(b_j^+ \wedge \mathbf{X}^{m+1} b_j^-\Big) \Leftrightarrow \mathbf{X}\Big(b_{j+1}^+ \Leftrightarrow \mathbf{X}^{m+1} b_{j+1}^-\Big)\Big]$$

  The first part of Φ_2 ensures that the b_1^- and b_1^+ alternate when considering two successive cells (except at the end of the execution before looping forever at E). The second part ensures that the sign of b_{j+1} changes between two successive cells if and only if the bit b_j goes from $\top$ (1) to $\bot$ (0).
- The successive cells along a *row* have to agree on the colour of the shared side (that is $f(i,j)_{right} = f(i,j+1)_{left}$ for $0 \leq j < 2^m - 1$):

$$\Phi_3 = \bigwedge_{t \in T} \mathbf{G}\left[t \Rightarrow \Big(\bigvee_{\substack{t' \in T \text{ s.t.} \\ t_{right} = t'_{left}}} (\mathbf{X}^{m+1} t') \vee \mathbf{X} E \vee (\bigwedge_{j=1}^{m} \mathbf{X}^j b_j^-)\Big)\right]$$

- The successive cells along a *column* have to agree on the colour of the shared side (that is $f(i,j)_{up} = f(i+1,j)_{down}$ for $0 \leq i < n-1$). This is the difficult part because the two cells are separated by an exponential number of states. To specify this property, we need the quantification of atomic propositions: we can store the number of the cell (*i.e.* the values of the b_js) and access the next cell with this number. For this, we define two shorthand, first we have:

$$\mathsf{StoreNb}(p) = (\bigwedge_{j=1}^{m} \mathbf{X}^{j-1}\, p) \wedge \mathbf{G}(p \Rightarrow \bigvee_{j=1}^{m} b_j^{\pm}) \wedge \bigwedge_{j=1}^{m} \mathbf{F}((b_j^+ \wedge \neg p) \vee (b_j^- \wedge \neg p))$$

  $\mathsf{StoreNb}$ labels m consecutive states by p and ensures that only b_js states can be labelled and that either b_j^+ or b_j^- is not labelled by p: therefore the p-labelling describes exactly the number of the column of the current cell (assuming that the formula is evaluated at the beginning of the cell, that is over b_1^+ or b_1^-). And now we can use $\mathsf{Nb}(p) = (\bigwedge_{j=1}^{m} \mathbf{X}^j\, p)$ to locate the cells of the row "p".

We can now state the property ensuring that two successive cells on the same column share the same colour on the common side:

$$\Phi_4 \;=\; \bigwedge_{t\in T} \mathbf{G}\Big[\Big(b_1^{\pm} \wedge \mathbf{X}^m t\Big) \;\Rightarrow\; \Big(\mathbf{XG}\,(\neg \bigwedge_{j=1}^{m-1} \mathbf{X}^j b_j^-) \vee \exists p.\big[\mathsf{StoreNb}(p) \wedge\; \mathbf{X}\big(\,(\neg\mathsf{Nb}(p))\;\mathbf{U}\;(\mathsf{Nb}(p) \wedge \bigvee_{\substack{t'\in T \text{ s.t.}\\ t_{up}=t'_{down}}} \mathbf{X}^{m+1}t')\big)\big]\Big)\Big]$$

The subformula $\mathbf{XG}(\neg \bigwedge_{j=1}^{m-1} \mathbf{X}^j b_j^-)$ is used to exclude the last column where there is no correspondence to ensure. The $\mathbf{U}$ operator allows us to select exactly the next cell of the column p.

Finally we have: $\Phi_{\mathcal{T}} = \Phi_1 \wedge \Phi_2 \wedge \Phi_3 \wedge \Phi_4$. And we can easily conclude that $\mathcal{T}$ is a positive instance of the tiling problem iff there exists an execution $(\rho,\ell) \in \mathsf{Exec}^{\mathsf{lab}}{}_{\mathcal{K}_{\mathcal{T}}}(q_{in})$ such that $(\rho,\ell) \models \Phi_{\mathcal{T}}$. ◀

From Propositions 8 and 9, we can deduce the main theorem about QLTL model-checking.

▶ **Theorem 10.** $\mathsf{MC}_\exists(\mathsf{QLTL})$ *and* $\mathsf{MC}_\forall(\mathsf{QLTL})$ *are EXPSPACE-complete.*

4.4 Relation with QCTL*

As CTL^* which extends both CTL and LTL, we can consider the logic QCTL^*. Two variants of QCTL^* have been defined in literature: in the first one, called FCTL^* (see [7]), the formula $\exists p.\varphi$ is defined as a *path* formula (as $\mathbf{U}$ or $\mathbf{X}$) and in the second variant, called QCTL^* (see [14]), it is defined as a *state* formula. For the tree semantics, both logics are equally expressive, but with the structure semantics, there is a big difference in terms of expressivity (FCTL^* may express the existence of an eulerian path which is not possible with QCTL^*) and while the model-checking problem is PSPACE-complete for QCTL^*, for FCTL^* the model-checking algorithm is based on a reduction to the tree semantics and its complexity is in k-EXPTIME where k is the number of alternations of quantifications. Clearly the logic QLTL we consider here can be seen as a fragment of FCTL^* but it is not included in QCTL^*: the expressiveness of the two logics are different and none of the two is strictly more expressive than the other. However, it is worth noting that selecting an execution first and then looking for a labelling, as for QLTL, induces a complexity blow-up.

However we can still use the result about QCTL^* model-checking to define a last verification problem for Prenex formulas $\mathcal{Q}.\psi$ with $\psi \in \mathsf{LTL}$, where we look for labellings of the *full* structure, and *then* select paths in the structure. Such an approach corresponds to a model-checking instance for QCTL^*. Formally we define these verification problems (denoted $\mathsf{MC}_\forall(\mathsf{q\text{-}LTL})$ or $\mathsf{MC}_\exists(\mathsf{q\text{-}LTL})$) as follows: given a structure $\mathcal{K} = \langle Q, q_0, R, \ell\rangle$ and a formula $\mathcal{Q}.\psi$ with $\psi \in \mathsf{LTL}$ and $\mathcal{Q}$ a block of quantifications, decide whether $\langle Q, q_0, R, \ell\rangle \models_\forall \mathcal{Q}.\psi$ with:

$$\begin{aligned}
\langle Q, q_0, R, \ell\rangle \models_\forall \exists p.\mathcal{Q}.\psi \quad &\Leftrightarrow \quad \exists Q' \subseteq Q \text{ s.t. } \langle Q, q_0, R, \ell[p \mapsto Q']\rangle \models_\forall \mathcal{Q}.\psi\\
\langle Q, q_0, R, \ell\rangle \models_\forall \forall p.\mathcal{Q}.\psi \quad &\Leftrightarrow \quad \forall Q' \subseteq Q, \text{ we have } \langle Q, q_0, R, \ell[p \mapsto Q']\rangle \models_\forall \mathcal{Q}.\psi
\end{aligned}$$

and where $\langle Q, q_0, R, \ell\rangle \models_\forall \psi$ with $\psi \in \mathsf{LTL}$ is interpreted as usual. In the same way, we can define the existential variant $\mathsf{MC}_\exists(\mathsf{q\text{-}LTL})$. Clearly $\mathsf{MC}_\forall(\mathsf{q\text{-}LTL})$ and $\mathsf{MC}_\exists(\mathsf{q\text{-}LTL})$ are PSPACE-complete (PSPACE-hard due to QBF, and PSPACE-easy due to QCTL^*).

This defines an interesting class of problems that are different from the ones we introduced before for QLTL and whose complexity is better. For example, if we consider a two-player turn-based game $\mathcal{G} = (Q_1, Q_2, q_0, R_\mathcal{G}, F_1, F_2)$ where Q_1 (resp. Q_2) are the states of Player 1 (resp. Player 2), $q_0 \in Q_1 \cup Q_2$ is the initial state and $R_\mathcal{G} \subseteq (Q_1 \cup Q_2) \times \{0,1\} \times (Q_1 \cup Q_2)$ is the transition relation (NB: we assume that in every state of Player i, there are exactly two possible moves labelled by 0 and 1, and we label every transition by the number of the move it corresponds to) and F_1 (resp. F_2) is the set of winning positions of Player 1 (resp. Player 2). The existence of a memoryless strategy for Player 1 (or 2) can easily be reduced to a model-checking problem of $\mathsf{MC}_\forall(\mathsf{q\text{-}LTL})$ by considering the following Kripke structure $\mathcal{K}_\mathcal{G} = \langle Q, q_0, R, \ell \rangle$:

- $Q = Q_1 \cup Q_2 \cup \{(q, q', \varepsilon) \mid q \in Q_1 \text{ and } (q, \varepsilon, q') \in R_\mathcal{G}\}$;
- $R = \{(q, q') \mid q \in Q_2 \text{ and } (q, -, q') \in R\} \cup \{(q, (q, \varepsilon, q')) \mid q \in Q_1 \text{ and } (q, \varepsilon, q') \in R\} \cup \{((q, \varepsilon, q'), q') \mid (q, \varepsilon, q') \in Q\}$;
- ℓ labels the structure $\mathcal{K}_\mathcal{G}$ as follows: states in Q_1 (resp. Q_2) are labelled with P_1 (resp. P_2), states in F_1 (resp. F_2) are labelled with W_1 (resp. W_2). And intermediary states (q, ε, q') are labelled by C_ε (in order to specify which move is currently played by Player 1).

Clearly a memoryless strategy for Player 1 consists in marking every Q_1 states by the move (0 or 1) corresponding to the strategy. Here with only two allowed moves, it suffices to use a single atomic proposition c. Therefore the existence of a memoryless strategy can be expressed with the following formula:

$$\Psi_{\text{strat}} = \exists c. \Big[\mathbf{G} \Big((P_1 \wedge c) \Rightarrow (\mathbf{X}\, C_1) \ \wedge\ (P_1 \wedge \neg c) \Rightarrow (\mathbf{X}\, C_0) \Big) \ \Rightarrow\ \mathbf{F}\, W_1 \Big]$$

5 Model Checking Paths and Flat Structures

We present here some restrictions on the considered structures which allow to obtain better complexity bounds for the model checking of QLTL formulas. We first consider ultimately periodic paths and use the facts that the model-checking problem for the branching logic QCTL (with the structure semantics) is PSPACE-complete [14] and that morally a QLTL formula over a path can be translated into a QCTL formula (in a path there is indeed no branching). We then study the model-checking problem for QLTL restricted to Kripke structures with no nested loop and show it is as well PSPACE-complete. To obtain the upper bound, we follow the same reasoning as the one presented in [4] to show that the model-checking problem for LTL with Past is in NP. It relies on two aspects: a stuttering theorem for QLTL and the fact that we can represent finitely all the executions of a flat structure with what we call iterated path schemas.

5.1 Path Model Checking

We first consider path as it is done in [17] for LTL. Given a set of states Q, a *labelled path* is an ultimately-periodic structure $(\rho_1 \cdot \rho_2^\omega, \ell)$ with $\rho_1 \in Q^*, \rho_2 \in Q^+$ and $\ell : Q \to 2^{\mathsf{AP}}$ is a labelling function. The size of such a structure is given by the sum of the length of the sequences ρ_1 and ρ_2. We use $\mathsf{MC_p}(\mathsf{QLTL})$ to denote this model-checking problem which takes as input a labelled path $(\rho_1 \cdot \rho_2^\omega, \ell)$ and a formula $\varphi \in \mathsf{QLTL}$ and which asks whether $(\rho_1 \cdot \rho_2^\omega, \ell) \models \varphi$. Note that in a path the same control state might appear more than one time in ρ_1 and ρ_2. We have then the following result:

▶ **Theorem 11.** $\mathsf{MC_p}(\mathsf{QLTL})$ *is PSPACE-complete.*

Proof. PSPACE-hardness comes from the QBF problem: a QBF instance Φ belongs to QLTL, and its validity can be directly reduced to some path model-checking problem $(q^\omega, \ell_\emptyset) \models \Phi$. The PSPACE-membership comes from the PSPACE-membership of the model-checking problem for QCTL in structure semantics [14]: when considering a single path, the QLTL formula φ has the same truth value as the QCTL formula $\widehat{\varphi}$ with every temporal modality $\mathbf{X}$, $\mathbf{U}$ or $\mathbf{R}$ associated with some existential or universal path quantifier. This provides the result. ◀

5.2 Flat Kripke Structures and Path Schemas

A Kripke structure is said to be *flat* (sometimes the term *weak* is used, see e.g. [12]) if every node in the underlying graph belongs to at most one simple cycle (a simple cycle is a cycle where each edge appears at most once) [3]. Figure 4 presents an example of a flat Kripke structure. Note that if we add an edge for instance from state q_4 to q_2 then the structure will not be flat anymore as the control states q_2, q_3 and q_4 will belong to two simple cycles. It is worth noticing that path model-checking is not subsumed by flat structure model-checking (and of course, neither does the opposite) because in a path there is no restriction on the occurrences of a state, whereas in a flat structure each occurrence of a state (except the last one) is necessarily followed by the same state.

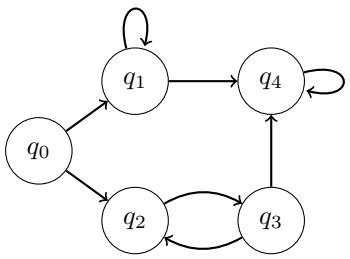


Figure 4 A flat Kripke structure.

We then denote by $\mathsf{MC}_{\mathsf{f},\exists}(\mathsf{QLTL})$ (resp. $\mathsf{MC}_{\mathsf{f},\forall}(\mathsf{QLTL})$) the existential (resp. universal) model-checking problem $\mathsf{MC}_{\exists}(\mathsf{QLTL})$ (resp. $\mathsf{MC}_{\forall}(\mathsf{QLTL})$) restricted to flat Kripke structures. As for the general case, our method to solve $\mathsf{MC}_{\mathsf{f},\exists}(\mathsf{QLTL})$ can be used (with the same complexity bound) to solve $\mathsf{MC}_{\mathsf{f},\forall}(\mathsf{QLTL})$ by taking the negation of the formula.

Flat Kripke structures are easier to analyse than general structures, because we can represent their executions thanks to a finite set of path schemas (of polynomial size). Let $\mathcal{K} = \langle Q, R, q_{in}, \ell \rangle$ be a flat Kripke structures. A path p in $\mathcal{K}$ is a finite (non-empty) sequence of control states $q_0, \ldots, q_k \in Q^+$ such that $(q_i, q_{i+1}) \in R$ for all $i \in [0, k-1]$. We denote by $\mathit{first}(p)$ the first control state q_0 of the sequence and $\mathit{last}(p)$ the last one equals to q_k. A loop is then a path p such that $\mathit{first}(p) = \mathit{last}(p)$. A path schema P is an expression of the form $p_1 l_1 p_2 l_2 \ldots p_k l_k$ such that :

- p_i is a path for all $i \in [1, k]$;
- l_i is a loop for all $i \in [1, k]$;
- $\mathit{first}(p_1) = q_{in}$ and $\mathit{first}(l_i) = \mathit{last}(p_i) = \mathit{first}(p_{i+1})$ for all $i \in [1, k-1]$ and $\mathit{first}(l_k) = \mathit{last}(p_k)$.

The size of a path schema $P = p_1 l_1 p_2 l_2 \dots p_k l_k$ is the sum of the lengths of each sequence composing it. We say that an execution $\rho \in Q^\omega$ respects a path schema $P = p_1 l_1 p_2 l_2 \dots p_k l_k$ iff there exists $n_1, \dots, n_{k-1} \in \mathbb{N} \setminus \{0\}$ such that $\rho = p_1 l_1^{n_1} p_2 l_2^{n_2} \dots p_{k-1} l_{k-1}^{n_{k-1}} p_k l_k^\omega$. Finally, for all $n_1, \dots, n_{k-1} \in \mathbb{N} \setminus \{0\}$, by definition of path schemas, we have that $p_1 l_1^{n_1} p_2 l_2^{n_2} \dots p_{k-1} l_{k-1}^{n_{k-1}} p_k l_k^\omega$ is an execution. From Section 3 of [4] we have the following proposition[2]:

▶ **Proposition 12** ([4]). *In a flat Kripke structure, for each execution ρ, there exists a path schema P of size smaller than $3 * |Q|$ such that ρ respects P.*

5.3 Stuttering Result for QLTL

In [4], the authors have shown a general stuttering theorem for LTL with past, they have proved that if an execution of the form $\rho_1 s^M \rho_2$ (where ρ_1 and s are finite sequence of states) satisfies an LTL formula φ of temporal height at most N and if $M > 2N$, then the execution $\rho_1 s^{2N+1} \rho_2$ satisfies φ as well. In other words, to satisfy φ there is no need to repeat the infix s more than $2N + 1$ times. We shall see that we have the same result for QLTL.

Let $\mathcal{K} = \langle Q, R, q_0, \ell \rangle$ be a Kripke structure (not necessarily flat) and assume that we have two executions $\rho = \rho_1 s^M \rho_2$ and $\rho' = \rho_1 s^{M'} \rho_2$ with $\rho_1, s \in Q^*$ and $\rho_2 \in Q^\omega$ and $M, M' > 2N$ for some $N \geq 2$. In Section 4 of [4], the authors present an equivalence relation (parametrised by N) between positions in ρ and ρ' which can be defined as for $i, i' \in \mathbb{N}$, we have $(\rho, i) \equiv_N (\rho', i')$ if and only if one of the following conditions holds:

1. $i, i' < |\rho_1| + N \cdot |s|$ and $i = i'$
2. $i \geq |\rho_1| + (M - N) \cdot |s|$ and $i' \geq |\rho_1| + (M' - N) \cdot |s|$ and $(i - i') = (M - M') \cdot |s|$
3. $|\rho_1| + N \cdot |s| \leq i < |\rho_1| + (M - N) \cdot |s|$ and $|\rho_1| + N \cdot |s| \leq i' < |\rho_1| + (M' - N) \cdot |s|$ and $|i - i'| = 0 \mod |s|$

Intuitively, this relation states that either i and i' should be at the same position in the parts consisting of ρ_1 and the first N copies of s and in the same relative positions in the last N copies of s and in ρ_2, otherwise i and i' should be at the same position in s. We can show following the exact same steps as for the proof of Theorem 4.1 of [4], that if φ is a QLTL formula such that $\mathsf{th}(\varphi) \leq N$ and if $(\rho, i) \equiv_N (\rho', i')$ then for all labellings λ, we have $(\rho_{\geq i}, \lambda) \models \varphi$ iff $(\rho'_{\geq i'}, \lambda) \models \varphi$. This proof is done by a double induction on the structure of φ and on N, and we should pay attention to two aspects. First, for LTL there is no need to consider labelling, however here we take them into account, but since the sequence of control states is the same (modulo the iterations of s) in ρ and ρ', we can universally quantify on labellings (for each labelling we get two new sequences of subset of atomic propositions, and we use the fact that results for LTL hold for such sequences). Second, in order to reuse the induction reasoning, we should take care of the specific case of subformula $\exists p.\ \psi$. Assume hence that we have $(\rho, i) \equiv_{N'} (\rho', i')$ and that for ψ such that $\mathsf{th}(\psi) \leq N'$, we have $(\rho_{\geq i}, \lambda') \models \psi$ iff $(\rho'_{\geq i'}, \lambda') \models \psi$ for all labelling λ'. Now for any labelling λ, we have that there exists $\lambda' \equiv_{\mathsf{AP} \setminus \{p\}} \lambda$ s.t. $(\rho_{\geq i}, \lambda') \models \psi$ if and only if there exists $\lambda'' \equiv_{\mathsf{AP} \setminus \{p\}} \lambda$ s.t. $(\rho'_{\geq i'}, \lambda'') \models \psi$ (simply take $\lambda' = \lambda''$). And we can hence conclude, since the temporal height of $\exists p.\ \psi$ is the same as ψ, that for all labellings λ, we have $(\rho_{\geq i}, \lambda) \models \exists p.\ \psi$ iff $(\rho'_{\geq i'}, \lambda) \models \exists p.\ \psi$. Finally, since $(\rho, 0) \equiv_N (\rho', 0)$ for any $N \geq 2$, we can adapt Theorem 4.1 of [4] to our case.

[2] In [4], the size of path schema is bounded by $2 * |R|$, since we consider here sequence of control states, we use 3 as a constant to stay on the safe side.

▶ **Proposition 13.** *Let $N \geq 2$ and $M, M' > 2N$ and $\rho = \rho_1 s^M \rho_2$ and $\rho' = \rho_1 s^{M'} \rho_2$ with $\rho_1, s \in Q^*$ and $\rho_2 \in Q^\omega$. For all* QLTL *formulas φ such that* $\mathsf{th}(\varphi) \leq N$*, we have $(\rho, \ell) \models \varphi$ iff $(\rho', \ell) \models \varphi$.*

5.4 Algorithm for Flat Kripke Structures

We now present the algorithm to solve $\mathsf{MC}_{\mathsf{f},\exists}(\mathsf{QLTL})$. Let $\mathcal{K} = \langle Q, R, q_{in}, \ell \rangle$ be a flat Kripke structure and φ a QLTL formula such that $\mathsf{th}(\varphi) \leq N$. Assume there exists a labelled execution $(\rho, \ell) \in \mathsf{Exec}^{\mathsf{lab}}_{\mathcal{K}}(q_0)$ satisfying φ. Using Proposition 12, there exists a path schema $P = p_1 l_1 p_2 l_2 \dots p_k l_k$ (of size smaller than $3 * |Q|$) and $n_1, \dots, n_{k-1} \in \mathbb{N} \setminus \{0\}$ such that $\rho = p_1 l_1^{n_1} p_2 l_2^{n_2} \dots p_{k-1} l_{k-1}^{n_{k-1}} p_k l_k^\omega$. Now for all $i \in \{1, \dots, k-1\}$, we define $n'_i = min(n_i, 2N+5)$ and let ρ' be the execution $p_1 l_1^{n'_1} p_2 l_2^{n'_2} \dots p_{k-1} l_{k-1}^{n'_{k-1}} p_k l_k^\omega$, thanks to Proposition 13, we get that $(\rho', \ell) \models \varphi$. This gives us the path for a non-deterministic PSPACE-algorithm. We seek for a path schema $P = p_1 l_1 p_2 l_2 \dots p_k l_k$ and for $(k-1)$ positive naturals $n'_1, \dots, n'_{k-1}$ smaller than $2N+5$ such that $(p_1 l_1^{n'_1} p_2 l_2^{n'_2} \dots p_{k-1} l_{k-1}^{n'_{k-1}} p_k l_k^\omega, \ell) \models \varphi$. Note that $p_1 l_1^{n'_1} p_2 l_2^{n'_2} \dots p_{k-1} l_{k-1}^{n'_{k-1}} p_k l_k^\omega$ is of polynomial size in the size of the flat Kripke structure $\mathcal{K}$ and the formula φ and that checking whether $(p_1 l_1^{n'_1} p_2 l_2^{n'_2} \dots p_{k-1} l_{k-1}^{n'_{k-1}} p_k l_k^\omega, \ell) \models \varphi$ can be done in polynomial space thanks to Theorem 11. We use then Savitch's theorem to obtain a PSPACE-algorithm. For the lower bound, the proof is the same as for the path model-checking.

▶ **Theorem 14.** $\mathsf{MC}_{\mathsf{f},\exists}(\mathsf{QLTL})$ *and* $\mathsf{MC}_{\mathsf{f},\forall}(\mathsf{QLTL})$ *are PSPACE-complete.*

6 Conclusion

We studied the model-checking problem for QLTL for the structure semantics. (In this semantics, executions are seen as an infinite sequence of control states, together with a labelling function that associates atomic propositions to the control states.) To begin with, we proved that the model-checking problem is EXPSPACE-complete. To obtain a better understanding of this semantics, we have considered some variants of the model-checking problem, with a restriction on the form of the structures: path model-checking and model-checking of flat structures. Both problems turn out to be PSPACE-complete. Our results are summarised in Figure 5. We also showed that the problems $\mathsf{MC}_\forall(\mathsf{q\text{-}LTL})$ and $\mathsf{MC}_\exists(\mathsf{q\text{-}LTL})$ corresponding to another way of considering quantifications are PSPACE-complete.

problem:	$\mathsf{MC}_\exists(\mathsf{QLTL})$	$\mathsf{MC}_\forall(\mathsf{QLTL})$	$\mathsf{MC}_\mathsf{p}(\mathsf{QLTL})$	$\mathsf{MC}_{\mathsf{f},\exists}(\mathsf{QLTL})$	$\mathsf{MC}_{\mathsf{f},\forall}(\mathsf{QLTL})$
complexity:	EXPSPACE-complete		PSPACE-complete		

Figure 5 Complexity of QLTL model-checking.

The interesting properties that this semantics can express (as the finite number of reachable control states, the determinism of an execution) leads us to continue studying this semantics by working on the satisfiability problem of fragments of QLTL and on the expressivity of prenex formulas.

References

1 B. Bednarczyk and S. Demri. Why Does Propositional Quantification Make Modal and Temporal Logics on Trees Robustly Hard? *Logical Methods in Computer Science*, 18(3):5:1–5:46, July 2022.

2 A. K. Chandra, D. Kozen, and L. J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, 1981.

3 H. Comon and Y. Jurski. Multiple counter automata, safety analysis and PA. In *CAV'98*, volume 1427 of *LNCS*, pages 268–279. Springer, 1998.
4 S. Demri, A. K. Dhar, and A. Sangnier. Taming past LTL and flat counter systems. *Inf. Comput.*, 242:306–339, 2015.
5 E. A. Emerson and A. P. Sistla. Deciding full branching time logic. *Information and Control*, 61(3):175–201, June 1984.
6 K. Etessami. Stutter-invariant languages, ω-automata, and temporal logic. In Nicolas Halbwachs and Doron A. Peled, editors, *CAV'99)*, volume 1633 of *LNCS*, pages 236–248. Springer-Verlag, July 1999.
7 T. French. Decidability of quantified propositional branching time logics. In *AJCAI'01*, volume 2256 of *LNCS*, pages 165–176. Springer-Verlag, December 2001.
8 T. French. Quantified propositional temporal logic with repeating states. In *TIME-ICTL'03*, pages 155–165. IEEE Comp. Soc. Press, July 2003.
9 T. French and M. Reynolds. A sound and complete proof system for QPTL. In Philippe Balbiani, Nobu-Yuki Suzuki, Frank Wolter, and Michael Zakharyaschev, editors, *AIML'02*, pages 127–148. King's College Publications, 2003.
10 A. Hossain and F. Laroussinie. QCTL model-checking with QBF solvers. *Inf. Comput.*, 280:104642, 2021.
11 Y. Kesten and A. Pnueli. Complete proof system for QPTL. *Journal of Logic and Computation*, 12(5):701–745, October 2002.
12 L. Kuhtz and B. Finkbeiner. Weak Kripke structures and LTL. In *CONCUR'11*, volume 6901 of *LNCS*, pages 419–433. Springer, 2011.
13 O. Kupferman. Augmenting branching temporal logics with existential quantification over atomic propositions. In *CAV'95*, volume 939 of *LNCS*, pages 325–338. Springer-Verlag, July 1995.
14 F. Laroussinie and N. Markey. Quantified CTL: expressiveness and complexity. *Logical Methods in Computer Science*, 10(4), 2014.
15 F. Laroussinie, N. Markey, and P. Schnoebelen. Temporal logic with forgettable past. In *LICS'02*, pages 383–392. IEEE Computer Society, 2002.
16 N. Markey and P.Schnoebelen. Model checking a path. In *CONCUR'03*, volume 2761 of *LNCS*, pages 248–262. Springer, 2003.
17 N. Markey and P. Schnoebelen. Model checking a path. In *CONCUR'03*, volume 2761 of *LNCS*, pages 248–262. Springer, 2003.
18 Markey N and P. Schnoebelen. Mu-calculus path checking. *Inf. Process. Lett.*, 97(6):225–230, 2006.
19 A. Pnueli. The temporal logic of programs. In *FOCS'77*, pages 46–57. IEEE Comp. Soc. Press, October-November 1977.
20 F. Schwarzentruber. The complexity of tiling problems. *CoRR*, abs/1907.00102, 2019.
21 A. P. Sistla. *Theoretical Issues in the Design and Verification of Distributed Systems*. PhD thesis, Harvard University, Cambridge, Massachussets, USA, 1983.
22 A. P. Sistla, M. Y. Vardi, and P. Wolper. The complementation problem for Büchi automata with applications to temporal logics. *Theoretical Computer Science*, 49:217–237, 1987.
23 M. Y. Vardi. Alternating automata and program verification. In Jan van Leeuwen, editor, *Computer Science Today: Recent Trends and Developments*, volume 1000 of *LNCS*, pages 471–485. Springer, 1995.
24 P. Wolper. Temporal logic can be more expressive. *Inf. Control.*, 56(1/2):72–99, 1983.

Limitations of Game Comonads for Invertible-Map Equivalence via Homomorphism Indistinguishability

Moritz Lichter
RWTH Aachen University, Germany

Benedikt Pago
University of Cambridge, UK

Tim Seppelt
RWTH Aachen University, Germany

Abstract

Abramsky, Dawar, and Wang (2017) introduced the pebbling comonad for k-variable counting logic and thereby initiated a line of work that imports category theoretic machinery to finite model theory. Such game comonads have been developed for various logics, yielding characterisations of logical equivalences in terms of isomorphisms in the associated co-Kleisli category. We show a first limitation of this approach by studying linear-algebraic logic, which is strictly more expressive than first-order counting logic and whose k-variable logical equivalence relations are known as invertible-map equivalences (IM). We show that there exists no finite-rank comonad on the category of graphs whose co-Kleisli isomorphisms characterise IM-equivalence, answering a question of Ó Conghaile and Dawar (CSL 2021). We obtain this result by ruling out a characterisation of IM-equivalence in terms of homomorphism indistinguishability and employing the Lovász-type theorem for game comonads established by Reggio (2022). Two graphs are *homomorphism indistinguishable* over a graph class if they admit the same number of homomorphisms from every graph in the class. The IM-equivalences cannot be characterised in this way, neither when counting homomorphisms in the natural numbers, nor in any finite prime field.

2012 ACM Subject Classification Theory of computation → Finite Model Theory

Keywords and phrases finite model theory, graph isomorphism, linear-algebraic logic, homomorphism indistinguishability, game comonads, invertible-map equivalence

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.36

Related Version *Full Version*: https://arxiv.org/abs/2308.05693

Funding *Moritz Lichter*: European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (EngageS: agreement No. 820148).
Tim Seppelt: German Research Foundation (DFG) via RTG 2236/2 (UnRAVeL), European Union (ERC, SymSim, 101054974).

Acknowledgements We would like to thank the anonymous reviewers for detailed comments. Moreover, we are greatful for discussions which took place at the "Resources and Co-Resources" workshop at the University of Cambridge in July 2023.

1 Introduction

Logic fragments such as k-variable first-order logic with or without counting quantifiers induce equivalence relations on graphs, or more generally, on structures: Two structures are equivalent in this sense if they satisfy exactly the same sentences of the respective logic fragment. Such equivalence relations are approximations of the isomorphism relation. The more expressive the logic fragment, the more non-isomorphic structures are distinguished by it. Classical model-comparison games and counterexamples like the Cai–Fürer–Immerman (CFI) construction show that k-variable first-order logic (even with counting) does not distinguish all pairs of non-isomorphic structures. Hence, the induced equivalence is indeed

strictly coarser than isomorphism. Such approximations of isomorphism can be studied from many different angles. For example, it is well-known that counting logic equivalence is the same as indistinguishability by the Weisfeiler–Leman graph isomorphism test [6].

Another perspective to approximations of isomorphism is offered by homomorphism indistinguishability: Two graphs G and H are *homomorphism indistinguishable* over a class of graphs $\mathcal{F}$ if for all graphs $F \in \mathcal{F}$ the number of homomorphisms from F to G is equal to the number of homomorphisms from F to H. Equivalence relations with respect to many logic fragments can be characterised as homomorphism indistinguishability relations over some graph class. For example, two graphs are counting logic equivalent if and only if they are homomorphism indistinguishable over all graphs of bounded treewidth [13, 12]. Besides counting logic equivalence, many other natural equivalence relations between graphs, including isomorphism [24], quantum isomorphism [26], cospectrality [12], and feasibility of integer programming relaxations for graph isomorphism [12, 18, 33] have been characterised as homomorphism indistinguishability relations over various graph classes. Characterising (logical) equivalences as homomorphism indistinguishability relations is desirable because such characterisations allow to compare the expressive power of logics solely by comparing the graph classes from which homomorphisms are counted [33, 32]. In this way, deep results from structural graph theory are made available for studying the expressive power of logics [34].

It is natural to ask whether this approach can be extended to interesting logics that are more expressive than counting logic, as they are for example studied in the quest for a logic for Ptime. Such examples are *rank logic* [9, 16] and the more general *linear-algebraic logic* (LA) [8]. We answer this question in the negative. The *invertible-map equivalence* $\equiv^{\mathrm{IM}}_{k,\mathbb{P}}$, as the equivalence of the k-variable fragment of LA is called, cannot be characterised as a homomorphism indistinguishability relation.

▶ **Theorem 1.** *For every $k \geq 6$, $\equiv^{\mathrm{IM}}_{k,\mathbb{P}}$ is not a homomorphism indistinguishability relation.*

The proof relies on a CFI-like construction similar to the one used by the first author to separate rank logic from polynomial time [23]. We combine this with results by Roberson [32] in order to obtain graphs that are invertible-map equivalent but not quantum isomorphic. As shown by the third author [34], this suffices to conclude that invertible-map equivalence is not a homomorphism indistinguishability relation – if it were, then it would have to be a refinement of quantum isomorphism.

Theorem 1 implies a negative answer to a question posed by Ó Conghaile and Dawar [30]. Their work is part of a recent line of research that explores connections between methods from finite model theory, descriptive complexity, and category theory [3]. One goal of these efforts is to characterise logical equivalences using *game comonads*. Concretely, Ó Conghaile and Dawar asked whether this is possible for linear-algebraic logic. Employing a categorical Lovász-type theorem [11, 31] that allows to infer a homomorphism indistinguishability characterisation from the existence of appropriate game comonads, we obtain the following result. To our knowledge, this is the first provable limitation of such comonadic characterisations.

▶ **Theorem 2.** *For every $k \geq 6$, there is no finite-rank comonad $\mathbb{C}$ on the category of graphs such that $\equiv^{\mathrm{IM}}_{k,\mathbb{P}}$ coincides with the isomorphism relation in the co-Kleisli category of $\mathbb{C}$.*

In this context, the concept of a comonad is best explained by recalling the *pebbling comonad* $\mathbb{T}_k$ introduced by Abramsky, Dawar, and Wang [1]. Designed to provide a categorical formulation of the k-pebble game from finite model theory, it can be thought of as map sending structures to structures encoding Spoiler's plays in this game. Being a comonad, it gives rise to a category, its co-Kleisli category, whose objects are graphs and whose morphisms

can be interpreted as winning strategies for Duplicator in the k-pebble game. Various notions from finite model theory can now be recovered from this construction: For example, a graph has treewidth less than k if and only if it admits a $\mathbb{T}_k$-coalgebra. Crucially, two graphs satisfy the same k-variable counting logic sentences if and only if they are isomorphic in the co-Kleisli category of $\mathbb{T}_k$. Subsequently, comonads for many fragments [1, 3, 27] and extensions [30] of first-order logic have been constructed. They have in common that their co-Kleisli morphisms and isomorphisms encode winning strategies for Duplicator in one-sided, symmetric, and bijective games. Our Theorem 2 rules out a characterisation of invertible-map equivalence via co-Kleisli isomorphisms. We note that our Theorem 1 does not exclude characterisations involving other comonadic constructions.

Comonads on the category of graphs and homomorphism indistinguishability are intimately connected. Every homomorphism indistinguishability relation over a graph class with mild closure properties can be characterised as co-Kleisli isomorphism over a comonad [2]. Conversely, the existence of co-Kleisli isomorphisms over a finite-rank comonad can be characterised as a homomorphism indistinguishability relation [11, 31]. This fundamental connection between comonads and homomorphism counting relations is exactly the reason why we can conclude the impossibility of the former from the impossibility of the latter: There is no finite-rank comonad for linear-algebraic logic.

Hence, linear-algebraic logic seems to be of a very different nature than the weaker counting logic as it does not connect with the theory revolving around homomorphism indistinguishability and game comonads. This raises the question as to what is the precise reason for this situation. What makes a logic "nice enough" to fit within the homomorphism indistinguishability and comonadic framework? We can at least say that the shortcomings of LA in this respect are not due to it being strictly stronger than counting logic. There does exist an extension of counting logic which admits a comonad construction and thereby a homomorphism indistinguishability relation: This is k-variable infinitary FO enriched with all possible n-ary *generalised quantifiers* over one-dimensional interpretations [30]. An n-ary generalised quantifier (also known as *Lindström quantifier*) is essentially a membership oracle for a class $\mathcal{K}$ (of at most n-ary structures) that allows to test whether some structure $\mathfrak{B}$ interpretable in the given structure $\mathfrak{A}$ is in $\mathcal{K}$. LA lies somewhere between counting logic and its extension by *all* binary Lindström quantifiers because LA is infinitary FO extended with a *proper subclass* of binary Lindström quantifiers. Counting logic itself is nothing but the extension of FO with *all unary* Lindström quantifiers [22]. Hence, we can describe the situation as follows: Whenever a Lindström-extension of infinitary FO contains *all* one-dimensional Lindström quantifiers up to a given arity n, then it admits a comonad. If it only contains a subset of these Lindström quantifiers, then this is not necessarily the case (our Theorem 1 is true even when we restrict LA to one-dimensional interpretations).

Finally, another direction that we explore in this paper is counting homomorphisms in *finite prime fields*. A large part of the theory of homomorphism indistinguishability that has been established so far works over the natural numbers. Given the fact that the linear-algebraic operators in LA are over finite fields, one might a priori suspect that the appropriate homomorphism indistinguishability relation must be based on homomorphism counts modulo a prime. However, this can also be ruled out, even when the homomorphisms are counted modulo several primes (Theorem 25).

As a positive result concerning homomorphism counting modulo primes, we find that Dvořák's proof [13] can be adapted to finite fields: Two graphs admit the same numbers of homomorphisms modulo p from all graphs of treewidth less than k if and only if they are equivalent with respect to k-variable modular counting logic (Theorem 26).

2 Preliminaries

All structures in this paper are relational and finite. General relational structures are usually denoted $\mathfrak{A}$ or $\mathfrak{B}$, with A or B, respectively, being used for the universe. When we speak of graphs, we mean $\{E\}$-structures, where E is binary, and we will write $V(G)$ and $E(G)$ for the vertices respectively edges of a graph G. When nothing else is specified, graphs are undirected and we may write $uv \in E(G)$ for edges $\{u, v\} \in E(G)$. The set $\{1, 2, ..., n\}$ is denoted by $[n]$, and $\mathbb{P} \subseteq \mathbb{N}$ denotes the set of primes.

Counting logic. The logic $\mathcal{C}^k$ is the k-variable fragment of first-order logic with counting quantifiers of the form $\exists^{\geq i} x$, for every $i \in \mathbb{N}$. The semantics is as expected, i.e., a structure $\mathfrak{A}$ satisfies a sentence $\exists^{\geq i} x \varphi(x)$ if there exist at least i distinct $a \in A$ such that $\mathfrak{A} \models \varphi(a)$. We write $\mathfrak{A} \equiv_{\mathcal{C}^k} \mathfrak{B}$ if $\mathfrak{A}$ and $\mathfrak{B}$ are $\mathcal{C}^k$-equivalent, i.e., they satisfy exactly the same $\mathcal{C}^k$-sentences.

Lindström quantifiers and interpretations. A more general way to extend FO is with *Lindström* quantifiers (also known as *generalised* quantifiers). A Lindström quantifier is essentially a membership oracle for a class of structures. Before introducing Lindström quantifiers, we need the concept of logical interpretations. Let σ and τ be relational vocabularies with $\tau = \{R_1, ..., R_m\}$ where each R_i is a relation symbol of arity r_i, and let $\mathcal{L}$ be a logic. An *ℓ-dimensional $\mathcal{L}[\sigma, \tau]$-interpretation* I is an $\mathcal{L}$-definable mapping from σ-structures to τ-structures. The elements of the τ-structure are sets of ℓ-tuples in the original σ-structure. Generally, interpretations can take a tuple of parameters $\overline{z}$: An ℓ-dimensional $\mathcal{L}$-interpretation (with parameters) is a tuple

$$I(\overline{z}) = \big(\varphi_\delta(\overline{x}, \overline{z}), \varphi_\approx(\overline{x}, \overline{y}, \overline{z}), \varphi_{R_1}(\overline{x}_1, ..., \overline{x}_{r_1}, \overline{z}), ..., \varphi_{R_m}(\overline{x}_1, ..., \overline{x}_{r_m}, \overline{z})\big),$$

where $\overline{x}$, $\overline{y}$, $\overline{x}_i$ are ℓ-tuples of variables, and $\varphi_\delta, \varphi_\approx, \varphi_{R_i}$ are σ-formulas of the logic $\mathcal{L}$. The interpretation $I(\overline{z})$ defines a partial mapping from σ-structures to τ-structures. For a given σ-structure $\mathfrak{A}$ and an assignment $\overline{z} \mapsto \overline{a}$, we define $\mathfrak{B}$ to be the τ-structure that has universe $B := \{\overline{b} \in A^k \mid \mathfrak{A} \models \varphi_\delta(\overline{b}, \overline{a})\}$ and, for all $i \in [m]$, has the relations $R_i^{\mathfrak{B}} := \{(\overline{b}_1, ..., \overline{b}_{r_i}) \in B^{r_i} \mid \mathfrak{A} \models \varphi_{R_i}(\overline{b}_1, ..., \overline{b}_{r_i}, \overline{a})\}$. From this structure, we obtain the "output" $I(\mathfrak{A}, \overline{z} \mapsto \overline{a})$ of I by factoring out the equivalence classes defined by $\varphi_\approx$. Formally, let $\mathcal{E} := \{(\overline{b}_1, \overline{b}_2) \in A^{2k} \mid \mathfrak{A} \models \varphi_\approx(\overline{b}_1, \overline{b}_2, \overline{a})\}$. If $\mathcal{E}$ is not a congruence relation on $\mathfrak{B}$, then $I(\mathfrak{A}, \overline{z} \mapsto \overline{a})$ is undefined. Otherwise, $I(\mathfrak{A}, \overline{z} \mapsto \overline{a})$ is defined as the quotient structure $\mathfrak{B}/\mathcal{E}$.

Let $\mathcal{K}$ be a class of τ-structures and $\mathcal{L}$ be a logic. The extension $\mathcal{L}(\mathcal{Q}_\mathcal{K})$ of $\mathcal{L}$ by the *Lindström quantifier* for $\mathcal{K}$ is obtained by closing $\mathcal{L}$ under the following formula formation rule: Whenever $I(\overline{z})$ is an $\mathcal{L}(\mathcal{Q}_\mathcal{K})[\sigma, \tau]$-interpretation, then $\mathcal{Q}_\mathcal{K} I(\overline{z})$ is a σ-formula of $\mathcal{L}(\mathcal{Q}_\mathcal{K})$ with free variables $\overline{z}$. For a σ-structure $\mathfrak{A}$ and an assignment $\overline{z} \mapsto \overline{a}$, it holds $(\mathfrak{A}, \overline{a}) \models \mathcal{Q}_\mathcal{K} I(\overline{z})$ if $I(\mathfrak{A}, \overline{z} \mapsto \overline{a}) \in \mathcal{K}$. If $\mathbf{Q}$ is a class of Lindström quantifiers, then $\mathcal{L}(\mathbf{Q})$ denotes the extension by all Lindström quantifiers in $\mathbf{Q}$. When we speak of the one-dimensional restriction of such a logic, we mean that in formulas $\mathcal{Q}_\mathcal{K} I(\overline{z})$, the interpretation I has to be one-dimensional.

Linear-algebraic logic and invertible-map equivalences. Linear-algebraic logic (LA) was introduced by Dawar, Grädel, and Pakusa [8] as an extension of infinitary first-order logic with all isomorphism-invariant linear-algebraic operators. As such, it extends *rank logic* [9, 16]. Rank logic in turn is an extension of FO with operators for determining the rank of a matrix that is definable in the input structure. In linear-algebraic logic, formulas have access to *any* isomorphism-invariant parameter of a definable matrix and not only to the rank. This logic was studied to show that no linear-algebraic operators whatsoever can

enhance the power of FO such that its k-variable fragment distinguishes all non-isomorphic structures, for some fixed k. For the detailed definition of LA, we refer to [8]. In short, a *linear-algebraic* function over some field $\mathbb{F}$ with some arity $m \geq 1$ is a function f that maps tuples $(M_1, ..., M_m)$ of linear transformations/matrices over $\mathbb{F}$ to natural numbers such that f is invariant under vector space isomorphisms. Formally, this means that whenever two sequences of matrices $M_1, ..., M_m$ and $M'_1, ..., M'_m$ over $\mathbb{F}$ are *simultaneously similar*, then $f(M_1, ..., M_m) = f(M'_1, ..., M'_m)$. Simultaneous similarity means that there is an invertible matrix S over $\mathbb{F}$ such that $M_i \cdot S = S \cdot M'_i$ for all $i \in [m]$. That is to say, there exists an isomorphism between the underlying vector spaces that maps each linear transformation M_i to the corresponding M'_i that operates on the isomorphic space. For instance, the rank operator is such a function with arity $m = 1$ that maps a given matrix to its rank.

With every m-ary linear-algebraic function f and every natural number r, we associate the class $\mathcal{K}_f^t$ of structures (for some appropriate vocabulary) that encode tuples of matrices $(M_1, ..., M_m)$ satisfying $f(M_1, ..., M_m) \geq t$. Now, linear-algebraic logic LA is the closure of FO under infinite conjunctions and disjunctions and under Lindström quantifiers for all classes $\mathcal{K}_f^t$: A structure $\mathfrak{A}$ satisfies $\mathcal{Q}_{\mathcal{K}_f^t} I(\overline{x})$ if $I(\mathfrak{A})$ is a structure that encodes a tuple $(M_1, ..., M_m)$ of matrices and satisfies $f(M_1, ..., M_m) \geq t$.

Fragments of LA yield interesting equivalence relations between structures, which are approximations of isomorphism. The fragments that are studied in the literature (e.g. in [8, 23]) are parametrized by $k \in \mathbb{N}$ and $Q \subseteq \mathbb{P}$. The logic $\mathrm{LA}^k(Q)$ is the k-variable fragment of LA that only uses linear-algebraic operators over finite fields of characteristic $p \in Q$. The equivalence relation induced by $\mathrm{LA}^k(Q)$ is called *invertible-map equivalence*. We write $\mathfrak{A} \equiv^{\mathrm{IM}}_{k,Q} \mathfrak{B}$ if the structures $\mathfrak{A}$ and $\mathfrak{B}$ satisfy exactly the same $\mathrm{LA}^k(Q)$-sentences. Invertible-map equivalence of two given structures can be tested in polynomial time [8].

The logic $\mathrm{LA}^k(Q)$ is at least as expressive as $\mathcal{C}^k$ because the quantifier $\exists^{\geq i} x\varphi(x)$ can be simulated with the rank operator [8]: We have $\mathfrak{A} \models \exists^{\geq i} x\varphi(x)$ if and only if the diagonal matrix that has a 1-entry at exactly those positions $(a, a) \in A^2$ such that $\mathfrak{A} \models \varphi(a)$ has rank at least i. This works irrespective of which primes are in Q. Hence, for every non-empty Q, the relation $\equiv^{\mathrm{IM}}_{k,Q}$ is at least as fine as $\equiv_{\mathcal{C}^k}$. In fact, it is strictly finer because there exist generalised CFI-structures that are $\equiv_{\mathcal{C}^k}$-equivalent but distinguishable in rank logic [9] using ranks over $\mathbb{F}_p$ for each $p \in Q$.

Invertible-map equivalence is also characterized by a Spoiler-Duplicator game called the *invertible-map game* [10]. We follow the exposition in [23]. Let $Q \subseteq \mathbb{P}$ and $k \in \mathbb{N}$. The IM-game $\mathcal{M}^{k,Q}$ is played on two structures $\mathfrak{A}$ and $\mathfrak{B}$. There are k pairs of pebbles labelled with $1, \ldots, k$. A position in the game is a pair $\overline{a}, \overline{b}$ of tuples $\overline{a} \in A^m$ and $\overline{b} \in B^m$ for some $m \leq k$. In position $\overline{a}, \overline{b}$ corresponding pebbles, i.e., pebbles with the same label, are placed on a_i and b_i for every $i \in [k]$. Initially, the pebbles are not on the board. If $|A| \neq |B|$, then Spoiler wins immediately. Otherwise, a round of the game is played as follows:

1. Spoiler chooses a prime $p \in Q$ and a number ℓ satisfying $2\ell \leq k$. Next, Spoiler picks up 2ℓ pebbles from $\mathfrak{A}$ and the corresponding pebbles (with the same labels) from $\mathfrak{B}$.
2. Duplicator picks a partition $\mathcal{P}$ of $A^\ell \times A^\ell$ and another one $\mathcal{P}'$ of $B^\ell \times B^\ell$ such that $|\mathcal{P}| = |\mathcal{P}'|$. Furthermore, Duplicator picks a bijection $h \colon \mathcal{P} \to \mathcal{P}'$ and an invertible $(A^\ell \times B^\ell)$-matrix S over $\mathbb{F}_p$ such that $\chi^P = S \cdot \chi^{h(P)} \cdot S^{-1}$ for every $P \in \mathcal{P}$. Here, χ^P denotes the characteristic matrix of P, which has a 1-entry at position $(\overline{u}, \overline{v})$ if $\overline{uv} \in P$ and is 0 otherwise.
3. Spoiler chooses a block $P \in \mathcal{P}$, a tuple $\overline{u} \in P$, and a tuple $\overline{v} \in h(P)$. Then for each $i \in [2\ell]$, Spoiler places one of the pebbles picked up from $\mathfrak{A}$ on u_i and the corresponding one picked up from $\mathfrak{B}$ on v_i.

After a round, Spoiler wins the game if the pebbles do not define a partial isomorphism or if Duplicator was not able to respond with a matrix satisfying the condition above. Note that this condition states that the characteristic matrices of the blocks are simultaneously similar.

▶ **Lemma 3.** *Let $k \in \mathbb{N}$, $Q \subseteq \mathbb{P}$, $\mathfrak{A}$ and $\mathfrak{B}$ be structures, $\overline{a} \in A^k$, and $\overline{b} \in B^k$. Then $(\mathfrak{A}, \overline{a}) \equiv^{\mathrm{IM}}_{k,Q} (\mathfrak{A}, \overline{b})$ if and only if Duplicator has a winning strategy in the invertible-map game $\mathcal{M}^{k,Q}$ on $\mathfrak{A}$ and $\mathfrak{B}$ in position $\overline{a}, \overline{b}$.*

The lemma follows from a combination of [10, 8], in which the game is also parametrised by the dimension 2ℓ of the interpretations. In [10], only finite sets of primes are considered because the logics considered there are not infinitary. The arguments straight-forwardly apply to arbitrary sets of primes.

Homomorphism Indistinguishability. Let F and G be graphs. A *homomorphism* ψ from F to G is a map $\psi\colon V(F) \to V(G)$ such that $\psi(u)\psi(v) \in E(G)$ for every edge $uv \in E(F)$. We write $\mathrm{Hom}(F, G)$ for set of homomorphisms from F to G and $\hom(F, G) \coloneqq |\mathrm{Hom}(F, G)|$. Homomorphism counts induce equivalence relations on graphs: Let $\mathcal{F}$ be a class of graphs. Two graphs G and H are *homomorphism indistinguishable* over $\mathcal{F}$, denoted by $G \equiv_{\mathcal{F}} H$, if for every $F \in \mathcal{F}$, it holds that $\hom(F, G) = \hom(F, H)$. An equivalence relation $\approx$ between graphs is a *homomorphism indistinguishability relation* if there exists a graph class $\mathcal{F}$ such that $\approx$ and $\equiv_{\mathcal{F}}$ coincide.

In this paper, we call two graphs *quantum isomorphic* if they are homomorphism indistinguishable over all planar graphs. The term was originally introduced as a quantum information theoretic notion [4]. The titular result of Mančinska's and Roberson's seminal work [26] asserts that it is the same as homomorphism indistinguishability over all planar graphs. Note that our results do not depend on [4, 26].

3 Homomorphisms to CFI-Like Graphs over Finite Abelian Groups

Roberson [32] studied homomorphisms to CFI-like graphs constructed over $\mathbb{Z}_2$. This variant of CFI graphs was introduced by Fürer [15]. Neuen and Schweitzer [29] generalised the more classical CFI construction from $\mathbb{Z}_2$ to arbitrary finite abelian groups. We combine both constructions and generalise the CFI construction from [15, 32] to arbitrary finite abelian groups. The goal of these efforts is to show that certain CFI graphs over planar base graphs and w.r.t. to arbitrary finite abelian groups are not quantum isomorphic.

We fix such a group Γ throughout this section and write its operation as addition. For a graph G and a vertex $v \in V(G)$, write $E(v) \coloneqq \{e \in E(G) \mid v \in e\}$ for the set of edges incident to v. We consider vectors $U \in \Gamma^X$ for finite sets X. For an element $x \in X$, we write $U(x) \in \Gamma$ for the x-th entry of U. We write $\sum U$ for $\sum_{x \in X} U(x)$.

▶ **Definition 4.** *A* base graph *is a connected graph. Let G be a base graph and $U \in \Gamma^{V(G)}$. For every vertex u of G, we define*

$$V_u \coloneqq \left\{ (u, S) \;\middle|\; S \in \Gamma^{E(u)}, \sum S = U(u) \right\}.$$

The CFI graph $\mathsf{CFI}[\Gamma, G, U]$ *over the finite abelian group Γ and the base graph G has vertex set $\bigcup_{u \in V(G)} V_u$ and edge set*

$$\left\{ \{(u, S), (v, T)\} \;\middle|\; (u, S) \in V_u, (v, T) \in V_v, uv \in E(G), S(uv) + T(uv) = 0 \right\}.$$

We say that the vertices in V_u have origin u.

The proof of the following Lemma 5 uses well-known arguments for CFI graphs [15, 29, 23].

▶ **Lemma 5.** *Let G be a base graph and $U, U' \in \Gamma^{V(G)}$. If $\sum U = \sum U'$, then $\mathsf{CFI}[\Gamma, G, U] \cong \mathsf{CFI}[\Gamma, G, U']$.*

Proof. Let $uv \in E(G)$. Denote the vertex set of $\mathsf{CFI}[\Gamma, G, U]$ respectively $\mathsf{CFI}[\Gamma, G, U']$ by V_U and $V_{U'}$. First consider $U' \coloneqq U + u - v$ where u and v denote the vectors in $\Gamma^{V(G)}$ with one at the u-th and v-th component, respectively, and zero otherwise. Define the map $\varphi\colon V_U \to V_{U'}$ by

$$\varphi((w,S)) \coloneqq \begin{cases} (u, S + uv), & \text{if } w = u, \\ (v, S - uv), & \text{if } w = v, \\ (w, S), & \text{otherwise.} \end{cases}$$

where uv denotes the vector in $\Gamma^{E(u)}$ in the first case or in $\Gamma^{E(v)}$ in the second case with one at the uv-th component and zero otherwise. Observe that $\sum_{e \in E(v)} (S - uv)(e) = U(v) - 1 = U'(v)$ and analogously for u. Hence, φ is indeed a well-defined map to $V_{U'}$. Clearly, φ is a bijection. Let $(x,S), (y,T) \in V_U$ be arbitrary vertices of $\mathsf{CFI}[\Gamma, G, U]$ and define $(x, S') \coloneqq \varphi(x,S)$ and $(y,T') \coloneqq \varphi(y,T)$. Then $S'(xy) + T'(xy) = S(xy) + T(xy)$. Hence, (x,S) and (y,T) are adjacent in $\mathsf{CFI}[\Gamma, G, U]$ if and only if they are adjacent in $\mathsf{CFI}[\Gamma, G, U']$.

Since G is connected, the maps constructed above can be composed to yield $\mathsf{CFI}[\Gamma, G, U] \cong \mathsf{CFI}[\Gamma, G, U + u - v]$ for every pair of vertices u, v. This yields $\mathsf{CFI}[\Gamma, G, U] \cong \mathsf{CFI}[\Gamma, G, U']$ as desired. ◀

We proceed by counting homomorphisms into the CFI graphs. For a graph G and $U \in \Gamma^{V(G)}$, consider the *projection map* $\rho\colon \mathsf{CFI}[\Gamma, G, U] \to G$ sending (v,S) to v. Clearly, ρ is a homomorphism. For a graph F and a homomorphism $\psi\colon F \to G$, define

$$\mathrm{Hom}_\psi(F, \mathsf{CFI}[\Gamma, G, U]) \coloneqq \left\{ \varphi \in \mathrm{Hom}(F, \mathsf{CFI}[\Gamma, G, U]) \;\middle|\; \rho \circ \varphi = \psi \right\}.$$

The sets $\mathrm{Hom}_\psi(F, \mathsf{CFI}[\Gamma, G, U])$ for all $\psi\colon F \to G$ partition the set $\mathrm{Hom}(F, \mathsf{CFI}[\Gamma, G, U])$ of homomorphisms $F \to \mathsf{CFI}[\Gamma, G, U]$. Write $\hom_\psi(F, \mathsf{CFI}[\Gamma, G, U])$ for the cardinality of $\mathrm{Hom}_\psi(F, \mathsf{CFI}[\Gamma, G, U])$.

▶ **Lemma 6.** *Let F be a graph and G be a base graph. Let $U \in \Gamma^{V(G)}$ and fix $\psi \in \mathrm{Hom}(F, G)$. Consider the system of equations $\mathsf{Hom}(F, G, U, \psi)$ with variables x_e^a for all $a \in V(F)$ and $e \in E(\psi(a))$ and equations*

$$\sum_{e \in E(\psi(a))} x_e^a = U(\psi(a)) \qquad \text{for all } a \in V(F), \tag{1}$$

$$x_e^a + x_e^b = 0 \qquad \text{for all } ab \in E(F) \text{ and } e = \psi(ab) \in E(G). \tag{2}$$

Then the number of solutions to $\mathsf{Hom}(F, G, U, \psi)$ over Γ is equal to $\hom_\psi(F, \mathsf{CFI}[\Gamma, G, U])$.

Proof. The proof is by giving a bijection between the solution set and $\mathrm{Hom}_\psi(F, \mathsf{CFI}[\Gamma, G, U])$. Let $\xi = (\xi_e^a)_{a \in V(F), e \in E(\psi(a))}$ be a solution to $\mathsf{Hom}(F, G, U, \psi)$ over Γ. Define a homomorphism $\varphi_\xi \in \mathrm{Hom}_\psi(F, \mathsf{CFI}[\Gamma, G, U])$ via $\varphi_\xi(a) \coloneqq \big(\psi(a), (\xi_e^a)_{e \in E(\psi(a))}\big)$. Equation (1) guarantees that this is indeed a map from the vertices of F to the ones of $\mathsf{CFI}[\Gamma, G, U]$. If a and b are adjacent in F, then so are $\psi(a)$ and $\psi(b)$ in G. Furthermore, $\xi^a_{\psi(ab)} + \xi^b_{\psi(ab)} = 0$ by Equation (2). Hence, $\varphi_\xi(a)$ and $\varphi_\xi(b)$ are adjacent in $\mathsf{CFI}[\Gamma, G, U]$.

It is easy to see that this construction is injective, i.e., if $\varphi_\xi = \varphi_\zeta$, then $\xi = \zeta$. For surjectivity, let $\varphi \in \mathrm{Hom}_\psi(F, \mathsf{CFI}[\Gamma, G, U])$. For every $a \in V(F)$ and $e \in E(\psi(a))$, define ξ_e^a as the second component of $\varphi(a)$, i.e. $\xi_e^a \coloneqq S_a(e)$ where $\varphi(a) = (\psi(a), S_a)$. Clearly, $\xi \coloneqq (\xi_e^a)$ is such that $\varphi_\xi = \varphi$. The fact that ξ satisfies Equations (1) and (2) is easily verified. ◀

▶ **Theorem 7.** *For a base graph G, $U \in \Gamma^{V(G)}$, and $\psi \in \mathrm{Hom}(F, G)$ for some graph F, the following hold:*

1. $\hom_\psi(F, \mathsf{CFI}[\Gamma, G, 0]) > 0$.
2. *If* $\mathsf{Hom}(F, G, U, \psi)$ *has a solution, then* $\hom_\psi(F, \mathsf{CFI}[\Gamma, G, 0]) = \hom_\psi(F, \mathsf{CFI}[\Gamma, G, U])$.
3. *If* $\mathsf{Hom}(F, G, U, \psi)$ *has no solution, then* $\hom_\psi(F, \mathsf{CFI}[\Gamma, G, U]) = 0$.

Proof. The system $\mathsf{Hom}(F, G, U, \psi)$ can be compressed into a matrix equation as follows: For $\psi \in \mathrm{Hom}(F, G)$ and $P \coloneqq \{(a, e) \mid a \in V(F), e \in E(\psi(a))\}$, let $A^\psi \in \Gamma^{V(F) \times P}$ and $B^\psi \in \Gamma^{E(F) \times P}$ be the matrices defined by

$$A^\psi_{b,(a,e)} \coloneqq \delta_{b=a} \quad \text{and} \quad B^\psi_{bc,(a,e)} \coloneqq \delta_{a \in \{b,c\} \wedge e = \psi(bc)}, \tag{3}$$

where δ_C, similar to the Kronecker delta, evaluates to 1 if the condition C is satisfied and is 0 otherwise. Then Equations (1) and (2) are equivalent to

$$\begin{pmatrix} A^\psi \\ B^\psi \end{pmatrix} x = \begin{pmatrix} U \circ \psi \\ 0 \end{pmatrix}. \tag{4}$$

If $U = 0$, then this system always has a solution, namely $\xi = 0$. In particular, by Lemma 6, $\mathrm{Hom}_\psi(F, \mathsf{CFI}[\Gamma, G, 0]) \neq \emptyset$. By Lemma 6, it remains to give a bijection between the sets of solutions to $\left(\begin{smallmatrix} A^\psi \\ B^\psi \end{smallmatrix}\right) x = \left(\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right)$ and the set of solutions to $\left(\begin{smallmatrix} A^\psi \\ B^\psi \end{smallmatrix}\right) x = \left(\begin{smallmatrix} U \circ \psi \\ 0 \end{smallmatrix}\right)$: Provided with a solution ξ to the latter system, $x \mapsto x + \xi$ can be taken to be this bijection. ◀

Theorem 7 yields Corollary 8 which gives a criterion for a CFI graph $\mathsf{CFI}[\Gamma, G, U]$ to have $\sum U = 0$ in terms of homomorphism counts from G. The condition in Item 3 is what allows us to infer the ultimate Corollary 9.

▶ **Corollary 8.** *Let G be a base graph and $U \in \Gamma^{V(G)}$. Then the following are equivalent:*

1. $\sum U = 0$,
2. $\mathsf{CFI}[\Gamma, G, U] \cong \mathsf{CFI}[\Gamma, G, 0]$,
3. $\hom(G, \mathsf{CFI}[\Gamma, G, U]) = \hom(G, \mathsf{CFI}[\Gamma, G, 0])$, *and*
4. $\hom_{\mathrm{id}}(G, \mathsf{CFI}[\Gamma, G, U]) = \hom_{\mathrm{id}}(G, \mathsf{CFI}[\Gamma, G, 0])$, *where* id *is the identity map on G.*

Proof. The fact that Item 1 implies Item 2 follows from Lemma 5. It is immediate that Item 2 implies Item 3. The fact that Item 3 implies Item 4 follows from Theorem 7.

It thus remains to prove that Item 4 implies Item 1. By Theorem 7, let ξ be a solution to Equation (4) for $\psi = \mathrm{id} \colon G \to G$. Then,

$$\sum_{a \in V(G)} U(a) \overset{(1)}{=} \sum_{a \in V(G)} \sum_{e \in E(a)} \xi^a_e = \sum_{e = ab \in E(G)} \xi^a_e + \xi^b_e \overset{(2)}{=} 0.$$

Hence, Item 1 holds. ◀

Thus, if G is planar, then G witnesses quantum non-isomorphism of its CFI graphs.

▶ **Corollary 9.** *If G is a planar base graph and $\sum U \neq 0$, then $\mathsf{CFI}[\Gamma, G, 0]$ and $\mathsf{CFI}[\Gamma, G, U]$ are not quantum isomorphic.*

4 Invertible-Map Equivalence and Homomorphism Indistinguishability

In this section we prove that, for every $k \geq 6$, the invertible-map equivalence $\equiv^{\mathrm{IM}}_{k,\mathbb{P}}$ over the set of all primes is not a homomorphism indistinguishability relation. The proof idea is the following: Using techniques from [23], we will construct, for every $k \in \mathbb{N}$, a planar base graph G such that we obtain non-isomorphic but $\equiv^{\mathrm{IM}}_{k,\mathbb{P}}$-equivalent generalised CFI graphs over G and $\mathbb{Z}_{2^i}$ for some $i \geq 1$. By Corollary 9, the two CFI graphs are not quantum isomorphic. Exploiting [34], we will see that this implies that $\equiv^{\mathrm{IM}}_{k,\mathbb{P}}$ is not a homomorphism-indistinguishability relation.

▶ **Lemma 10.** *Let $k \geq 6$. If $\equiv^{\mathrm{IM}}_{k,\mathbb{P}}$ (over graphs) is a homomorphism indistinguishability relation, then all $\equiv^{\mathrm{IM}}_{k,\mathbb{P}}$-equivalent graphs are quantum isomorphic.*

Proof. For every (self-complementary) logic $\mathcal{L}$, the following holds [34, Theorem 22]: If $\mathcal{L}$-equivalence is a homomorphism indistinguishability relation, and if, for every $\ell \in \mathbb{N}$, there are $\mathcal{C}^\ell$-equivalent but not $\mathcal{L}$-equivalent graphs H and H', then all $\mathcal{L}$-equivalent graphs are quantum isomorphic. Here, $\mathcal{L}$ is $\mathrm{LA}^k(\mathbb{P})$. We show that for every $\ell \in \mathbb{N}$, there are $\mathcal{C}^\ell$-equivalent but not $\mathrm{LA}^k(\mathbb{P})$-equivalent graphs H and H'. Let $\ell \in \mathbb{N}$. It is well-known [6] that there is a base graph G such that the two non-isomorphic CFI graphs H and H' over $\mathbb{Z}_2$ and G, using the classical CFI construction [6] (which we have not presented in this paper), are $\mathcal{C}^\ell$-equivalent. However, the CFI graphs H and H' are not equivalent in rank logic [9]. The interpretation defining the distinguishing matrices is actually one-dimensional and requires 6 variables [21]. Thus, H and H' are not $\mathrm{LA}^k(\mathbb{P})$-equivalent. ◀

The missing fundamental lemma to prove Theorem 1 is the following:

▶ **Lemma 11.** *For every $k \in \mathbb{N}$, there is a planar base graph G and an $i \in \mathbb{N}$ such that, for all $U, U' \in \mathbb{Z}_{2^i}^{V(G)}$ satisfying $\sum U = \sum U' + 2^{i-1}$, we have $\mathsf{CFI}[\mathbb{Z}_{2^i}, G, U] \equiv^{\mathrm{IM}}_{k,\mathbb{P}} \mathsf{CFI}[\mathbb{Z}_{2^i}, G, U']$.*

We first show how Theorem 1 can be proved using Lemma 11 and afterwards spend the rest of this section on the proof of Lemma 11.

Proof of Theorem 1. Let $k \geq 6$. By Lemma 11, there is a planar base graph G and an $i \in \mathbb{N}$ such that $\mathsf{CFI}[\mathbb{Z}_{2^i}, G, 0] \equiv^{\mathrm{IM}}_{k,\mathbb{P}} \mathsf{CFI}[\mathbb{Z}_{2^i}, G, U]$ for some $U \in \mathbb{Z}_{2^i}$ with $\sum U = 2^{i-1}$. These two CFI graphs are not quantum isomorphic by Corollary 9. Hence, the invertible-map equivalence $\equiv^{\mathrm{IM}}_{k,\mathbb{P}}$ is not a homomorphism indistinguishability relation by Lemma 10. ◀

Because the interpretation in the proof of Lemma 10 is one-dimensional, the result of Theorem 1 also holds for equivalence in the fragment of k-variable linear-algebraic logic that is restricted to one-dimensional interpretations.

It remains to prove Lemma 11. Without the planarity requirement, non-isomorphic but $\equiv^{\mathrm{IM}}_{k,\mathbb{P}}$-equivalent generalised CFI structures were constructed in [23]. By a careful analysis of the proof, the construction can be adapted to certain planar base graphs, which we will show now. However, we first have to extend our CFI graphs by additional relations. An *ordered graph* is a pair $(G, \leq)$ of a graph G and a total order $\leq$ on $V(G)$. If G is an ordered graph, we denote its vertex set, its edge set, and its order by $V(G)$, $E(G)$, and $\leq^G$, respectively.

▶ **Definition 12.** *Let i be a positive integer, G be an ordered base graph, and $U \in \mathbb{Z}_{2^i}^{V(G)}$. We define the* CFI structure $\mathsf{CFI}^*[\mathbb{Z}_{2^i}, G, U]$ *on the same vertex set as $\mathsf{CFI}[\mathbb{Z}_{2^i}, G, U]$, that is, on $\bigcup_{u \in V(G)} V_u$ (recall Definition 4). We first define a total preorder $\preceq$ on the vertices: $(u, S) \preceq (v, T)$ if and only if $u \leq^G v$. For every $uv \in E(G)$, we define the following relations:*

$$N_{u,v} := \{\, ((u,S),(u,T)) \in V_u^2 \mid S(uv) = T(uv) \,\},$$
$$C_{u,v} := \{\, ((u,S),(u,T)) \in V_u^2 \mid S(uv) + 1 = T(uv) \,\}.$$

Finally, we add for every $j \in \mathbb{Z}_{2^i}$ the following relation:

$$I_j := \{\, \{(u,S),(v,T)\} \mid (u,S) \in V_u, (v,T) \in V_v, uv \in E(G), S(uv) + T(uv) = j \,\}.$$

The structure $\mathsf{CFI}^*[\mathbb{Z}_{2^i}, G, U]$ can be seen as a vertex-coloured and edge-coloured directed graph (with an order on the colours), where two vertices receive the same colour if and only if they are $\preceq$-equivalent. This means that precisely vertices with the same origin receive the same colour. The other relations colour edges by the set of relations in which they are contained. Note that I_0 coincides with the edge relation of the CFI graph $\mathsf{CFI}[\mathbb{Z}_{2^i}, G, U]$. The additional relations are, apart from the preorder, already implicit in $\mathsf{CFI}[\mathbb{Z}_{2^i}, G, U]$ and are made explicit to ensure definability of certain properties in logics.

Non-isomorphic but $\equiv^{\mathrm{IM}}_{k,\mathbb{P}}$-equivalent CFI graphs were constructed using a class of regular base graphs, in which the degree, the girth, and the vertex-connectivity are simultaneously unbounded [23]. We will show that it suffices that the graph only satisfies these properties "locally". The *r-ball around a vertex $v \in V$* is the set of vertices with distance at most r to v.

▶ **Definition 13.** *Let G be a base graph and $r, d, g, c \in \mathbb{N}$. We say that G is* (r,d,g,c)-nice *if there is some vertex $w \in V(G)$ such that the r-ball W around w satisfies the following:*

1. *Every vertex in W has degree at least d.*
2. *Every cycle in G containing a vertex of W as length at least g.*
3. *For every set $V' \subseteq V(G)$ of size at most c, all vertices in $W \setminus V'$ are contained in the same connected component of $G - V'$.*
4. *For every set $V' \subseteq V(G)$ of size $c' \leq c$, there is at most one connected component $X \subseteq V(G)$ of $G - V'$ such that $G[X]$ has treewidth[1] larger than c'.*

▶ **Lemma 14.** *For every $n \in \mathbb{N}$, there is a planar graph G that is $(n, 2n, 2n, n)$-nice.*

Proof. We start with a complete $2n$-ary tree (with fixed root w) of depth $4n$. For every $i \geq 1$, the i-th level of the tree consists of $(2n)(2n-1)^{i-1}$ vertices. In particular, the tree has $(2n)(2n-1)^{4n-1}$ leaves. Next, we attach a grid of height $2n$ and width $(2n)(2n-1)^{4n-1}$ to the tree as follows: The i-th leaf from the left (according to the usual drawing of a tree in the plane) is identified with the i-th vertex of the grid in the first row. Denote this graph by G. It is easy to see that G is planar. We prove that G is $(n, 2n, 2n, n)$-nice, which is witnessed by the root w. Let W be the n-ball around w, that is, the set of vertices whose level is at most $n+1$ in the tree. By construction, every vertex in W has degree $2n$ and every cycle, in which a vertex of W is contained, has length at least $2n$ because the tree has depth $4n$.

For every vertex $u \in W$, there are at least $2n$ paths from u into the grid that are disjoint apart from u. Let $V' \subset V(G)$ be a set of at most n vertices. We show that all vertices in $W \setminus V'$ are connected in $G - V'$. Let $u, v \in W \setminus V'$. If there is a path from u to v only using vertices of the tree, we are done. Otherwise, there are at most n paths disjoint apart from u respectively v into the grid (because there were $2n$ such paths for u respectively v before removing n vertices). Let V_u and V_v be the sets of endpoints of these paths, i.e., sets of size at least n of vertices in the first row of the grid. Because there is no path between u and v

[1] For a definition of treewidth, the reader is referred to [5].

in the tree, at most $n-1$ vertices of the grid are removed in $G - V'$ (we count the leaves of the tree as vertices of the grid). By removing $n-1$ vertices from a grid of height $2n$ (and larger width) it is not possible to separate the sets V_u and V_v because they are of size at least n each. Hence, some vertex of V_u is connected to some vertex of V_v in $G - V'$ and thus u and v are connected in $G - V'$.

We finally show that at most one connected component of $G - V'$ is not an induced subgraph of a grid of height at most $|V'|$. First, we claim that all vertices of the tree are in the same connected component of $G - V'$ (again, we count the leaves as vertices of the grid). One easily sees that the argument above actually works for all vertices of the tree because for all vertices of the tree there are $2n$ disjoint paths into the grid. So there is a component containing all vertices of the tree and some vertices of the grid. Second, because the grid has height and length greater than n, by removing $|V'| \leq n$ vertices from G we can only "cut out" holes or corners of the grid. This means that the component containing the tree vertices also contains all grid vertices apart from the holes and corners cut out. Each of them contains at most $|V'|$ vertices per column and thus all these holes and corners are induced subgraphs of a grid of height $|V'|$. It is well-known [5] that grids of height at most $|V'|$ have treewidth at most $|V'|$ and the same holds for induced subgraphs of them. ◀

We now analyse properties of CFI structures over nice base graphs. The following proofs assume that the reader is familiar with the CFI construction. For more details we refer for example to [6, 15, 16, 23]. For some number $c \in \mathbb{N}$, a *c-orbit* of a structure $\mathfrak{A}$ is a maximal set of c-tuples of $\mathfrak{A}$ that are all related by an automorphism of $\mathfrak{A}$. That is, $\overline{x}, \overline{y} \in A^c$ are in the same orbit if and only if there is an automorphism φ of $\mathfrak{A}$ such that $\varphi(\overline{x}) = \overline{y}$. The set of c-orbits is a partition of A^c.

We often need isomorphisms of a particular kind between generalised CFI structures. We have seen in Lemma 5 that two CFI graphs $\mathsf{CFI}[\mathbb{Z}_{2^i}, G, U]$ and $\mathsf{CFI}[\mathbb{Z}_{2^i}, G, U']$ over some base graph G are isomorphic if, and actually only if, $\sum U = \sum U'$. The same reasoning applies to the CFI structures $\mathsf{CFI}^*[\mathbb{Z}_{2^i}, G, U]$ and $\mathsf{CFI}^*[\mathbb{Z}_{2^i}, G, U']$ (see also [23]). Let $p = u_1, \dots, u_m$ be a path in G and $j \in \mathbb{Z}_{2^i}$. Now we can construct an isomorphism φ between $\mathsf{CFI}^*[\mathbb{Z}_{2^i}, G, U]$ and $\mathsf{CFI}^*[\mathbb{Z}_{2^i}, G, U - ju_1 + ju_m]$ (where jv denotes the vector in $V(G)^{\mathbb{Z}_{2^i}}$ that has entry j at position v and is zero otherwise) such that φ is the identity map on all vertices whose origin is not contained in p. This isomorphism can be composed out of the maps constructed in Lemma 5 by following the path p. We call such isomorphisms *path-isomorphisms*. If p is a closed cycle, then the associated path-isomorphism is an automorphism of the structure, which we call *cycle-automorphism* (again see [23]).

▶ **Lemma 15.** *Let $i \in \mathbb{N}$, G be an (r, d, g, c)-nice ordered base graph, and $U \in \mathbb{Z}_{2^i}^{V(G)}$. Then two tuples of length $c' \leq c$ of $\mathsf{CFI}^*[\mathbb{Z}_{2^i}, G, U]$ are $\mathcal{C}^{3c'}$-equivalent if and only if they are in the same c'-orbit.*

Proof. We start with the following special case:

▷ Claim 16. Let $\overline{a} = \overline{\gamma}x$ and $\overline{b} = \overline{\gamma}y$ be tuples of length $c' \leq c$ of $\mathsf{CFI}^*[\mathbb{Z}_{2^i}, G, U]$. If $\overline{a}$ and $\overline{b}$ are $\mathcal{C}^{3c'}$-equivalent, then $\overline{a}$ and $\overline{b}$ are in the same c'-orbit.

Proof Sketch. The vertices x and y must have the same origin v because otherwise they are easily distinguished in $\mathcal{C}^3$. So let $x = (v, S)$ and $y = (v, T)$ for some $S, T \in \mathbb{Z}_{2^i}^{E(v)}$. To construct an automorphism π that pointwise fixes $\overline{\gamma}$ and maps x to y, we have to shift the edges $F := \{e \in E(v) \mid S(e) \neq T(e)\}$. Let B be the set of all origins of vertices in $\overline{\gamma}$. Let $\mathcal{P}$ be the partition of the edges F such that two edges are in the same part of $\mathcal{P}$ if and only if they lead into the same connected component of $G - B - \{v\}$. Such an automorphism π

exists if and only if every $P \in \mathcal{P}$ satisfies $\sum_{e \in P} S(e) - T(e) = 0$. Suppose this is not the case. At least two parts of $\mathcal{P}$ do not satisfy the condition, since $\sum S = \sum T$. Because G is nice, the corresponding connected component of at least one of the parts is an induced subgraph of a grid of height c'. Because non-isomorphic CFI graphs over base graphs of treewidth at most c' are not $\mathcal{C}^{c'+1}$-equivalent [5, 17, 19], the tuples $\overline{a}$ and $\overline{b}$ are not $\mathcal{C}^{3c'}$-equivalent, which is a contradiction. ◁

To prove the lemma, first note that if two tuples are in the same orbit, then they are equivalent in every logic. So it remains to prove the other direction. We show by induction on the length c' of the tuples $\overline{a}$ and $\overline{b}$ that if $\overline{a}$ and $\overline{b}$ are $\mathcal{C}^{3c'}$-equivalent, then they are in the same c'-orbit, i.e., there is an automorphism of $\mathsf{CFI}^*[\mathbb{Z}_{2^i}, G, U]$ that maps $\overline{a}$ to $\overline{b}$.

For $c' = 1$, the result follows from Claim 16 using γ as the empty tuple. For the inductive step, assume $\overline{a} = \overline{a}'x$ and $\overline{b} = \overline{b}'y$ are $\mathcal{C}^{3(c'+1)}$-equivalent. Then $\overline{a}'$ and $\overline{b}'$ are $\mathcal{C}^{3c'}$-equivalent. By induction, there exists an automorphism π' such that $\pi'(\overline{a}') = \overline{b}'$. Then the tuples $\pi'(\overline{a})$ and $\overline{b}$ agree on all entries except potentially the last one. They are $\mathcal{C}^{3(c'+1)}$-equivalent because logical formulas do not distinguish between tuples in the same orbit. By Claim 16, there is an automorphism π such that $\pi(\pi'(\overline{a})) = \overline{b}$. So $\overline{a}$ and $\overline{b}$ are in the same orbit. ◀

For a graph G, we call two sets $V, W \subseteq V(G)$ *adjacent* if there are $v \in V$ and $w \in W$ such that v and w are adjacent in G.

▶ **Lemma 17.** *Let $i \in \mathbb{N}$, G be an (r, d, g, c)-nice ordered base graph witnessed by a vertex $w \in V(G)$, and let $U \in \mathbb{Z}_{2^i}^{V(G)}$. Furthermore, let φ be an automorphism of $\mathsf{CFI}^*[\mathbb{Z}_{2^i}, G, U]$. If $\overline{x}$, $\overline{y}$, and $\overline{z}$ are tuples of $\mathsf{CFI}^*[\mathbb{Z}_{2^i}, G, U]$ such that*

1. *$|\overline{xyz}| \leq c$,*
2. *the sets of all origins of vertices in $\overline{x}$, $\overline{y}$, and $\overline{z}$, respectively, are pairwise not adjacent in G, and*
3. *all origins of vertices in $\overline{x}$ and $\overline{y}$ are contained in the $(r-1)$-ball around w,*

then $\overline{xyz}$, $\varphi(\overline{x})\overline{yz}$, and $\overline{x}\varphi(\overline{y})\overline{z}$ are in the same orbit of $\mathsf{CFI}^*[\mathbb{Z}_{2^i}, G, U]$.

The proof of Lemma 17 makes use of standard arguments for CFI graphs and cycle-automorphisms [23]. Such cycles can always be found for $\overline{x}$ and $\overline{y}$ because removing all origins of vertices in $\overline{x}$ and $\overline{y}$ does not disconnect G because G is nice.

▶ **Lemma 18.** *For every $k \in \mathbb{N}$, there are $r, d, g, c, i \in \mathbb{N}$ such that, for every (r, d, g, c)-nice ordered base graph G and every $U, U' \in \mathbb{Z}_{2^i}^V$ such that $\sum U = \sum U' + 2^{i-1}$, we have $\mathsf{CFI}^*[\mathbb{Z}_{2^i}, G, U] \equiv^{\mathrm{IM}}_{k,\{2\}} \mathsf{CFI}^*[\mathbb{Z}_{2^i}, G, U']$.*

Proof. The proof is based on a close inspection of the proof in [23]: For every $2m \leq k$, base graphs of degree at least $d(m, k-2m)$, girth at least $g(m, k-2m)$, and vertex-connectivity at least $c(m, k-2m)$ are considered (for the definitions of d, g, and c, see [23]). Of particular interest is the $r(m, k-2m)$-ball around some vertex, which we will see later. The CFI graphs are constructed over $\mathbb{Z}_{2^i}$, for some $i(m, k-2m) \in \mathbb{N}$. Define $d = d(k) := \max_{2m \leq k} d(m, k-2m)$ and define $g = g(k)$, $c = c(k)$, $r = r(k)$, and $i = i(k)$ analogously. We finally define $r' := 2r+4$ and $g' := \max\{r', g\}$.

Assume G is $\mathrm{a}(r', d, g', c)$-nice and ordered base graph and let $u \in V(G)$ be a vertex witnessing this. We call the r'-ball around u the *nice region* of G. Let $U, U' \in \mathbb{Z}_{2^i}^V$ with $\sum U = \sum U' + 2^{i-1}$ and consider $\mathfrak{A} := \mathsf{CFI}^*[\mathbb{Z}_{2^i}, G, U]$ and $\mathfrak{B} := \mathsf{CFI}^*[\mathbb{Z}_{2^i}, G, U']$. To prove $\mathfrak{A} \equiv^{\mathsf{IM}}_{k,\{2\}} \mathfrak{B}$, we show that Duplicator wins the characteristic 2 IM-game with k-pebbles $\mathcal{M}^{k,\{2\}}$ played on $\mathfrak{A}$ and $\mathfrak{B}$. Duplicator maintains as invariant that in position $\overline{v}, \overline{v}'$, there is an isomorphism $\varphi \colon \mathfrak{B} \to \mathfrak{B}'$ where $\mathfrak{B}' := \mathsf{CFI}[\mathbb{Z}_{2^i}, G, U'']$ for some $U'' \in \mathbb{Z}_{2^i}^V$ such that

1. $\varphi(\overline{v}') = \overline{v}$,
2. there is only a single vertex $w \in V$ such that $U(w) \neq U''(w)$ that we call *twisted*, and
3. the $(r+1)$-ball around w is contained in the nice region and does not contain the origin of a vertex in $\overline{v}$.

Clearly, the invariant holds initially. So assume that the invariant holds by the inductive hypothesis and that it is Spoiler's turn. Up to isomorphism, we can assume to play on $\mathfrak{A}$ and $\mathfrak{B}'$ in position $\overline{v}, \overline{v}$. Spoiler chooses an arity $2m \leq k$ and picks up $2m$ pebbles from $\mathfrak{A}$ and the corresponding ones (with the same labels) from $\mathfrak{B}'$. Duplicator picks the $2m$-orbit partition $\mathcal{P}$ of $(\mathfrak{A}, \overline{v})$, and the $2m$-orbit partition $\mathcal{P}'$ of $(\mathfrak{B}, \overline{v})$. We construct a suitable bijection $\mathcal{P} \to \mathcal{P}'$ using the techniques of [23]. If G was regular with degree at least $d(m, k-2m)$, of girth at least $g(m, k-2m)$, and of vertex-connectivity at least $c(m, k-2m)$, then there would indeed be a similarity matrix as required by the game [23]. One crucial property of base graphs with vertex-connectivity strictly larger than k is the following: Let $\overline{xy}$ be a tuple of $\mathfrak{A}$ of length at most k such that the set of all origins of vertices in $\overline{x}$ is not adjacent to the same set for $\overline{y}$. In this case, automorphisms can be applied independently, that is, if φ is an automorphism, then $\overline{xy}$ is in the same orbit as $\varphi(\overline{x})\overline{y}$, $\overline{x}\varphi(\overline{y})$, and $\varphi(\overline{xy})$. The construction of the similarity matrix in [23] heavily depends on this fact. However, non-trivial automorphisms are only applied to such parts of tuples, for which all entries are contained in the $r(m, k-2m)$-ball around the twisted vertex (called the "active region" in [23]). This still holds for the (r', d, g', c)-nice base graph G, if the $r(m, k-2m)$-ball around the twisted vertex w is contained in the nice region: Let $\overline{xyz}$ be a tuple of vertices of $\mathfrak{A}$ of length at most k such that the sets of all origins of vertices of $\overline{x}$, $\overline{y}$, and respectively $\overline{z}$ are pairwise not adjacent and the sets of all origins of vertices of $\overline{x}$ and $\overline{y}$ are contained within the r-ball around w. Then automorphisms can be applied independently in the sense above (Lemma 17). Hence, the same construction of the similarity matrix of [23] can also be applied here. All arguments requiring large girth and degree only consider vertices in the "active region", for which we also have long cycles and large degree in the nice region.

Spoiler places $2m$ pebbles on the vertices in a $2m$-tuple in some block $P \in \mathcal{P}$ and the corresponding ones on a $2m$-tuple in $f(P) \in \mathcal{P}'$ resulting in the position $\overline{v}''$ and $\overline{v}'''$. By the properties of the similarity matrix and the bijection from [23], the pebbles define a partial isomorphism, and there is an isomorphism $\psi \colon \mathfrak{B}' \to \mathfrak{B}''$ such that $\psi(\overline{v}''') = \overline{v}''$ and there is only a single twisted vertex between $\mathfrak{A}$ and $\mathfrak{B}''$. Hence, Conditions 1 and 2 of the invariant are satisfied.

To satisfy Condition 3, we move the twist to a vertex that has distance at least $r+2$ to the origins of all vertices in $\overline{v}$ using a path-isomorphism as follows. Let $O \subseteq V(G)$ be this set of at most k origins. Because G is nice, we can move the twist to all vertices in the nice region apart from whose in O (because removing the vertics in O does not separate the nice region). Since the nice region is an r'-ball around u and $g' \geq r'$, there are no cycles in the nice region. Hence, the nice region induces a tree T of height r' with root u where each non-leaf has degree at least d and every leaf has distance r' to u. We call a neighbour u' of u *blocked*, if the subtree $T_{u'}$ of T rooted at u' contains some vertex from O. Because $k < d$ [23], there is a neigbor u' of u that is not blocked. Hence, if a vertex w' in $T_{u'}$ has distance at least $r+1$ to u' in $T_{u'}$, then w' has distance at most $r+2$ in T to all vertices in O. Such a vertex w' of distance exactly $r+1$ to u' always exists in T beause every leaf has distance r' to u. Because $r' = 2r+4$, the $(r+1)$-ball around w' in G is contained in the nice region. Because $g' \geq r'$ and w' is in the nice region, w' has distance at least $r+2$ to each vertex of O in G. We move the twist to this vertex w'. Duplicator maintains the invariant and thus wins the invertible-map game. ◀

▶ **Lemma 19.** *For every $k \in \mathbb{N}$, there is a planar ordered base graph G and an $i \in \mathbb{N}$ such that, for all $U, U' \in \mathbb{Z}_{2^i}^{V(G)}$ with $\sum U = \sum U' + 2^{i-1}$, we have $\mathsf{CFI}^*[\mathbb{Z}_{2^i}, G, U] \equiv^{\mathrm{IM}}_{k,\mathbb{P}} \mathsf{CFI}^*[\mathbb{Z}_{2^i}, G, U']$.*

Proof. Let $k \in \mathbb{N}$ be arbitrary. Let r, d, g, c, and i be the constants given by Lemma 18 for $k' := 3k + 1$ and let $\ell := \max\{r, d, g, c\}$. By Lemma 14, there is a planar graph G that is $(\ell, 2\ell, 2\ell, \ell)$-nice. One easily sees that G is also (r, d, g, c)-nice. Hence,

$$\mathsf{CFI}^*[\mathbb{Z}_{2^i}, G, U] \equiv^{\mathsf{IM}}_{3k+1,\{2\}} \mathsf{CFI}^*[\mathbb{Z}_{2^i}, G, U']$$

by Lemma 18 for all $U, U' \in \mathbb{Z}_{2^i}^{V(G)}$ with $\sum U = \sum U' + 2^{i-1}$. By Lemma 15, the k'-orbits of these CFI structures are $\mathcal{C}^{3k'}$-definable and hence the class of CFI structures over $(\ell, 2\ell, 2\ell, \ell)$-nice and ordered base graphs is homogeneous in the sense of [8]. From [8] it follows that

$$\mathsf{CFI}^*[\mathbb{Z}_{2^i}, G, U] \equiv^{\mathsf{IM}}_{3k+1,\mathbb{P}\setminus\{2\}} \mathsf{CFI}^*[\mathbb{Z}_{2^i}, G, U'].$$

To show that these two equivalences imply

$$\mathsf{CFI}^*[\mathbb{Z}_{2^i}, G, U] \equiv^{\mathsf{IM}}_{k,\mathbb{P}} \mathsf{CFI}^*[\mathbb{Z}_{2^i}, G, U'],$$

we use the arguments from [7, Lemma 10]. The authors prove for $k'' = k + 2$ the following: If the k-orbits of two structures H and H' are definable in $\mathcal{C}^{k''}$ and for two sets of primes P and Q we have $H \equiv^{\mathsf{IM}}_{k''+1,P} H'$ and $H \equiv^{\mathsf{IM}}_{k''+1,Q} H'$, then $H \equiv^{\mathsf{IM}}_{k,P\cup Q} H'$. The same argument also applies for $k'' = 3k$ and the claim of the lemma is proven. ◀

Proof of Lemma 11. Because $\mathsf{CFI}[\mathbb{Z}_{2^i}, G, U]$ is up to renaming relation symbols a reduct of $\mathsf{CFI}^*[\mathbb{Z}_{2^i}, G, U]$ (only the relation I_0 is kept), $\mathsf{CFI}^*[\mathbb{Z}_{2^i}, G, U] \equiv^{\mathsf{IM}}_{k,\mathbb{P}} \mathsf{CFI}^*[\mathbb{Z}_{2^i}, G, U']$ (Lemma 19) implies $\mathsf{CFI}[\mathbb{Z}_{2^i}, G, U] \equiv^{\mathsf{IM}}_{k,\mathbb{P}} \mathsf{CFI}[\mathbb{Z}_{2^i}, G, U']$. ◀

5 Comonads

Certain comonads on the category of relational structures capture equivalences over certain fragments of first-order logic [1]. For example, the *pebbling comonad* $\mathbb{T}_k$ has the property that two structures $\mathfrak{A}$ and $\mathfrak{B}$ satisfy the same sentences over k-variable first-order logic with counting quantifiers if and only if they are isomorphic in the co-Kleisli-category of $\mathbb{T}_k$. We refer the reader to [11, 31] and the previously mentioned references for formal definitions. The following Lovász-type theorem for comonads allows us to derive Theorem 2 from Theorem 1:

▶ **Theorem 20** ([31])**.** *Let $\mathbb{C}$ be a finite-rank comonad on the category of (not necessarily finite) graphs. Then there exists a graph class $\mathcal{F}$ such that two finite graphs are isomorphic in the co-Kleisli category of $\mathbb{C}$ if and only if they are homomorphism indistinguishable over $\mathcal{F}$.*

For a definition of finite rank, see [31, Definition B.3]. Less generally, one may think of a finite-rank comonad as a comonad which sends finite structures to finite structures, cf. [11]. Note that Theorem 2 does not rule out that invertible-map equivalence can be characterised comondically in a different way, i.e., not as co-Kleisli isomorphism but via a more involved construction.

Proof of Theorem 2. Towards a contradiction, suppose that $\equiv^{\mathrm{IM}}_{k,\mathbb{P}}$ coincides with the isomorphism relation in the co-Kleisli category of some finite-rank comonad. Then, by Theorem 20, $\equiv^{\mathrm{IM}}_{k,\mathbb{P}}$ is a homomorphism indistinguishability relation contradicting Theorem 1. ◀

6 Modular Homomorphism Indistinguishability

In this section, we consider homomorphism indistinguishability modulo integers $n \in \mathbb{N}$. For a graph class $\mathcal{F}$, two graphs G and H are said to be *homomorphism indistinguishable over $\mathcal{F}$ modulo n*, in symbols $G \equiv^n_{\mathcal{F}} H$, if $\hom(F, G) \equiv \hom(F, H) \mod n$ for every $F \in \mathcal{F}$. We write $G \equiv^N_{\mathcal{F}} H$ for a set $N \subseteq \mathbb{N}$ if $G \equiv^n_{\mathcal{F}} H$ for every $n \in N$.

In contrary to the classical result of Lovász [24] asserting that two graphs are homomorphism indistinguishable over all graphs if and only if they are isomorphic, homomorphism counts modulo a prime p do not suffice to determine a graph up to isomorphism. In [14], homomorphism indistinguishability over all graphs modulo p was characterised as follows: For a graph G with automorphism σ, write G^σ for the subgraph of G induced by the fixed points of σ. Write $G \to_p G'$ for two graphs G and G' if there is an automorphism σ of G of order p such that $G^\sigma \cong G'$ and write $G \to^*_p H$ if there is a sequence of graphs $G_1, \dots, G_n$ such that $G \to_p G_1 \to_p G_2 \to_p \cdots \to_p G_n \to_p H$. By [14, Theorem 3.7], for every graph G and prime p, there is a graph G^*_p, unique up to isomorphism, such that G^*_p has no automorphisms of order p and $G \to^*_p G^*_p$. Furthermore, by [14, Theorem 3.4], G and G^*_p are homomorphism indistinguishable over all graphs modulo p. A characterisation of homomorphism indistinguishability over all graphs modulo p can now be stated as follows:

▶ **Theorem 21** ([14, Lemma 3.10]). *Let p be a prime. Two graphs G and H are homomorphism indistinguishable over all graphs modulo p if and only if G^*_p and H^*_p are isomorphic.*

In general, modular homomorphism indistinguishability relations are rather oblivious to striking differences between graphs:

▶ **Example 22.** For $n \in \mathbb{N}$, the one-vertex graph K_1 and the coclique $\overline{K_{n+1}}$ are homomorphism indistinguishable over all graphs modulo n, i.e. $\hom(F, K_1) \equiv \hom(F, \overline{K_{n+1}}) \mod n$ for all graphs F.

Proof. If F is an edgeless graph, then $\hom(F, K_1) = 1 \equiv (n+1)^{|V(F)|} = \hom(F, \overline{K_{n+1}}) \mod n$. If otherwise F contains an edge, then $\hom(F, K_1) = 0 = \hom(F, \overline{K_{n+1}})$. ◀

Before we move to modular homomorphism indistinguishability characterisations for certain logic fragments, we clarify the relationship between the various notions introduced so far:

▶ **Lemma 23.** *Let $\mathcal{F}$ and $\mathcal{K}$ be graph classes. Let $N \subseteq \mathbb{N}$ and $n \in \mathbb{N}$.*

1. *If N is infinite, then $\equiv^N_{\mathcal{F}}$ and $\equiv_{\mathcal{F}}$ coincide.*
2. *If N is finite and m is the least common multiple of the numbers in N, then $\equiv^N_{\mathcal{F}}$ and $\equiv^m_{\mathcal{F}}$ coincide.*
3. *If $\equiv_{\mathcal{F}}$ and $\equiv^n_{\mathcal{K}}$ coincide, then $\mathcal{F} = \emptyset$, i.e., all graphs are $\equiv_{\mathcal{F}}$-equivalent.*

Proof. For the first claim, let G and H be graphs and $F \in \mathcal{F}$. Since N is infinite, there exists $n \in N$ greater than $|V(G)|^{|V(F)|}$ and $|V(H)|^{|V(F)|}$. Then $\hom(F, G) \equiv \hom(F, H) \mod n$ implies that $\hom(F, G) = \hom(F, H)$.

For the second claim, first observe that $G \equiv^m_{\mathcal{F}} H$ entails $G \equiv^N_{\mathcal{F}} H$ since all $n \in N$ divide m. Conversely, for a prime p write $\nu(p)$ for the greatest integer $k \geq 0$ such that there is an $n \in N$ that is divisible by p^k. Then $m = \prod_{p \in \mathbb{P}} p^{\nu(p)}$, where the product ranges over all primes. Hence, if $\hom(F, G) \equiv \hom(F, H) \mod n$ for all $n \in N$, then $\hom(F, G) \equiv \hom(F, H) \mod p^{\nu(p)}$ for all primes p appearing as divisors of elements in N, i.e., $\nu(p) > 0$. Hence, by the Chinese Remainder Theorem [28, Theorem 2.10], also $\hom(F, G) \equiv \hom(F, H) \mod m$.

Towards the third claim, we first show the following Claim 24: Write ℓ for the maximum integer such that p^ℓ divides n for some prime p. Write $\varphi \colon \mathbb{N} \to \mathbb{N}$ for Euler's totient function [28, Section 2.3] and $G^{\times k}$ for the k-th categorical power of the graph G, cf. [25, p. 40].

▷ Claim 24. For every graph G, the graphs $G^{\times(\varphi(n)+\ell)}$ and $G^{\times\ell}$ are homomorphism indistinguishable over all graphs modulo n.

Proof. We show that $a^\ell(a^{\varphi(n)}-1) \equiv 0 \mod n$ for every $a \in \mathbb{N}$. By the Chinese Remainder Theorem [28, Theorem 2.10], writing $n = \prod p_i^{\ell_i}$ as product of powers of distinct primes, it suffices to show that this equality holds modulo $p_i^{\ell_i}$ for every i. By Euler's Theorem [28, Theorem 2.12], $a^{\varphi(p_i^{\ell_i})} \equiv 1 \mod p_i^{\ell_i}$ if a and p_i are coprime. Since $\varphi(n) = \prod \varphi(p_i^{\ell_i})$ [28, Theorem 2.7], also $a^{\varphi(n)} \equiv 1 \mod p_i^{\ell_i}$. If p_i divides a, then $a^\ell \equiv 0 \mod p_i^{\ell_i}$ as $\ell_i \leq \ell$. Finally, for every graph F, $\hom(F, G^{\times(\varphi(n)+\ell)}) = \hom(F,G)^{\varphi(n)+\ell} \equiv \hom(F,G)^\ell \mod n$ by [25, (5.30)]. ◁

Let $F \in \mathcal{F}$, $m := |V(F)|$, and write K_m for the clique on m vertices. Then $\hom(F, K_m) > 1$. Define $G := K_m^{\times(\varphi(n)+\ell)}$ and $H := K_m^{\times\ell}$. By [25, (5.30)] and $\varphi(n) \geq 1$, it holds that $\hom(F,G) = \hom(F,K_m)^{\varphi(n)+\ell} \neq \hom(F,K_m)^\ell = \hom(F,H)$. Hence, $G \not\equiv_{\mathcal{F}} H$. However, $G \equiv^n_{\mathcal{K}} H$ by Claim 24 contradicting that $\equiv_{\mathcal{F}}$ and $\equiv^n_{\mathcal{K}}$ coincide. ◀

Lemma 23 shows that non-trivial modular homomorphism indistinguishability relations cannot be expressed by (non-modular) homomorphism indistinguishability relations. Furthermore, considering sets of moduli does not yield more relations. We may restrict our attention to homomorphism indistinguishability relations modulo some not necessarily prime $n \in \mathbb{N}$. In the remainder of this section, we give an example and a non-example of a logic whose equivalence can be characterised as modular homomorphism indistinguishability relation.

We have seen already that the relation $\equiv^{\mathsf{IM}}_{k,\mathbb{P}}$ is not a homomorphism indistinguishability relation over any graph class. But since $\equiv^{\mathsf{IM}}_{k,\mathbb{P}}$ is a relation based on linear algebra over finite fields, it might a priori be that it can be characterised as a homomorphism indistinguishability relation *modulo a prime*. This can be ruled out, at least in the following sense:

▶ **Theorem 25.** *Let $k \geq 2$ and Q be a set of primes. Then there exists no graph class $\mathcal{F}$ and no $n \in \mathbb{N}$ such that $\equiv^{\mathsf{IM}}_{k,Q}$ and $\equiv^n_{\mathcal{F}}$ coincide.*

Proof. Towards a contradiction, suppose that $\equiv^{\mathsf{IM}}_{k,Q}$ and $\equiv^n_{\mathcal{F}}$ coincide for some graph class $\mathcal{F}$ and some $n \in \mathbb{N}$. Recall from Example 22 that the clique K_1 and the coclique $\overline{K_{n+1}}$ are homomorphism indistinguishable over all graphs modulo n. However K_1 and $\overline{K_{n+1}}$ are easily distinguished in 2-variable FO and thus $K_1 \not\equiv^{\mathsf{IM}}_{k,Q} \overline{K_{n+1}}$. ◀

By extending techniques of [13], we prove that homomorphism indistinguishability over graphs of bounded treewidth counted modulo a prime characterises equivalence in first-order logic with modular counting quantifiers. The strategy is to construct, for every graph F of treewidth $\leq k$ and every $m \in \mathbb{F}_p$, a modular counting logic formula with $\leq k+1$ variables such that a graph satisfies the formula if and only if it admits $m \mod p$ many homomorphisms from F. Conversely, counting logic formulas are translated into $\mathbb{F}_p$-linear combinations of graphs of bounded treewidth such that the linear combination of their homomorphism counts in a graph is $1 \mod p$ if and only if the formula is satisfied. In this direction, it is crucial that $\mathbb{F}_p$ is a field for an interpolation argument to carry through.

Modular counting logic is defined as follows: Let p be a prime. Formulas of $\mathcal{C}[p]$ are boolean combinations of atomic formulas, equality, and modular counting quantifiers $\exists^c x\varphi$ for every $c \in \mathbb{F}_p$. The semantics of modular counting quantifiers is as expected, i.e., a structure $\mathfrak{A}$ satisfies a sentence $\exists^c x\varphi(x)$ if there exist $c \mod p$ distinct $a \in A$ such that $\mathfrak{A} \models \varphi(a)$. Let $\mathcal{C}^{k+1}[p]$ denote the $(k+1)$-variable fragment of this logic.

▶ **Theorem 26.** *Let p be a prime and $k \geq 0$. Two arbitrary graphs G and H are homomorphism indistinguishable over all graphs of treewidth at most k modulo p if and only if G and H are $\mathcal{C}^{k+1}[p]$-equivalent.*

As a consequence, Theorem 26 in conjunction with Lemma 23 and Theorem 20 yields that $\mathcal{C}^{k+1}[p]$-equivalence cannot be characterised as a co-Kleisli isomorphism with respect to a finite-rank comonad.

7 Conclusion

We studied linear-algebraic logic, a logic stronger than first-order logic with counting, and proved that equivalence with respect to it can neither be characterised as a homomorphism indistinguishability relation, nor as co-Kleisli isomorphism for a finite-rank comonad. The latter answers an open question of Ó Conghaile and Dawar [30] and shows a limitation of the game comonad programme for capturing logical equivalences. It would be desirable to understand more generally which properties are responsible for making a logic suitable for a homomorphism indistinguishability or game comonad characterisation. We know that game comonads can be defined for FO with all Lindström quantifiers up to a fixed arity [30]. What we do not know is whether these are the only Lindström extensions of FO admitting such a characterisation. Other interesting classes of Lindström quantifiers to look at besides the linear-algebraic ones could be CSP quantifiers. The corresponding logic defined in [20] comes with a fairly natural game characterising equivalence. Thus, one may ask whether this CSP logic admits a game comonad or if this can be ruled out with similar methods as in this paper. The same question is also open for (bounded variable fragments of) counting monadic second order logic CMSO. In principle, our approach works for every extension of counting logic for which there exists a CFI-like lower bound construction that works over planar base graphs and with only one binary relation. It remains to devise such a construction for CSP logic and CMSO.

A different topic, that we have merely touched upon, is homomorphism counting in prime fields. We have shown that the corresponding homomorphism indistinguishability relations do not characterise IM-equivalence. On the other hand, we stated an example of a logic that is captured by a modular homomorphism indistinguishability relation, namely modular counting logic. A more comprehensive theory of modular homomorphism counting is yet to be developed. A particularly interesting question, which is not in the scope of this paper, is whether the known connections between homomorphism counting and solutions to semidefinite/linear programs for graph isomorphism [33] have a meaningful generalisation to prime fields.

References

1 Samson Abramsky, Anuj Dawar, and Pengming Wang. The pebbling comonad in Finite Model Theory. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavík, Iceland, June 20-23, 2017*, pages 1–12. IEEE Computer Society, 2017. doi:10.1109/LICS.2017.8005129.

2 Samson Abramsky, Tomáš Jakl, and Thomas Paine. Discrete Density Comonads and Graph Parameters. In Helle Hvid Hansen and Fabio Zanasi, editors, *Coalgebraic Methods in Computer Science*, pages 23–44, Cham, 2022. Springer International Publishing. doi:10.1007/978-3-031-10736-8_2.

3 Samson Abramsky and Nihil Shah. Relating structure and power: Comonadic semantics for computational resources. *Journal of Logic and Computation*, 31(6):1390–1428, September 2021. doi:10.1093/logcom/exab048.

4 Albert Atserias, Laura Mančinska, David E. Roberson, Robert Šámal, Simone Severini, and Antonios Varvitsiotis. Quantum and non-signalling graph isomorphisms. *J. Comb. Theory, Ser. B*, 136:289–328, 2019. doi:10.1016/j.jctb.2018.11.002.

5 Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1):1–45, December 1998. doi:10.1016/S0304-3975(97)00228-4.

6 Jin-Yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, December 1992. doi:10.1007/BF01305232.

7 Anuj Dawar, Erich Grädel, and Moritz Lichter. Limitations of the invertible-map equivalences. *J. Log. Comput.*, 33(5):961–969, 2023. doi:10.1093/logcom/exac058.

8 Anuj Dawar, Erich Grädel, and Wied Pakusa. Approximations of Isomorphism and Logics with Linear-Algebraic Operators. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 112:1–112:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2019.112.

9 Anuj Dawar, Martin Grohe, Bjarki Holm, and Bastian Laubner. Logics with Rank Operators. In *Proceedings of the 24th Annual IEEE Symposium on Logic in Computer Science, LICS 2009, 11-14 August 2009, Los Angeles, CA, USA*, pages 113–122. IEEE Computer Society, 2009. doi:10.1109/LICS.2009.24.

10 Anuj Dawar and Bjarki Holm. Pebble games with algebraic rules. In *39th International Colloquium on Automata, Languages, and Programming, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part II*, volume 7392 of *Lecture Notes in Computer Science*, pages 251–262. Springer, 2012. doi:10.1007/978-3-642-31585-5_25.

11 Anuj Dawar, Tomáš Jakl, and Luca Reggio. Lovász-Type Theorems and Game Comonads. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*, pages 1–13. IEEE, 2021. doi:10.1109/LICS52264.2021.9470609.

12 Holger Dell, Martin Grohe, and Gaurav Rattan. Lovász Meets Weisfeiler and Leman. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, volume 107 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 40:1–40:14, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2018.40.

13 Zdeněk Dvořák. On recognizing graphs by numbers of homomorphisms. *Journal of Graph Theory*, 64(4):330–342, August 2010. doi:10.1002/jgt.20461.

14 John Faben and Mark Jerrum. The Complexity of Parity Graph Homomorphism: An Initial Investigation. *Theory of Computing*, 11(2):35–57, 2015. doi:10.4086/toc.2015.v011a002.

15 Martin Fürer. Weisfeiler-Lehman refinement requires at least a linear number of iterations. In *28th International Colloquium on Automata, Languages, and Programming, ICALP 2001, Crete, Greece, July 8-12, 2001, Proceedings*, volume 2076 of *Lecture Notes in Computer Science*, pages 322–333. Springer, 2001. doi:10.1007/3-540-48224-5_27.

16 Erich Grädel and Wied Pakusa. Rank logic is dead, long live rank logic! *J. Symb. Log.*, 84(1):54–87, 2019. doi:10.1017/jsl.2018.33.

17 Martin Grohe, Moritz Lichter, Daniel Neuen, and Pascal Schweitzer. Compressing CFI graphs and lower bounds for the weisfeiler-leman refinements. *CoRR*, abs/2308.11970, 2023. to appear at FOCS 2023. doi:10.48550/ARXIV.2308.11970.

18 Martin Grohe, Gaurav Rattan, and Tim Seppelt. Homomorphism Tensors and Linear Equations. In Mikołaj Bojańczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, volume 229 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 70:1–70:20, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2022.70.

19 Lauri Hella. Logical hierarchies in PTIME. *Information and Computation*, 129(1):1–19, 1996. doi:10.1006/inco.1996.0070.

20 Lauri Hella. The Expressive Power of CSP-Quantifiers. In Bartek Klin and Elaine Pimentel, editors, *31st EACSL Annual Conference on Computer Science Logic (CSL 2023)*, volume 252 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 25:1–25:19, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.CSL.2023.25.

21 Bjarki Holm. *Descriptive complexity of linear algebra*. PhD thesis, University of Cambridge, 2011. URL: https://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.609435.

22 Phokion G. Kolaitis and Jouko A. Väänänen. Generalized quantifiers and pebble games on finite structures. *Annals of Pure and Applied Logic*, 74(1):23–75, June 1995. doi:10.1016/0168-0072(94)00025-X.

23 Moritz Lichter. Separating rank logic from polynomial time. *Journal of the ACM*, 70(2):1–53, 2023. doi:10.1145/3572918.

24 László Lovász. Operations with structures. *Acta Mathematica Academiae Scientiarum Hungarica*, 18(3):321–328, September 1967. doi:10.1007/BF02280291.

25 László Lovász. *Large networks and graph limits*. Number volume 60 in American Mathematical Society colloquium publications. American Mathematical Society, Providence, Rhode Island, 2012. doi:10.1090/coll/060.

26 Laura Mančinska and David E. Roberson. Quantum isomorphism is equivalent to equality of homomorphism counts from planar graphs. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 661–672, 2020. doi:10.1109/FOCS46700.2020.00067.

27 Yoàv Montacute and Nihil Shah. The Pebble-Relation Comonad in Finite Model Theory. In Christel Baier and Dana Fisman, editors, *LICS '22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2 - 5, 2022*, pages 13:1–13:11. ACM, 2022. doi:10.1145/3531130.3533335.

28 Melvyn B. Nathanson. *Elementary Methods in Number Theory*, volume 195 of *Graduate Texts in Mathematics*. Springer New York, New York, NY, 2000. doi:10.1007/b98870.

29 Daniel Neuen and Pascal Schweitzer. Benchmark graphs for practical graph isomorphism. In *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria*, volume 87 of *LIPIcs*, pages 60:1–60:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ESA.2017.60.

30 Adam Ó Conghaile and Anuj Dawar. Game Comonads & Generalised Quantifiers. In Christel Baier and Jean Goubault-Larrecq, editors, *29th EACSL Annual Conference on Computer Science Logic (CSL 2021)*, volume 183 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 16:1–16:17, Dagstuhl, Germany, 2021. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.CSL.2021.16.

31 Luca Reggio. Polyadic sets and homomorphism counting. *Advances in Mathematics*, 410:108712, December 2022. doi:10.1016/j.aim.2022.108712.

32 David E. Roberson. Oddomorphisms and homomorphism indistinguishability over graphs of bounded degree, 2022. arXiv:2206.10321.

33 David E. Roberson and Tim Seppelt. Lasserre Hierarchy for Graph Isomorphism and Homomorphism Indistinguishability. In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *50th International Colloquium on Automata, Languages, and Programming (ICALP 2023)*, volume 261 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 101:1–101:18, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2023.101.

34 Tim Seppelt. Logical Equivalences, Homomorphism Indistinguishability, and Forbidden Minors. In Jérôme Leroux, Sylvain Lombardy, and David Peleg, editors, *48th International Symposium on Mathematical Foundations of Computer Science (MFCS 2023)*, volume 272 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 82:1–82:15, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.MFCS.2023.82.

Confluence of Conditional Rewriting Modulo

Salvador Lucas
DSIC & VRAIN, Universitat Politècnica de València, Spain

Abstract

We investigate confluence of rewriting with *Equational Generalized Term Rewriting Systems* $\mathcal{R}$, consisting of *Horn clauses*, some of them defining *conditional equations* $s = t \Leftarrow c$ and *rewriting rules* $\ell \to r \Leftarrow c$. In both cases, c is a sequence of *atoms*, possibly defined by using additional Horn clauses. Such systems include *Equational Term Rewriting Systems* and (join, oriented, and semi-equational) *Conditional Term Rewriting Systems*. A set of equations E defines an equivalence $=_E$ and quotient set $\mathcal{T}(\mathcal{F},\mathcal{X})/=_E$ of terms, where reductions $s \to_{\mathcal{R}/E} t$ using rules in $\mathcal{R}$ occur. For such systems, we obtain a *finite* set of conditional pairs π, which can be viewed as logical sentences, to prove and disprove confluence of $\to_{\mathcal{R}/E}$ by (dis)proving joinability of such conditional pairs π.

2012 ACM Subject Classification Theory of computation → Automated reasoning; Theory of computation → Logic and verification; Theory of computation → Equational logic and rewriting

Keywords and phrases Conditional rewriting, Confluence, Program analysis

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.37

Funding *Salvador Lucas*: Supported by project PID2021-122830OB-C42 funded by MCIN/AEI/10.13039/501100011033 and by "ERDF A way of making Europe" and by the grant CIPROM/2022/6 funded by Generalitat Valenciana.

Acknowledgements I thank the anonymous referees for their helpful remarks.

1 Introduction

A sequence 0, $\mathsf{s}(0)$, $\mathsf{s}(\mathsf{s}(0))$ of numbers in Peano's notation is usually written as a *term* by using a "pairing" (binary) operator $++$ as in $t_1 = (0 ++ \mathsf{s}(0)) ++ \mathsf{s}(\mathsf{s}(0))$ or $t_2 = 0 ++ (\mathsf{s}(0) ++ \mathsf{s}(\mathsf{s}(0)))$. This is necessary when computing with (variants of) Term Rewriting Systems (TRSs [1]). However, *multiple* presentations of the sequence are possible. We can overcome this if $++$ is *associative*, i.e., the equation $xs ++ (ys ++ zs) = (xs ++ ys) ++ zs$ is satisfied for all terms xs, ys, and zs. Then, t_1 and t_2 are made *equivalent modulo associativity* and become members of an *equivalence class* $[t]$, consisting of all terms which are equivalent to t modulo associativity. Here, t can be t_1 or t_2, the specific choice being immaterial.

In general, if $\mathcal{T}(\mathcal{F},\mathcal{X})$ is the set of terms built from a signature $\mathcal{F}$ and variables in $\mathcal{X}$, a set of equations E on terms defines an equivalence $=_E$ and a partition $\mathcal{T}(\mathcal{F},\mathcal{X})/=_E$ of $\mathcal{T}(\mathcal{F},\mathcal{X})$ into equivalence classes. When additionally considering a set of rules $\mathcal{R}$, it is natural to view rewriting computations as transformations $[s]_E \to_{\mathcal{R}/E} [t]_E$ of *equivalence classes*. Here, $[s]_E \to_{\mathcal{R}/E} [t]_E$ (i.e., *rewriting modulo*) means that $s' \to_{\mathcal{R}} t'$ for some $s' \in [s]_E$ and $t' \in [t]_E$. We often just write $s \to_{\mathcal{R}/E} t$. In this paper we are interested in *E-confluence* of $\mathcal{R}$, i.e., the commutation of the following diagram:

$$\begin{array}{ccc} [s]_E & \xrightarrow[\mathcal{R}/E]{*} & [t]_E \\ {\scriptstyle \mathcal{R}/E}\downarrow{\scriptstyle *} & & {\scriptstyle \mathcal{R}/E}\vdots{\scriptstyle *} \\ [t']_E & \xrightarrow[\mathcal{R}/E]{*} & [u]_E \end{array}$$

In [10], Jouannaud addressed the problem of proving E-confluence of *equational term rewriting*, where E and $\mathcal{R}$ consist of (unconditional) equations and rewrite rules, respectively. In this paper we consider *conditional rules* $\ell \to r \Leftarrow c$ and *conditional equations* $s = t \Leftarrow d$, where c and d are sequences of *atoms*, possibly defined by Horn clauses.

▶ **Example 1.** The signature $\mathcal{F} = \{0, \mathsf{s}, +\!\!+\}$ can be used to represent nonempty sequences of natural numbers in Peano's notation. A single number is considered a sequence as well.

$$
\begin{array}{rcll}
xs +\!\!+ (ys +\!\!+ zs) & = & (xs +\!\!+ ys) +\!\!+ zs & (1)\\
\mathsf{Nat}(0) & & & (2)\\
\mathsf{Nat}(\mathsf{s}(n)) & \Leftarrow & \mathsf{Nat}(n) & (3)\\
x \approx y & \Leftarrow & x \to^* y & (4)
\end{array}
\qquad
\begin{array}{rcll}
0 + n & \to & n & (5)\\
\mathsf{s}(m) + n & \to & \mathsf{s}(m+n) & (6)\\
\mathsf{sum}(n) & \to & n \Leftarrow \mathsf{Nat}(n) & (7)\\
\mathsf{sum}(m +\!\!+ ns) & \to & m + n & \\
& \Leftarrow & \mathsf{Nat}(m), \mathsf{sum}(ns) \approx n & (8)
\end{array}
$$

Predicate Nat defined by clauses (2) and (3) identifies an expression (without variables) as representing a *natural number*; clause (4) describes the interpretation of conditions $s \approx t$ as *reachability* in conditional rules like (8). The application of a rule like (8) to a term $\mathsf{sum}(t)$ is as follows: for each substitution σ, if (i) $t =_E \sigma(m +\!\!+ ns)$, (ii) $\mathsf{Nat}(\sigma(m))$ holds, and (iii) $\mathsf{sum}(\sigma(ns))$ rewrites to $\sigma(n)$, then we obtain $\sigma(m) + \sigma(n)$. Note that associativity of $+\!\!+$ is essential to obtain the expected functionality of sum as it permits the "reorganization" of t into t', i.e., $\sigma(m) +\!\!+ \sigma(ns)$, so that, for the first member $\sigma(m)$ of t', $\mathsf{Nat}(\sigma(m))$ holds.

For the analysis of E-confluence of ETRSs, *E-critical pairs* were considered [10, Definition 10]. Given unconditional (variable disjoint) rules $\ell \to r$ and $\ell' \to r'$, a nonvariable position $p \in \mathcal{P}os(\ell)$ and an E-unifier θ such that $\theta(\ell|_p) =_E \theta(\ell')$, an E-critical pair $\langle \theta(\ell)[\theta(r')]_p, \theta(r)\rangle$ is obtained. However, in sharp contrast with TRSs [13, 9], (i) there is no general *E-unification algorithm* and *"for each equational theory one must invent a special algorithm"* [22, page 74]. Furthermore, even for E-unifying terms, (ii) there can be several, even *infinitely many* E-unifiers θ which must be considered to obtain a *complete set of E-critical pairs* which can be used to check E-confluence of $\mathcal{R}$ [2, 20]. In order to improve this situation, we propose the use of *Logic-based Conditional Critical Pairs* instead.

▶ **Example 2** (Continuing Example 1)**.** Terms $\mathsf{sum}(n)$ and $\mathsf{sum}(m +\!\!+ ns)$ syntactically unify with *mgu* $\theta = \{n \mapsto m +\!\!+ ns\}$. However, there are infinitely many E-unifiers $\theta_{a,b,c} = \{n \mapsto (\mathsf{s}^a(0) +\!\!+ \mathsf{s}^b(0)) +\!\!+ \mathsf{s}^c(0), m \mapsto \mathsf{s}^a(0), ns \mapsto (\mathsf{s}^b(0) +\!\!+ \mathsf{s}^c(0))\}$ for all $a, b, c \geq 0$ which *cannot* be seen as refinements $\tau \circ \theta$ of θ for some substitution τ (in the usual way). This leads to infinitely many (conditional) critical pairs for (7) and (8). Instead, a single *logic-based conditional critical pair* would represent them all:

$$\langle m' + n', n\rangle \quad \Leftarrow \quad \mathsf{sum}(n) = \mathsf{sum}(m' +\!\!+ ns'), \mathsf{Nat}(n), \mathsf{Nat}(m'), \mathsf{sum}(ns') \approx n' \qquad (9)$$

After some preliminary notions and notations (Section 2) and a summary of Jouannaud and Kirchner's results [11] we rely on (Section 3), the contributions of this paper are: (i) we introduce *Equational Generalized Term Rewriting Systems* (EGTRSs) $\mathcal{R}$ consisting of a set of *conditional equations* E and *conditional rules* R whose conditional parts are sequences of *atoms*, possibly defined by definite Horn clauses in a set H; then, (ii) rewriting computations (modulo) are described as *deduction* in a first-order theory obtained from E, H, and R (Section 4). After that, (iii) confluence of EGTRSs modulo is investigated by considering the structure of *peaks* that may lead to diverging computations. We distinguish between *rewriting* and *coherence* peaks and show that the first ones can be used to *disprove* confluence modulo (Sections 5 and 6). Also, (iv) we provide a *logic-based* definition of (conditional) critical pair which avoids the explicit computation of E-unifiers. We also show that other conditional pairs (namely, *conditional variable pairs* and *down conditional critical pairs*) are necessary to capture (non-)E-confluence of EGTRSs (Section 7). Finally, (v) we show that by using appropriate notions of joinability (modulo), such pairs permit to obtain proofs of E-confluence and non-E-confluence (Section 8). Section 9 discusses some related work. Section 10 concludes and points to some future work. For the sake of clarity, additional details about the analysis of confluence of $\mathcal{R}$ in Example 1 are supplied in Appendix A.

2 Preliminaries

In the following, *s.t.* means *such that* and *iff* means *if and only if*. We assume some familiarity with the basic notions of term rewriting [1, 19] and first-order logic [5, 18]. For the sake of readability, though, here we summarize the main notions and notations we use.

Abstract Reduction Relations. Given a binary relation $\mathsf{R} \subseteq A \times A$ on a set A, we often write $a\ \mathsf{R}\ b$ or $b\ \mathsf{R}^{-1}\ a$ instead of $(a,b) \in \mathsf{R}$. The *composition* of two relations R and R' is written $\mathsf{R} \circ \mathsf{R}'$ and defined as follows: for all $a, b \in A$, $a\ \mathsf{R} \circ \mathsf{R}'\ b$ iff there is $c \in A$ such that $a\ \mathsf{R}\ c$ and $c\ \mathsf{R}'\ b$. The *reflexive* closure of R is denoted by $\mathsf{R}^{=}$; the *transitive* closure of R is denoted by R^{+}; and the *reflexive and transitive* closure by R^{*}. An element $a \in A$ is R-*irreducible* (or just *irreducible* if no confusion arises) if there is no b such that $a\ \mathsf{R}\ b$. We say that $b \in A$ is R-reachable from $a \in A$ if $a\ \mathsf{R}^{*}\ b$. We say that $a, b \in \mathsf{R}$ are R-*joinable* if there is $c \in A$ such that $a\ \mathsf{R}^{*}\ c$ and $b\ \mathsf{R}^{*}\ c$. Also, $a, b \in \mathsf{R}$ are R-*convertible* if $a\ (\mathsf{R} \cup \mathsf{R}^{-1})^{*}\ b$. Given $a \in A$, if there is no infinite sequence $a = a_1\ \mathsf{R}\ a_2\ \mathsf{R}\ \cdots\ \mathsf{R}\ a_n\ \mathsf{R} \cdots$, then a is R-*terminating*; R is *terminating* if a is R-terminating for all $a \in A$. We say that R is (locally) *confluent* if, for all $a, b, c \in A$, if $a\ \mathsf{R}^{*}\ b$ and $a\ \mathsf{R}^{*}\ c$ (resp. $a\ \mathsf{R}\ b$ and $a\ \mathsf{R}\ c$), then b and c are R-joinable.

Signatures, Terms, Positions. In this paper, $\mathcal{X}$ denotes a countable set of *variables*. A *signature of symbols* is a set of *symbols* each with a fixed *arity*. We use $\mathcal{F}$ to denote a *signature of function symbols*, i.e., $\{f, g, \ldots\}$ whose arity is given by a mapping $ar : \mathcal{F} \to \mathbb{N}$. The set of terms built from $\mathcal{F}$ and $\mathcal{X}$ is $\mathcal{T}(\mathcal{F}, \mathcal{X})$. The set of variables occurring in t is $\mathcal{V}ar(t)$. Terms are viewed as labeled trees in the usual way. *Positions* p are represented by chains of positive natural numbers used to address subterms $t|_p$ of t. The *set of positions* of a term t is $\mathcal{P}os(t)$. The set of positions of a subterm s in t is denoted $\mathcal{P}os_s(t)$. The set of positions of non-variable symbols in t are denoted as $\mathcal{P}os_{\mathcal{F}}(t)$. Positions are ordered by the *prefix ordering* $\leq$ on sequences: given positions p, q, we write $p \leq q$ iff p is a prefix of q. If $p \not\leq q$ and $q \not\leq p$, we say that p and q are *disjoint* (written $p \parallel q$).

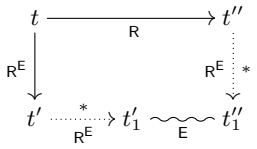
First-Order Logic. Here, Π denotes a signature of *predicate symbols*. First-order formulas are built using function symbols from $\mathcal{F}$, predicate symbols from Π, and variables from $\mathcal{X}$ in the usual way. In particular, *atomic formulas* A (often called *atoms* in the realm of automated theorem proving [23, page 2], but also in [12, pages 79 & 149]) are expressions $P(t_1, \ldots, t_n)$ where $P \in \Pi$ and $t_1, \ldots, t_n$ are terms; we often refer to P as $root(A)$.

A first-order theory (FO-theory for short) Th is a set of *sentences* (formulas whose variables are all *quantified*). In the following, given an FO-theory Th and a formula φ, $\mathsf{Th} \vdash \varphi$ means that φ is *deducible* from (or a *logical consequence* of) Th by using a correct and complete deduction procedure [5, 18]. A sequence $A_1, \ldots, A_n$ of atoms A_i, $1 \leq i \leq n$ is Th-*feasible* with respect to a theory Th (or just *feasible* if no confusion arises), if there is a substitution σ such that $\mathsf{Th} \vdash \sigma(A_i)$ holds for all $1 \leq i \leq n$; otherwise, it is *infeasible* [7].

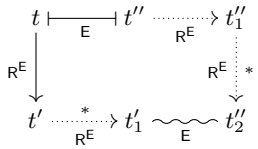
3 Abstract analysis of confluence of rewriting modulo

Following [11, Section 2], in this section $t, t', \ldots$ refer to elements of a set A. Let $\vdash\!\!\!\dashv_{\mathsf{E}}$ be a *symmetric* relation on A and $\sim_{\mathsf{E}}$ be its reflexive and transitive closure: an *equivalence* relation often called E-*equality*. Let $\to_{\mathsf{R}}$ (R for short) be a binary relation on A. Given R and E, the relation $\to_{\mathsf{R/E}}$ (R / E for short), is called *reduction (with* $\to_{\mathsf{R}}$*) modulo* $\sim_{\mathsf{E}}$ and *defined* as

$$\to_{\mathsf{R/E}} \quad = \quad \sim_{\mathsf{E}} \circ \to_{\mathsf{R}} \circ \sim_{\mathsf{E}} \tag{10}$$



Local confluence modulo E *of* R^{E} *with* R



Local coherence modulo E *of* R^{E}

Figure 1 Confluence and coherence properties: 3^{rd} and 5^{th} diagrams in [11, Figure 2.1].

▶ **Definition 3** (Confluence and termination of R modulo E). *Let* R *and* E *be as above. Then,*

- R *is* confluent modulo E *(or* E-confluent*) iff for all* t, $t_1, t_2 \in A$, *if* $t \to^*_{\mathsf{R}/\mathsf{E}} t_1$ *and* $t \to^*_{\mathsf{R}/\mathsf{E}} t_2$, *then there are* t'_1 *and* t'_2 *such that* $t_1 \to^*_{\mathsf{R}/\mathsf{E}} t'_1$, $t_2 \to^*_{\mathsf{R}/\mathsf{E}} t'_2$ *and* $t'_1 \sim_{\mathsf{E}} t'_2$ *[11, Def. 1].*[1]
- R *is* terminating modulo E *(or* E-terminating*) iff* $\to_{\mathsf{R}/\mathsf{E}}$ *is terminating [11, p. 1158].*

Computing with R / E is difficult as it may involve searching inside an infinite E-equivalence class $[t]_{\mathsf{E}}$ for some t' on which a R-reduction step can be performed. Peterson and Stickel investigated this problem for TRSs $\mathcal{R}$ and equational theories E. They introduced a reduction relation on terms, usually denoted $\to_{\mathcal{R},E}$, which can be advantageously used for this purpose [20]. In their abstract setting, Jouannaud and Kirchner use a relation $\to_{\mathsf{R}^{\mathsf{E}}}$ (R^{E} for short) satisfying the following *fundamental assumption* [11, page 1158]:

$$\mathsf{R} \subseteq \mathsf{R}^{\mathsf{E}} \subseteq \mathsf{R} \,/\, \mathsf{E} \tag{11}$$

Then, confluence of R / E is investigated by means of appropriate properties of R^{E}. As in [10, 11], we rely on the following related properties of (abstract) relations.

▶ **Definition 4.** *Consider* R, E, R^{E}, *and* R / E *as above, and* $t_1, t_2 \in A$. *A pair* $\langle t_1, t_2 \rangle$ *is*

1. R / E*-joinable* ($t_1 \downarrow_{\mathsf{R}/\mathsf{E}} t_2$), *iff* $\exists\, t'_1, t'_2$ *s.t.* $t_1 \to^*_{\mathsf{R}/\mathsf{E}} t'_1$, $t_2 \to^*_{\mathsf{R}/\mathsf{E}} t'_2$, *and* $t'_1 \sim_{\mathsf{E}} t'_2$.[2]
2. R^{E}*-joinable modulo* E, ($t_1 \downarrow_{\mathsf{R}^{\mathsf{E}}} t_2$), *iff* $\exists\, t'_1, t'_2$ *s.t.* $t_1 \to^*_{\mathsf{R}^{\mathsf{E}}} t'_1$, $t_2 \to^*_{\mathsf{R}^{\mathsf{E}}} t'_2$, *and* $t'_1 \sim_{\mathsf{E}} t'_2$ *[11, Def. 2].*
3. right-strict R^{E}*-joinable modulo* E, ($t_1 \downarrow^{rs}_{\mathsf{R}^{\mathsf{E}}} t_2$), *iff* $\exists\, t'_1, t'_2$ *s.t.* $t_1 \to^*_{\mathsf{R}^{\mathsf{E}}} t'_1$, $t_2 \to^+_{\mathsf{R}^{\mathsf{E}}} t'_2$, *and* $t'_1 \sim_{\mathsf{E}} t'_2$.

▶ **Definition 5** (Abstract confluence and coherence). *Consider* R, E, R^{E}, *and* R / E *as above. According to [11, Definition 3] (see Figure 1),*

1. R *is* R^{E}*-Church-Rosser modulo* E *iff for all* t *and* t', $t\,(\vdash\!\!\dashv_{\mathsf{E}} \cup \to_{\mathsf{R}} \cup \,{}_{\mathsf{R}}\!\leftarrow)^* t'$ *implies* $t \downarrow_{\mathsf{R}^{\mathsf{E}}} t'$.
2. R^{E} *is* locally confluent modulo E with R *iff for all* t, t', *and* t'', *if* $t \to_{\mathsf{R}^{\mathsf{E}}} t'$ *and* $t \to_{\mathsf{R}} t''$, *then* $t' \downarrow_{\mathsf{R}^{\mathsf{E}}} t''$.
3. R^{E} *is* locally coherent modulo E *iff for all* t, t', *and* t'', *if* $t \to_{\mathsf{R}^{\mathsf{E}}} t'$ *and* $t \vdash\!\!\dashv_{\mathsf{E}} t''$, *then* $t' \downarrow^{rs}_{\mathsf{R}^{\mathsf{E}}} t''$.

---

[1] Definition 1 in [11] does *not* use the last requirement $t'_1 \sim_{\mathsf{E}} t'_2$ as the authors assume t, t_1, and t_2 to be E*-equivalence classes* on A (i.e., $t, t_1, t_2 \in A/\!\sim_{\mathsf{E}}$) rather than $t, t_1, t_2 \in A$. In order to make the difference explicit, consider $A = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$, E be given by (the reflexive, transitive, and symmetric closure of) $\mathsf{b} \sim_{\mathsf{E}} \mathsf{c}$, and R be given by $\mathsf{a}\ \mathsf{R}\ \mathsf{b}$ and $\mathsf{a}\ \mathsf{R}\ \mathsf{c}$. Then, $\mathsf{a} \to_{\mathsf{R}/\mathsf{E}} \mathsf{b}$ and also $\mathsf{a} \to_{\mathsf{R}/\mathsf{E}} \mathsf{c}$, but b and c are $\to_{\mathsf{R}/\mathsf{E}}$-irreducible. And $\to^*_{\mathcal{R}/E} = \{(\mathsf{a},\mathsf{a})(\mathsf{b},\mathsf{b}), (\mathsf{c},\mathsf{c}), (\mathsf{a},\mathsf{b}), (\mathsf{a},\mathsf{c})\}$. Thus, neither $\mathsf{b} \to^*_{\mathcal{R}/E} \mathsf{c}$ nor $\mathsf{c} \to^*_{\mathcal{R}/E} \mathsf{b}$, i.e., as a relation on A, $\to_{\mathsf{R}/\mathsf{E}}$ is *not* confluent. However, $\mathsf{b} \sim_{\mathsf{E}} \mathsf{c}$. As a relation on $A/\!\sim_{\mathsf{E}}$, $\to_{\mathsf{R}/\mathsf{E}}$ is confluent.

[2] Continuing footnote 1, requiring this last equivalence step can be essential to achieve "joinability". For instance, this is necessary for b and c in the example of the footnote to be R / E-joinable.

▶ **Proposition 6** ([11, p. 1160, bullet 1]). *If* R *is* R^{E}*-Church-Rosser modulo* E*, then* R *is* E*-confluent.*

▶ **Theorem 7** ([11, Theorem 5]). *If* R *is* E*-terminating, then* R *is* R^{E}*-Church-Rosser modulo* E *iff* R^{E} *is (i) locally confluent modulo* E *with* R *and (ii) locally coherent modulo* E*.*

Theorem 7 and Proposition 6, yield a *sufficient condition* for E-confluence of R.

▶ **Corollary 8.** *If* R *is* E*-terminating, then* R *is* E*-confluent if* R^{E} *is (i) locally confluent with* R *modulo* E *and (ii) locally coherent modulo* E*.*

In the following, we investigate how to deal with the *abstract* peaks displayed in Figure 1:

$$t' \;{}_{\mathsf{R}^{\mathsf{E}}}\!\leftarrow\; t \;\rightarrow_{\mathsf{R}} t'' \qquad (12) \qquad t' \;{}_{\mathsf{R}^{\mathsf{E}}}\!\leftarrow\; t \;\vdash\!\!\!\rightarrow_{\mathsf{E}} t'' \qquad (13)$$

that we call R-peaks (12) and E-peaks (13), as they share the same leftmost part, with R^{E}, but differ on the rightmost part, with R and E, respectively. In general, *non*-joinability of these peaks does *not* entail non-E-confluence of R (Corollary 8 is just a sufficient condition for E-confluence). However, we have:

▶ **Proposition 9.** *If* t' *and* t'' *in (12) are not* R/E*-joinable, then* R *is not* E*-confluent.*

Note that coherence peaks (13) are trivially R/E-joinable, as $t'' \rightarrow_{\mathsf{R/E}} t'$.

4 Equational Generalized Term Rewriting Systems

The following definition introduces the kind of computational systems we consider here which can be viewed as an specialization of *Generalized Term Rewriting Systems* (GTRSs) introduced in [15] (see Section 9 for a more detailed comparison).

▶ **Definition 10** (Equational Generalized Term Rewriting Systems). *An* Equational Generalized Term Rewriting System (EGTRS) *is a tuple* $\mathcal{R} = (\mathcal{F}, \Pi, E, H, R)$ *where* $\mathcal{F}$ *is a signature of function symbols* Π *is a signature of predicate symbols with* $=, \rightarrow, \rightarrow^* \in \Pi$*, and, for* c *a sequence* $A_1, \ldots, A_n$ *of atomic formulas,*

- E *is a set of conditional equations* $s = t \Leftarrow c$*, for terms* s *and* t*;*
- H *is a set of definite Horn clauses* $A \Leftarrow c$ *where* $A = P(t_1, \ldots, t_n)$ *for some terms* $t_1, \ldots, t_n$*,* $n \geq 0$*, is such that* $P \notin \{=, \rightarrow, \rightarrow^*\}$*; and*
- R *is a set of conditional rules* $\ell \rightarrow r \Leftarrow c$ *for terms* $\ell \notin \mathcal{X}$ *and* r*.*

Note that $E \cup H \cup R$ *is a set of (definite) Horn clauses.*

Requiring $root(A) \notin \{=, \rightarrow, \rightarrow^*\}$ for all $A \Leftarrow c \in H$ ensures that computational predicates $=$, $\rightarrow$, and $\rightarrow^*$ are defined by E and R only (with an *auxiliary* use of H).

▶ Remark 11 (Conditions $s \approx t$ and their interpretation). In the literature about *Conditional TRSs* (CTRSs, see, e.g., [19, Chapter 7]), symbol $\approx$ is often used to specify conditions $s \approx t$ in rules having different *interpretations*: as *joinability*, *reachability*, etc. [19, Definition 7.1.3]. In EGTRSs, $\approx$ would be treated as a predicate in Π and the desired interpretation is *explicitly* obtained by including in H an appropriate set of clauses defining $\approx$. For instance, (4) in Example 1 interprets $\approx$ as *reachability*.

▶ **Notation 12.** *Equations* $s = t \Leftarrow c$ *in* E *are often* transformed *into rules by chosing a* left-to-right *or* right-to-left *orientation:* $\overrightarrow{E} = \{s \rightarrow t \Leftarrow c \mid s = t \Leftarrow c \in E\}$*, and* $\overleftarrow{E} = \{t \rightarrow s \Leftarrow c \mid s = t \Leftarrow c \in E\}$*. We let* $\overleftrightarrow{E} = \overrightarrow{E} \cup \overleftarrow{E}$*. Note that* $\overleftrightarrow{E}$ *may contain rules* $\lambda \rightarrow \rho \Leftarrow c$ *whose left-hand side* λ *is a* variable*. Let* $\mathcal{D}(\overleftrightarrow{E}) = \{root(\lambda) \mid \lambda \rightarrow \rho \Leftarrow d \in \overleftrightarrow{E}\}$*.*

Table 1 Generic sentences of the FO-theory of EGTRSs.

Label	Sentence
$(\mathrm{Rf})^{\bowtie}$	$(\forall x)\ x \bowtie x$
$(\mathrm{Tr})^{\bowtie}$	$(\forall x,y,z)\ x \bowtie y \wedge y \bowtie z \Rightarrow x \bowtie z$
$(\mathrm{Sy})^{\bowtie}$	$(\forall x,y)\ y \bowtie x \Rightarrow x \bowtie y$
$(\mathrm{Co})^{\bowtie}$	$(\forall x,y,z)\ x \bowtie y \wedge y \bowtie^* z \Rightarrow x \bowtie^* z$
$(\mathrm{Pr})^{\bowtie}_{f,i}$	$(\forall x_1,\dots,x_k,y_i)\ x_i \bowtie y_i \Rightarrow f(x_1,\dots,x_i,\dots,x_k) \bowtie f(x_1,\dots,y_i,\dots,x_k)$
$(\mathrm{HC})_{A \Leftarrow A_1,\dots,A_n}$	$(\forall \vec{x})\ A_1 \wedge \dots \wedge A_n \Rightarrow A$
$(\mathrm{R,E})_{\ell \to r \Leftarrow A_1,\dots,A_n}$	$(\forall x,\vec{x})\ x = \ell \wedge A_1 \wedge \dots \wedge A_n \Rightarrow x \overset{ps}{\to} r$
(R/E)	$(\forall x,x',y,y')\ x = x' \wedge x' \to y' \wedge y' = y \Rightarrow x \overset{rm}{\to} y$

▶ **Definition 13.** *We say that* $P \in \Pi$ depends on R *if* $P \in \{\to, \to^*\}$ *or there is* $A \Leftarrow A_1,\dots,A_n \in E \cup H$ *with* $root(A) = P$ *such that* $root(A_i)$ *depends on* R *for some* $1 \le i \le n$.

In this paper, *computational relations* (e.g., $=_E$, $\to_{\mathcal{R}}$, $\to_{\mathcal{R},E}$, $\to_{\mathcal{R}/E}$,...) induced by an EGTRS $\mathcal{R} = (\mathcal{F}, \Pi, E, H, R)$ are defined by deduction of atoms $s = t$ (equality in E), $s \to t$ (one-step rewriting in the usual sense), $s \overset{ps}{\to} t$ (rewriting modulo à la Peterson & Stickel), $s \overset{rm}{\to} t$ (rewriting modulo), etc., in some FO-theory. We *extend* Π with $\overset{ps}{\to}$, $\overset{rm}{\to}$, etc., and also $\approx_{ps}$, $\approx_{rm}$ (as they depend on the previous predicates). Our FO-theories are obtained from the generic sentences in Table 1, where:

- Sentences $(\mathrm{Rf})^{\bowtie}$, $(\mathrm{Tr})^{\bowtie}$, and $(\mathrm{Sy})^{\bowtie}$, which are parametric on a binary relation $\bowtie$, express *reflexivity*, *transitivity*, and *symmetry* of $\bowtie$, respectively;
- $(\mathrm{Co})^{\bowtie}$ expresses *compatibility* of one-step and many-step reduction with $\bowtie$;
- for each k-ary function symbol f, $1 \le i \le k$, and $x_1,\dots,x_k$ and y_i distinct variables, $(\mathrm{Pr})^{\bowtie}_{f,i}$ *propagates* an $\bowtie$-step to the i-th immediate subterm of an f-rooted term;
- $(\mathrm{HC})_{\alpha}$ presents a clause $\alpha : A \Leftarrow A_1,\dots,A_n$, with variables $\vec{x}$ as a sentence.
- $(\mathrm{R,E})_{\alpha}$ defines a Peterson & Stickel rewriting step $s \to_{\mathcal{R},E} t$ (at the root) using rule $\alpha : \ell \to r \Leftarrow c$ with variables $\vec{x}$. Here, $x \notin \vec{x}$.
- (R/E) defines reduction modulo $\to_{\mathcal{R}/E}$ in the usual way.

The following example illustrates the differences between (i) rewriting with $\to_{\mathcal{R}}$ (where a term t is rewritten if $t|_p = \sigma(\ell)$ for some position p in t, rule $\ell \to r \Leftarrow c$ in $\mathcal{R}$, and substitution σ such that $\sigma(c)$ holds), (ii) rewriting modulo $\to_{\mathcal{R}/E}$ (where a term t is rewritten if it is E-equivalent to another term t' to which a rewrite rule applies as above), and (iii) rewriting *à la Peterson & Stickel* $\to_{\mathcal{R},E}$ (where a term t is rewritten if some subterm $t|_p$ is E-equivalent to an instance $\sigma(\ell)$ of the left-hand side ℓ of a rewrite rule $\ell \to r \Leftarrow c$ and $\sigma(c)$ holds).

▶ **Example 14.** Consider $E = \{(14),(15)\}$ and $\mathcal{R} = (\mathcal{F}, R)$ with $R = \{(16),(17)\}$, where

$$\mathsf{b} = \mathsf{f(a)} \quad (14) \qquad \mathsf{c} \to \mathsf{d} \quad (16)$$
$$\mathsf{a} = \mathsf{c} \quad (15) \qquad \mathsf{b} \to \mathsf{d} \quad (17)$$

Then, $\mathsf{f(a)}$ is $\to_{\mathcal{R}}$-irreducible. However, $\mathsf{f(a)} \to_{\mathcal{R},E} \mathsf{f(d)}$ because $\mathsf{a} =_E \underline{\mathsf{c}} \to_{\mathcal{R}} \mathsf{d}$. Furthermore, $\mathsf{f(a)} \to_{\mathcal{R}/E} \mathsf{f(d)}$ because $\mathsf{f(a)} =_E \mathsf{f}(\underline{\mathsf{c}}) \to_{\mathcal{R}} \mathsf{f(d)}$. However, $\mathsf{f(a)} \to_{\mathcal{R}/E} \mathsf{d}$ because $\mathsf{f(a)} =_E \underline{\mathsf{b}} \to_{\mathcal{R}} \mathsf{d}$, but $\mathsf{f(a)} \not\to_{\mathcal{R},E} \mathsf{d}$ because $\mathsf{f(a)}$ is not E-equivalent to the left-hand side of any rule.

Now, consider the following *parametric* theories with parameters $\mathbb{S}$ (referring to a signature), $\mathbb{E}$ (a set of equations), and $\mathbb{R}$ (a set of rules):

$$\begin{array}{rcl}
\mathsf{Th}_{\mathrm{Eq}}[\mathbb{S},\mathbb{E}] & = & \{(\mathrm{Rf})^{=},(\mathrm{Sy})^{=},(\mathrm{Tr})^{=}\}\cup\{(\mathrm{Pr})^{=}_{f,i}\mid f\in\mathbb{S},1\le i\le ar(f)\}\cup\{(\mathrm{HC})_e\mid e\in\mathbb{E}\}\\
\mathsf{Th}_{\mathrm{R}}[\mathbb{S},\mathbb{R}] & = & \{(\mathrm{Rf})^{\rightarrow^*},(\mathrm{Co})^{\rightarrow}\}\cup\{(\mathrm{Pr})^{\rightarrow}_{f,i}\mid f\in\mathbb{S},1\le i\le ar(f)\}\cup\{(\mathrm{HC})_\alpha\mid \alpha\in\mathbb{R}\}\}\\
\mathsf{Th}_{\mathrm{R,M}}[\mathbb{S},\mathbb{R}] & = & \{(\mathrm{Rf})\xrightarrow{ps}{}^{*},(\mathrm{Co})^{\overset{ps}{\rightarrow}}\}\cup\{(\mathrm{Pr})^{\overset{ps}{\rightarrow}}_{f,i}\mid f\in\mathbb{S},1\le i\le ar(f)\}\cup\{(\mathrm{R,E})_\alpha\mid \alpha\in\mathbb{R}\}\\
\mathsf{Th}_{\mathrm{R/M}}[\mathbb{S},\mathbb{R}] & = & \{(\mathrm{Rf})\xrightarrow{rm}{}^{*},(\mathrm{Co})^{\overset{rm}{\rightarrow}},\}\cup\{(\mathrm{Pr})^{\rightarrow}_{f,i}\mid f\in\mathbb{S},1\le i\le ar(f)\}\cup\{(\mathrm{HC})_\alpha\mid \alpha\in\mathbb{R}\}\}\cup\{(\mathrm{R/E})\}
\end{array}$$

Note that, in $\mathsf{Th}_{\mathrm{R/M}}[\mathbb{S},\mathbb{R}]$ propagation sentences are given for $\rightarrow$ rather than for $\overset{rm}{\rightarrow}$; this is consistent with the usual definition of rewriting modulo implemented by (R/E).

Since rules $\alpha:\ell\to r\Leftarrow c$ in R are used to specify *different* computational relations (see Definition 17), conditions $s\approx t\in c$ have different *interpretations* depending on the targetted relation: the meaning of $\approx$ is based on predicate $\rightarrow$ if α is used to describe the usual (conditional) rewrite relation $\rightarrow_{\mathcal{R}}$; however, $\approx$ should be treated using $\overset{ps}{\rightarrow}$ if α is used to describe Peterson & Stickel's rewriting modulo $\rightarrow_{\mathcal{R},E}$; and $\approx$ should be treated using $\overset{rm}{\rightarrow}$ if α is used to describe $\rightarrow_{\mathcal{R}/E}$. A simple way to deal with this situation is the following.

▶ **Definition 15.** *Let $\mathcal{R}=(\mathcal{F},\Pi,E,H,R)$ be an EGTRS.*

- *H^{ps} (resp. H^{rm}) is obtained from H by replacing in each $A\Leftarrow A_1,\ldots,A_n\in H$ all occurrences of $\approx$, $\rightarrow$ and $\rightarrow^*$ by $\approx_{ps}$, $\overset{ps}{\rightarrow}$ and $\xrightarrow{ps}{}^*$ (resp. $\approx_{rm}$, $\overset{rm}{\rightarrow}$, and $\xrightarrow{rm}{}^*$).*
- *R^{ps} (R^{rm}) is obtained from R by replacing all occurrences of $\approx$ in each $\ell\to r\Leftarrow c\in R$ by $\approx_{ps}$ ($\approx_{rm}$).*

▶ **Example 16.** For $\mathcal{R}$ in Example 1, H^{rm} and R^{rm} are (H^{ps} and R^{ps} are similar):

$$\begin{array}{rcll}
\mathsf{Nat}(0) & & & (18)\\
\mathsf{Nat}(\mathsf{s}(n)) & \Leftarrow & \mathsf{Nat}(n) & (19)\\
x\approx_{rm} y & \Leftarrow & x\xrightarrow{rm}{}^* y & (20)
\end{array}
\qquad
\begin{array}{rcll}
0+n & \to & n & (21)\\
\mathsf{s}(m)+n & \to & \mathsf{s}(m+n) & (22)\\
\mathsf{sum}(n) & \to & n\Leftarrow\mathsf{Nat}(n) & (23)\\
\mathsf{sum}(m:{:}ns) & \to & m+n\Leftarrow\mathsf{Nat}(m),\mathsf{sum}(ns)\approx_{rm} n & (24)
\end{array}$$

Given an EGTRS $\mathcal{R}=(\mathcal{F},\Pi,E,H,R)$ whose equality predicate $=$ does *not* depend on R,[3] the following theories are obtained:

$$\begin{array}{rcl}
\mathsf{Th}_{E} & = & \mathsf{Th}_{\mathrm{Eq}}[\mathcal{F},E]\cup\{(\mathrm{HC})_\alpha\mid\alpha\in H\}\\
\mathsf{Th}_{\mathcal{R}} & = & \mathsf{Th}_{\mathrm{Eq}}[\mathcal{F},E]\cup\mathsf{Th}_{\mathrm{R}}[\mathcal{F},R]\cup\{(\mathrm{HC})_\alpha\mid\alpha\in H\}\\
\mathsf{Th}_{\mathcal{R},E} & = & \mathsf{Th}_{\mathrm{Eq}}[\mathcal{F},E]\cup\mathsf{Th}_{\mathrm{R,M}}[\mathcal{F},R^{ps}]\cup\{(\mathrm{HC})_\alpha\mid\alpha\in H^{ps}\}\\
\mathsf{Th}_{\mathcal{R}/E} & = & \mathsf{Th}_{\mathrm{Eq}}[\mathcal{F},E]\cup\mathsf{Th}_{\mathrm{R/M}}[\mathcal{F},R^{rm}]\cup\{(\mathrm{HC})_\alpha\mid\alpha\in H^{rm}\}
\end{array}$$

These theories are used to define the expected computational relations as follows.

▶ **Definition 17.** *Let $\mathcal{R}=(\mathcal{F},\Pi,E,H,R)$ be an EGTRS and $s,t\in\mathcal{T}(\mathcal{F},\mathcal{X})$.*

- *We write $s=_E t$ (resp. $s\to_{\overset{\leftrightarrow}{E}} t$) iff $\mathsf{Th}_E\vdash s=t$ (resp. $\mathsf{Th}_R[\mathcal{F},\overset{\leftrightarrow}{E}]\vdash s\to t$) holds.*
- *We write $s\to_{\mathcal{R}} t$ (resp. $s\to_{\mathcal{R},E} t$ and $s\to_{\mathcal{R}/E} t$) iff $\mathsf{Th}_{\mathcal{R}}\vdash s\to t$ (resp. $\mathsf{Th}_{\mathcal{R},E}\vdash s\overset{ps}{\to} t$ and $\mathsf{Th}_{\mathcal{R}/E}\vdash s\overset{rm}{\to} t$) holds. Similarly for $s\to^*_{\mathcal{R}} t$ (resp. $s\to^*_{\mathcal{R},E} t$ and $s\to^*_{\mathcal{R}/E} t$).*

▶ **Definition 18** (Confluence and termination modulo of an EGTRS). *Let $\mathcal{R}=(\mathcal{F},\Pi,E,H,R)$ be an EGTRS. We say that*

- *$\mathcal{R}$ is* confluent modulo *E (or E-*confluent*) iff for all terms t, t_1, and t_2, if $t\to^*_{\mathcal{R}/E} t_1$ and $t\to^*_{\mathcal{R}/E} t_2$, then there are t_1' and t_2' such that $t_1\to^*_{\mathcal{R}/E} t_1'$, $t_2\to^*_{\mathcal{R}/E} t_2'$ and $t_1'=_E t_2'$.*
- *$\mathcal{R}$ is* terminating modulo *E (or E-*terminating*) iff $\to_{\mathcal{R}/E}$ is terminating.*

[3] Requiring that $=$ does not depend on R implies that the "meaning" of equational atoms $s=t$ does *not* depend on the *rules* in R.

Table 2 Abstract notions in Section 3 applied to EGTRSs.

Abstract reduction:	$\vdash_{\mathsf{E}}$	$\sim_{\mathsf{E}}$	$\to_{\mathsf{R}}$	$\to_{\mathsf{R}^{\mathsf{E}}}$	$\to_{\mathsf{R}/\mathsf{E}}$
Application to EGTRSs:	$\to_{\overset{\leftrightarrow}{E}}$	$=_E$	$\to_{\mathcal{R}^{rm}}$	$\to_{\mathcal{R}^{rm},E}$	$\to_{\mathcal{R}/E}$

Note that (i) $\to_{\overset{\leftrightarrow}{E}}$ is *symmetric* by definition of $\overset{\leftrightarrow}{E}$; (ii) $=_E$ is an equivalence due to $(\mathrm{Rf})^{=}$ (reflexivity), $(\mathrm{Sy})^{=}$ (symmetry), and $(\mathrm{Tr})^{=}$ (transitivity), all included in Th_E; and (iii) $=_E$ is the reflexive and transitive closure of $\to_{\overset{\leftrightarrow}{E}}$. Unfortunately, the relationship between $\to_{\mathcal{R}}$, $=_E$, and $\to_{\mathcal{R}/E}$, is *not* as required. In particular, $\to_{\mathcal{R}/E} = (=_E \circ \to_{\mathcal{R}} \circ =_E)$ does *not* hold.

▶ **Example 19.** Consider the following EGTRS

$$\mathsf{a} = \mathsf{b} \quad (25) \qquad \mathsf{a} \to \mathsf{c} \quad (27)$$
$$x \approx y \Leftarrow x \to^* y \quad (26) \qquad \mathsf{a} \to \mathsf{d} \Leftarrow \mathsf{b} \approx \mathsf{c} \quad (28)$$

We have $\mathsf{a} \to_{\mathcal{R}} \mathsf{c}$; but (28) is $\mathsf{Th}_{\mathcal{F},\mathcal{R}}$-infeasible, hence $\mathsf{a} \not\to_{\mathcal{R}} \mathsf{d}$. However, $\mathsf{a} \to_{\mathcal{R}/E} \mathsf{d}$, as $\mathsf{b} =_E \mathsf{a} \to_{\mathcal{R}} \mathsf{c}$, i.e., $\mathsf{b} \to_{\mathcal{R}/E} \mathsf{c}$ and (28) can be used. Thus, $\to_{\mathcal{R}} = \{(\mathsf{a},\mathsf{c})\}$, $(=_E \circ \to_{\mathcal{R}} \circ =_E) = \{(\mathsf{a},\mathsf{c}),(\mathsf{b},\mathsf{c})\}$, and $\to_{\mathcal{R}/E} = \{(\mathsf{a},\mathsf{c}),(\mathsf{b},\mathsf{c}),(\mathsf{a},\mathsf{d}),(\mathsf{b},\mathsf{d})\}$, i.e., $\to_{\mathcal{R}/E} \neq (=_E \circ \to_{\mathcal{R}} \circ =_E)$.

▶ Remark 20 (Rewriting modulo and rewriting in conditional systems). Example 19 shows a *mismatch* between the definition of $\to_{\mathcal{R}/E}$ for an EGTRS $\mathcal{R}$ (Definition 17) and the *abstract* definition (10), usually understood for ETRSs. For EGTRSs (and already for CTRSs), the connection $\to_{\mathcal{R}/E} = (=_E \circ \to_{\mathcal{R}} \circ =_E)$ is *lost*. This is because the conditions in rules are treated using, e.g., $\to^*_{\mathcal{R}}$ to obtain $\to_{\mathcal{R}}$ (see, e.g., [19, Definition 7.1.4]), whereas computations with $\to_{\mathcal{R}/E}$ evaluate conditions using $\to^*_{\mathcal{R}/E}$ instead, see, e.g., [4, page 819].

We overcome this problem as follows.

▶ **Definition 21** (CR-theory of an EGTRS). *Let $\mathcal{R} = (\mathcal{F}, \Pi, E, H, R)$ be an EGTRS. The CR-theory of $\mathcal{R}$ is*

$$\overline{\mathcal{R}^{\mathrm{CR}}} = \mathsf{Th}_{Eq}[\mathcal{F}, E] \cup \mathsf{Th}_{R}[\mathcal{F}, R^{rm}] \cup \mathsf{Th}_{R,M}[\mathcal{F}, R^{rm}] \cup \mathsf{Th}_{R/M}[\mathcal{F}, R^{rm}] \cup \{(HC)_\alpha \mid \alpha \in H^{rm}\}$$

Then, $\to_{\mathcal{R}^{rm}}$ and $\to_{\mathcal{R}^{rm},E}$ (and also $\to^*_{\mathcal{R}^{rm}}$ and $\to^*_{\mathcal{R}^{rm},E}$) are defined as follows:

$$s \to_{\mathcal{R}^{rm}} t \;\Leftrightarrow\; \overline{\mathcal{R}^{\mathrm{CR}}} \vdash s \to t \qquad \textit{and} \qquad s \to_{\mathcal{R}^{rm},E} t \;\Leftrightarrow\; \overline{\mathcal{R}^{\mathrm{CR}}} \vdash s \overset{ps}{\to} t$$

In contrast to $\to_{\mathcal{R}}$ and $\to_{\mathcal{R},E}$, to obtain $\to_{\mathcal{R}^{rm}}$ and $\to_{\mathcal{R}^{rm},E}$ conditions in rules are evaluated using $\to_{\mathcal{R}/E}$ (instead of $\to_{\mathcal{R}}$ and $\to_{\mathcal{R},E}$). Now, the requirements (10) and (11) are fulfilled.

▶ **Proposition 22.** *Let $\mathcal{R} = (\mathcal{F}, \Pi, E, H, R)$ be an EGTRS. Then, $\to_{\mathcal{R}/E} = (=_E \circ \to_{\mathcal{R}^{rm}} \circ =_E)$. Also, $\to_{\mathcal{R}^{rm}} \subseteq \to_{\mathcal{R}^{rm},E} \subseteq \to_{\mathcal{R}/E}$.*

▶ Remark 23 (Use of Jouannaud & Kirchner's abstract framework). By the first statement in Proposition 22, E-confluence of EGTRSs (Definition 18) and E-confluence of $\to_{\mathcal{R}^{rm}}$ (as an abstract relation on terms, Definition 3) *coincide*. This enables the use of the results of Section 3 to analyze E-confluence of EGTRSs.

The results in Section 3 apply to EGTRSs according to the correspondence in Table 2. As a consequence of Proposition 6 and Theorem 7 we have the following.

▶ **Corollary 24.** *Let $\mathcal{R} = (\mathcal{F}, \Pi, E, H, R)$ be an EGTRS. If $\mathcal{R}$ is E-terminating and $\to_{\mathcal{R}^{rm},E}$ is locally confluent modulo E with $\to_{\mathcal{R}^{rm}}$ and locally coherent modulo E, then $\mathcal{R}$ is E-confluent.*

The following result is essential in the analysis of peaks in Sections 5 and 6.

▶ **Proposition 25.** *Let $\mathcal{R} = (\mathcal{F}, \Pi, E, H, R)$ be a EGTRS and $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$.*

- *If $s \to_{\mathcal{R}^{rm}} t$, then there is $p \in \mathcal{P}os(s)$ and $\ell \to r \Leftarrow c \in R^{rm}$ such that (i) $s|_p = \sigma(\ell)$ for some substitution σ, (ii) for all $A \in c$, $\overline{\mathcal{R}^{\mathrm{CR}}} \vdash \sigma(A)$ holds, and (iii) $t = s[\sigma(r)]_p$.*
- *If $s \to_{\mathcal{R}^{rm},E} t$, then there is $p \in \mathcal{P}os(s)$ and $\ell \to r \Leftarrow c \in R^{rm}$ such that (i) $s|_p =_E \sigma(\ell)$ for some substitution σ, (ii) for all $A \in c$, $\overline{\mathcal{R}^{\mathrm{CR}}} \vdash \sigma(A)$ holds, and (iii) $t = s[\sigma(r)]_p$.*

5 Analysis of local confluence modulo E of $\to_{\mathcal{R}^{rm},E}$ with $\to_{\mathcal{R}^{rm}}$

Given an EGTRS $\mathcal{R} = (\mathcal{F}, \Pi, E, H, R)$ and *terms* s, t, and t', *rewriting peaks* (12) become:

$$t \; {}_{\mathcal{R}^{rm},E}\!\leftarrow s \to_{\mathcal{R}^{rm}} t' \tag{29}$$

▶ **Example 26.** Consider E and R $(=R^{rm})$ in Example 14 viewed as an EGTRS $\mathcal{R} = (\{\mathsf{a},\mathsf{b},\mathsf{c},\mathsf{d},\mathsf{f}\}, \{=, \to, \to^*\}, E, \emptyset, R)$. Since (i) $\mathsf{f}(\mathsf{c}) =_E \mathsf{f}(\mathsf{a}) =_E \underline{\mathsf{b}} \to_{(17)} \mathsf{d}$ and (ii) $\underline{\mathsf{c}} \to_{(16)} \mathsf{d}$, we have the following rewriting peak:

$$\mathsf{d} \; {}_{(17),E}\!\leftarrow \overleftarrow{\mathsf{f}(\underrightarrow{\mathsf{c}})} \to_{(16)} \mathsf{f}(\mathsf{d}) \tag{30}$$

The $\mathcal{R}^{rm}, E$-joinability of all peaks (29) characterizes local confluence modulo E of $\to_{\mathcal{R}^{rm},E}$ with $\to_{\mathcal{R}^{rm}}$, which provides an ingredient to prove E-confluence (see Corollary 24). By Proposition 25, there are positions $\overline{p}, \overline{p'} \in \mathcal{P}os(s)$, rules $\alpha : \ell \to r \Leftarrow c, \alpha' : \ell' \to r' \Leftarrow c' \in R^{rm}$ sharing no variable (rename if necessary), and substitution σ, such that (i) $s|_{\overline{p}} = w =_E \sigma(\ell)$ and $\sigma(c)$ hold; and (ii) $s|_{\overline{p'}} = \sigma(\ell')$ and $\sigma(c')$ hold, i.e., every rewriting peak is of the form

$$t = s[\sigma(r)]_{\overline{p}} \; {}_{\mathcal{R}^{rm},E}\!\leftarrow s[\underleftarrow{w}]_{\overline{p}} = s = s[\underrightarrow{\sigma(\ell')}]_{\overline{p'}} \to_{\mathcal{R}^{rm}} s[\sigma(r')]_{\overline{p'}} = t' \tag{31}$$

Depending on the relative location of $\overline{p}$ and $\overline{p'}$, different classes of peaks (31) are distinguished.

***Disjoint* rewriting peaks.** If $\overline{p}$ and $\overline{p'}$ in (31) are *disjoint*, i.e., $\overline{p} \parallel \overline{p'}$, then $s = s[w]_{\overline{p}}[\sigma'(\ell')]_{\overline{p'}} = s[\sigma'(\ell')]_{\overline{p'}}[w]_{\overline{p}}$. Accordingly, (31) can be written as follows:

$$t = s[\sigma(r)]_{\overline{p}}[\sigma(\ell')]_{\overline{p'}} \; {}_{\mathcal{R}^{rm},E}\!\leftarrow \; s[\underleftarrow{w}]_{\overline{p}}[\underrightarrow{\sigma(\ell')}]_{\overline{p'}} \to_{\mathcal{R}^{rm}} s[w]_{\overline{p}}[\sigma(r')]_{\overline{p'}} = t' \tag{32}$$

Now, $t = s[\sigma(r)]_{\overline{p}}[\underrightarrow{\sigma(\ell')}]_{\overline{p'}} \to_{\mathcal{R}^{rm}} s[\sigma(r)]_{\overline{p}}[\sigma(r')]_{\overline{p'}} \; {}_{\mathcal{R}^{rm},E}\!\leftarrow s[\underleftarrow{w}]_{\overline{p}}[\sigma(r')]_{\overline{p'}} = t'$, i.e., t and t' are $\mathcal{R}^{rm}, E$*-joinable.*

***Nested* rewriting peaks.** If $\overline{p'} = \overline{p}.p \in \mathcal{P}os(s)$ for $\overline{p}$ and $\overline{p'}$ in (31) and some position p, then (31) can be written in one of the following possibilities, according to the position where the $\to_{\mathcal{R}^{rm}}$-step applies (by abuse, we also use t and t').

1. In the first case, rewriting with $\to_{\mathcal{R}^{rm}}$ occurs *above or on* the $\to_{\mathcal{R}^{rm},E}$-step and we have
$$t = \sigma(\ell')[\sigma(r)]_p \; {}_{\mathcal{R}^{rm},E}\!\leftarrow \; \underrightarrow{\sigma(\ell')[\overleftarrow{w}]_p} \to_{\mathcal{R}^{rm}} \sigma(r') = t' \tag{33}$$
where $w =_E \sigma(\ell)$ and $p \geq \Lambda$. We call (33) an $\mathcal{R}^{rm}$*-up* peak. We distinguish two cases:
   - **a.** If $p \in \mathcal{P}os_{\mathcal{F}}(\ell')$, then $\sigma(\ell')|_p = \sigma(\ell'|_p) = w =_E \sigma(\ell)$, i.e., ℓ and $\ell'|_p$ E-unify and we say that (33) is an *E-critical $\mathcal{R}^{rm}$-up peak*;
   - **b.** If $p \notin \mathcal{P}os_{\mathcal{F}}(\ell')$, there is $x \in \mathcal{V}ar(\ell')$ such that $\ell'|_q = x$ for some $q \leq p$ and (33) is a *variable $\mathcal{R}^{rm}$-up peak.*

2. In the second case, rewriting with $\to_{\mathcal{R}^{rm}}$ occurs *below* the $\to_{\mathcal{R}^{rm},E}$-step; we have
$$t = \sigma(r) \ {}_{\mathcal{R}^{rm},E}\!\leftarrow \ \overleftarrow{w[\underrightarrow{\sigma(\ell')}]_p} \ \to_{\mathcal{R}^{rm}} \ w[\sigma(r')]_p = t' \tag{34}$$
where $w = w[\sigma(\ell')]_p =_E \sigma(\ell)$ and $p > \Lambda$, as $p = \Lambda$ coincide with $p = \Lambda$ in (33). We call (34) an $\mathcal{R}^{rm}$-*down* peak. For instance, (30) is an $\mathcal{R}^{rm}$-*down* peak.

▶ Remark 27. If $E = \emptyset$, then $\to_{\mathcal{R}^{rm},E} = \to_{\mathcal{R}^{rm}}$, and $\mathcal{R}^{rm}$-up and $\mathcal{R}^{rm}$-down peaks boil down into a unique class of peaks, just distinguishing critical and variable peaks.

From the proof of [11, Theorem 16] (Case 4 in page 1171), for $\mathcal{R}^{rm}$-down peaks involving unconditional rules, we have:

▶ **Proposition 28.** *Let $\mathcal{R} = (\mathcal{F}, \Pi, E, H, R)$ be an EGTRS. If $\to_{\mathcal{R}^{rm},E}$ is locally coherent modulo E, then $\mathcal{R}^{rm}$-down peaks (34) such that α and α' are* unconditional rules *are $\mathcal{R}^{rm}, E$-joinable.*

6 Analysis of local coherence modulo E of $\to_{\mathcal{R}^{rm},E}$

Given an EGTRS $\mathcal{R} = (\mathcal{F}, \Pi, E, H, R)$ and terms s, t, t', *coherence peaks* (13) are of the form
$$t \ {}_{\mathcal{R}^{rm},E}\!\leftarrow s \to_{\overset{\leftrightarrow}{E}} t' \tag{35}$$
Given a term s, positions $\overline{p}, \overline{p'} \in \mathcal{P}os(s)$, an *oriented* equation $\lambda \to \rho \Leftarrow d \in \overset{\leftrightarrow}{E}$, a rule $\alpha : \ell \to r \Leftarrow c \in R^{rm}$ (sharing no variables), and substitution σ, such that (i) $s|_{\overline{p}} = w =_E \sigma(\ell)$ and $\sigma(c)$ hold; and (ii) $s|_{\overline{p'}} = \sigma(\lambda)$ and $\sigma(d)$ hold, every coherence peak (35) is of the form
$$t = s[\sigma(r)]_{\overline{p}} \ {}_{\mathcal{R}^{rm},E}\!\leftarrow s[\underleftarrow{w}]_{\overline{p}} \ = \ s \ = \ s[\underrightarrow{\sigma(\lambda)}]_{\overline{p'}} \ \to_{\overset{\leftrightarrow}{E}} \ s[\sigma(\rho)]_{\overline{p'}} = t' \tag{36}$$

***Disjoint* coherence peaks.** If $\overline{p}$ and $\overline{p'}$ in (36) are *disjoint*, then $s = s[w]_{\overline{p}}[\sigma(\lambda)]_{\overline{p'}} = s[\sigma(\lambda)]_{\overline{p'}}[w]_{\overline{p}}$ and we have:
$$t = s[\sigma(r)]_{\overline{p}}[\sigma(\lambda)]_{\overline{p'}} \ {}_{\overset{\leftrightarrow}{E}}\!\leftarrow \ s[\sigma(r)]_{\overline{p}}[\sigma(\rho)]_{\overline{p'}} \ {}_{\mathcal{R}^{rm},E}\!\leftarrow s[w]_{\overline{p}}[\sigma(\rho)]_{\overline{p'}} = t'$$
Since $t \ \to_{\overset{\leftrightarrow}{E}} \ s[\sigma(r)]_{\overline{p}}[\sigma(\rho)]_{\overline{p'}}$ implies $t \ =_E \ s[\sigma(r)]_{\overline{p}}[\sigma(\rho)]_{\overline{p'}}$, we conclude that t and t' are right-strict $\mathcal{R}^{rm}, E$-joinable.

***Nested* coherence peaks.** If $\overline{p}$ and $\overline{p'}$ in (36) are *not* disjoint, then (36) can be written in one of the following three ways (again, by abuse, we use t and t'):
$$t = \sigma(\lambda)[\sigma(r)]_p \ {}_{\mathcal{R}^{rm},E}\!\leftarrow \ \overleftarrow{\underrightarrow{\sigma(\lambda)}} \ \to_{\overset{\leftrightarrow}{E}} \ \sigma(\rho) = t' \tag{37}$$
$$t = \sigma(\lambda)[\sigma(r)]_p \ {}_{\mathcal{R}^{rm},E}\!\leftarrow \ \underrightarrow{\sigma(\lambda)[\overleftarrow{w}]_p} \ \to_{\overset{\leftrightarrow}{E}} \ \sigma(\rho) = t' \tag{38}$$
$$t = \ \sigma(r) \ {}_{\mathcal{R}^{rm},E}\!\leftarrow \ \overleftarrow{w[\underrightarrow{\sigma(\lambda)}]_p} \ \to_{\overset{\leftrightarrow}{E}} \ w[\sigma(\rho)]_p = t' \tag{39}$$
that we call E-root (37), E-up (38), and E-down (39) coherence peaks, depending on the application of the $\to_{\overset{\leftrightarrow}{E}}$-step. Note that $\sigma(\ell) =_E \sigma(\lambda)$ in (37), and $p > \Lambda$ in (38) and (39).

▶ **Proposition 29.** *Coherence peaks (37) and (39) are right-strict $\mathcal{R}^{rm}, E$-joinable.*

Now, we investigate finite representations of nested rewriting and coherence peaks.

7 Conditional pairs for proving E-confluence of EGTRSs

In the following, we deal with *general conditional pairs*, or just *conditional pairs*, as follows, see [15, Section 5]:

▶ **Definition 30** (Conditional pair). *A* conditional pair *is an expression* $\underbrace{\langle s,t\rangle}_{peak} \Leftarrow \underbrace{A_1,\ldots,A_n}_{conditional\ part}$, *where* s *and* t *are terms and* $A_1,\ldots,A_n$ *are atoms.*

▶ **Definition 31** (Joinability of conditional pairs). *Let* $\mathcal{R}=(\mathcal{F},\Pi,E,H,E)$ *be an EGTRS. A conditional pair* $\pi:\langle s,t\rangle \Leftarrow c$ *is* $\mathcal{R}^{rm},E$*-joinable (resp. right-strict* $\mathcal{R}^{rm},E$*-joinable,* $\mathcal{R}/E$-joinable*) iff for all substitutions* σ*, if* $\overline{\mathcal{R}^{\mathrm{CR}}}\vdash\sigma(A)$ *holds for all* $A\in c$*, then* $\sigma(s)$ *and* $\sigma(t)$ *are* $\mathcal{R}^{rm},E$*-joinable (resp. right-strict* $\mathcal{R}^{rm},E$*-joinable,* $\mathcal{R}/E$*-joinable).*

▶ **Definition 32** (Feasible conditional pair). *Let* $\mathcal{R}$ *be an EGTRS. A general conditional pair* $\langle s,t\rangle\Leftarrow c$ *is* $\overline{\mathcal{R}^{\mathrm{CR}}}$-feasible *(or just* feasible *if* $\overline{\mathcal{R}^{\mathrm{CR}}}$ *is clear from the context) if* c *is* $\overline{\mathcal{R}^{\mathrm{CR}}}$*-feasible.*

The following result is immediate from Definitions 31 and 32.

▶ **Proposition 33.** *Let* $\mathcal{R}$ *be an EGTRS.* $\overline{\mathcal{R}^{\mathrm{CR}}}$*-infeasible conditional pairs are* $\mathcal{R}^{rm},E$*-joinable (resp. right-strict* $\mathcal{R}^{rm},E$*-joinable,* $\mathcal{R}/E$*-joinable)*

We describe three families of conditional pairs which are useful to prove and disprove E-confluence.

7.1 Logic-based conditional critical pairs

These pairs capture $\mathcal{R}^{rm}$-up and E-up critical peaks.

▶ **Definition 34** (Logic-based conditional critical pair). *Let* $\alpha:\ell\to r\Leftarrow c$ *and* $\alpha':\ell'\to r'\Leftarrow c'$ *be two rules sharing no variables, together with a non-variable position* $p\in\mathcal{P}os_{\mathcal{F}}(\ell)$*. The* logic-based conditional critical pair *(LCCP for short)* $\pi_{\alpha,p,\alpha'}$ *of* α *at position* p *with* α' *is:*

$$\pi_{\alpha,p,\alpha'}:\langle\ell[r']_p,r\rangle\Leftarrow\ell|_p=\ell',c,c' \tag{40}$$

Our terminology "*logic-based conditional critical pair*" tries to avoid confusion with the *E-critical pairs* for ETRSs of [11, Definition 12] and also the *conditional critical pairs* for rewrite theories of [4, Definition 6] which make an *explicit* use of E-unifiers which we avoid by including the *atom* $\ell|_p=\ell'$ in the conditional part of (40). Given sets of rules U and V, we let $\mathsf{GLCCP}(U,V)=\{\pi_{\alpha,p,\alpha'}\mid\alpha:\ell\to r\Leftarrow c\in U,p\in\mathcal{P}os_{\mathcal{F}}(\ell),\alpha'\in V\}$. For $\mathcal{R}=(\mathcal{F},\Pi,E,H,R)$, we let

$$\mathsf{LCCP}(\mathcal{R})=\mathsf{GLCCP}(R^{rm},R^{rm})\quad\text{and}\quad\mathsf{LCCP}(E,\mathcal{R})=\mathsf{GLCCP}(\overleftrightarrow{E},R^{rm})$$

For GTRSs involving finite sets of equations and rules, both $\mathsf{LCCP}(\mathcal{R})$ and $\mathsf{LCCP}(E,\mathcal{R})$ are *finite*. The following result shows that they suffice to capture any possible critical peak.

▶ **Proposition 35** (Critical peaks and LCCPs). *Let* $\mathcal{R}=(\mathcal{F},\Pi,E,H,R)$ *be a EGTRS.*

- *Let* $\alpha:\ell\to r\Leftarrow c,\alpha':\ell'\to r'\Leftarrow c'\in R^{rm}$*, sharing no variable, induce* $\mathcal{R}^{rm}$*-up critical peaks (33) with* $p\in\mathcal{P}os_{\mathcal{F}}(\ell')$ *as in (33). Then, (33) is* $\mathcal{R}^{rm},E$*-joinable (*$\mathcal{R}/E$*-joinable), iff* $\pi_{\alpha,p,\alpha'}\in\mathsf{LCCP}(\mathcal{R})$ *is* $\mathcal{R}^{rm},E$*-joinable (*$\mathcal{R}/E$*-joinable).*
- *Let* $\alpha:\ell\to r\Leftarrow c\in R^{rm}$ *and* $\beta:\lambda\to\rho\Leftarrow d\in\overleftrightarrow{E}$*, sharing no variable, induce* E*-up critical peaks (38) with* $p\in\mathcal{P}os_{\mathcal{F}}(\lambda)$ *as in (38). Then, (38) is right-strict* $\mathcal{R}^{rm},E$*-joinable (*$\mathcal{R}/E$*-joinable) iff* $\pi_{\alpha,p,\beta}\in\mathsf{LCCP}(E,\mathcal{R})$ *is right-strict* $\mathcal{R}^{rm},E$*-joinable (*$\mathcal{R}/E$*-joinable).*

▶ **Example 36.** (Continuing Example 26) $\mathsf{LCCP}(\mathcal{R})$ consists of:

α	p	α'	LCCP	
(16)	Λ	(16)	$\langle \mathsf{d}, \mathsf{d} \rangle \Leftarrow \mathsf{c} = \mathsf{c}$	(41)
(16)	Λ	(17)	$\langle \mathsf{d}, \mathsf{d} \rangle \Leftarrow \mathsf{c} = \mathsf{b}$	(42)
(17)	Λ	(17)	$\langle \mathsf{d}, \mathsf{d} \rangle \Leftarrow \mathsf{b} = \mathsf{c}$	(43)

which are all trivially $\mathcal{R}^{rm}, E$-joinable. Note that $\pi_{(17),\Lambda,(16)}$ is *not* considered as it is $\mathcal{R}^{rm}, E$-joinable iff $\pi_{(16),\Lambda,(17)}$ is. Regarding $\mathsf{LCCP}(E, \mathcal{R})$, we have that $\pi_{\overleftarrow{(14)},1,(16)}$ i.e.,

$$\langle \mathsf{f}(\mathsf{d}), \mathsf{b} \rangle \Leftarrow \mathsf{a} = \mathsf{c}$$

is *not* right-strict $\mathcal{R}^{rm}, E$-joinable: although the conditional part $\mathsf{a} = \mathsf{c}$ holds, and $\mathsf{b} \to_{\mathcal{R}^{rm},E} \mathsf{d}$, $\langle \mathsf{f}(\mathsf{d}), \mathsf{d} \rangle$ is *not* $\mathcal{R}^{rm}, E$-joinable.

▶ **Example 37.** For $\mathcal{R}$ in Example 1, $\mathsf{LCCP}(\mathcal{R})$ consists of 22 LCCPs. The complete list and proofs of $\mathcal{R}^{rm}, E$-joinability are given in Appendix A. Representative examples are:

- $\pi_{(24),\Lambda,(24)}$ is

$$\langle m' + n', m + n \rangle \Leftarrow \mathsf{sum}(m +\!\!+ ns) = \mathsf{sum}(m' +\!\!+ ns'), \mathsf{Nat}(m), \mathsf{sum}(ns) \approx_{rm} n, \mathsf{Nat}(m'), \mathsf{sum}(ns') \approx_{rm} n'$$

  If a substitution σ satisfies the conditional part, then $\sigma(m +\!\!+ ns) =_E \sigma(m' +\!\!+ ns')$, i.e., the sequences $\sigma(m +\!\!+ ns)$ and $\sigma(m' +\!\!+ ns')$ are identical except for the distribution of parenthesis. Furthermore, $\sigma(m)$ and $\sigma(m')$ are the same natural number in Peano's notation. Therefore, $\sigma(ns)$ and $\sigma(ns')$ are identical up to parenthization. Hence, $\mathsf{sum}(\sigma(ns))$ and $\mathsf{sum}(\sigma(ns'))$ can be reduced by $\to^*_{\mathcal{R}/E}$ to the same expression, i.e., we can assume that $\sigma(n)$ and $\sigma(n')$ coincide and thus that $\sigma(m + n)$ and $\sigma(m' + n')$ are $\mathcal{R}^{rm}, E$-joinable.
- $\pi_{(21),\Lambda,(22)}$ is $\langle \mathsf{s}(m' + n'), n \rangle \Leftarrow \mathsf{0} + n = \mathsf{s}(m') + n'$. Since the conditional part is clearly infeasible, $\pi_{(21),\Lambda,(22)}$ is trivially $\mathcal{R}^{rm}, E$-joinable.

$\mathsf{LCCP}(E, \mathcal{R})$ consists of 16 LCCPs. Every $\pi \in \mathsf{LCCP}(E, \mathcal{R})$ is *infeasible*: they include a condition $s = t$ where s is rooted with $+\!\!+$ and t is always rooted with a *different* symbol. For instance, $\pi_{\overrightarrow{(1)},\Lambda,(21)}$ is $\langle n, (xs +\!\!+ ys) +\!\!+ zs \rangle \Leftarrow xs +\!\!+ (ys +\!\!+ zs) = \mathsf{0} + n$. Thus, every $\pi \in \mathsf{LCCP}(E, \mathcal{R})$ is right-strict $\mathcal{R}^{rm}, E$-joinable, see Appendix A too. Hence, all pairs in $\mathsf{LCCP}(\mathcal{R})$ are $\mathcal{R}^{rm}, E$-joinable and all pairs in $\mathsf{LCCP}(E, \mathcal{R})$ are right-strict $\mathcal{R}^{rm}, E$-joinable.

7.2 Conditional variable pairs

These pairs capture $\mathcal{R}^{rm}$-up and E-up variable peaks.

▶ **Definition 38** (Parametric conditional variable pair)**.** *Let $\alpha : s \to t \Leftarrow c$ be a rule where s can be a variable, $x \in \mathcal{V}ar(s)$, $p \in \mathcal{P}os_x(s)$, and $x' \notin \alpha$ be a fresh variable. Let $\bowtie$ be a binary relation on terms. Define a $\bowtie$-parametric* Conditional Variable Pair *(CVP) $\pi^{\bowtie}_{\alpha,x,p}$ as follows:*

$$\pi^{\bowtie}_{\alpha,x,p} : \langle s[x']_p, t \rangle \Leftarrow x \bowtie x', c \tag{44}$$

In the following, $\mathsf{CVP}(U, \bowtie)$ is the set of all $\bowtie$-parametric CVPs in a set of rules U. We let

$$\mathsf{CVP}(\mathcal{R}) = \mathsf{CVP}(R^{rm}, \overset{ps}{\to}) \quad \text{and} \quad \mathsf{CVP}(E) = \mathsf{CVP}(\overset{\leftrightarrow}{E}, \overset{ps}{\to})$$

▶ **Proposition 39** (Variable peaks and CVPs)**.** *Let $\mathcal{R} = (\mathcal{F}, \Pi, E, H, R)$ be an EGTRS.*

- *Let $\alpha : \ell \to r \Leftarrow c, \alpha' : \ell' \to r' \Leftarrow c' \in R^{rm}$, sharing no variable, determine variable $\mathcal{R}^{rm}$-up peaks (33) with $p \notin \mathcal{P}os_{\mathcal{F}}(\ell')$ as in (33). Then, (33) is $\mathcal{R}^{rm}, E$-joinable ($\mathcal{R}/E$-joinable) iff $\pi^{\overset{ps}{\to}}_{\alpha',x,q} \in \mathsf{CVP}(\mathcal{R})$ (for some $x \in \mathcal{V}ar(\ell')$ and $q \in \mathcal{P}os_x(\ell')$ such that $q \leq p$) is $\mathcal{R}^{rm}, E$-joinable ($\mathcal{R}/E$-joinable).*

- *Let $\alpha : \ell \to r \Leftarrow c \in R^{rm}$ and $\beta : \lambda \to \rho \Leftarrow d \in \overset{\leftrightarrow}{E}$, sharing no variable, determine variable E-up peaks (38) with $p \notin \mathcal{P}os_{\mathcal{F}}(\lambda)$ as in (38). Then, (38) is right-strict $\mathcal{R}^{rm}, E$-joinable ($\mathcal{R}/E$-joinable) iff $\pi^{ps}_{\beta,x,q} \in \mathsf{CVP}(E)$ (for some $q \in \mathcal{P}os_x(\lambda)$ and $x \in \mathcal{V}ar(\lambda)$ such that $q \leq p$) is right-strict $\mathcal{R}^{rm}, E$-joinable ($\mathcal{R}/E$-joinable).*

For unconditional rules, we have the following.

▶ **Proposition 40.** *Let $\mathcal{R} = (\mathcal{F}, \Pi, E, H, R)$ be an EGTRS and $\alpha : \lambda \to \rho$ be an* unconditional rule*, where λ can be a variable. Then, for all $x \in \mathcal{V}ar(\lambda)$ and $p \in \mathcal{P}os_x(\lambda)$, $\pi^{ps}_{\alpha,x,p}$ is $\mathcal{R}^{rm}, E$-joinable. If $x \in \mathcal{V}ar(\rho)$, then π is right-strict joinable.*

Accordingly, in the following we dismiss from $\mathsf{CVP}(\mathcal{R})$ those CVPs obtained from unconditional rules in $\mathcal{R}$; and we also dismiss from $\mathsf{CVP}(E)$ those CVPs obtained from unconditional rules $\lambda \to \rho \in \overset{\leftrightarrow}{E}$ whose critical variable x occurs in ρ.

▶ **Example 41.** For $\mathcal{R}$ in Example 1, $\mathsf{CVP}(\mathcal{R})$ consists of the following:

α	var.	p	CVP	
(23)	n	1	$\langle \mathsf{sum}(n'), n\rangle \Leftarrow n \overset{ps}{\to} n', \mathsf{Nat}(n)$	(45)
(24)	m	1.1	$\langle \mathsf{sum}(m' +\!\!+ ns), m+n\rangle \Leftarrow m \overset{ps}{\to} m', \mathsf{Nat}(m), \mathsf{sum}(ns) \approx_{rm} n$	(46)
(24)	ns	1.2	$\langle \mathsf{sum}(m +\!\!+ ns'), m+n\rangle \Leftarrow ns \overset{ps}{\to} ns', \mathsf{Nat}(m), \mathsf{sum}(ns) \approx_{rm} n$	(47)

These CVPs are *infeasible*, as terms t satisfying $\mathsf{Nat}(t)$ are of the form $\mathsf{s}^p(\mathsf{0})$ for some $p \geq 0$ and hence $\to_{\mathcal{R}^{rm},E}$-irreducible. Hence they are $\mathcal{R}^{rm}, E$-joinable. The set $\mathsf{CVP}(E)$ is empty, as all variables in the left hand side of the unconditional rules $\overrightarrow{(1)}$ and $\overleftarrow{(1)}$ also occur in the corresponding right-hand side (Proposition 40).

For $\mathcal{R}$ in Example 26, $\mathsf{CVP}(\mathcal{R}) = \emptyset$.

7.3 Down conditional critical pairs

$\mathcal{R}^{rm}$-down peaks (34) combine possible *rule overlaps* (modulo) and the application of rules "below" a variable. Unfortunately, these two sources of divergence do not admit a neat separation (as done for $\mathcal{R}^{rm}$-up peaks) into "critical" and "variable" $\mathcal{R}^{rm}$-down peaks to be captured by means of LCCPs and CVPs. Alternatively, *down conditional critical pairs* capture these two (mingled) situations at once. First consider the predicate $\rhd^{\times}$ defining a strict *subterm* relation on *pairs* (s,t) of terms by the following clauses:

$(\mathrm{Sb})^{\rhd^{\times}}_{f,i}$	$(f(x_1,\ldots,x_i,\ldots,x_k), f(x_1,\ldots,x'_i,\ldots,x_k)) \rhd^{\times} (x_i, x'_i)$
$(\mathrm{Sb2})^{\rhd^{\times}}_{f,i}$	$(f(x_1,\ldots,x_i,\ldots,x_k), f(x_1,\ldots,x'_i,\ldots,x_k)) \rhd^{\times} (x, x') \Leftarrow (x_i, x'_i) \rhd^{\times} (x, x')$

Let $\mathsf{Th}_{\rhd^{\times}}(\mathcal{F}) = \{(\mathrm{HC})_{(\mathrm{Sb})^{\rhd^{\times}}_{f,i}}, (\mathrm{HC})_{(\mathrm{Sb2})^{\rhd^{\times}}_{f,i}} \mid f \in \mathcal{F}, 1 \leq i \leq ar(f)\}$.

▶ **Proposition 42.** *Let $\mathcal{F}$ be a signature and s,t,u,v be terms. Then, $\mathsf{Th}_{\rhd^{\times}}(\mathcal{F}) \vdash (s,t) \rhd^{\times} (u,v)$ holds iff there is a nonempty context $C[\,]$ such that $s = C[u]$ and $t = C[v]$.*

▶ **Definition 43** (Down conditional critical pairs)**.** *Let $\mathcal{R} = (\mathcal{F}, \Pi, E, H, R)$ be an EGTRS. Rules $\alpha : \ell \to r \Leftarrow c, \alpha' : \ell' \to r' \Leftarrow c' \in R^{rm}$ (sharing no variables) define a* Down Conditional Critical Pair *(DCCP for short) as follows:*

$$\pi_{\alpha,\alpha'} : \langle r, x'\rangle \Leftarrow x = \ell, (x,x') \rhd^{\times} (\ell', r'), c, c' \tag{48}$$

where x and x' are fresh variables. The set of DCCPs of $\mathcal{R}$ is

$$\mathsf{DCCP}(\mathcal{R}) = \{\pi_{\alpha,\alpha'} \mid \alpha, \alpha' \in R^{rm}\}$$

▶ **Proposition 44** ($\mathcal{R}^{rm}$-peaks and DCCPs). *Let $\mathcal{R} = (\mathcal{F}, \Pi, E, H, R)$ be an EGTRS. Let $\alpha : \ell \to r \Leftarrow c, \alpha' : \ell' \to r' \Leftarrow c' \in R^{rm}$, sharing no variable, determine $\mathcal{R}^{rm}$-down critical peaks (34). Then, (34) is $\mathcal{R}^{rm}, E$-joinable ($\mathcal{R}/E$-joinable) iff $\pi_{\alpha,\alpha'} \in \mathsf{DCCP}(\mathcal{R})$ is $\mathcal{R}^{rm}, E$-joinable ($\mathcal{R}/E$-joinable).*

▶ Remark 45 (Continuing Remark 27). If $E = \emptyset$ in an EGTRS $\mathcal{R} = (\mathcal{F}, \Pi, E, H, R)$, then all peaks represented by $\mathsf{DCCP}(\mathcal{R})$ are captured by $\mathsf{LCCP}(\mathcal{R})$ and $\mathsf{CVP}(\mathcal{R})$.

▶ **Example 46.** For $\mathcal{R}$ in Example 26, $\mathsf{DCCP}(\mathcal{R})$ consists of the following DCCPs:

α	α'	DCCP	
(16)	(16)	$\langle \mathsf{d}, x' \rangle \Leftarrow x = \mathsf{c}, (x, x') \rhd^{\times} (\mathsf{c}, \mathsf{d})$	(49)
(16)	(17)	$\langle \mathsf{d}, x' \rangle \Leftarrow x = \mathsf{b}, (x, x') \rhd^{\times} (\mathsf{c}, \mathsf{d})$	(50)
(17)	(16)	$\langle \mathsf{d}, x' \rangle \Leftarrow x = \mathsf{c}, (x, x') \rhd^{\times} (\mathsf{b}, \mathsf{d})$	(51)
(17)	(17)	$\langle \mathsf{d}, x' \rangle \Leftarrow x = \mathsf{b}, (x, x') \rhd^{\times} (\mathsf{b}, \mathsf{d})$	(52)

As for (50), $\sigma = \{x \mapsto \mathsf{f}(\mathsf{c}), x' \mapsto \mathsf{f}(\mathsf{d})\}$ satisfies the conditional part as $\sigma(x) = \mathsf{f}(\mathsf{c}) =_E \mathsf{f}(\mathsf{a}) =_E \mathsf{b}$ and $(\mathsf{f}(\mathsf{c}), \mathsf{f}(\mathsf{d}) \rhd^{\times} (\mathsf{c}, \mathsf{d})$. However, d and $\mathsf{f}(\mathsf{d})$ are *not* $\mathcal{R}^{rm}, E$-joinable.

▶ **Proposition 47.** *Let $\mathcal{R} = (\mathcal{F}, \Pi, E, H, R)$ be an EGTRS and $\alpha : \ell \to r \Leftarrow c, \alpha' : \ell' \to r' \Leftarrow c' \in R^{rm}$ be such that $\ell = \ell[x_1, \ldots, x_n]$ is linear, $\mathcal{V}ar(\ell) = \{x_1, \ldots, x_n\}$, and for all terms t, if $t =_E \sigma(\ell)$ for some substitution σ satisfying c and c', then $t = \ell[t_1, \ldots, t_n]$ for some terms $t_1, \ldots, t_n$. If every LCCP $\pi_{\alpha,p,\alpha'}$ is $\mathcal{R}^{rm}, E$-joinable for all $p \in \mathcal{P}os_{\mathcal{F}}(\ell)$, and every CVP $\pi^{ps}_{\alpha,x,q}$ is $\mathcal{R}^{rm}, E$-joinable for all $x \in \mathcal{V}ar(\ell)$ and $q \in \mathcal{P}os_x(\ell)$, then the DCCP $\pi_{\alpha,\alpha'}$ is $\mathcal{R}^{rm}, E$-joinable.*

By Propositions 28 and 44, dealing with EGTRSs $\mathcal{R}$ such that $\to_{\mathcal{R}^{rm},E}$ is locally coherent modulo E, we can dismiss DCCPs for *unconditional* rules α and α'.

▶ **Example 48.** For $\mathcal{R}$ in Example 1, $\mathsf{DCCP}(\mathcal{R})$ consists of 16 DCCPs (involving a conditional rule). The complete list (with all $\mathcal{R}^{rm}, E$-joinability proofs) is given in Appendix A. A representative example is $\pi_{(21),(24)}$, i.e.,

$$\langle n, x' \rangle \Leftarrow x = \mathsf{0} + n, (x, x') \rhd^{\times} (\mathsf{sum}(m' +\!\!+ ns'), m' + n'), \mathsf{Nat}(m'), \mathsf{sum}(ns') \approx_{rm} n'$$

If σ satisfies the conditional part, then $\sigma(x) =_E \mathsf{0} + \sigma(n)$ holds. Since $+ \notin \mathcal{D}(\overset{\leftrightarrow}{E})$, it follows that $\sigma(x) = \mathsf{0} + \sigma(n)$. Thus, by Proposition 47 and since every $\pi \in \mathsf{LCCP}(\mathcal{R}) \cup \mathsf{CVP}(\mathcal{R})$ is $\mathcal{R}^{rm}, E$-joinable (Examples 37 and 41), $\pi_{(21),(24)}$ is $\mathcal{R}^{rm}, E$-joinable.

Example 46 shows that $\mathcal{R}^{rm}, E$-joinability of all $\pi \in \mathsf{LCCP}(\mathcal{R}) \cup \mathsf{CVP}(\mathcal{R})$ does *not* imply $\mathcal{R}^{rm}, E$-joinability of DCCPs in $\mathsf{DCCP}(\mathcal{R})$ unless the conditions in Proposition 47 are fulfilled.

8 Proving and disproving E-confluence

The following result shows how to prove and disprove E-confluence.

▶ **Theorem 49.** *Let $\mathcal{R} = (\mathcal{F}, \Pi, E, H, R)$ be an EGTRS.*

1. *$\to_{\mathcal{R}^{rm},E}$ is locally confluent modulo E with $\to_{\mathcal{R}^{rm}}$ iff every $\pi \in \mathsf{LCCP}(\mathcal{R}) \cup \mathsf{CVP}(\mathcal{R}) \cup \mathsf{DCCP}(\mathcal{R})$ is $\mathcal{R}^{rm}, E$-joinable.*
2. *$\to_{\mathcal{R}^{rm},E}$ is locally coherent modulo E iff every $\pi \in \mathsf{LCCP}(E, \mathcal{R}) \cup \mathsf{CVP}(E)$ is right-strict $\mathcal{R}^{rm}, E$-joinable.*
3. *If $\mathcal{R}$ is E-terminating, then $\mathcal{R}$ is $\to_{\mathcal{R}^{rm},E}$-Church-Rosser modulo E iff every $\pi \in \mathsf{LCCP}(\mathcal{R}) \cup \mathsf{CVP}(\mathcal{R}) \cup \mathsf{DCCP}(\mathcal{R})$ is $\mathcal{R}^{rm}, E$-joinable, and every $\pi \in \mathsf{LCCP}(E, \mathcal{R}) \cup \mathsf{CVP}(E)$ is right-strict $\mathcal{R}^{rm}, E$-joinable.*

4. *If $\mathcal{R}$ is E-terminating, every $\pi \in \mathsf{LCCP}(\mathcal{R}) \cup \mathsf{CVP}(\mathcal{R}) \cup \mathsf{DCCP}(\mathcal{R})$ is $\mathcal{R}^{rm}, E$-joinable, and every $\pi \in \mathsf{LCCP}(E, \mathcal{R}) \cup \mathsf{CVP}(E)$ is right-strict $\mathcal{R}^{rm}, E$-joinable, then $\mathcal{R}$ is E-confluent.*
5. *If there is $\pi \in \mathsf{LCCP}(\mathcal{R}) \cup \mathsf{CVP}(\mathcal{R}) \cup \mathsf{DCCP}(\mathcal{R})$ which is* not *$\mathcal{R}/E$-joinable, then $\mathcal{R}$ is not E-confluent.*

▶ **Example 50.** (Continuing Example 1) For $\mathcal{R}$ in Example 1, every $\pi \in \mathsf{LCCP}(\mathcal{R}) \cup \mathsf{CVP}(\mathcal{R}) \cup \mathsf{DCCP}(\mathcal{R})$ is $\mathcal{R}^{rm}, E$-joinable (see Examples 37, 41, and 48) and every $\pi \in \mathsf{LCCP}(E, \mathcal{R}) \cup \mathsf{CVP}(E)$ is right-strict $\mathcal{R}^{rm}, E$-joinable (see Examples 37 and 41). It is not difficult to see that $\mathcal{R}$ is E-terminating. Thus, by Theorem 49(4), $\mathcal{R}$ is E-confluent.

▶ **Example 51.** For $\mathcal{R}$ in Example 26, the DCCP (50) has been proved non-$\mathcal{R}/\mathcal{E}$-joinable in Example 46. By Theorem 49(5), $\mathcal{R}$ is *not* E-confluent. Note that all pairs in $\mathsf{LCCP}(\mathcal{R})$ are joinable (see Example 36) and $\mathsf{CVP}(\mathcal{R})$ is empty. Thus, $\mathsf{DCCP}(\mathcal{R})$ is the only set of conditional pairs that can be used to disprove E-confluence of $\mathcal{R}$.

9 Related work

GTRSs and EGTRSs. A Generalized Term Rewriting System (GTRS, [15, Definition 51]) is a tuple $\mathcal{R} = (\mathcal{F}, \Pi, \mu, H, R)$, where $\mathcal{F}$, Π, H and R are defined as above, and μ is a *replacement map* establishing which arguments $\mu(f)$ can be rewritten for each function symbol $f \in \mathcal{F}$ [14]. EGTRSs do not use replacement maps, which corresponds to "use" the so-called *top* replacement map $\mu_\top$ which permits all rewritings in all arguments of symbols. We have borrowed from [15, Definition 30] the notion of *conditional variable pair*, although we use it here in a slightly different way, as conditional pairs $\langle s, t\rangle \Leftarrow x \overset{ps}{\to} x', c$ where $\overset{ps}{\to}$ is interpreted as $\to_{\mathcal{R},E}$, but possible rewritings in c may correspond to $\to_{\mathcal{R}/E}$, $\to^*_{\mathcal{R}/E}$, etc. Conditional variable pairs of GTRSs $\mathcal{R}$ are written $\langle s, t\rangle \Leftarrow x \to x', c$, where $\to$ is the one-step rewrite relation $\to_{\mathcal{R}}$ of $\mathcal{R}$ and conditions in c are treated using $\to_{\mathcal{R}}$, $\to^*_{\mathcal{R}}$, etc.

Plaisted proposed quite a general notion of conditional rewrite systems where rules are viewed as clauses including (possibly many) negative literals [21]. In this respect, EGTRSs are particular cases of Plaisted's conditional rewrite systems. Plaisted also provides a complete specification of the logical theory which could be used (together with such rules) to obtain the desired reduction, see [21, page 217]. However, equational components are *not* allowed.

Jouannaud and Kirchner's main result for ETRSs [11, Theorem 16], cannot be used to *disprove* E-confluence of ETRSs. For instance, the proof of non-E-confluence of $\mathcal{R}$ in Example 26 would not be obtained. Actually, (30) in Example 26 is an $\mathcal{R}$-*down* rewriting peak. Such peaks are explictly *excluded* to obtain E-critical pairs in [10, 11][4]. Our *down conditional critical pairs* (DCCPs) fill this gap. On the other hand, Jouannaud and Kirchner's results for proving confluence of ETRSs modulo permit an application to relations R^{E} on terms like $\to_{\mathcal{L}} \cup \to_{\mathcal{N},E}$, where $\mathcal{L}$ and $\mathcal{N}$ are a partition of $\mathcal{R}$ where $\mathcal{L}$ includes left-linear rules only and $\mathcal{N}$ includes any other rules [11, Section 3.5]. The case considered here, $\to_{\mathcal{R},E}$, is a particular case of the previous one, where $\mathcal{L} = \emptyset$ and $\mathcal{N} = \mathcal{R}$. Under these conditions, [11, Theorem 16] treats proofs of E-confluence essentially as our Theorem 49.(4) (as $\mathsf{CVP}(\mathcal{R})$ and $\mathsf{DCCP}(\mathcal{R})$ can be dismissed according to the discussion above). The aforementioned more general treatment for EGTRSs is left as an interesting subject for future work.

[4] *"we do not consider the case where $\to_{\mathcal{R}}$ applies at an occurrence p and $\to_{\mathcal{R},\mathcal{E}}$ at the outermost occurrence Λ"* [11, below E-critical pairs lemma] (notation adapted).

Durán and Meseguer investigated E-confluence of *conditional rewrite theories* $\mathcal{R} = (\mathcal{F}, A, R)$ [4]. Such theories include CTRSs. Conditional equations $s = t \Leftarrow c$ can be specified, but they are treated as conditional rewrite rules (in R) by imposing some specific orientation (e.g., $s \to t \Leftarrow c$). Only unconditional equations $s = t$ (called *axioms*) which are *linear* and *regular* (i.e., $\mathcal{V}ar(s) = \mathcal{V}ar(t)$ [4, page 819]) are used in E (denoted A in [4]). The main result about E-confluence is [4, Theorem 2], which characterizes E-confluence by the joinability of the set of conditional E-critical pairs obtained from rules $\ell \to r \Leftarrow c$ and $\ell' \to r' \Leftarrow c'$ in $\mathcal{R}$ (which may include oriented conditional equations) by computing (if possible) the A-unifiers of $\ell|_p$ and ℓ' for some nonvariable position $p \in \mathcal{P}os_{\mathcal{F}}(\ell)$ [4, Definition 6]. However, a number of restrictions are imposed: (i) A is a set of linear and regular unconditional equations; (ii) R is strongly A-coherent (i.e., for all terms u, u', and v, if $u \to_{\mathcal{R}/A} v$ and $u =_A u'$, then $u' \to_{\mathcal{R},A} v'$ and $v =_A v'$ [4, page 819]); (iii) the rules in R are *strongly deterministic* [4, Definition 1]; (iv) $\mathcal{R}$ is *quasi-decreasing* [4, Definition 2]. Again, this result would *not* apply to disprove E-confluence of $\mathcal{R}$ in Example 26; note that E satisfies the requirements for axioms A in [4]. We have: $\mathsf{b} =_E \mathsf{f}(\underline{\mathsf{a}}) \to_{\mathcal{R}} \mathsf{f}(\mathsf{d})$, i.e., $\mathsf{b} \to_{\mathcal{R}/E} \mathsf{f}(\mathsf{d})$, but the only $\to_{\mathcal{R},E}$-step on b is $\mathsf{b} \to_{\mathcal{R},E} \mathsf{d}$, and $\mathsf{f}(\mathsf{d}) \neq_E \mathsf{d}$. Thus, $\mathcal{R}$ is not strongly E-coherent and [4, Theorem 2] does not apply. Also, the proof of E-confluence for $\mathcal{R}$ in Example 1 could not (in principle) be obtained from [4, Theorem 2]: since the set of E-unifiers for the left-hand sides of rules is infinite (Example 2), the joinability of infinitely many conditional E-critical pairs should be checked. Also, we do not require (i)-(iv) above in Theorem 49; only E-termination.

10 Conclusion and future work

We have introduced *Equational Generalized Term Rewriting Systems* (EGTRSs) consisting of *conditional rules* and *conditional equations* whose conditions are sequences of *atoms*, possibly defined by additional Horn clauses. Rewriting computations with EGTRSs $\mathcal{R}$ are described by *deduction* in appropriate FO-theories. We show that E-confluence of EGTRSs can be *proved* and *disproved* by checking (right-strict) $\mathcal{R}^{rm}, E$-joinability or non-$\mathcal{R}/E$-joinability of *finite sets* of conditional pairs of three kinds: *Logic-based Conditional Critical Pairs* (LCCPs), *Conditional Variable Pairs* (CVPs), and *Down Conditional Critical Pairs* (DCCPs). As far as we know, none of them had been used in proofs of E-confluence yet. The discussed examples suggest that the new techniques can be useful.

Future work. Much work remains to be done for a *practical* use of these new proposals. Theorem 49 heavily relies on checking (right-strict) $\mathcal{R}^{rm}, E$-joinability and non-$\mathcal{R}/E$-joinability of LCCPs, CVPs, and DCCPs, to obtain proofs of (non) E-confluence. We plan to improve our tool CONFident [8], which implements methods for the analysis of similar conditional pairs (see [15, 16]) and heavily relies on the (in)feasibility results developed in [7] and implemented in the tool infChecker. It also uses theorem provers like Prover9 [17] and model generators like Mace4 [17] and AGES [6] to implement these checkings. Overall, this approach has proved useful to prove confluence of variants of TRSs, see [8, Section 9] for an account. Unfortunately, our preliminary attempts to follow this methodology to prove E-confluence of EGTRSs in CONFident suggest that the use of the generic reasoning methods implemented in these tools is not powerful enough as to deal with the conditional pairs involved in E-confluence proofs. For instance, LCCPs avoid considering (infinitely many) conditional E-critical pairs. From a practical point of view, though (in particular, to obtain an efficient implementation), it is important to investigate the possibility of a *mixed* use of conditional

E-critical pairs together with LCCPs. For instance, if there is an E-unification algorithm and the set of (complete) E-unifiers is finite, then the corresponding (computable and finite set of) conditional E-critical pairs could be used instead of LCCPs. In spite of this, DCCPs and CVPs remain as main ingredients in proofs of (non-)E-confluence.

Finally, exploring the impact of our techniques in *first-order deduction modulo* see, e.g., [3] and the references therein, is another interesting subject of future work.

References

1 Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998. doi:10.1017/CBO9781139172752.

2 Franz Baader and Wayne Snyder. Unification theory. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning (in 2 volumes)*, pages 445–532. Elsevier and MIT Press, 2001. doi:10.1016/b978-044450813-3/50010-2.

3 Gilles Dowek. Automated theorem proving in first-order logic modulo: on the difference between type theory and set theory. *CoRR*, abs/2306.00498, 2023. doi:10.48550/ARXIV.2306.00498.

4 Francisco Durán and José Meseguer. On the church-rosser and coherence properties of conditional order-sorted rewrite theories. *J. Log. Algebraic Methods Program.*, 81(7-8):816–850, 2012. doi:10.1016/j.jlap.2011.12.004.

5 Melvin Fitting. *First-Order Logic and Automated Theorem Proving, Second Edition*. Graduate Texts in Computer Science. Springer, 1996. doi:10.1007/978-1-4612-2360-3.

6 Raúl Gutiérrez and Salvador Lucas. Automatic Generation of Logical Models with AGES. In Pascal Fontaine, editor, *Automated Deduction - CADE 27 - 27th International Conference on Automated Deduction, Proceedings*, volume 11716 of *Lecture Notes in Computer Science*, pages 287–299. Springer, 2019. doi:10.1007/978-3-030-29436-6_17.

7 Raúl Gutiérrez and Salvador Lucas. Automatically Proving and Disproving Feasibility Conditions. In Nicolas Peltier and Viorica Sofronie-Stokkermans, editors, *Automated Reasoning - 10th International Joint Conference, IJCAR 2020, Proceedings, Part II*, volume 12167 of *Lecture Notes in Computer Science*, pages 416–435. Springer, 2020. doi:10.1007/978-3-030-51054-1_27.

8 Raúl Gutiérrez, Miguel Vítores, and Salvador Lucas. Confluence Framework: Proving Confluence with CONFident. In Alicia Villanueva, editor, *Logic-Based Program Synthesis and Transformation - 32nd International Symposium, LOPSTR 2022, Proceedings*, volume 13474 of *Lecture Notes in Computer Science*, pages 24–43. Springer, 2022. doi:10.1007/978-3-031-16767-6_2.

9 Gérard P. Huet. Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems. *J. ACM*, 27(4):797–821, 1980. doi:10.1145/322217.322230.

10 Jean-Pierre Jouannaud. Confluent and coherent equational term rewriting systems: Application to proofs in abstract data types. In Giorgio Ausiello and Marco Protasi, editors, *CAAP'83, Trees in Algebra and Programming, 8th Colloquium, Proceedings*, volume 159 of *Lecture Notes in Computer Science*, pages 269–283. Springer, 1983. doi:10.1007/3-540-12727-5_16.

11 Jean-Pierre Jouannaud and Hélène Kirchner. Completion of a set of rules modulo a set of equations. *SIAM J. Comput.*, 15(4):1155–1194, 1986. doi:10.1137/0215084.

12 Stephen Cole Kleene. *Mathematical Logic*. Dover Publications, 2002. Originally published by John Wiley & Sons, 1967.

13 Donald E. Knuth and Peter E. Bendix. Simple Word Problems in Universal Algebra. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, 1970.

14 Salvador Lucas. Context-sensitive Rewriting. *ACM Comput. Surv.*, 53(4):78:1–78:36, 2020. doi:10.1145/3397677.

15 Salvador Lucas. Local confluence of conditional and generalized term rewriting systems. *Journal of Logical and Algebraic Methods in Programming*, 136:paper 100926, pages 1–23, 2024. doi:10.1016/j.jlamp.2023.100926.

16 Salvador Lucas, Miguel Vítores, and Raúl Gutiérrez. Proving and disproving confluence of context-sensitive rewriting. *Journal of Logical and Algebraic Methods in Programming*, 126:100749, 2022. doi:10.1016/j.jlamp.2022.100749.

17 William McCune. Prover9 & Mace4. Technical report, University of New Mexico, 2005–2010. URL: http://www.cs.unm.edu/~mccune/prover9/.

18 Elliott Mendelson. *Introduction to mathematical logic (4. ed.)*. Chapman and Hall, 1997.

19 Enno Ohlebusch. *Advanced Topics in Term Rewriting*. Springer, 2002. doi:10.1007/978-1-4757-3661-8.

20 Gerald E. Peterson and Mark E. Stickel. Complete sets of reductions for some equational theories. *J. ACM*, 28(2):233–264, 1981. doi:10.1145/322248.322251.

21 David A. Plaisted. A Logic for Conditional Term Rewriting Systems. In Stéphane Kaplan and Jean-Pierre Jouannaud, editors, *Conditional Term Rewriting Systems, 1st International Workshop, Proceedings*, volume 308 of *Lecture Notes in Computer Science*, pages 212–227. Springer, 1987. doi:10.1007/3-540-19242-5_16.

22 Gordon D. Plotkin. Building-in equational theories. *Machine Intelligence*, 7:73–90, 1972.

23 John Alan Robinson. A review of automatic theorem proving. In J.T. Schwartz, editor, *Mathematical Aspects of Computer Science*, volume XIX of *Proceedings of Simposia in Applied Mathematics*, pages 1–18, 1967. doi:10.1090/psapm/019.

A Conditional pairs for the main running example

A.1 LCCP($\mathcal{R}$) for $\mathcal{R}$ in Example 1

The LCCPs in $\mathsf{LCCP}(\mathcal{R})$ for $\mathcal{R}$ in Example 1 are displayed in Figure 2. The following result is useful to analyze their joinability.

▶ **Proposition 52.** *Let $\mathcal{R}$ be an EGTRS and $\pi : \langle \ell[r']_p, r\rangle \Leftarrow \ell|_p = \ell', c, c' \in \mathsf{LCCP}(\mathcal{R})$. If $root(\ell|_p), root(\ell') \notin \mathcal{D}(\overset{\leftrightarrow}{E})$, and $root(\ell|_p) \neq root(\ell')$, then π is $\overline{\mathcal{R}^{\mathrm{CR}}}$-infeasible.*

Proof. By definition of LCCP, since $p \in \mathcal{P}os_{\mathcal{F}}(\ell)$ and $\alpha' \in R^{rm}$, we have that $\ell|_p, \ell' \notin \mathcal{X}$. Thus, for all substitutions σ satisfying $\ell|_p = \ell'$, we have $\sigma(\ell|_p) \to^*_{\overset{\leftrightarrow}{E}} \sigma(\ell')$. Since $root(\ell|_p) \neq root(\ell')$ and reductions with $\to_{\overset{\leftrightarrow}{E}}$ cannot ultimately lead to a root symbol $root(\sigma(\ell')) = root(\ell')$ in the sequence above, π is $\overline{\mathcal{R}^{\mathrm{CR}}}$-infeasible. ◀

Regarding $\mathcal{R}^{rm}, E$-joinability of these LCCPS,

- By Proposition 52, the LCCPs (54)–(60); (62)–(67); and (71)–(74) are $\overline{\mathcal{R}^{\mathrm{CR}}}$-infeasible, hence $\mathcal{R}^{rm}, E$-joinable.
- (69) is also infeasible as the satisfaction of $\mathsf{Nat}(n)$ by a substitution σ is possible only if $\sigma(n) = \mathsf{s}^p(\mathsf{0})$ for some $p \geq 0$; in this case, $\mathsf{sum}(\sigma(n)) = \mathsf{sum}(\sigma(m') +\!\!+ \sigma(ns'))$, i.e., $\mathsf{sum}(\mathsf{s}^p(\mathsf{0})) = \mathsf{sum}(\sigma(m') +\!\!+ \sigma(ns'))$ does *not* hold in $\overline{\mathcal{R}^{\mathrm{CR}}}$.
- The $\mathcal{R}^{rm}, E$-joinability of (70) is discussed in Example 37. The $\mathcal{R}^{rm}, E$-joinability of (53), (61), (68) and (70) is concluded in a similar way.

Thus, every $\pi \in \mathsf{LCCP}(\mathcal{R})$ is $\mathcal{R}^{rm}, E$-joinable.

α	p	α'	LCCP	
(21)	Λ	(21)	$\langle n', n\rangle \Leftarrow 0 + n = 0 + n'$	(53)
(21)	Λ	(22)	$\langle \mathsf{s}(m' + n'), n\rangle \Leftarrow 0 + n = \mathsf{s}(m') + n'$	(54)
(21)	Λ	(23)	$\langle n', n\rangle \Leftarrow 0 + n = \mathsf{sum}(n'), \mathsf{Nat}(n')$	(55)
(21)	Λ	(24)	$\langle m' + n', n\rangle \Leftarrow 0 + n = \mathsf{sum}(m' +\!\!+ ns'), \mathsf{Nat}(m'), \mathsf{sum}(ns') \approx_{rm} n'$	(56)
(21)	1	(21)	$\langle n' + n, n\rangle \Leftarrow 0 = 0 + n$	(57)
(21)	1	(22)	$\langle \mathsf{s}(m' + n') + n, n\rangle \Leftarrow 0 = \mathsf{s}(m') + n'$	(58)
(21)	1	(23)	$\langle n' + n, n\rangle \Leftarrow 0 = \mathsf{sum}(n'), \mathsf{Nat}(n')$	(59)
(21)	1	(24)	$\langle (m' + n') + n, n\rangle \Leftarrow 0 = \mathsf{sum}(m' +\!\!+ ns'), \mathsf{Nat}(m'), \mathsf{sum}(ns') \approx_{rm} n'$	(60)
(22)	Λ	(22)	$\langle \mathsf{s}(m' + n'), \mathsf{s}(m + n)\rangle \Leftarrow \mathsf{s}(m) + n = \mathsf{s}(m') + n'$	(61)
(22)	Λ	(23)	$\langle n', \mathsf{s}(m + n)\rangle \Leftarrow \mathsf{s}(m) + n = \mathsf{sum}(n'), \mathsf{Nat}(n')$	(62)
(22)	Λ	(24)	$\langle m' + n', \mathsf{s}(m + n)\rangle \Leftarrow \mathsf{s}(m) + n = \mathsf{sum}(m' +\!\!+ ns'), \mathsf{Nat}(m'), \mathsf{sum}(ns')$	(63)
(22)	1	(21)	$\langle n' + n, \mathsf{s}(m + n)\rangle \Leftarrow \mathsf{s}(m) = 0 + n'$	(64)
(22)	1	(22)	$\langle \mathsf{s}(m' + n') + n, \mathsf{s}(m + n)\rangle \Leftarrow \mathsf{s}(m) = \mathsf{s}(m') + n'$	(65)
(22)	1	(23)	$\langle n' + n, \mathsf{s}(m + n)\rangle \Leftarrow \mathsf{s}(m) = \mathsf{sum}(n'), \mathsf{Nat}(n')$	(66)
(22)	1	(24)	$\langle m' + n' + n, \mathsf{s}(m + n)\rangle \Leftarrow \mathsf{s}(m) = \mathsf{sum}(m' +\!\!+ ns'), \mathsf{Nat}(m'), \mathsf{sum}(ns')$	(67)
(23)	Λ	(23)	$\langle n', n\rangle \Leftarrow \mathsf{sum}(n) = \mathsf{sum}(n'), \mathsf{Nat}(n), \mathsf{Nat}(n')$	(68)
(23)	Λ	(24)	$\langle m' + n', n\rangle \Leftarrow \mathsf{sum}(n) = \mathsf{sum}(m' +\!\!+ ns'), \mathsf{Nat}(n), \mathsf{Nat}(m'), \mathsf{sum}(ns') \approx_{rm} n'$	(69)
(24)	Λ	(24)	$\langle m' + n', m + n\rangle \Leftarrow \mathsf{sum}(m +\!\!+ ns) = \mathsf{sum}(m' +\!\!+ ns'), \mathsf{Nat}(m), \mathsf{sum}(ns) \approx_{rm} n, \mathsf{Nat}(m'), \mathsf{sum}(ns') \approx_{rm} n'$	(70)
(24)	1	(21)	$\langle \mathsf{sum}(n'), m + n\rangle \Leftarrow m +\!\!+ ns = 0 + n', \mathsf{Nat}(m), \mathsf{sum}(ns) \approx_{rm} n$	(71)
(24)	1	(22)	$\langle \mathsf{sum}(\mathsf{s}(m' + n')), m + n\rangle \Leftarrow m +\!\!+ ns = \mathsf{s}(m') + n', \mathsf{Nat}(m), \mathsf{sum}(ns) \approx_{rm} n$	(72)
(24)	1	(23)	$\langle \mathsf{sum}(n'), m + n\rangle \Leftarrow m +\!\!+ ns = \mathsf{sum}(n'), \mathsf{Nat}(m), \mathsf{sum}(ns) \approx_{rm} n, \mathsf{Nat}(n')$	(73)
(24)	1	(24)	$\langle \mathsf{sum}(m' + n'), m + n\rangle \Leftarrow m +\!\!+ ns = \mathsf{sum}(m' +\!\!+ ns'), \mathsf{Nat}(m), \mathsf{sum}(ns) \approx_{rm} n, \mathsf{Nat}(m'), \mathsf{sum}(ns') \approx_{rm} n'$	(74)

Figure 2 LCCPs of $\mathcal{R}$ in Example 1.

A.2 LCCP$(E, \mathcal{R})$ for $\mathcal{R}$ in Example 1

The LCCPs in $\mathsf{LCCP}(E, \mathcal{R})$ for $\mathcal{R}$ in Example 1 are displayed in Figure 3. The following result is useful to analyze their joinability. Here, we say that a set U of conditional rules is *collapsing* if there is a *feasible* rule in U whose right-hand side is a variable.

▶ **Proposition 53.** *Let $\mathcal{R}$ be an EGTRS and $\pi : \langle \ell[r']_p, r\rangle \Leftarrow \ell|_p = \ell', c, c' \in \mathsf{LCCP}(E, \mathcal{R})$. If $\overset{\leftrightarrow}{E}$ is* not *collapsing and $root(\ell') \notin \mathcal{D}(\overset{\leftrightarrow}{E})$, then π is $\overline{\mathcal{R}^{\mathrm{CR}}}$-infeasible.*

Proof. By definition of LCCP, since $p \in \mathcal{P}os_{\mathcal{F}}(\ell)$ and $\alpha' \in R^{rm}$, we have that $\ell|_p, \ell' \notin \mathcal{X}$. Thus, for all substitutions σ satisfying $\ell|_p = \ell'$, we have $\sigma(\ell|_p) \to^*_{\overset{\leftrightarrow}{E}} \sigma(\ell')$. Since $\overset{\leftrightarrow}{E}$ is not collapsing and $root(\ell') \notin \mathcal{D}(\overset{\leftrightarrow}{E})$, reductions with $\to_{\overset{\leftrightarrow}{E}}$ cannot ultimately lead to a root symbol $root(\sigma(\ell')) = root(\ell') \notin \mathcal{D}(\overset{\leftrightarrow}{E})$ in the sequence above, π is $\overline{\mathcal{R}^{\mathrm{CR}}}$-infeasible. ◀

The following example shows that non-collapsingness of $\overset{\leftrightarrow}{E}$ is necessary for Proposition 53 to hold.

▶ **Example 54.** Consider the following EGTRS

$$0 + x = x \tag{75}$$
$$\mathsf{f}(0) \to 0 \tag{76}$$

We have the following LCCP in $\mathsf{LCCP}(E, \mathcal{R})$:

$$\langle x, x\rangle \Leftarrow 0 + x = \mathsf{f}(0) \tag{77}$$

α	p	α'	LCCP	
$\overrightarrow{(1)}$	Λ	(21)	$\langle n, (xs +\!\!+ ys) +\!\!+ zs\rangle \Leftarrow xs +\!\!+ (ys +\!\!+ zs) = 0 + n$	(78)
$\overrightarrow{(1)}$	Λ	(22)	$\langle \mathsf{s}(m+n), (xs +\!\!+ ys) +\!\!+ zs\rangle \Leftarrow xs +\!\!+ (ys +\!\!+ zs) = \mathsf{s}(m) + n$	(79)
$\overrightarrow{(1)}$	Λ	(23)	$\langle n, (xs +\!\!+ ys) +\!\!+ zs\rangle \Leftarrow xs +\!\!+ (ys +\!\!+ zs) = \mathsf{sum}(n), \mathsf{Nat}(n)$	(80)
$\overrightarrow{(1)}$	Λ	(24)	$\langle m+n, (xs +\!\!+ ys) +\!\!+ zs\rangle \Leftarrow xs +\!\!+ (ys +\!\!+ zs) = \mathsf{sum}(m +\!\!+ ns), \mathsf{Nat}(m), \mathsf{sum}(ns) \approx_{rm} n$	(81)
$\overrightarrow{(1)}$	2	(21)	$\langle xs +\!\!+ n, (xs +\!\!+ ys) +\!\!+ zs\rangle \Leftarrow ys +\!\!+ zs = 0 + n$	(82)
$\overrightarrow{(1)}$	2	(22)	$\langle xs +\!\!+ \mathsf{s}(m+n), (xs +\!\!+ ys) +\!\!+ zs\rangle \Leftarrow ys +\!\!+ zs = \mathsf{s}(m) + n$	(83)
$\overrightarrow{(1)}$	2	(23)	$\langle xs +\!\!+ n, (xs +\!\!+ ys) +\!\!+ zs\rangle \Leftarrow ys +\!\!+ zs = \mathsf{sum}(n), \mathsf{Nat}(n)$	(84)
$\overrightarrow{(1)}$	2	(24)	$\langle xs +\!\!+ (m+n), (xs +\!\!+ ys) +\!\!+ zs\rangle \Leftarrow ys +\!\!+ zs = \mathsf{sum}(m +\!\!+ ns), \mathsf{Nat}(m), \mathsf{sum}(ns) \approx_{rm} n$	(85)
$\overleftarrow{(1)}$	Λ	(21)	$\langle n, xs +\!\!+ (ys +\!\!+ zs)\rangle \Leftarrow (xs +\!\!+ ys) +\!\!+ zs = 0 + n$	(86)
$\overleftarrow{(1)}$	Λ	(22)	$\langle \mathsf{s}(m+n), xs +\!\!+ (ys +\!\!+ zs)\rangle \Leftarrow (xs +\!\!+ ys) +\!\!+ zs = \mathsf{s}(m) + n$	(87)
$\overleftarrow{(1)}$	Λ	(23)	$\langle n, xs +\!\!+ (ys +\!\!+ zs)\rangle \Leftarrow (xs +\!\!+ ys) +\!\!+ zs = \mathsf{sum}(n), \mathsf{Nat}(n)$	(88)
$\overleftarrow{(1)}$	Λ	(24)	$\langle m+n, xs +\!\!+ (ys +\!\!+ zs)\rangle \Leftarrow (xs +\!\!+ ys) +\!\!+ zs = \mathsf{sum}(m +\!\!+ ns), \mathsf{Nat}(m), \mathsf{sum}(ns) \approx_{rm} n$	(89)
$\overleftarrow{(1)}$	1	(21)	$\langle n +\!\!+ zs, xs +\!\!+ (ys +\!\!+ zs)\rangle \Leftarrow xs +\!\!+ ys = 0 + n$	(90)
$\overrightarrow{(1)}$	1	(22)	$\langle \mathsf{s}(m+n) +\!\!+ zs, xs +\!\!+ (ys +\!\!+ zs)\rangle \Leftarrow xs +\!\!+ ys = \mathsf{s}(m) + n$	(91)
$\overleftarrow{(1)}$	1	(23)	$\langle n +\!\!+ zs, xs +\!\!+ (ys +\!\!+ zs)\rangle \Leftarrow xs +\!\!+ ys = \mathsf{sum}(n), \mathsf{Nat}(n)$	(92)
$\overleftarrow{(1)}$	1	(24)	$\langle (m+n) +\!\!+ zs, xs +\!\!+ (ys +\!\!+ zs)\rangle \Leftarrow xs +\!\!+ ys = \mathsf{sum}(m +\!\!+ ns), \mathsf{Nat}(m) \; \mathsf{sum}(ns) \approx_{rm} n$	(93)

Figure 3 LCCPs of E and $\mathcal{R}$ in Example 1.

Substitution $\sigma = \{x \mapsto \mathsf{f}(0)\}$ satisfies the conditional part of (77), i.e., it is $\overline{\mathcal{R}^{\text{CR}}}$-feasible.

By Proposition 53, the LCCPs in Figure 3 are all $\overline{\mathcal{R}^{\text{CR}}}$-infeasible, hence right-strict $\mathcal{R}^{rm}, E$-joinable. Thus, every $\pi \in \mathsf{LCCP}(E, \mathcal{R})$ is right-strict $\mathcal{R}^{rm}, E$-joinable.

A.3 DCCP($\mathcal{R}$) for $\mathcal{R}$ in Example 1

The DCCPs in $\mathsf{DCCP}(\mathcal{R})$ for $\mathcal{R}$ in Example 1 are displayed in Figure 4.

α	α'	DCCP	
(21)	(23)	$\langle n, x'\rangle \Leftarrow x = 0 + n, (x, x') \rhd^{\times} (\mathsf{sum}(n'), n'), \mathsf{Nat}(n')$	(94)
(21)	(24)	$\langle n, x'\rangle \Leftarrow x = 0 + n, (x, x') \rhd^{\times} (\mathsf{sum}(m' +\!\!+ ns'), m' + n'), \mathsf{Nat}(m'), \mathsf{sum}(ns') \approx_{rm} n'$	(95)
(22)	(23)	$\langle \mathsf{s}(m+n), x'\rangle \Leftarrow x = \mathsf{s}(m) + n, (x, x') \rhd^{\times} (\mathsf{sum}(n'), n'), \mathsf{Nat}(n')$	(96)
(22)	(24)	$\langle \mathsf{s}(m+n), x'\rangle \Leftarrow x = \mathsf{s}(m) + n, (x, x') \rhd^{\times} (\mathsf{sum}(m' +\!\!+ ns'), m' + n'), \mathsf{Nat}(m'), \mathsf{sum}(ns') \approx_{rm} n'$	(97)
(23)	(21)	$\langle n, x'\rangle \Leftarrow x = \mathsf{sum}(n), (x, x') \rhd^{\times} (0 + n', n'), \mathsf{Nat}(n)$	(98)
(23)	(22)	$\langle n, x'\rangle \Leftarrow x = \mathsf{sum}(n), (x, x') \rhd^{\times} (\mathsf{s}(m') + n', \mathsf{s}(m' + n')), \mathsf{Nat}(n)$	(99)
(23)	(23)	$\langle n, x'\rangle \Leftarrow x = \mathsf{sum}(n), (x, x') \rhd^{\times} (\mathsf{sum}(n'), n'), \mathsf{Nat}(n), \mathsf{Nat}(n')$	(100)
(23)	(24)	$\langle n, x'\rangle \Leftarrow x = \mathsf{sum}(n), (x, x') \rhd^{\times} (\mathsf{sum}(m' +\!\!+ ns'), m' + n'), \mathsf{Nat}(n), \mathsf{Nat}(m'), \mathsf{sum}(ns') \approx_{rm} n'$	(101)
(24)	(21)	$\langle m+n, x'\rangle \Leftarrow x = \mathsf{sum}(m +\!\!+ ns), (x, x') \rhd^{\times} (0 + n', n'), \mathsf{Nat}(m), \mathsf{sum}(ns) \approx_{rm} n$	(102)
(24)	(22)	$\langle m+n, x'\rangle \Leftarrow x = \mathsf{sum}(m +\!\!+ ns), (x, x') \rhd^{\times} (\mathsf{s}(m') + n', \mathsf{s}(m' + n')), \mathsf{Nat}(m), \mathsf{sum}(ns) \approx_{rm} n$	(103)
(24)	(23)	$\langle m+n, x'\rangle \Leftarrow x = \mathsf{sum}(m +\!\!+ ns), (x, x') \rhd^{\times} (\mathsf{sum}(n'), n'), \mathsf{Nat}(m), \mathsf{sum}(ns) \approx_{rm} n, \mathsf{Nat}(n')$	(104)
(24)	(24)	$\langle m+n, x'\rangle \Leftarrow x = \mathsf{sum}(m +\!\!+ ns), (x, x') \rhd^{\times} (\mathsf{sum}(m' +\!\!+ ns'), m' + n'), \mathsf{Nat}(m), \mathsf{sum}(ns) \approx_{rm} n, \mathsf{Nat}(m'), \mathsf{sum}(ns') \approx_{rm} n'$	(105)

Figure 4 DCCPs for $\mathcal{R}$ in Example 1.

- By Proposition 47, since we have proven above that every $\pi \in \mathsf{LCCP}(\mathcal{R}) \cup \mathsf{CVP}(\mathcal{R})$ is $\mathcal{R}^{rm}, E$-joinable, the DCCPs (94)–(101) are $\mathcal{R}^{rm}, E$-joinable.
- Regarding (102)–(105), notice that all these DCCPs contain the following conditions in the conditional part: (i) $x = \mathsf{sum}(m +\!\!+ ns)$ and (ii) $\mathsf{Nat}(m)$. Therefore, for all substitutions σ satisfying them, $\sigma(m) = \mathsf{s}^p(\mathsf{0})$ for some $p \geq 0$ and $\sigma(x) = \mathsf{sum}(\mathsf{s}^p(\mathsf{0}) +\!\!+ \sigma(ns))$. Therefore, Proposition 47 can be applied to conclude $\mathcal{R}^{rm}, E$-joinability of all of them.

Therefore, every $\pi \in \mathsf{DCCP}(\mathcal{R})$ is $\mathcal{R}^{rm}, E$-joinable.

A First Order Theory of Diagram Chasing

Assia Mahboubi
Nantes Université, École Centrale Nantes, CNRS, INRIA, LS2N, UMR 6004, France
Vrije Universiteit Amsterdam, The Netherlands

Matthieu Piquerez
Nantes Université, École Centrale Nantes, CNRS, INRIA, LS2N, UMR 6004, France

Abstract

This paper discusses the formalization of proofs "by diagram chasing", a standard technique for proving properties in abelian categories. We discuss how the essence of diagram chases can be captured by a simple many-sorted first-order theory, and we study the models and decidability of this theory. The longer-term motivation of this work is the design of a computer-aided instrument for writing reliable proofs in homological algebra, based on interactive theorem provers.

2012 ACM Subject Classification Theory of computation → Logic and verification

Keywords and phrases Diagram chasing, formal proofs, abelian categories, decidability

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.38

Related Version
Full Version: `https://hal.science/hal-04266479`
Full Version: `https://arxiv.org/abs/2311.01790`

Supplementary Material *Software*: `https://gitlab.inria.fr/mpiquere/coq-diagram-chasing` archived at `swh:1:dir:be0a95b35dcfefe6d05a2446be611fa81f424995`

Funding This work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 101001995).

Acknowledgements The authors would like to thank Kenji Maillard, Loïc Pujet and the anonymous reviewers for their useful comments and suggestions on this work.

1 Introduction

Homological algebra [12, 14] attaches and studies a sequence of algebraic objects, typically groups or modules, to a certain space, e.g., a ring or a topological space, in order to better understand the latter. In this field, diagram chasing is a major proof technique, which is usually carried out via a form of diagrammatic reasoning on abelian categories. Diagrams appear as early as in the introduction of Mac Lane's classic reference book [13], while Riehl's textbook devotes a section to the *art of diagram chase* [18, Section 1.6]. Lawvere and Schanuel's pedagogical introduction to category theory [11, Session 17] uses an entire session to discuss the role of graphs in diagrammatic categorical reasoning.

A diagram can be seen as a functor $F\colon J \to \mathcal{C}$, whose domain J, the indexing category, is a small category [18] sometimes also called the *shape* of the diagram [14]. Diagrams are usually represented as directed multi-graphs, also called *quivers*, whose vertices are decorated with objects of $\mathcal{C}$, and arrows with morphisms. Paths in such graphs thus correspond to chains of composable arrows. Diagrams allow for visualizing the existence of certain morphisms, and to study identities between certain compositions of morphisms. In particular, a diagram *commutes* when any two paths with same source and target lead to identical composite. For instance, the commutativity of the diagram on Figure 1 asserts that morphism e is equal to the composition of morphisms a and b, denoted by $b \circ a$, as well as to the composition $d \circ c$. Commutativity of diagrams in certain categories can be used to state more involved properties, and diagram chasing essentially consists in establishing the existence, injectivity,

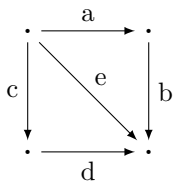


Figure 1 Square diagram with a diagonal arrow.

surjectivity of certain morphisms, or the exactness of some sequences, using hypotheses of the same nature. The *five lemma* or the *snake lemma* are typical examples of results with proofs "by diagram chasing", also called diagram chases. On paper, diagrams help conveying in a convincing manner proofs otherwise consisting of overly pedestrian chains of equations. The tension between readability and elusiveness may however become a challenge. For instance, diagram chases may rely on non-trivial duality arguments, that is, on the fact that a property about diagrams in any abelian category remains true after reversing all the involved arrows, although the replay of a given proof *mutatis mutandis* cannot be fulfilled in general.

Motivated in part by the second author's experience in writing intricate diagram chases (see for instance [16, p.338]), this work aims at laying the foundations of a computer-aided instrument for writing reliable proofs in homological algebra, based on interactive theorem provers. The present article discusses the design of a formal language for statements of properties amenable to proofs by diagram chasing, according to three objectives. The first is *simplicity and expressivity*: this language should be at the same time simple enough to be implemented in a formal library, and expressive enough to encompass the desired corpus of results. Then, *duality* arguments in proofs shall follow directly from a meta-property of the language. Finally, the corresponding proof system should allow for *automating* proofs of commutativity clauses. Observe for instance, that the commutativity of the square diagram of Figure 1 follows from that of the two triangle sub-diagrams: the concluding step in diagram chases usually amounts to a such a commutativity clause. Automating the mundane proofs of commutativity clauses amounts to studying a decision problem hereafter referred to as the *commerge* problem: *Given a collection of sub-diagrams of a larger diagram which commute, must the entire diagram commute?*

For this purpose, we introduce the two following formal languages.

▶ **Definition 1.** *We define a many-sorted signature* $\mathring{\Sigma}$ *with sorts the collection of finite quivers. Signature* $\mathring{\Sigma}$ *has one function symbol* $\mathrm{restr}_{m\colon Q'\to Q}$*, of arity* $Q \to Q'$*, per each quiver morphism* $m\colon Q' \to Q$ *between two quivers* Q *and* Q'*, and one predicate symbol* $\mathrm{commute}_Q$*, on sort* Q*, for each finite quiver* Q*.*

Similarly, we define a many-sorted signature Σ *with sorts the collection of finite* acyclic *quivers. Signature* Σ *has one function symbol* $\mathrm{restr}_{m\colon Q'\hookrightarrow Q}$*, of arity* $Q \to Q'$*, per each* injective *quiver morphism* $m\colon Q' \hookrightarrow Q$ *between two quivers* Q *and* Q'*, and one predicate symbol* $\mathrm{commute}_Q$*, on sort* Q*, for each finite acyclic quiver* Q*.*

The thesis of the present article is that signature Σ fulfills the three above objectives. We validate this thesis by giving a first-order theory for diagrams in small and abelian small categories respectively. We state and prove a duality theorem and motivate the choice of Σ over the possibly more intuitive $\mathring{\Sigma}$ by the automation objective.

The rest of the article is organized as follows. We first fix some vocabulary and notations in Section 2, so as in particular to make Definition 1 precise. Then, Section 3 introduces a theory for small categories and describes its models, Section 4 discusses duality, before Section 5 provides an analogue study for abelian categories. Finally in Section 6, we formalize and study the decidability of several variants of the commerge problem.

2 Preliminaries

In all what follows, $\mathbb{N} \coloneqq \{0, 1, \dots\}$ refers to the set of non-negative integers. If $k \in \mathbb{N}$, then $[k]$ denotes the finite collection $\{0, \dots, k-1\}$. We denote $\mathrm{card}(A)$ the cardinal of a finite set A. We use the notation id for the identity map.

2.1 Quivers

In this section, we introduce some vocabulary and notations related to directed multi-graphs, also called simply *graphs* in some standard category theory textbooks [11, 13]. In this article, we depart from these texts and use instead the term *quiver*.

▶ **Definition 2** (General quiver, dual). *A* general quiver $\mathcal{Q}$ *is a quadruple* $(V_\mathcal{Q}, A_\mathcal{Q}, s_\mathcal{Q} \colon A_\mathcal{Q} \to V_\mathcal{Q}, t_\mathcal{Q} \colon A_\mathcal{Q} \to V_\mathcal{Q})$ *where* $V_\mathcal{Q}$ *and* $A_\mathcal{Q}$ *are two sets. The element of* $V_\mathcal{Q}$ *are called the* vertices *of* $\mathcal{Q}$ *and the element of* $A_\mathcal{Q}$ *are called* arrows. *If* $a \in A_\mathcal{Q}$, $s_\mathcal{Q}(a)$ *is called the* source *of* a *and* $t_\mathcal{Q}(a)$ *is called its* target. *The* dual *of a quiver* $\mathcal{Q}$ *is the quiver* $\mathcal{Q}^\dagger \coloneqq (V_\mathcal{Q}, A_\mathcal{Q}, t_\mathcal{Q}, s_\mathcal{Q})$, *which swaps the source and the target maps of* $\mathcal{Q}$.

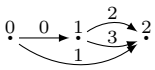
▶ **Definition 3** (Morphism, embedding, restriction). *A* morphism of quivers $m \colon \mathcal{Q} \to \mathcal{Q}'$, *is the data of two maps* $m_V \colon V_\mathcal{Q} \to V_{\mathcal{Q}'}$ *and* $m_A \colon A_\mathcal{Q} \to A_{\mathcal{Q}'}$ *such that* $m_V \circ s_\mathcal{Q} = s_{\mathcal{Q}'} \circ m_A$ *and* $m_V \circ t_\mathcal{Q} = t_{\mathcal{Q}'} \circ m_A$. *Such a morphism is called an* embedding of quivers *if moreover both* m_V *and* m_A *are injective. In this case we write* $m \colon \mathcal{Q} \hookrightarrow \mathcal{Q}'$.

If A *is a subset of* $A_\mathcal{Q}$, *the* (spanning) restriction of $\mathcal{Q}$ to A *denoted by* $\mathcal{Q}|_A$ *is the quiver* $(V_\mathcal{Q}, A, s_\mathcal{Q}|_A, t_\mathcal{Q}|_A)$. *There is a canonical embedding* $\mathcal{Q}|_A \hookrightarrow \mathcal{Q}$.

We denote by $\varnothing$ the empty quiver with no vertex and no arrow, and by $\mathring{S}$ the set of quivers Q such that V_Q and A_Q are finite subsets of $\mathbb{N}$. In this article, a *quiver* refers to an element of $\mathring{S}$. We use a non-cursive Q for elements of $\mathring{S}$, and a cursive $\mathcal{Q}$ for general quivers.

For the sake of readability, we use drawings to describe some elements of $\mathring{S}$, as for instance:

For a quiver Q denoted by such a drawing, the convention is that $V_Q = [\mathrm{card}(V_Q)]$ and $A_Q = [\mathrm{card}(A_Q)]$. From left to right, the drawn vertices correspond to $0, 1, \dots, \mathrm{card}(V_Q) - 1$. Arrows are then numbered by sorting pairs (s_Q, t_Q) in increasing lexicographical order, as in:



We also use drawings to denote embeddings. The black part represents the domain of the morphism, the union of black and gray parts represents its codomain. Here is an example of an embedding of the quiver  into the quiver drawn above.

▶ **Definition 4** (Path-quiver). *The* path-quiver of length k, *denoted by* PQ_k, *is the quiver with* $k+1$ *vertices and* k *arrows* $([k+1], [k], \mathrm{id}, (i \mapsto i+1))$.

A path-quiver can be drawn as:

with at least one vertex. Such a path-quiver is called *nontrivial* if it has at least two vertices.

If $0 \leq k \leq l$ are two integers, we denote by $\mathsf{sp}_{k,l} \colon \mathrm{PQ}_k \hookrightarrow \mathrm{PQ}_l$ the leftmost embedding of PQ_k into PQ_l, i.e., such that $(\mathsf{sp}_{k,l})_V(0) = 0$. If k and l are clear from the context, we draw $\mathsf{sp}_{k,l}$ as $\rightsquigarrow$ if $k \neq 0$ and as $\bullet\!\!\rightsquigarrow$ if $k = 0$. Moreover, we denote by $\mathsf{tp}_{k,l} \colon \mathrm{PQ}_k \hookrightarrow \mathrm{PQ}_l$ the rightmost embedding of PQ_k into PQ_l, i.e., such that $(\mathsf{tp}_{k,l})_V(k) = l$. The corresponding drawings are $\rightsquigarrow\!\!\bullet$ and $\bullet\!\!\rightsquigarrow\!\!\bullet$. Moreover, if P is a nontrivial path-quiver, we define $\mathsf{st}_P \colon \bullet\,\bullet \hookrightarrow P$ to be the embedding mapping the first vertex on the leftmost vertex of P and the second vertex on the rightmost vertex of P. We denote this embedding $\bullet\!\!\rightarrow\!\!\bullet$.

If $\mathcal{Q}$ is a general quiver, a morphism of the form $p \colon \mathrm{PQ}_k \to \mathcal{Q}$, for some k, is called a *path of $\mathcal{Q}$ from u to v of length k*, where $u \coloneqq p(0)$ and $v \coloneqq p(k)$. If moreover p_A is injective, it is called a *trail* [24]. A trail p such that $p_V(0) = p_V(k)$ is called a *cycle*. Two paths $p_1 \colon P_1 \to \mathcal{Q}$, $p_2 \colon P_2 \to \mathcal{Q}$ of $\mathcal{Q}$ *have the same extremities* if $p_1 \circ \mathsf{st}_{P_1} = p_2 \circ \mathsf{st}_{P_2}$. We denote by $\mathcal{BP}_\mathcal{Q}$, resp. $\mathcal{BT}_Q$, the set of pairs of paths, resp. of trails, of $\mathcal{Q}$ having the same extremities. Such a pair is called a *bipath*, resp. a *bitrail*.

A general quiver is *acyclic* if any path of this quiver is an embedding. The set of acyclic quivers in $\mathring{S}$ is denoted by S.

We now recall the definition of a *free category* (see for instance [13, Section II.7]).

▶ **Definition 5** (Free category). *For a general quiver $\mathcal{Q}$, the* free category over $\mathcal{Q}$, *denoted by $\langle \mathcal{Q} \rangle$ is the category with objects $\mathrm{Ob}_{\langle \mathcal{Q} \rangle} = V_\mathcal{Q}$ whose morphisms $\mathrm{Hom}_{\langle \mathcal{Q} \rangle}(u, v)$, for two vertices u and v are the paths from u to v. The identity map from u to u is the empty path, and the composition is defined as the concatenation of paths.*

Note that a morphism m of quivers induces a functor between the corresponding categories that we denote by Φ_m. In the other direction, any small category $\mathcal{C}$ has an *underlying quiver*.

2.2 Diagrams

The purpose of this section is to introduce diagrams in a category, and to define what it means for a diagram to commute. We also introduce useful constructions later used for building new diagrams by restricting and pasting existing ones.

▶ **Definition 6** (Diagram). *For any category $\mathcal{C}$ and any quiver Q, a* diagram in $\mathcal{C}$ over Q *is a functor from $\langle Q \rangle$ to $\mathcal{C}$.*

Definition 6 is actually a special case of $\mathcal{D}$-shaped diagrams, for $\mathcal{D}$ a small category [14]. Thanks to the universal properties of free categories, it coincides with Mac Lane's Q-shaped diagrams in category $\mathcal{C}$ [13, Section II.7].

Let P be a path-quiver from vertex u to vertex v. To a diagram $D \colon \langle P \rangle \to \mathcal{C}$ over P one can associate the corresponding composition of morphisms in the category $\mathcal{C}$, which is an element of $\mathrm{Hom}_\mathcal{C}(D(u), D(v))$. We denote this element $\mathrm{comp}(D)$. By convention, when the path-quiver P is trivial, $\mathrm{comp}(D)$ is the identity map $\mathrm{id}_{D(u)}$.

▶ **Definition 7** (Pullback). *Let Q, Q' be two quivers and let D be a diagram over Q. Let $m \colon Q' \to Q$ be a morphism of quivers. We define the* pullback of D by m, *denoted by $m^*(D)$, as the diagram $D \circ \Phi_m$ over Q'.*

Note that term pullback here refers to pre-composition rather than to fiber products.

▶ **Definition 8** (Commutative diagram). *For any category $\mathcal{C}$ and any quiver Q, a diagram D over Q is* commutative *if $\mathrm{comp}(p_1^*(D))$ and $\mathrm{comp}(p_2^*(D))$ coincide for any two paths p_1 and p_2 in Q with same extremities, that is:*

$$\forall (p_1, p_2) \in \mathcal{BP}_Q, \quad \mathrm{comp}(p_1^*(D)) = \mathrm{comp}(p_2^*(D)).$$

▶ **Definition 9** (Pushout configuration). *Consider four quivers Q, Q_1, Q_2, Q' and four embeddings $m_1 \colon Q \hookrightarrow Q_1$, $m_2 \colon Q \hookrightarrow Q_2$, $m'_1 \colon Q_1 \hookrightarrow Q'$ and $m'_2 \colon Q_2 \hookrightarrow Q'$:*

$$\begin{array}{ccc} Q & \xhookrightarrow{m_1} & Q_1 \\ \downarrow{\scriptstyle m_2} & & \downarrow{\scriptstyle m'_1} \\ Q_2 & \xhookrightarrow{m'_2} & Q'. \end{array}$$

This data is a pushout configuration *if:*

- $m'_1 \circ m_1 = m'_2 \circ m_2$,
- $Q' = \mathrm{Im}(m'_1) \cup \mathrm{Im}(m'_2)$, *(i.e.,* $V_{Q'} = \mathrm{Im}(m'_{1,V}) \cup \mathrm{Im}(m'_{2,V})$ *and* $A_{Q'} = \mathrm{Im}(m'_{1,A}) \cup \mathrm{Im}(m'_{2,A})$*),*
- $\mathrm{Im}(m'_1 \circ m_1) = \mathrm{Im}(m'_1) \cap \mathrm{Im}(m'_2)$.

In this case, the triple (Q', m'_1, m'_2) is called a pushout of (Q, Q_1, Q_2, m_1, m_2).

▶ Remark 10. Any tuple (Q, Q_1, Q_2, m_1, m_2) has a pushout, and any two pushouts of the same tuple are isomorphic, see for instance Figure 2.

The next lemma allows to reduce the number of distinct diagrams involved in a formula.

▶ **Lemma 11.** *Let $\mathcal{C}$ be some category. Consider a pushout configuration as in Definition 9. Consider two diagrams D_1 and D_2 in $\mathcal{C}$ over Q_1 and Q_2, respectively. If the pullback of D_1 by m_1 coincides with the pullback of D_2 by m_2, i.e.,*

$$m_1^*(D_1) = m_2^*(D_2)$$

then there exists a unique diagram D' over Q' such that D_1, resp. D_2, is the pullback of D' by m'_1, resp. m'_2, i.e.,

$$D_1 = m_1'^*(D') \quad \text{and} \quad D_2 = m_2'^*(D')$$

Proof. Immediate. ◀

2.3 Category congruences, path relations and quotient categories

We first introduce category congruences and quotients, following Mac Lane [13, Section II.8], and provide an important example thereof. However, we slightly depart from this reference by only considering the special case of quotients of categories by congruences, as it is the only one used in the present article.

▶ **Definition 12** (Category congruence). *A* category congruence *r on $\mathcal{C}$ is the data of an equivalence relation $r_{A,B} \subseteq \mathrm{Hom}_{\mathcal{C}}(A, B)^2$ for any pair of objects A and B such that, for any objects A, B, C and any morphisms $f, g \in \mathrm{Hom}_{\mathcal{C}}(A, B)$ and $f', g' \in \mathrm{Hom}_{\mathcal{C}}(B, C)$,*

$$f \sim g \quad \text{and} \quad f' \sim g' \quad \Longrightarrow \quad f' \circ f \sim g' \circ g,$$

where we write $h \sim h'$ if h and h' are in relation, i.e., $(h, h') \in r_{E,F}$ for some objects E and F. Such a relation is said complete *if $r_{A,B} = \mathrm{Hom}_{\mathcal{C}}(A, B)^2$ for any pair of objects A and B.*

Figure 2 A pushout Q' of two path-quivers P_1 and P_2 with respect to st_{P_1} and st_{P_2}.

▶ **Proposition 13** (Quotient category). *Given such a category congruence, the data* $\mathrm{Ob}_{\mathcal{C}/r} \coloneqq \mathrm{Ob}_{\mathcal{C}}$ *and* $\mathrm{Hom}_{\mathcal{C}/r}(A,B) \coloneqq \mathrm{Hom}_{\mathcal{C}}(A,B)/r_{A,B}$ *define a category* $\mathcal{C}/r$ *called the* quotient category *of* $\mathcal{C}$ *by* r*.*

Proof. Immediate. ◀

We now introduce relations on pairs of paths with same extremities in a general quiver: the ones that are induced from a congruence on the corresponding free category are of special interest.

▶ **Definition 14.** *A* relation between paths with same extremities in $\mathcal{Q}$ *is by definition a subset of* $\mathcal{BP}_{\mathcal{Q}}$*. If* $r \subseteq \mathcal{BP}_{\mathcal{Q}}$ *is such a relation then, for* $(p,q) \in \mathcal{BP}_{\mathcal{Q}}$*, we write* $p \sim q$ *if* $(p,q) \in r$*. Note that* $\mathcal{BP}_{\mathcal{Q}} = \bigsqcup_{A,B \in \mathrm{Ob}_{\langle \mathcal{Q} \rangle}} \mathrm{Hom}_{\langle \mathcal{Q} \rangle}(A,B)^2$*. Such a relation* r *is called a* path relation *if it is a category congruence on* $\langle \mathcal{Q} \rangle$*. The complete path relation on* $\mathcal{Q}$*, i.e.,* $\mathcal{BP}_{\mathcal{Q}}$*, is denoted by* $\mathrm{tot}_{\mathcal{Q}}$*.*

For instance, in a small category $\mathcal{C}$ the composition axiom induces a path relation on the underlying quiver $\mathcal{Q}$. To be more precise, this relation is given by

$$\{(p_1,p_2) \in \mathcal{BP}_{\mathcal{Q}} \mid \mathrm{comp}(F \circ \Phi_{p_1}) = \mathrm{comp}(F \circ \Phi_{p_2})\}$$

where F is the canonical functor from $\langle \mathcal{Q} \rangle$ to $\mathcal{C}$. Following the usual notation of ideals, if $r_0, \dots, r_{l-1}$ are some relations between paths with same extremities, we denote by $(r_i \mid i \in [l])$ the smallest path relation containing $r_0, \dots, r_{l-1}$.

Let $\mathcal{Q}'$ be another general quiver and let $m \colon \mathcal{Q} \to \mathcal{Q}'$ be a morphism. If $r \subseteq \mathcal{BP}_{\mathcal{Q}}$, we denote by $m_*(r)$ the relation induced by the image by m of r in $\mathcal{BP}_{\mathcal{Q}'}$.

2.4 A finite characterization of commutative diagrams

Definition 8 about the commutativity of a finite diagram a priori relies on an infinite number of equations. In this section, we give an equivalent formulation based on the finite set $\mathcal{BT}_Q$.

▶ **Lemma 15.** *Let* $\mathcal{Q}$ *be a quiver. The smallest path relation containing* $\mathcal{BT}_{\mathcal{Q}} \subseteq \mathcal{BP}_{\mathcal{Q}}$ *is the total relation* $\mathrm{tot}_{\mathcal{Q}}$*.*

Proof. Denote this smallest path relation by r. It suffices to prove that any path is related with a trail. Let p be a path which is not a trail. Then p contains a nontrivial cycle, i.e., if we denote by $\gg$ the concatenation operator on paths, $p = p_1 \gg c \gg p_2$ for some paths p_1 and p_2 and some nontrivial cycle c. Since any cycle is a trail, the pair $(c, c \circ \leftsquigarrow)$ belongs to $\mathcal{BT}_{\mathcal{Q}}$, where $c \circ \leftsquigarrow$ is the path of length 0 based on $c_V(0)$. Hence $p = p_1 \gg c \gg p_2 \sim p_1 \gg p_2$. This last past is strictly smaller, and we conclude by infinite descent. ◀

As a direct consequence of the previous lemma, we get the following proposition.

▶ **Proposition 16.** *A diagram* D *over a finite quiver* Q *is commutative if and only if*

$$\forall (p_1,p_2) \in \mathcal{BT}_Q, \quad \mathrm{comp}(p_1^*(D)) = \mathrm{comp}(p_2^*(D)).$$

2.5 Many-sorted logic, categorical interpretation

We first recall a few basic definitions mostly pertaining to many-sorted logic, but instantiated to the signatures introduced by Definition 1, and we set the corresponding notations.

Remember that signature Σ only differs from $\mathring{\Sigma}$ by restricting the allowed sorts to the acyclic quivers and the allowed quiver morphisms to embeddings.

Let us first fix a countable set X, so that for each quiver Q in $\mathring{S}$ (resp. in S), elements of the set $X_Q \coloneqq X \times \{Q\}$ are the *variables of sort* Q. A *term* of sort Q either is a variable of sort Q or has the form $\mathrm{restr}_{m\colon Q\to Q'}(t)$, with t a term of sort Q', Q' a quiver and $m\colon Q \to Q'$ a morphism of quivers. When possible, we leave the sorts implicit and simplify the notation of symbol $\mathrm{restr}_{m\colon Q'\to Q}$ into restr_m.

We denote the equality symbols by $\approx$, leaving sorts implicit. An *atom* is thus of the form $s \approx t$ with s and t two terms of the same sort, or of the form $\mathrm{commute}_Q(t)$ with t a term of sort Q. We consider first-order many-sorted formulas and write the sort of quantifiers as a subscript, i.e., $\exists_Q x_Q, \phi$ and $\forall_Q x_Q, \phi$ where $Q \in \mathring{S}, x_Q \in X_Q$ and ϕ is a formula.

We shorten $\forall_Q x, \forall_Q y, P(x,y)$, resp. $\exists_Q x, \exists_Q y, P(x,y)$, into $\forall_Q x, y, P(x,y)$, resp. into $\exists_Q x, y, P(x,y)$. We write $\exists!_Q\, y, P(y)$ for formula $\big(\exists_Q\, y,\ P(y)\big) \wedge \big(\forall_Q\, y_1, y_2,\ P(y_1) \wedge P(y_2) \to y_1 \approx y_2\big)$. A formula with free variables $x_1, \dots, x_n$ of respective sorts $Q_1, \dots, Q_n$ is said to be of *arity* $Q_1 \times \dots Q_n$.

We now define standard, sometimes also called Tarskian [20], semantics for first-order theories on signatures $\mathring{\Sigma}$ and Σ. We mostly follow classic presentations [9] but adapted to the context of multi-sorted signatures.

▶ **Definition 17** (Structures, models). *A* $\mathring{\Sigma}$-structure $\mathcal{M}$*, also called* interpretation of $\mathring{\Sigma}$*, is defined by:*

- *a collection of disjoint non-empty domain sets* $(\mathcal{M}_Q)_Q$*, indexed by the collection of (quiver) sorts;*
- *an interpretation of each function symbol* $\mathrm{restr}_{m\colon Q'\to Q}$*, as a function* $\mathrm{restr}^{\mathcal{M}}_{m\colon Q'\to Q}$ *with domain* $\mathcal{M}_Q$ *and codomain* $\mathcal{M}_{Q'}$*;*
- *an interpretation of each predicate symbol* $\mathrm{commute}_Q$*, as a subset* $\mathrm{commute}^{\mathcal{M}}_Q$ *of* $\mathcal{M}_Q$*.*

A given $\mathring{\Sigma}$*-structure together with a variable assignment mapping any variable of sort* Q *to an element of domain* $\mathcal{M}_Q$ *entail a truth value for any first-order formula* ϕ *on language* $\mathring{\Sigma}$*. If* ϕ *has no free variable, we write* $\mathcal{M} \models \phi$ *if* $\phi^{\mathcal{M}}$ *is true and we say that* ϕ *is valid in* $\mathcal{M}$*. A formula* ϕ *is* satisfiable *when there is a variable assignment which makes it true, and* unsatisfiable *in the opposite case. A* model of a theory $\mathcal{T}$ *on signature* $\mathring{\Sigma}$ *is the interpretation of a* $\mathring{\Sigma}$-structure *such that every formula in* $\mathcal{T}$ *is true.*

We also define analogue structures, interpretations, models for signature Σ*.*

The standard models of the signatures introduced in Definition 1 are actually diagrams in a certain category.

▶ **Definition 18** (Categorical interpretation). *To each small category* $\mathcal{C}$*, we associate an interpretation of* Σ*, resp.* $\mathring{\Sigma}$*, that we also denote by* $\mathcal{C}$*, as follows:*

- *To each sort* Q *we associate the set* $\mathcal{C}_Q$ *of diagrams in* $\mathcal{C}$ *over* Q*.*
- restr_m *is interpreted as the function mapping a diagram* D *to the diagram* $m^*(D)$*.*
- $\mathrm{commute}^{\mathcal{C}}_Q$ *is the set of commutative diagrams in* $\mathcal{C}$ *over* Q*.*

We call such an interpretation a categorical interpretation of Σ, resp. $\mathring{\Sigma}$*.*

3 A theory for diagrams in small categories

This section introduces a theory whose models can be seen as categorical interpretations.

3.1 Axioms

We now introduce the different axioms of the theory. A formula F with free variables $x_1, \dots, x_n$ is written $F(x_1, \dots, x_n)$ so as to clarify the sorts of each variable in the arity of F.

Existence and uniqueness of the empty diagram

$$\text{EmptyEU}: \qquad \exists!_{\varnothing}\, x, \quad x \approx x$$

Compatibility of restrictions

For any quivers Q, Q', Q'' and morphisms $m\colon Q \to Q'$ and $m'\colon Q' \to Q''$, we define:

$$\text{RestrComp}_{m,m'}: \qquad \forall_{Q''}\, x'', \quad \text{restr}_m(\text{restr}_{m'}(x'')) \approx \text{restr}_{m'\circ m}(x'').$$

Pushout

For any pushout configuration as in Definition 9, and using the same notations as this definition, we define the following formulas of arity $Q_1 \times Q_2 \times Q'$:

$$\text{Cospan}_{m'_1,m'_2}(x_1, x_2, x'): \qquad \text{restr}_{m'_1}(x') \approx x_1 \ \wedge\ \text{restr}_{m'_2}(x') \approx x_2.$$

$$\text{PushoutEU}_{m_1,m_2,m'_1,m'_2}: \qquad \forall_{Q_1}\, x_1, \forall_{Q_2}\, x_2, \quad \text{restr}_{m_1}(x_1) \approx \text{restr}_{m_2}(x_2) \to\ \exists!_{Q'}\, x',\ \text{Cospan}_{m'_1,m'_2}(x_1, x_2, x').$$

Composition

The following predicate, of arity $\cdot\!\longrightarrow\!\cdot \times \cdot\!\longrightarrow\!\cdot \times \cdot\!\longrightarrow\!\cdot$, describes composite of arrows:

$$\text{Comp}(x, y, z): \qquad \exists_{\cdot\rightrightarrows\cdot}\, w, \quad \text{restr}_{\cdot\to\cdot}(w) \approx x \ \wedge\ \text{restr}_{\cdot\to\cdot}(w) \approx y \ \wedge\ \text{restr}_{\cdot\to\cdot}(w) \approx z \ \wedge\ \text{commute}(w)$$

while the following one ensures the existence of compositions:

$$\text{CompE}: \qquad \forall_{\cdot\to\cdot} x, y, \quad \text{restr}_{\cdot\to\cdot}(x) \approx \text{restr}_{\cdot\to\cdot}(y) \ \to\ \exists_{\cdot\to\cdot} z,\ \text{Comp}(x, y, z).$$

Equality of nontrivial paths

For any two nontrivial path-quivers P_1 and P_2, it is possible to choose in a canonical way a quiver Q' together with two embeddings m'_1, m'_2 such that the following diagram forms a pushout configuration (as for instance on Figure 2):

$$\begin{array}{ccc} \cdot\cdot & \xrightarrow{\mathsf{st}_{P_1}} & P_1 \\ \downarrow{\scriptstyle \mathsf{st}_{P_2}} & & \downarrow{\scriptstyle m'_1} \\ P_2 & \xrightarrow{m'_2} & Q' \end{array}$$

We thus define the following predicate of arity $P_1 \times P_2$:

$$\text{EqPath}_{P_1,P_2}(x_1, x_2): \qquad \text{restr}_{\cdot\rightsquigarrow\cdot}(x_1) \approx \text{restr}_{\cdot\rightsquigarrow\cdot}(x_2) \ \wedge\ \big(\forall_{Q'}\, x,\ \text{Cospan}_{m'_1,m'_2}(x_1, x_2, x) \to \text{commute}(x)\big).$$

Identity

The following predicate, of arity $\cdot \times \cdot \longrightarrow \cdot$, defines the identity map:

$$\begin{aligned}\mathrm{Id}(x,y)\colon \quad & \mathrm{restr}_{\cdot\to\cdot}(y) \approx x \quad \wedge \quad \forall_{\cdot\to\cdot} z, w, \\ & \big(\mathrm{Comp}(y,z,w) \to \mathrm{EqPath}(z,w)\big) \ \wedge \ \big(\mathrm{Comp}(z,y,w) \to \mathrm{EqPath}(z,w)\big)\end{aligned}$$

and the following formula ensures the existence of identity maps.

$$\mathrm{IdE}\colon \quad \forall_{\cdot} x, \exists_{\cdot\to\cdot} y, \quad \mathrm{Id}(x,y).$$

Note that $\mathrm{Id}(x,y)$ and IdE entail that $\mathrm{restr}_{\cdot\to\cdot}(y) \approx x$.

Equality for general paths

For any nontrivial path-quiver P, we define the following formulas:

$$\begin{aligned}&\mathrm{EqPath}_{\cdot,\cdot}(x,y)\colon && x \approx y,\\ &\mathrm{EqPath}_{\cdot,P}(x,y)\colon && \exists_{\cdot\to\cdot} z, \quad \mathrm{Id}(x,z) \ \wedge \ \mathrm{EqPath}_{\cdot\to\cdot,P}(z,y),\\ &\mathrm{EqPath}_{P,\cdot}(x,y)\colon && \mathrm{EqPath}_{\cdot,P}(y,x).\end{aligned}$$

Hence, we can see EqPath as a relation with arity any pair of path-quivers. We ensure this relation to be an equivalence relation by defining for any path-quivers P_1, P_2 and P_3 the three formulas $\mathrm{EqPathRefl}_{P_1}$, $\mathrm{EqPathSym}_{P_1,P_2}$ and $\mathrm{EqPathTrans}_{P_1,P_2,P_3}$ stating that the relation EqPath is respectively reflexive, symmetric and transitive.

We also make sure to enforce the properties of a category congruence. For this purpose, for any four path-quivers P_1, P_1', P_2, and P_2', of respective length k_1, k_1', k_2 and k_2', we define the following formula where bound variables x_1, x_2, x_1', x_2' respectively have sort P_1, P_2, P_1' and P_2' and the sort of x_i'', $i \in \{1,2\}$, is $\mathrm{PQ}_{k_i+k_i'}$.

$$\begin{aligned}&\mathrm{EqPathConcat}_{P_1,P_2,P_1',P_2'}\colon \quad \forall x_1, x_2, x_1', x_2',\\ &\quad \mathrm{EqPath}(x_1,x_2) \ \wedge \ \mathrm{EqPath}(x_1',x_2') \ \wedge \ \mathrm{restr}_{\cdot\smile\cdot}(x_1) \approx \mathrm{restr}_{\cdot\smile\cdot}(x_1')\\ &\quad \to \quad \forall x_1'', x_2'', \quad \mathrm{Cospan}_{\smile,\smile}(x_1,x_1',x_1'') \ \wedge \ \mathrm{Cospan}_{\smile,\smile}(x_2,x_2',x_2'')\\ &\qquad\qquad \to \quad \mathrm{EqPath}(x_1'',x_2'').\end{aligned}$$

Commutativity

We relate commutativity and equality by the following formula:

$$\mathrm{ComEq}\colon \quad \forall_{\cdot\circlearrowright\cdot} x, \quad \mathrm{commute}(x) \ \to \ \mathrm{restr}_{\cdot\circlearrowright\cdot}(x) \approx \mathrm{restr}_{\cdot\circlearrowleft\cdot}(x).$$

For any quiver Q, the following formula provides an analogue of the notion of commutativity of diagrams via the characterization given in Proposition 16,

$$\mathrm{PathCom}_Q\colon \quad \forall_Q x, \quad \bigwedge_{(p_1,p_2)\in\mathcal{BT}_Q} \mathrm{EqPath}(\mathrm{restr}_{p_1}(x), \mathrm{restr}_{p_2}(x)) \quad \leftrightarrow \quad \mathrm{commute}(x).$$

where we recall that $\mathcal{BT}_Q$ denotes the (finite) set of pairs of trails of Q having the same extremities.

▶ **Definition 19.** *Theory $\mathring{\mathcal{T}}_{\mathrm{cat}}$, over signature $\mathring{\Sigma}$, consists of the following formulas:*

- EmptyEU, CompE, IdE, ComEq,
- $\mathrm{RestrComp}_{m,m'}$ *for any pair of maps m and m' such that the codomain of m is the domain of m',*
- $\mathrm{PushoutEU}_{m_1,m_2,m_1',m_2'}$ *for a pushout configuration as in Definition 9,*
- $\mathrm{EqPathRefl}_{P_1}, \mathrm{EqPathSym}_{P_1,P_2}, \mathrm{EqPathTrans}_{P_1,P_2,P_3}, \mathrm{EqPathConcat}_{P_1,P_2,P_3,P_4}$ *for any quadruple of path-quivers P_1, P_2, P_3 and P_4,*
- $\mathrm{PathCom}_Q$ *for any quiver Q.*

Theory $\mathcal{T}_{\mathrm{cat}}$ is defined as the restriction of $\mathring{\mathcal{T}}_{\mathrm{cat}}$ to Σ.

3.2 Models

Models of $\mathcal{T}_{\mathrm{cat}}$, resp. $\mathring{\mathcal{T}}_{\mathrm{cat}}$, are in fact exactly what we have called categorical interpretations.

▶ **Theorem 20.** *Every categorical interpretation of Σ, resp. $\mathring{\Sigma}$, is a model of $\mathcal{T}_{\mathrm{cat}}$, resp. $\mathring{\mathcal{T}}_{\mathrm{cat}}$. Moreover, any model $\mathcal{M}$ of $\mathcal{T}_{\mathrm{cat}}$, resp. $\mathring{\mathcal{T}}_{\mathrm{cat}}$, has an isomorphic categorical interpretation.*

Proof. We only prove the theorem for $\mathcal{T}_{\mathrm{cat}}$, as the proof for $\mathring{\mathcal{T}}_{\mathrm{cat}}$ is similar.

If $\mathcal{C}$ is a small category, a routine check shows that the associated model $\mathcal{C}$ of Σ verifies the theory $\mathcal{T}_{\mathrm{cat}}$. For instance,

- the formulas PushoutEU follows from Lemma 11;
- IdE and CompE come from the existence of the identity map and the existence of the composition respectively;
- for two diagrams D_1 and D_2 respectively over path-quivers P_1 and P_2, the formula $\mathrm{EqPath}_{P_1,P_2}(D_1, D_2)$ corresponds to the relation $\mathrm{comp}(D_1) = \mathrm{comp}(D_2)$, which is a path relation;
- ComEq and $\mathrm{PathCom}_Q$ follow from Proposition 16.

Let us prove the other direction. Let Q be an acyclic quiver. By an abuse of the notations, if $v \in V_Q$, resp. $a \in A_Q$, we also denote by v, resp. a, the corresponding embedding $v\colon \cdot \hookrightarrow Q$, resp. $a\colon \cdot \longrightarrow \cdot \hookrightarrow Q$. Here is a crucial lemma for the proof of the theorem.

▶ **Lemma 21** (General pushout)**.** *Let $\mathcal{M}$ be a model of $\mathcal{T}_{\mathrm{cat}}$. Let Q be an acyclic quiver, $\beta_V\colon V_Q \to \mathcal{M}_{\cdot}$ and $\beta_A\colon A_Q \to \mathcal{M}_{\cdot\to\cdot}$ be two maps. Then the following statements are equivalent.*

1. *There exists an element β of $\mathcal{M}_Q$ such that, for any $v \in V_Q$ and any $a \in A_Q$, $\mathrm{restr}_v^{\mathcal{M}}(\beta) = \beta_V(v)$ and $\mathrm{restr}_a^{\mathcal{M}}(\beta) = \beta_A(a)$.*
2. *For any $a \in A_Q$, $\mathrm{restr}^{\mathcal{M}}_{\cdot\to\cdot}(\beta_A(a)) = \beta_V(s_Q(a))$ and $\mathrm{restr}^{\mathcal{M}}_{\cdot\to\cdot}(\beta_A(a)) = \beta_V(t_Q(a))$.*

Moreover, when both statements hold, the element β is unique.

Proof. The fact that the first point induces the second one follows directly from RestrComp. Hence we focus on the other direction and on the uniqueness.

We proceed by induction on the structure of Q. In the case of an empty Q, the second point holds trivially. The first point and the uniqueness follows from EmptyEU.

Assume first that the lemma holds for some quiver Q_1 with no arrow. Let Q be the quiver Q_1 with an extra vertex v_0. Let β_V and β_A be two maps as in the statement of the lemma, and $\beta_{1,V}$ the restriction of β_V to V_{Q_1}. We have the following pushout configuration:

$$\begin{array}{ccc} \varnothing & \xrightarrow{m_1} & Q_1 \\ \downarrow{\scriptstyle m_2} & & \downarrow{\scriptstyle m_1'} \\ \bullet & \xrightarrow[m_2'=v_0]{} & Q \end{array}$$

Point 2 holds trivially. Let us prove the existence and uniqueness of an element β satisfying the property of Point 1. By induction, we get a unique $\beta_1 \in \mathcal{M}_{Q_1}$ compatible with $\beta_{1,V}$ and β_A. Set $\beta_2 \coloneqq \beta_V(v_0)$. By EmptyEU, $\mathrm{restr}^{\mathcal{M}}_{m_1}(\beta_1) = \mathrm{restr}^{\mathcal{M}}_{m_2}(\beta_2)$. Hence we can apply $\mathrm{PushoutEU}_{m_1,m_2,m_1',m_2'}$ to get a unique element $\beta \in \mathcal{M}_Q$ such that $\mathrm{Cospan}^{\mathcal{M}}_{m_1',m_2'}(\beta_1,\beta_2,\beta)$. From RestrComp and the induction hypothesis, it follows that for $\beta' \in \mathcal{M}_Q$, $\mathrm{Cospan}^{\mathcal{M}}_{m_1',m_2'}(\beta_1,\beta_2,\beta')$ is equivalent to $\mathrm{restr}^{\mathcal{M}}_v(\beta') = \beta_V(v)$ for any $v \in V_Q$. This concludes the induction.

Assume more generally that the lemma holds for some acyclic quiver Q_1. Let Q be an acyclic quiver obtained from Q_1 by adding one arrow a_0. Let β_V and β_A be two maps as before and $\beta_{1,A}$ the restriction of β_A to A_{Q_1}. Let $m_1 \colon \bullet\,\bullet \hookrightarrow Q_1$ mapping the first point to $s_Q(a_0)$ and the second point to $t_Q(a_0)$. Let $m_2 \coloneqq \mathsf{st}_{\bullet\to\bullet} = \bullet\longrightarrow\bullet$. Once again, we get a pushout configuration:

$$\begin{array}{ccc} \bullet\,\bullet & \xrightarrow{m_1} & Q_1 \\ \downarrow{\scriptstyle m_2} & & \downarrow{\scriptstyle m_1'} \\ \bullet\longrightarrow\bullet & \xrightarrow[m_2':=a_0]{} & Q \end{array}$$

Assume that Point 2 holds. By induction, we get a unique element $\beta_1 \in \mathcal{M}_{Q_1}$ compatible with β_V and $\beta_{1,A}$. Set $\beta_2 \coloneqq \beta_A(a_0)$. We have already proven the lemma for the quiver $\bullet\,\bullet$. Hence we deduce that

$$\mathrm{restr}^{\mathcal{M}}_{m_1}(\beta_1) = \mathrm{restr}^{\mathcal{M}}_{m_2}(\beta_2).$$

We can apply PushoutEU as before to get Point 1 as well as the uniqueness part. This concludes the proof of the lemma. ◀

We now continue the proof of Theorem 20. Let $\mathcal{M}$ be a model of $\mathcal{T}_{\mathrm{cat}}$. We define the general quiver $\mathcal{Q}$ associated to $\mathcal{M}$ by

$$V_{\mathcal{Q}} \coloneqq \mathcal{M}_{\bullet}, \quad A_{\mathcal{Q}} \coloneqq \mathcal{M}_{\bullet\to\bullet}, \quad s_{\mathcal{Q}} \coloneqq \mathrm{restr}^{\mathcal{M}}_{\bullet\to\bullet} \text{ and } \quad t_{\mathcal{Q}} \coloneqq \mathrm{restr}^{\mathcal{M}}_{\bullet\to\bullet}.$$

Set $\widetilde{\mathcal{C}} \coloneqq \langle\mathcal{Q}\rangle$. Thanks to Lemma 21, to each acyclic quiver Q and to each element $\beta \in \mathcal{M}_Q$, we can associate a unique diagram $\widetilde{\Psi}(\beta)$ in $\widetilde{C}$ verifying:

- for any $v \in V_Q$, $\widetilde{\Psi}(\beta)(v) = \mathrm{restr}^{\mathcal{M}}_v(\beta)$.
- for any $a \in A_Q$, $\widetilde{\Psi}(\beta)(a)$ is the path of length one with arrow $\mathrm{restr}^{\mathcal{M}}_a(\beta)$.

The image of $\widetilde{\Psi}$ is exactly the set of diagrams whose morphisms are paths of lengths one.

Let us now introduce another important map. Let A and B be two objects of $\widetilde{\mathcal{C}}$, and let $p \in \mathrm{Hom}_{\widetilde{\mathcal{C}}}(A,B)$. Recall that p is just a path from A to B in $\mathcal{Q}$. Let k be the length of p. By Lemma 21, there exists a unique element $\Theta(p) \in \mathcal{M}_{\mathrm{PQ}_k}$ such that

- for each $v \in V_{\mathrm{PQ}_k}$, $\mathrm{restr}^{\mathcal{M}}_v(\Theta(p)) = p(v)$,
- for each $a \in A_{\mathrm{PQ}_k}$, $\mathrm{restr}^{\mathcal{M}}_a(\Theta(p)) = p(a)$.

Relation $\mathrm{EqPath}^{\mathcal{M}}$ thus induces a relation r on morphisms of $\widetilde{\mathcal{C}}$. Moreover, EqPathRefl, EqPathSym, EqPathTrans and EqPathConcat, together with Lemma 21, make r a category congruence. We can hence define the category $\mathcal{C} \coloneqq \widetilde{\mathcal{C}}/r$. Now $\widetilde{\Psi}$ induces a map $\Psi\colon \mathcal{M}_Q \to \mathcal{C}_Q$ for any quiver Q, and we claim that Ψ induces a model isomorphism between $\mathcal{M}$ and $\mathcal{C}$.

Let Q be any acyclic quiver. We first prove that Ψ is injective. Let $\beta, \gamma \in \mathcal{M}_Q$ such that $\Psi(\beta) = \Psi(\gamma)$. For any vertex $v \in V_Q$, $\widetilde{\Psi}(\beta)(v) = \widetilde{\Psi}(\gamma)(v)$, i.e., $\mathrm{restr}_v^{\mathcal{M}}(\beta) = \mathrm{restr}_v^{\mathcal{M}}(\gamma)$. Let a be an arrow of Q. Then we have the relation $\widetilde{\Psi}(\beta)(a) \sim \widetilde{\Psi}(\gamma)(a)$. By definition of $\mathrm{EqPath}_{\bullet\to\bullet\,\bullet\to\bullet}$, we validate the premise of ComEq, and thus the equality $\widetilde{\Psi}(\beta)(a) = \widetilde{\Psi}(\gamma)(a)$, i.e., $\mathrm{restr}_a^{\mathcal{M}}(\beta) = \mathrm{restr}_a^{\mathcal{M}}(\gamma)$. By the uniqueness part of Lemma 21, we get $\beta = \gamma$.

We now consider the surjectivity of Ψ. It suffices to prove that any morphism $p \in \mathrm{Hom}_{\widetilde{\mathcal{C}}}(A, B)$, for any $A, B \in \mathrm{Ob}_{\widetilde{\mathcal{C}}}$, is in relation via r to a path of length one. Indeed, in such a case, for any diagram $\widetilde{D}$ in $\widetilde{\mathcal{C}}$ over Q, one can find another diagram $\widetilde{D}'$ over Q whose morphisms are path of size one and such that any morphism of $\widetilde{D}$ is in relation with the corresponding morphism of $\widetilde{D}'$. Hence the induced diagrams in $\mathcal{C}$ are equal. Moreover, $\widetilde{D}'$ is in the image of $\widetilde{\Psi}$, and we would get the surjectivity.

Let P be a path-quiver and let $\beta \in \mathcal{M}_P$. We have to find an element $\gamma \in \mathcal{M}_{\bullet\to\bullet}$ such that $\mathrm{EqPath}^{\mathcal{M}}_{P,\bullet\to\bullet}(\beta, \gamma)$. If P has length one, this is trivial. If P has length zero, then by IdE, there exists γ such that $\mathrm{Id}^{\mathcal{M}}(\beta, \gamma)$. Moreover, by the definition $\mathrm{EqPath}_{\bullet,\bullet\to\bullet}$ and using the reflexivity of EqPath, we get $\mathrm{EqPath}^{\mathcal{M}}_{\bullet,\bullet\to\bullet}(\beta, \gamma)$. If P has length two, then by CompE, we can find an element γ such that

$$\mathrm{Comp}(\mathrm{restr}^{\mathcal{M}}_{\bullet\to\bullet\to\bullet}(\beta), \mathrm{restr}^{\mathcal{M}}_{\bullet\to\bullet\to\bullet}(\beta), \gamma).$$

The commutativity of the triangle induces $\mathrm{EqPath}^{\mathcal{M}}_{\bullet\to\bullet\to\bullet,\bullet\to\bullet}(\beta, \gamma)$.

For P with length $k > 2$, we proceed by induction on the length of P. Using EqPathTrans, it suffices to find γ over PQ_{k-1} such that $\mathrm{EqPath}^{\mathcal{M}}_{P,\mathrm{PQ}_{k-1}}(\beta, \gamma)$. To do so, we see P as the pushout of PQ_2 and PQ_{k-2} along $m_1' = \rightsquigarrow\colon \mathrm{PQ}_2 \to P$ and $m_2' = \rightsquigarrow\colon \mathrm{PQ}_{k-2} \to P$. By the case $k = 2$, we can find $\gamma_1 \in \mathcal{M}_{\bullet\to\bullet}$ such that $\mathrm{EqPath}(\mathrm{restr}^{\mathcal{M}}_{m_1'}(\beta), \gamma_1)$. We set $\gamma_2 = \mathrm{restr}^{\mathcal{M}}_{m_2'}(\beta)$. In particular we have $\mathrm{EqPath}(\gamma_2, \gamma_2)$. By EqPathConcat, we get $\mathrm{EqPath}(\beta, \gamma)$ where $\gamma \in \mathcal{M}_{\mathrm{PQ}_{k-1}}$ is such that $\mathrm{Cospan}_{m_1', m_2'}(\gamma_1, \gamma_2, \gamma)$, and the result follows.

We have shown that Ψ induces a bijection between the corresponding domains. In order to conclude the proof, it remains to prove that Ψ commutes with function restr and with predicate commute. The commutativity with restr follows from the definition of Ψ and the formulas RestrComp.

The compatibility with the predicate commute can be reduced to the compatibility of EqPath via the formula PathCom and the characterization of commutativity given in Proposition 16. Let P_1 and P_2 be two path-quivers and $\beta_1 \in \mathcal{M}_{P_1}$ and $\beta_2 \in \mathcal{M}_{P_2}$. These elements correspond to paths p_1 and p_2 in $\mathcal{Q}$. Using the definitions of the different elements, we get the following chain of equivalences:

$$\begin{aligned}\mathrm{EqPath}^{\mathcal{C}}(\Psi(\beta_1), \Psi(\beta_2)) &\Leftrightarrow \mathrm{comp}_{\mathcal{C}}(\Psi(\beta_1)) = \mathrm{comp}_{\mathcal{C}}(\Psi(\beta_2)) \\ &\Leftrightarrow \mathrm{comp}_{\widetilde{\mathcal{C}}}(\widetilde{\Psi}(\beta_1)) \sim \mathrm{comp}_{\widetilde{\mathcal{C}}}(\widetilde{\Psi}(\beta_2)) \Leftrightarrow p_1 \sim p_2 \Leftrightarrow \mathrm{EqPath}(\beta_1, \beta_2).\end{aligned}$$

This concludes the proof of the theorem. ◀

4 Duality

Signatures of Definition 1 are tailored to enforce a built-in, therefore easy to prove, duality principle, which we now make precise. Recall from Definition 2 that duality is an involution on quivers, as well as on acyclic quivers. We define the dual of a formula over $\mathring{\Sigma}$ as follows:

- if $m\colon Q \to Q'$ is a morphism, then $m^\dagger\colon Q^\dagger \to Q'^\dagger$ is defined by $m^\dagger_V = m_V$ and $m^\dagger_A = m_A$.
- if $x = (\mathrm{x}, Q)$ is a variable in $X \times \mathring{S}$ then $x^\dagger := (\mathrm{x}, Q^\dagger)$,
- $(\mathrm{restr}_m(x))^\dagger := \mathrm{restr}_{m^\dagger}(x^\dagger)$ and $(\mathrm{commute}_Q(x))^\dagger := \mathrm{commute}_{Q^\dagger}(x^\dagger)$,
- $(x \approx y)^\dagger := x^\dagger \approx y^\dagger$,
- $(\forall_Q\, x, \phi)^\dagger := \forall_{Q^\dagger}\, x^\dagger, \phi^\dagger$ and $(\exists_Q\, x, \phi)^\dagger := \exists_{Q^\dagger}\, x^\dagger, \phi^\dagger$,
- $(\phi \wedge \psi)^\dagger := \phi^\dagger \wedge \psi^\dagger$, etc.

For Y a set of variables, $Y^\dagger$ denotes $\{x^\dagger \mid x \in Y\}$. For any theory $\mathcal{T}$, $\mathcal{T}^\dagger$ denotes $\{\phi^\dagger \mid \phi \in \mathcal{T}\}$.

For $\mathcal{M}$ an interpretation of $\mathring{\Sigma}$ over a set of variables Y, we define its dual model $\mathcal{M}^\dagger$ over $Y^\dagger$ as:

- $\mathcal{M}^\dagger_Q := \mathcal{M}_{Q^\dagger}$,
- for $x \in Y^\dagger$, $x^{\mathcal{M}^\dagger} := (x^\dagger)^{\mathcal{M}}$.
- $\mathrm{restr}^{\mathcal{M}^\dagger}_m := \mathrm{restr}^{\mathcal{M}}_{m^\dagger}$ and $\mathrm{commute}^{\mathcal{M}^\dagger}_Q := \mathrm{commute}^{\mathcal{M}}_{Q^\dagger}$,

The duality involution also induces involutions respectively on formulas, theories and models over Σ.

▶ **Example 22.** If $\mathcal{C}$ is a small category, then the dual interpretation $\mathcal{C}^\dagger$ is isomorphic to the model of the dual category, both with respect to $\mathring{\Sigma}$ and to Σ.

▶ **Theorem 23** (Duality theorem). *Let ϕ be a formula with free variables included in $Y \subseteq X \times S$, resp. in $Y \subseteq X \times \mathring{S}$, and let $\mathcal{M}$ be a model of Σ, resp. of $\mathring{\Sigma}$. Then*

$$\mathcal{M} \models \phi \quad \iff \quad \mathcal{M}^\dagger \models \phi^\dagger.$$

Proof. The proof is direct. ◀

▶ Remark 24. The duality principle has some useful direct consequences:

- If ϕ is a valid, resp. satisfiable, resp. unsatisfiable, formula, then so is $\phi^\dagger$.
- Let $\mathcal{T}$ be theory such that any model of $\mathcal{T}$ verifies $\mathcal{T}^\dagger$. If ϕ is a valid, resp. satisfiable, resp. unsatisfiable, formula among models of $\mathcal{T}$, so is $\phi^\dagger$.
- We have the following reciprocal. Let $\mathcal{T}$ be a theory such that any model $\mathcal{M}$ of $\mathcal{T}$ verifies that $\mathcal{M}^\dagger \models \mathcal{T}$, then every model of $\mathcal{T}$ verifies $\mathcal{T}^\dagger$.

The following fact follows directly from this last point, Theorem 20 and Example 22.

▶ **Proposition 25.** *Models of $\mathcal{T}_{\mathrm{cat}}$ verify $\mathcal{T}^\dagger_{\mathrm{cat}}$, and models of $\mathring{\mathcal{T}}_{\mathrm{cat}}$ verify $\mathring{\mathcal{T}}^\dagger_{\mathrm{cat}}$.*

5 A theory for diagrams in abelian categories

We now introduce a theory whose models are diagrams in small abelian categories. We rely on the set of axioms given by Freyd in [7]. This reference is particularly well-suited for our purpose. Indeed, the author does not impose the homomorphisms between any two objects of an abelian category to form an abelian group, but this fact rather follows from the axioms.

Let us first formulate common notions of category theory in the languages introduced in Section 2.5. Here is a predicate of arity $\bullet \longrightarrow \bullet$ which corresponds to monicity of a map in a category.

$$\begin{aligned}\mathrm{Mono}(x) : \quad & \forall_{\bullet \rightrightarrows \bullet \to \bullet}\, y, \quad \mathrm{restr}_{\bullet \rightrightarrows \bullet \to \bullet}(y) \approx x \\ & \wedge\ \mathrm{commute}(\mathrm{restr}_{\bullet \rightrightarrows \bullet \to \bullet}(y)) \ \wedge\ \mathrm{commute}(\mathrm{restr}_{\bullet \rightrightarrows \bullet \to \bullet}(y)) \\ & \to\ \mathrm{commute}(\mathrm{restr}_{\bullet \rightrightarrows \bullet \to \bullet}(y)).\end{aligned}$$

The dual predicate to $\mathrm{Mono}(x)$ is named $\mathrm{Epi}(x)$.

Let Q be a quiver. We define the *cone of* Q as the quiver

$$\mathrm{cone}(Q) \coloneqq (V_Q \sqcup \{v_0\},\ A_Q \sqcup \{a_v \mid v \in V_Q\}, s_{\mathrm{cone}(Q)}, t_{\mathrm{cone}(Q)}),$$

where $s_{\mathrm{cone}(Q)}$ and $t_{\mathrm{cone}(Q)}$ are extensions of s_Q and t_Q by $s_Q(a_v) = v_0$ and $t_Q(a_v) = v$. We define by $i_Q \colon Q \hookrightarrow \mathrm{cone}(Q)$ the corresponding embedding. If $m \colon Q \to Q'$ is a morphism of quivers, we get a canonical morphism $\mathrm{cone}(m) \colon \mathrm{cone}(Q) \to \mathrm{cone}(Q')$.

Abusing the notations, if a is an arrow of Q, we also denote by $a \colon \bullet\longrightarrow\bullet \hookrightarrow Q$ the corresponding morphism. We then introduce the usual notion of cones of diagrams by the following formula of arity $Q \times \mathrm{cone}(Q)$.

$$\mathrm{Cone}_Q(x,y) \colon \qquad \mathrm{restr}_{i_Q}(y) \approx x \quad \wedge \quad \bigwedge_{a \in A_Q} \mathrm{commute}(\mathrm{restr}_{\mathrm{cone}(a)}(y)).$$

We now formulate the notion of limit.

$$\begin{aligned} \mathrm{Limit}_Q(x,y) \colon \qquad & \mathrm{Cone}_Q(x,y) \quad \wedge \\ & \forall_{\mathrm{cone}(Q)}\, z, \quad \mathrm{Cone}_Q(x,z) \ \to\ \exists!_{\mathrm{cone}(\mathrm{cone}(Q))}\, w,\ \mathrm{Cone}(y,w) \wedge \mathrm{restr}_{\mathrm{cone}(i_Q)}(w) \approx z. \end{aligned}$$

We also introduce the dual $\mathrm{Colimit}_Q(x,y) \coloneqq \mathrm{Limit}^{\dagger}_{Q^{\dagger}}(x,y)$.

The introduction of monos, epis, limits and colimits allows to state Freyd's axioms of abelian categories [7]. First, we define zero objects and kernels of respective arity $\bullet$ and $\bullet\longrightarrow\bullet \times \bullet\longrightarrow\bullet$ as follows:

$$\begin{aligned} \mathrm{Zero}(x) \colon \qquad & \forall_{\varnothing}\, y, \quad \mathrm{Limit}_{\varnothing}(y,x) \ \wedge\ \mathrm{Colimit}_{\varnothing}(y,x), \\ \mathrm{Ker}(x,y) \colon \qquad & \exists_{\bullet \rightrightarrows \bullet} z, \qquad \mathrm{restr}_{\bullet\to\bullet}(z) \approx x \quad \wedge \quad \mathrm{restr}_{\bullet\to\bullet}(z) \approx y \\ & \wedge \quad \mathrm{Zero}(\mathrm{restr}_{\bullet}(z)) \quad \wedge \quad \mathrm{Limit}(\mathrm{restr}_{\bullet\to\bullet}(z), z). \end{aligned}$$

We also define $\mathrm{Coker}(x,y) \coloneqq \mathrm{Ker}^{\dagger}(x,y)$.

We define the category $\mathcal{T}_{\mathrm{ab}}$, resp. $\mathring{\mathcal{T}}_{\mathrm{ab}}$ as the extension of $\mathcal{T}_{\mathrm{cat}}$, resp. $\mathring{\mathcal{T}}_{\mathrm{cat}}$, by the following formulas.

$$\begin{aligned} \text{ZeroE}: \qquad & \exists_{\bullet} x, \quad \mathrm{Zero}(x), \\ \text{ProductE}: \qquad & \forall_{\bullet\bullet} x, \exists_{\mathrm{cone}(\bullet\bullet)} y, \quad \mathrm{Limit}_{\bullet\bullet}(x,y), \\ \text{CoproductE}: \qquad & \text{ProductE}^{\dagger}, \\ \text{KerE}: \qquad & \forall_{\bullet\to\bullet} x, \exists_{\bullet\to\bullet} y, \quad \mathrm{Ker}(x,y), \\ \text{CokerE}: \qquad & \text{KerE}^{\dagger}, \\ \text{MonoNormal}: \qquad & \forall_{\bullet\to\bullet} x, \quad \mathrm{Mono}(x) \ \to\ \exists_{\bullet\to\bullet} y,\ \mathrm{Ker}(y,x), \\ \text{EpiNormal}: \qquad & \text{MonoNormal}^{\dagger}. \end{aligned}$$

The following theorem states that $\mathcal{T}_{\mathrm{ab}}$ is a theory for diagrams in abelian categories.

▶ **Theorem 26.** *The categorical interpretation induced by any small abelian category is a model of* $\mathcal{T}_{\mathrm{ab}}$*. Conversely, any model of* $\mathcal{T}_{\mathrm{ab}}$ *is isomorphic to the categorical interpretation associated to some small abelian category.*

Proof. This follows from Theorem 20 and from [7, Chapter 2]. ◀

▶ **Proposition 27.** *The theory $\mathcal{T}_{\mathrm{ab}}$ implies its dual $\mathcal{T}_{\mathrm{ab}}^{\dagger}$.*

Proof. The theory $\mathcal{T}_{\mathrm{cat}}$ implies its dual by Proposition 25. Moreover, ZeroE clearly implies its dual. Finally, the other axioms have their dual in the theory, by definition thereof. ◀

▶ Remark 28. The theorem and the proposition also hold for theory $\mathring{\mathcal{T}}_{\mathrm{ab}}$.

6 Decidability of the commerge problem

In this section, we use the notations of Section 2.3. Let Q be a quiver, $k \in \mathbb{N}$ and, for each $i \in [k]$, let Q_i be a quiver and $m_i \colon Q_i \to Q$ be a morphism. We define the following formula:

$$\mathrm{Commerge}_{m_0,\dots,m_{k-1}} : \qquad \forall_Q\, x, \qquad \bigwedge_{i=0}^{k-1} \mathrm{commute}(\mathrm{restr}_{m_i}(x)) \quad \to \quad \mathrm{commute}(x).$$

▶ **Definition 29.** *Notations as above, the* acyclic, resp. cyclic, commerge problem for morphisms, or embeddings, *$m_0, \dots, m_{k-1}$ and for a theory $\mathcal{T}$ on language Σ, resp. $\mathring{\Sigma}$, is the problem of deciding the validity of $\mathrm{Commerge}_{m_0,\dots,m_{k-1}}$ in models of theory $\mathcal{T}$.*

We recall that a *thin category* is a category with at most one morphism between any pair of objects. Let $\mathrm{tot}_{Q_i} = \mathcal{BP}_{Q_i}$ be the complete path relation on Q_i. Set $r_i \coloneqq m_{i\,*}(\mathrm{tot}_{Q_i})$ for $i \in [k]$. Recall that $(r_i \mid i \in [k])$ is the smallest path relation containing the r_i for all $i \in [k]$.

▶ **Lemma 30.** *Notations as above, the formula $\mathrm{Commerge}_{m_0,\dots,m_{k-1}}$ is valid for models of $\mathcal{T}_{\mathrm{cat}}$, resp. $\mathring{\mathcal{T}}_{\mathrm{cat}}$, if and only if $\langle Q \rangle / (r_i \mid i \in [k])$ is a thin category.*

Proof. Set $\mathcal{C} \coloneqq \langle Q \rangle / (r_i \mid i \in [k])$. It is a model of $\mathcal{T}_{\mathrm{cat}}$, resp. $\mathring{\mathcal{T}}_{\mathrm{cat}}$. Moreover, the canonical diagram $D \colon \langle Q \rangle \to \mathcal{C}$ verifies the premise of $\mathrm{Commerge}_{m_0,\dots,m_{k-1}}$. If $\mathcal{C}$ is not thin, then there are two paths p and q in $\langle Q \rangle$ with the same extremities which are not in relation. Then $\mathrm{comp}(p^*(D))$ is the class of p in the quotient, which is different of the class of q, that is of $\mathrm{comp}(q^*(D))$. Hence D is not commutative.

For the other direction, by Theorem 20, it suffices to study diagrams in small categories. It is easy to check that any diagram D' over Q in a category $\mathcal{C}'$ which verifies the condition of $\mathrm{Commerge}_{m_0,\dots,m_{k-1}}$ factors through D, i.e., $D' = F \circ D$ for some functor $F \colon \mathcal{C} \to \mathcal{C}'$. If $\mathcal{C}$ is thin, then for any two paths p and q with same extremities in Q, we have:

$$\mathrm{comp}(p^*(D')) = F(\mathrm{comp}(p^*(D))) = F(\mathrm{comp}(q^*(D))) = \mathrm{comp}(q^*(D')).$$

Hence $\mathrm{Commerge}_{m_0,\dots,m_{k-1}}$ is valid. ◀

▶ **Theorem 31.** *The acyclic commerge problem for embeddings and $\mathcal{T}_{\mathrm{cat}}$ is decidable.*

Proof. By Lemma 30, it suffices to decide if $\langle Q \rangle / (r_i \mid i \in [k])$ is a thin category. Since the set of paths in $\langle Q \rangle$ is finite, we can compute the relation $(m_{i\,*}(\mathrm{tot}_{Q_i}) \mid i \in [k])$ extensively and check whether it is complete. ◀

▶ **Proposition 32.** *The cyclic commerge problem for morphisms and $\mathring{\mathcal{T}}_{\mathrm{cat}}$ is undecidable.*

Proof. We proceed by reduction to an undecidability result, independently due to Adyan and Rabin [2, 17]. For B an arbitrary finite set and $\langle B \rangle$ the associated free monoid, let M be the finitely presentable monoid $\langle B \mid r = 1, r \in R \rangle$, for R a nonempty finite subset of $\langle B \rangle \setminus \{1\}$. In particular, any finitely presentable group is of this form. Hence, by the Adyan-Rabin theorem, determining the triviality of M from B and R is undecidable.

Let B, R and M as above. Let $Q = (\{v\}, B \sqcup \{e\}, s_Q, t_Q)$ be a quiver with one vertex and loops labeled by elements of B plus one loop e. To each element $\rho \in R$ corresponds a path $p_\rho \colon \mathrm{PQ}_{k_\rho} \to Q$, for some $k_\rho \in \mathbb{N}$. Let Q_ρ be a pushout of the morphisms $\bullet\!\rightharpoonup\!\bullet \colon \bullet\,\bullet \hookrightarrow \mathrm{PQ}_{k_\rho}$ and $\bullet \longrightarrow \bullet$. Let m_ρ be the extension to Q_ρ of p_ρ obtained by mapping the new arrow onto e. Also set $m_e \colon \circlearrowright \to Q$ which maps the loop on e. Now, $\langle Q\rangle/\big(m_e^*(\mathrm{tot}_{\circlearrowright}), (m_\rho^*(\mathrm{tot}_{Q_\rho}) \mid \rho \in R)\big)$ is the category associated to the monoid M. Hence this category is thin if and only if the monoid is trivial. The Adyan-Rabin theorem and Lemma 30 conclude the proof. ◀

We strengthen the previous proposition to the case of embeddings.

▶ **Theorem 33.** *The cyclic commerge problem for embeddings and $\mathring{\mathcal{T}}_{\mathrm{cat}}$ is undecidable.*

Proof. Let B, R, M as in the proof of Proposition 32. Let Q be the quiver $(\{v\}, B, s_Q, t_Q)$ (note that we removed the arc e). If $n \geq 2$, we define the quiver $\mathring{Q}^n$ as

$$\mathring{Q}^n \coloneqq \Big(\{v_i \mid i \in [n]\}, \{b_{i,j} \mid b \in B, 0 \leq i < j < n\} \sqcup \{e_{i,j} \mid i,j \in [n]\}, s_{\mathring{Q}^n}, t_{\mathring{Q}^n}\Big) \text{ where}$$
$$s_{\mathring{Q}^n}(b_{i,j}) = v_i, \quad s_{\mathring{Q}^n}(e_{i,j}) = v_i, \quad t_{\mathring{Q}^n}(b_{i,j}) = v_j, \quad t_{\mathring{Q}^n}(e_{i,j}) = v_j.$$

We have a projection $\pi : \langle \mathring{Q}^n\rangle \to \langle Q\rangle$, which maps $b_{i,j}$ on b and $e_{i,j}$ on id_v, and a section $\iota : \langle Q\rangle \to \langle \mathring{Q}^n\rangle$ defined by mapping v onto v_0 and b onto $b_{0,n-1} \circ e_{n-1,0}$, where, as usual, we denote in a same way an arrow and the corresponding path of length one.

For A any subset of $A_{\mathring{Q}^n}$, let $m_A \colon \mathring{Q}^n|_A \hookrightarrow \mathring{Q}^n$ be the canonical embedding, and let $r_A \coloneqq m_{A*}(\mathrm{tot}_{\mathring{Q}^n|_A})$. Set $r' \coloneqq (r_A \mid A \in \mathcal{A})$ for $\mathcal{A} \subset \mathcal{P}(A_{\mathring{Q}^n})$ defined as the set containing

- $A_e \coloneqq \{e_{i,j} \mid i,j \in [n]\}$,
- for $i < j$ and $b \in B$,

$$A_{b,i,j} \coloneqq \{\underbrace{e_{0,i}}_{\text{if } i \neq 0}, b_{i,j}, \underbrace{e_{j,n-1}}_{\text{if } j \neq n-1}, b_{0,n-1}\}.$$

We claim that π and ι induce an equivalence of category between $\langle Q\rangle$ and $\langle \mathring{Q}^n\rangle/r'$. From the definition of A_e, for any $i,j,l \in [n]$, we have $e_{i,j} \circ e_{j,l} \sim e_{i,l}$ and $e_{i,i} \sim \mathrm{id}_i$. Now the definition of $A_{b,i,j}$, for $b \in B$ and $0 \leq i < j < n$, induces that $e_{0,i} \circ b_{i,j} \circ e_{j,n-1} \sim b_{0,n-1}$. These relations generate all r', and they become equalities by applying the projection. Hence $\pi_* \colon \langle \mathring{Q}^n\rangle/r' \to \langle Q\rangle$ is well-defined. Clearly $\pi \circ \iota$ is identity. Concerning the other direction, for $b \in B$ and $0 \leq i < j < n$, we have

$$\iota \circ \pi(b_{i,j}) = b_{0,n-1} \circ e_{n-1,0} \sim e_{0,i} \circ b_{i,j} \circ e_{j,n-1} \circ e_{n-1,0} \sim e_{0,i} \circ b_{i,j} \circ e_{j,0}.$$

Hence we get a natural transformation η between the identity functor and $\iota \circ \pi_*$ by setting $\eta_i \coloneqq e_{i,0} \in \mathrm{Hom}_{\langle \mathring{Q}^n\rangle/r'}(i, \iota \circ \pi(i) = v_0)$. Since $e_{i,0}$ is an isomorphism, we conclude that there is an equivalence of category between $\langle Q\rangle$ and $\langle \mathring{Q}^n\rangle/r'$.

Recall that M is the finitely presentable monoid $\langle B \mid r = 1, r \in R\rangle$. Assume that n is greater than the longest word in R. To any word $\rho = b^0 b^1 \cdots b^{l-1}$ in R corresponds a subset

$$A_\rho \coloneqq \{e_{0,l}, b^0_{0,1}, b^1_{1,2}, \ldots, b^{l-1}_{l-1,l}\} \subseteq A_{\mathring{Q}^n}.$$

Let $\mathcal{A}' \coloneqq \mathcal{A} \cup \{A_\rho \mid \rho \in R\}$. Then, $\langle \mathring{Q}^n\rangle/(r_A \mid A \in \mathcal{A}')$ is equivalent as a category to $\langle Q\rangle/(\pi_*(r_{A_\rho}) \mid \rho \in R)$ which is the category of the monoid M. Hence the validity of $\mathrm{Commerge}_{(m_A)_{A\in\mathcal{A}'}}$ is equivalent to the triviality of M. Once again, we conclude using Lemma 30 and the Adyan-Rabin theorem. ◀

▶ **Theorem 34.** *The theory $\mathcal{T}_{\mathrm{cat}}$ is undecidable.*

Proof. Let M, B, R be as in the proof of Theorem 33. Consider the acyclic quiver $Q := (\{v_0, v_1\}, B, s_Q, t_Q)$ where $s_Q(b) = v_0$ and $t_Q(b) = v_1$ for any $b \in B$. To a word $\rho = b^0 b^1 \cdots b^{l-1}$ with $b^0, \ldots, b^{l-1} \in B$, for some $l \in \mathbb{N}$, we associate the predicate of arity Q

$$\mathrm{EqId}_\rho(x) : \quad \exists_{\mathrm{PQ}_l}\, y, \quad \bigwedge_{i \in [l]} \mathrm{restr}_{a_i}(y) \approx \mathrm{restr}_{b^i}(x) \quad \wedge \quad \mathrm{EqPath}_{\bullet, \mathrm{PQ}_l}(\mathrm{restr}_{\frown}(y), y)$$

where, for $i \in [l]$, $a_i \colon \bullet \longrightarrow \bullet \to \mathrm{PQ}_l$ maps the arc of $\bullet \longrightarrow \bullet$ onto the i-th arc of PQ_l and, as usual, for $b \in B$, $b \colon \bullet \longrightarrow \bullet \to Q$ maps the arc of $\bullet \longrightarrow \bullet$ onto the arc $b \in A_Q$. If D is a diagram over Q in some category $\mathcal{C}$, then $\mathrm{EqId}_\rho(D)$ is equivalent to the fact that $D(v_0) = D(v_1)$ and $D(b^{l-1}) \circ \cdots \circ D(b^0) = \mathrm{id}_{D(v_0)}$. In particular, a diagram $D \colon \langle Q \rangle \to \mathcal{C}$ verifies $\bigwedge_{\rho \in R} \mathrm{EqId}_\rho(D)$ if and only if it factorizes through the category associated to the monoid M. Hence, the triviality of the monoid M is equivalent to the validity of the following formula in $\mathcal{T}_{\mathrm{cat}}$:

$$\forall_Q\, x, \quad \bigwedge_{\rho \in R} \mathrm{EqId}_\rho(x) \;\to\; \mathrm{commute}(x).$$

The Adyan-Rabin theorem concludes the proof. ◀

7 Conclusion

We have shown that the many-sorted signature Σ is expressive enough to formulate a theory $\mathcal{T}_{\mathrm{cat}}$ whose models are exactly diagrams in small categories, as well as an extension $\mathcal{T}_{\mathrm{ab}}$ of $\mathcal{T}_{\mathrm{cat}}$ whose models are exactly diagrams in small *abelian* categories. Restricting sorts to acyclic quivers and morphisms to embeddings makes the commerge problem for $\mathcal{T}_{\mathrm{cat}}$ decidable, that is, one can decide when the commutativity of a given diagram follows from that of a given collection of sub-diagrams. Generalizing this study to monoidal categories [19] or more generally to higher category theory does not seem immediate. However, type-theoretic approaches have been successfully applied to devise a syntactic description of weak ω-categories [6] and of opetopes [22], and both these works laid the foundations of prototype proof assistants.

The signatures and theories presented in this article are shaped by their subsequent usage as *interfaces* in a computer-aided tool for diagram chases. This tool eventually produces formal proofs of the corresponding theorems for a given mathematical structure. Interfaces should indeed be convenient enough to fullfil for concrete applications, e.g., abelian groups. This motivation explains some seemingly odd choices, including the use of a commutation predicate instead of the arguably more natural equivalence relation on paths.

A companion file [1] to this submission illustrates how to implement a deep embedding of formulas of Σ using the Coq proof assistant [21]; its content should be easy to transpose to other proof systems. Theorem `duality_theorem_with_theory` expresses a general duality principle. It can be specialized, e.g., to any formalized definition of abelian categories, and the resulting instance of the theorem ensures that a formula of the language is valid for abelian categories if and only if its dual is valid. This companion file however does not provide any specific such formal definition of abelian categories. Theorem 31 results in a complete decision procedure for commutativity clauses. The optimizations that make it work on concrete examples however go beyond the scope of the present article.

Similar concerns about diagrammatic reasoning have motivated the implementation of the accomplished Globular proof assistant [3], for higher-dimensional category theory. Based on higher-dimensional rewriting, it implements various algorithms for constructing and

comparing diagrams in higher categories. It is geared towards visualization rather than formal verification. The closest related work we are aware of seem unpublished at the time of writing. Lafont's categorical diagram editor [10], based on the Unimath library [23] and Barras and Chabassier's graphical interface for diagrammatic proofs [4] both provide a graphical interface for generating Coq proof scripts and visualizing Coq goals as diagrams. Himmel [8] describes a formalization of abelian categories in Lean [5], including proofs of the five lemma and of the snake lemma, and proof (semi-)automation tied to this specific formalization. Duality arguments are not addressed. Monbru [15] also discusses automation issues in diagram chases, and provides heuristics for generating them automatically, albeit expressed in a pseudo-language.

References

1 http://matthieu.piquerez.fr/partage/FANL_duality.v.

2 Sergei Ivanovich Adyan. Algorithmic undecidability of problems of recognition of certain properties of groups. *Dokl. Akad. Nauk SSSR*, 103:533–535, 1955.

3 Krzysztof Bar, Aleks Kissinger, and Jamie Vicary. Globular: an online proof assistant for higher-dimensional rewriting. *Log. Methods Comput. Sci.*, 14(1), 2018. doi:10.23638/LMCS-14(1:8)2018.

4 Luc Chabassier and Bruno Barras. A graphical interface for diagrammatic proofs in proof assistants. Contributed talks in the 29th International Conference on Types for Proofs and Programs (TYPES 2023), 2023. URL: https://types2023.webs.upv.es/TYPES2023.pdf.

5 Leonardo Mendonça de Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von Raumer. The lean theorem prover (system description). In Amy P. Felty and Aart Middeldorp, editors, *Automated Deduction - CADE-25 - 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings*, volume 9195 of *Lecture Notes in Computer Science*, pages 378–388. Springer, 2015. doi:10.1007/978-3-319-21401-6_26.

6 Eric Finster and Samuel Mimram. A type-theoretical definition of weak ω-categories. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–12. IEEE Computer Society, 2017. doi:10.1109/LICS.2017.8005124.

7 Peter Freyd. *Abelian categories. An introduction to the theory of functors.* Harper's Series in Modern Mathematics. Harper & Row Publishers, New York, 1964.

8 Markus Himmel. Diagram chasing in interactive theorem proving. Bachelorarbeit. Karlsruher Institut für Technologie, 2020. URL: https://pp.ipd.kit.edu/uploads/publikationen/himmel20bachelorarbeit.pdf.

9 Wilfrid Hodges. *A shorter model theory.* Cambridge: Cambridge University Press, 1997.

10 Ambroise Lafont. A categorical diagram editor to help formalising commutation proofs. https://amblafont.github.io/graph-editor/index.html.

11 F. William Lawvere and Stephen H. Schanuel. *Conceptual mathematics. A first introduction to categories.* Cambridge: Cambridge University Press, 2nd ed. edition, 2009.

12 Saunders Mac Lane. *Homology.* Class. Math. Berlin: Springer-Verlag, reprint of the 3rd corr. print. 1975 edition, 1995.

13 Saunders Mac Lane. *Categories for the working mathematician*, volume 5 of *Grad. Texts Math.* New York, NY: Springer, 2nd ed edition, 1998.

14 J. Peter May. *A concise course in algebraic topology.* Chicago, IL: University of Chicago Press, 1999.

15 Yannis Monbru. Towards automatic diagram chasing. M1 report. École Normale Supérieure Paris-Saclay, 2022. URL: https://github.com/ymonbru/Diagram-chasing/blob/main/MONBRU_Yannis_Rapport.pdf.

16 Matthieu Piquerez. *Tropical Hodge theory and applications.* PhD thesis, Institut Polytechnique de Paris, November 2021. URL: https://theses.hal.science/tel-03499730#.

17 Michael O. Rabin. Recursive unsolvability of group theoretic problems. *Ann. Math. (2)*, 67:172–194, 1958. doi:10.2307/1969933.

18 Emily Riehl. *Category Theory in Context*. Dover Publications, 2017. URL: https://math.jhu.edu/~eriehl/context.pdf.

19 Peter Selinger. A survey of graphical languages for monoidal categories. In Bob Coecke, editor, *New Structures for Physics*, volume 813 of *Lecture Notes in Physics*, pages 289–355. Springer, 2011. doi:10.1007/978-3-642-12821-9_4.

20 Alfred Tarski. The semantic conception of truth and the foundations of semantics. Philos. Phenomenol. Res. 4, 341-376 (1944)., 1944.

21 The Coq Development Team. The coq proof assistant, June 2023. doi:10.5281/zenodo.8161141.

22 Cédric Ho Thanh, Pierre-Louis Curien, and Samuel Mimram. A sequent calculus for opetopes. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*, pages 1–12. IEEE, 2019. doi:10.1109/LICS.2019.8785667.

23 Vladimir Voevodsky. Univalent semantics of constructive type theories. In Jean-Pierre Jouannaud and Zhong Shao, editors, *Certified Programs and Proofs - First International Conference, CPP 2011, Kenting, Taiwan, December 7-9, 2011. Proceedings*, volume 7086 of *Lecture Notes in Computer Science*, page 70. Springer, 2011. doi:10.1007/978-3-642-25379-9_7.

24 Douglas B. West. *Introduction to graph theory*. New Delhi: Prentice-Hall of India, 2nd ed. edition, 2005.

What Monads Can and Cannot Do with a Bit of Extra Time

Rasmus Ejlers Møgelberg
IT University of Copenhagen, Denmark

Maaike Annebet Zwart
IT University of Copenhagen, Denmark

Abstract

The delay monad provides a way to introduce general recursion in type theory. To write programs that use a wide range of computational effects directly in type theory, we need to combine the delay monad with the monads of these effects. Here we present a first systematic study of such combinations.

We study both the coinductive delay monad and its guarded recursive cousin, giving concrete examples of combining these with well-known computational effects. We also provide general theorems stating which algebraic effects distribute over the delay monad, and which do not. Lastly, we salvage some of the impossible cases by considering distributive laws up to weak bisimilarity.

2012 ACM Subject Classification Theory of computation → Program semantics; Theory of computation → Type structures

Keywords and phrases Delay Monad, Monad Compositions, Distributive Laws, Guarded Recursion, Type Theory

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.39

Related Version An appendix containing the full proofs is available at arXiv.
Extended Version: `http://arxiv.org/abs/2311.15919`

Supplementary Material *Software (Source Code)*: `https://github.com/maaikezwart/Agda-proofs/tree/main/What%20monads%20can%20and%20cannot%20do`
archived at `swh:1:dir:e29201a731f2bffb8137ffa7b0eb09fb388d62d1`

Funding Independent Research Fund Denmark, grant number 2032-00134B.

1 Introduction

Martin Löf type theory [29] is a language that can be understood both as a logic and a programming language. For the logical interpretation it is crucial that all programs terminate. Still, one would like to reason about programming languages with general recursion, or even write general recursive programs inside type theory. One solution to this problem is to encapsulate recursion in a monad, such as the delay monad D. This monad maps an object X to the coinductive solution to $DX \cong X + DX$. The right inclusion into the sum of the above isomorphism introduces a computation step, and infinitely many steps correspond to divergence. Capretta [8] showed how D introduces general recursion via an iteration operator of type $(X \to D(X + Y)) \to X \to DY$. For this reason, D has been used to model recursion in type theory [14, 41, 4], and in particular forms part of the basis of interaction trees [42].

The delay monad has a guarded recursive variant defined using Nakano's [33] fixed point modality $\triangleright$. Data of type $\triangleright X$ should be thought of as data of type X available only one time step from now. This modal operator has a unit $\mathsf{next} : X \to \triangleright X$ transporting data to the future, and a fixed point operator $\mathsf{fix} : (\triangleright X \to X) \to X$ mapping productive definitions to their fixed points satisfying $\mathsf{fix}(f) = f(\mathsf{next}(\mathsf{fix}(f)))$. Guarded recursion can be modelled in the topos of trees [7] – the category $\mathsf{Set}^{\omega^{\mathrm{op}}}$ of presheaves on the ordered natural

numbers – by defining $(\triangleright X)(0) = 1$ and $(\triangleright X)(n+1) = X(n)$. The guarded delay monad D^{g} is defined as the free monad on $\triangleright$, i.e., the inductive (and provably also coinductive) solution to $D^{\mathsf{g}}X \cong X + \triangleright(D^{\mathsf{g}}X)$.

The two delay monads can be formally related by moving to a multiclock variant of guarded recursion, in which the modal operator $\triangleright$ is indexed by a clock variable κ, which can be universally quantified. Then by defining the guarded delay monad $D^\kappa X$ to be the unique solution to $D^\kappa X \cong X + \triangleright^\kappa(D^\kappa X)$, the coinductive variant can be encoded [2] as $DX \stackrel{\mathsf{def}}{=} \forall\kappa.D^\kappa X$. In this paper we will work informally in Clocked Cubical Type Theory (CCTT) [5], a type theory in which such encodings of coinductive types can be formalised and proven correct.

Unlike the coinductive variant, the guarded delay monad has a fixed point operator of type $((X \to D^\kappa Y) \to (X \to D^\kappa Y)) \to (X \to D^\kappa Y)$, defined using fix. For the coinductive delay monad, fixed points only exist for continuous maps, but in the guarded case, continuity is a consequence of a causality property enforced in types using $\triangleright$. As a consequence, higher order functional programming languages with recursion can be embedded in type theories like CCTT by interpreting function spaces as Kleisli exponentials for D^κ. For example, Paviotti et al. [36] showed how to model the simply typed lambda calculus with fixed point terms (PCF), and proved adequacy of the model, all in a type theory with guarded recursion. These results have since been extended to languages with recursive types [31] and (using an impredicative universe) languages with higher-order store [38]. This suggests that the guarded delay monad can be used for programming and reasoning about programs using a wide range of advanced computational effects directly in type theory. However, a mathematical theory describing the interaction of the delay monads with other monads is still lacking, even for basic computational effects.

1.1 Combining the Delay Monad With Other Effects

In this paper, we present a first systematic study of monads combining delay with other effects. We first show how to combine the delay monad with standard monads known from computational effects: exceptions, reader, global state and the selection monad. Most of these follow standard combinations of effects and non-termination known from domain theory, but, the algebraic status of these combinations is simpler than in the domain theoretic case: Whereas the latter can be understood as free monads for order-enriched algebraic theories [20], the combinations with the guarded recursive delay monad are simply free models of theories in the standard sense, with the caveat that the arity of the step operation is non-standard.

The rest of the paper is concerned with distributive laws of the form $TD \to DT$, where T is any monad and D is the delay monad in any of the two forms mentioned above. Such a distributive law distributes the operations of T over steps, and equips the composite DT with a monad structure that describes computations whose other effects are only visible upon termination. This is the natural monad for example in the case of writing to state, when considering non-determinism and observing must-termination, or for computing data contained in data structures such as trees or lists.

There are two natural ways of distributing an n-ary operation op over computation steps: The first is to execute each of the n input computations in sequence until they have all terminated, the second is to execute the n inputs in parallel, delaying terminated computations until all inputs have terminated. We show that sequential execution yields a distributive law for algebraic monads (monads generated by algebraic theories) where all equations are balanced, i.e., the number of occurrences of each variable on either side is the same. Trees, lists, and multisets are examples of such monads.

The requirement of balanced equations is indeed necessary. This was observed already by Møgelberg and Vezzosi [32] who showed that for the finite powerset monad, sequential distribution of steps over the union operator was not well defined, and parallel distribution did not yield a distributive law due to miscounting of steps. Here we strengthen this result to show that no distributive law is possible for the finite powerset monad over the coinductive delay monad. At first sight it might seem that the culprit in this case is the idempotency axiom. However, we show that in some cases it is possible to distribute idempotent operations over the coinductive delay monad, but not over the guarded one (we show this just for commutative operations).

Finally, we show that if one is willing to work up to weak bisimilarity, i.e., equating elements of the delay monad that only differ by a finite number of computation steps, then one can construct a distributive law $TD \to DT$ for any monad T generated by an algebraic theory with no drop equations (equations where a variable appears on one side, but not the other). To make this precise, we formulate this result as a distributive law of monads on a category of setoids.

Agda Formalisation

Some of the results presented in this paper have been formalised in Agda using Vezzosi's Guarded Cubical library[1]. The code can be found at `https://tinyurl.com/WMCDAgda`.

2 Monads and Algebraic Theories

In this background section we briefly remind the reader of algebraic theories and free model monads for an algebraic theory. We mention different classes of equations that play a role in our analysis of monad compositions, and we discuss distributive laws for composing monads.

▶ **Definition 1** (Algebraic Theory). *An algebraic theory A consists of a signature Σ_A and a set of equations E_A. The signature is a set of operation symbols with arities given by natural numbers. The signature Σ_A together with a set of variables X inductively defines the set of A-terms: Every variable $x : X$ is a term, and for each operation symbol* op *in Σ_A, if* op *has arity n and $t_1, \dots, t_n$ are terms, then $\mathrm{op}(t_1, \dots, t_n)$ is a term. The set of equations contains pairs of terms (s, t) in a finite variable context which are to be considered equal, often written as $s = t$. These pairs then define an equivalence relation on terms via equational logic.*

▶ **Example 2** (Monoids). The algebraic theory of monoids has a signature consisting of a constant c and a binary operation $*$, satisfying the left and right unital equations: $\forall x : X. c * x = x$ and $\forall x : X. x * c = x$, and associativity: $\forall x, y, z : X. (x * y) * z = x * (y * z)$. Commutative monoids also include the commutativity equation: $\forall x, y : X. x * y = y * x$.

▶ **Definition 3** (Category of Models). *A model of an algebraic theory (Σ_A, E_A) is a set X together with an interpretation $\mathrm{op}_X : X^n \to X$ of each n-ary operation* op *in Σ_A, such that $s_X = t_X$ for each equation $s = t$ in E_A. Here s_X is the interpretation of s defined inductively using the interpretation of operations.*

A homomorphism between two models $(X, (-)_X)$ and $(Y, (-)_Y)$ is a morphism $h : X \to Y$ such that $h(\mathrm{op}_X(x_1, \dots, x_n)) = \mathrm{op}_Y(h(x_1), \dots, h(x_n))$ for each n-ary operation op *in Σ_A and $x_1, \dots, x_n : X$. The models of an algebraic theory and homomorphisms between them form a category called the* category of algebras *of the algebraic theory, denoted A-alg.*

[1] `https://github.com/agda/guarded`

The free model of an algebraic theory A with variables in a set X consists of the set of equivalence classes of A-terms in context X. The functor $F : \mathbf{Set} \to A\text{-alg}$ sending each set X to the free model of A on X is left adjoint to the forgetful functor sending each A-model to its underlying set. This adjunction induces a monad on **Set**, called the *free model monad* of the algebraic theory [24, 23, 25]. The category of algebras of A is isomorphic to the Eilenberg-Moore category of this monad. If a monad T is isomorphic to the free model monad of an algebraic theory, then we say that T *is presented* by that algebraic theory.

▶ **Example 4** (Boom Hierarchy Monads [30])**.** The binary tree monad, the list monad, the multiset monad and the powerset monad are the free model monads of respectively the theories of:

- Magmas, the theory consisting of a constant and a binary operation satisfying the left and right unit equations.
- Monoids, which are magmas satisfying the associativity equation.
- Commutative monoids.
- Idempotent commutative monoids, which are also known as semilattices.

The equations of an algebraic theory determine much of the behaviour of its free model monad. For example, linear equations result in monads that always compose with commutative monads [26, 34]. In this paper, we built upon the ideas of Gautam [16] and Parlant et al [35], and distinguish the following classes of equations:

▶ **Definition 5.** *Write* $\mathrm{var}(t)$ *for the set of variables that appear in a term* t*. We say that an equation* $s = t$ *is:*

Linear *if* $\mathrm{var}(s) = \mathrm{var}(t)$ *and each variable in these sets appears exactly once in both* s *and* t*. Example:* $x * y = y * x$*.*

Balanced *if* $\mathrm{var}(s) = \mathrm{var}(t)$ *and each variable in these sets appears equally many times in* s *and* t*. Example:* $(x * y) * (y * z) = (y * y) * (x * z)$*.*

Dup *if there is an* $x : \mathrm{var}(s) \cup \mathrm{var}(t)$*, such that* x *appears* ≥ 2 *times in* s *and/or* t*. Example: the balanced equation above, as well as* $x * x = x * x * x$ *and* $x \wedge (x \vee y) = x$*.*

Drop *if* $\mathrm{var}(s) \neq \mathrm{var}(t)$*. Example:* $x \wedge (x \vee y) = x$*.*

▶ Remark 6. Notice that these types of equations are not mutually exclusive. An equation can for instance be both dup and drop, such as the absorption equation $x \wedge (x \vee y) = x$.

2.1 Distributive Laws

One way of composing two monads is via a *distributive law* describing the interaction between the two monads [6].

▶ **Definition 7** (Distributive Law)**.** *Given monads* $\langle S, \eta^S, \mu^S \rangle$ *and* $\langle T, \eta^T, \mu^T \rangle$*, a distributive law distributing* S *over* T *is a natural transformation* $\zeta : ST \to TS$ *satisfying the following axioms:*

$$\zeta \circ \eta^S T = T\eta^S \qquad \zeta \circ S\eta^T = \eta^T S \qquad \text{(unit axioms)}$$
$$\zeta \circ \mu^S T = T\mu^S \circ \zeta S \circ S\zeta \qquad \zeta \circ S\mu^T = \mu^T S \circ T\zeta \circ \zeta T \qquad \text{(multiplication axioms)}$$

▶ **Example 8** (Lists and Multisets)**.** Distributive laws are named after the well-known distributivity of multiplication over addition: $a * (b + c) = (a * b) + (a * c)$. Many distributive laws follow the same distribution pattern. For example, the list monad distributes over the multiset monad in this way: $\zeta[\wr a \wr, \wr b, c \wr] = \wr [a, b], [a, c] \wr$. However, this is by no means the only way a distributive law can function.

▶ **Theorem 9** (Beck [6]). *Let $\mathcal{C}$ be a category, and $\langle S, \eta^S, \mu^S\rangle$ and $\langle T, \eta^T, \mu^T\rangle$ two monads on $\mathcal{C}$. If $\zeta : ST \to TS$ is a distributive law, then the functor TS carries a monad structure with unit $\eta^T\eta^S$ and multiplication $\mu^T\mu^S \circ T\zeta S$.*

We frequently use the following equivalence in our proofs:

▶ **Theorem 10** (Beck [6]). *Given two monads $\langle S, \eta^S, \mu^S\rangle$ and $\langle T, \eta^T, \mu^T\rangle$ on a category $\mathcal{C}$, there is a bijective correspondence between distributive laws of type $ST \to TS$, and liftings of T to the Eilenberg-Moore category $\mathcal{C}^S$ of S.*

Here, a lifting of T to $\mathcal{C}^S$ is an assignment mapping S-algebra structures on a set X to S-algebra structures on TX such that η^T and μ^T are S-algebra homomorphisms.

3 Guarded Recursion and the Delay Monad

In this paper we work informally in Clocked Cubical Type Theory (CCTT) [5]. At present, this is the only known theory combining the features we need: Multiclocked guarded recursion and quotient types (to express free monads). Here we remind the reader of the basic principles of CCTT, but we refer to Kristensen et al. [5] for the full details, including a denotational semantics for CCTT.

3.1 Algebraic Theories in Cubical Type Theory

CCTT is an extension of Cubical Type Theory (CTT) [12], which in turn is a version of Homotopy Type Theory (HoTT) [39] that gives computational content to the univalence axiom. In CTT, the identity type of Martin-Löf type theory is replaced by a path type, which we shall write infix as $t =_A u$, often omitting the type A of t and u. We will work informally with $=$, using its standard properties such as function extensionality.

A type A is a *homotopy proposition* (or hprop) in HoTT and CTT, if any two elements of A are equal, and an *hset* if $x =_A y$ is an hprop for all $x, y : A$. Assuming a universe of small types, one can encode universes hProp and hSet of homotopy sets and propositions in the standard way. The benefit of working with hsets is that there is no higher structure to consider. In particular, the collection of hsets and maps between these forms a category in the sense of HoTT [39], and so basic category theoretic notions such as functors and monads on hsets can be formulated in the standard way.

The notion of algebraic theory can also be read directly in CTT this way. Moreover, the free monads on algebraic theories can be defined using higher inductive types (HITs). These are types given inductively by constructors for terms as well as for equalities. For example, the format for HITs used in CCTT [5] (adapted from Cavallo and Harper [9]) is expressive enough[2].

We write type equivalence as $A \simeq B$. For hsets this just means that there are maps $f : A \to B$ and $g : B \to A$ that are inverses of each other, as expressed in CTT using path equality.

3.2 Multi-Clocked Guarded Recursion

CCTT extends CTT with multi-clock guarded recursion. The central component in this is a modal type operator $\triangleright$ indexed by clocks κ, used to classify data that is delayed by one time step on clock κ. The most important typing rules of CCTT are collected in Figure 1.

[2] Full details are in the appendix in the extended version on ArXiv

$$\frac{\Gamma \vdash}{\Gamma, \kappa : \mathsf{clock} \vdash} \qquad \frac{\kappa : \mathsf{clock} \in \Gamma}{\Gamma, \alpha : \kappa \vdash} \qquad \frac{\Gamma, \mathsf{TimeLess}(\Gamma') \vdash t : \triangleright\,(\alpha : \kappa).A \qquad \Gamma, \beta : \kappa, \Gamma' \vdash}{\Gamma, \beta : \kappa, \Gamma' \vdash t\,[\beta] : A\,[\beta/\alpha]}$$

$$\frac{\Gamma, \alpha : \kappa \vdash t : A}{\Gamma \vdash \lambda(\alpha : \kappa).t : \triangleright\,(\alpha : \kappa).A} \qquad \frac{\Gamma, \kappa : \mathsf{clock} \vdash t : A}{\Gamma \vdash \Lambda\kappa.t : \forall\kappa.A} \qquad \frac{\Gamma \vdash t : \forall\kappa.A \qquad \Gamma \vdash \kappa' : \mathsf{clock}}{\Gamma \vdash t[\kappa'] : A[\kappa'/\kappa]}$$

$$\frac{\Gamma \vdash t : \triangleright^\kappa A \to A}{\Gamma \vdash \mathsf{dfix}^\kappa\, t : \triangleright^\kappa A} \qquad \frac{\Gamma \vdash t : \triangleright^\kappa A \to A}{\Gamma \vdash \mathsf{pfix}^\kappa\, t : \triangleright\,(\alpha : \kappa).(\mathsf{dfix}^\kappa t)\,[\alpha] =_A t(\mathsf{dfix}^\kappa t)}$$

Figure 1 Selected typing rules for Clocked Cubical Type Theory [5]. The telescope $\mathsf{TimeLess}(\Gamma')$ is composed of the timeless assumptions in Γ, i.e. interval variables and faces (as in Cubical Type Theory) as well as clock variables.

Clocks are introduced as special assumptions $\kappa : \mathsf{clock}$ in a context, and can be abstracted and applied to terms of the type $\forall\kappa.A$ which behaves much like a Π-type for clocks. Like function extensionality, extensionality for $\forall\kappa.A$ also holds in CCTT.

The rules for $\triangleright$ also resemble those of Π-types: Introduction is by abstracting special assumptions $\alpha : \kappa$ called *ticks* on the clock κ. Since ticks can appear in terms, the modal type $\triangleright\,(\alpha : \kappa).A$ binds α in A, just like a Π-type binds a variable. We write $\triangleright^\kappa A$ for $\triangleright\,(\alpha : \kappa).A$ when α does not appear in A. The introduction rule for $\triangleright$ can be read as stating that if t has type A after the tick α, then $\lambda(\alpha : \kappa).t$ has type $\triangleright\,(\alpha : \kappa).A$ now.

The modality $\triangleright$ is eliminated by applying a term to a tick. Note that the term t applied to the tick β cannot already contain β freely. This restriction prevents t from being applied twice to the same tick, which would construct terms of type $\triangleright^\kappa \triangleright^\kappa A \to \triangleright^\kappa A$, collapsing two steps into one. Moreover, t cannot contain any variables nor other ticks occurring in the context after β, only *timeless* assumptions, i.e., clocks, interval assumptions and faces. One application of timeless assumptions is to type the extensionality principle for $\triangleright$:

$$(t =_{\triangleright\,(\alpha:\kappa).A} u) \simeq \triangleright\,(\alpha : \kappa).(t\,[\alpha] =_A u\,[\alpha]). \tag{1}$$

For all explicit applications of terms to ticks in this paper, the term will not use timeless assumptions. The usual η and β laws hold for tick abstraction and application.

The use of ticks for programming with $\triangleright$ implies that $\triangleright$ is an applicative functor, and can even be given a dependent applicative action of type

$$\Pi(f : \triangleright^\kappa(\Pi(X : A).B(x)).\Pi(y : \triangleright^\kappa A).\triangleright\,(\alpha : \kappa).B(y\,[\alpha]).$$

Ticks are named in CCTT for reasons of normalisation [3], but are essentially identical. This is expressed in type theory as the *tick irrelevance principle*:

$$\mathsf{tirr}^\kappa : \Pi(x : \triangleright^\kappa A).\triangleright\,(\alpha : \kappa).\triangleright\,(\beta : \kappa).(x\,[\alpha] =_A x\,[\beta]). \tag{2}$$

The term tirr^κ is defined in CCTT using special combinators on ticks, allowing for computational content to tirr^κ. This means that the rule for tick application is more general than the one given in Figure 1. However, we will not need this further generality for anything apart from tirr^κ, which we use directly.

Finally, CCTT has a fixed point operator dfix which unfolds up to path equality as witnessed by pfix. Using these, one can define $\mathsf{fix}^\kappa : (\triangleright^\kappa A \to A) \to A$ as $\mathsf{fix}^\kappa(t) = t(\mathsf{dfix}^\kappa t)$ and prove $\mathsf{fix}^\kappa(t) = t(\mathrm{next}^\kappa(\mathsf{fix}^\kappa(t)))$ where $\mathrm{next}^\kappa \stackrel{\mathsf{def}}{=} (\lambda(x : A).\lambda(\alpha : \kappa).x) : A \to \triangleright^\kappa A$. Note that this uses that also variables appearing before a tick in a context can be introduced. This is not the case in all Fitch-style modal type theories [11, 17].

3.3 Guarded Recursive Types

A guarded recursive type is a recursive type in which the recursive occurrences of the type are all guarded by a $\triangleright$. These can be encoded up to equivalence of types using fixed points of maps on the universe. Our primary example is the guarded recursive delay monad D^κ defined to map an X to the recursive type

$$D^\kappa X \simeq X + \triangleright^\kappa(D^\kappa X).$$

We write $\mathrm{now} : X \to D^\kappa X$ and $\mathrm{step} : \triangleright^\kappa(D^\kappa X) \to D^\kappa X$ for the two maps given by inclusion and the equivalence above.

Since $\triangleright^\kappa$ preserves the property of being an hset, one can prove by guarded recursion that $D^\kappa X$ is an hset whenever X is. D^κ can be seen as a free construction in the following sense.

▶ **Definition 11.** *A* delay algebra *on the clock κ is an hset X together with a map $\triangleright^\kappa X \to X$.*

Given an hset X, the hset $D^\kappa X$ carries a delay algebra structure. It is the free delay algebra in the sense that given any other delay algebra (Y, ξ), and a map $f : X \to Y$, there is a unique homomorphism $\overline{f} : D^\kappa X \to Y$ extending f along now, defined by the clause

$$\overline{f}(\mathrm{step}(x)) = \xi(\lambda(\alpha : \kappa).\overline{f}(x\,[\alpha])). \tag{3}$$

This is a recursive definition that can be encoded as a fixed point of a map $h : \triangleright^\kappa(D^\kappa X \to Y) \to (D^\kappa X \to Y)$ defined using the clause $h(g)(\mathrm{step}(x)) = \xi(\lambda(\alpha : \kappa).(g\,[\alpha])(x\,[\alpha]))$. In this paper we use the simpler notation of Equation (3) for such definitions rather than the explicit use of fix^κ.

We sketch the proof that $\overline{f}$ is the unique homomorphism extending f, to illustrate the use of fix^κ for proofs. Suppose g is another such extension. To use guarded recursion, assume that $\triangleright^\kappa(g = \overline{f})$. We show that $g(\mathrm{step}(x)) = \xi(\lambda(\alpha : \kappa).\overline{f}(x\,[\alpha]))$. Since g is a homomorphism: $g(\mathrm{step}(x)) = \xi(\lambda(\alpha : \kappa).g(x\,[\alpha]))$. So by extensionality for $\triangleright$ (1) it suffices to show that $\triangleright\,(\alpha : \kappa).(g(x\,[\alpha]) = \overline{f}(x\,[\alpha]))$, which follows from the guarded recursion hypothesis.

Tick irrelevance implies that D^κ is a commutative monad in the sense of Kock [22].

3.4 Encoding Coinductive Types

Coinductive types can be encoded using a combination of guarded recursive types and quantification over clocks. This was first observed by Atkey and McBride [2]. We recall the following special case of a more general theorem for this in CCTT [5]. First a definition.

▶ **Definition 12.** *A functor $F : \mathsf{hSet} \to \mathsf{hSet}$* commutes with clock quantification*, if the canonical map $F(\forall\kappa.(X\,[\kappa])) \to \forall\kappa.F(X\,[\kappa])$ is an equivalence for all $X : \forall\kappa.\mathsf{hSet}$. An hset X is* clock irrelevant *if the constant functor to X commutes with clock quantification, i.e. if the canonical map $X \to \forall\kappa.X$ is an equivalence.*

Note that functors commuting with clock quantification map clock irrelevant types to clock irrelevant types.

▶ **Theorem 13** ([5])**.** *Let F be an endofunctor on the category of hsets commuting with clock quantification, and let $\nu^\kappa F$ be the guarded recursive type satisfying $F(\triangleright^\kappa(\nu^\kappa F)) \simeq \nu^\kappa F$, then $\nu F \stackrel{\mathrm{def}}{=} \forall\kappa.\nu^\kappa F$ carries a final coalgebra structure for F.*

In order to apply Theorem 13, of course, one needs a large collection of functors F commuting with clock quantification. Fortunately, the collection of such functors is closed under almost all type constructors, including finite sum and product, Π and Σ types, $\triangleright$, $\forall\kappa$,

and guarded recursive types [5, Lemma 4.2]. Clock irrelevant types are likewise closed under the same type constructors, and path equality. The only exception to clock irrelevance is the universe type.

For example, if X is clock irrelevant, then $F(Y) = X + Y$ commutes with clock quantification, and so $DX \stackrel{\text{def}}{=} \forall\kappa.D^\kappa X$ is the coinductive solution to $DX \simeq X + DX$.

CCTT moreover has a principle of *induction under clocks* allowing one to prove that many HITs are clock irrelevant, including the empty type, booleans and natural numbers. Moreover, one can prove the following.

▶ **Proposition 14.** *Let $A = (\Sigma_A, E_A)$ be an equational theory such that Σ_A and E_A are clock irrelevant. Then the free model monad T commutes with clock quantification. In particular, $T(X)$ is clock irrelevant for all clock irrelevant X.*

The collection of clock irrelevant propositions can be shown to be closed under standard logical connectives. Alternatively, one can assume a global clock constant κ_0, which then can be used to prove that all propositions are clock irrelevant.

▶ **Convention 15.** *In the remainder of this paper, the word* set *will refer to a clock-irrelevant hset, and the word* proposition *will refer to clock-irrelevant homotopy propositions. We will write* Set *and* Prop *for the universes of these. Similarly, whenever we mention functors these are assumed to commute with clock quantification.*

4 Specific Combinations with Delay

In this section we look at some specific examples of monads, and see how they combine with the delay monads. In particular, we will look at the exception, reader, state, and selection monads. Intuitively, these monads model (parts of) the process: read input - compute - do something with the output. For instance, the state monad reads a state, then both updates the current state and gives an output. Combining the state monad with the delay monads allows us to model the fact that the computation in between reading the input and giving the output takes time, and might not terminate.

The examples we give follow the same pattern as the adaptation of these monads to domain theory: we insert a delay monad where one would use lifting in the domain theoretic case. However, we also show that the algebraic status of these monads is much simpler in the guarded recursive case than in the domain theoretic one: they can simply be understood as being generated by algebraic theories where one operation (step) has a non-standard arity. In the domain theoretic case, the algebraic description is in terms of enriched Lawvere theories [20]. We give no algebraic description of the combinations with the coinductive delay monads, because this does not by itself have an algebraic description.

First note that for combinations with delay via a distributive law, it is enough to find a distributive law for the guarded recursive version D^κ.

▶ **Lemma 16.** *Let T be a monad. A distributive law $\zeta_X : \forall\kappa.T(D^\kappa(X)) \to D^\kappa(T(X))$ for the guarded delay monad induces a distributive law $TD \to DT$ for the coinductive delay monad. Similarly, if T^κ is a family of monads indexed by κ then $T(X) = \forall\kappa.T^\kappa(X)$ carries a monad structure.*

Proof. The distributive law can be constructed as the composite

$$T(\forall\kappa.D^\kappa(X)) \to (\forall\kappa.T(D^\kappa(X))) \to \forall\kappa.D^\kappa(T(X)),$$

where κ is fresh for T and X. For the second statement define the multiplication as the composite $\forall\kappa.T^\kappa(\forall\kappa.T^\kappa(X)) \to (\forall\kappa.T^\kappa(T^\kappa(X))) \to \forall\kappa.T^\kappa(X)$. ◀

Exceptions

The first monad we consider is the exception monad. For a set of exceptions E, the exception monad is given by the functor $(- + E)$, with obvious unit and multiplication. The exception monad is the free model monad of the algebraic theory consisting of a signature with a constant e for each exception in E, and no equations.

It is well known that the exception monad distributes over any monad, and therefore we have a distributive law $\zeta : (D^\kappa(-) + E) \to D^\kappa(- + E)$. The resulting composite monad $D^\kappa(- + E)$ is the free model monad of the theory consisting of constants $e : E$ and a step-operator forming a delay algebra, with no additional equations.

Reading

The reader monad $(-)^R$ is presented by the algebraic theory consisting of a single operation $\mathrm{lookup} : X^R \to X$, satisfying the equations

$$\forall x : X.\ \mathrm{lookup}(\lambda r.x) = x \qquad \forall g : (X^R)^R.\ \mathrm{lookup}(\mathrm{lookup} \circ g) = \mathrm{lookup}(\lambda s.g\, s\, s).$$

To combine the reader monad with the delay monad, we define a distributive law $D^\kappa R \to RD^\kappa$ by the clauses $\zeta(\mathrm{now}\, f) = \lambda r.\, \mathrm{now}(fr)$ and $\zeta(\mathrm{step}\, d) = \lambda r.\, \mathrm{step}(\lambda(\alpha : \kappa).(\zeta(d\,[\alpha]))r)$, where $f : X^R$ and $d : \triangleright^\kappa(X^R)$. The resulting composite monad RD^κ is the free model monad of the theory consisting of lookup and step satisfying the above equations for lookup and

$$\forall d : \triangleright^\kappa(X^R).\ \mathrm{step}(\lambda(\alpha : \kappa).\, \mathrm{lookup}(d\,[\alpha])) = \mathrm{lookup}(\lambda r.\, \mathrm{step}(\lambda(\alpha : \kappa).d\,[\alpha]\, r)). \tag{4}$$

Global State

Plotkin and Power [37] show that the global state monad $(S \times -)^S$ can be described algebraically by two operations: $\mathrm{lookup} : X^S \to X$ and $\mathrm{update} : X \to X^S$, satisfying four interaction diagrams. They call the category of such algebras *GS-algebras*.

The natural combination of global state and D^κ is $(D^\kappa(S \times -))^S$ describing computations whose steps occur between reading the initial state and writing back the updated state. To describe this monad algebraically define a GSD-algebra to be a GS-algebra which also carries a delay algebra structure satisfying (4) and

$$\forall x : \triangleright^\kappa X.\, \lambda s.\, \mathrm{update}(\mathrm{step}\, x)s = \lambda s.\, \mathrm{step}(\lambda(\alpha : \kappa).\, \mathrm{update}(x\,[\alpha])s).$$

Diagrammatically:

$$\begin{array}{ccccc}
\triangleright^\kappa(X^S) & \longrightarrow & (\triangleright^\kappa X)^S & \xrightarrow{\mathrm{step}^S} & X^S \\
\downarrow{\scriptstyle \triangleright^\kappa(\mathrm{lookup})} & & & & \downarrow{\scriptstyle \mathrm{lookup}} \\
\triangleright^\kappa X & & \xrightarrow{\qquad\mathrm{step}\qquad} & & X
\end{array}
\qquad\qquad
\begin{array}{ccccc}
\triangleright^\kappa X & \xrightarrow{\triangleright^\kappa(\mathrm{update})} & \triangleright^\kappa(X^S) & \longrightarrow & (\triangleright^\kappa X)^S \\
\downarrow{\scriptstyle \mathrm{step}} & & & & \downarrow{\scriptstyle \mathrm{step}^S} \\
X & & \xrightarrow{\qquad\mathrm{update}\qquad} & & X^S
\end{array}$$

▶ **Theorem 17.** *The monad $(D^\kappa(S \times -))^S$ is the free model monad of the theory of GSD-algebras.*

Note that also $(D(S \times -))^S$ is a monad by Lemma 16, since the assumption of S being clock irrelevant implies $(D(S \times -))^S \simeq \forall\kappa.((D^\kappa(S \times -))^S)$.

Selecting

The selection monad $\mathcal{J}X = (X \to S) \to X$ takes a function $X \to S$, and selects an input $x : X$ to return [15]. This could, for example, be an input for which the function attains an optimal value. It is a monad similar to the reader monad, with a more advanced input. It is also a close companion to the continuation monad $(X \to S) \to S$, and it has many applications in for example game theory and functional programming [18].

The selection monad combines with the delay monad via a distributive law of type $D^\kappa \mathcal{J} \to \mathcal{J} D^\kappa$. Intuitively, it first gathers all the data from the function $X \to S$, and then computes which element from X to select. This computation takes time and might not terminate. We assume that the initial input is readily available, even though the resulting type of the monad composition is $(D^\kappa X \to S) \to D^\kappa X$. This fact is reflected in the definition of the distributive law below.

The distributive law is similar to the distributive law for the delay monad over the reader monad, and is given by:

$$\zeta(\mathrm{step}(d)) = \lambda g.\, \mathrm{step}(\lambda(\alpha : \kappa).(\zeta(d\,[\alpha]))g),$$

where $f : (X \to S) \to X$ and $d : \triangleright^\kappa(D^\kappa((X \to S) \to X))$. As a result, both $\mathcal{J}D^\kappa$ and $\mathcal{J}D$ can be equipped with monad structures.

Free Combinations With Delay

The sum of two monads T and S is a monad $T \oplus S$ whose algebras are objects X with algebra structures for both T and S [19]. In terms of algebraic theories, the sum can be understood as combining two theories with no equations between them. The sum of D^κ with any other monad always exists [19, Theorem 4]:

▶ **Corollary 18.** *Let T be a monad, and define $T \oplus D^\kappa$ as the guarded recursive type:*

$$(T \oplus D^\kappa)X \simeq T(X + \triangleright^\kappa((T \oplus D^\kappa)X)).$$

Then $(T \oplus D^\kappa)(X)$ is the carrier of the free T-algebra and delay-algebra structure.

The monad mapping X to $\forall\kappa.(T \oplus D^\kappa)(X)$ includes the coinductive delay monad and T, but we have not been able to prove a general universal property for this. We believe that it is not the sum of the two.

5 Parallel and Sequential Distribution of Operations

We now consider distributive laws of type $TD \to DT$, where D is one of the delay monads and T is any presentable monad. Such laws equip the composite DT with a monad structure, which is the natural one in particular for monads describing data structures, such as those in the Boom hierarchy.

We again focus on distributive laws involving the guarded version of the delay monad, invoking Lemma 16. Intuitively, such a distributive law pulls all the steps out of the algebraic structure of T: it turns a T-structure with delayed elements into a delayed T-structure. There are two obvious candidates for such a lifting: *parallel* and *sequential* computation. We define both of these on operations using guarded recursion. A lifting of terms then follows inductively from lifting each operation in the signature of the presentation of T.

▶ **Definition 19** (Parallel Lifting of Operators). *Let A be an algebraic theory, and let X be an A-model. Define, for each n-ary operation* op *in A, a lifting* $\mathrm{op}^{par}_{D^\kappa X} : (D^\kappa X)^n \to D^\kappa X$ *by:*

$$\mathrm{op}^{par}_{D^\kappa X}(\mathrm{now}\, x_1, \dots, \mathrm{now}\, x_n) = \mathrm{now}(\mathrm{op}_X(x_1, \dots, x_n))$$
$$\mathrm{op}^{par}_{D^\kappa X}(x_1, \dots, x_n) = \mathrm{step}(\lambda\alpha.(\mathrm{op}^{par}_{D^\kappa X}(x'_1, \dots, x'_n))),$$

where the second clause only applies if one of the x_i is of the form $\mathrm{step}(x''_i)$ *and*

$$x'_i = \begin{cases} x_i & \text{if } x_i = \mathrm{now}(x''_i) \\ x''_i\,[\alpha] & \text{if } x_i = \mathrm{step}(x''_i) \end{cases}$$

▶ **Definition 20** (Sequential Lifting of Operators). *Let A be an algebraic theory, and let X be an A-model. Define, for each n-ary operation* op *in A, a lifting* $\mathrm{op}^{seq}_{D^\kappa X} : (D^\kappa X)^n \to D^\kappa X$ *by:*

$$\mathrm{op}^{seq}_{D^\kappa X}(\mathrm{now}\, x_1, \dots, \mathrm{now}\, x_n) = \mathrm{now}(\mathrm{op}_X(x_1, \dots, x_n))$$
$$\mathrm{op}^{seq}_{D^\kappa X}(\mathrm{now}\, x_1, \dots, \mathrm{step}\, x_i, \dots x_n) = \mathrm{step}(\lambda\alpha.(\mathrm{op}^{seq}_{D^\kappa X}(\mathrm{now}\, x_1, \dots, (x_i\,[\alpha]), \dots, x_n))),$$

where, in the second clause, the ith argument is the first not of the form $\mathrm{now}(x'_k)$.

In general, for an n-ary operation op, parallel lifting evaluates all arguments of the form $\mathrm{step}(x_i)$ in parallel, and sequential lifting evaluates them one by one from the left. Parallel lifting of an operator therefore terminates in as many steps as the maximum required for each of its inputs to terminate, while sequential lifting terminates in the sum of the number of steps required for each input.

The evaluation order of arguments in the case of sequential lifting is inessential, which can be proved using guarded recursion and tick irrelevance.

▶ **Lemma 21.** *Let A be an algebraic theory, and let* op *be an n-ary operation in A. Then*

$$\mathrm{op}^{seq}_{D^\kappa X}(x_1, \dots, \mathrm{step}(x_i), \dots x_n) = \mathrm{step}(\lambda(\alpha : \kappa).\mathrm{op}^{seq}_{D^\kappa X}(x_1, \dots, x_i\,[\alpha], \dots, x_n)).$$

5.1 Preservation of Equations

Parallel lifting preserves all non-drop equations, whereas sequential lifting only preserves balanced equations. We prove this in the two following propositions. We write $s^{par}_{D^\kappa X}$ for the interpretation of a term s on $D^\kappa X$ defined by induction of s using the parallel lifting of operations, and likewise $s^{seq}_{D^\kappa X}$ for the interpretation defined using sequential lifting of operations.

▶ **Proposition 22** (Parallel Preserves Non-Drop). *Let $A = (\Sigma_A, E_A)$ be an algebraic theory, X an A-model, and $s = t$ a non-drop equation that is valid in A. Then also $s^{par}_{D^\kappa X} = t^{par}_{D^\kappa X}$.*

The restriction to non-drop equations is necessary, because divergence in a dropped variable leads to divergence on one side of the equation, but not on the other.

Møgelberg and Vezzosi [32] observed that parallel lifting does not define a distributive law in the case of the finite powerset monad. Their proof uses idempotency, but in fact parallel lifting does not define a monad even just in the presence of a single binary operation.

▶ **Theorem 23.** *Let T be an algebraic monad with a binary operation* op. *Then the natural transformation $\zeta : TD \to DT$ induced by parallel lifting does not define a distributive law, because it fails the second multiplication axiom.*

Proof. The counter example is the same as used by Møgelberg and Vezzosi:

$$\mathrm{op}^{par}_{DDX}(\mu^D(\mathrm{now}(\mathrm{step}\,\mathrm{now}\,x)), \mu^D(\mathrm{step}(\mathrm{now}(\mathrm{now}\,y))) = \mathrm{step}(\mathrm{now}(\mathrm{op}_X(x,y)))$$
$$\mu^D(\mathrm{op}^{par}_{DDX}(\mathrm{now}(\mathrm{step}(\mathrm{now}\,x)), \mathrm{step}(\mathrm{now}(\mathrm{now}\,y)))) = \mathrm{step}(\mathrm{step}(\mathrm{now}(\mathrm{op}_X(x,y)))). \quad \blacktriangleleft$$

Note that we used the coinductive version of the delay monad in the above theorem. By Lemma 16, this implies the same result for the guarded recursive version.

▶ **Proposition 24** (Sequential Preserves Balanced)**.** *Let A be an algebraic theory, and s, t be two A-terms such that $s = t$ is a balanced equation that is valid in A. Then also $s^{seq}_{D^\kappa X} = t^{seq}_{D^\kappa X}$.*

Balance is necessary. For example, if t and s are terms in a single variable which occurs twice in t and once in s, then $t^{seq}_{D^\kappa X}(\mathrm{step}(x))$ takes at least two steps, but $s^{seq}_{D^\kappa X}(\mathrm{step}(x))$ might take only one. Building on Proposition 24, one can prove the following.

▶ **Theorem 25.** *Let T be the free model monad of algebraic theory $\mathbb{T} = (\Sigma_\mathbb{T}, E_\mathbb{T})$, such that $E_\mathbb{T}$ only contains* balanced *equations. Then sequential lifting defines a distributive law $TD^\kappa \to D^\kappa T$.*

Combining this with Lemma 16 we obtain a distributive law $TD \to DT$ for all T as in Theorem 25.

▶ Remark 26. Since D^κ is a commutative monad, we already know from Manes and Mulry [26] and Parlant [34] that there is a distributive law in the case where $\mathbb{T}$ only has *linear* equations. We can extend this linearity requirement here to allow duplications of variables, as long as there are equally many duplicates on either side of each equation.

▶ **Example 27.** The sequential distributive law successfully combines the delay monad with the binary tree monad, the list monad, and the multiset monad, resulting in the monads $D^\kappa B$, $D^\kappa L$, and $D^\kappa M$, respectively.

6 Idempotent Equations

This section studies distributive laws $TD \to DT$ for T an algebraic monad with an idempotent binary operation "op". Since idempotency is not a balanced equation, as remarked after Proposition 24, sequential distribution does not respect it, and so neither parallel nor sequential distribution define distributive laws in this case. Idempotency turns out to be a tricky equation: We first show an example of such a theory T where no distributive law $TD \to DT$ is possible, then a theory where it is, and finally we show that no distributive law of type $TD^\kappa \to D^\kappa T$ is possible. First observe the following.

▶ **Lemma 28.** *Let T be an algebraic monad with an idempotent binary operation* op *and let $\zeta : TD \to DT$ be a distributive law. There exist binary T-operations* op_1 *and* op' *such that for any T-model X, the lifting of* op *to DX satisfies* $\mathrm{op}(\mathrm{step}\,x, \mathrm{step}\,y) = \mathrm{step}(\mathrm{op}_1(x,y))$ *and either 1)* $\mathrm{op}(\mathrm{step}\,x, y) = \mathrm{step}(\mathrm{op'}(x,y))$ *and* $\mathrm{op'}(x, \mathrm{step}\,y) = \mathrm{op}_1(x,y)$ *or 2)* $\mathrm{op}(\mathrm{step}\,x, y) = \mathrm{op'}(x,y)$ *and* $\mathrm{op'}(x, \mathrm{step}\,y) = \mathrm{step}(\mathrm{op}_1(x,y))$.

Proof Sketch. We just sketch the proof of existence of op_1. Consider the naturality diagram for the unique map $! : 2 \to 1$ from the 2-element set $\{\mathsf{tt}, \mathsf{ff}\}$ to the singleton set $\{\star\}$.

$$\begin{array}{ccc} D(T(2)) \times D(T(2)) & \xrightarrow{\mathrm{op}} & D(T(2)) \\ {\scriptstyle D(T(!))\times D(T(!))}\downarrow & & \downarrow{\scriptstyle D(T(!))} \\ D(T(1)) \times D(T(1)) & \xrightarrow{\mathrm{op}} & D(T(1)) \end{array}$$

By idempotency, the lower composition maps $(\mathrm{step}(\mathrm{now}(\eta^T(\mathsf{tt}))), \mathrm{step}(\mathrm{now}(\eta^T(\mathsf{ff}))))$ to $\mathrm{step}\,\mathrm{now}(\eta^T(\star))$. Therefore it must be the case that $\mathrm{op}(\mathrm{step}(\mathrm{now}(\eta^T(\mathsf{tt}))), \mathrm{step}(\mathrm{now}(\eta^T(\mathsf{ff}))))$ is $\mathrm{step}(\mathrm{now}(\mathrm{op}_1(\eta^T(\mathsf{tt}), \eta^T(\mathsf{ff}))))$ for some op_1. ◀

▶ **Proposition 29.** *There is no distributive law $\mathsf{P_f}D \to D\mathsf{P_f}$ for $\mathsf{P_f}$ the finite powerset functor.*

Proof Sketch. There are only four possible cases for $\mathrm{op}_1(x, y)$ and op'(x, y): $\emptyset, \{x\}, \{y\}$ and $\{x, y\}$. An easy analysis rules out the first three. Lemma 28 then implies that $\{\mathrm{step}(x), y\} = \mathrm{step}(\{x, y\})$. This leads to a contradiction as follows

$$\mathrm{step}(\{x\}) = \{\mathrm{step}(x)\} = \{\mathrm{step}(x), \mathrm{step}(x)\} = \mathrm{step}(\{x, \mathrm{step}(x)\}) = \mathrm{step}^2(\{x\}).$$ ◀

▶ **Example 30.** Let A be the algebraic theory with one idempotent binary operation $*$ and one unary operator !, with no further equations. Let T be the monad generated by A. There is a distributive law $\zeta : TD \to DT$ given by the following clauses

$$!(\mathrm{step}(x)) = x \qquad \mathrm{step}(x) * y = \mathrm{step}(x * (!y)) \qquad x * \mathrm{step}(y) = \mathrm{step}((!x) * y).$$

Note in particular, that $\mathrm{step}(x) * \mathrm{step}(y) = \mathrm{step}(x * (!(\mathrm{step}(y)))) = \mathrm{step}(x * y)$. This example can be extended to $*$ associative, if the equation $!(x * y) = (!x) * (!y)$ is added.

▶ **Theorem 31** (No-Go Theorem)**.** *Let T be a monad with a binary algebraic operation that is commutative and idempotent. Then there is no distributive law of type $TD^\kappa \to D^\kappa T$.*

7 Semi-Go Theorem: Up to Weak Bisimilarity

In the proof of Theorem 31 the failure of existence of distributive laws comes down to a miscounting of steps. This section shows that this is indeed all that fails, and that parallel lifting defines a distributive law *up to weak bisimilarity* for algebraic monads with no drop equations. Weak bisimilarity is a relation on the coinductive delay monad, which relates computations that only differ by a finite number of steps. To make this precise, we work in a category of setoids. The objects are pairs (X, R), where R is an equivalence relation on X, and morphisms are equivalence classes of maps f between the underlying sets respecting the relations. Two such maps are equivalent if their values on equal input are related by the equivalence relation on the target type.

We first define a lifting of the coinductive delay monad D to the category of setoids. We do this via a similar relation (taken from Møgelberg and Paviotti [31]) defined for the guarded delay monad, because that allows us to reason using guarded recursion. We write δ^κ for $\mathrm{step} \circ \mathrm{next} : D^\kappa X \to D^\kappa X$.

▶ **Definition 32** ([31])**.** *Let X, Y be sets, and suppose $R : X \to Y \to \mathsf{Prop}$ is a relation. Define weak bisimilarity up to R, written $\sim_R^\kappa : D^\kappa X \to D^\kappa Y \to \mathsf{Prop}$, by:*

$$\mathrm{now}(x) \sim_R^\kappa y \stackrel{def}{=} \exists(n : \mathbb{N}, y' : Y).y = (\delta^\kappa)^n(\mathrm{now}(y'))\mathit{and}\ R(x, y'),$$
$$x \sim_R^\kappa \mathrm{now}(y) \stackrel{def}{=} \exists(n : \mathbb{N}, x' : X).x = (\delta^\kappa)^n(\mathrm{now}(x'))\mathit{and}\ R(x', y),$$
$$\mathrm{step}(x) \sim_R^\kappa \mathrm{step}(y) \stackrel{def}{=} \triangleright(\alpha : \kappa).(x\,[\alpha] \sim_R^\kappa y\,[\alpha]).$$

Note that the two first cases both apply for $\mathrm{now}(x) \sim_R^\kappa \mathrm{now}(y)$, but that they are equivalent in that case. If R is symmetric and reflexive, then the same properties hold for $\sim_R^\kappa$, but transitivity is not preserved. In fact, if $\sim_=^\kappa$ were transitive, then one could prove that it is the total relation, which is not the case.

▶ **Definition 33.** *Let* $R : X \to Y \to \mathsf{Prop}$ *be a relation. Define* $\sim_R : DX \to DY \to \mathsf{Prop}$ *as*

$$x \sim_R y \stackrel{def}{=} \forall \kappa . x\,[\kappa] \sim_R^\kappa y\,[\kappa].$$

The above definition is an encoding (using guarded recursion) of the standard coinductive definition of weak bisimilarity. We note the following, which was also observed by Chapman et al [10].

▶ **Proposition 34.** *The mapping* $D_{\mathsf{sd}}(X, R) = (DX, \sim_R)$ *defines a monad on the category of setoids.*

In fact, the multiplication for D^κ preserves the guarded recursive definition of weak bisimilarity.

Similarly, any algebraic monad T can be lifted to the category of setoids by defining $T(R)$ to be the smallest equivalence relation relating an equivalence class $[t(x_1, \ldots, x_n)]$ to $[t(y_1, \ldots, y_n)]$ if $R(x_i, y_i)$ for all i. We write T_{sd} for this.

Recall that by Proposition 22, if T is an algebraic monad given by a theory with no drop equations, then parallel lifting defines a natural transformation $TD \to DT$ on the category of sets. We show that this map lifts to a distributive law on the category of setoids.

▶ **Theorem 35.** *Let* T *be the free model monad of algebraic theory* $\mathbb{T} = (\Sigma_\mathbb{T}, E_\mathbb{T})$*, such that* $E_\mathbb{T}$ *contains no drop equations. Then parallel lifting defines a distributive law of monads* $T_{\mathsf{sd}} D_{\mathsf{sd}} \to D_{\mathsf{sd}} T_{\mathsf{sd}}$.

▶ Remark 36. The free monad on an algebraic theory could alternatively be expressed on the category of setoids by taking the set to be the free monad just on operations, introducing the equations of the theory into the equivalence relation. In the presence of the axiom of choice this generates a monad equivalent to T_{sd}, and we expect that the proof above can be adapted to that choice as well.

8 Related Work

Møgelberg and Vezzosi [32] study two combinations of the guarded delay monad D^κ with the finite powerset monad $\mathsf{P_f}$ expressed as a HIT in CCTT. They use these to show that applicative simulation is a congruence for the untyped lambda calculus with finite non-determinism using denotational techniques. One combination is the sum $\mathsf{P_f} \oplus D^\kappa$, which is used for the case of may-convergence, and the other is the composite $D^\kappa \mathsf{P_f}$ equipped with the parallel lifting, which is used for must-convergence. They observe that only the former is a monad. In this paper, we not only provide a more general study of such combinations, but also suggest a way to remedy the situation in the latter case by considering weak bisimilarity.

Weak bisimilarity for the coinductive delay monad was first defined by Capretta [8]. Møgelberg and Paviotti [31] show that their embedding of FPC in guarded dependent type theory respects weak bisimilarity and use that to prove an adequacy theorem up to weak bisimilarity.

Chapman et al. [10] observe that quotienting the coinductive delay monad by weak bisimilarity appears to not yield a monad unless countable choice is assumed. Altenkirch et al. [1] propose a solution to this problem by constructing the quotient and the weak bisimilarity relation simultaneously, as a higher inductive-inductive type. Chapman et al. themselves suggest a different solution, constructing the quotient as the free ω-cpo using an ordinary HIT. These quotients have not (to the best of our knowledge) been studied in combination with other effects.

Interaction trees [42] are essentially monads of the form $\forall\kappa.(T \oplus D^\kappa)(-)$ for T an algebraic monad generated by operations with no equations. Much work has gone into building libraries for working with these up to weak bisimilarity in Coq, and these allow for interaction trees to be used for program verification. To our knowledge, versions of interaction trees with equations between terms have not been considered.

As mentioned in the introduction, the guarded recursive delay monad has two benefits over the coinductive one: Firstly, it has a fixed point operator of the type (rather than an iteration operator), which means that it allows for embedding languages with recursion directly in type theory. In the coinductive case, one must either use some encoding of recursion using the iteration operator, or prove that all constructions used are continuous. We believe this is a considerable burden for higher order functions. The second advantage is that guarded recursion allows for also advanced notions of state to be encoded, as shown recently by Sterling et al. [38]. Neither the interaction trees nor the quotiented delay monads appear to have these benefits.

Related Work on Monad Compositions

The field of monad compositions in general has attracted quite a bit of attention lately. After Plotkin proved that there is no distributive law combining probability and non-determinism [40], Klin and Salamanca [21] studied impossible distributions of the powerset monad over itself, while Zwart and Marsden provided a general study on what makes distributive laws fail [43]. Meanwhile, the initial study of monad compositions by Manes and Mulry [26, 27] was continued by Parlant et al [13, 34, 35]. In both the positive and the negative theorems on distributive laws in these papers, certain classes of equations were identified as causes for making or breaking the monad composition. Idempotence, duplication, and dropping variables came out as especially noteworthy types of equations, which the findings in this paper confirm.

Our study of the delay monad provides an interesting extension on the previous works, because of its non-standard algebraic structure given by delay algebras, and the fact that the delay monad is neither affine nor relevant, which are the main properties studied by Parlant et al.

9 Conclusion and Future Work

We have studied how both the guarded recursive and the coinductive version of the delay monad combine with other monads. After studying some specific examples and free combinations, we looked more generally at possible distributive laws of $TD^\kappa \to D^\kappa T$. We found two natural candidates for such distributive laws, induced by *parallel* and *sequential* lifting of operations on T. We showed that:

- Sequential lifting provides a distributive law for monads presented by theories with balanced equations.
- There is no distributive law possible for monads with a binary operation that is commutative and idempotent over D^κ, but this does not rule out a distributive law of such monads over D.
- Parallel lifting does not define a distributive law, but it does define one *up to weak bisimilarity*, for monads presented by theories with non-drop equations.

It is unfortunate that weak bisimilarity requires working with setoids, but this is due to the quotient of D up to weak bisimilarity not being a monad [10]. It is not clear how to adapt the solutions to this problem mentioned above [10, 1] to the guarded recursive setting.

This paper only considers the case of finite arity operations (except for state, which can be of any arity). Distributive laws for countable arity operations such as countable non-deterministic choice are more difficult. In those cases sequential lifting seems an unnatural choice, not only because it does not interact well with idempotency, but also because it introduces divergence even in the cases where there is an upper limit to the number of steps taken by the arguments. Extending our parallel lifting operation to the countable case requires deciding whether all the countably many input operations are values, which is not possible in type theory.

The results presented in this paper are formulated and proven in CCTT. It is natural to ask whether the results proven for the coinductive delay monad D also hold for D considered as a monad on the category Set of sets. For some of the results proven in this paper (Proposition 29 and Example 30) both the statements and proofs can be read in Set. These results can therefore easily be seen to hold in this setting. In many other cases, our constructions use guarded recursion (e.g. the definitions of parallel and sequential lifting of operators). To lift these results to Set, one would need to redo the constructions and argue for their productivity. However, we believe that using guarded recursion is the natural way to work with coinductive types and proofs. Another approach could therefore be to use guarded recursion as a language to reason about Set. This should be possible because the universe used to model clock irrelevant types in the extensional model of Clocked Type Theory [28] classifies a category equivalent to Set. We leave this as a direction for future research.

References

1 Thorsten Altenkirch, Nils Anders Danielsson, and Nicolai Kraus. Partiality, revisited. In *International Conference on Foundations of Software Science and Computation Structures*, pages 534–549. Springer, 2017.

2 Robert Atkey and Conor McBride. Productive coprogramming with guarded recursion. *ACM SIGPLAN Notices*, 48(9):197–208, 2013.

3 Patrick Bahr., Hans Bugge Grathwohl, and Rasmus Ejlers Møgelberg. The clocks are ticking: No more delays! In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–12. IEEE, 2017.

4 Patrick Bahr and Graham Hutton. Monadic compiler calculation (functional pearl). *Proc. ACM Program. Lang.*, 6(ICFP), August 2022. doi:10.1145/3547624.

5 Magnus Baunsgaard Kristensen, Rasmus Ejlers Mogelberg, and Andrea Vezzosi. Greatest hits: Higher inductive types in coinductive definitions via induction under clocks. In *Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 1–13, 2022.

6 Jon Beck. Distributive laws. In B. Eckmann, editor, *Seminar on Triples and Categorical Homology Theory*, pages 119–140. Springer Berlin Heidelberg, 1969.

7 Lars Birkedal, Rasmus Ejlers Møgelberg, Jan Schwinghammer, and Kristian Støvring. First steps in synthetic guarded domain theory: step-indexing in the topos of trees. *Logical Methods in Computer Science*, 8(4), 2012.

8 Venanzio Capretta. General recursion via coinductive types. *Logical Methods in Computer Science*, 1, 2005.

9 Evan Cavallo and Robert Harper. Higher inductive types in cubical computational type theory. *Proceedings of the ACM on Programming Languages*, 3(POPL):1–27, 2019.

10 James Chapman, Tarmo Uustalu, and Niccolò Veltri. Quotienting the delay monad by weak bisimilarity. *Mathematical Structures in Computer Science*, 29(1):67–92, 2019.

11 Ranald Clouston. Fitch-style modal lambda calculi. In *International Conference on Foundations of Software Science and Computation Structures*, pages 258–275. Springer, 2018.

12 Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. Cubical type theory: A constructive interpretation of the univalence axiom. In *21st International Conference on Types for Proofs and Programs (TYPES 2015)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

13 Fredrik Dahlqvist, Louis Parlant, and Alexandra Silva. Layer by layer - combining monads. In *Theoretical Aspects of Computing – ICTAC 2018 – 15th International Colloquium, Stellenbosch, South Africa, October 16-19, 2018, Proceedings*, pages 153–172, 2018. doi:10.1007/978-3-030-02508-3_9.

14 Nils Anders Danielsson. Operational semantics using the partiality monad. *SIGPLAN Not.*, 47(9):127–138, September 2012. doi:10.1145/2398856.2364546.

15 Martín Escardó and Paulo Oliva. Selection functions, bar recursion and backward induction. *Mathematical Structures in Computer Science*, 20(2):127–168, 2010. doi:10.1017/S0960129509990351.

16 ND Gautam. The validity of equations of complex algebras. *Archiv für mathematische Logik und Grundlagenforschung*, 3(3):117–124, 1957.

17 Daniel Gratzer, GA Kavvos, Andreas Nuyts, and Lars Birkedal. Multimodal dependent type theory. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 492–506, 2020.

18 Jules Hedges. Monad transformers for backtracking search. In Paul Levy and Neel Krishnaswami, editors, Proceedings 5th Workshop on *Mathematically Structured Functional Programming,* Grenoble, France, 12 April 2014, volume 153 of *Electronic Proceedings in Theoretical Computer Science*, pages 31–50. Open Publishing Association, 2014. doi:10.4204/EPTCS.153.3.

19 Martin Hyland, Gordon Plotkin, and John Power. Combining effects: Sum and tensor. *Theoretical Computer Science*, 357(1):70–99, 2006. doi:10.1016/j.tcs.2006.03.013.

20 Martin Hyland and John Power. Discrete lawvere theories and computational effects. *Theoretical Computer Science*, 366(1-2):144–162, 2006.

21 Bartek Klin and Julian Salamanca. Iterated covariant powerset is not a monad. In *Proceedings 34th Conference on the Mathematical Foundations of Programming Semantics, MFPS 2018*, 2018.

22 Anders Kock. Monads on symmetric monoidal closed categories. *Archiv der Mathematik*, 21(1):1–10, 1970.

23 F. William Lawvere. Functorial semantics of algebraic theories. *Proceedings of the National Academy of Sciences of the United States of America*, 50(5):869–872, 1963.

24 Fred EJ Linton. Some aspects of equational categories. In *Proceedings of the Conference on Categorical Algebra*, pages 84–94. Springer, 1966.

25 Ernie Manes. *Algebraic theories*, volume 26. Springer, 1976.

26 Ernie Manes and Philip Mulry. Monad compositions I: general constructions and recursive distributive laws. *Theory and Applications of Categories*, 18:172–208, April 2007.

27 Ernie Manes and Philip Mulry. Monad compositions II: Kleisli strength. *Mathematical Structures in Computer Science*, 18(3):613–643, 2008. doi:10.1017/S0960129508006695.

28 Bassel Mannaa, Rasmus Ejlers Møgelberg, and Niccolò Veltri. Ticking clocks as dependent right adjoints: Denotational semantics for clocked type theory. *Logical Methods in Computer Science*, 16, 2020.

29 Per Martin-Löf. *Intuitionistic Type Theory.* Bibliopolis, Napoli, 1984.

30 Lambert Meertens. Algorithmics, towards programming as a mathematical activity. In J.W. De Bakker, M. Hazewinkel, and J.K. Lenstra, editors, *Mathematics and Computer Science: Proceedings of the CWI Symposium, November 1983*, CWI monographs, pages 289–334. North-Holland, 1986.

31 Rasmus E Møgelberg and Marco Paviotti. Denotational semantics of recursive types in synthetic guarded domain theory. *Mathematical Structures in Computer Science*, 29(3):465–510, 2019.

32 Rasmus Ejlers Møgelberg and Andrea Vezzosi. Two guarded recursive powerdomains for applicative simulation. In Ana Sokolova, editor, *Proceedings 37th Conference on Mathematical Foundations of Programming Semantics, MFPS 2021, Hybrid: Salzburg, Austria and Online, 30th August - 2nd September, 2021*, volume 351 of *EPTCS*, pages 200–217, 2021. doi:10.4204/EPTCS.351.13.

33 Hiroshi Nakano. A modality for recursion. In *Proceedings Fifteenth Annual IEEE Symposium on Logic in Computer Science*, pages 255–266. IEEE, 2000.

34 Louis Parlant. *Monad Composition via Preservation of Algebras*. PhD thesis, UCL (University College London), 2020.

35 Louis Parlant, Jurriaan Rot, Alexandra Silva, and Bas Westerbaan. Preservation of equations by monoidal monads. In *45th International Symposium on Mathematical Foundations of Computer Science (MFCS 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

36 Marco Paviotti, Rasmus Ejlers Møgelberg, and Lars Birkedal. A model of PCF in guarded type theory. *Electronic Notes in Theoretical Computer Science*, 319:333–349, 2015.

37 Gordon Plotkin and John Power. Notions of computation determine monads. In *International Conference on Foundations of Software Science and Computation Structures*, pages 342–356. Springer, 2002.

38 Jonathan Sterling, Daniel Gratzer, and Lars Birkedal. Denotational semantics of general store and polymorphism. Unpublished manuscript, July 2022.

39 The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. https://homotopytypetheory.org/book, Institute for Advanced Study, 2013.

40 Daniele Varacca and Glynn Winskel. Distributing probability over non-determinism. *Mathematical Structures in Computer Science*, 16(1):87–113, 2006.

41 Niccolò Veltri and Niels F. W. Voorneveld. Inductive and coinductive predicate liftings for effectful programs. In Ana Sokolova, editor, *Proceedings 37th Conference on Mathematical Foundations of Programming Semantics, MFPS 2021, Hybrid: Salzburg, Austria and Online, 30th August - 2nd September, 2021*, volume 351 of *EPTCS*, pages 260–277, 2021. doi:10.4204/EPTCS.351.16.

42 Li-yao Xia, Yannick Zakowski, Paul He, Chung-Kil Hur, Gregory Malecha, Benjamin C. Pierce, and Steve Zdancewic. Interaction trees: Representing recursive and impure programs in Coq. *Proc. ACM Program. Lang.*, 4(POPL), December 2019. doi:10.1145/3371119.

43 Maaike Zwart and Dan Marsden. No-go theorems for distributive laws. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science (LiCS)*, pages 1–13, 2019. doi:10.1109/LICS.2019.8785707.

Syntactically and Semantically Regular Languages of λ-Terms Coincide Through Logical Relations

Vincent Moreau
IRIF & Université Paris Cité & Inria Paris, France

Lê Thành Dũng (Tito) Nguyễn
Laboratoire de l'informatique du parallélisme (LIP), École normale supérieure de Lyon, France

Abstract

A fundamental theme in automata theory is regular languages of words and trees, and their many equivalent definitions. Salvati has proposed a generalization to regular languages of simply typed λ-terms, defined using denotational semantics in finite sets.

We provide here some evidence for its robustness. First, we give an equivalent syntactic characterization that naturally extends the seminal work of Hillebrand and Kanellakis connecting regular languages of words and syntactic λ-definability. Second, we show that any finitary extensional model of the simply typed λ-calculus, when used in Salvati's definition, recognizes exactly the same class of languages of λ-terms as the category of finite sets does.

The proofs of these two results rely on logical relations and can be seen as instances of a more general construction of a categorical nature, inspired by previous categorical accounts of logical relations using the gluing construction.

2012 ACM Subject Classification Theory of computation → Denotational semantics; Theory of computation → Regular languages

Keywords and phrases regular languages, simple types, denotational semantics, logical relations

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.40

Related Version *Full version with additional proofs*: `https://arxiv.org/abs/2308.00198`

Funding *Lê Thành Dũng (Tito) Nguyễn*: Supported by the LABEX MILYON (ANR-10-LABX-0070) of Université de Lyon, within the program "Investissements d'Avenir" (ANR-11-IDEX-0007) operated by the French National Research Agency (ANR).

Acknowledgements We would like to thank Amina Doumane, Sam van Gool, Paul-André Melliès and Sylvain Salvati for in-depth discussions that significantly helped us refine our ideas. We are also grateful to Sam and Paul-André for proof-reading drafts of this paper, and to the ReFL discussion group `https://www.engboris.fr/refl/` for hosting a reading group on logical relations and normalization by evaluation. The first author would like to thank the Felicissimo family for their support during the writing process of this article.

1 Introduction

Much work has been devoted to the study of regular languages of words and trees – also called recognizable – and their equivalent characterizations, typically in terms of automata, algebra, and logic. The remarkable robustness of this notion of regularity has led to attempts to extend it to several other structures, such as infinite words/trees or graphs of bounded treewidth – many examples can be found, for instance, in [4].

This paper focuses on a less studied extension: recognizable languages of simply typed λ-terms, introduced by Salvati [27]. They are a conservative generalization [27, §3]:[1] using the Church encoding, finite words and trees can be represented in the simply typed λ-calculus,

[1] Alternatively, see [34, Proposition 7.1] for the case of words.

and recognizability for Church encodings coincides with the usual notion of regularity. Furthermore, as Salvati explains in his habilitation thesis [28], regular languages of λ-terms can be used to shed light on several classical topics concerning the simply typed λ-calculus, such as higher-order matching [27, §6.2] and higher-order grammars [17].

However, not many characterizations of the class of recognizable languages of simply typed λ-terms are known, and it would be desirable to have more evidence that it is robust. Moreover, if we know that this class of languages is a truly canonical object, then so is its Stone dual, namely the recently introduced space of profinite λ-terms [34]. Currently, there exist two definitions of recognizable languages of λ-terms, both provided in [27]:

- The first one [27, Definition 1] uses denotational semantics into the finite standard models of the simply typed λ-calculus. This may be understood as generalizing to higher orders the computational aspects of deterministic finite automata. In the same vein, the concrete construction of profinite λ-terms in [34] depends on specific properties of the category **FinSet** of finite sets and functions between them.
- The second one, grounded in intersection types [27, §4], turns out to admit an equivalent presentation in terms of a denotational semantics in finite domains [28, Theorem 25]. Indeed, the connection between intersection types and semantics is standard, see e.g. [26].

Both definitions can be seen as an instance of the following pattern: the interpretation of a simply typed λ-term in some denotational model with a *finitary* flavor should suffice to know whether the term belongs to our language of interest. This is closely analogous to the algebraic definition of regular word languages. Indeed, viewing the set Σ^* of words over a finite alphabet Σ as the free monoid generated by Σ, a language $L \subseteq \Sigma^*$ is regular if and only if, for some homomorphism $\varphi\colon \Sigma^* \to M$ to a finite monoid M, the "interpretation" $\varphi(w)$ determines[2] whether $w \in L$ for each $w \in \Sigma^*$. Asking for φ to be a homomorphism parallels the compositionality property of denotational semantics.

We may therefore ask:

- What kind of semantics yield the same notion of recognizable language? More precisely, with a definition of **C**-recognizable language for any cartesian closed category (CCC) **C**, i.e. any categorical model of the simply typed λ-calculus with products, the question becomes: when do recognizability by **C** and by **FinSet** coincide?
- Alternatively, is there any characterization of regular languages of λ-terms that does not involve denotational semantics?

For the latter, we propose a positive answer inspired by the following result of Hillebrand and Kanellakis [14, Theorem 3.4]: a language of words is regular if and only if it can be decided by a simply typed λ-term operating on Church-encoded words. By replacing the type of Church encodings with any simple type A, we get a natural notion of syntactically regular language of λ-terms, defined by means purely internal to the simply typed λ-calculus.

Contributions and proof methods. The main results of this paper are as follows:

Theorem 3.2 any non-degenerate cartesian closed category can recognize at least every language which is syntactically regular.

Theorem 5.9 every language recognized by a locally finite and well-pointed CCC – in other words, a finitary extensional model of the simply typed λ-calculus – is recognized by **FinSet**. This was first stated by Salvati without proof in [28, Lemma 20].

Theorem 6.5 every language recognized by **FinSet** is syntactically regular.

[2] More formally, $L = \varphi^{-1}(P)$ for some $P \subseteq M$.

These three theorems, taken together, show that all non-degenerate, well-pointed and locally finite CCCs yield the same notion of regular language of λ-terms, which is the same as the syntactic one.

To achieve this goal, we introduce a construction on CCCs, which we call squeezing, and combine it with the standard categorical account of logical relations based on sconing. Indeed, Salvati claims in [28] that Theorem 5.9 can be established via logical relations, and it turns out that this falls out directly from our squeezing construction; but its versatility also allows us to apply it to prove the more diffcult Theorem 6.5 on syntactic recognizability.

Related work. Morally, the study of regular languages of λ-terms amounts to understanding what information can be extracted by evaluating simply typed λ-terms in finitary models. A seminal result in this spirit is Statman's finite completeness theorem [30], which can be rephrased as the regularity of all singleton languages of λ-terms – a perspective that has led to a simplified proof [29]. The idea of using another CCC than **FinSet**, easier to use to show Statman's theorem, has been exploited in [17]. This shows the advantage to use an appropriate CCC to recognize a given language, a possibility which is extended to a vast class of CCCs in this paper (see Proposition 7.3 for an example of application).

Finitary semantics are powerful tools, in particular, for understanding the computational power of the simply typed λ-calculus. For instance, in [14], Hillebrand and Kanellakis use the finite set semantics to prove their aforementioned theorem on regular word languages; as for finite Scott domains presented as intersection types, they have been applied by Terui [33] to study the complexity of normalizing simply typed λ-terms.

As can be seen *inter alia* from rather surprising results of Statman [31] and Plotkin [25], finitary models are also useful to tame the infinitary aspects of an extension of the simply typed λ-calculus with a fixed-point operator, called the λY-calculus. Furthermore, the well-studied higher-order model checking problem is about testing regular properties on infinite trees that can be Church-encoded in the λY-calculus; it sits at the interface between automata and programming languages, with applications to the formal verification of functional programs (see e.g. [16]). Decidability of higher-order model checking, first established by Ong through game semantics [23], now admits proofs based on intersection types [15, 24] and on finitary semantics [35, 10]. Drawing on this line of work, higher-order parity automata [19] generalize to λY-terms the recognizable languages of simply typed λ-terms.

The theme of syntactic recognizability *à la* Hillebrand and Kanellakis, for its part, has been recently revived in Nguyễn and Pradic's implicit[3] automata theory. They use substructural λ-calculi and Church encodings to characterize star-free languages [22] and classes of string-to-string functions computed by transducers [21].

Plan of the paper. We start by recalling in Section 2 the semantics of the simply typed λ-calculus, the notion of language recognized by a CCC as defined in [27] for finite sets, and by introducing the notion of syntactically regular language, generalizing recognition as defined in [14]. In Theorem 3.2 of Section 3, we show that every non-degenerate CCC recognizes all syntactically regular languages. In Section 4, we recall the definition of logical relations and introduce the squeezing construction $\mathbf{Sqz}(-)$ which will be a crucial tool for

[3] The name is a nod to implicit computational complexity, a field concerned with alternative definitions of complexity classes that avoid low-level machine models and explicit resource bounds. As an example, in addition to their result on regular word languages in the simply typed λ-calculus, Hillebrand and Kanellakis's paper [14] also contains characterizations of the k-EXPTIME and k-EXPSPACE hierarchies based on λ-terms, that are again proved by evaluation in finite sets.

the two next sections. In Section 5, we recall the definition of locally finite and well-pointed CCCs, and show in Theorem 5.9 that CCCs enjoying both conditions do not recognize more languages than finite sets do. In Theorem 6.5 of Section 6, we show that languages recognized by finite sets are syntactically regular. We finish this paper by giving some consequences of the equivalence established by these three theorems in Section 7.

2 Languages of λ-terms

Syntax and semantics

We first specify the syntax we are working with. The grammars of types and preterms are

$$A, B ::= \mathfrak{o} \mid A \Rightarrow B \mid A \times B \mid 1 \quad \text{and} \quad t, u ::= x \mid \lambda(x : A).\, t \mid t\ u \mid \langle t, u\rangle \mid t_i \text{ for } i = 1, 2$$

and we consider the usual typing rules and $\beta\eta$-conversion rules, see e.g. [2, §4.1]. We extend the notation of t_i, for the projection to the i^{th} coordinate, to the case where t is of type $A_1 \times \cdots \times A_n$ and i is between 1 and n. As the λ-abstractions are annotated, a closed λ-term has at most one type derivation. For any simple type A, we write $\Lambda(A)$ for the set of closed simply typed λ-terms of type A, taken modulo $\beta\eta$-conversion.

We recall the semantics of the simply typed λ-calculus into cartesian closed categories, abbreviated as CCC, see [2, Chapter 4] for more details. For any CCC $\mathbf{C}$, object c of $\mathbf{C}$ and simple type A, we define an object $[\![A]\!]_c$ of $\mathbf{C}$ by induction on A as follows:

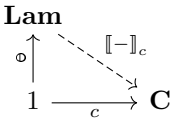
$$[\![\mathfrak{o}]\!]_c := c \qquad [\![A \Rightarrow B]\!]_c := [\![A]\!]_c \Rightarrow [\![B]\!]_c \qquad [\![A \times B]\!]_c := [\![A]\!]_c \times [\![B]\!]_c \qquad [\![1]\!]_c := 1$$

Using the CCC structure of $\mathbf{C}$, one can define a family of set-theoretic functions

$$[\![-]\!]_c \quad : \quad \Lambda(A) \longrightarrow \mathbf{C}(1, [\![A]\!]_c) \qquad \text{for every simple type } A$$

called semantic brackets, sending closed λ-terms to points of the objects $[\![A]\!]_c$.

These assignments can be described in another way. Let **Lam** be the category whose objects are simple types and whose set of morphisms from A to B is $\Lambda(A \Rightarrow B)$, with the expected composition. This category is the free CCC on one object, i.e., for every CCC $\mathbf{C}$ and object c of $\mathbf{C}$, there exists a unique CCC functor $[\![-]\!]_c : \mathbf{Lam} \to \mathbf{C}$ such that $[\![\mathfrak{o}]\!]_c = c$. This can be represented by the commutativity of the following diagram:



(1)

In this paper, the CCCs come with specified terminal object, cartesian products, and exponentials, and CCC functors are required to respect these structures strictly, *on the nose*. In that way, the unicity in the universal property of **Lam** depicted in Equation (1) holds up to equality, and not merely isomorphism.

We write **FinSet** for the cartesian closed category of finite sets. The semantics of the simply typed λ-calculus in this CCC corresponds to its naive set-theoretic interpretation. For ease of notation, we identify the finite set Q with the set of functions $\mathbf{FinSet}(1, Q)$.

Recognizable languages of λ-terms, semantically

We now define the notion of $\mathbf{C}$-recognizable language of λ-terms, for any CCC $\mathbf{C}$. The case $\mathbf{C} = \mathbf{FinSet}$ corresponds to the notion of regular language of simply typed λ-terms introduced in [27, Definition 1].

▶ **Definition 2.1.** *Let* $\mathbf{C}$ *be a CCC and* c *be an object of* $\mathbf{C}$*. For every simple type* A *and subset* $F \subseteq \mathbf{C}(1, [\![A]\!]_c)$*, the language* L_F *of* λ*-terms of type* A *is defined as*

$$L_F \quad := \quad \{t \in \Lambda(A) \mid [\![t]\!]_c \in F\} \ .$$

We define the set $\mathrm{Rec}_c(A)$ *of languages of* λ*-terms of type* A *recognized by* c *as*

$$\mathrm{Rec}_c(A) \quad := \quad \{L_F : F \subseteq \mathbf{C}(1, [\![A]\!]_c)\} \ .$$

Finally, a language L *of* λ*-terms of type* A *is* $\mathbf{C}$-recognizable *if there exists an object* c *of* $\mathbf{C}$ *such that* L *belongs to the set* $\mathrm{Rec}_c(A)$*.*

▶ **Example 2.2.** For any natural number n, we define the associated simple type

$$\mathsf{Church}_n \ := \ (o \Rightarrow o)^n \Rightarrow o \Rightarrow o \ .$$

There is a bijection between the sets $\Lambda(\mathsf{Church}_n)$ and $\{1, \dots, n\}^*$, the set of finite words over an alphabet with n letters, called the *Church encoding*. For example, the word 12212 over the two letter-alphabet $\{1, 2\}$ is encoded as the λ-term

$$\lambda(a : (o \Rightarrow o)^2).\, \lambda(e : o).\ a_2\ (a_1\ (a_2\ (a_2\ (a_1\ e)))) \quad \in \quad \Lambda(\mathsf{Church}_2) \ .$$

Under this bijection, a language of λ-terms of type Church_n is **FinSet**-recognizable, in the sense of Definition 2.1, if and only if the language of words associated by the Church encoding is a regular language of finite words, see [34, Proposition 7.1].

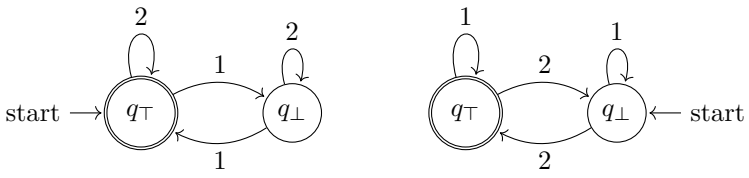
▶ **Example 2.3.** We give a detailed example using the Church encoding. We show that the language L of λ-terms of type Church_2 which are encodings of words in $\{1, 2\}^*$ that contain an even number of 1s and an odd number of 2s, is **FinSet**-recognizable.

Let Q be the finite set $\{q_\top, q_\bot\}$ and F be the subset of $[\![\mathsf{Church}_2]\!]_Q$ defined as

$$F \ := \ \{f \in (Q \Rightarrow Q) \times (Q \Rightarrow Q) \Rightarrow Q \Rightarrow Q \mid f(\mathrm{not}, \mathrm{Id}_Q)(q_\top) = f(\mathrm{Id}_Q, \mathrm{not})(q_\bot) = q_\top\}$$

where $\mathrm{not} : Q \to Q$ is the function defined as $\mathrm{not}(q_\top) = q_\bot$ and $\mathrm{not}(q_\bot) = q_\top$. The language L is equal to L_F which belongs to $\mathrm{Rec}_Q(\mathsf{Church}_2)$, so L is **FinSet**-recognizable.

The idea is that, given the semantic interpretation $f \in [\![\mathsf{Church}_2]\!]_Q$ of the encoding of a word $w \in \{1, 2\}^*$, the states $f(\mathrm{not}, \mathrm{Id}_Q)(q_\top)$ and $f(\mathrm{Id}_Q, \mathrm{not})(q_\bot)$ are the states reached respectively, after reading w, in the two following deterministic finite automata:



The language L is the intersection of the two languages recognized by these automata.

▶ **Example 2.4.** We consider the simple type

$$\mathsf{UntypedTerms} \quad := \quad ((o \Rightarrow o) \Rightarrow o) \Rightarrow (o \Rightarrow o \Rightarrow o) \Rightarrow o \ .$$

There is a canonical bijection – which is classical, see e.g. [3] for an in-depth treatment – between $\Lambda(\mathsf{UntypedTerms})$ and the set of closed untyped λ-terms modulo α-renaming, i.e.

syntax trees with binders, without β-conversion. Here are examples of encodings of the latter into the former (for the general definition, see Appendix A):

$$\begin{aligned}
\lambda x.\, x\, x &\quad\rightsquigarrow\quad \lambda(\ell : (\mathfrak{o} \Rightarrow \mathfrak{o}) \Rightarrow \mathfrak{o}).\, \lambda(a : \mathfrak{o} \Rightarrow \mathfrak{o} \Rightarrow \mathfrak{o}).\\
&\qquad\qquad \ell\ (\lambda(x : \mathfrak{o}).\, a\ x\ x)\\
(\lambda x.\, x\, x)\ (\lambda x.\, x\, x) &\quad\rightsquigarrow\quad \lambda(\ell : (\mathfrak{o} \Rightarrow \mathfrak{o}) \Rightarrow \mathfrak{o}).\, \lambda(a : \mathfrak{o} \Rightarrow \mathfrak{o} \Rightarrow \mathfrak{o}).\\
&\qquad\qquad a\ (\ell\ (\lambda(x : \mathfrak{o}).\, a\ x\ x))\ (\ell\ (\lambda(x : \mathfrak{o}).\, a\ x\ x))\\
\lambda f.\, (\lambda x.\, x\, x)\, (\lambda x.\, f\, (x\, x)) &\quad\rightsquigarrow\quad \lambda(\ell : (\mathfrak{o} \Rightarrow \mathfrak{o}) \Rightarrow \mathfrak{o}).\, \lambda(a : \mathfrak{o} \Rightarrow \mathfrak{o} \Rightarrow \mathfrak{o}).\\
&\qquad\qquad \ell\ (\lambda(f : \mathfrak{o}).\, a\ (\ell\ (\lambda(x : \mathfrak{o}).\, a\ x\ x))\\
&\qquad\qquad\qquad\qquad (\ell\ (\lambda(x : \mathfrak{o}).\, a\ f\ (a\ x\ x))))
\end{aligned}$$

This can be seen as an extension of Church encodings to higher-order abstract syntax: indeed, the variable ℓ plays the role of a constructor and introduces a bound variable.

A closed untyped term is *affine* if and only if every bound variable occurs at most once. We now give the outline of the proof, detailed in Appendix A, that the encodings in $\Lambda(\mathsf{UntypedTerms})$ of closed untyped affine terms form a **FinSet**-recognizable language.

Let Q be the finite set $\{0, 1, \infty\} \times \{\top, \bot\}$, where $\{0, 1, \infty\}$ is seen as the additive monoid $\mathbb{N}$ truncated to $2 = 3 = \cdots = \infty$. We consider the two set-theoretic functions $f_{\mathsf{app}} : Q \to (Q \Rightarrow Q)$ and $f_{\mathsf{abs}} : (Q \Rightarrow Q) \to Q$ defined as

$$f_{\mathsf{app}}(k, b)(k', b') \;:=\; (k + k',\ b \wedge b') \quad\text{and}\quad f_{\mathsf{abs}}(g) \;:=\; (g_1(0, \top),\ g_2(0, \top) \wedge (g_1(1, \top) \leq 1))$$

where $g_1 : Q \to \{0, 1, \infty\}$ and $g_2 : Q \to \{\top, \bot\}$ are the compositions of $g : Q \to Q$ with the two projections. We verify that, for any closed untyped term t, we have

$$[\![t]\!]_Q\, (f_{\mathsf{abs}})(f_{\mathsf{app}}) \quad=\quad (0, b) \qquad \text{where } b \text{ is } \top \text{ if and only if } t \text{ is affine.}$$

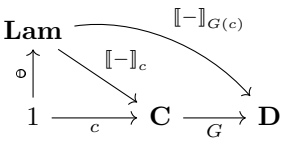
Therefore, if F is the subset of $[\![\mathsf{UntypedTerms}]\!]_Q$ defined as

$$F \;:=\; \{s \in ((Q \Rightarrow Q) \Rightarrow Q) \Rightarrow (Q \Rightarrow Q \Rightarrow Q) \Rightarrow Q \mid s(f_{\mathsf{abs}})(f_{\mathsf{app}}) = (0, \top)\}$$

then the **FinSet**-recognizable language L_F of terms of type $\mathsf{UntypedTerms}$ is the language of affine terms.

To the best of our knowledge, this regularity result is original. It also strongly suggests that the notion of **FinSet**-recognizable language, when applied to the type $\mathsf{UntypedTerms}$ of syntax trees with binders, differs from the recognizability of these syntax trees by the nominal tree automata of [13, §3.1]. Indeed, nominal automata cannot recognize the language of data words whose letters are all different [5, Proof of Lemma 5.4].

▶ Remark 2.5. Let $\mathbf{C}$ and $\mathbf{D}$ be CCCs and $G : \mathbf{C} \to \mathbf{D}$ be a CCC functor. By the universal property of **Lam**, see Equation (1), for every object c of $\mathbf{C}$, the following diagram commutes:



In particular, this means that for every simple type A, the objects $[\![A]\!]_{G(c)}$ and $G([\![A]\!]_c)$ are equal, and that for every simply typed λ-terms t and t' of type A,

$$\text{if}\quad [\![t]\!]_c \;=\; [\![t']\!]_c \qquad \text{then}\quad [\![t]\!]_{G(c)} \;=\; [\![t']\!]_{G(c)}\ .$$

This means that interpreting the simply typed λ-calculus at the object c will always be at least as fine as interpreting it at $G(c)$. In particular, this entails that $\mathrm{Rec}_{G(c)}(A) \subseteq \mathrm{Rec}_c(A)$. If G is moreover faithful, we then have that $\mathrm{Rec}_{G(c)}(A) = \mathrm{Rec}_c(A)$ for any object c of $\mathbf{C}$. Therefore, all languages which are $\mathbf{C}$-recognizable are $\mathbf{D}$-recognizable.

Recognizable languages of λ-terms, syntactically

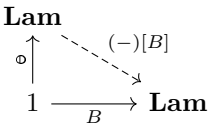
A syntactic approach to recognition is described in [14]. This syntactic approach uses the type substitution, also called cast, whose definition we now recall.

▶ **Definition 2.6.** *If A and B are simple types, we define a simple type $A[B] = A\{\mathfrak{o} := B\}$ by replacing every occurence of $\mathfrak{o}$ in A by B. We extend this to λ-terms by induction:*

$$x[B] := x \qquad (\lambda(x:A).t)[B] := \lambda(x:A[B]).t[B] \qquad (t\ u)[B] := t[B]\ u[B]$$
$$\langle t,u\rangle[B] := \langle t[B], u[B]\rangle \qquad t_i[B] := (t[B])_i$$

▷ Claim 2.7. For every simple types A and B and $t \in \Lambda(A)$, we have $t[B] \in \Lambda(A[B])$.

▶ Remark 2.8. A more categorical way to understand casting is to see it as the unique CCC functor $(-)[B] : \mathbf{Lam} \to \mathbf{Lam}$ such that the following diagram commutes:



As such, it is the semantic bracket functor $[\![-]\!]_B : \mathbf{Lam} \to \mathbf{Lam}$.

Finally, we recall the encoding of Booleans into the simply typed λ-calculus.

▶ **Definition 2.9.** *Let* Bool *be the simple type $\mathfrak{o}^2 \Rightarrow \mathfrak{o}$. Its two inhabitants are the λ-terms*

$$\mathsf{true} := \lambda(x:\mathfrak{o}^2).x_1 \qquad \textit{and} \qquad \mathsf{false} := \lambda(x:\mathfrak{o}^2).x_2\ .$$

The following definition naturally generalizes the one given in [14] for $A = \mathsf{Church}_n$.

▶ **Definition 2.10.** *For any simple type A, a language $L \subseteq \Lambda(A)$ is* syntactically regular *if there exists a simple type B and a λ-term r of type $A[B] \Rightarrow \mathsf{Bool}$ such that*

$$L = \{t \in \Lambda(A) \mid r\ t[B] =_{\beta\eta} \mathsf{true}\}\ .$$

▶ **Theorem 2.11** (Hillebrand & Kanellakis, [14, Theorem 3.4])**.** *A language of λ-terms of type Church_n is syntactically regular if and only if the associated language of finite words by the Church encoding is regular in the usual sense.*

▶ **Example 2.12.** The Church encodings of words in $\{1,2\}^*$ with an even number of 1s and an odd number of 2s is syntactically regular. Indeed, we consider the following λ-terms

$$\mathsf{and} := \lambda(p:\mathsf{Bool}\times\mathsf{Bool}).\lambda(x:\mathfrak{o}\times\mathfrak{o}).p_1\ \langle p_2\ x, x_2\rangle$$
$$\mathsf{id} := \lambda(b:\mathsf{Bool}).b$$
$$\mathsf{not} := \lambda(b:\mathsf{Bool}).\lambda(x:\mathfrak{o}\times\mathfrak{o}).b\ \langle x_2, x_1\rangle$$

and choose, as in Definition 2.10, the type B to be Bool and the simply typed λ-term r to be

$$\lambda(w:\mathsf{Church}_2[\mathsf{Bool}]).\mathsf{and}\ \langle w\ \mathsf{not}\ \mathsf{id}\ \mathsf{true},\ w\ \mathsf{id}\ \mathsf{not}\ \mathsf{false}\rangle \quad : \quad \mathsf{Church}_2[\mathsf{Bool}] \Rightarrow \mathsf{Bool}\ .$$

Just as in Example 2.3, this term can be seen as running two DFAs, both having two states, over the encoding of an input word.

When restricted to types Church_n for any natural number n, the notion of **FinSet**-recognizable and syntactically regular languages coincide, as they both boil down to the usual notion of regular language of finite words, as seen in Example 2.2 and Theorem 2.11 respectively. One of the contributions of the present paper is to show that these two notions coincide at every simple type.

3 Syntactic recognition implies semantic recognition

In this section, we state Theorem 3.2 and prove it by extending the semantic evaluation argument, described by Hillebrand and Kanellakis in their proof of Theorem 2.11 for the case of languages of words, to the more general case of languages of any type.

▶ **Definition 3.1.** *A CCC* $\mathbf{C}$ *is said to be* non-degenerate *if there exist two objects* c', c *of* $\mathbf{C}$ *such that there exist two distinct morphisms* $f, g \colon c' \to c$.

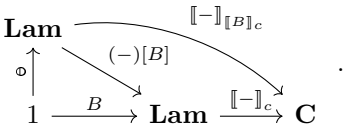
▶ **Theorem 3.2.** *If* $\mathbf{C}$ *is a non-degenerate CCC, then any language of λ-terms which is syntactically regular is* $\mathbf{C}$*-recognizable.*

Proof. We observe first that the non-degeneracy assumption means that the two projections $\pi_1, \pi_2 \colon c \times c \to c$ are not equal, since they yield different results when pre-composed with the morphism $c' \to c \times c$ obtained by pairing f and g.

Let A be a simple type and L be a language of λ-terms of type A which is syntactically regular. There exists a simple type B together with a λ-term $r \in \Lambda(A[B] \Rightarrow \mathsf{Bool})$ such that

$$L \quad = \quad \{t \in \Lambda(A) \mid r\ t[B] =_{\beta\eta} \mathsf{true}\}\ .$$

In order to show that L belongs to $\mathrm{Rec}_{[\![B]\!]_c}(A)$, we work with the interpretation of the simply typed λ-calculus at the object $[\![B]\!]_c$ of $\mathbf{C}$. By the universal property of the CCC **Lam**, the following diagram commutes



More concretely, this states that, for every simply typed λ-term t, the two morphisms $[\![t[B]]\!]_c$ and $[\![t]\!]_{[\![B]\!]_c}$ are equal. By viewing r as a morphism from $A[B]$ to Bool in the category **Lam**, the compositionality of the semantic interpretation gives us that

$$[\![r\ t[B]]\!]_c \ = \ [\![r]\!]_c \circ [\![t[B]]\!]_c \ = \ [\![r]\!]_c \circ [\![t]\!]_{[\![B]\!]_c}\ .$$

By non-degeneracy, the semantic interpretation $[\![-]\!]_c : \Lambda(\mathsf{Bool}) \to \mathbf{C}(1, [\![\mathsf{Bool}]\!]_c)$ is injective: indeed, $[\![\mathsf{true}]\!]_c = \pi_1 \neq \pi_2 = [\![\mathsf{false}]\!]_c$. Therefore, we get the following equivalences

$$t \in L \iff r\ t[B] =_{\beta\eta} \mathsf{true} \iff [\![r\ t[B]]\!]_c = [\![\mathsf{true}]\!]_c \iff [\![t]\!]_{[\![B]\!]_c} \in F$$

where F is the subset of $\mathbf{C}(1, [\![A]\!]_{[\![B]\!]_c})$ defined as $\{q \in \mathbf{C}(1, [\![A]\!]_{[\![B]\!]_c}) \mid [\![r]\!]_c \circ q = [\![\mathsf{true}]\!]_c\}$. This shows that L belongs to $\mathrm{Rec}_{[\![B]\!]_c}(A)$, hence that it is a $\mathbf{C}$-recognizable language. ◀

4 Logical relations and the squeezing construction

Sconing in a nutshell

In this paragraph, we recall the construction of a CCC of logical relations from one CCC to another. We first recall the construction of logical predicates, also called sconing, which will

be general enough to give logical relations as a special case. This method is well-known, see for instance [20] for an introductory account.

▶ **Definition 4.1.** *Let* $\mathbf{C}$ *be a CCC. The category of logical predicates over* $\mathbf{C}$*, that we denote by* $\mathbf{P}(\mathbf{C})$*, is defined as follows:*

- *its objects are the pairs* (c, S) *of an object* c *of* $\mathbf{C}$ *together with a subset* $S \subseteq \mathbf{C}(1, c)$*,*
- *its morphisms from* (c, S) *to* (c', S') *are the morphisms* $f : c \to c'$ *of* $\mathbf{C}$ *such that* $f \circ (-)$ *restricts to a set-theoretic function* $S \to S'$*.*

▷ Claim 4.2. This category $\mathbf{P}(\mathbf{C})$ is a CCC, with exponentiation given by

$$(c, S) \Rightarrow (c', S') \quad = \quad (c \Rightarrow c', \{f \in \mathbf{C}(1, c \Rightarrow c') \mid \forall s \in S, \mathrm{ev}_{c,c'} \circ \langle f, s\rangle \in S'\})$$

The forgetful functor $(c, S) \mapsto c$ is a CCC functor.

Proof. See [20, p. 5], where the notation $\tilde{\mathbf{C}}$ is used for the category $\mathbf{P}(\mathbf{C})$. ◁

▶ **Definition 4.3.** *Let* $\mathbf{C}_1$ *and* $\mathbf{C}_2$ *be two CCCs. The* CCC of logical relations *from* $\mathbf{C}_1$ *to* $\mathbf{C}_2$ *is the CCC* $\mathbf{P}(\mathbf{C}_1 \times \mathbf{C}_2)$*, which admits a CCC functor* $\mathbf{P}(\mathbf{C}_1 \times \mathbf{C}_2) \to \mathbf{C}_1 \times \mathbf{C}_2$*.*

▶ Remark 4.4. We have defined the CCC of logical relations in terms of the logical predicate construction $\mathbf{P}(-)$ of Definition 4.1. More concretely, this construction gives a category that can be described in the following way:

- its objects are triples $(c_1, c_2, \Vdash)$ where c_i is an object of $\mathbf{C}_i$ for $i = 1, 2$ and $\Vdash$ is a subset of $\mathbf{C}_1(1, c_1) \times \mathbf{C}_2(1, c_2)$, and is thus a relation between the points of c_1 and of c_2,
- its morphisms from $(c_1, c_2, \Vdash)$ to $(c_1', c_2', \Vdash')$ are pairs $(f_1, f_2) \in \mathbf{C}_1(c_1, c_1') \times \mathbf{C}_2(c_2, c_2')$ such that for every pair $(x_1, x_2) \in \mathbf{C}_1(1, c_1) \times \mathbf{C}_2(1, c_2)$,

$$\text{if } x_1 \Vdash x_2 \ , \quad \text{then } f_1 \circ x_1 \Vdash' f_2 \circ x_2 \ .$$

For the proof that this category is a CCC, see [20, Proposition 4.3].

▶ Remark 4.5. The CCC of logical relations from $\mathbf{C}_1$ to $\mathbf{C}_2$ comes with two projections to $\mathbf{C}_1$ and to $\mathbf{C}_2$ which are CCC functors. By Remark 2.5, we get that for any relation $(c_1, c_2, \Vdash)$,

$$[\![A]\!]_{(c_1,c_2,\Vdash)} \quad = \quad ([\![A]\!]_{c_1}, [\![A]\!]_{c_2}, \Vdash^A) \quad \text{for some } \Vdash^A \subseteq \ \mathbf{C}_1(1, [\![A]\!]_{c_1}) \times \mathbf{C}_2(1, [\![A]\!]_{c_2}).$$

The interpretation of a λ-term $t \in \Lambda(A)$ at an object $(c_1, c_2, \Vdash)$ is a morphism of the form

$$([\![t]\!]_{c_1}, [\![t]\!]_{c_2}) \quad : \quad 1 \longrightarrow [\![A]\!]_{(c_1,c_2,\Vdash)} \qquad \text{which means that} \qquad ([\![t]\!]_{c_1}, [\![t]\!]_{c_2}) \in \Vdash^A \ .$$

This is the *fundamental lemma of logical relations*, see e.g. [2, Lemma 4.5.3].

▶ Remark 4.6. At this stage, the categories $\mathbf{C}_1$ and $\mathbf{C}_2$ play a symmetric role. However, this will not always be the case in the rest of the paper, and we therefore say CCC of logical relations from $\mathbf{C}_1$ to $\mathbf{C}_2$ to emphasize the order.

The squeezing construction

We describe here a construction, which we call squeezing, which produces a CCC $\mathbf{Sqz}(\mathbf{C})$ from a CCC $\mathbf{C}$ equipped with an additional structure that we call a squeezing structure. Intuitively, the objects of $\mathbf{Sqz}(\mathbf{C})$ are objects of $\mathbf{C}$ coming with bounds induced by the structure, inspired by the squeeze theorem of calculus. This construction can be seen as the proof-irrelevant counterpart to the twisted gluing construction described in [1, Definition 5].

The notion of squeezing structure that is used is related to the hypotheses of [8, Lemma 6]; this pattern also occurs in the older proof theory tradition, see for instance [9, §8.A].

Throughout this paragraph, we fix any CCC $\mathbf{C}$. We recall that a wide subcategory is a subcategory containing all objects, and can hence be seen as a predicate on morphisms, closed under finite compositions.

▶ **Definition 4.7.** *A squeezing structure on $\mathbf{C}$ is the data of*

- *two wide subcategories $\mathbf{C}_{\text{left}}$ and $\mathbf{C}_{\text{right}}$ of $\mathbf{C}$ with associated notations $\xrightarrow{\text{l}}$ and $\xrightarrow{\text{r}}$ for morphisms, which are stable under finite cartesian products and such that for all $u : c_l \xrightarrow{\text{l}} c'_l$ and $v : c_r \xrightarrow{\text{r}} c'_r$,*

$$v \Rightarrow u \ : \ c'_r \Rightarrow c_l \xrightarrow{\text{l}} c_r \Rightarrow c'_l \qquad \text{and} \qquad u \Rightarrow v \ : \ c'_l \Rightarrow c_r \xrightarrow{\text{r}} c_l \Rightarrow c'_r \ .$$

- *for every object c of $\mathbf{C}$, two objects L_c and R_c of $\mathbf{C}$ such that there exists morphisms:*

$$\begin{array}{lll} L_1 \xrightarrow{\text{l}} 1 & L_{c\times c'} \xrightarrow{\text{l}} L_c \times L_{c'} & L_{c\Rightarrow c'} \xrightarrow{\text{l}} R_c \Rightarrow L_{c'} \\ 1 \xrightarrow{\text{r}} R_1 & R_c \times R_{c'} \xrightarrow{\text{r}} R_{c\times c'} & L_c \Rightarrow R_{c'} \xrightarrow{\text{r}} R_{c\Rightarrow c'} \ . \end{array} \tag{2}$$

▶ Remark 4.8. As we work in a proof-irrelevant setting, we are merely interested in the existence of these morphisms. Nonetheless, knowing that they belong to $\mathbf{C}_{\text{left}}$ or $\mathbf{C}_{\text{right}}$ gets us back some precious information, as we will see in Lemma 4.11 and Section 6.

▶ **Definition 4.9.** *Given a squeezing structure on $\mathbf{C}$, the category $\mathbf{Sqz}(\mathbf{C})$ is the full subcategory of $\mathbf{C}$ whose objects are the objects c of $\mathbf{C}$ for which there exist both a left morphism $L_c \xrightarrow{\text{l}} c$ and a right morphism $c \xrightarrow{\text{r}} R_c$.*

Notice that we write $\mathbf{Sqz}(\mathbf{C})$ even though this construction depends both on the CCC $\mathbf{C}$ and on a squeezing structure on $\mathbf{C}$.

▶ **Theorem 4.10.** *For a squeezing structure on $\mathbf{C}$, the category $\mathbf{Sqz}(\mathbf{C})$ is a sub-CCC of $\mathbf{C}$.*

Partial surjections

Logical relations which are partial surjections, i.e. both functional and surjective relations, can be a useful tool to obtain partial equivalence relations and to prove semantic results, see [6, §3], [7, §1.4.2] and [34, Theorem A]. We now show that the squeezing construction can be applied to get partial surjections for free.

▶ **Lemma 4.11.** *Let $\mathbf{C}_1$ and $\mathbf{C}_2$ be two CCCs and $\mathbf{R}$ be the CCC of logical relations from $\mathbf{C}_1$ to $\mathbf{C}_2$, whose objects are triples containing relations. Suppose that we are given a squeezing structure on $\mathbf{R}$ such that*

- *the relations in the objects L_c are surjective,*
- *the relations in the objects R_c are functional,*
- *the morphisms (u_1, u_2) in $\mathbf{R}_{\text{left}}$ are such that $\mathbf{C}_2(1, u_2)$ is a surjective function,*
- *the morphisms (v_1, v_2) in $\mathbf{R}_{\text{right}}$ are such that $\mathbf{C}_2(1, v_2)$ is an injective function,*

where, for $u \in \mathbf{C}(a, b)$, the function $\mathbf{C}(1, u) : \mathbf{C}(1, a) \to \mathbf{C}(1, b)$ is the composition $u \circ (-)$.

Then, the relation of any object belonging to $\mathbf{Sqz}(\mathbf{R})$ is a partial surjection.

We end this section by giving a definition which will appear in squeezing structures in Proposition 6.4 and in the proof of Proposition 5.8.

▶ **Definition 4.12.** *Let* $\mathbf{R}$ *be the CCC of logical relations from* $\mathbf{C}_1$ *to* $\mathbf{C}_2$*. We say that a morphism* $(f_1, f_2) : (c_1, c_2, \Vdash) \to (c'_1, c'_2, \Vdash')$ *is a* target-identity *if* c_2 *and* c'_2 *are the same object and if* f_2 *is the identity morphism.*

▶ Remark 4.13. We remark that the target-identities form a wide subcategory and are stable under products and exponentiation.

5 From well-pointed locally finite CCCs to finite sets

We now recall the definition of the class of CCCs $\mathbf{C}$ for which we will show that $\mathbf{C}$-recognizable languages coincide with our other definitions of regular languages of λ-terms. Recall that in a CCC $\mathbf{C}$, a *point* of an object c is a morphism $1 \to c$ from the terminal object to c.

▶ **Definition 5.1.** *A CCC is said to be:*

- well-pointed *if every morphism is determined by its action on points of its domain;*
- locally finite *if all its hom-sets are finite sets.*

We start by introducing the following constructions, which will help us to use partial surjections.

▶ **Definition 5.2.** *Let* $\mathbf{C}$ *be a category with a terminal object. We define the following full subcategories of* $\mathbf{C}$*:*

- $\mathbf{C}_{\geq 1}$ *containing the objects* c *that have at least one point; we call these objects* inhabited*,*
- $\mathbf{C}_{\leq 1}$ *containing the objects* c *that have at most one point,*
- $\mathbf{C}_{=1}$ *containing the objects* c *that have exactly one point.*

When instantiated to the CCC **Lam** of simple types and λ-terms, the notion of inhabited object coincides with the usual notion of inhabited simple type.

▶ **Proposition 5.3.** *If* $\mathbf{C}$ *is a CCC, then the category* $\mathbf{C}_{\geq 1}$ *is a sub-CCC of* $\mathbf{C}$*.*

▶ **Proposition 5.4.** *If* $\mathbf{E}$ *is a well-pointed CCC, then the category* $\mathbf{E}_{\leq 1}$ *is a sub-CCC of* $\mathbf{E}$*. Moreover, the category* $\mathbf{E}_{=1}$ *is equivalent to the terminal category.*

We now prove the interesting fact that inhabited objects characterize the recognized languages of λ-terms in a well-pointed CCC.

▶ **Proposition 5.5.** *If* $\mathbf{E}$ *is a well-pointed CCC, then a language of* λ*-terms is* $\mathbf{E}$*-recognizable if and only if it is* $\mathbf{E}_{\geq 1}$*-recognizable.*

Proof. Let $\mathbf{E}$ be a well-pointed CCC. By Proposition 5.3, $\mathbf{E}_{\geq 1}$ is a sub-CCC of $\mathbf{E}$ so any language which is $\mathbf{E}_{\geq 1}$-recognizable is $\mathbf{E}$-recognizable, as explained in Remark 2.5.

We now show that the only language recognized by $\mathbf{E}_{\leq 1}$ is the empty and full languages. Let A be a simple type and c be an object of $\mathbf{E}_{\leq 1}$. As $\mathbf{E}$ is well-pointed and by Proposition 5.4, we know that $[\![A]\!]_c$ belongs to $\mathbf{E}_{\leq 1}$, which means that $\mathbf{C}(1, [\![A]\!]_c)$ is empty or a singleton, so $\mathrm{Rec}_c(A)$ contains at most the empty and full languages, which are the same when A is not inhabited.

The empty and full languages are recognized by any CCC, so in particular by $\mathbf{E}_{\geq 1}$. Therefore, all $\mathbf{E}$-recognizable languages are $\mathbf{E}_{\geq 1}$-recognizable. ◀

▶ **Example 5.6.** The category $\mathbf{FinSet}_{\geq 1}$ is the CCC of *non-empty* finite sets.

- An implicative semilattice is a meet-semilattice such that each meet operation has an upper adjoint. Implicative semilattices are CCCs, however, they are degenerate and so never distinguish different λ-terms of the same type.
  Another way to understand this fact is to remark that their full subcategory of inhabited objects is the terminal category.

Next, we introduce the basic partial relations at which we will interpret the λ-calculus.

▶ **Definition 5.7.** *Let* $\mathbf{E}$ *be a well-pointed locally finite CCC and* $\mathbf{R}$ *be the CCC of logical relations from* **FinSet** *to* $\mathbf{E}$*. For any object* e *of* $\mathbf{E}$*, we consider the triple*

$$T_e \quad := \quad (\mathbf{E}(1,e), e, \sim_e) \qquad \text{where } \sim_e \text{ is the identity relation of } \mathbf{E}(1,e).$$

which extends to a functor $T : \mathbf{E} \to \mathbf{R}$*.*

We now prove a converse to Lemma 4.11 in the present case.

▶ **Proposition 5.8.** *Let* $\mathbf{E}$ *be a well-pointed locally finite CCC whose objects are all inhabited and* $\mathbf{R}$ *be the CCC of logical relations from* **FinSet** *to* $\mathbf{E}$*. Then, the full subcategory of partial surjections is a sub-CCC of* $\mathbf{R}$*.*

The proof is in Appendix B and uses a squeezing structure.

▶ **Theorem 5.9** (claimed in [28, Lemma 20])**.** *For every well-pointed locally finite CCC* $\mathbf{E}$*, any* $\mathbf{E}$*-recognizable language is* **FinSet***-recognizable.*

Proof. Let A be a simple type and L be a language of λ-terms of type A which is $\mathbf{E}$-recognizable. By Proposition 5.5, it is $\mathbf{E}_{\geq 1}$-recognizable. Let e be an object of $\mathbf{E}_{\geq 1}$ such that $L \in \mathrm{Rec}_e(A)$. We consider the object $T_e = (\mathbf{E}(1,e), e, \sim_e)$ from Definition 5.7, whose relation is a partial surjection. The object $[\![A]\!]_{T_e}$ is of the form $([\![A]\!]_{\mathbf{E}(1,e)}, [\![A]\!]_e, \sim_e^A)$, where the relation $\sim_e^A$ is a partial surjection, as explained in Remark 2.5.

Let F be a subset of $\mathbf{E}(1, [\![A]\!]_e)$ such that L is L_F. We consider the subset F' of $[\![A]\!]_{\mathbf{E}(1,e)}$ defined as the inverse image

$$F' \quad := \quad \{q \in [\![A]\!]_{\mathbf{E}(1,e)} \mid \exists q' \in F \text{ s.t. } q \sim_e^A q'\}$$

By the fundamental lemma of logical relations, for λ-term t of type A, we have

$$[\![t]\!]_{\mathbf{E}(1,e)} \sim_e^A [\![t]\!]_e \ .$$

which proves that $L_F \subseteq L_{F'}$. Moreover, as $\sim_e^A$ is a functional relation, we get the converse inclusion. This proves that L is **FinSet**-recognizable. ◀

6 From finite sets to λ-terms

In this section, we apply the squeezing construction of Definition 4.9 on a CCC of logical relations to show that every **FinSet**-recognizable language is syntactically regular, through an encoding of finite sets into the simply typed λ-calculus. To achieve that, we need to change slightly of setting, by moving from finite sets to finite ordinals. This will make it possible to define the functor $\mathrm{Fin}(-)$ without ambiguity.

Therefore, we consider the category **FinOrd** whose objects are natural numbers and whose morphisms are the set-theoretic maps between the associated finite cardinals $\langle n \rangle := \{1, \dots, n\}$. The inclusion of **FinOrd** in **FinSet** is a fully faithful functor that is essentially surjective, henceforth we get an equivalence between **FinOrd** and **FinSet** using the axiom of choice. In particular, **FinOrd** is a CCC that recognizes the same languages as **FinSet**.

The encoding of finite sets and its squeezing structure

We take $\mathbf{R}$ to be the CCC of logical relations from **Lam** to **FinOrd**. We recall that the objects of $\mathbf{R}$ are therefore triples $R = (B, n, \Vdash)$ where B is a simple type, n is a natural number and $\Vdash$ is a subset of the product $\Lambda(B) \times \langle n \rangle$. A morphism of $\mathbf{R}$ is a pair of morphisms of **Lam** and **FinOrd** which respect the relations.

▶ **Definition 6.1.** *We define the functor* $\mathrm{Fin}(-) : \mathbf{FinOrd} \to \mathbf{Lam}$ *as* $\mathrm{Fin}(n) := o^n \Rightarrow o$ *and, for every* $f : n \to n'$*, the* λ*-term* $\mathrm{Fin}(f) : \mathrm{Fin}(n) \to \mathrm{Fin}(n')$ *is*

$$\lambda(p : \mathrm{Fin}(n)).\, \lambda(x : o^{n'}).\, p\ \langle x_{f(1)}, \ldots, x_{f(n)} \rangle\ .$$

▶ Remark 6.2. As **FinOrd** is equivalent to the free cocartesian category and $\mathbf{Lam}^{\mathrm{op}}$ is cocartesian, we get a functor $n \mapsto o^n$. The composition of the two functors

$$\mathbf{FinOrd} \xrightarrow{n \mapsto o^n} \mathbf{Lam}^{\mathrm{op}} \xrightarrow{(-) \Rightarrow o} \mathbf{Lam}$$

is precisely the functor $\mathrm{Fin}(-)$.

Our goal is now to exhibit a squeezing structure on $\mathbf{R}$ in order to show Theorem 6.5. We consider the target-identities for the two wide subcategories of the structure. We now define the following family of objects.

▶ **Definition 6.3.** *For any natural number* n*, we define the object* Bij_n *as*

$$\mathrm{Bij}_n \quad := \quad (\mathrm{Fin}(n), n, \Vdash_n)$$

where $\Vdash_n$ *is the bijection between the sets* $\Lambda(\mathrm{Fin}(n))$ *and* $\langle n \rangle$ *defined as*

$$\Vdash_n \quad := \quad \{(\pi_i, i) : 1 \leq i \leq n\} \qquad \text{with } \pi_i \text{ the } \lambda\text{-term } \lambda(x : o^n).\, x_i\ .$$

This assignment extends to a functor $\mathrm{Bij}_{(-)} : \mathbf{FinOrd} \to \mathbf{R}$.

▶ **Proposition 6.4.** *There is a squeezing structure on the CCC* $\mathbf{R}$ *such that:*

- *the left and right morphisms are the target-identities of* $\mathbf{R}$*,*
- *for any object* $c = (B, n, \Vdash)$ *of* $\mathbf{R}$*, the objects* L_c *and* R_c *are both equal to* Bij_n*.*

The proof is in Appendix B. Proposition 6.4 shows that we have a sub-CCC $\mathbf{Sqz}(\mathbf{C})$ of the CCC of logical relations from **Lam** to **FinOrd**, whose objects are tuples $(B, n, \Vdash)$ such that there exists λ-terms $u : \mathrm{Fin}(n) \to B$ and $v : B \to \mathrm{Fin}(n)$ lifting to the two following target-identities:

$$(\mathrm{Fin}(n), n, \Vdash_n) \xrightarrow{(u, \mathrm{Id}_n)} (B, n, \Vdash) \quad \text{and} \quad (B, n, \Vdash) \xrightarrow{(v, \mathrm{Id}_n)} (\mathrm{Fin}(n), n, \Vdash_n)\ .$$

Encoding recognizability by finite sets

We have shown in Proposition 6.4 that we have a squeezing structure on the CCC of logical relations $\mathbf{R}$ from **Lam** to **FinOrd**. We now show how to use this structure, culminating in the link established in Theorem 6.5 between **FinOrd**-recognizable and syntactically regular languages.

▶ **Theorem 6.5.** *If a language is* **FinSet***-recognizable, then it is syntactically regular.*

Proof. Let A be a simple type and $L \subseteq \Lambda(A)$ be any **FinSet**-recognizable language. There exists a finite set Q and a subset $F \subseteq \langle [\![A]\!]_n \rangle$ such that

$$L \;=\; \{t \in \Lambda(A) \mid [\![t]\!]_n \in F\} \;.$$

We take n to be the cardinality of Q and note $\chi : [\![A]\!]_n \to 2$ the characteristic function associated to the subset F. By applying the functor $\mathrm{Bij}_{(-)}$, we get a morphism of relations

$$\mathrm{Bij}_\chi := (\mathrm{Fin}(\chi)\,,\,\chi) \quad : \quad \Big(\mathrm{Fin}([\![A]\!]_n), [\![A]\!]_n, \Vdash_{[\![A]\!]_n}\Big) \;\longrightarrow\; (\mathrm{Fin}(2), 2, \Vdash_2) \;.$$

The interpretation $[\![A]\!]_{\mathrm{Bij}_n}$ is of the form $(A[\mathrm{Fin}(n)], [\![A]\!]_n, \Vdash_n^A)$ as explained in Remark 2.5. As it is an object of $\mathbf{Sqz}(\mathbf{R})$, it has a target-identity into $\mathrm{Bij}_{[\![A]\!]_n}$. By composing this morphism with Bij_χ, we obtain a morphism

$$(r\,,\,\chi) \quad : \quad (A[\mathrm{Fin}(n)], [\![A]\!]_n, \Vdash_n^A) \;\longrightarrow\; (\mathrm{Fin}(2), 2, \Vdash_2) \;.$$

By the fundamental lemma of logical relations, we get that, for every λ-term t of type A,

$$t[\mathrm{Fin}(n)] \Vdash_n^A [\![t]\!]_n \qquad \text{on which we apply } (r,\chi) \text{ to get} \qquad r\; t[\mathrm{Fin}(n)] \Vdash_2 \chi([\![t]\!]_n)$$

which states that $r\; t[\mathrm{Fin}(n)] =_{\beta\eta} \mathsf{true}$ if and only if $\chi([\![t]\!]_n)$ is 1. This proves that r recognizes the language L_F given by $F \subseteq \langle [\![A]\!]_n \rangle$, and so that L is syntactically regular. ◀

7 Regular languages

In this section, we want to point out a few consequences of the equivalence previously proved through Theorem 3.2, Theorem 5.9 and Theorem 6.5. Using these theorems, the following definition of regular languages is well-defined.

▶ **Definition 7.1.** *Let A be a simple type. A regular language of λ-terms of type A is a subset $L \subseteq \Lambda(A)$ such that one of the following equivalent propositions holds:*

- *L is syntactically regular;*
- *L is $\mathbf{C}$-recognizable, for some non-degenerate, well-pointed and locally finite CCC $\mathbf{C}$;*
- *L is **FinSet**-recognizable.*

We denote by $\mathrm{Reg}(A)$ the set of regular languages of λ-terms of type A.

▶ Remark 7.2. Note that **FinSet** recognizes all the regular languages of λ-terms. In that sense, it plays the same role as the monoid

$$M \;:=\; \{f : \mathbb{N} \to \mathbb{N} \mid \exists N \in \mathbb{N}, \forall n \geq N, f(n) = n\} \quad \text{with the composition of functions}$$

which recognizes all the regular languages of finite words as all finite monoids can be embedded into M. Such a monoid cannot be finite; however, M is a *locally finite monoid*, i.e. all its finitely generated submonoids are finite (this is a standard notion, see e.g. [11, §V.5]).

In the case of finite words, recognizability by finite monoids and locally finite monoids are equivalent when the alphabet is finite. In the case of λ-terms however, finite CCCs are all degenerate whereas the locally finite case yields regular languages of λ-terms, with some additional conditions.

▶ **Proposition 7.3.** *The set $\mathrm{Reg}(A)$ of regular languages of λ-terms of some simple type A is a Boolean algebra.*

This fact boils down to stability by union or intersection. It is proved in [27, Theorem 8] using intersection types and in [34, Proposition 2.5] using logical relations. We provide another proof, showing that it is a direct consequence of our results.

Proof. Using any of the three conditions of Definition 7.1, it is clear that regular languages are closed under complement.

The product CCC **FinSet** × **FinSet** is non-degenerate, well-pointed and locally finite. It comes with two projections which are both CCC functors **FinSet** × **FinSet** → **FinSet**. Let Q and Q' be two finite sets; we consider the object (c, c') of **FinSet** × **FinSet**. As explained in Remark 2.5, for any simple type A, we get that $\mathrm{Rec}_Q(A) \subseteq \mathrm{Rec}_{(Q,Q')}(A)$ and $\mathrm{Rec}_{Q'}(A) \subseteq \mathrm{Rec}_{(Q,Q')}(A)$.

Moreover, $\mathrm{Rec}_{(Q,Q')}(A)$ is a Boolean algebra. This shows that the intersection of a language in $\mathrm{Rec}_Q(A)$ with another in $\mathrm{Rec}_{Q'}(A)$ can be taken in $\mathrm{Rec}_{(Q,Q')}(A)$, so it is still a regular language. Therefore, $\mathrm{Reg}(A)$ is a Boolean algebra. ◀

We now point out two other consequences of the equivalence in Definition 7.1.

- As stated in the introduction, Statman's finite completeness theorem tells us that singleton languages of λ-terms, taken modulo $\beta\eta$-conversion, are regular languages. It has multiple proofs, see [32] for proof directly in the finite standard model, [27] using intersection types, [17] in the model of complete lattices and [29] using Böhm trees.
  These results are usually proved in one CCC. Using Theorem 3.2, Theorem 5.9 and Theorem 6.5, we get that the singleton languages are recognized by any non-degenerate, well-pointed and locally finite CCC and are also syntactically regular.
- Some CCCs satisfying these three conditions are the coKleisli categories of a model of linear logic, see [18]. In [19], a notion of higher-order automaton is presented, which recognizes a language of λ-terms of a given simple type. The run-trees for these non-deterministic automata are defined using an intersection type system, which is an equivalent way of presenting the semantic interpretation $[\![-]\!]$ of the simply typed λ-calculus in the coKleisli category $\mathbf{ScottL}_!$ of the Scott model of linear logic. Using the equivalence proved in the present article, a language is recognized by a higher-order automaton, i.e. $\mathbf{ScottL}_!$-recognizable, if and only if it satisfies one of the conditions of Definition 7.1.

8 Conclusion and future perspectives

In this article, we have shown that every non-degenerate, well-pointed and locally finite CCCs recognizes exactly Salvati's regular languages of λ-terms [27], and that those also coincide with syntactically regular languages. This is evidence for the robustness of this notion, and therefore of the dual notion of profinite λ-term introduced in [34].

Among the aforementioned conditions, non-degeneracy is needed to recognize non-trivial languages, and local finiteness is clearly crucial: in the case of finite words and trees, regularity is closely related to recognition by finitary structures. What about well-pointedness? In other words, one question that remains open is the following: is there a locally finite CCC that recognizes languages of λ-terms that are not regular, i.e. not recognizable by **FinSet**? For example, sequential algorithms famously form a locally finite CCC which is *not* well-pointed, cf. [2, Chapter 14]; we would like to understand its recognition power.

As explained in Example 2.2, the regular languages of λ-terms of type Church_n for some natural number n are exactly the usual regular languages of the finite words associated by the Church encoding. It is possible to encode words in other calculi, like the non-commutative affine λ-calculus in which case a syntactic approach analogous to Definition 2.10 yields the

star-free languages, see [22]. Moreover, gluing techniques have been studied for other calculi, see [12] for the linear case. One can therefore wonder whether it is possible to develop a semantic approach *à la* Salvati, analogous to Definition 2.1, for other calculi.

References

1 Thorsten Altenkirch, Martin Hofmann, and Thomas Streicher. Categorical reconstruction of a reduction free normalization proof. In David H. Pitt, David E. Rydeheard, and Peter T. Johnstone, editors, *Category Theory and Computer Science, 6th International Conference, CTCS '95, Cambridge, UK, August 7-11, 1995, Proceedings*, volume 953 of *Lecture Notes in Computer Science*, pages 182–199. Springer, 1995. doi:10.1007/3-540-60164-3_27.

2 Roberto M. Amadio and Pierre-Louis Curien. *Domains and Lambda-Calculi*, volume 46 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1998. doi:10.1017/CBO9780511983504.

3 Robert Atkey. Syntax for free: Representing syntax with binding using parametricity. In Pierre-Louis Curien, editor, *Typed Lambda Calculi and Applications, 9th International Conference, TLCA 2009, Brasilia, Brazil, July 1-3, 2009. Proceedings*, volume 5608 of *Lecture Notes in Computer Science*, pages 35–49. Springer, 2009. doi:10.1007/978-3-642-02273-9_5.

4 Mikołaj Bojańczyk. Languages recognised by finite semigroups, and their generalisations to objects such as trees and graphs, with an emphasis on definability in monadic second-order logic. *CoRR*, abs/2008.11635, 2020. arXiv:2008.11635.

5 Mikołaj Bojańczyk, Bartek Klin, and Sławomir Lasota. Automata theory in nominal sets. *Logical Methods in Computer Science*, 10(3), 2014. doi:10.2168/LMCS-10(3:4)2014.

6 Antonio Bucciarelli. Logical relations and lambda theories. In *Advances in Theory and Formal Methods of Computing, proceedings of the 3rd Imperial College Workshop*, pages 37–48, 1996.

7 Thomas Ehrhard. The Scott model of linear logic is the extensional collapse of its relational model. *Theoretical Computer Science*, 424:20–45, 2012. doi:10.1016/j.tcs.2011.11.027.

8 Marcelo Fiore. Semantic analysis of normalisation by evaluation for typed lambda calculus. *Mathematical Structures in Computer Science*, 32(8):1028–1065, 2022. doi:10.1017/S0960129522000263.

9 Jean-Yves Girard. *The Blind Spot: Lectures on logic*. European Mathematical Society, September 2011. doi:10.4171/088.

10 Charles Grellois. *Semantics of linear logic and higher-order model-checking*. PhD thesis, Université Paris 7, April 2016. URL: https://hal.science/tel-01311150.

11 Pierre A. Grillet. *Semigroups. An introduction to the structure theory*. Chapman & Hall/CRC Pure and Applied Mathematics. Dekker, 1995. doi:10.4324/9780203739938.

12 Masahito Hasegawa. Logical predicates for intuitionistic linear type theories. In Jean-Yves Girard, editor, *Typed Lambda Calculi and Applications, 4th International Conference, TLCA'99, L'Aquila, Italy, April 7-9, 1999, Proceedings*, volume 1581 of *Lecture Notes in Computer Science*, pages 198–212. Springer, 1999. doi:10.1007/3-540-48959-2_15.

13 Gerco van Heerdt, Tobias Kappé, Jurriaan Rot, Matteo Sammartino, and Alexandra Silva. Tree Automata as Algebras: Minimisation and Determinisation. In Markus Roggenbach and Ana Sokolova, editors, *8th Conference on Algebra and Coalgebra in Computer Science (CALCO 2019)*, volume 139 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 6:1–6:22, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.CALCO.2019.6.

14 Gerd G. Hillebrand and Paris C. Kanellakis. On the expressive power of simply typed and let-polymorphic lambda calculi. In *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996*, pages 253–263. IEEE Computer Society, 1996. doi:10.1109/LICS.1996.561337.

15 Naoki Kobayashi. Model checking higher-order programs. *Journal of the ACM*, 60(3):20:1–20:62, 2013. doi:10.1145/2487241.2487246.

16 Naoki Kobayashi. 10 years of the higher-order model checking project (extended abstract). In Ekaterina Komendantskaya, editor, *Proceedings of the 21st International Symposium on Principles and Practice of Programming Languages, PPDP 2019, Porto, Portugal, October 7-9, 2019*, pages 2:1–2:2. ACM, 2019. doi:10.1145/3354166.3354167.

17 Gregory M. Kobele and Sylvain Salvati. The IO and OI hierarchies revisited. *Information and Computation*, 243:205–221, 2015. doi:10.1016/j.ic.2014.12.015.

18 Paul-André Melliès. Categorical Semantics of Linear Logic. In P.-L. Curien, H. Herbelin, J.-L. Krivine, and P.-A. Melliès, editors, *Interactive models of computation and program behaviour*, volume 27 of *Panoramas et Synthèses*. Société Mathématique de France, 2009. URL: https://smf.emath.fr/publications/semantique-categorielle-de-la-logique-lineaire.

19 Paul-André Melliès. Higher-order parity automata. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–12, Reykjavik, Iceland, June 2017. IEEE. doi:10.1109/LICS.2017.8005077.

20 John C. Mitchell and Andre Scedrov. Notes on sconing and relators. In Egon Börger, Gerhard Jäger, Hans Kleine Büning, Simone Martini, and Michael M. Richter, editors, *Computer Science Logic, 6th Workshop, CSL '92, San Miniato, Italy, September 28 - October 2, 1992, Selected Papers*, volume 702 of *Lecture Notes in Computer Science*, pages 352–378. Springer, 1992. doi:10.1007/3-540-56992-8_21.

21 Lê Thành Dũng Nguyễn. *Implicit automata in linear logic and categorical transducer theory*. PhD thesis, Université Paris XIII, 2021. URL: https://hal.science/tel-04132636.

22 Lê Thành Dũng Nguyễn and Cécilia Pradic. Implicit automata in typed λ-calculi I: aperiodicity in a non-commutative logic. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 135:1–135:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.135.

23 C.-H. Luke Ong. On model-checking trees generated by higher-order recursion schemes. In *21th IEEE Symposium on Logic in Computer Science (LICS 2006), 12-15 August 2006, Seattle, WA, USA, Proceedings*, pages 81–90. IEEE Computer Society, 2006. doi:10.1109/LICS.2006.38.

24 C.-H. Luke Ong. Higher-order model checking: An overview. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015*, pages 1–15. IEEE Computer Society, 2015. doi:10.1109/LICS.2015.9.

25 Gordon D. Plotkin. Recursion does not always help. In Fairouz Kamareddine, editor, *A Century since Principia's Substitution Bedazzled Haskell Curry. In Honour of Jonathan Seldin's 80th Anniversary*. College Publications, 2023. arXiv:2206.08413.

26 Simona Ronchi Della Rocca. Intersection Types and Denotational Semantics: An Extended Abstract (Invited Paper). In Silvia Ghilezan, Herman Geuvers, and Jelena Ivetić, editors, *22nd International Conference on Types for Proofs and Programs (TYPES 2016)*, volume 97 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 2:1–2:7, Dagstuhl, Germany, 2018. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.TYPES.2016.2.

27 Sylvain Salvati. Recognizability in the simply typed lambda-calculus. In Hiroakira Ono, Makoto Kanazawa, and Ruy J. G. B. de Queiroz, editors, *Logic, Language, Information and Computation, 16th International Workshop, WoLLIC 2009, Tokyo, Japan, June 21-24, 2009. Proceedings*, volume 5514 of *Lecture Notes in Computer Science*, pages 48–60. Springer, 2009. doi:10.1007/978-3-642-02261-6_5.

28 Sylvain Salvati. *Lambda-calculus and formal language theory*. Habilitation à diriger des recherches, Université de Bordeaux, 2015. URL: https://hal.science/tel-01253426.

29 B. Srivathsan and Igor Walukiewicz. An alternate proof of Statman's finite completeness theorem. *Information Processing Letters*, 112(14-15):612–616, 2012. doi:10.1016/j.ipl.2012.04.014.

30 Richard Statman. Completeness, invariance and lambda-definability. *Journal of Symbolic Logic*, 47(1):17–26, 1982. doi:10.2307/2273377.

31 Richard Statman. On the λY calculus. *Annals of Pure and Applied Logic*, 130(1-3):325–337, 2004. doi:10.1016/j.apal.2004.04.004.

32 Richard Statman and Gilles Dowek. On Statman's finite completeness theorem, 1992. arXiv:2309.03602.

33 Kazushige Terui. Semantic Evaluation, Intersection Types and Complexity of Simply Typed Lambda Calculus. In *23rd International Conference on Rewriting Techniques and Applications (RTA'12)*, pages 323–338, 2012. doi:10.4230/LIPIcs.RTA.2012.323.

34 Sam van Gool, Paul-André Melliès, and Vincent Moreau. Profinite lambda-terms and parametricity. *Electronic Notes in Theoretical Informatics and Computer Science*, Volume 3 – Proceedings of MFPS XXXIX, November 2023. doi:10.46298/entics.12280.

35 Igor Walukiewicz. Automata theory and higher-order model-checking. *ACM SIGLOG News*, 3(4):13–31, 2016. doi:10.1145/3026744.3026745.

A The regular language of affine untyped terms

The goal of this appendix is to provide a detailed explaination of Example 2.4. We first introduce a grammar for the simply typed λ-terms of type UntypedTerms through the following notion of scoped term.

▶ **Definition A.1.** *We consider untyped terms with de Bruijn indices, given by the grammar*

$$u, v \quad ::= \quad \mathtt{var}_i \text{ for } i \in \mathbb{N}^* \mid \mathtt{abs}(u) \mid \mathtt{app}(u, v)$$

where we write $\mathtt{abs}(-)$ *for the abstraction to distinguish it from the simply typed* λ*-abstraction.*

For any natural number n *and untyped term* u*, we define the judgment* $n \vdash u$ *by induction with the rules*

$$\frac{}{n \vdash \mathtt{var}_i}\ 1 \leq i \leq n \qquad \frac{n+1 \vdash u}{n \vdash \mathtt{abs}(u)} \qquad \frac{n \vdash u \qquad n \vdash v}{n \vdash \mathtt{app}(u, v)}\ .$$

The judgment $n \vdash u$ *has at most one derivation. We call a scoped term any pair of* n *and* u *such that* $n \vdash u$ *is derivable.*

In the rest of the appendix, we will simply say that $n \vdash u$ is a scoped term whenever this judgment is derivable. We now give the encoding of scoped terms into the simply typed λ-calculus.

▶ **Definition A.2.** *Let us consider a fixed sequence of simply typed variables* $x_k : \mathfrak{o}$ *for* $k \in \mathbb{N}$*. We define the context* Γ_n *as* $\ell : (\mathfrak{o} \Rightarrow \mathfrak{o}) \Rightarrow \mathfrak{o}, a : \mathfrak{o} \Rightarrow \mathfrak{o} \Rightarrow \mathfrak{o}, x_1 : \mathfrak{o}, \dots, x_n : \mathfrak{o}$*.*

For any natural number n *and untyped term* u*, we consider the encoding* $\overline{n \vdash u}$ *of a scoped term defined by induction as*

$$\begin{aligned}
\overline{n \vdash \mathtt{var}_i} &:= x_{n+1-i} \\
\overline{n \vdash \mathtt{abs}(u)} &:= \ell\ (\lambda(x_{n+1} : \mathfrak{o}).\ \overline{n+1 \vdash u}) \\
\overline{n \vdash \mathtt{app}(u, v)} &:= a\ (\overline{n \vdash u})\ (\overline{n \vdash v})
\end{aligned}$$

which is such that $\Gamma_n \vdash \overline{n \vdash u} : \mathfrak{o}$*.*

Using normalization for the simply typed λ-calculus, we can claim the following fact.

▷ Claim A.3. For every natural number n, the encoding of scoped terms $n \vdash u$ into the open λ-terms $\overline{n \vdash u}$ of type $\mathfrak{o}$ in context Γ_n, taken modulo $\beta\eta$-conversion, is bijective.

In particular, the encoding induces a bijection between the set Λ(UntypedTerms) and the set of closed untyped terms, i.e. the u such that $0 \vdash u$ is a scoped term.

▶ **Definition A.4.** *For any scoped term $n \vdash u$ and $1 \leq i \leq n$, we define the natural number $\mathtt{occ}_i(n \vdash u)$, the number of occurences of the i^{th} variable, by induction as:*

$$\begin{aligned}
\mathtt{occ}_i(n \vdash \mathtt{var}_j) &:= 1 \text{ if } i = j \text{ otherwise } 0 \\
\mathtt{occ}_i(n \vdash \mathtt{abs}(u)) &:= \mathtt{occ}_{i+1}(n+1 \vdash u) \\
\mathtt{occ}_i(n \vdash \mathtt{app}(u,v)) &:= \mathtt{occ}_i(n \vdash u) + \mathtt{occ}_i(n \vdash v)\ .
\end{aligned}$$

When the scoped term $n \vdash u$ is clear from context, we will simply write $\mathtt{occ}_i$.

▶ **Definition A.5.** *For any finite set Q and functions $f_{\mathtt{abs}} : (Q \Rightarrow Q) \to Q$ and $f_{\mathtt{app}} : Q \to Q \Rightarrow Q$, we interpret any scoped term $n \vdash u$ as its semantics*

$$n \vdash u \qquad \rightsquigarrow \qquad (\!|n \vdash u|\!) \quad : \quad Q^n \longrightarrow Q$$

which is the set-theoretic function defined, for $q_1, \dots, q_n \in Q$, by induction as:

$$\begin{aligned}
(\!|n \vdash \mathtt{var}_i|\!)[q_1, \dots, q_n] &:= q_i \\
(\!|n \vdash \mathtt{abs}(u)|\!)[q_1, \dots, q_n] &:= f_{\mathtt{abs}}(q \mapsto (\!|n+1 \vdash u|\!)[q, q_1, \dots, q_n]) \\
(\!|n \vdash \mathtt{app}(u,v)|\!)[q_1, \dots, q_n] &:= f_{\mathtt{app}}((\!|n \vdash u|\!)[q_1, \dots, q_n])((\!|n \vdash v|\!)[q_1, \dots, q_n])\ .
\end{aligned}$$

We write the arguments of the function $(\!|n \vdash u|\!)$ between square brackets [and].

▶ Remark A.6. The semantics of Definition A.5 factor through the encoding of Definition A.2 in the simply typed λ-calculus and its semantic interpretation as for all $q_1, \dots, q_n \in Q$,

$$(\!|n \vdash u|\!)(q_1, \dots, q_n) \quad = \quad [\![\overline{n \vdash u}]\!]_Q\,(f_{\mathtt{abs}})(f_{\mathtt{app}})(q_n, \dots, q_1)\ .$$

We now instantiate Definition A.5 with the following values of Q, $f_{\mathtt{abs}}$ and $f_{\mathtt{app}}$:

- Q is the set $\{0, 1, \infty\} \times \{\bot, \top\}$, with its product monoid structure. For any $q \in Q$, we write $q_1 \in \{0, 1, \infty\}$ and $q_2 \in \{\bot, \top\}$ for its two components.
- $f_{\mathtt{abs}}$ is the function $(Q \Rightarrow Q) \to Q$ defined as

$$g \quad \longmapsto \quad (g(0,\top)_1\ ,\ g(0,\top)_2 \wedge (g(1,\top)_1 \neq \infty))$$

- $f_{\mathtt{app}}$ is the curried monoid product of Q, i.e. the function $Q \to (Q \Rightarrow Q)$ defined as

$$(n, b) \quad \longmapsto \quad (n', b') \mapsto (n + n', b \wedge b')\ .$$

▶ **Proposition A.7** (Left part of the tuple). *For any scoped term $n \vdash u$ and any elements $k_1, \dots k_n$ of $\{0, 1, \infty\}$, we have*

$$(\!|n \vdash u|\!)[(k_1, \top)\ ,\ \dots\ ,\ (k_n, \top)]_1 \quad = \quad \mathtt{occ}_1 \cdot k_1 + \dots + \mathtt{occ}_n \cdot k_n$$

where the product $\cdot : \mathbb{N} \times \{0, 1, \infty\} \to \{0, 1, \infty\}$ comes from the monoid structure of $\{0, 1, \infty\}$.

Proof. We verify this by induction on the scoped term $n \vdash u$.

$$\begin{aligned}
&(\!|n \vdash \mathtt{var}_i|\!)[(k_1, \top), \dots, (k_n, \top)]_1 \\
&:= k_i \\
&= \mathtt{occ}_1 \cdot k_1 + \cdots + \mathtt{occ}_n \cdot k_n
\end{aligned}$$

$$\begin{aligned}
&(\!|n \vdash \mathtt{abs}(u)|\!)[(k_1, \top), \dots, (k_n, \top)]_1 \\
&:= f_{\mathtt{abs}}\left((k, b) \mapsto (\!|n+1 \vdash u|\!)[(k, b), (k_1, \top), \dots, (k_n, \top)]\right)_1 \\
&= (\!|n+1 \vdash u|\!)[(0, \top), (k_1, \top), \dots, (k_n, \top)]_1 \\
&= \mathtt{occ}_1 \cdot k_1 + \cdots + \mathtt{occ}_n \cdot k_n
\end{aligned}$$

$$\begin{aligned}
&(\!|n \vdash \mathtt{app}(u, v)|\!)[(k_1, \top), \dots, (k_n, \top)]_1 \\
&:= f_{\mathtt{app}}((\!|n \vdash u|\!)[(k_1, \top), \dots, (k_n, \top)])((\!|n \vdash v|\!)[(k_1, \top), \dots, (k_n, \top)])_1 \\
&= (\!|n \vdash u|\!)[(k_1, \top), \dots, (k_n, \top)]_1 \;+\; (\!|n \vdash v|\!)[(k_1, \top), \dots, (k_n, \top)]_1 \\
&= \mathtt{occ}_1 \cdot k_1 + \cdots + \mathtt{occ}_n \cdot k_n
\end{aligned}$$

In the `abs` case, the last equality is obtained by remarking that $\mathtt{occ}_1(n+1 \vdash u)$ is multiplied by 0 and that $\mathtt{occ}_{i+1}(n+1 \vdash u) = \mathtt{occ}_i(n \vdash \mathtt{abs}(u))$ for $i \geq 1$. ◀

We introduce the following definition.

▶ **Definition A.8.** *We define the property of a scoped term to be* affine in its bound variables *by induction as follows:*

- $n \vdash \mathtt{var}_i$ *is always affine in its bound variables,*
- $n \vdash \mathtt{abs}(u)$ *is affine in its bound variables if and only if*

  $\mathtt{occ}_1(n+1 \vdash u) \;\leq\; 1$ *and* $n+1 \vdash u$ *is affine in its bound variables*

- $n \vdash \mathtt{app}(u, v)$ *is affine in its bound variables if and only if*

  $n \vdash u$ *and* $n \vdash v$ *are both affine in their bound variables.*

A λ-term $t \in \Lambda(\mathsf{UntypedTerms})$ will said to be affine *when the closed untyped term $0 \vdash u$ bijectively associated to t by Claim A.3 is affine in its bound variables.*

▶ **Proposition A.9** (Right part of the tuple)**.** *For any scoped term $n \vdash u$, we have*

$$(\!|n \vdash u|\!)[(0, \top), \dots, (0, \top)]_2 \quad = \quad b$$

where b is $\top$ if and only if the scoped term $n \vdash u$ is affine in its bound variables.

Proof. We prove this by induction on the scoped term $n \vdash u$.

- For any $1 \leq i \leq n$, $n \vdash \mathtt{var}_i$ is always affine in its bound variables, and we always have

  $$(\!|n \vdash \mathtt{var}_i|\!)[(0, \top), \dots, (0, \top)]_2 \;=\; \top\,.$$

- For any scoped term $n+1 \vdash u$, we have

  $$\begin{aligned}
  &(\!|n \vdash \mathtt{abs}(u)|\!)[(0, \top), \dots, (0, \top)]_2 \\
  &:= f_{\mathtt{abs}}((k, b) \mapsto (\!|n+1 \vdash u|\!)[(k, b), (0, \top), \dots, (0, \top)])_2 \\
  &= (\!|n+1 \vdash u|\!)[(0, \top), (0, \top), \dots, (0, \top)]_2 \\
  &\quad \wedge \quad (\!|n+1 \vdash u|\!)[(1, \top), (0, \top), \dots, (0, \top)]_1 \neq \infty \\
  &= (\!|n+1 \vdash u|\!)[(0, \top), (0, \top), \dots, (0, \top)]_2 \\
  &\quad \wedge \quad \mathtt{occ}_1(n+1 \vdash u) \leq 1\,.
  \end{aligned}$$

where the last step comes from Proposition A.7. By the induction hypothesis on the scoped term $n+1 \vdash u$, we get that $n \vdash \mathtt{abs}(u)$ is affine in its bound variables if and only if $(\!|n \vdash \mathtt{abs}(u)|\!)[(0,\top), \dots, (0,\top)]_2$ is $\top$.

- For any scoped terms $n \vdash u$ and $n \vdash v$, we have

$$\begin{aligned}&(\!|n \vdash \mathtt{app}(u,v)|\!)[(0,\top), \dots, (0,\top)]_2\\ &:= f_{\mathtt{app}}((\!|n \vdash u|\!)[(0,\top), \dots, (0,\top)])((\!|n \vdash v|\!)[(0,\top), \dots, (0,\top)])_2\\ &= (\!|n \vdash u|\!)[(0,\top), \dots, (0,\top)]_2 \ \wedge \ (\!|n \vdash v|\!)[(0,\top), \dots, (0,\top)]_2\end{aligned}$$

which shows, by the induction hypotheses on $n \vdash u$ and $n \vdash v$, that $n \vdash \mathtt{app}(u,v)$ is affine in its bound variables if and only if $(\!|n \vdash \mathtt{app}(u,v)|\!)[(0,\top), \dots, (0,\top)]_2$ is $\top$. ◀

▶ **Theorem A.10.** *The language of closed affine untyped terms is regular in* **FinSet**.

Proof. We consider the language

$$L := \{t \in \Lambda(\mathsf{UntypedTerms}) \mid t \text{ is affine}\} .$$

By definition, $t \in \Lambda(\mathsf{UntypedTerms})$ is affine if and only if $0 \vdash u$ is affine in its bound variables, where $0 \vdash u$ is the unique scoped term such that $t = \overline{0 \vdash u}$, given by Claim A.3. Moreover, by Proposition A.9, we have that $0 \vdash u$ is affine in its bound variables if and only if

$$(\!|0 \vdash u|\!)_2 \quad = \quad \top$$

and, by Remark A.6, $(\!|0 \vdash u|\!)_2 = [\![t]\!]_Q\,(f_{\mathtt{abs}})(f_{\mathtt{app}})$. Therefore, if we define the subset F of $[\![\mathsf{UntypedTerms}]\!]_Q$ as

$$F \quad := \quad \{s \in [\![\mathsf{UntypedTerms}]\!]_Q \mid s(f_{\mathtt{abs}})(f_{\mathtt{app}})_2 = \top\}$$

we get that $L = L_F$ and therefore that L is **FinSet**-recognizable. ◀

B Squeezing structures

Proof of Proposition 5.8. There exists a squeezing structure such that

- for any object $R = (Q, e, \Vdash)$ of $\mathbf{R}$, the objects L_R and R_R are both equal to T_e;
- the two wide subcategories $\mathbf{R}_{\text{left}}$ and $\mathbf{R}_{\text{right}}$ are both taken to be the wide subcategory of target-identities.

The fact that the point functor T is product-preserving gives us all the target-identities of the squeezing structure, except for the case of the morphism

$$(\mathbf{E}(1,e), e, \sim_e) \Rightarrow (\mathbf{E}(1,e'), e', \sim_{e'}) \ \longrightarrow \ (\mathbf{E}(1, e \Rightarrow e'), e \Rightarrow e', \sim_{e \Rightarrow e'}) .$$

For this morphism, we will crucially use the fact that we relate **FinSet** and $\mathbf{E}$, which is well-pointed. We have a set-theoretic function

$$i \quad : \quad \mathbf{E}(1, e \Rightarrow e') \ \longrightarrow \ \mathbf{E}(1,e) \Rightarrow \mathbf{E}(1,e')$$

which is injective as $\mathbf{E}$ is well-pointed and lifts to a target-identity. As the object $e \Rightarrow e'$ of $\mathbf{E}$ is inhabited, the set $\mathbf{E}(1, e \Rightarrow e')$ is non-empty and the set-theoretic function i admits a retraction

$$r \quad : \quad \mathbf{E}(1,e) \Rightarrow \mathbf{E}(1,e') \ \longrightarrow \ \mathbf{E}(1, e \Rightarrow e')$$

which lifts to a target-identity $T_e \Rightarrow T_{e'} \to T_{e \Rightarrow e'}$. By Lemma 4.11, we know that the objects of $\mathbf{Sqz}(\mathbf{R})$ are partial surjections. Conversely, suppose that $R = (Q, e, \Vdash)$ is such that $\Vdash$ is a partial surjection. We the two following target-identities:

- The fact that the relation $\Vdash$ is surjective yields a set-theoretic function $\mathbf{E}(1,e) \to Q$ which lifts to a target-identity $T_e \to R$.
- As the relation $\Vdash$ is functional and $\mathbf{E}(1,e)$ is non-empty as e is inhabited, there exists a set-theoretic function $Q \to \mathbf{E}(1,e)$ extending $\Vdash$, and any such set-theoretic function lifts to a target-identity $R \to T_e$.

This shows that the objects of $\mathbf{Sqz}(\mathbf{R})$ are exactly the partial surjections, which then form a sub-CCC of $\mathbf{R}$ by Theorem 4.10. ◀

Proof of Proposition 6.4. It is clear that target-identities are composable and stable under finite products and exponentials, which is what is asked given that left and right morphisms are the same in the present case.

We are left with the task to show the existence of the morphisms as described in Equation (2) in our particular setting. As there exists at most one target-identity whose **Lam** component is a given λ-term, we give these λ-terms.

- **Case** $\mathrm{Bij}_1 \to 1$**:** The unique morphism $\mathrm{Bij}_1 \to 1$ is a target-identity.
- **Case** $1 \to \mathrm{Bij}_1$**:** The λ-term $\lambda(y:1).\,\lambda(x:\mathfrak{o}).\,x$ lifts to a target-identity

$$1 \quad \longrightarrow \quad (\mathrm{Fin}(1), 1, \Vdash_1)\ .$$

- **Case** $\mathrm{Bij}_{n\times n'} \to \mathrm{Bij}_n \times \mathrm{Bij}_{n'}$**:** the fact that $\mathrm{Bij}_{(-)}$ is a functor gives us directly such a morphism which can be verified to be a target-identity.
- **Case** $\mathrm{Bij}_n \times \mathrm{Bij}_{n'} \to \mathrm{Bij}_{n\times n'}$**:** The morphism is given by the λ-term

$$\lambda(p : \mathrm{Fin}(n) \times \mathrm{Fin}(n')).\,\lambda(x : \mathfrak{o}^{n\times n'}).\,p_1\ \langle \mathrm{Fin}(1 \times \mathrm{Id})\ p_2\ x, \ldots, \mathrm{Fin}(n \times \mathrm{Id})\ p_2\ x\rangle$$

where $i \times \mathrm{Id}$ is the function $n' \to n \times n'$ sending j on (i,j), from which we get the λ-term $\mathrm{Fin}(i \times \mathrm{Id})$ of simple type $\mathrm{Fin}(n') \Rightarrow \mathrm{Fin}(n \times n')$.
This λ-term lifts to a target-identity

$$\mathrm{Bij}_n \times \mathrm{Bij}_{n'} \ \longrightarrow\ \mathrm{Bij}_{n\times n'}\ .$$

- **Case** $\mathrm{Bij}_{n\Rightarrow n'} \to \mathrm{Bij}_n \Rightarrow \mathrm{Bij}_{n'}$**:** We have the target-identity

$$\mathrm{Bij}_{n\Rightarrow n'} \times \mathrm{Bij}_n \ \longrightarrow\ \mathrm{Bij}_{(n\Rightarrow n')\times n}$$

which, when composed with the morphism $\mathrm{Bij}_{\mathrm{ev}_{n,n'}}$ which has the evaluation morphism $\mathrm{ev}_{n,n'}$ as target-component, yields a morphism

$$\mathrm{Bij}_{n\Rightarrow n'} \times \mathrm{Bij}_n \ \longrightarrow\ \mathrm{Bij}_{n'}$$

which, after curryfication, gets us a target-identity $\mathrm{Bij}_{n\Rightarrow n'} \to \mathrm{Bij}_n \Rightarrow \mathrm{Bij}_{n'}$.
- **Case** $\mathrm{Bij}_n \Rightarrow \mathrm{Bij}_{n'} \to \mathrm{Bij}_{n\Rightarrow n'}$**:** Notice that the equality

$$n \Rightarrow n' \ = \ \underbrace{n' \times \cdots \times n'}_{n \text{ times}}$$

shows that the λ-term of type $\mathrm{Fin}(n) \Rightarrow \mathrm{Fin}(n') \to (\mathrm{Fin}(n'))^n$

$$\lambda(F : \mathrm{Fin}(n) \Rightarrow \mathrm{Fin}(n')).\,\langle F\,\pi_1, \ldots, F\,\pi_n\rangle$$

lifts to a target-identity $\mathrm{Bij}_n \Rightarrow \mathrm{Bij}_{n'} \to \mathrm{Bij}_{n'}{}^n$. By postcomposing this target-identity with an iteration of the target-identities $\mathrm{Bij}_m \times \mathrm{Bij}_{m'} \to \mathrm{Bij}_{m\times m'}$, we obtain the target-identity

$$\mathrm{Bij}_n \Rightarrow \mathrm{Bij}_{n'} \ \longrightarrow\ \mathrm{Bij}_{n\Rightarrow n'}\ .$$

This finishes the proof that there is a squeezing structure as described in the statement of the proposition. ◀

Promise and Infinite-Domain Constraint Satisfaction

Antoine Mottet
Hamburg University of Technology, Research Group for Theoretical Computer Science, Germany

Abstract

Two particularly active branches of research in constraint satisfaction are the study of *promise* constraint satisfaction problems (PCSPs) with finite templates and the study of *infinite-domain* constraint satisfaction problems with ω-categorical templates. In this paper, we explore some connections between these two hitherto unrelated fields and describe a general approach to studying the complexity of PCSPs by constructing suitable infinite CSP templates. As a result, we obtain new characterizations of the power of various classes of algorithms for PCSPs, such as first-order logic, arc consistency reductions, and obtain new proofs of the characterizations of the power of the classical linear and affine relaxations for PCSPs.

2012 ACM Subject Classification Theory of computation → Complexity theory and logic

Keywords and phrases promise constraint satisfaction problems, polymorphisms, homogeneous structures, first-order logic

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.41

1 Introduction

Promise constraint satisfaction problems (PCSPs) are problems of the following form: given a set of constraints on some variables, each of which coming as a pair of strong/weak constraints, determine whether the set of strong constraints is satisfiable or not even the set of weak constraints is satisfiable. Formally, such problems are parametrized by a pair $(\mathbb{A}, \mathbb{B})$ of relational structures such that $\mathbb{A}$ admits a homomorphism to $\mathbb{B}$. Typically, both structures are assumed to be finite. An instance of $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ is a structure $\mathbb{X}$, whose domain are understood as variables, and whose tuples in relations correspond to constraints in the same signature as $\mathbb{A}$ and $\mathbb{B}$. The problem is to decide whether there exists a homomorphism $\mathbb{X} \to \mathbb{A}$, representing a solution to a system of strong constraints, or no homomorphism $\mathbb{X} \to \mathbb{B}$, representing the absence of a solution to a weakening of the constraints. When $\mathbb{A} = \mathbb{B}$, one recovers the classical framework of constraint satisfaction problems, for which it is known that if $\mathbb{A}$ is finite, then the associated CSP is either solvable in polynomial time or is NP-hard [44, 43, 21]. Other refined classifications are known, for instance it is known which CSPs are definable by a first-order sentence [2, 41], or by a sentence in fixpoint logic with counting [3], or solvable by Datalog programs [9], or by the basic linear relaxation [35].

The study of PCSPs started recently in the works of [6] and [18], whose motivation was to define a structurally rich framework dedicated to the study of the complexity of approximation problems such as approximate graph coloring. These problems form a framework suitable to the study of a combinatorial, or qualitative, form of approximation, compared to the usual quantitative form. This combinatorial viewpoint allows for conceptually simpler proofs of inapproximability results, such as a combinatorial version of the PCP theorem [10].

A powerful algebraic approach to the study of the complexity of PCSPs was given by [8], building on the existing algebraic tools developed in the context of constraint satisfaction. A plethora of results ensued, providing new polynomial-time algorithms solving such problems [23, 19, 27] and new tools for proving hardness [42, 34].

Another active branch of research in constraint satisfaction is the study of CSPs where the template is an infinite structure. Here, no dichotomy similar to the Bulatov-Zhuk theorem is possible, as every computational problem is equivalent to the CSP of an infinite structure [11, 30]. Nonetheless, a similar algebraic approach has been developed for a class of infinite structures (so called *ω-categorical structures*) that is suitable for the study the complexity of the associated CSPs. We refer to [39] and the introduction of [37] for a description of the state of the art in the area.

There is until now little to no interplay between these two directions of research. While the tractability of some PCSPs with a finite template has been shown by a reduction to infinite-domain CSPs (typically through the use of a linear relaxation or systems of linear equations over the integers), the infinite templates that arise here are not ω-categorical and are not subject to the aforementioned algebraic approach. We note that Barto [7] and Barto and Asimi [1] proved that there are finite PCSP templates $(\mathbb{A}, \mathbb{B})$ such that for every finite structure $\mathbb{C}$ such that $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ reduces to $\mathrm{CSP}(\mathbb{C})$ by a trivial reduction, then $\mathrm{CSP}(\mathbb{C})$ is NP-hard. Thus, in a sense, the use of infinite-domain CSPs is sometimes necessary.

1.1 Contributions

In this work, we show how the by-now classical tools used to study CSPs with ω-categorical templates can also be used essentially as black boxes to tackle some questions arising in the study of PCSPs with finite templates.

Descriptive Complexity of PCSPs

While algorithmic and algebraic questions concerning PCSPs have received much attention in the past years, the logical aspects pertaining to these problems, and in particular questions about their descriptive complexity, remain mostly unexplored. Here, one of the questions of interest is to determine criteria for the existence of a sentence Φ in a given logic $\mathcal{L}$ whose class of finite models separates the yes-instances from the no-instances of a PCSP. More specifically, we say that $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ is *solvable* by a sentence Φ if the following holds:

- For every finite structure $\mathbb{X}$ such that $\mathbb{X}$ admits a homomorphism to $\mathbb{A}$, then $\mathbb{X} \models \Phi$,
- For every finite structure $\mathbb{X}$ such that $\mathbb{X}$ does not admit a homomorphism to $\mathbb{B}$, then $\mathbb{X} \not\models \Phi$.

▶ **Problem 1** (Separability problem for $\mathcal{L}$). *For which promise constraint satisfaction problems* $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ *does there exist a sentence* $\Phi \in \mathcal{L}$ *such that* $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ *is solvable by* Φ*?*

In the context of finite-domain CSPs, i.e., when $\mathbb{A} = \mathbb{B}$, questions of this type have been answered for several logics including first-order logic [2, 41] and some fixpoint logics [3, 9], while some important cases remain open, e.g., for the case of CSPs definable in linear Datalog or in fixpoint logics with a rank operator. Atserias and Dalmau [4] have given necessary algebraic conditions for a given PCSP to be solvable by a Datalog program, but no characterization is known at the moment.

We give an answer to Problem 1 in the case that $\mathcal{L}$ is first-order logic.

▶ **Theorem 2.** *Let* $(\mathbb{A}, \mathbb{B})$ *be a finite PCSP template. The following are equivalent:*

1. $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ *is solvable in first-order logic;*
2. $(\mathbb{A}, \mathbb{B})$ *has finite duality;*
3. *There exists a finite structure* $\mathbb{C}$ *with finite duality and such that* $\mathbb{A} \to \mathbb{C} \to \mathbb{B}$.

Interestingly, although Theorem 2 is a statement purely about finite structures, our proof uses a combination of techniques coming from the study of CSPs with infinite structures. We first construct an infinite structure $\mathbb{C}$ with the same properties as in Item 3 of Theorem 2, and using a Ramsey argument we prove that a finite factor of $\mathbb{C}$ also satisfies the required properties. This method appears to be quite flexible and underlies the proofs of the main results of this paper.

Application to the Search PCSP

Arguably, the variant of the PCSP that is most interesting from an application point of view is the following: given a structure $\mathbb{X}$ with the promise that there exists a homomorphism $\mathbb{X} \to \mathbb{A}$, compute a homomorphism $\mathbb{X} \to \mathbb{B}$. For CSPs, the search version can be solved in polynomial time, given an oracle deciding the decision version; this is not known to hold for promise CSPs.

Very little is known in general concerning the search version of promise CSPs: the tractability of the search version of $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ is currently only known when there exists a structure $\mathbb{C}$ such that $\mathbb{A} \to \mathbb{C} \to \mathbb{B}$, such that $\mathrm{CSP}(\mathbb{C})$ can be solved in polynomial-time, and such that a homomorphism $\mathbb{C} \to \mathbb{B}$ can be computed in polynomial time. So far, this is only used when $\mathbb{C}$ is a finite structure.

A straightforward consequence of Theorem 2 is that if $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ is solvable in first-order logic, then its search version is tractable. We are in the advantageous situation where $\mathbb{C}$ can be taken finite, but in fact the result would already follow from the existence of a suitable infinite $\mathbb{C}$. Thus, we anticipate that our methods can prove the tractability of the search version of PCSPs in much more general settings than first-order solvability.

Revisiting Arc Consistency and the Arc Consistency Reduction

Arc consistency is a common heuristic employed in constraint solvers to reduce the search space and potentially speed up the process of finding a solution to a CSP instance (or to prove that no solution exists). In this heuristic, one stores for every variable of the instance a set of possible values that this variable can take in a homomorphism $\mathbb{X} \to \mathbb{A}$, and one makes gradual refinements until a fixed point is reached.

Some PCSPs, known as width-1 PCSPs, are in fact completely solved by this heuristic: whenever the heuristic does not yield conclude the absence of a homomorphism $\mathbb{X} \to \mathbb{A}$, then there does actually exist a homomorphism $\mathbb{X} \to \mathbb{B}$. A characterization of finite width-1 CSPs was given by Feder and Vardi [29] in their seminal paper, and generalized to the case of PCSPs [8]. We reprove in Section 4 this characterization by a straightforward adaptation of the method used to characterize the power of first-order logic for PCSPs. Using the same methods, we also give in Section 4 an alternative description of the recent arc consistency reduction proposed in [28].

A current open problem in the theory of PCSPs is to characterize those templates whose PCSP is solved by a strengthening of the arc consistency heuristic, where information about tuples of variables of bounded length can be stored. This corresponds to solving Problem 1 in the case of Datalog, or the existential positive fragment of least fixpoint logic. While we do not solve this problem here, we argue in Section 4 that our method could shed some light on this problem.

Revisiting Linear and Affine Relaxations

Another heuristic to solve PCSPs is to consider linear relaxations of CSP instances, where a CSP instance is mapped to a linear program (known as the basic linear programming relaxation) or to a system of linear equations over the integers (known as the affine integer programming relaxation). The class of PCSPs that are correctly solved by these heuristic have been characterized in [8]. With the methods used in the previous sections, we give another viewpoint on these classes and reprove those characterizations.

2 Definitions

A signature σ is a set of relation symbols, each of which having an arity. A σ-structure $\mathbb{A}$ consists of a set A, called its *domain*, together with an interpretation $R^{\mathbb{A}} \subseteq A^r$ for each relation symbol $R \in \sigma$ of arity $r \geq 1$. A σ-structure $\mathbb{B}$ with $B \subseteq A$ is a *substructure* of $\mathbb{A}$ if for every R in σ of arity r, we have $R^{\mathbb{B}} = R^{\mathbb{A}} \cap B^r$.[1] If $\tau \subseteq \sigma$ and $\mathbb{A}$ is a σ-structure, the τ-reduct of $\mathbb{A}$ is the τ-structure obtained by forgetting the interpretation of the symbols in $\sigma \setminus \tau$. An *expansion* of a structure $\mathbb{A}$ is a structure in a larger signature obtained by adding new relations to $\mathbb{A}$. All relational structures in this paper are at most countable and have a finite signature unless specified otherwise.

A homomorphism $h\colon \mathbb{A} \to \mathbb{B}$ between two σ-structures is a map $A \to B$ such that for all $R \in \sigma$ of arity r and $(a_1, \dots, a_r) \in R^{\mathbb{A}}$, we have $(h(a_1), \dots, h(a_r)) \in R^{\mathbb{B}}$. We use the notation $\mathbb{A} \to \mathbb{B}$ to denote the existence of a homomorphism from $\mathbb{A}$ to $\mathbb{B}$.

A cycle in a structure $\mathbb{A}$ is a set of tuples $\overline{a}^1, \dots, \overline{a}^k$ of length $r_1, \dots, r_k$, each of which appearing in a relation of $\mathbb{A}$, such that the set consisting of all entries of the tuples has size at most $\sum_i (r_i - 1)$. The smallest k for which there exists such a cycle in $\mathbb{A}$ is called the *girth* of $\mathbb{A}$. A *tree* is a structure with no cycles.

2.1 Promise Constraint Satisfaction Problems

For every two structures $\mathbb{A}, \mathbb{B}$ such that there exists a homomorphism $\mathbb{A} \to \mathbb{B}$, we define $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ as the problem of deciding, given a finite structure $\mathbb{X}$, if there exists a homomorphism $\mathbb{X} \to \mathbb{A}$ or no homomorphism $\mathbb{X} \to \mathbb{B}$; the promise is that at least one of these cases holds, and the existence of a homomorphism $\mathbb{A} \to \mathbb{B}$ ensures that at most one case holds. The pair of structures $(\mathbb{A}, \mathbb{B})$ is called the *template* of the PCSP. The search version of $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ asks to compute a homomorphism $\mathbb{X} \to \mathbb{B}$, given a finite structure $\mathbb{X}$ that is promised to admit a homomorphism to $\mathbb{A}$ (although a homomorphism $\mathbb{X} \to \mathbb{A}$ is of course not given). We define $\mathrm{CSP}(\mathbb{A})$ as $\mathrm{PCSP}(\mathbb{A}, \mathbb{A})$. Every tuple $(x_1, \dots, x_k) \in R^{\mathbb{X}}$ in an instance $\mathbb{X}$ of $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ is called a *constraint*.

We say that $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ is first-order solvable if there exists a first-order sentence Φ such that the following items hold for every finite structure $\mathbb{X}$:

- if $\mathbb{X}$ has a homomorphism to $\mathbb{A}$, then $\mathbb{X} \models \Phi$,
- if $\mathbb{X} \models \Phi$, then $\mathbb{X}$ admits a homomorphism to $\mathbb{B}$.

Thus, the set of finite models of Φ separates the yes-instances of $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ from the no-instances. If $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ is first-order solvable, one trivially gets a logspace algorithm solving $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$.[2]

[1] As usual in model theory, all substructures are necessarily *induced substructures*.

[2] The truth of a first-order formula with k quantifiers can be checked by iterating over all elements of the input structure and storing k logarithmically-sized pointers.

Let $\mathcal{F}$ be a family of finite structures. A structure $\mathbb{X}$ is *$\mathcal{F}$-free* if there does not exist any $\mathbb{F} \in \mathcal{F}$ such that $\mathbb{F}$ admits a homomorphism to $\mathbb{X}$. We say that a PCSP template $(\mathbb{A}, \mathbb{B})$ has duality $\mathcal{F}$ if the following hold for every finite $\mathbb{X}$:

- if $\mathbb{X}$ admits a homomorphism to $\mathbb{A}$, then $\mathbb{X}$ is $\mathcal{F}$-free,
- if $\mathbb{X}$ is $\mathcal{F}$-free, then $\mathbb{X}$ admits a homomorphism to $\mathbb{B}$.

We say that $(\mathbb{A}, \mathbb{B})$ has *finite duality* if it has duality $\mathcal{F}$ for a finite set $\mathcal{F}$. A structure has *finite tree duality* if it has a finite duality consisting only of trees.

An operation $f\colon A^n \to A$ is a *polymorphism* of $\mathbb{A}$ if it is a homomorphism from $\mathbb{A}^n$ to $\mathbb{A}$, where $\mathbb{A}^n$ is the structure with domain A^n, the same signature as $\mathbb{A}$, and with relations

$$R^{\mathbb{A}^n} := \left\{(\overline{a}^1, \dots, \overline{a}^r) \mid \forall j, (a_j^1, \dots, a_j^r) \in R^{\mathbb{A}}\right\}.$$

In other words, f is a polymorphism if, whenever $\overline{a}^1, \dots, \overline{a}^n$ are in a relation R of $\mathbb{A}$, then $f(\overline{a}^1, \dots, \overline{a}^n)$ is in R, where f is applied componentwise to every tuple $\overline{a}^i$. We define a polymorphism of a PCSP template $(\mathbb{A}, \mathbb{B})$ similarly, as a homomorphism $\mathbb{A}^n \to \mathbb{B}$.

A polymorphism f of a structure $\mathbb{A}$ is *1-tolerant* if it satisfies that $f(\overline{a}^1, \dots, \overline{a}^n)$ is in $R^{\mathbb{A}}$, whenever all but at most one of $\overline{a}^1, \dots, \overline{a}^n$ is in $R^{\mathbb{A}}$. For finite-domain or ω-categorical CSPs, the following characterization of first-order solvability is known.

▶ **Theorem 3** ([36, 2, 41, 12]). *Let $\mathbb{A}$ be a finite or ω-categorical structure. The following are equivalent:*

- *$\mathbb{A}$ has finite duality,*
- *$\mathbb{A}$ has finite tree duality,*
- *$\mathrm{CSP}(\mathbb{A})$ is solvable in first-order logic,*
- *$\mathbb{A}$ has a 1-tolerant polymorphism.*

It is proven in [36] that it is possible to decide, for a finite structure $\mathbb{A}$, whether $\mathbb{A}$ has finite duality.

2.2 ω-categorical structures

An *embedding* $e\colon \mathbb{A} \to \mathbb{B}$ is an injective homomorphism such that for every relation R of arity r and every $a_1, \dots, a_r \in A$, one has $(e(a_1), \dots, e(a_r)) \in R^{\mathbb{B}}$ if, and only if, $(a_1, \dots, a_r) \in R^{\mathbb{A}}$. An *automorphism* of a structure $\mathbb{A}$ is an embedding $\alpha\colon \mathbb{A} \to \mathbb{A}$ that is surjective. In other words, both α and its inverse are homomorphisms $\mathbb{A} \to \mathbb{A}$.

A structure $\mathbb{A}$ if *ω-categorical* is for all $n \geq 1$, the equivalence relation $\sim_n^{\mathbb{A}}$ on A^n defined by $x \sim_n^{\mathbb{A}} y$ iff there exists an automorphism $\alpha \in \mathrm{Aut}(\mathbb{A})$ with $\alpha(x) = y$ has finitely many equivalence classes, . These equivalence classes are called *orbits under* $\mathrm{Aut}(\mathbb{A})$. Typical examples of ω-categorical structures are the "structure with no structure" $(\mathbb{N}; =)$, for which the classes of the equivalence relation $\sim_n$ are in 1-to-1 correspondence with partitions of $\{1, \dots, n\}$, and $(\mathbb{Q}; <)$, for which the classes of the equivalence relation $\sim_n$ correspond to weak linear orders on $\{1, \dots, n\}$.

A structure $\mathbb{A}$ is *homogeneous* if every isomorphism $f\colon \mathbb{B} \to \mathbb{C}$ between finite substructures of $\mathbb{A}$ extends to an automorphism of $\mathbb{A}$. Thus, in a homogeneous structure, the orbits under $\mathrm{Aut}(\mathbb{A})$ are completely determined by the isomorphism types of n-element substructures of $\mathbb{A}$, or equivalently by the quantifier-free formulas with n variables up to equivalence over $\mathbb{A}$.

A countable set $\mathcal{C}$ of finite structures is said to have the *amalgamation property* if for all structures $\mathbb{X}, \mathbb{Y}_1, \mathbb{Y}_2 \in \mathcal{C}$ and embeddings $f_i\colon \mathbb{X} \to \mathbb{Y}_i$, there exist a structure $\mathbb{Z} \in \mathcal{C}$ and embeddings $e_i\colon \mathbb{Y}_i \to \mathbb{Z}$ such that $e_1 \circ f_1 = e_2 \circ f_2$. We say that $\mathbb{Z}$ is an *amalgam* over $\mathbb{Y}_1, \mathbb{Y}_2$ over $\mathbb{X}$. We say that $\mathbb{Z}$ is a *strong amalgam* if $e_1(Y_1) \cap e_2(Y_2) = e_1(f_1(X))$, and we

say $\mathcal{C}$ has the *strong amalgamation property* when $\mathbb{Z}$ can always be chosen to be a strong amalgam, regardless of $\mathbb{X}, \mathbb{Y}_1, \mathbb{Y}_2$. We say that $\mathbb{Z}$ is a *free amalgam* if it is strong and no tuple containing entries from both $e_1(Y_1 \setminus f_1(X))$ and $e_2(Y_2 \setminus f_2(X))$ belongs to a relation of $\mathbb{Z}$. By a classical result of Fraïssé, for every countable class of finite structures $\mathcal{C}$ that is closed under substructures, there exists a countable homogeneous structure $\mathbb{C}$ whose finite substructures are exactly those structures that are isomorphic to a member of $\mathcal{C}$. The structure $\mathbb{C}$ is called the *Fraïssé limit* of $\mathcal{C}$.

2.3 Ramsey expansions and canonical polymorphisms

An operation $f\colon A^n \to B$ is *canonical from* $\mathbb{A}$ *to* $\mathbb{B}$ if it is a homomorphism from the nth power of $(A; \sim_1^{\mathbb{A}}, \sim_2^{\mathbb{A}}, \dots)$ to $(B; \sim_1^{\mathbb{B}}, \sim_2^{\mathbb{B}}, \dots)$. In other words, f is canonical from $\mathbb{A}$ to $\mathbb{B}$ if, and only if, for all m-tuples $\overline{a}^1, \dots, \overline{a}^n$, and all $\alpha_1, \dots, \alpha_n$ automorphisms of $\mathbb{A}$, the tuples $f(\overline{a}^1, \dots, \overline{a}^m)$ and $f(\alpha_1(\overline{a}^1), \dots, \alpha_n(\overline{a}^n))$ are in the same orbit under $\mathrm{Aut}(\mathbb{B})$. Canonical functions typically arise by an application of the following result. This result uses the *Ramsey* property of some homogeneous structures; we will only use the Ramsey property as a blackbox in this paper and therefore omit the definition.

▶ **Theorem 4** ([16, 40]). *Let $\mathbb{A}$ be a homogeneous structure with the Ramsey property, let $\mathbb{B}$ be an ω-categorical structure, and let $f\colon A^n \to B$ be an arbitrary function. Then there exists $g\colon A^n \to B$ that is canonical from $\mathbb{A}$ to $\mathbb{B}$ and such that for every finite subset S of A^m, there exist $\alpha_1, \dots, \alpha_n \in \mathrm{Aut}(\mathbb{A}), \beta \in \mathrm{Aut}(\mathbb{B})$ such that $g(\overline{a}^1, \dots, \overline{a}^n) = \beta f(\alpha_1(\overline{a}^1), \dots, \alpha_n(\overline{a}^n))$ holds for all $\overline{a}^1, \dots, \overline{a}^n$ in S.*

In case $\mathbb{A} = \mathbb{B}$ in Theorem 4, we say that f *locally interpolates* g *modulo* $\mathrm{Aut}(\mathbb{A})$.

3 PCSPs solvable in First-Order Logic

It was first proven by [2] that a finite structure has finite duality if, and only if, its CSP can be defined in first-order logic. Another proof of this result was obtained by [41], who proved the following stronger statement.[3] For a first-order sentence Φ, let $\mathrm{Mod}(\Phi)$ be the class of all finite structures $\mathbb{X}$ such that $\mathbb{X} \models \Phi$.

▶ **Theorem 5** (Theorem 4.11 in [41]). *Let $\mathcal{P} \subseteq \mathcal{Q}$ be classes of structures, and Φ be a first-order sentence such that:*

- *for all finite $\mathbb{X}, \mathbb{Y}$ such that $\mathbb{X} \in \mathcal{P}$ and $\mathbb{X} \to \mathbb{Y}$, we have $\mathbb{Y} \models \Phi$,*
- *for all finite $\mathbb{X}, \mathbb{Y}$ such that $\mathbb{X} \models \Phi$ and $\mathbb{X} \to \mathbb{Y}$, we have $\mathbb{Y} \in \mathcal{Q}$.*

Then there exists an existential positive sentence Ψ such that $\mathcal{P} \subseteq \mathrm{Mod}(\Psi) \subseteq \mathcal{Q}$.

The following result, initially by Cherlin, Shelah, and Shi [22], and later improved by Hubička and Nešetřil [32], has found several applications in the study of infinite-domain CSPs in the recent years [13, 14, 15]. A structure is *connected* if it is not isomorphic to the disjoint union of two non-empty structures.

▶ **Theorem 6.** *Let $\mathcal{F}$ be a finite set of finite connected structures. There exists an ω-categorical structure $\mathbb{C}$ such that $\mathbb{C}$ has duality $\mathcal{F}$. Moreover, $\mathbb{C}$ can be chosen to have an expansion $\mathbb{C}^+$ by finitely many relations such that $\mathbb{C}^+$ is homogeneous with the Ramsey property.*

[3] To see that Theorem 5 implies Atserias's result, apply the theorem to $\mathcal{P} = \mathcal{Q} = \{\mathbb{X} \mid \mathbb{X} \not\to \mathbb{A}\}$.

In particular, $\mathbb{C}^+$ meets the hypothesis of Theorem 4, and therefore every polymorphism of $\mathbb{C}$ locally interpolates a polymorphism that is canonical with respect to $\mathrm{Aut}(\mathbb{C}^+)$. We now show that the property of being 1-tolerant is preserved under this local interpolation.

▶ **Lemma 7.** *Let $\mathbb{A}$ be an arbitrary structure. Let f be a 1-tolerant polymorphism of $\mathbb{A}$ and let Γ be a subset of $\mathrm{Aut}(\mathbb{A})$. Then every operation g that is locally interpolated by f modulo Γ is a 1-tolerant polymorphism of $\mathbb{A}$.*

Proof. Let g be locally interpolated by f modulo Γ, and let R be a relation of $\mathbb{A}$. Let $\overline{a}^1, \dots, \overline{a}^n \in A^r$ be such that all but at most one of them are in R. By assumption, there exist $\alpha_1, \dots, \alpha_n, \beta \in \Gamma$ such that $g(\overline{a}^1, \dots, \overline{a}^n) = \beta f(\alpha_1(\overline{a}^1), \dots, \alpha_n(\overline{a}^n))$. Since $\alpha_1, \dots, \alpha_n$ are automorphisms of $\mathbb{A}$, we obtain that for all but at most one j, one has $\alpha_j(\overline{a}^j) \in R$. Since f is 1-tolerant, $f(\alpha_1(\overline{a}^1), \dots, \alpha_n(\overline{a}^n))$ is in R, and thus $g(\overline{a}^1, \dots, \overline{a}^n)$ is in R since β is an automorphism of $\mathbb{A}$. ◀

In the following, recall that a first-order formula is *primitive positive* if it only consists of existential quantifications, conjunctions, and atomic formulas only. Every primitive positive formula $\varphi(x_1, \dots, x_n)$ without equalities corresponds in a canonical way to a relational structure $\mathbb{A}$, its *canonical database*, whose domain is the set of variables of the formula, and whose relations are determined by the conjuncts of the formula.

▶ **Theorem 2.** *Let $(\mathbb{A}, \mathbb{B})$ be a finite PCSP template. The following are equivalent:*

1. $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ *is solvable in first-order logic;*
2. $(\mathbb{A}, \mathbb{B})$ *has finite duality;*
3. *There exists a finite structure $\mathbb{C}$ with finite duality and such that $\mathbb{A} \to \mathbb{C} \to \mathbb{B}$.*

Proof. (1) implies (2). This is an immediate consequence of Theorem 5. Let $\mathcal{P}$ be the class of finite structures that do not admit a homomorphism to $\mathbb{B}$, and $\mathcal{Q}$ be the class of finite structures that do not admit a homomorphism to $\mathbb{A}$. Let Φ be a first-order sentence proving that $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ is in FO. Then $\mathcal{P} \subseteq \mathrm{Mod}(\neg\Phi) \subseteq \mathcal{Q}$ holds by definition. Moreover, if $\mathbb{X} \in \mathcal{P}$ and $\mathbb{X} \to \mathbb{Y}$, then $\mathbb{Y} \not\to \mathbb{B}$, hence $\mathbb{Y} \models \neg\Phi$. Similarly, if $\mathbb{X} \to \mathbb{Y}$ and $\mathbb{X} \models \neg\Phi$, then $\mathbb{X}$ does not admit a homomorphism to $\mathbb{A}$, so $\mathbb{Y}$ does not admit a homomorphism to $\mathbb{A}$, i.e., $\mathbb{Y} \in \mathcal{Q}$. Thus, Theorem 5 applies, and there exists an existential positive formula Ψ such that $\mathcal{P} \subseteq \mathrm{Mod}(\Psi) \subseteq \mathcal{Q}$, and Ψ is equivalent to a disjunction $\bigvee \Psi_i$ where each Ψ_i is a primitive positive sentence. Moreover, each Ψ_i can be assumed without loss of generality to not contain any equalities.[4] Let $\mathcal{F}$ be the set of canonical databases for each Ψ_i. For any finite $\mathbb{X}$, if there exists $\mathbb{F} \in \mathcal{F}$ such that $\mathbb{F} \to \mathbb{X}$, then $\mathbb{X} \models \Psi$, so that $\mathbb{X} \in \mathcal{Q}$ and $\mathbb{X}$ does not admit a homomorphism to $\mathbb{A}$. If $\mathbb{X}$ does not admit a homomorphism to $\mathbb{B}$, then $\mathbb{X} \in \mathcal{P}$ so that $\mathbb{X} \models \Psi$, and therefore there is $\mathbb{F} \in \mathcal{F}$ such that $\mathbb{F} \to \mathbb{X}$. Thus, $\mathcal{F}$ forms a duality for $(\mathbb{A}, \mathbb{B})$.

(2) implies (3). Let $\mathcal{F}$ be a duality for $(\mathbb{A}, \mathbb{B})$. Without loss of generality, we can assume that $\mathcal{F}$ consists of connected structures. Indeed, suppose that $\mathbb{F} \in \mathcal{F}$ is isomorphic to a disjoint union of non-empty structures $\mathbb{F}_1, \mathbb{F}_2$. Since $\mathbb{A}$ is $\mathcal{F}$-free, there is no homomorphism $\mathbb{F} \to \mathbb{A}$ and therefore one of $\mathbb{F}_1$ or $\mathbb{F}_2$ does not admit a homomorphism to $\mathbb{A}$, say without loss of generality that $\mathbb{F}_1$ does not. Consider $\mathcal{F}' := (\mathcal{F} \cup \{\mathbb{F}_1\}) \setminus \{\mathbb{F}\}$, which we prove is a duality for $(\mathbb{A}, \mathbb{B})$. Suppose that $\mathbb{X}$ admits a homomorphism to $\mathbb{A}$. Then $\mathbb{X}$ is $\mathcal{F}$-free and also $\mathbb{F}_1$-free since $\mathbb{F}_1 \not\to \mathbb{A}$, so that $\mathbb{X}$ is $\mathcal{F}'$-free. If $\mathbb{X}$ is $\mathcal{F}'$-free, then $\mathbb{X}$ is $\mathcal{F}$-free, so that $\mathbb{X}$ admits a homomorphism to $\mathbb{B}$.

[4] An equality in any Ψ_i can be removed by merging the corresponding variables.

By Theorem 6, there exists an ω-categorical structure $\mathbb{C}$ that has duality $\mathcal{F}$. Since $\mathbb{A}$ is $\mathcal{F}$-free, we have $\mathbb{A} \to \mathbb{C}$. Since every finite substructure of $\mathbb{C}$ is $\mathcal{F}$-free, there is a homomorphism from every finite substructure of $\mathbb{C}$ to $\mathbb{B}$. By compactness, there exists a homomorphism $\mathbb{C} \to \mathbb{B}$.

Since $\mathbb{C}$ is ω-categorical and has finite duality, there is an $f\colon \mathbb{C}^n \to \mathbb{C}$ that is a 1-tolerant polymorphism of $\mathbb{C}$ by Theorem 3. Moreover, $\mathbb{C}$ admits a Ramsey expansion $\mathbb{C}^+$. By applying Theorem 4 to $\mathbb{C}^+$, f locally interpolates an operation g modulo $\mathrm{Aut}(\mathbb{C}^+)$ that is canonical with respect to $\mathbb{C}^+$. By Lemma 7, g is a 1-tolerant polymorphism of $\mathbb{C}$.

Consider the structure $\mathbb{C}' := \mathbb{C}/\mathrm{Aut}(\mathbb{C}^+)$ whose domain consists of the classes of the equivalence relation $\sim_1$ induced by $\mathrm{Aut}(\mathbb{C}^+)$, and such that for every relation symbol R of arity r in the signature of $\mathbb{C}$, one has $(O_1, \dots, O_r) \in R^{\mathbb{C}'}$ if, and only if, there exist $a_1 \in O_1, \dots, a_r \in O_r$ such that $(a_1, \dots, a_r) \in R^{\mathbb{C}}$. Then g induces a 1-tolerant polymorphism of $\mathbb{C}'$: define $\tilde{g}(O_1, \dots, O_n)$ to be the $\sim_1$-class of $g(a_1, \dots, a_n)$, for arbitrary $a_1 \in O_1, \dots, a_n \in O_n$. Since g is canonical with respect to $\mathbb{C}^+$, the definition of $\tilde{g}$ does not depend on the chosen elements $a_1, \dots, a_n$. One readily checks that $\tilde{g}$ thus defined is a 1-tolerant polymorphism of $\mathbb{C}'$.

Moreover, let h be a homomorphism $\mathbb{C} \to \mathbb{B}$. By Theorem 4 applied with $\mathbb{C}^+$ and $\mathbb{B}$, there exists a homomorphism $h'\colon \mathbb{C} \to \mathbb{B}$ that is canonical from $\mathbb{C}^+$ to $\mathbb{B}$. Similarly as above, h' induces a homomorphism $\tilde{h}'$ from $\mathbb{C}'$ to $\mathbb{B}$. Thus we get that $\mathbb{A} \to \mathbb{C}' \to \mathbb{B}$. Moreover $\mathbb{C}'$ has a 1-tolerant polymorphism, so by Theorem 3, $\mathbb{C}'$ has finite duality.

(3) implies (1). By Theorem 3, $\mathrm{CSP}(\mathbb{C})$ can be defined by a first-order sentence Φ. This sentence shows that $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ is solvable in first-order logic. ◀

An anonymous reviewer of this paper provided another proof of the implication from (2) to (3) in Theorem 2 using the sparse incomparability lemma (see, e.g., [29]) to show directly that the duality $\mathcal{F}$ can be taken to consist of trees (which in our case follows from an application of Theorem 3), and then using the fact that finite families of trees admit a finite dual structure [38]. Namely, given an arbitrary finite duality $\mathcal{F}$ for $(\mathbb{A}, \mathbb{B})$, consider the family $\mathcal{G}$ consisting of homomorphic images of structures from $\mathcal{F}$ and that are trees. If $\mathbb{X}$ admits a homomorphism to $\mathbb{A}$, then it is $\mathcal{F}$-free and therefore $\mathcal{G}$-free. Suppose now that $\mathbb{X} \not\to \mathbb{B}$. By the sparse incomparability lemma, one can find a structure $\mathbb{X}'$ such that $\mathbb{X}' \to \mathbb{X}$ and $\mathbb{X}' \not\to \mathbb{B}$, and $\mathbb{X}'$ has girth larger than the size of any structure in $\mathcal{F}$. Since $\mathbb{X}' \not\to \mathbb{B}$, there exists $\mathbb{F} \in \mathcal{F}$ and a homomorphism $h\colon \mathbb{F} \to \mathbb{X}'$, and the image of $\mathbb{F}$ under h must be a tree, which implies that $\mathbb{X}'$ is not $\mathcal{G}$-free. Since $\mathbb{X}' \to \mathbb{X}$, $\mathbb{X}$ is not $\mathcal{G}$-free either.

Thus, the construction of an infinite structure $\mathbb{C}$ as in our proof of Theorem 2 is not necessary; however our method here applies to a wider setting as the next sections show.

We obtain as a corollary to Theorem 2 a characterization of the pair $(\mathbb{A}, \mathbb{B})$ that have finite duality, in the case that $\mathbb{A}$ is a digraph containing a directed cycle.

▶ **Corollary 8.** *Let $(\mathbb{A}, \mathbb{B})$ be a PCSP template where $\mathbb{A}$ is a digraph containing a directed cycle. Then $(\mathbb{A}, \mathbb{B})$ has finite duality if, and only if, $\mathbb{B}$ contains a loop.*

Proof. If $\mathbb{B}$ has a loop, then the empty set is a duality for $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$. Suppose now that $(\mathbb{A}, \mathbb{B})$ has finite duality. By Theorem 2, there exists a finite $\mathbb{C}$ with finite duality and such that $\mathbb{A}$ admits a homomorphism to $\mathbb{C}$ and $\mathbb{C} \to \mathbb{B}$. Since $\mathbb{A}$ has a directed cycle, so does $\mathbb{C}$.

Since $\mathbb{C}$ has finite duality, its duality must consist of trees. Note that every orientation of a tree admits a homomorphism to $\mathbb{C}$ since $\mathbb{C}$ contains a directed cycle, and therefore its duality must be empty. This implies in particular that $\mathbb{C}$ has a loop, and so does $\mathbb{B}$ since $\mathbb{C} \to \mathbb{B}$. ◀

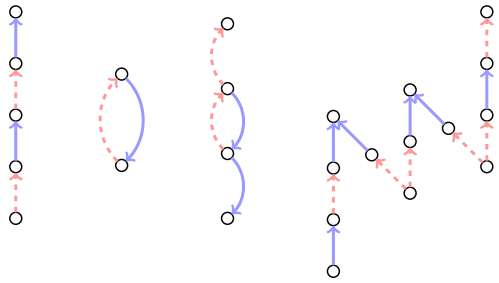


Figure 1 Illustration of the structures $\mathbb{A}$, $\mathbb{C}$, $\mathbb{B}$, and $\mathbb{F}_2$ (from left to right) in Proposition 9. The red dashed arcs correspond to pairs in the relation R, while blue solid arcs correspond to pairs in the relation B.

We conclude this section by showing that there are proper examples of PCSP templates with finite duality.

▶ **Proposition 9.** *There exists a PCSP template* $(\mathbb{A}, \mathbb{B})$ *with finite duality such that neither* $\mathbb{A}$ *nor* $\mathbb{B}$ *has finite duality.*

Proof. Consider the structures in a binary relational signature with two binary symbols R and B displayed in Figure 1. Then $\mathbb{A} \to \mathbb{C} \to \mathbb{B}$ and $\mathbb{C}$ has the duality that consists of the following structures: an R-path of length 2, a B-path of length 2, a vertex with incoming B- and R-edges, and a vertex with outgoing B- and R-edges.

We show that the structure $\mathbb{A}$ does not have finite duality. Let $\mathbb{P}$ be an R-edge followed by a B-edge. We use the notation $+$ and $-$ to denote the obvious amalgamation of copies of $\mathbb{P}$. For every $n \geq 1$, consider the structure $\mathbb{F}_n$ defined by taking $\mathbb{P}+(\mathbb{P}-\mathbb{P})+\cdots+(\mathbb{P}-\mathbb{P})+\mathbb{P}+\mathbb{P}$, with a total of $2n+3$ copies of $\mathbb{P}$, and removing the first vertex and its adjacent R-edge and the last vertex and its adjacent B-edge ($\mathbb{F}_2$ is showed in Figure 1). No $\mathbb{F}_n$ admits a homomorphism to $\mathbb{A}$, although every structure obtained by removing an edge does. Thus, every $\mathbb{F}_n$ must be in a duality for $\mathbb{A}$. Since all the structures $\mathbb{F}_n$ are homomorphically incomparable, we obtain that $\mathbb{A}$ does not have finite duality.

The proof that $\mathbb{B}$ does not have finite duality is similar, where this time one defines $\mathbb{P}$ to be an R-path or a B-path of length 2. ◀

Non-sufficient conditions

For finite-domain CSPs, a number of other conditions are known to be equivalent to the fact that $\mathbb{A}$ has finite duality:

- as mentioned, the existence of a 1-tolerant polymorphism of $\mathbb{A}$,
- the connectivity of a specific graph $L(\mathbb{G}, \mathbb{A}')$ for some retract $\mathbb{A}'$ of $\mathbb{A}$ and all finite structures $\mathbb{G}$ [20],
- the fact that there exists a retract $\mathbb{A}'$ of $\mathbb{A}$ is such that $(\mathbb{A}')^2$ dismantles to its diagonal [36].

It is not clear what could be generalizations of the last two items in the case of promise templates $(\mathbb{A}, \mathbb{B})$. However, the first item has a clear candidate for a generalization, namely the existence of a 1-tolerant polymorphism of $(\mathbb{A}, \mathbb{B})$, i.e., a map $f\colon A^n \to B$ such that for every relation symbol R, and every $\overline{a}^1, \dots, \overline{a}^n$ such that all but at most one are in $R^{\mathbb{A}}$, then $f(\overline{a}^1, \dots, \overline{a}^n)$ is in $R^{\mathbb{B}}$.

We remark that a 1-tolerant polymorphism of arity n of a structure $\mathbb{C}$ can be composed with itself to obtain a 1-tolerant polymorphism of any arity $m \geq n$. Thus, in the case of CSPs, first-order solvability can also be characterized by the existence of 1-tolerant polymorphisms of all but finitely many arities.

▶ **Proposition 10.** *Let $(\mathbb{A},\mathbb{B})$ be a PCSP template. If $(\mathbb{A},\mathbb{B})$ has finite duality, then it has a 1-tolerant polymorphism. However, there exists a PCSP template with 1-tolerant polymorphisms of all arities but not finite duality.*

Proof. Suppose that $(\mathbb{A},\mathbb{B})$ has finite duality. By Theorem 2, there exists a finite structure $\mathbb{C}$ with finite duality and homomorphisms $h\colon \mathbb{A}\to\mathbb{C}$ and $g\colon \mathbb{C}\to\mathbb{B}$. By Theorem 3, $\mathbb{C}$ has a 1-tolerant polymorphism $f\colon \mathbb{C}^n\to\mathbb{C}$. The composition $g\circ f\circ h$ is then a 1-tolerant polymorphism of $(\mathbb{A},\mathbb{B})$.

Consider now $\mathbb{A}:=K_c$ and $\mathbb{B}:=K_{c^2}$ to be complete graphs on c and c^2 vertices, for any $c\geq 2$. Let $n\geq 1$. We simply name the vertices of $\mathbb{B}$ by pairs (a,b) of elements of $\mathbb{A}$. Then the map $f\colon A^n\to B$ defined by $f(a_1,\dots,a_n):=(a_1,a_2)$ is a 1-tolerant n-ary polymorphism of $(\mathbb{A},\mathbb{B})$: if $(a_1,b_1),\dots,(a_n,b_n)$ are pairs such that $a_j\neq b_j$ for all but at most one j, then $(a_1,a_2)\neq(b_1,b_2)$. However, it follows from Corollary 8 that $(\mathbb{A},\mathbb{B})$ does not have finite duality, since $\mathbb{A}$ has a directed cycle of length 2 and $\mathbb{B}$ does not have a loop. ◀

4 Local Consistency for PCSPs

We now apply the same reasoning as in the previous section to characterize the power of the arc consistency reduction, recently introduced in [28, 33]. For this reduction, we need the following concepts.

Arc consistency is a polynomial-time algorithm that takes as input a structure $\mathbb{X}$, as an instance of $\mathrm{CSP}(\mathbb{A})$, and that computes for every $x\in X$ a set $P_x\subseteq A$, and for every constraint C of the form $(x_1,\dots,x_n)\in R^{\mathbb{X}}$ a set $Q_C\subseteq R^{\mathbb{A}}$ such that:

1. for every constraint C whose ith variable is $x_i\in X$, the ith projection of Q_C is equal to P_{x_i},
2. for every homomorphism $h\colon\mathbb{X}\to\mathbb{A}$ and every $x\in X$, we have $h(x)\in P_x$. In particular, if P_x is empty then $\mathbb{X}$ does not admit a homomorphism to $\mathbb{A}$.

A minion is a functor $\mathscr{M}$ from finite sets to sets: for every finite set X, one has a set $\mathscr{M}X$ and for every function $\sigma\colon X\to Y$, one has a function $\mathscr{M}\sigma\colon \mathscr{M}X\to\mathscr{M}Y$, such that $\mathscr{M}\mathrm{id}_X=\mathrm{id}_{\mathscr{M}X}$ for all X and $\mathscr{M}(\sigma\circ\tau)=\mathscr{M}\sigma\circ\mathscr{M}\tau$ whenever the composition of σ and τ is well defined.

A *minor identity* is a formal statement of the form $f^\sigma\approx g^\tau$ where f is a symbol of type X, g is a symbol of type Y, $\sigma\colon X\to Z$ and $\tau\colon Y\to Z$ are functions, and X,Y,Z are arbitrary finite sets. A *minor condition* is a set Σ of minor identities. A minor condition Σ is *satisfied* in $\mathscr{M}$ if the symbols in Σ of any type X can be mapped to elements of $\mathscr{M}X$ such that if $f^\sigma\approx g^\tau$ is in Σ, then $(\mathscr{M}\sigma)(f)=(\mathscr{M}\tau)(g)$. The set $\mathrm{Pol}(\mathbb{A},\mathbb{B})$ can be seen to be such a minion $\mathscr{M}$, where $\mathscr{M}X$ consists of the homomorphisms $\mathbb{A}^X\to\mathbb{B}$, and the functions $\mathscr{M}\sigma$ are obtained by identifying arguments of such homomorphisms according to σ.

The problem $\mathrm{PMC}_N(\mathscr{M})$ is the problem taking as input a minor condition Σ whose symbols have sorts of size at most N, whose yes-instances are those Σ that can be satisfied in *every* minion, and whose no-instances are those that are not satisfiable in $\mathscr{M}$. It is known that for N large enough only depending on the size of $\mathbb{A}$ and the size of its relations, $\mathrm{PMC}_N(\mathrm{Pol}(\mathbb{A},\mathbb{B}))$ and $\mathrm{PCSP}(\mathbb{A},\mathbb{B})$ are equivalent under logspace reductions [8]. In the following, we drop the subscript and always take N large enough for this equivalence to hold. The arc consistency reduction described below is a complete reduction from $\mathrm{PCSP}(\mathbb{A},\mathbb{B})$ to $\mathrm{PMC}(\mathscr{M})$ for a minion $\mathscr{M}$, although it is not necessarily sound; by the previous sentence, this is essentially the same as trying to reduce from one PCSP to another.

This reduction applied to an input $\mathbb{X}$ of $\mathrm{PCSP}(\mathbb{A},\mathbb{B})$, and whose output is an instance of $\mathrm{PMC}(\mathscr{M})$, works as follows:

- First, apply the arc consistency algorithm to $\mathbb{X}$, seen as an instance of CSP($\mathbb{A}$); one obtains a family of subsets $(P_x)_{x\in X}$ and (Q_C) satisfying the arc consistency condition above,
- Associate a function symbol f_x with every $x \in X$, whose arguments are labelled by the elements in P_x; associate a function symbol f_C with every constraint $C = R(x_1, \dots, x_k)$, whose arguments are labelled by the tuples in Q_C;
- Output the minor condition containing the identities $f_{x_i} \approx f_C^\sigma$ where $\sigma\colon Q_C \to P_{x_i}$ is the projection on the ith component, as an input to PMC($\mathscr{M}$).

We turn the set of instances $\mathbb{X}$ that are not rejected by this reduction into a class with the amalgamation property. The additional symbols $H_{(P,f)}$ are indexed by pairs (P, f) where P is a non-empty subset of A or of $R^{\mathbb{A}}$ for some R, and f is an element of $\mathscr{M}P$. Let $\mathbb{X}$ be an arbitrary finite structure such that there exist a family of subsets $(P_x)_{x\in X}$ and (Q_C) satisfying the arc consistency condition (Item 1) together with a map ξ witnessing the satisfiability of the corresponding minor condition Σ in $\mathscr{M}$. Let $\mathbb{X}^*$ be the expansion of $\mathbb{X}$ where:

- $x \in H_{(P,f)}$ iff $P = P_x$ and $\xi(f_x) = f$,
- for every constraint $(x_1, \dots, x_n) \in R^{\mathbb{X}}$, we let $(x_1, \dots, x_n) \in H_{(Q,g)}$ iff $Q_C = Q$ and $\xi(f_C) = g$.

We say that $\mathbb{X}^*$ is a *valid encoding*, witnessed by the sets $(P_x), (Q_C)$ and the map ξ. Let $\mathcal{C}_{red}$ be the class of valid encodings.

▶ **Proposition 11.** *$\mathcal{C}_{red}$ is closed under substructures and has the strong amalgamation property.*

Proof. The fact that $\mathcal{C}_{red}$ is closed under substructures is readily checked.

Let now $\mathbb{X}^*, \mathbb{Y}_1^*, \mathbb{Y}_2^* \in \mathcal{C}_{red}$ and embeddings $f_i\colon \mathbb{X}^* \to \mathbb{Y}_i^*$. Without loss of generality, we can suppose that $X \subseteq Y_1, Y_2$ and that f_i is the inclusion map. Take $\mathbb{Z}^*$ to be the free amalgam of $\mathbb{Y}_1^*$ and $\mathbb{Y}_2^*$ over $\mathbb{X}^*$. We write $\mathbb{Z}$ for the reduct of $\mathbb{Z}^*$ to the signature of $\mathbb{A}$. Let $(P_y^i), (Q_C^i), \xi^i$ be the witnesses for the fact that $\mathbb{Y}_i^*$ is a valid encoding, for $i \in \{1, 2\}$.

Let $P_z := P_z^i$ if $z \in Y_i$; this is well defined since if $z \in Y_1 \cap Y_2 = X$, then we have $P_z^1 = P_z^2$. If C is a constraint $(z_1, \dots, z_n) \in R^{\mathbb{Z}}$, then by definition $\{z_1, \dots, z_n\} \subseteq Y_i$ for some $i \in \{1, 2\}$. Let $Q_C := Q_C^i$, and note again that if $\{z_1, \dots, z_n\} \subseteq Y_1 \cap Y_2$ then $Q_C^1 = Q_C^2$. Define similarly ξ, where Σ is the minor condition arising from the sets (P_z) and (Q_C), by defining $\xi(f_z) := \xi^i(f_z)$ and $\xi(f_C) := \xi^i(f_C)$ for a suitable i.

We prove that the family of sets $(P_z), (Q_C)$ satisfies Item 1. If C is a constraint $(z_1, \dots, z_n) \in R^{\mathbb{Z}}$, then by definition $\{z_1, \dots, z_n\} \subseteq Y_i$ for some i. Thus, since $(P_y^i), (Q_C^i)$ satisfies Item 1, the projection of Q_C^i to each component coincides with the corresponding P_z^i, and we are done.

Similarly, it is easy to see that ξ witnesses that Σ is satisfiable in $\mathscr{M}$. Thus, $\mathbb{Z}^*$ is a valid encoding, witnessed by the sets $(P_z), (Q_C)$, and ξ. ◀

Consider the Fraïssé limit $\mathrm{AC}(\mathbb{A}, \mathscr{M})^*$ of $\mathcal{C}_{red}$ and its reduct $\mathrm{AC}(\mathbb{A}, \mathscr{M})$ to the signature of $\mathbb{A}$. One obtains a structure that gives a classification of PCSPs for which the arc consistency reduction correctly solves the problem.

▶ **Theorem 12.** *Let $(\mathbb{A}, \mathbb{B})$ be a PCSP template. The following hold:*

- *There exists a homomorphism $\mathbb{A} \to \mathrm{AC}(\mathbb{A}, \mathscr{M})$,*
- *For every finite structure $\mathbb{B}$, arc consistency is a correct reduction from* PCSP($\mathbb{A}, \mathbb{B}$) *to* PMC($\mathscr{M}$) *if, and only if, there exists a homomorphism $\mathrm{AC}(\mathbb{A}, \mathscr{M}) \to \mathbb{B}$.*

Proof. Since $\mathbb{A}$ is not rejected by the arc consistency reduction, there exists a valid encoding $\mathbb{A}^* \in \mathcal{C}_{red}$, and therefore $\mathbb{A}^*$ embeds into $\mathrm{AC}(\mathbb{A}, \mathscr{M})^*$. It follows that $\mathbb{A}$ embeds into $\mathrm{AC}(\mathbb{A}, \mathscr{M})$.

If arc consistency is a correct reduction, then all the structures that have a valid encoding in $\mathcal{C}_{red}$ admit a homomorphism to $\mathbb{B}$; thus, a compactness argument gives that $\mathrm{AC}(\mathbb{A}, \mathscr{M})$ admits a homomorphism to $\mathbb{B}$.

For the other direction, we remark that the reduction is always *complete*, i.e., every yes-instance of $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ is mapped to a yes-instance of $\mathrm{PMC}(\mathscr{M})$. Conversely, suppose that the minor condition Σ that is computed by the algorithm on an instance $\mathbb{X}$ admits a solution in $\mathscr{M}$. We give a homomorphism $\mathbb{X} \to \mathbb{B}$ as follows. Let $\mathbb{X}^* \in \mathcal{C}_{red}$ be a valid encoding, which exists since $\mathbb{X}$ is not rejected by the reduction. We get an embedding $\mathbb{X}^* \to \mathrm{AC}(\mathbb{A}, \mathscr{M})^*$, giving a homomorphism $\mathbb{X} \to \mathrm{AC}(\mathbb{A}, \mathscr{M})$, which can then be composed with the homomorphism $\mathrm{AC}(\mathbb{A}, \mathscr{M}) \to \mathbb{B}$ that exists by assumption, from which we get a homomorphism $\mathbb{X} \to \mathbb{B}$. ◀

Note that if the number of elements in $\mathscr{M}P$ is bounded for every subset P of A or of a relation of A, then $\mathrm{AC}(\mathbb{A}, \mathscr{M})^*$ is homogeneous in a finite language, and is therefore ω-categorical. By [32, Theorem 2.11], the expansion of $\mathrm{AC}(\mathbb{A}, \mathscr{M})^*$ by a free linear order is a Ramsey structure. Let $\mathscr{G}$ be its automorphism group. Thus, the existence of a homomorphism $\mathrm{AC}(\mathbb{A}, \mathscr{M}) \to \mathbb{B}$ is equivalent, by Theorem 4, to the existence of a homomorphism $\mathrm{AC}(\mathbb{A}, \mathscr{M})/\mathscr{G} \to \mathbb{B}$, which can be effectively tested when $\mathrm{AC}(\mathbb{A}, \mathscr{M})/\mathscr{G}$ is finite. The domain of $\mathbb{D} := \mathrm{AC}(\mathbb{A}, \mathscr{M})/\mathscr{G}$ consists, by homogeneity, of pairs (P, f) where P is a non-empty subset of A and f is in $\mathscr{M}P$. Moreover, if R is an n-ary relation symbol, then $((P_1, f_1), \dots, (P_n, f_n)) \in R^{\mathbb{D}}$ if, and only if, there exists a $g \in \mathscr{M}(R^{\mathbb{A}} \cap (P_1 \times \cdots \times P_n))$ such that $(\mathscr{M}\sigma_i)(g) = f_i$ holds for all $i \in \{1, \dots, n\}$, where $\sigma_i \colon R^{\mathbb{A}} \cap (P_1 \times \cdots \times P_n) \to P_i$ is the ith projection. Provided that the elements of $\mathscr{M}$ can be presented to an algorithm in some way, then the structure $\mathbb{D}$ can be computed according to the definition above, and the existence of a homomorphism $\mathbb{D} \to \mathbb{B}$ can be tested. This gives a decision procedure to check whether the arc consistency reduction to $\mathscr{M}$ solves a given PCSP.

It is noted in [28] that arc consistency as an algorithm solving $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ can be seen as a reduction from $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ to $\mathrm{PMC}(\mathscr{M}_0)$, where $\mathscr{M}_0$ is the minion consisting of all operations of finite arity on a 1-element set (i.e., for every n, $\mathscr{M}_0$ contains a single function of arity n). In this case, the elements of the finite structure $\mathbb{D}$ correspond exactly to non-empty subsets of $\mathbb{A}$, and we obtain another proof of the characterization of the power of the arc consistency algorithm for PCSPs by means of the powerset structure defined by Feder and Vardi [29].

The arc consistency procedure can be generalized to the *k-consistency algorithm* by computing sets $P_{x_1,\dots,x_k} \subseteq A^k$ for every k-tuple of elements from $\mathbb{X}$, and asking for similar conditions as in Item 1. We say that a PCSP template $(\mathbb{A}, \mathbb{B})$ has *width k* if every instance $\mathbb{X}$ of $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ that is not rejected by the k-consistency algorithm (when seeing $\mathbb{X}$ as an instance of $\mathrm{CSP}(\mathbb{A})$) admits a homomorphism to $\mathbb{B}$.

When a PCSP template has bounded width (i.e., has width k for some k) then the corresponding PCSP can be solved in polynomial time. It is hitherto not known whether the search version of the PCSP can then be solved in polynomial time. The existence of a structure $\mathbb{C}$ that would allow us to follow the same line of reasoning as for arc consistency is open. Atserias and Toruńczyk [5] proved that the class of locally consistent systems of linear equations over $\mathbb{Z}_2$ cannot be turned into a class with the amalgamation property using an expansion by finitely many relations (i.e., this class is not *homogenizable*). However, in order to obtain a structure $\mathbb{A}'$ playing the role of $\mathrm{AC}(\mathbb{A}, \mathscr{M}_0)$ in Theorem 12 for any $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$

that is solvable by, say, 3-consistency, it is plausible that one only needs the number of orbits of elements of $\mathbb{A}'$ to be finite, a much weaker condition than homogeneity in a finite relational language, or even ω-categoricity.

5 The Basic Linear Programming Relaxation

The *basic linear programming relaxation* of an instance $\mathbb{X}$ of CSP($\mathbb{A}$) is the following linear program with variables $\lambda_x(a)$ for $x \in \mathbb{X}$ and $a \in \mathbb{A}$:

$$\begin{array}{rcll} \sum_{a\in A} \lambda_x(a) & = & 1 & \text{for all } x \in X \\ \sum_{\overline{a}\in R^{\mathbb{A}}} \lambda_C(\overline{a}) & = & 1 & \text{for all constraints } C := \overline{y} \in R^{\mathbb{X}} \\ \sum_{\overline{a}:a_i=b} \lambda_C(\overline{a}) & = & \lambda_x(b) & \text{for all } x, b, \text{ and } \overline{y} \text{ s.t. } y_i = x \\ \lambda_x(a), \lambda_C(\overline{a}) & \geq & 0 & \text{for all variables} \end{array} \qquad (\mathrm{BLP}(\mathbb{X}, \mathbb{A}))$$

Note that if $\mathbb{X}$ admits a homomorphism to $\mathbb{A}$, then the λ's can be taken to have values in $\{0, 1\}$ and to describe completely a homomorphism $\mathbb{X} \to \mathbb{A}$. However, there can be proper solutions to the program BLP($\mathbb{X}, \mathbb{A}$) that do not correspond to any homomorphism.

We say that PCSP($\mathbb{A}, \mathbb{B}$) is *solvable by BLP* if whenever BLP($\mathbb{X}, \mathbb{A}$) has a solution for a given $\mathbb{X}$, then $\mathbb{X}$ admits a homomorphism to $\mathbb{B}$. Note that if PCSP($\mathbb{A}, \mathbb{B}$) is solvable by BLP, then it is in particular solvable in polynomial time. It is proven in [8] that PCSP($\mathbb{A}, \mathbb{B}$) is solvable by BLP if, and only if, a certain structure LP($\mathbb{A}$) whose domain is the set of probability distributions on A admits a homomorphism to $\mathbb{B}$. It is unknown whether it is always true that a PCSP($\mathbb{A}, \mathbb{B}$) that is solvable by BLP is also polynomially solvable in its search variant. Moreover, it is not known whether the BLP-solvability of a given PCSP is decidable.

The λ's represent probability distributions on $\mathbb{A}$ and $R^{\mathbb{A}}$ that are required to be consistent; one sees that the supports of any solution to BLP($\mathbb{X}, \mathbb{A}$) forms a family of sets P_x, Q_C satisfying the arc consistency condition. Thus, the BLP relaxation is more powerful than arc consistency.

The approach used in the previous sections can be applied here, by encoding the probability distributions λ's arising from a solution to BLP($\mathbb{X}, \mathbb{A}$) as suitable relations over $\mathbb{X}$ using additional symbols $P_{\overline{a},q}$ for tuples a from A and $q \in \mathbb{Q} \cap [0, 1]$ to encode the probability distributions λ's. This gives a class $\mathcal{C}_{\mathrm{BLP}}$ of structures that has the amalgamation property. Let $\mathbb{C}^*_{\mathrm{BLP}}$ be the Fraïssé limit of $\mathcal{C}_{\mathrm{BLP}}$, and let $\mathbb{C}_{\mathrm{BLP}}$ be its reduct of the signature of $\mathbb{A}$.

▶ **Proposition 13.** *Let $(\mathbb{A}, \mathbb{B})$ be a PCSP template. The following are equivalent:*

- PCSP($\mathbb{A}, \mathbb{B}$) *is solvable by BLP,*
- *there exists a homomorphism* $\mathbb{C}_{\mathrm{BLP}} \to \mathbb{B}$.

We note that infinitely many new predicates are required for this encoding, and therefore $\mathbb{C}_{\mathrm{BLP}}$ is not ω-categorical, just as is the case of LP($\mathbb{A}$) [8]. In fact, LP($\mathbb{A}$) can be seen to be homomorphically equivalent to $\mathbb{C}_{\mathrm{BLP}}$, using the natural correspondence between probability distributions on A and the orbits of $\mathbb{C}^*_{\mathrm{BLP}}$, which are described by the unary predicates $P_{a,q}$ for $a \in A$ and $q \in \mathbb{Q} \cap [0, 1]$.

However, for every $N \geq 1$, one can consider the class $\mathcal{C}^{(N)}_{\mathrm{BLP}}$ of structures $\mathbb{X}$ endowed with rational probability distributions where no denominator is greater than N. Every $\mathcal{C}^{(N)}_{\mathrm{BLP}}$ has the amalgamation property, and therefore a Fraïssé limit $\mathbb{C}^{*,N}_{\mathrm{BLP}}$. Since the language is now finite, each $\mathbb{C}^{*,N}_{\mathrm{BLP}}$ is ω-categorical, and for every $N < M$ we have embeddings $\mathbb{C}^{*,N}_{\mathrm{BLP}} \hookrightarrow \mathbb{C}^{*,M}_{\mathrm{BLP}}$. We thus get the following refinement of Proposition 13.

▶ **Proposition 14.** *Let* $(\mathbb{A}, \mathbb{B})$ *be a PCSP template. The following are equivalent:*

- $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ *is solvable by BLP,*
- $\mathbb{A} \to \mathbb{C}^1_{\mathrm{BLP}} \to \mathbb{C}^2_{\mathrm{BLP}} \to \cdots \to \mathbb{C}_{\mathrm{BLP}} \to \mathbb{B}$.

Once more, [32, Theorem 2.11] applies and gives that each $\mathbb{C}^N_{\mathrm{BLP}}$ admits a homogeneous Ramsey expansion by finitely many relations. It is not immediately clear if this observation and Proposition 14 can be used to derive a decision procedure for solvability of a PCSP by BLP.

6 The Affine Integer Relaxation

Similarly as in Section 5, we can obtain a limit structure characterizing the power of the so-called affine integer relaxation (AIP) [8]. Given an input $\mathbb{X}$ to $\mathrm{CSP}(\mathbb{A})$, the system $\mathrm{AIP}(\mathbb{X}, \mathbb{A})$ is like $\mathrm{BLP}(\mathbb{X}, \mathbb{A})$ except that the variables are integer-valued. If $\mathbb{X} \to \mathbb{A}$, then the same $\{0, 1\}$ solution to $\mathrm{BLP}(\mathbb{X}, \mathbb{A})$ is a solution to $\mathrm{AIP}(\mathbb{X}, \mathbb{A})$, and we say that AIP solves $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ if whenever a solution to $\mathrm{AIP}(\mathbb{X}, \mathbb{A})$ exists, then $\mathbb{X} \to \mathbb{B}$. The power of AIP to solve PCSPs has been characterized in [8] by means of the existence of a structure $\mathrm{IP}(\mathbb{A})$, similarly as $\mathrm{LP}(\mathbb{A})$ characterizes the power of BLP.

The class of structures $\mathbb{X}$ for which $\mathrm{AIP}(\mathbb{X}, \mathbb{A})$ has a solution can be expanded by relations encoding, for each $\mathbb{X}$, a possible solution to $\mathrm{AIP}(\mathbb{X}, \mathbb{A})$. The resulting class $\mathcal{C}_{\mathrm{AIP}}$ of structures has the amalgamation property (where the amalgam is always free), and therefore it has a Fraïssé limit $\mathbb{C}^*_{\mathrm{AIP}}$, whose reduct $\mathbb{C}_{\mathrm{AIP}}$ to the signature of $\mathbb{A}$ characterizes solvability by AIP.

▶ **Proposition 15.** *Let* $(\mathbb{A}, \mathbb{B})$ *be a PCSP template. The following are equivalent:*

- $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ *is solvable by AIP,*
- *there exists a homomorphism* $\mathbb{C}_{\mathrm{AIP}} \to \mathbb{B}$.

7 Further connections to infinite-domain CSPs

We conclude this paper by hinting at further connections between algorithms solving PCSPs and infinite-domain CSPs.

Sandwiches and Monotone Algorithms

All current polynomial-time algorithms for solving or reducing PCSPs feature the use of algorithms solving CSPs either via a trivial reduction, via computationally simple many-one reductions, or as oracles.

In the first case, one solves $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ through a "trivial" reduction to a problem of the form $\mathrm{CSP}(\mathbb{C})$, where the reduction does not transform the input. By definition, this "do-nothing" reduction is valid if, and only if, there exists a homomorphism from $\mathbb{A}$ to $\mathbb{C}$ and from $\mathbb{C}$ to $\mathbb{B}$, which we denote by $\mathbb{A} \to \mathbb{C} \to \mathbb{B}$. Because of this characterization, characterizing when a "do-nothing" reduction is a valid reduction from $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ to $\mathrm{CSP}(\mathbb{C})$ relies on studying the templates that are *sandwiched* between $\mathbb{A}$ and $\mathbb{B}$ in the homomorphism preorder. Barto [7] and Barto and Asimi [1] have showed examples of problems of the form $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ where such a $\mathbb{C}$ with $\mathrm{CSP}(\mathbb{C})$ polynomial-time tractable exists but cannot be taken to be a finite structure. As explained in Sections 5 and 6, the tractability of PCSPs that are solvable by relaxations like the basic linear relaxation and integer affine relaxation can also be explained by sandwiches, taking $\mathbb{C}$ to be a reduct of a given structure whose domain consists of tuples of rational or integer numbers.

A second type of algorithms solving PCSPs consists in having a computationally simple, but non-trivial, reduction to a CSP. The power of reductions known as *gadget reductions* is completely classified (even where the target problem is itself a PCSP). As in the case of CSPs, the existence of a gadget reduction between two PCSPs is equivalent to the existence of a certain type of map between the sets of polymorphisms of the corresponding templates. A more powerful type of reductions, called *k-reductions* or *k-consistency reductions*, has emerged recently [33, 28] and the computational power of such reductions is still unclear.

The computationally more powerful algorithms leverage algorithms for CSPs as blackboxes, mainly using the solvability of linear programming over $\mathbb{Q}$ or of linear diophantine equations over $\mathbb{Z}$. This is for example the case of CLAP [23], BLP+AIP [19], and cohomological consistency [27].

For each of the three types of algorithms or reductions proposed above, it is in general hard to characterize the power of the respective methods, and in particular it is hard to prove that a given approach does *not* solve a given PCSP, even in concrete cases. For example, considerable effort has been put recently into proving that specific polynomial-time algorithms do not solve $\mathrm{PCSP}(K_s, K_c)$ for $c \geq s > 2$ [25, 26], which is conjectured to be NP-hard and therefore should not be solvable by *any* polynomial-time algorithm if $\mathsf{P} \neq \mathsf{NP}$. Beyond the main result of this paper, the thesis we put forward here is that the tools from various fields of logic can be used to study these questions in a more general setting than Theorem 2.

We note that by allowing arbitrary logspace reductions, one can prove the tractability of *every* PCSP by a reduction to a CSP (potentially with an infinite template).

▶ **Observation 16.** *For every PCSP template* $(\mathbb{A}, \mathbb{B})$ *such that* $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ *is in* P*, there exists a structure* $\mathbb{C}$ *such that* $\mathrm{CSP}(\mathbb{C})$ *is in* P *and such that* $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ *reduces to* $\mathrm{CSP}(\mathbb{C})$ *in logspace.*

Proof. Let L be the set of instances accepted by a given polynomial-time algorithm solving $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$. By [11, Theorem 1], there exists a structure $\mathbb{C}$ such that L admits a logspace reduction to $\mathrm{CSP}(\mathbb{C})$, and $\mathrm{CSP}(\mathbb{C})$ admits a polynomial-time Turing reduction to L (and is therefore in P). Since $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ reduces to L (by a trivial reduction), we have that $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ reduces to $\mathrm{CSP}(\mathbb{C})$. ◀

We mention that while every finite-domain PCSP reduces to a problem in NP, this is not the case for infinite-domain CSPs, even for decidable ones (as there exist, e.g., NEXPTIME-complete CSPs [31]). Thus, it is probable that structural restrictions can be imposed on the infinite templates appearing in Observation 16, so that their CSPs are still able to "solve" all the finite-domain PCSPs. While the assumption that $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ is in P is difficult to use, it seems that assuming that the tractability of $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ comes from specific polynomial-time algorithms *does* allow us to provide better constructions for $\mathbb{C}$, as in the cases explored in the previous sections.

The power of the sandwich approach

Provided that $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ is solvable by a "natural" algorithm, we even obtain a structure $\mathbb{C}$ such that $\mathrm{CSP}(\mathbb{C})$ is in P and such that $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ reduces to $\mathrm{CSP}(\mathbb{C})$ by the do-nothing reduction, i.e., $\mathbb{A} \to \mathbb{C} \to \mathbb{B}$. We call an algorithm M *natural* if it satisfies the following conditions:

1. for every finite $\mathbb{X}$, M accepts the input $\mathbb{X}$ iff M accepts all the connected components of $\mathbb{X}$,
2. for every finite $\mathbb{X}, \mathbb{Y}$ such that M accepts $\mathbb{Y}$ and such that $\mathbb{X}$ admits a homomorphism to $\mathbb{Y}$, M accepts $\mathbb{X}$.

Let us call M *monotone* if it satisfies Item 2.[5] If $\mathrm{PCSP}(\mathbb{A},\mathbb{B})$ is solvable by a monotone polynomial-time algorithm M, then there exists a polynomial-time algorithm M' satisfying Item 1, making polynomially many calls to M, and such that M' solves $\mathrm{PCSP}(\mathbb{A},\mathbb{B})$. Indeed, on input $\mathbb{X}$, M' simply calls M on all the connected components of $\mathbb{X}$ and accepts $\mathbb{X}$ if all its connected components are accepted by M. Note that M' still solves $\mathrm{PCSP}(\mathbb{A},\mathbb{B})$ since for every $\mathbb{X}$, whether $\mathbb{X}$ homomorphically maps to $\mathbb{A}$ or $\mathbb{B}$ only depends on whether all its connected components do.

▶ **Observation 17.** *Let $(\mathbb{A},\mathbb{B})$ be a PCSP template. The following are equivalent:*

1. *There exists a structure $\mathbb{C}$ such that $\mathbb{A}\to\mathbb{C}\to\mathbb{B}$ and such that $\mathrm{CSP}(\mathbb{C})$ is in* P*,*
2. *$\mathrm{PCSP}(\mathbb{A},\mathbb{B})$ is solvable by a monotone polynomial-time algorithm.*

Proof. The implication from Item 1 to Item 2 is immediate, as any algorithm solving $\mathrm{CSP}(\mathbb{C})$ must be natural, and $\mathrm{PCSP}(\mathbb{A},\mathbb{B})$ reduces to $\mathrm{CSP}(\mathbb{C})$ by a trivial reduction. This gives a monotone algorithm solving $\mathrm{PCSP}(\mathbb{A},\mathbb{B})$.

Suppose now that $\mathrm{PCSP}(\mathbb{A},\mathbb{B})$ is solvable by a monotone polynomial-time algorithm M. By the argument above Observation 17, one can even assume that M is natural. Let $\mathbb{C}$ be the disjoint union of all the finite structures $\mathbb{X}$ such that M accepts $\mathbb{X}$. Since M needs to accept $\mathbb{A}$, we have $\mathbb{A}\to\mathbb{C}$. Moreover, since every accepted structure admits a homomorphism to $\mathbb{B}$, a compactness argument gives that $\mathbb{C}\to\mathbb{B}$.

Note that for every finite $\mathbb{X}$, M accepts $\mathbb{X}$ iff $\mathbb{X}\to\mathbb{C}$. Indeed, if $\mathbb{X}$ is accepted by M then it is even an induced substructure of $\mathbb{C}$. Conversely, if $\mathbb{X}$ has a homomorphism to $\mathbb{M}$, every connected component of $\mathbb{X}$ admits a homomorphism to a structure $\mathbb{Y}$ such that $\mathbb{Y}$ is accepted by M. By assumption, this means that $\mathbb{X}$ is accepted by M. Thus, $\mathrm{CSP}(\mathbb{C})$ is solved by M.

Finally, note that the trivial reduction $\mathbb{X}\mapsto\mathbb{X}$ is a reduction from $\mathrm{PCSP}(\mathbb{A},\mathbb{B})$ to $\mathrm{CSP}(\mathbb{C})$. ◀

It was conjectured in [17] that every tractable PCSP must sandwich a tractable CSP. By Observation 17, this is equivalent to conjecturing that every polynomial-time tractable PCSP can be solved by a *monotone* algorithm running in polynomial time. Since all the known algorithms for PCSPs satisfy the condition of being natural, we see that the sandwich approach, although looking at first sight more limited than the other two mentioned approaches, is in fact currently also the most general.

8 Conclusion

We have provided in Sections 4–7 results relating the power of certain algorithms and reductions to solve PCSPs that has already been studied in the literature [8, 28, 19] providing characterizations of the applicability of a given algorithm by properties of the polymorphisms of the PCSP templates. We give here a logical take on the problem of characterizing the power of these algorithms. The general approach that we give has the advantage that it is fairly automatic: given the description of an algorithm, it is quite immediate to encode the inputs of the algorithm that are accepted as a class of relational structures, and study the potential generic objects for such a class. Remarkably, the powerful Ramsey theorem of [32] provides "out-of-the-box" strong combinatorial properties for these objects that can be potentially be used to prove tractability properties for the search variant of PCSPs, as well as decidable conditions for the applicability of a given algorithm.

[5] This can be seen as a special case of the *monotone reductions* described in [34].

The algorithms that arise as "higher levels" of algorithms described here (e.g., k-consistency as a higher level of arc consistency, kth level of the Sherali-Adams hierarchy as a "higher level" version of BLP) escape both the algebraic methods and the methods presented here. We note that [24] have algebraic characterizations of the instances that are accepted by a given algorithm, however this does not answer the question of which PCSP templates have the property that *all* their instances that are accepted by this algorithm have a solution.

We are thankful for the numerous comments by anonymous reviewers that helped improve the quality of this paper.

References

1. Kristina Asimi and Libor Barto. Finitely tractable promise constraint satisfaction problems. In Filippo Bonchi and Simon J. Puglisi, editors, *46th International Symposium on Mathematical Foundations of Computer Science, MFCS 2021, August 23-27, 2021, Tallinn, Estonia*, volume 202 of *LIPIcs*, pages 11:1–11:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.MFCS.2021.11.
2. Albert Atserias. On digraph coloring problems and treewidth duality. *Eur. J. Comb.*, 29(4):796–820, 2008. doi:10.1016/j.ejc.2007.11.004.
3. Albert Atserias, Andrei A. Bulatov, and Anuj Dawar. Affine systems of equations and counting infinitary logic. *Theor. Comput. Sci.*, 410(18):1666–1683, 2009. doi:10.1016/j.tcs.2008.12.049.
4. Albert Atserias and Víctor Dalmau. Promise constraint satisfaction and width. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9–12, 2022*, pages 1129–1153. SIAM, 2022. doi:10.1137/1.9781611977073.48.
5. Albert Atserias and Szymon Toruńczyk. Non-homogenizable classes of finite structures. In Jean-Marc Talbot and Laurent Regnier, editors, *25th EACSL Annual Conference on Computer Science Logic, CSL 2016, August 29 – September 1, 2016, Marseille, France*, volume 62 of *LIPIcs*, pages 16:1–16:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.CSL.2016.16.
6. Per Austrin, Venkatesan Guruswami, and Johan Håstad. $(2+\epsilon)$-Sat is NP-hard. *SIAM J. Comput.*, 46(5):1554–1573, 2017. doi:10.1137/15M1006507.
7. Libor Barto. Promises make finite (constraint satisfaction) problems infinitary. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*, pages 1–8. IEEE, 2019. doi:10.1109/LICS.2019.8785671.
8. Libor Barto, Jakub Bulin, Andrei A. Krokhin, and Jakub Opršal. Algebraic approach to promise constraint satisfaction. *J. ACM*, 68(4):28:1–28:66, 2021. doi:10.1145/3457606.
9. Libor Barto and Marcin Kozik. Constraint satisfaction problems solvable by local consistency methods. *J. ACM*, 61(1):3:1–3:19, 2014. doi:10.1145/2556646.
10. Libor Barto and Marcin Kozik. Combinatorial gap theorem and reductions between promise CSPs. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9–12, 2022*, pages 1204–1220. SIAM, 2022. doi:10.1137/1.9781611977073.50.
11. Manuel Bodirsky and Martin Grohe. Non-dichotomies in constraint satisfaction complexity. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II – Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, volume 5126 of *Lecture Notes in Computer Science*, pages 184–196. Springer, 2008. doi:10.1007/978-3-540-70583-3_16.
12. Manuel Bodirsky, Martin Hils, and Barnaby Martin. On the scope of the universal-algebraic approach to constraint satisfaction. In *Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science, LICS 2010, 11-14 July 2010, Edinburgh, United Kingdom*, pages 90–99. IEEE Computer Society, 2010. doi:10.1109/LICS.2010.13.

13 Manuel Bodirsky, Florent R. Madelaine, and Antoine Mottet. A proof of the algebraic tractability conjecture for monotone monadic SNP. *SIAM J. Comput.*, 50(4):1359–1409, 2021. doi:10.1137/19M128466X.

14 Manuel Bodirsky, Antoine Mottet, Miroslav Olsak, Jakub Opršal, Michael Pinsker, and Ross Willard. ω-categorical structures avoiding height 1 identities. *Trans. Amer. Math. Soc.*, 374:327–350, 2021.

15 Manuel Bodirsky, Antoine Mottet, Miroslav Olšák, Jakub Opršal, Michael Pinsker, and Ross Willard. Topology is relevant (in a dichotomy conjecture for infinite-domain constraint satisfaction problems). In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*, pages 1–12. IEEE, 2019. doi:10.1109/LICS.2019.8785883.

16 Manuel Bodirsky, Michael Pinsker, and Todor Tsankov. Decidability of definability. *The Journal of Symbolic Logic*, 78(4):1036–1054, 2013. doi:10.2178/jsl.7804020.

17 Joshua Brakensiek and Venkatesan Guruswami. An algorithmic blend of LPs and ring equations for promise CSPs. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 436–455. SIAM, 2019. doi:10.1137/1.9781611975482.28.

18 Joshua Brakensiek and Venkatesan Guruswami. Promise constraint satisfaction: Algebraic structure and a symmetric boolean dichotomy. *SIAM J. Comput.*, 50(6):1663–1700, 2021. doi:10.1137/19M128212X.

19 Joshua Brakensiek, Venkatesan Guruswami, Marcin Wrochna, and Stanislav Zivný. The power of the combined basic linear programming and affine relaxation for promise constraint satisfaction problems. *SIAM J. Comput.*, 49(6):1232–1248, 2020. doi:10.1137/20M1312745.

20 Raimundo Briceno, Andrei Bulatov, Víctor Dalmau, and Benoit Larose. Dismantlability, connectedness, and mixing in relational structures. *J. Comb. Theory, Ser. B*, 147:37–70, 2021. doi:10.1016/j.jctb.2020.10.001.

21 Andrei A. Bulatov. A dichotomy theorem for nonuniform CSPs. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 319–330. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.37.

22 Gregory Cherlin, Saharon Shelah, and Niandong Shi. Universal graphs with forbidden subgraphs and algebraic closure. *Advances in Applied Mathematics*, 22:454–491, 1999.

23 Lorenzo Ciardo and Stanislav Živný. CLAP: A new algorithm for promise CSPs. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9–12, 2022*, pages 1057–1068. SIAM, 2022. doi:10.1137/1.9781611977073.46.

24 Lorenzo Ciardo and Stanislav Zivný. Hierarchies of minion tests for PCSPs through tensors. *CoRR*, abs/2207.02277, 2022. doi:10.48550/arXiv.2207.02277.

25 Lorenzo Ciardo and Stanislav Zivný. Approximate graph colouring and crystals. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 2256–2267. SIAM, 2023. doi:10.1137/1.9781611977554.CH86.

26 Lorenzo Ciardo and Stanislav Zivný. Approximate graph colouring and the hollow shadow. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 623–631. ACM, 2023. doi:10.1145/3564246.3585112.

27 Adam Ó Conghaile. Cohomology in constraint satisfaction and structure isomorphism. In Stefan Szeider, Robert Ganian, and Alexandra Silva, editors, *47th International Symposium on Mathematical Foundations of Computer Science, MFCS 2022, August 22-26, 2022, Vienna, Austria*, volume 241 of *LIPIcs*, pages 75:1–75:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.MFCS.2022.75.

28 Víctor Dalmau and Jakub Opršal. Local consistency as a reduction between constraint satisfaction problems. *CoRR*, abs/2301.05084, 2023. doi:10.48550/arXiv.2301.05084.

29 Tomás Feder and Moshe Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory. *SIAM J. Comput.*, 28(1):57–104, 1998. doi:10.1137/S0097539794266766.

30 Pierre Gillibert, Julius Jonusas, Michael Kompatscher, Antoine Mottet, and Michael Pinsker. Hrushovski's encoding and ω-categorical CSP monsters. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 131:1–131:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.131.

31 Pierre Gillibert, Julius Jonusas, Michael Kompatscher, Antoine Mottet, and Michael Pinsker. When symmetries are not enough: A hierarchy of hard constraint satisfaction problems. *SIAM J. Comput.*, 51(2):175–213, 2022. doi:10.1137/20m1383471.

32 Jan Hubička and Jaroslav Nešetřil. All those Ramsey classes (Ramsey classes with closures and forbidden homomorphisms). *CoRR*, abs/1606.07979, 2016. arXiv:1606.07979.

33 Andrei A. Krokhin and Jakub Opršal. An invitation to the promise constraint satisfaction problem. *ACM SIGLOG News*, 9(3):30–59, 2022. doi:10.1145/3559736.3559740.

34 Andrei A. Krokhin, Jakub Opršal, Marcin Wrochna, and Stanislav Zivný. Topology and adjunction in promise constraint satisfaction. *SIAM J. Comput.*, 52(1):38–79, 2023. doi:10.1137/20M1378223.

35 Gábor Kun, Ryan O'Donnell, Suguru Tamaki, Yuichi Yoshida, and Yuan Zhou. Linear programming, width-1 CSPs, and robust satisfaction. In Shafi Goldwasser, editor, *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, pages 484–495. ACM, 2012. doi:10.1145/2090236.2090274.

36 Benoit Larose, Cynthia Loten, and Claude Tardif. A characterisation of first-order constraint satisfaction problems. *Log. Methods Comput. Sci.*, 3(4), 2007. doi:10.2168/LMCS-3(4:6)2007.

37 Antoine Mottet and Michael Pinsker. Smooth approximations and CSPs over finitely bounded homogeneous structures. In Christel Baier and Dana Fisman, editors, *LICS '22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2–5, 2022*, pages 36:1–36:13. ACM, 2022. doi:10.1145/3531130.3533353.

38 Jaroslav Nešetřil and Claude Tardif. Duality theorems for finite structures (characterising gaps and good characterisations). *Journal of Combinatorial Theory, Series B*, 80(1):80–97, 2000. doi:10.1006/jctb.2000.1970.

39 Michael Pinsker. Current challenges in infinite-domain constraint satisfaction: Dilemmas of the infinite sheep. In *52nd IEEE International Symposium on Multiple-Valued Logic, ISMVL 2022, Dallas, TX, USA, May 18-20, 2022*, pages 80–87. IEEE, 2022. doi:10.1109/ISMVL52857.2022.00019.

40 Michael Pinsker and Manuel Bodirsky. Canonical functions: a proof via topological dynamics. *Contributions Discret. Math.*, 16(2):36–45, 2021. URL: https://cdm.ucalgary.ca/article/view/71724.

41 Benjamin Rossman. Homomorphism preservation theorems. *J. ACM*, 55(3):15:1–15:53, 2008. doi:10.1145/1379759.1379763.

42 Marcin Wrochna and Stanislav Živný. Improved hardness for H-colourings of G-colourable graphs. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1426–1435. SIAM, 2020. doi:10.1137/1.9781611975994.86.

43 Dmitriy Zhuk. A proof of CSP dichotomy conjecture. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 331–342. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.38.

44 Dmitriy Zhuk. A proof of the CSP dichotomy conjecture. *J. ACM*, 67(5):30:1–30:78, 2020. doi:10.1145/3402029.

Local Operators in Topos Theory and Separation of Semi-Classical Axioms in Intuitionistic Arithmetic

Satoshi Nakata
Research Institute for Mathematical Sciences, Kyoto University, Japan

Abstract

There has been work on the strength of semi-classical axioms over Heyting arithmetic such as Σ_n-**DNE** (double negation elimination) and Π_n-**LEM** (law of excluded middle). Among other things, Akama et al. show that Σ_n-**DNE** does not imply Π_n-**LEM** for any $n \geq 1$ by using Kleene realizability relativized to Turing degrees. These realizability notions are expressed by subtoposes of the effective topos $\mathcal{E}ff$ and thus by corresponding local operators (a.k.a. Lawvere-Tierney topologies).

Our purpose is to provide a topos-theoretic explanation for separation of semi-classical axioms. It consists of determining the least dense local operator of a given axiom φ in a topos $\mathcal{E}$, which completely characterizes the dense subtoposes of $\mathcal{E}$ satisfying φ. This idea is motivated by Caramello's study of intermediate propositional logics and van Oosten's study of Lifschitz realizability.

We first investigate sufficient conditions for an arithmetical formula to have a least dense operator. In particular, we show that each semi-classical axiom has a least dense operator in every elementary topos with natural number object. This is a generalization of van Oosten's result for $\Pi_1 \vee \Pi_1$-**DNE** in $\mathcal{E}ff$. We next determine least dense operators of semi-classical axioms in $\mathcal{E}ff$ in terms of (generalized) Turing degrees. Not only does it immediately imply some separation results of Akama et al. but also explain that realizability notions they used are optimal in the sense of minimality. We finally point out a negative consequence that Π_n-**LEM**, Σ_n-**LEM** and Σ_{n+1}-**DNE** are never separable by any subtopos of $\mathcal{E}ff$ for any $n \geq 0$.

2012 ACM Subject Classification Theory of computation → Constructive mathematics

Keywords and phrases local operator, elementary topos, effective topos, realizability, intuitionistic arithmetic

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.42

Funding This work is supported by JST Grant Number JPMJFS2123.

Acknowledgements I would like to thank my supervisor, Kazushige Terui, for careful reading and many helpful suggestions. I am also grateful to Hisashi Aratake, Yutaka Maita, Takayuki Kihara and the anonymous referees for their invaluable comments.

1 Introduction

Toposes are useful as semantics for logical systems and programming languages. In this context, the *effective topos* of Hyland [10] and its generalization, *realizability toposes* [11], have multiple applications. In particular, it is well known that the interpretation of logic and arithmetic in realizability toposes corresponds to the traditional realizability interpretation in intuitionistic proof theory. Van Oosten and others deeply investigate this correspondence and analyze various realizability notions from a topos-theoretic perspective [25].

In this paper, we mainly focus on toposes as models of *first-order intuitionistic arithmetic*, which is rich enough to encode and reason about programs and computations.

1.1 Various realizability methods and semi-classical axioms

Since Kleene [15] defined the first realizability interpretation (*Kleene realizability*) for *Heyting arithmetic* **HA**, many variants have been proposed in the literature. For example,

(1) Relativization to Turing degree d (*d-realizability*) [20].
(2) *Lifschitz realizability* [18, 23].
(3) Kreisel's *modified realizability* [16, 24].

These realizability methods are strongly related to the hierarchy of *semi-classical axioms* introduced by Akama, Berardi, Hayashi and Kohlenbach [1].

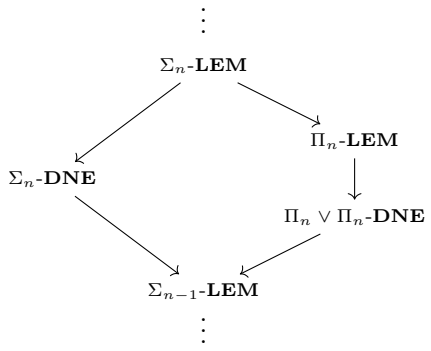


Figure 1 The hierarchy of semi-classical axioms.

In Figure 1, **DNE** and **LEM** stand for the *double negation elimination* and the *law of excluded middle*, respectively. Of course, **DNE** is equivalent to **LEM** in intuitionistic propositional logic. However, a difference arises when restricted to a class of arithmetical formulas such as Σ_n and Π_n. Indeed, Σ_n-**LEM** implies Σ_n-**DNE** in **HA** but the converse does not hold. More interestingly, an axiom scheme often corresponds to a semi-constructive principle such as the *lesser limited principle of omniscience*, the *constant domain axiom*, and even some variant of *Ramsey theorem* (*constructive reverse mathematics* over **HA**) [1, 2, 9].

Akama et al. separate the axioms in Figure 1 by using the realizability notions (1), (2) and a monotone variant of (3) above [1]. For instance, they show that Σ_n-**DNE** is realizable while Σ_n-**LEM**, Π_n-**LEM** and $\Pi_n \vee \Pi_n$-**DNE** are not under $\emptyset^{(n-1)}$-realizability, meaning that the former does not imply the latter. Similarly, Lifschitz realizability relativized to degree $\emptyset^{(n-1)}$ is used to separate $\Pi_n \vee \Pi_n$-**DNE** and Π_n-**LEM**.

It is known that the realizability notions (1), (2) and (3) correspond to subtoposes of (extensions of) the effective topos $\mathcal{E}ff$ [10, 20, 23, 24]. Above all, van Oosten studied the *Lifschitz topos* $\mathcal{L}if \subseteq \mathcal{E}ff$, where a first-order arithmetical formula φ is true iff φ is Lifschitz realizable. This representation leads to a topos-theoretic approach to the separation problem.

1.2 Least dense operators of logical and arithmetical formulas

Given an elementary topos $\mathcal{E}$ with subobject classifier Ω, the subtoposes of $\mathcal{E}$ are in one-to-one correspondence with the *local operators* in $\mathcal{E}$, that is, the meet-preserving closure operators $j\colon \Omega \to \Omega$ (a.k.a. *Lawvere-Tierney topologies*). A typical example is the *double negation operator* $\neg\neg\colon \Omega \to \Omega$, which exists in every topos. No matter which logic $\mathcal{E}$ models, the corresponding subtopos $\mathcal{E}_{\neg\neg}$ is always a model of classical logic. This link between the local operators and the intermediate logics has been further explored by Caramello [5, 6] in the context of categorical logic. A key notion there is what we call the *least dense operator* of a formula φ, that completely determines the dense subtoposes of $\mathcal{E}$ which satisfy φ. The same notion appears in van Oosten's study of categorical realizability. He showed that the local operator $j_{\mathcal{L}if}$ in $\mathcal{E}ff$ corresponding to the Lifschitz topos $\mathcal{L}if$ is the least dense operator of an arithmetical formula, which is equivalent to $\Pi_1 \vee \Pi_1$-**DNE** over **HA** [23].

The specifics of least dense operators are explained in Subsection 3.1. Our emphasis here is the following observation: given two axioms, if their least dense operators are different, then it automatically follows that they are separable.

1.3 Contents of this paper

We investigate the least dense operators of arithmetical formulas in relation to the separation of semi-classical axioms. Throughout the investigation, our aim is to demonstrate the utility of such a topos-theoretic notion in the study of intuitionistic proof theory. For the readers interested in proof theory, we try to make this paper as self-contained as possible.

In Section 2, we give some background. In Section 3, we study sufficient conditions for an arithmetical formula φ to have a least dense operator. For this purpose, we introduce two properties for formulas: *transparency* and *closedness*. Transparency ensures that a formula has a least dense operator under a mild assumption, while closedness is an intermediary means to show that a formula is transparent. Our argument here is a reconstruction and generalization of van Oosten's [23] for all toposes. The main result of Section 3 is that all semi-classical axioms in Figure 1 have least dense operators in every elementary topos with natural number object (Theorem 33, Corollary 34).

In Section 4, we apply the general theory in the previous section to the effective topos $\mathcal{E}ff$. As shown by Hyland [10], the poset of Turing degrees can be embedded into the poset of local operators in $\mathcal{E}ff$ (Figure 4). This allows us to relate the least dense operators of semi-classical axioms to (generalized) Turing degrees. For example, the least operator of Σ_n-**DNE** corresponds to Turing degree $\emptyset^{(n-1)}$, while that of $\Pi_n \vee \Pi_n$-**DNE** corresponds to another degree $\mathcal{L}if^{(n-1)}$ (Theorems 45, 46, Figure 5). Noting that least dense operators characterize separability by dense subtoposes, these expressions not only immediately imply some separation results of [1] but also a negative consequence that Π_n-**LEM**, Σ_n-**LEM** and Σ_{n+1}-**DNE** are never separable by any subtopos of $\mathcal{E}ff$ for any $n \geq 0$ (Corollary 47).

2 Preliminary

In this section, we review some basic facts on first-order intuitionistic arithmetic and interpretation of first-order logic and arithmetic in a topos. Most of the facts mentioned here can be found in standard textbooks [22, 21, 13, 19].

2.1 First-order intuitionistic arithmetic

Let $\mathcal{L}_{\mathrm{A}}$ be the language of arithmetic that consists of constant 0, successor Suc, and function symbols for all primitive recursive functions. *Heyting arithmetic*, written as **HA**, is a first-order intuitionistic $\mathcal{L}_{\mathrm{A}}$-theory consisting of $\forall x\, \neg(\mathrm{Suc}(x) = 0)$, defining equations for all primitive recursive functions, and the induction axiom scheme for all $\mathcal{L}_{\mathrm{A}}$-formulas. *Peano arithmetic*, written as **PA**, is defined by $\mathbf{PA} = \mathbf{HA} + \mathbf{LEM}$. We inductively define the classes Σ_n, Π_n of $\mathcal{L}_{\mathrm{A}}$-formulas as follows: $\Sigma_0 = \Pi_0$ are the set of all quantifier-free formulas, while Σ_{n+1}, Π_{n+1} are defined by $\Sigma_{n+1} \coloneqq \{\, \exists x_1 \cdots \exists x_k \varphi \mid \varphi \in \Pi_n,\, 0 \leq k \,\}$ and $\Pi_{n+1} \coloneqq \{\, \forall x_1 \cdots \forall x_k \varphi \mid \varphi \in \Sigma_n,\, 0 \leq k \,\}$. $\Pi_n \vee \Pi_n$ denotes the set of formulas of the form $\varphi \vee \psi$ with $\varphi, \psi \in \Pi_n$.

Given a formula φ, the *universal closure* of φ is denoted by ${}^{\forall}\varphi$. As in [1], we define some semi-classical axiom schemes as follows: for a subclass Γ of $\mathcal{L}_{\mathrm{A}}$-formulas, let

$$\Gamma\text{-}\mathbf{DNE} \coloneqq \{\, {}^{\forall}(\neg\neg\varphi \to \varphi) \mid \varphi \in \Gamma \,\}, \quad \Gamma\text{-}\mathbf{LEM} \coloneqq \{\, {}^{\forall}(\varphi \vee \neg\varphi) \mid \varphi \in \Gamma \,\}.$$

It is well known that **HA** proves Σ_0-**DNE** and Σ_0-**LEM**, and that for every n, Π_{n+1}-**DNE** is equivalent to Σ_n-**DNE** over **HA**. In this paper, a *semi-classical axiom* refers to either Γ-**DNE** or Γ-**LEM**, where Γ is Σ_n, Π_n, or $\Pi_n \vee \Pi_n$ for some $n \geq 0$.

Recall that **HA** formalizes a bijective primitive recursive pairing function $\langle -, - \rangle \colon \mathbb{N}^2 \to \mathbb{N}$. This allows us to code a finite sequence of natural numbers by a single one. Hence, without loss of generality, we can assume that each $\mathcal{L}_{\mathrm{A}}$-formula φ has at most one free variable.

By formalizing Post's theorem in **HA**, we obtain a *universal formula* $\varphi_{\Sigma_n}(e, x)$ for Σ_n with $n \geq 1$. That is, for any Σ_n-formula $A(x)$, there exists a numeral e_A such that ${}^{\forall}(A(x) \leftrightarrow \varphi_{\Sigma_n}(e_A, x))$ is provable in **HA** (folklore). The same holds for Π_n and $\Pi_n \vee \Pi_n$. For example, universal formulas φ_{Σ_1}, φ_{Π_1} for Σ_1, Π_1 are given by $\varphi_{\Sigma_1}(e, x) \coloneqq \exists w T(e, x, w)$, $\varphi_{\Pi_1}(e, x) \coloneqq \forall w \neg T(e, x, w)$, where $T(e, x, w)$ is Kleene's T-predicate. Thus, each axiom scheme in Figure 1 is finitely axiomatizable in **HA**.

2.2 Interpretation of intuitionistic logic in a topos

An *(elementary) topos* is a cartesian closed category with all finite limits and subobject classifier $\mathrm{true} \colon 1 \rightarrowtail \Omega$. According to the standard interpretation of many-sorted first-order logic in a topos, each formula is interpreted by a subobject in a suitable subobject poset. So let us first review the logical structure of a subobject poset.

For an object X in a topos $\mathcal{E}$, we write $U \rightarrowtail X$ for a subobject U of X and write $(\mathrm{Sub}_{\mathcal{E}}(X), \leq)$ for the poset of subobjects of X. Given a morphism $f \colon X \to Y$, f^* stands for the pullback functor along f. In a topos $\mathcal{E}$, $(\mathrm{Sub}_{\mathcal{E}}(X), \leq)$ forms a Heyting algebra.

▶ **Theorem 1.** *Let $\mathcal{E}$ be a topos.*

(1) *$\mathcal{E}$ is a* coherent category*. In particular, for any object X, $\mathrm{Sub}_{\mathcal{E}}(X)$ forms a distributive lattice with meet $\wedge$, join $\vee$, top 1 and bottom 0. In addition, for any morphism $f \colon X \to Y$, $f^* \colon \mathrm{Sub}_{\mathcal{E}}(Y) \to \mathrm{Sub}_{\mathcal{E}}(X)$ has a left adjoint $\exists_f \colon \mathrm{Sub}_{\mathcal{E}}(X) \to \mathrm{Sub}_{\mathcal{E}}(Y)$.*

(2) *$\mathcal{E}$ is further a* Heyting category*. In particular, for any $f \colon X \to Y$, f^* has a right adjoint $\forall_f \colon \mathrm{Sub}_{\mathcal{E}}(X) \to \mathrm{Sub}_{\mathcal{E}}(Y)$.*

The last $\forall_f$ induces Heyting implication $\Rightarrow$ on $\mathrm{Sub}_{\mathcal{E}}(X)$. In fact, $U \Rightarrow V$ can be defined to be $\forall_{m_U}(U \wedge V)$, where m_U is a representative of $U \rightarrowtail X$. If f is a projection $\pi \colon X \times Z \to Z$, $\forall_\pi$ (resp. $\exists_\pi$) provides an interpretation of first-order quantification $\forall x$ (resp. $\exists x$).

Now assume that to each sort A is assigned an object $[\![A]\!]$ and to each function symbol $f \colon A_1 \times \cdots \times A_n \to B$ a morphism $[\![f]\!] \colon [\![\vec{A}]\!] \to [\![B]\!]$, where $[\![\vec{A}]\!] \coloneqq [\![A_1]\!] \times \cdots \times [\![A_n]\!]$. Then each term $t = t(x_1^{A_1}, \cdots, x_n^{A_n}) \colon B$ is interpreted by a morphism $[\![t]\!] \colon [\![\vec{A}]\!] \to [\![B]\!]$ and equality $t =_B u$ by the equalizer $[\![t =_B u]\!] \rightarrowtail [\![\vec{A}]\!]$ of $[\![t]\!]$, $[\![u]\!] \colon [\![\vec{A}]\!] \rightrightarrows [\![B]\!]$. Interpretation of logical connectives is as above.

For a formula $\varphi = \varphi(x_1^{A_1}, \cdots, x_n^{A_n})$, if the interpretation $[\![\varphi]\!] \rightarrowtail [\![\vec{A}]\!]$ is identical to the greatest element of $\mathrm{Sub}_{\mathcal{E}}([\![\vec{A}]\!])$ (that is the equivalence class of the identity $\mathrm{id} : [\![\vec{A}]\!] \to [\![\vec{A}]\!]$), we say that *$\varphi$ is true in $\mathcal{E}$ (under $[\![-]\!]$)* and write $\mathcal{E} \models \varphi$. Under this interpretation, every topos satisfies all axioms of first-order intuitionistic logic.

2.3 Local operators and subtoposes

Local operator (a.k.a. *Lawvere-Tierney topology*) is one of the most important tools for creating a new topos from a given one.

▶ **Theorem 2.** *In a topos $\mathcal{E}$, there is a one-to-one correspondence among the following notions:*

(1) Local operator $j \colon \Omega \to \Omega$*, that is an endomorphism on the subobject classifier Ω of $\mathcal{E}$ which is an "internal"* nucleus *(recall that a nucleus on a lattice is a meet-preserving closure operator).*

(2) Universal closure operation $c \coloneqq \{\, c^X \colon \mathrm{Sub}_{\mathcal{E}}(X) \to \mathrm{Sub}_{\mathcal{E}}(X) \,\}_{X \in \mathcal{E}}$, *that is a family of nuclei on subobject posets which is natural in* $X \in \mathcal{E}$. *When it is clear from the context, we will omit superscript* X *in* c^X.

(3) Subtopos $\mathcal{F} \hookrightarrow \mathcal{E}$, *that is a full subcategory of* $\mathcal{E}$ *which is itself a topos such that the inclusion functor* $i \colon \mathcal{F} \hookrightarrow \mathcal{E}$ *has a cartesian left adjoint* $L \colon \mathcal{E} \to \mathcal{F}$. *Such an* L *is called a* sheafification functor (or associated sheaf functor) on $\mathcal{F}$.

Hereafter, $\mathcal{E}_j$, c_j and L_j denote the subtopos, the universal closure operation and the sheafification functor associated with a local operator j, respectively. (Note that the subtopos is usually denoted by $\mathbf{sh}_j(\mathcal{E})$.) We write $\mathbf{Lop}(\mathcal{E})$ for the class of local operators in $\mathcal{E}$.

▶ **Example 3.**

(1) The identity $\mathrm{id}_\Omega \colon \Omega \to \Omega$ and $\top \coloneqq \mathrm{true} \circ ! \colon \Omega \to \Omega$ are local operators in $\mathcal{E}$, where $!$ is the unique morphism from Ω to the terminal object 1. The corresponding subtoposes are $\mathcal{E}$ itself and the degenerate topos, respectively. $\top$ is called the *degenerate local operator*.

(2) For every topos $\mathcal{E}$, the family $\{\, ((\cdot \Rightarrow 0) \Rightarrow 0) \colon \mathrm{Sub}_{\mathcal{E}}(X) \to \mathrm{Sub}_{\mathcal{E}}(X) \,\}_{X \in \mathcal{E}}$ always forms a universal closure operation. The associated local operator is called the *double negation operator* $\neg\neg \colon \Omega \to \Omega$. The corresponding subtopos $\mathcal{E}_{\neg\neg}$ is a model of classical logic.

The correspondence in Theorem 2 induces a natural order on $\mathbf{Lop}(\mathcal{E})$.

▶ **Lemma 4.** *For* $j, k \in \mathbf{Lop}(\mathcal{E})$, *the following are equivalent:*

(1) $\mathcal{E}_k$ *is a subtopos of* $\mathcal{E}_j$.

(2) *For any object* $X \in \mathcal{E}$ *and any subobject* $U \rightarrowtail X$, $c_j(U) \leq c_k(U)$.

We write $j \leq k$ if the above equivalent conditions hold. $(\mathbf{Lop}(\mathcal{E}), \leq)$ forms a poset with the bottom element id_Ω and the top element $\top$.

Next, let us introduce two important notions.

▶ **Definition 5.** *Let* X *be an object of* $\mathcal{E}$, $U \rightarrowtail X$ *a subobject of* X *and* $j \in \mathbf{Lop}(\mathcal{E})$. *We say that* U is j-dense *if* $c_j(U) = X$, *and that* U is j-closed *if* $c_j(U) = U$. *Let* $\mathrm{Cl}_j\mathrm{Sub}_{\mathcal{E}}(X)$ *denote the class of* j*-closed subobjects of* X.

The j-dense elements in $\mathrm{Sub}_{\mathcal{E}}(X)$ are sent to the greatest element in $\mathrm{Sub}_{\mathcal{E}_j}(L_jX)$ by $L_j \colon \mathcal{E} \to \mathcal{E}_j$.

▶ **Lemma 6.** *For a subobject* $U \rightarrowtail X$ *and a local operator* j, U *is* j*-dense if and only if* $L_jU \rightarrowtail L_jX$ *is an isomorphism.*

On the other hand, the j-closed objects form a subobject lattice in $\mathcal{E}_j$: whenever F is an object of $\mathcal{E}_j \subseteq \mathcal{E}$, we have $\mathrm{Cl}_j\mathrm{Sub}_{\mathcal{E}}(F) = \mathrm{Sub}_{\mathcal{E}_j}(F)$. Moreover, the logical operations $(\wedge_j, \vee_j, \Rightarrow_j, \neg_j, \forall^j_f, \exists^j_f)$ on $\mathrm{Sub}_{\mathcal{E}_j}(F)$ are derived from $(\wedge, \vee, \Rightarrow, \neg, \forall_f, \exists_f)$ on $\mathrm{Sub}_{\mathcal{E}}(F)$ by means of the closure operation c_j as follows:

▶ **Lemma 7.** *Let* F, G *be objects of* $\mathcal{E}_j$, $f \colon F \to G$ *a morphism of* $\mathcal{E}_j$, *and* $A, B \in \mathrm{Sub}_{\mathcal{E}_j}(F) = \mathrm{Cl}_j\mathrm{Sub}_{\mathcal{E}}(F)$. *Then*

$$A \wedge_j B = A \wedge B, \qquad A \vee_j B = c_j(A \vee B), \qquad A \Rightarrow_j B = A \Rightarrow B,$$
$$\forall^j_f A = \forall_f A, \qquad \exists^j_f A = c_j(\exists_f A), \qquad \neg_j(A) = A \Rightarrow c_j(0).$$

2.4 Preservation of logical operations and degrees of openness

In this subsection, we have a look at when a sheafification functor L_j preserves a logical operation. This leads to a distinction of various degrees of openness of local operators. Note that L_j always preserves finite limits by definition, hence it preserves monomorphisms. Thus L_j induces a map $L_j\colon (\mathrm{Sub}_{\mathcal{E}}(X), \leq) \to (\mathrm{Sub}_{\mathcal{E}_j}(L_jX), \leq)$ for each object $X \in \mathcal{E}$. As is well known in categorical logic, L_j always preserves $\wedge$, $\vee$, 0 and $\exists_f$.

▶ **Proposition 8** ([19, Chapter IX]). *For any $j \in \mathbf{Lop}(\mathcal{E})$, L_j is a* coherent functor. *In particular, for any objects X, Y, morphism $f\colon X \to Y$ and subobjects U, $V \in \mathrm{Sub}_{\mathcal{E}}(X)$,*

$$L_j(U \circ V) = L_jU \circ_j L_jV, \quad L_j0 = 0_j, \quad L_j(\exists_f U) = \exists^j_{L_jf} L_jU,$$

where $\circ \in \{\wedge, \vee\}$ and $\circ_j$, 0_j, $\exists^j_{L_jf}$ are logical operations on $\mathrm{Sub}_{\mathcal{E}_j}(L_jX)$.

In addition, the following proposition shows that $\forall_f$ and $\Rightarrow$ are preserved by L_j under an assumption of closedness.

▶ **Proposition 9** ([10, Theorem 5.1]). *Let X be an object of $\mathcal{E}$, U, $V \rightarrowtail X$ subobjects of X and $j \in \mathbf{Lop}(\mathcal{E})$. If V is j-closed, then $L_j(\forall_f V) = \forall^j_{L_jf}(L_jV)$ and $L_j(U \Rightarrow V) = L_jU \Rightarrow_j L_jV$.*

However, L_j does not in general preserve universal quantification $\forall_f$, Heyting implication $\Rightarrow$ and negation $\neg$. Preservation of these operations is related to *opneness* of geometric morphisms [13, Proposition A4.5.1]. Motivated by this observation, Caramello gave the following definitions.

▶ **Definition 10** ([6, Definition 3.2]). *Let j be a local operator in $\mathcal{E}$.*
(1) *j is* open *if L_j preserves universal quantification on every subobject lattice.*
(2) *j is* implicationally open *if L_j preserves Heyting implication on every subobject lattice.*
(3) *j is* weakly open *if L_j preserves negation on every subobject lattice.*

We finally introduce another openness notion, *denseness*. This should not be confused with the notion of j-dense subobject in Definition 5.

▶ **Definition 11** ([6, Proposition 3.1]). *For $j \in \mathbf{Lop}(\mathcal{E})$, we say that j (or the corresponding subtopos $\mathcal{E}_j$) is* dense *if it satisfies one of the following equivalent conditions:*
(1) *The inclusion functor $i : \mathcal{E}_j \hookrightarrow \mathcal{E}$ preserves the initial object 0.*
(2) *$j \leq \neg\neg$, where $\neg\neg$ is the double negation operator.*
(3) *c_j preserves negation on every subobject lattice.*
(4) *The least subobject 0 of the terminal object 1 is j-closed.*

Considering the least element 0 as a subobject V in Proposition 9, the fourth condition of Definition 11 implies $L_j(\neg U) = L_j(U \Rightarrow 0) = (L_j(U) \Rightarrow_j 0_j) = \neg_j L_j(U)$ (weak openness). As a consequence, we have the following implications among the openness notions.

▶ **Proposition 12** ([6, Section 3]). *The following implications hold for local operators:*

$$\textit{open} \implies \textit{implicationally open} \implies \textit{weakly open} \impliedby \textit{dense}.$$

In fact, the implications are strict as there is a topos in which all the openness notions are different. The effective topos $\mathcal{E}ff$ provides such an example.

2.5 Preservation of arithmetical equality

If a topos $\mathcal{E}$ has a *natural number object (NNO)* N, it is possible to interpret $\mathcal{L}_{\mathrm{A}}$-terms and $\mathcal{L}_{\mathrm{A}}$-formulas in it. That is, we can assign to each function symbol $f \in \mathcal{L}_{\mathrm{A}}$ a morphism $[\![f]\!] \colon N^k \to N$ so that the defining equation for f is true in $\mathcal{E}$ by the universal property of NNO. Other axioms of Heyting arithmetic can also be verified ([17, Theorem 4.1]). Thus we can regard every topos with NNO as a model of **HA**.

Every sheafification functor $L_j \colon \mathcal{E} \to \mathcal{E}_j$ preserves NNO ([13, Lemma A2.5.6]). That is, if N is an NNO in $\mathcal{E}$, then so is $N_j \coloneqq L_j N$ in $\mathcal{E}_j$. It automatically follows that L_j preserves the interpretation of an atomic formula $f(\vec{x}) = g(\vec{x})$.

▶ **Lemma 13.** *Let $f(\vec{x})$ and $g(\vec{x})$ be k-ary function symbols. We have*

$$[\![f(\vec{x}) = g(\vec{x})]\!]_{\mathcal{E}_j} = \theta^*(L_j[\![f(\vec{x}) = g(\vec{x})]\!]_{\mathcal{E}}) \quad \text{in} \quad \mathrm{Sub}_{\mathcal{E}_j}(N_j^k),$$

where $\theta \colon N_j^k \to L_j(N^k)$ is the canonical isomorphism.

Proof. One can show that $[\![f]\!]_{\mathcal{E}_j} : N_j^k \to N_j$ coincides with $(L_j[\![f]\!]_{\mathcal{E}}) \circ \theta$ by induction on the construction of primitive recursive functions. The result then follows since the interpretation of $f(\vec{x}) = g(\vec{x})$ in $\mathcal{E}$ (resp. $\mathcal{E}_j$) is given by an equalizer of $[\![f]\!]_{\mathcal{E}}, [\![g]\!]_{\mathcal{E}} : N^k \to N$ (resp. $[\![f]\!]_{\mathcal{E}_j}$, $[\![g]\!]_{\mathcal{E}_j}$), which is preserved by L_j. ◀

3 Least dense operators of arithmetical formulas

Given a topos $\mathcal{E}$ and a formula φ, it is often possible to associate a local operator $j_\varphi^{\mathcal{E}}$ that completely determines the subtoposes of $\mathcal{E}$ which validate φ, in the sense that $j_\varphi^{\mathcal{E}} \leq k$ if and only if $\mathcal{E}_k \models \varphi$ for any local operator k in $\mathcal{E}$. Our purpose in this section is to develop a general theory of such local operators. Although the main focus of this paper lies on the effective topos $\mathcal{E}ff$, we anticipate that our general theory will find a wide range of applications in future, as will be discussed in Section 5. To achieve this, we need to restrict our treatment of local operators to *dense* ones. This restriction is not essential since all nondegenerate operators are dense in $\mathcal{E}ff$. See Example 16 and Remark 35 for further justifications.

In Subsection 3.1, we introduce the notion of *least dense operator* and see how it is relevant to the separation of subclassical axioms. In Subsections 3.2 and 3.3, we look at two properties of arithmetical formulas: *transparency* and *closedness*. Transparent formulas have least dense operators under a mild assumption, while the class of transparent and closed formulas enjoys good closure properties. In particular, all Σ_2-formulas are transparent and thus have least dense operators (under a mild assumption). However, there is a non-transparent formula in Π_3 which does not have a least dense operator, so the above result is optimal. These considerations lead us to a new technique to iterate the transparency argument. We show in Subsection 3.4 that all axioms in Figure 1 have least dense operators in an arbitrary topos with natural number object.

Throughout this section, we fix a topos $\mathcal{E}$ with natural number object N. By the assumption that each $\mathcal{L}_{\mathrm{A}}$-formula φ has at most one free variable (Subsection 2.1), the interpretation of φ in $\mathcal{E}$ can be simply regarded as subobject $[\![\varphi]\!]_{\mathcal{E}} \rightarrowtail N$.

3.1 Least dense operators

▶ **Notation 14.** *Let $\mathbf{DLop}(\mathcal{E}) \coloneqq \{\, k \in \mathbf{Lop}(\mathcal{E}) \mid k \leq \neg\neg \,\}$ denote the class of dense local operators in $\mathcal{E}$. For a local operator j, define $\mathbf{DLop}(\mathcal{E})^{\geq j} \coloneqq \{\, k \in \mathbf{DLop}(\mathcal{E}) \mid j \leq k \,\}$. $\mathbf{Lop}(\mathcal{E})^{\geq j}$ is similarly defined. For an $\mathcal{L}_{\mathrm{A}}$-formula φ, let $\langle\varphi\rangle^{\mathcal{E}} \coloneqq \{\, k \in \mathbf{DLop}(\mathcal{E}) \mid \mathcal{E}_k \models \varphi \,\}$.*

▶ **Definition 15.** *Let φ be an $\mathcal{L}_{\mathrm{A}}$-formula. A dense local operator j in $\mathcal{E}$ is called the* least dense operator of φ in $\mathcal{E}$*, written $j^{\mathcal{E}}_{\varphi}$, if it satisfies*

$$\langle\varphi\rangle^{\mathcal{E}} = \mathbf{DLop}(\mathcal{E})^{\geq j}.$$

Similarly, for an $\mathcal{L}_{\mathrm{A}}$-theory T, we use notations $\langle T\rangle^{\mathcal{E}} \coloneqq \{\, k \in \mathbf{DLop}(\mathcal{E}) \mid \mathcal{E}_k \models T \,\}$ and $j^{\mathcal{E}}_{T}$.

The concept of least dense operator is illustrated in Figure 2. The filled region corresponds to the class of dense operators whose associated subtoposes satisfy φ.

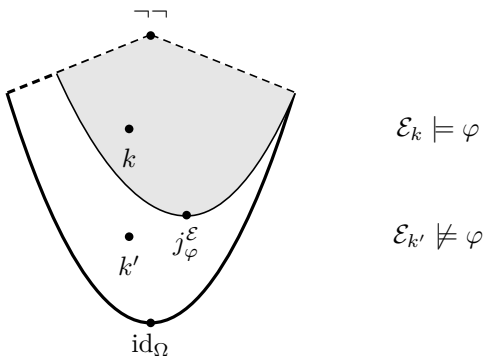


Figure 2 Least dense operator in $\mathbf{DLop}(\mathcal{E})$.

▶ **Example 16.** Similar concepts have been studied in various contexts.

(1) Blass and Šščedrov [4], in their investigation of categorical logic, proved that in any topos, propositional formula $p \vee \neg p$ has a unique local operator $\neg\neg$, which is nothing but the least dense operator in our terminology. It is significant for the study of the *classifying toposes* of geometric theories ([12, 4, 5], [7, Section 4.2.3]). Caramello [6] further investigated least dense operators for more general propositional formulas. She revealed that propositional formulas have least dense operators in any topos as far as they are *implication-free*. The reason for this restriction is that dense local operators are not implicationally open in general.

(2) Least local operators of arithmetical formulas in $\mathcal{E}ff$ are studied by van Oosten [23]. It is well known that in this topos, all nondegenerate operators are dense [10]. He identified a certain restricted class of $\mathcal{L}_{\mathrm{A}}$-formulas which have least local operators in $\mathcal{E}ff$. He then showed that the local operator $j_{\mathcal{L}if}$ corresponding to the *Lifschitz topos* $\mathcal{L}if \subseteq \mathcal{E}ff$ is the least operator of an $\mathcal{L}_{\mathrm{A}}$-formula (O) in that class. It is known that (O) is equivalent to $\Pi_1 \vee \Pi_1$-**DNE** over **HA**, so $j_{\mathcal{L}if}$ is the least operator of $\Pi_1 \vee \Pi_1$-**DNE** in $\mathcal{E}ff$ too.

We remark that both lines of work rely on the denseness of local operators, either explicitly or implicitly.

We also remark that least dense operators provide a good notion of "invariant", which is useful for the separation of subclassical axioms. In fact, the subset $\langle\varphi\rangle^{\mathcal{E}} \subseteq \mathbf{DLop}(\mathcal{E})$ is invariant under **HA**-provable equivalence:

$$\mathbf{HA} \vdash \varphi \leftrightarrow \psi \implies \langle\varphi\rangle^{\mathcal{E}} = \langle\psi\rangle^{\mathcal{E}}.$$

If φ and ψ further have least dense operators, we have

$$j^{\mathcal{E}}_{\varphi} \neq j^{\mathcal{E}}_{\psi} \implies \mathbf{HA} \nvdash \varphi \leftrightarrow \psi.$$

That is, we can separate two axioms just by showing that their least dense operators are different. Moreover, even if $j^{\mathcal{E}}_{\varphi} = j^{\mathcal{E}}_{\psi}$, we obtain a negative consequence that φ and ψ are never separable by a dense subtopos of $\mathcal{E}$ (Corollary 47). Thus, least dense operators provide us with sufficient information on separability by dense subtoposes.

All least operators in Example 16 are obtained based on the theorem below due to Joyal.

▶ **Theorem 17** ([13, Corollary A4.5.13]). *Let X be an object in $\mathcal{E}$ and $U \rightarrowtail X$. There is a unique local operator ℓ in $\mathcal{E}$ such that $\mathbf{Lop}(\mathcal{E})^{\geq \ell} = \{\, j \in \mathbf{Lop}(\mathcal{E}) \mid U \rightarrowtail X : j\text{-dense} \,\}$ holds.*

▶ **Definition 18.** *The above ℓ is called the* least operator of U in $\mathcal{E}$ *and written as $\ell^{\mathcal{E}}_{U}$.*

In particular, suppose that X is the natural number object N, U is the interpretation $[\![\varphi]\!]_{\mathcal{E}} \rightarrowtail N$ of an $\mathcal{L}_{\mathrm{A}}$-formula φ, and $j \in \mathbf{Lop}(\mathcal{E})$ satisfies the following condition:

$$L_j[\![\varphi]\!]_{\mathcal{E}} = [\![\varphi]\!]_{\mathcal{E}_j}. \qquad (j\text{-transparency})$$

Then Lemma 6 implies that $\mathcal{E}_j \models \varphi$ if and only if $[\![\varphi]\!]_{\mathcal{E}}$ is j-dense. Hence by Theorem 17 we have $\mathbf{Lop}(\mathcal{E})^{\geq \ell^{\mathcal{E}}_U} = \{\, j \in \mathbf{Lop}(\mathcal{E}) \mid \mathcal{E}_j \models \varphi \,\}$. What is critical here is the assumption of j-transparency, which will be the subject of the following subsections.

3.2 Transparency and closedness

▶ **Definition 19.** *For a local operator j and an $\mathcal{L}_{\mathrm{A}}$-formula φ,*
(1) *φ is j-transparent if $L_j[\![\varphi]\!]_{\mathcal{E}} = [\![\varphi]\!]_{\mathcal{E}_j}$ holds. Let $\mathrm{Trp}^{\mathcal{E}}_j := \{\, \varphi \mid \varphi$ is j-transparent $\,\}$.*
(2) *φ is j-closed if $c_j[\![\varphi]\!]_{\mathcal{E}} = [\![\varphi]\!]_{\mathcal{E}}$ holds. Let $\mathrm{Cl}^{\mathcal{E}}_j := \{\, \varphi \mid \varphi$ is j-closed $\,\}$.*
Let $\mathrm{Trp}^{\mathcal{E}} := \bigcap_{j \leq \neg\neg} \mathrm{Trp}^{\mathcal{E}}_j$ and $\mathrm{Cl}^{\mathcal{E}} := \bigcap_{j \leq \neg\neg} \mathrm{Cl}^{\mathcal{E}}_j$. We say φ is transparent in $\mathcal{E}$ if $\varphi \in \mathrm{Trp}^{\mathcal{E}}$.

As an example, every quantifier-free formula is transparent and closed in $\mathcal{E}$.

▶ **Lemma 20.** *$\mathrm{Cl}^{\mathcal{E}} = \mathrm{Cl}^{\mathcal{E}}_{\neg\neg}$ holds. Hence, for every $\mathcal{L}_{\mathrm{A}}$-formula φ,*
(1) *$\varphi \in \mathrm{Cl}^{\mathcal{E}}$ if and only if $\mathcal{E} \models \neg\neg\varphi \to \varphi$.*
(2) *Σ_0, $\Pi_0 \subseteq \mathrm{Trp}^{\mathcal{E}} \cap \mathrm{Cl}^{\mathcal{E}}$.*

Proof. Notice that $j \leq \neg\neg$ implies $c_j[\![\varphi]\!]_{\mathcal{E}} \leq c_{\neg\neg}[\![\varphi]\!]_{\mathcal{E}}$ (Lemma 4). This leads to $\mathrm{Cl}^{\mathcal{E}} = \mathrm{Cl}^{\mathcal{E}}_{\neg\neg}$. (1) immediately follows from the fact that $\mathcal{E} \models \neg\neg\varphi \to \varphi$ if and only if φ is $\neg\neg$-closed.

To show (2), suppose that $\varphi \in \Sigma_0 (= \Pi_0)$. We then obtain $\varphi \in \mathrm{Cl}^{\mathcal{E}}$ by (1) since $\mathbf{HA} \vdash \varphi \leftrightarrow \neg\neg\varphi$. By noting that every quantifier-free formula is equivalent to an atomic formula in $\mathbf{HA}$, $\varphi \in \mathrm{Trp}^{\mathcal{E}}$ follows from Lemma 13. ◀

Transparency and closedness are strongly related to the concept of least dense operator. More precisely, we will show the following correspondence: under a natural assumption,

- $\mathrm{Trp}^{\mathcal{E}}$ is a class of formulas which have least dense operators.
- $\mathrm{Trp}^{\mathcal{E}} \cap \mathrm{Cl}^{\mathcal{E}}$ is a class of formulas whose least dense operators are trivial.

Let us consider the latter first.

▶ **Lemma 21.** *Let $j \in \mathbf{DLop}(\mathcal{E})$ and $\varphi \in \mathrm{Trp}^{\mathcal{E}}_j \cap \mathrm{Cl}^{\mathcal{E}}_j$. Then $\mathcal{E}_j \models \varphi$ if and only if $\mathcal{E} \models \varphi$.*

Proof. The backward direction is clear since L_j preserves isomorphisms and $L_j[\![\varphi]\!]_{\mathcal{E}} = [\![\varphi]\!]_{\mathcal{E}_j}$.

For the forward direction, first note that $[\![\varphi]\!]_{\mathcal{E}_j} = L_j[\![\varphi]\!]_{\mathcal{E}}$ is the greatest element in $\mathrm{Sub}_{\mathcal{E}_j}(N_j)$. It then follows from Lemma 6 that $[\![\varphi]\!]_{\mathcal{E}}$ is j-dense. Since φ is j-closed, $[\![\varphi]\!]_{\mathcal{E}}$ is also the greatest element in $\mathrm{Sub}_{\mathcal{E}}(N)$. ◀

Therefore, we obtain the following.

▶ **Theorem 22.** *Let φ be an $\mathcal{L}_{\mathrm{A}}$-formula. The following are equivalent:*
(1) *$\varphi \in \mathrm{Trp}^{\mathcal{E}} \cap \mathrm{Cl}^{\mathcal{E}}$ and $\langle\varphi\rangle^{\mathcal{E}}$ is nonempty.*
(2) *$\langle\varphi\rangle^{\mathcal{E}} = \mathbf{DLop}(\mathcal{E})$, that is, $j^{\mathcal{E}}_{\varphi} = \mathrm{id}_{\Omega}$.*

Proof. If (1) holds, then we have $\mathcal{E}_j \models \varphi$ for every dense local operator j by Lemma 21. Hence $\langle\varphi\rangle^{\mathcal{E}} = \mathbf{DLop}(\mathcal{E})$. Conversely, assume that (2) holds. By assumption, $[\![\varphi]\!]_{\mathcal{E}_j}$ is the greatest element in $\mathrm{Sub}_{\mathcal{E}_j}(N_j)$ for any $j \in \mathbf{DLop}(\mathcal{E})$, so in particular $[\![\varphi]\!]_{\mathcal{E}}$ is also the greatest in $\mathrm{Sub}_{\mathcal{E}}(N)$. This implies $[\![\varphi]\!]_{\mathcal{E}_j} = L_j[\![\varphi]\!]_{\mathcal{E}}$ since L_j preserves isomorphisms. It is obvious that φ is in $\mathrm{Cl}^{\mathcal{E}}$ by $\mathcal{E} \models \varphi$. ◀

▶ **Corollary 23.** *Let φ be an $\mathcal{L}_{\mathrm{A}}$-formula.*
(1) *If φ is provable in* **HA**, *$\varphi \in \mathrm{Trp}^{\mathcal{E}} \cap \mathrm{Cl}^{\mathcal{E}}$ and $j^{\mathcal{E}}_{\varphi} = \mathrm{id}_{\Omega}$.*
(2) *If φ is provable in* **PA**, *$\varphi \in \mathrm{Trp}^{\mathcal{E}} \cap \mathrm{Cl}^{\mathcal{E}}$ iff $j^{\mathcal{E}}_{\varphi} = \mathrm{id}_{\Omega}$.*

Proof. If φ is provable in **HA**, then we have $\langle\varphi\rangle^{\mathcal{E}} = \mathbf{DLop}(\mathcal{E})$ since φ is true in any topos. Hence we also have $\varphi \in \mathrm{Trp}^{\mathcal{E}} \cap \mathrm{Cl}^{\mathcal{E}}$ by Theorem 22.

On the other hand, if φ is provable in **PA**, then the double negation operator $\neg\neg$ is in $\langle\varphi\rangle^{\mathcal{E}}$ since the corresponding subtopos $\mathcal{E}_{\neg\neg}$ satisfies any classically true formula, including φ. This implies that $\langle\varphi\rangle^{\mathcal{E}}$ is nonempty. ◀

3.3 Transparency yields least dense operators

We now turn our attention to transparent (but not necessarily closed) formulas. The argument below is a reconstruction of van Oosten's [23] in terms of transparency and closedness. The following lemma (cf. [23, Proposition 2.1]) plays a crucial role.

▶ **Lemma 24** (MAIN LEMMA)**.** *Let $\mathcal{E}$ be an arbitrary topos with natural number object.*
(1) *Suppose that an $\mathcal{L}_{\mathrm{A}}$-formula φ is transparent in $\mathcal{E}$. Then either $\langle\varphi\rangle^{\mathcal{E}} = \emptyset$ or φ has least dense operator $j^{\mathcal{E}}_{\varphi} = \ell^{\mathcal{E}}_{[\![\varphi]\!]_{\mathcal{E}}}$, where $\ell^{\mathcal{E}}_{[\![\varphi]\!]_{\mathcal{E}}}$ is the least operator of $[\![\varphi]\!]_{\mathcal{E}}$.*
(2) *Suppose that $\mathcal{L}_{\mathrm{A}}$-formulas φ, ψ are transparent in $\mathcal{E}$ and $\mathbf{HA} \vdash \varphi \to \psi$. Then for $\rho \coloneqq \psi \to \varphi$, either $\langle\rho\rangle^{\mathcal{E}} = \emptyset$ or ρ has least dense operator $j^{\mathcal{E}}_{\rho}$.*

Proof. By taking an **HA**-provable formula as ψ, (1) can be regarded as a special case of (2) (notice Corollary 23 (1)).

Let j be a dense local operator in $\mathcal{E}$. The assumption $\mathbf{HA} \vdash \varphi \to \psi$ implies that $\varphi \to \psi$ is true in any topos, including $\mathcal{E}$ and $\mathcal{E}_j$. So $[\![\varphi]\!]_{\mathcal{E}} \leq [\![\psi]\!]_{\mathcal{E}}$ and $[\![\varphi]\!]_{\mathcal{E}_j} \leq [\![\psi]\!]_{\mathcal{E}_j}$ hold. Now consider a subobject $U \coloneqq [\![\varphi]\!]_{\mathcal{E}} \rightarrowtail [\![\psi]\!]_{\mathcal{E}}$. Since φ and ψ are transparent, we get the equation $L_jU = (L_j[\![\varphi]\!]_{\mathcal{E}} \rightarrowtail L_j[\![\psi]\!]_{\mathcal{E}}) = ([\![\varphi]\!]_{\mathcal{E}_j} \rightarrowtail [\![\psi]\!]_{\mathcal{E}_j})$. Thus the following equivalence holds: $\mathcal{E}_j \models \rho$ iff $L_jU = [\![\varphi]\!]_{\mathcal{E}_j} \rightarrowtail [\![\psi]\!]_{\mathcal{E}_j}$ is an isomorphism iff U is j-dense (the last equivalence follows from Lemma 6).

By Theorem 17, we have the least operator $\ell^{\mathcal{E}}_U$ such that U is dense. If $\ell^{\mathcal{E}}_U \notin \mathbf{DLop}(\mathcal{E})$, then $\langle\rho\rangle^{\mathcal{E}} = \emptyset$. If $\ell^{\mathcal{E}}_U \in \mathbf{DLop}(\mathcal{E})$, $\ell^{\mathcal{E}}_U$ is the least dense operator of ρ in $\mathcal{E}$. ◀

The next step of our reconstruction is to examine the closure properties satisfied by $\mathrm{Trp}^{\mathcal{E}}$ and $\mathrm{Cl}^{\mathcal{E}}$. The proof of the case $\neg$ relies on the restriction to dense local operators.

▶ **Lemma 25.**
(1) *$\mathrm{Trp}^{\mathcal{E}}$ is closed under $\wedge$, $\vee$, $\exists$, $\neg$.*
(2) *$\mathrm{Cl}^{\mathcal{E}}$ is closed under $\wedge$, $\to$, $\forall$, $\neg$.*
(3) *Suppose that $\varphi \in \mathrm{Trp}^{\mathcal{E}} \cap \mathrm{Cl}^{\mathcal{E}}$ and $\psi \in \mathrm{Trp}^{\mathcal{E}}$. Then $\psi \to \varphi$ and $\forall x\, \varphi$ belong to $\mathrm{Trp}^{\mathcal{E}} \cap \mathrm{Cl}^{\mathcal{E}}$. In particular, $\mathrm{Trp}^{\mathcal{E}} \cap \mathrm{Cl}^{\mathcal{E}}$ is closed under $\wedge$, $\to$, $\forall$, $\neg$.*

Proof. (1) Closure under $\wedge$, $\vee$, $\exists$ is due to Proposition 8. For $\neg$, just recall that dense local operators are weakly open (Proposition 12). (2) $\mathrm{Cl}^{\mathcal{E}}$ is closed under $\wedge$, $\to$, $\forall$ by Lemma 7, and under $\neg$ by denseness (Definition 11 (3)). (3) holds by Proposition 9. ◀

Together with Lemma 20, we obtain the following:

▶ **Corollary 26.** *$\Pi_1 \subseteq \mathrm{Trp}^{\mathcal{E}} \cap \mathrm{Cl}^{\mathcal{E}}$ and $\Sigma_2 \subseteq \mathrm{Trp}^{\mathcal{E}}$.*

▶ **Corollary 27.** *Suppose that an $\mathcal{L}_{\mathrm{A}}$-formula φ is transparent in $\mathcal{E}$. Then $\varphi \vee \neg\varphi$ and $\neg\neg\varphi \to \varphi$ have least dense operators in $\mathcal{E}$.*

Proof. Note that $\langle \varphi \vee \neg\varphi \rangle^{\mathcal{E}}$ and $\langle \neg\neg\varphi \to \varphi \rangle^{\mathcal{E}}$ are always nonempty because both formulas are provable in **PA**, so true in $\mathcal{E}_{\neg\neg}$. $\varphi \vee \neg\varphi$ belongs to $\mathrm{Trp}^{\mathcal{E}}$ by Lemma 25 (1), hence $\varphi \vee \neg\varphi$ has least dense operator in $\mathcal{E}$ by Lemma 24 (1).

For $\neg\neg\varphi \to \varphi$, we apply the case (2) of Lemma 24. ◀

This corollary can be extended to semi-classical axioms. Recall that for $n \geq 1$, there exist universal formulas φ_{Σ_n} and φ_{Π_n} for the classes Σ_n and Π_n. Hence axiom scheme Σ_n-**LEM**, for example, is equivalent to formula ${}^{\forall}(\varphi_{\Sigma_n} \vee \neg\varphi_{\Sigma_n})$ over **HA**. Moreover, we have $\mathcal{E} \models^{\forall} (\varphi_{\Sigma_n} \vee \neg\varphi_{\Sigma_n})$ iff $\mathcal{E} \models \varphi_{\Sigma_n} \vee \neg\varphi_{\Sigma_n}$. Thus we obtain:

▶ **Lemma 28.** *Let Γ be one of Σ_n, Π_n and $\Pi_n \vee \Pi_n$ and assume that $\Gamma \subseteq \mathrm{Trp}^{\mathcal{E}}$. Then Γ-**LEM** and Γ-**DNE** have least dense operators in $\mathcal{E}$.*

As we saw in Corollary 26, $\Sigma_2 \subseteq \mathrm{Trp}^{\mathcal{E}}$ always holds. This fact ensures that Σ_2-**DNE** and Σ_2-**LEM** have least dense operators in $\mathcal{E}$. On the other hand, we can show that $\Pi_3 \subseteq \mathrm{Trp}^{\mathcal{E}}$ does not hold in general (see Theorem 49 in Appendix A). Therefore, to show the existence of least dense operators of semi-classical axioms for $n \geq 3$, we need another technique, that is to be discussed in the next subsection.

3.4 Iteration argument and least dense operators of semi-classical axioms

In this subsection, we prove that all semi-classical axioms have least dense operators in every topos. The key idea is to iterate the construction of Lemma 28.

▶ **Lemma 29.** *Let $n \geq 0$. If $\mathcal{E} \models \Sigma_n$-**DNE**, then $\Pi_{n+1} \subseteq \mathrm{Trp}^{\mathcal{E}} \cap \mathrm{Cl}^{\mathcal{E}}$, so $\Sigma_{n+2} \subseteq \mathrm{Trp}^{\mathcal{E}}$.*

Proof. By induction on n, recalling that $\mathcal{E} \models \Sigma_n$-**DNE** iff $\Sigma_n \subseteq \mathrm{Cl}^{\mathcal{E}}$ (Lemma 20). The base case is true by Corollary 26. Next assume that $\Sigma_{n+1} \subseteq \mathrm{Cl}^{\mathcal{E}}$. By the induction hypothesis $\Pi_n \subseteq \Pi_{n+1} \subseteq \mathrm{Trp}^{\mathcal{E}} \cap \mathrm{Cl}^{\mathcal{E}}$, so $\Sigma_{n+1} \subseteq \mathrm{Trp}^{\mathcal{E}}$ by Lemma 25 (1), that is, $\Sigma_{n+1} \subseteq \mathrm{Trp}^{\mathcal{E}} \cap \mathrm{Cl}^{\mathcal{E}}$. By Lemma 25 (3), we conclude $\Pi_{n+2} \subseteq \mathrm{Trp}^{\mathcal{E}} \cap \mathrm{Cl}^{\mathcal{E}}$. ◀

For the sake of the argument below, let us note a natural correspondence between $\mathbf{Lop}(\mathcal{E})^{\geq j}$ and $\mathbf{Lop}(\mathcal{E}_j)$.

▶ **Notation 30.** *Let $j \in \mathbf{Lop}(\mathcal{E})$. There is a one-to-one correspondence between $\mathbf{Lop}(\mathcal{E})^{\geq j}$ and $\mathbf{Lop}(\mathcal{E}_j)$. We write k_j for the local operator in $\mathbf{Lop}(\mathcal{E}_j)$ corresponding to $k \in \mathbf{Lop}(\mathcal{E})^{\geq j}$. Under this notation we have $(\mathcal{E}_j)_{k_j} = \mathcal{E}_k$.*

▶ **Lemma 31** ([7, Corollary 4.2.9])**.** *If j is a dense local operator in $\mathcal{E}$, $\neg\neg_j$ is identical to the double negation operator in $\mathbf{Lop}(\mathcal{E}_j)$. Thus, the correspondence in Notation 30 holds even if restricted to the dense local operators (Figure 3).*

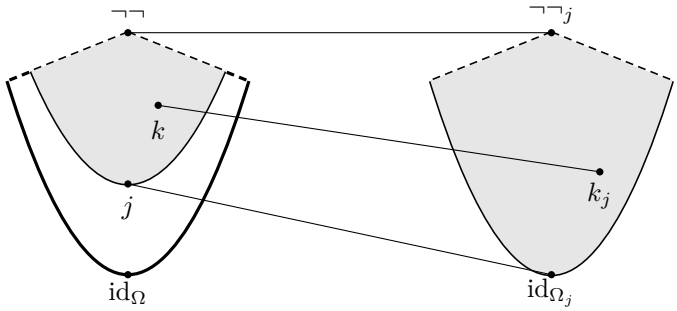


Figure 3 Correspondence between $\mathbf{DLop}(\mathcal{E})^{\geq j}$ and $\mathbf{DLop}(\mathcal{E}_j)$.

The following lemma allows us to iterate Lemma 28.

▶ **Lemma 32.** *Let T, S be $\mathcal{L}_{\mathrm{A}}$-theories such that $\mathbf{HA} + T \vdash S$. Suppose further that*
(1) *S has least dense operator $j_S \coloneqq j_S^{\mathcal{E}} \in \mathbf{DLop}(\mathcal{E})$ in $\mathcal{E}$.*
(2) *T has least dense operator $j'_T \coloneqq j_T^{\mathcal{E}_{j_S}} \in \mathbf{DLop}(\mathcal{E}_{j_S})$ in $\mathcal{E}_{j_S}$.*
Then $j_T \in \mathbf{DLop}(\mathcal{E})^{\geq j_S}$ corresponding to j'_T is the least dense operator of T in $\mathcal{E}$.

Proof. Let $k \in \mathbf{DLop}(\mathcal{E})$. We show that $\mathcal{E}_k \models T$ if and only if $j_T \leq k$. Assume that $\mathcal{E}_k \models T$. $j_S \leq k$ clearly follows from the assumption (1) and $\mathbf{HA} + T \vdash S$. So $k \in \mathbf{DLop}(\mathcal{E})^{\geq j_S}$ and the corresponding operator $k_{j_S} \in \mathbf{DLop}(\mathcal{E}_{j_S})$ yields a dense subtopos of $\mathcal{E}_{j_S}$ that satisfies T. Hence, by the assumption (2), we get $j'_T \leq k_{j_S}$. This implies $j_T \leq k$.

Conversely, suppose $j_T \leq k$. Since $j_T \in \mathbf{DLop}(\mathcal{E})^{\geq j_S}$, the corresponding operator $j'_T \in \mathbf{DLop}(\mathcal{E}_{j_S})$ satisfies $j'_T \leq k_{j_S}$. By the assumption (2) again, $(\mathcal{E}_{j_S})_{k_{j_S}} = \mathcal{E}_k \models T$ holds. ◀

▶ **Theorem 33.** *For every topos $\mathcal{E}$ with natural number object and $n \geq 0$, Σ_n-**DNE** has least dense operator $j_n \coloneqq j^{\mathcal{E}}_{\Sigma_n\text{-}\mathbf{DNE}}$ in $\mathcal{E}$.*

Proof. By induction on n. For $n = 0$, let $j_0 = \mathrm{id}_\Omega$ (See Corollary 23).

Next, assume that Σ_n-**DNE** has least dense operator j_n in $\mathcal{E}$. Then Σ_n-**DNE** is true in $\mathcal{E}_{j_n}$, hence we have $\Sigma_{n+1} \subseteq \Sigma_{n+2} \subseteq \mathrm{Trp}^{\mathcal{E}_{j_n}}$ by Lemma 29. Thus, it follows from Lemma 28 that Σ_{n+1}-**DNE** has least dense operator in $\mathcal{E}_{j_n}$. Since $\mathbf{HA} + \Sigma_{n+1}$-**DNE** implies Σ_n-**DNE**, all assumptions of Lemma 32 are satisfied. We therefore conclude that Σ_{n+1}-**DNE** has least dense operator in $\mathcal{E}$. ◀

The local operators $\{ j_n \}$ can be used as the "footholds" to obtain least dense operators of other semi-classical axioms.

▶ **Corollary 34.** *For every topos $\mathcal{E}$ with natural number object and $n \geq 0$, Σ_n-**LEM**, Π_n-**LEM** and $\Pi_n \vee \Pi_n$-**DNE** have least dense operators in $\mathcal{E}$.*

Proof. We here focus on Π_{n+1}-**LEM**. Considering the least dense operator j_n of Σ_n-**DNE** in Theorem 33, we have $\Pi_{n+1} \subseteq \Sigma_{n+2} \subseteq \mathrm{Trp}^{\mathcal{E}_{j_n}}$. Thus, it follows from Lemma 28 that Π_{n+1}-**LEM** has least dense operator in $\mathcal{E}_{j_n}$. Since $\mathbf{HA} + \Pi_{n+1}$-**LEM** proves Σ_n-**DNE** (Figure 1), it also has least dense operator in $\mathcal{E}$ by Lemma 32. ◀

▶ Remark 35. Let us finally discuss (dis)advantages of the restriction to dense operators. One clear disadvantage is that it forces us to introduce an additional assumption $\langle \varphi \rangle^{\mathcal{E}} \neq \emptyset$ in Lemma 24 (MAIN LEMMA). Although this may appear inconvenient, it does not cause any problem as long as semi-classical axioms are concerned (See the proof of Corollary 27).

On the other hand, the restriction is really essential for Lemma 20. In fact, this lemma does not hold without the assumption of denseness, as indicated by the following:

▶ **Theorem 36.** *Let j be a local operator. All Σ_0-formulas are j-closed iff j is dense.*

Proof. The backward direction is shown in Lemma 20. To see the forward direction, consider Σ_0-formula $\neg(x = x)$, whose interpretation is the least element 0 in the subobject poset of NNO. If it is j-closed, then the least subobject 0 of the terminal object 1 is also j-closed. The latter condition is equivalent to being a dense local operator (Definition 11 (4)). ◀

Failure of Lemma 20 would affect most of the subsequent theorems in Section 3. For example, there is no guarantee that all Σ_2-formulas are transparent. We would say that denseness is a price to pay to obtain these theorems in the general setting.

4 Least operators in the effective topos

In this section, we apply the general theory developed in the previous section to the effective topos $\mathcal{E}ff$. As explained in Example 16, any non-degenerate operator is dense in this topos. Henceforth, we speak of *least operators* instead of least dense ones.

In Subsection 4.1, we briefly review the structure of subobjects and that of local operators in $\mathcal{E}ff$. We also mention that there are local operators corresponding to Turing degrees. In Subsection 4.2, we express all least dense operators of semi-classical axioms in terms of (generalized) Turing degrees. This immediately leads to some separation results conforming to [1]. For details on $\mathcal{E}ff$, the reader is referred to [25].

4.1 Subobjects of NNO and local operators in $\mathcal{E}ff$

Let us first recall the effective topos and associated concepts.

▶ **Notation 37.** *We fix a primitive recursive pairing function $\langle -, - \rangle : \mathbb{N}^2 \to \mathbb{N}$ with the associated projections $(-)_0$, $(-)_1 : \mathbb{N} \to \mathbb{N}$. For natural numbers e and n, we write $e \cdot n$ for the result of applying the e-th partial computable function to n, and write $e \cdot n \downarrow$ if the computation terminates. If ψ is a closed $\mathcal{L}_{\mathrm{A}}$-formula, $n\ \mathbf{r}_{\mathrm{K}}\ \psi$ means that n realizes ψ under Kleene realizability. Also we use λ-notation: for a partial computable function $t :\subseteq \mathbb{N} \to \mathbb{N}$ in variable x, $\lambda x.t$ denotes an index of t. Similarly, for $u :\subseteq \mathbb{N}^2 \to \mathbb{N}$ in variable x and y, $\lambda xy.u$ is an abbreviation for $\lambda x.(\lambda y.u)$.*

Given a set X, we consider the following operations on $\mathcal{P}(\mathbb{N})^X$. For any $\varphi, \psi : X \to \mathcal{P}(\mathbb{N})$,

$$\begin{aligned}
\varphi \wedge \psi(x) &\coloneqq \{\, \langle n, m \rangle \mid n \in \varphi(x) \wedge m \in \psi(x) \,\}, & \top(x) &\coloneqq \mathbb{N}, \\
\varphi \vee \psi(x) &\coloneqq \{\, \langle 0, n \rangle \mid n \in \varphi(x) \,\} \cup \{\, \langle 1, m \rangle \mid m \in \psi(x) \,\}, & \bot(x) &\coloneqq \emptyset, \\
\varphi \to \psi(x) &\coloneqq \{\, e \mid \forall n \in \varphi(x)\ (e \cdot n \downarrow \wedge\, e \cdot n \in \psi(x)) \,\}, & \neg\varphi(x) &\coloneqq \varphi \to \bot(x).
\end{aligned}$$

In addition, we define a preorder $\sqsubseteq$ on $\mathcal{P}(\mathbb{N})^X$: $\varphi \sqsubseteq \psi$ if $\bigcap_{x \in X}(\varphi \to \psi(x))$ is nonempty. Then $(\mathcal{P}(\mathbb{N})^X, \sqsubseteq)$ forms a Heyting prealgebra and induces the *effective tripos* $\mathrm{P}_{\mathcal{E}ff} : X \mapsto (\mathcal{P}(\mathbb{N})^X, \sqsubseteq)$. The *effective topos* $\mathcal{E}ff$ is given by the *tripos-to-topos construction* on $\mathrm{P}_{\mathcal{E}ff}$. For example, an object X of $\mathcal{E}ff$ is a pair $X = (X, =_X)$ where X is a set and $=_X : X \times X \to \mathcal{P}(\mathbb{N})$ is a "$\mathcal{P}(\mathbb{N})$-valued equality" with respect to $\mathrm{P}_{\mathcal{E}ff}$, and a morphism $f : X \to Y$ of $\mathcal{E}ff$ is an (equivalence class of) "$\mathcal{P}(\mathbb{N})$-valued functional relation" $F : X \times Y \to \mathcal{P}(\mathbb{N})$ that respects $=_X$ and $=_Y$ ([25, Chapter 2]).

$\mathcal{E}ff$ has a natural number object $N = (\mathbb{N}, =_N)$, where $[n =_N m] \coloneqq \{\, n \,\}$ if $n = m$ and $\coloneqq \emptyset$ otherwise. A subobject classifier $\Omega = (\mathcal{P}(\mathbb{N}), =_\Omega)$ is given by defining $[p =_\Omega q] \coloneqq (p \to q) \wedge (q \to p)$, where $\wedge$, $\to$ are operations on $\mathcal{P}(\mathbb{N}) \cong \mathcal{P}(\mathbb{N})^{\{*\}}$.

Similarly, the structure of subobjects in $\mathcal{E}ff$ is determined by $\mathrm{P}_{\mathcal{E}ff}$. Indeed, a subobject U of $X \in \mathcal{E}ff$ can be described by a "strict relational" function $U : X \to \mathcal{P}(\mathbb{N})$ with respect to $=_X$. As far as the subobjects of N are concerned, relationality is trivial so that they admit much simpler descriptions below.

▶ **Definition 38.** *A function $U : \mathbb{N} \to \mathcal{P}(\mathbb{N})$ is called a* (partial) multifunction *and written as $U :\subseteq \mathbb{N} \rightrightarrows \mathbb{N}$. Let* **Mfunc** *denote the set of all multifunctions. A preorder on* **Mfunc** *can be defined by invoking Notation 37 for the case of $X = \mathbb{N}$: $U \sqsubseteq_s V$ iff $U' \sqsubseteq V$, where $U'(e) \coloneqq \{\, \langle e, n\rangle \mid n \in U(e) \,\}$. The latter means that*

$$\exists f \in \mathbb{N} \forall e, n \in \mathbb{N} \quad (n \in U(e) \implies f \cdot \langle e, n\rangle \in V(e)).$$

We write $U \equiv_s V$ if $U \sqsubseteq_s V$ and $V \sqsubseteq_s U$.

This preorder induces a correspondence between the multifunctions and the subobjects of N. The reason for using U' is that U is not strict with respect to $=_N$ in general.

▶ **Proposition 39.** $(\mathbf{Mfunc}, \sqsubseteq_s) \simeq (\mathrm{Sub}_{\mathcal{E}ff}(N), \leq)$.

We thus think of a multifunction $U :\subseteq \mathbb{N} \rightrightarrows \mathbb{N}$ as a subobject of N. In the sequel, we are mainly interested in the subobjects of N that are interpretations of $\mathcal{L}_{\mathrm{A}}$-formulas. Given an $\mathcal{L}_{\mathrm{A}}$-formula φ, the interpretation $[\![\varphi]\!]_{\mathcal{E}ff}$ corresponds to a multifunction $[\![\varphi]\!]_{\mathbf{Mfunc}}$ as follows:

$$[\![\varphi]\!]_{\mathbf{Mfunc}}(e) \coloneqq \{\, n \in \mathbb{N} \mid n \mathbf{r}_{\mathrm{K}} \varphi(e) \,\}.$$

Hence, φ is true in $\mathcal{E}ff$ iff $[\![\varphi]\!]_{\mathcal{E}ff}$ is the greatest element in $\mathrm{Sub}_{\mathcal{E}ff}(N)$ iff $\top \sqsubseteq_s [\![\varphi]\!]_{\mathbf{Mfunc}}$ iff $\exists f \in \mathbb{N}\, \forall e \in \mathbb{N}\, (f \cdot e \downarrow$ and $f \cdot e \mathbf{r}_{\mathrm{K}} \varphi(e))$ iff ${}^{\forall}\varphi$ is Kleene realizable [25]. Notice that $[\![\varphi]\!]_{\mathbf{Mfunc}}$ is coherent with the operations introduced in Notation 37. That is, for each $\circ \in \{\, \wedge, \vee, \to \,\}$,

$$[\![\varphi \circ \psi]\!]_{\mathbf{Mfunc}} = [\![\varphi]\!]_{\mathbf{Mfunc}} \circ [\![\psi]\!]_{\mathbf{Mfunc}}, \quad [\![\neg\varphi]\!]_{\mathbf{Mfunc}} = \neg[\![\varphi]\!]_{\mathbf{Mfunc}}.$$

Each local operator in $\mathcal{E}ff$ has a simple expression as an endofunction on $\mathcal{P}(\mathbb{N})$, but we omit its details here. It is important that there are local operators corresponding to *Turing degrees*. Recall Joyal's theorem (Theorem 17) in the previous section, which shows that every subobject $U \rightarrowtail N$ has least local operator $\ell^{\mathcal{E}}_U$ that makes U dense. The following observation is due to Hyland [10].

▶ **Theorem 40** ([10, Theorem 17.2])**.** *Let A, $B \subseteq \mathbb{N}$. A is* Turing reducible *to B if and only if $j_A \leq j_B$ in $\mathbf{Lop}(\mathcal{E}ff)$, where j_A is the least operator of the characteristic function χ_A of A in* **Mfunc**. *Hence the poset of Turing degrees can be embedded into $(\mathbf{Lop}(\mathcal{E}ff), \leq)$.*

In other words, the local operators in $\mathcal{E}ff$ can be regarded as *generalized Turing degrees*. For example, the Lifschitz operator $j_{\mathcal{L}if}$ in Example 16 is in between $j_\emptyset$ and $j_{\emptyset^{(1)}}$, but it is never equal to j_d for any Turing degree d (see the notation below and Figure 4). This connection with degree theory is further extended and refined by Faber and van Oosten [8] and Kihara [14].

▶ **Notation 41.** *Given a set $D \subseteq \mathbb{N}$ with Turing degree d (i.e., $D \in d$), we write j_d to denote the least operator $\ell^{\mathcal{E}ff}_{\chi_D}$ of $\chi_D : \mathbb{N} \to \mathbb{N}$. For example, when D is a decidable set, we have $j_d = \ell^{\mathcal{E}ff}_{\chi_D} = j_\emptyset = \mathrm{id}_\Omega$. It is known that a closed formula is true in the subtopos $\mathcal{E}ff_{j_d}$ if and*

*only if it is realizable under the Kleene realizability relativized to d (*d-realizability*) [20]. For* $n \in \mathbb{N}$, $\emptyset^{(n)}$ *denotes the n-th Turing jump of* $\emptyset$ *and* $\mathcal{E}ff^{(n)}$ *the subtopos* $\mathcal{E}ff_{j_{\emptyset^{(n)}}}$. *By letting* $\mathbf{r}_{\mathrm{K}}^{(n)}$ *be the* $\emptyset^{(n)}$*-realizability relation, the structure of subobjects in* $\mathcal{E}ff^{(n)}$ *can be described by* $\mathbf{r}_{\mathrm{K}}^{(n)}$. *For instance,* $[\![\varphi]\!]_{\mathcal{E}ff^{(n)}}$ *corresponds to a multifunction* $[\![\varphi]\!]^{(n)}_{\mathbf{Mfunc}}(e) \coloneqq \{\, m \mid m\ \mathbf{r}_{\mathrm{K}}^{(n)}\ \varphi(e) \,\}$.

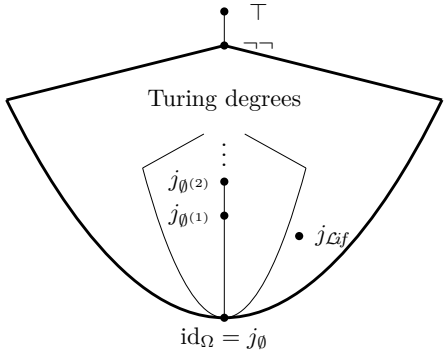


Figure 4 Turing degrees embedded in $\mathbf{Lop}(\mathcal{E}ff)$.

4.2 Turing degrees and least operators of semi-classical axioms

In this subsection, we give concrete representations to the least operators of semi-classical axioms in terms of generalized Turing degrees. First of all, the following is straightforward by Corollary 23 and Figure 1.

▶ **Lemma 42.** *For every topos* $\mathcal{E}$ *with natural number object and* $n \geq 0$,
(1) $j^{\mathcal{E}}_{\Sigma_0\text{-}\mathbf{DNE}} = j^{\mathcal{E}}_{\Pi_0\text{-}\mathbf{LEM}} = j^{\mathcal{E}}_{\Sigma_0\text{-}\mathbf{LEM}} = \mathrm{id}_\Omega$.
(2) $j^{\mathcal{E}}_{\Pi_n\text{-}\mathbf{LEM}} \leq j^{\mathcal{E}}_{\Sigma_n\text{-}\mathbf{LEM}} \leq j^{\mathcal{E}}_{\Sigma_{n+1}\text{-}\mathbf{DNE}}$.

It is well known that Σ_1-**DNE** is Kleene realizable, so the least dense operator of Σ_1-**DNE** in $\mathcal{E}ff$ is just id_Ω. Using Notation 41, this can be expressed by Turing degree $\emptyset$:

$$j^{\mathcal{E}ff}_{\Pi_0\text{-}\mathbf{LEM}} = j^{\mathcal{E}ff}_{\Sigma_0\text{-}\mathbf{LEM}} = j^{\mathcal{E}ff}_{\Sigma_1\text{-}\mathbf{DNE}} = j_\emptyset. \tag{♣}$$

This equation (♣) can be extended to any $n \geq 1$ (Theorem 45). Here we give only a proof of the case $n = 1$ because there is no great difficulty in generalizing the following arguments. The complete proof can be found in Appendix B.

Let us recall Subsection 2.1 and a primitive recursive function $S(e, x)$ obtained by formalizing the *parameter theorem* in **HA**. $S(e, x)$ has the ability to "shift" a variable, that is, "$S(e, x) \cdot y \downarrow$ iff $e \cdot \langle x, y\rangle \downarrow$" is provable in **HA**. Then we can describe the universal formula for Σ_2 by $\varphi_{\Sigma_2}(e, x) \coloneqq \exists y\, \varphi_{\Pi_1}(S(e, x), y) = \exists y \forall w\, \neg T(S(e, x), y, w)$, which is equivalent to $\exists y \forall w\, \neg T(e, \langle x, y\rangle, w)$ in **HA**.

Now let $p_1(e, x) \coloneqq \lambda w.0$ and define $s_2(e, x) \coloneqq \langle y_0, \lambda w.0\rangle$, where y_0 is the least number such that $\mathbb{N} \models \varphi_{\Pi_1}(S(e, x), y_0)$ holds if it exists. Note that p_1 is a total computable function and s_2 a partial $\emptyset^{(1)}$-computable one. Then, by the standard definition of realizability interpretation, we can easily verify that
(1) For any e, $x \in \mathbb{N}$, $\mathbb{N} \models \varphi_{\Pi_1}(e, x)$ implies $p_1(e, x)\ \mathbf{r}_{\mathrm{K}}\ \varphi_{\Pi_1}(e, x)$.
(2) For any e, $x \in \mathbb{N}$, $\mathbb{N} \models \varphi_{\Sigma_2}(e, x)$ implies $s_2(e, x)\ \mathbf{r}_{\mathrm{K}}^{(1)}\ \varphi_{\Sigma_2}(e, x)$.

The second property immediately implies that $\lambda ex.\lambda m.s_2(e,x)$ realizes ${}^{\forall}(\neg\neg\varphi_{\Sigma_2} \to \varphi_{\Sigma_2})$ under $\emptyset^{(1)}$-realizability, where $\lambda m.$ is a dummy abstraction. Hence Σ_2-**DNE** is true in $\mathcal{E}ff^{(1)}$. This allows us to estimate an upper bound for $j^{\mathcal{E}ff}_{\Sigma_2\text{-}\mathbf{DNE}}$. This argument can be straightforwardly extended to $j^{\mathcal{E}ff}_{\Sigma_{n+1}\text{-}\mathbf{DNE}}$ (Lemma 50 in Appendix B). So we have

▶ **Lemma 43.** *For any $n \geq 0$, $\mathcal{E}ff^{(n)} \models \Sigma_{n+1}$-**DNE**. Thus $j^{\mathcal{E}ff}_{\Sigma_{n+1}\text{-}\mathbf{DNE}} \leq j_{\emptyset^{(n)}}$.*

Next, notice that if a formula φ is transparent in $\mathcal{E}ff$, then we have $j^{\mathcal{E}ff}_{\varphi\vee\neg\varphi} = \ell^{\mathcal{E}ff}_{[\![\varphi\vee\neg\varphi]\!]_{\mathcal{E}ff}}$, where the latter is Joyal's least operator of subobject $[\![\varphi \vee \neg\varphi]\!]_{\mathcal{E}ff}$ (Lemma 24, Corollary 27). The following lemma gives us a simple description of $[\![\varphi_{\Pi_1} \vee \neg\varphi_{\Pi_1}]\!]_{\mathcal{E}ff}$.

▶ **Lemma 44.** *Let $\varphi(x)$ be an $\mathcal{L}_{\mathrm{A}}$-formula and let χ_φ be the characteristic function of $\{\, m \in \mathbb{N} \mid \mathbb{N} \models \varphi(m) \,\}$. Suppose further that there is a* total *computable function $p(x)$ such that the following two conditions hold:*
(a) *For any $m \in \mathbb{N}$, $\mathcal{E}ff \models \varphi(m)$ implies $\mathbb{N} \models \varphi(m)$.*
(b) *For any $m \in \mathbb{N}$, $\mathbb{N} \models \varphi(m)$ implies $p(m)\ \mathbf{r}_{\mathrm{K}}\ \varphi(m)$.*
Then $[\![\neg\varphi \vee \varphi]\!]_{\mathbf{Mfunc}} \equiv_s \chi_\varphi$ in $\mathbf{Mfunc} \simeq \mathrm{Sub}_{\mathcal{E}ff}(N)$.

Proof. Assume that $\varphi(x)$ and $p(x)$ satisfy (a) and (b), and let $m \in \mathbb{N}$. The assumptions imply that $p(m)\ \mathbf{r}_{\mathrm{K}}\ \varphi(m)$ iff $\mathbb{N} \models \varphi(m)$. Because $\neg\psi$ is Kleene realizable by any natural number iff ψ is not Kleene realizable for a closed formula ψ, we have $p(m)\ \mathbf{r}_{\mathrm{K}}\ \neg\varphi(m)$ iff $\mathbb{N} \models \neg\varphi(m)$. Hence we obtain that $\langle i, p(m)\rangle \in [\![\neg\varphi \vee \varphi]\!]_{\mathbf{Mfunc}}(m)$ iff $\chi_\varphi(m) = i$ for any $i \in \{\,0,1\,\}$. Therefore, $\lambda x.(x_1)_0$ and $\lambda x.\langle x_1, p(x_0)\rangle$ witness $[\![\neg\varphi \vee \varphi]\!]_{\mathbf{Mfunc}} \sqsubseteq_s \chi_\varphi$ and $\chi_\varphi \sqsubseteq_s [\![\neg\varphi \vee \varphi]\!]_{\mathbf{Mfunc}}$, respectively. ◀

Let us apply Lemma 44 to $\varphi \coloneqq \varphi_{\Pi_1}$ and $p \coloneqq p_1$. We have already mentioned that (b) holds. Moreover, (a) follows from the categorical equivalence $\mathcal{E}ff_{\neg\neg} \simeq \mathbf{Set}$ ([10, Proposition 4.4]) and Lemma 21 (this is another transparency argument). Thus we obtain $\ell^{\mathcal{E}ff}_{[\![\varphi\vee\neg\varphi]\!]_{\mathcal{E}ff}} = \ell^{\mathcal{E}ff}_{\chi_\varphi}$.

In addition, the subset of natural numbers defined by φ_{Π_1} has Turing degree $\emptyset^{(1)}$ by *Post's theorem*, so $\ell^{\mathcal{E}ff}_{\chi_{\varphi_{\Pi_1}}} = j_{\emptyset^{(1)}}$ holds in the sense of Notation 41. Hence we have the equation:

$$j^{\mathcal{E}ff}_{\Pi_1\text{-}\mathbf{LEM}} = j^{\mathcal{E}ff}_{\varphi_{\Pi_1}\vee\neg\varphi_{\Pi_1}} = \ell^{\mathcal{E}ff}_{[\![\varphi_{\Pi_1}\vee\neg\varphi_{\Pi_1}]\!]_{\mathcal{E}ff}} = \ell^{\mathcal{E}ff}_{\chi_{\varphi_{\Pi_1}}} = j_{\emptyset^{(1)}}.$$

This argument can also be extended to $j^{\mathcal{E}ff}_{\Pi_n\text{-}\mathbf{LEM}}$ (Lemma 51, 52 in Appendix B). Therefore, combining this with Lemma 42 (2) and Lemma 43, we obtain

▶ **Theorem 45.** *For any $n \geq 0$, $j^{\mathcal{E}ff}_{\Pi_n\text{-}\mathbf{LEM}} = j^{\mathcal{E}ff}_{\Sigma_n\text{-}\mathbf{LEM}} = j^{\mathcal{E}ff}_{\Sigma_{n+1}\text{-}\mathbf{DNE}} = j_{\emptyset^{(n)}}$.*

On the other hand, as in van Oosten's work [23], the least operator of $\Pi_1 \vee \Pi_1$-**DNE** is equal to Lifschitz operator $j_{\mathcal{L}if}$ in $\mathcal{E}ff$. His argument can be lifted to $\mathcal{E}ff^{(n)}$. We define a multifunction $U_{\mathcal{L}if^{(n)}} :\subseteq \mathbb{N} \rightrightarrows \mathbb{N}$ by

$$U_{\mathcal{L}if^{(n)}}(e) \coloneqq \{\, \langle 0, e\rangle \mid \mathbb{N} \models \varphi_{\Pi_{n+1}}(e_0) \,\} \cup \{\, \langle 1, e\rangle \mid \mathbb{N} \models \varphi_{\Pi_{n+1}}(e_1) \,\}.$$

Following his observation in [23], we see that the least operator of $U_{\mathcal{L}if^{(n)}}$ in $\mathcal{E}ff^{(n)}$ represents Lifschitz realizability relativized to degree $\emptyset^{(n)}$. Hence we write $j_{\mathcal{L}if^{(n)}}$ for that least operator. Recall that $j_{\mathcal{L}if}$ is strictly in between $j_\emptyset$ and $j_{\emptyset^{(1)}}$. This can be generalized to $j_{\emptyset^{(n)}} < j_{\mathcal{L}if^{(n)}} < j_{\emptyset^{(n+1)}}$ for any $n \geq 0$ since the proof can be relativized to $\emptyset^{(n)}$.

By using a "realizer" of $\varphi_{\Pi_{n+1}}$ (p_{n+1} in Lemma 50), we can prove the equivalence of $[\![\psi]\!]_{\mathcal{E}ff^{(n)}} \rightarrowtail [\![\neg\neg\psi]\!]_{\mathcal{E}ff^{(n)}}$ and $U_{\mathcal{L}if^{(n)}}$ as subobject in $\mathcal{E}ff^{(n)}$, where ψ is a universal formula for $\Pi_{n+1} \vee \Pi_{n+1}$. By reasoning similarly to the proof of Theorem 45, we conclude

▶ **Theorem 46.** *For any $n \geq 0$, $j^{\mathcal{E}ff}_{\Pi_{n+1}\vee\Pi_{n+1}\text{-}\mathbf{DNE}} = j_{\mathcal{L}if^{(n)}}$.*

We have thus determined all least operators of semi-classical axioms in $\mathcal{E}ff$ (Figure 5).

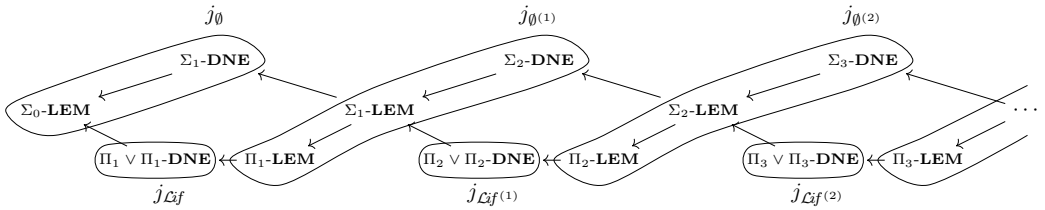


Figure 5 Summary of least operators in $\mathbf{Lop}(\mathcal{E}ff)$.

5 Conclusion

As we explained in Subsection 3.1, least dense operators behave as "invariants" under **HA**-provability. Figure 5 gives us the complete information about separation of semi-classical axioms in Figure 1 by subtoposes of $\mathcal{E}ff$.

▶ **Corollary 47.**

(1) *Any two axioms belonging to different circles in Figure 5 are separable.*

(2) *Those in the same circle are never separable by any subtopos of* $\mathcal{E}ff$.

While the first part of Corollary 47 is already established in [1], the second part is genuinely our original contribution. In addition:

- We have a complete characterization of separability of semi-classical axioms by a subtopos. Take Σ_n-**DNE** and $\Pi_n \vee \Pi_n$-**DNE** as an example. It follows from the nature of least operator that for any $k \in \mathbf{Lop}(\mathcal{E}ff)$, $\mathcal{E}ff_k$ separates them if and only if $j_{\emptyset^{(n-1)}} \leq k$ and $j_{\mathcal{L}if^{(n-1)}} \not\leq k$. This is a refinement of [1] and indicates that realizability notions they used are "optimal" in the sense of minimality.
- In addition to the separation results explained above, [1] also separates Π_n-**LEM** and Σ_n-**LEM** by using *monotone modified realizability*. As a by-product of Corollary 47 (2), we find that this realizability notion cannot be captured by a subtopos of $\mathcal{E}ff$. We are thus led to look for another suitable topos. A candidate is the topos $\mathcal{E}ff_{\rightarrow}$. proposed by van Oosten [24], that contains the *modified realizability topos* in addition to $\mathcal{E}ff$. In future work, we plan to explore such richer toposes and to determine the least dense operators of various axioms in them.
- Our detailed analysis of transparency gives a systematic account on previous work on least dense operators by Caramello and van Oosten. Since we have worked on an arbitrary topos, our results in Section 3 may also be applied to another semantics instead of realizability, *sheaf semantics* including Kripke frame semantics and Beth semantics [3].
- The major advantage of our framework is that we acquire a new methodology to prove impossibility of separation (Corollary 47 (2)). In further study of intuitionistic arithmetic, other variants of semi-classical axioms have been proposed, but many of them have not yet been separated [9]. The least dense operators may allow us to analyze the "difficulty" of separation from a topos-theoretic point of view.

References

1 Yohji Akama, Stefano Berardi, Susumu Hayashi, and Ulrich Kohlenbach. An arithmetical hierarchy of the law of excluded middle and related principles. In *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science (LICS'04)*, pages 192–201, 2004.

2 Stefano Berardi and Silvia Steila. Ramsey theorem for pairs as a classical principle in intuitionistic arithmetic. In *19th International Conference on Types for Proofs and Programs (TYPES 2013)*, pages 64–83, 2014.

3 Guram Bezhanishvili and Wesley H. Holiday. A semantic hierarchy for intuitionistic logic. *Indagationes Mathematicae*, 30(3):403–469, 2019.

4 Andreas Blass and Andrej Ščedrov. Boolean classifying topoi. *Journal of Pure and Applied Algebra*, 28(1):15–30, 1983.

5 Olivia Caramello. De Morgan classifying toposes. *Advances in Mathematics*, 222(6):2117–2144, 2009.

6 Olivia Caramello. Topologies for intermediate logics. *Mathematical Logic Quarterly*, 60(4):335–347, 2014.

7 Olivia Caramello. *Theories, Sites, Toposes Relating and studying mathematical theories through topos-theoretic 'bridges'*. Oxford University Press, 2018.

8 Eric Faber and Jaap van Oosten. More on geometric morphisms between realizability toposes. *Theory and Applications of Categories*, 29:874–895, 2014.

9 Makoto Fujiwara and Taishi Kurahashi. Refining the arithmetical hierarchy of classical principles. *Mathematical Logic Quarterly*, 68(3):318–345, 2022.

10 J. M. E. Hyland. The effective topos. In *The L. E. J. Brouwer Centenary Symposium*, volume 110 of Stud. Logic Foundations Math. North-Holland, pages 165–216, 1982.

11 J. M. E. Hyland, Peter T. Johnstone, and Andrew M. Pitts. Tripos theory. *Math. Proc. Cambridge Philos. Soc.*, 88:205–232, 1980.

12 Peter T. Johnstone. Conditions related to De Morgan's law. *Applications of sheaves*, 753:479–491, 1979.

13 Peter T. Johnstone. *Sketches of an Elephant: A Topos Theory Compendium, 2 vols*. Oxford Logic Guides 43, 44. Clarendon Press, 2002.

14 Takayuki Kihara. Lawvere-tierney topologies for computability theorists. *Transactions of the American Mathematical Society, Series B*, pages 48–85, 2023.

15 Stephen C. Kleene. On the interpretation of intuitionistic number theory. *Journal of Symbolic Logic*, 10:109–124, 1945.

16 Georg Kreisel. Interpretation of analysis by means of constructive functionals of finite types. In A. Heyting, editor, *Constructivity in Mathematics*, pages 101–128. Amsterdam: North-Holland Pub. Co., 1959.

17 J. Lambek and P. J. Scott. *Introduction to Higher Order Categorical Logic*. Cambridge University Press, 1986.

18 Vladimir Lifschitz. CT_0 is stronger than $CT_0!$. *Proceedings of the American Mathematical Society*, 73:101–106, 1979.

19 Saunders MacLane and Ieke Moerdijk. *Sheaves in Geometry and Logic: A First Introduction to Topos Theory*. Springer-Verlag, 1992.

20 Wesley Phoa. Relative computability in the effective topos. *Mathematical Proceedings of the Cambridge Philosophical Society*, 106:419–422, 1989.

21 Anne Sjerp Troelstra and Dirk van Dalen. *Constructivism in Mathematics, Vol.I*, volume 121 of *Lecture Notes in Mathematics*. Elsevier, 1988.

22 Anne Sjerp Troelstra (ed.). *Metamathematical investigation of intuitionistic arithmetic and analysis*, volume 344 of *Studies in Logic and the Foundations of Mathematics*. Springer-Verlag, 1973.

23 Jaap van Oosten. Two remarks on the Lifschitz realizability topos. *Journal of Symbolic Logic*, 61(1):70–79, 1996.

24 Jaap van Oosten. The modified realizability topos. *Journal of Pure and Applied Algebra*, 116:273–289, 1997.

25 Jaap van Oosten. *Realizability: an introduction to its categorical side*, volume 152 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, 2008.

A $\mathrm{Trp}^{\mathcal{E}}$ and Π_3

We here give an example of a Π_3-formula that is non-transparent. In fact, non-transparent formulas are easily found as in the following lemma.

▶ **Lemma 48.** *Let $\mathcal{E}$ be a topos and φ a formula such that $\mathcal{E} \models \varphi$ but $\mathcal{E}_{\neg\neg} \not\models \varphi$. Then $\langle\varphi\rangle^{\mathcal{E}} \neq \emptyset$ and φ has no least dense operator in $\mathcal{E}$. So φ is not transparent in $\mathcal{E}$.*

Proof. By assumption, we have $\mathrm{id}_\Omega \in \langle\varphi\rangle^{\mathcal{E}}$, so $\langle\varphi\rangle^{\mathcal{E}} \neq \emptyset$. Now assume that φ has least dense operator $j^{\mathcal{E}}_\varphi$ in $\mathcal{E}$. Then, $j^{\mathcal{E}}_\varphi = \mathrm{id}_\Omega$ holds by minimality. This leads to $\langle\varphi\rangle^{\mathcal{E}} = \mathbf{DLop}(\mathcal{E})$, and in particular $\neg\neg \in \langle\varphi\rangle^{\mathcal{E}}$, which contradicts the assumption that $\mathcal{E}_{\neg\neg} \not\models \varphi$. ◀

This suggests that "constructively true" but "classically false" formulas are likely to be non-transparent. Let us consider the effective topos $\mathcal{E}ff$ and recall that for a closed $\mathcal{L}_{\mathrm{A}}$-formula ψ, $\mathcal{E}ff \models \psi$ iff ψ is Kleene realizable. With this in mind, define Σ_2-formula $H(x,y)$ to be the graph of the characteristic function of *Halting problem*. That is,

$$H(x,y) \coloneqq \exists v \forall u((T(x,x,u) \to y = 1) \wedge (\neg T(x,x,v) \to y = 0)).$$

Furthermore, define *Church's thesis* with respect to H by

$$\mathrm{CT}_0^H \coloneqq \forall x \exists y H(x,y) \to \exists e \forall x \exists w (H(x, U(w)) \wedge T(e,x,w)),$$

where U is Kleene's U-function. Then $\mathcal{E}ff \models \mathrm{CT}_0^H$ is obtained by the fact that Church's thesis is Kleene realizable [22]. However, $\mathcal{E}ff_{\neg\neg} \not\models \mathrm{CT}_0^H$ because $\mathcal{E}ff_{\neg\neg}$ is equivalent to the category **Set** of sets.

▶ **Theorem 49.** $\Pi_3 \not\subseteq \mathrm{Trp}^{\mathcal{E}ff}$ *in the effective topos $\mathcal{E}ff$.*

Proof. It is obvious that the antecedent of CT_0^H is Π_3 and the consequent is equivalent to a Σ_4-formula in **HA**. In addition, the converse of CT_0^H is provable in **HA**.

Now assume that $\Pi_3 \subseteq \mathrm{Trp}^{\mathcal{E}ff}$. Then $\Sigma_4 \subseteq \mathrm{Trp}^{\mathcal{E}ff}$ by Lemma 25 (1), hence CT_0^H has least dense operator in $\mathcal{E}ff$ by Lemma 24 (2). This contradicts Lemma 48. ◀

B Proof of Theorem 45 in the general case

To show Theorem 45, we provide generalized versions of p_1, s_2 and Lemma 44 in Subsection 4.2. Recall that $S(e,x)$ denotes the function from the parameter theorem. Then we can inductively define universal formulas for Σ_n and Π_n by $\varphi_{\Sigma_{n+1}} \coloneqq \exists y\, \varphi_{\Pi_n}(S(e,x),y)$ and $\varphi_{\Pi_{n+1}} \coloneqq \forall y\, \varphi_{\Sigma_n}(S(e,x),y)$.

▶ **Lemma 50.** *For every $n \geq 0$, there are a* total $\emptyset^{(n)}$*-computable function $p_{n+1}(e,x)$ and a* partial $\emptyset^{(n)}$*-computable function $s_{n+1}(e,x)$ such that the following two conditions hold:*
(1) *For any e, $x \in \mathbb{N}$, $\mathbb{N} \models \varphi_{\Pi_{n+1}}(e,x)$ implies $p_{n+1}(e,x)\ \mathbf{r}_{\mathrm{K}}^{(n)}\ \varphi_{\Pi_{n+1}}(e,x)$.*
(2) *For any e, $x \in \mathbb{N}$, $\mathbb{N} \models \varphi_{\Sigma_{n+1}}(e,x)$ implies $s_{n+1}(e,x)\ \mathbf{r}_{\mathrm{K}}^{(n)}\ \varphi_{\Sigma_{n+1}}(e,x)$.*

Proof. By induction on n. For $n = 0$, we have already given $p_1(e,x) \coloneqq \lambda w.0$. Define a partial computable function s_1 by $s_1(e,x) \coloneqq \langle w_0, 0\rangle$, where w_0 is the code of computation history of $e \cdot x$ when $e \cdot x \downarrow$. It is clear from the description of $\varphi_{\Sigma_1}(e,x)$ that (2) holds.

Next assume that the statement holds for n. Let us define $p_{n+2}(e,x) \coloneqq \lambda y.s_{n+1}(S(e,x),y)$ and $s_{n+2}(e,x) \coloneqq \langle y_0, p_{n+1}(S(e,x),y_0)\rangle$, where y_0 is the least number such that $\varphi_{\Pi_{n+1}}(S(e,x),y_0)$ is true in $\mathbb{N}$ if it exists. Note that p_{n+2} can be obtained as a total $\emptyset^{(n)}$-computable function and s_{n+2} as a partial $\emptyset^{(n+1)}$-computable one. Then, for any $e,x \in \mathbb{N}$,

$$\begin{aligned}
\mathbb{N} \models \varphi_{\Pi_{n+2}}(e,x) &\Longrightarrow \forall y \in \mathbb{N} \quad \mathbb{N} \models \varphi_{\Sigma_{n+1}}(S(e,x),y) \\
&\Longrightarrow \forall y \in \mathbb{N} \quad s_{n+1}(S(e,x),y)\ \mathbf{r}_{\mathrm{K}}^{(n)}\ \varphi_{\Sigma_{n+1}}(S(e,x),y) \\
&\Longrightarrow p_{n+2}(e,x)\ \mathbf{r}_{\mathrm{K}}^{(n+1)}\ \forall y\, \varphi_{\Sigma_{n+1}}(S(e,x),y) \\
\mathbb{N} \models \varphi_{\Sigma_{n+2}}(e,x) &\Longrightarrow \exists y \in \mathbb{N} \quad \mathbb{N} \models \varphi_{\Pi_{n+1}}(S(e,x),y) \\
&\Longrightarrow \exists y \in \mathbb{N} \quad (\mathbb{N} \models \varphi_{\Pi_{n+1}}(S(e,x),y) \wedge p_{n+1}(S(e,x),y)\ \mathbf{r}_{\mathrm{K}}^{(n)}\ \varphi_{\Pi_{n+1}}(S(e,x),y)) \\
&\Longrightarrow s_{n+2}(e,x)\ \mathbf{r}_{\mathrm{K}}^{(n+1)}\ \exists y\, \varphi_{\Pi_{n+1}}(S(e,x),y). \qquad \blacktriangleleft
\end{aligned}$$

The condition (2) of Lemma 50 implies that $\lambda ex.\lambda m.s_{n+1}(e,x)$ realizes ${}^{\forall}(\neg\neg\varphi_{\Sigma_{n+1}} \to \varphi_{\Sigma_{n+1}})$ under $\emptyset^{(n)}$-realizability. This means that we have confirmed Lemma 43.

Next, let us generalize Lemma 44. It is simply given by relativizing to $\emptyset^{(n)}$, so the proof is obtained exactly in the same way.

▶ **Lemma 51.** *Let $\varphi(x)$ be an $\mathcal{L}_{\mathrm{A}}$-formula and let χ_φ be the characteristic function of $\{\, m \in \mathbb{N} \mid \mathbb{N} \models \varphi(m) \,\}$. Suppose further that there is a* total *$\emptyset^{(n)}$-computable function $p(x)$ such that the following two conditions hold:*
(a) *For any $m \in \mathbb{N}$, $\mathcal{E}ff^{(n)} \models \varphi(m)$ implies $\mathbb{N} \models \varphi(m)$.*
(b) *For any $m \in \mathbb{N}$, $\mathbb{N} \models \varphi(m)$ implies $p(m)\ \mathbf{r}_{\mathrm{K}}^{(n)}\ \varphi(m)$.*
Then $[\![\neg\varphi \vee \varphi]\!]^{(n)}_{\mathbf{Mfunc}} \equiv_s \chi_\varphi$ in $\mathrm{Sub}_{\mathcal{E}ff^{(n)}}(N)$.

Let $\varphi \coloneqq \varphi_{\Pi_{n+1}}$ and $p \coloneqq p_{n+1}$. Note that $\varphi \in \Pi_{n+1}$ is transparent in $\mathcal{E}ff^{(n)}$ by Lemma 29 and Lemma 43, so $\varphi \vee \neg\varphi$ has least operator $j^{\mathcal{E}ff^{(n)}}_{\varphi\vee\neg\varphi}$ in $\mathcal{E}ff^{(n)}$. By Lemma 51 and the same reasoning as in the proof for $n = 0$, we have the equation below in $\mathbf{Lop}(\mathcal{E}ff^{(n)})$:

$$j^{\mathcal{E}ff^{(n)}}_{\varphi\vee\neg\varphi} = \ell^{\mathcal{E}ff^{(n)}}_{[\![\varphi\vee\neg\varphi]\!]_{\mathcal{E}ff^{(n)}}} = \ell^{\mathcal{E}ff^{(n)}}_{\chi_\varphi}. \tag{♦}$$

The following lemma is the final piece to establish Theorem 45.

▶ **Lemma 52.** *Let j be a local operator in a topos $\mathcal{E}$ and $U \rightarrowtail X$ a subobject in $\mathcal{E}$. Further suppose that $j \le \ell^{\mathcal{E}}_U$, where $\ell^{\mathcal{E}}_U$ is the least operator of U in $\mathcal{E}$. Then $\ell^{\mathcal{E}}_U \in \mathbf{Lop}(\mathcal{E})^{\ge j}$ corresponds to the least operator $\ell^{\mathcal{E}_j}_{L_jU} \in \mathbf{Lop}(\mathcal{E}_j)$ of the subobject $L_jU \rightarrowtail L_jX$ in $\mathcal{E}_j$ in the sense of Notation 30.*

Proof. Fix $k \in \mathbf{Lop}(\mathcal{E})^{\ge j}$ and the corresponding local operator k_j in $\mathbf{Lop}(\mathcal{E}_j)$. Note that $L_k \simeq L_{k_j} \circ L_j$ holds since $\mathcal{E}_k$ is equal to the subtopos $(\mathcal{E}_j)_{k_j}$ of $\mathcal{E}_j$ corresponding k_j. Hence $L_kU \rightarrowtail L_kX$ is an isomorphism if and only if so is $L_{k_j}(L_jU) \rightarrowtail L_{k_j}(L_jX)$. Recalling Lemma 6, we obtain that $\ell^{\mathcal{E}}_U \le k$ in $\mathbf{Lop}(\mathcal{E})^{\ge j}$ iff U is k-dense in $\mathcal{E}$ iff $L_{k_j}(L_jU)$ is k_j-dense in $\mathcal{E}_j$ iff $\ell^{\mathcal{E}_j}_{L_jU} \le k_j$ in $\mathbf{Lop}(\mathcal{E}_j)$. This means that $\ell^{\mathcal{E}}_U$ corresponds to $\ell^{\mathcal{E}_j}_{L_jU}$. ◀

In particular, considering $X \coloneqq N$, $U \coloneqq \chi_\varphi$ and $j \coloneqq j_{\emptyset^{(n)}}$ in $\mathcal{E}ff$, we have the correspondence between $\ell^{\mathcal{E}ff}_U = j_{\emptyset^{(n+1)}} \in \mathbf{Lop}(\mathcal{E}ff)$ and $\ell^{\mathcal{E}ff^{(n)}}_{L_jU} \in \mathbf{Lop}(\mathcal{E}ff^{(n)})$. In addition, $\ell^{\mathcal{E}ff^{(n)}}_{L_jU}$ can be regarded as the least operator $\ell^{\mathcal{E}ff^{(n)}}_{\chi_\varphi}$ because the subobject $L_jU \rightarrowtail L_jN(= N_j)$ can be described by a multifunction χ_φ in $\mathcal{E}ff^{(n)}$. So we have the following correspondence:

$$\ell^{\mathcal{E}ff^{(n)}}_{\chi_\varphi} \in \mathbf{Lop}(\mathcal{E}ff^{(n)}) \quad \longleftrightarrow \quad j_{\emptyset^{(n+1)}} \in \mathbf{Lop}(\mathcal{E}ff). \tag{♠}$$

Thus, combining this correspondence (♠) with the equation (♦), we obtain

▶ **Theorem 53.** *For any $n \ge 0$, $j^{\mathcal{E}ff^{(n)}}_{\Pi_{n+1}\text{-}\mathbf{LEM}} \in \mathbf{Lop}(\mathcal{E}ff^{(n)})$ corresponds to $j_{\emptyset^{(n+1)}} \in \mathbf{Lop}(\mathcal{E}ff)$.*

Finally, we prove Theorem 45 by using the iteration argument developed in Subsection 3.4.

Proof of Theorem 45. It is sufficient to show that $j^{\mathcal{E}ff}_{\Pi_n\text{-}\mathbf{LEM}} = j_{\emptyset^{(n)}}$ by induction on n. The base case is already discussed in Subsection 4.2.

Assume that it holds for n; in particular, $k \coloneqq j^{\mathcal{E}ff}_{\Sigma_n\text{-}\mathbf{LEM}} = j_{\emptyset^{(n)}}$ holds. Recall that Π_{n+1}-**LEM** implies Σ_n-**LEM** over **HA** (Figure 1). By Lemma 32, the least operator $j^{\mathcal{E}ff}_{\Pi_{n+1}\text{-}\mathbf{LEM}}$ of Π_{n+1}-**LEM** in $\mathcal{E}ff$ corresponds to the least operator $j^{\mathcal{E}ff_k}_{\Pi_{n+1}\text{-}\mathbf{LEM}}$ of that in $\mathcal{E}ff_k$, which is described as $j^{\mathcal{E}ff^{(n)}}_{\Pi_{n+1}\text{-}\mathbf{LEM}}$. Therefore, we conclude $j^{\mathcal{E}ff}_{\Pi_{n+1}\text{-}\mathbf{LEM}} = j_{\emptyset^{(n+1)}}$ by Theorem 53. ◀

Coherence by Normalization for Linear Multicategorical Structures

Federico Olimpieri
School of Mathematics, University of Leeds, UK

Abstract

We establish a formal correspondence between resource calculi and appropriate linear multicategories. We consider the cases of (symmetric) representable, symmetric closed and autonomous multicategories. For all these structures, we prove that morphisms of the corresponding free constructions can be presented by means of typed resource terms, up to a reduction relation and a structural equivalence. Thanks to the linearity of the calculi, we can prove strong normalization of the reduction by combinatorial methods, defining appropriate decreasing measures. From this, we achieve a general coherence result: morphisms that live in the free multicategorical structures are the same whenever the normal forms of the associated terms are equal. As further application, we obtain syntactic proofs of Mac Lane's coherence theorems for (symmetric) monoidal categories.

2012 ACM Subject Classification Theory of computation → Type theory; Theory of computation → Categorical semantics

Keywords and phrases Coherence, Linear Multicategories, Resource Calculi, Normalization

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.43

Related Version *Full Version*: `https://arxiv.org/abs/2302.05755`

Funding *Federico Olimpieri*: This work was supported by the US Air Force Office for Scientific Research under award number FA9550-21-1-0007.

1 Introduction

The basis of the celebrated *Curry-Howard-Lambek correspondence* is that logical systems, typed λ-calculi and appropriate categorical constructions are different presentations of the same mathematical structure. An important consequence of the correspondence is that we can give *syntactical presentations* of categories, that can be exploited to prove general results by means of elementary methods, such as induction. At the same time, we can use categorical methods to obtain a more modular and clean design of programming languages. The classic example is given by simply typed λ-calculi and cartesian closed categories [22]. The idea is well-known: morphisms in free cartesian closed categories over sets are identified with equivalence classes of λ-terms up to $\beta\eta$-equality. Another important setting is the *linear* one, where we consider *monoidal* categories instead of cartesian ones. In this case, *linear logic* [11] enters the scene: symmetric monoidal closed categories correspond to *linear* λ-calculi. Computationally, this is a huge restriction, since linear terms can neither copy nor delete their inputs during computation. A refinement of this picture can be obtained by switching from categories to *multicategories* [21]. These structures were indeed first introduced by Lambek to achieve a categorical framework formally closer to typed calculi/proof systems. Morphisms of multicategories can have multiple sources $f : a_1, \dots, a_n \to a$, recalling the structure of a *type judgment* $x_1 : a_1, \dots, x_n : a_n \vdash f : a$.

We are interested in establishing a Curry-Howard-Lambek style correspondence for appropriate linear multicategories and then employ it to obtain *coherence results*. When we deal with complex structures such as tensor products, it becomes crucial to have a *decision process* to establish whether two arrows are equal. This is called a *coherence* problem. The

main example is Mac Lane's original result [24], which states that *all structural diagrams* in monoidal categories commute. If one considers more complex structures, the class of commutative diagrams is normally more restrictive. In the case of closed monoidal categories, Kelly and Mac Lane [17] associated *graphs* to structural morphisms, obtaining the following coherence result: two structural arrows between appropriate objects[1] are equal whenever their graph is the same. We aim to achieve coherence results for linear multicategories, building on Lambek's and Mints [28] intuition that coherence problems can be rephrased in the language of proof theory and obtained by exploiting appropriate notions of *normalization* for proofs/terms [21]. We do so by establishing a formal connection between *resource* calculi and *linear* multicategorical structures.

Main Results. We study free multicategorical constructions for (symmetric) representable and closed structures. Representability consists of the multicategorical monoidal structure [12]. We prove that free linear multicategories built on appropriate signatures can be presented by means of typed resource calculi, where morphisms correspond to equivalence classes of terms up to a certain equivalence. We handle the tensor product *via pattern-matching*, presented as a syntactic *explicit substitution*. The definition of our type systems is given in *natural deduction* style: we have introduction and elimination rules for each type constructor. Our work is conceptually inspired by an "*adjoint functors* point-of-view". A basic fact of the classic Curry-Howard-Lambek correspondence is that $\beta\eta$-equality can be expressed by means of the *unit* (η) and the *counit* (β) of the adjunction between products and arrow types. We generalize this observation to the multicategorical setting, thus introducing an appropriate reduction relation that corresponds to the representable structure. Indeed, a fundamental aspect of our work consists of the in depth study of resource terms rewriting. We introduce confluent and strongly normalizing reductions, that express the appropriate equalities. In order to do so, we exploit *action-at-distance* to define our operational semantics, that has proven to be a successful approach to calculi with explicit substitution [18, 1, 2]. An important feature of this approach is to distinguish the operational semantics, defined by action-at-distance, from a notion of *structural equivalence*, that deals with commutations of explicit substitution with the other syntactic constructors. This approach overcomes the classic difficulties of rewriting systems with explicit substitution, allowing us to obtain confluence and strong normalization in an elegant way. In this way, we get a general *coherence result*: two structural morphisms of linear multicategories are equal whenever the *normal forms* of their associated terms are equal. In the context of (symmetric) representable multicategories, we apply this result to obtain a *syntactic proof* of stronger coherence theorems, that can be seen as multicategorical versions of the classic MacLane coherence theorems for (symmetric) monoidal categories [24]. The coherence theorem for representable multicategories was already proved in [12]. We give an alternative type-theoretic proof for it. To our knowledge, the other coherence results that we present are new. Moreover, exploiting the equivalence between monoidal categories and representable multicategories established by Hermida [12], we are able to obtain the original Mac Lane's results as corollaries of our coherence theorems.

Related Work. Building on Lambek's original ideas, several researchers have advocated the use of multicategories to model computational structures. Hyland [14] proposed to rebuild the theory of pure λ-calculus by means of *cartesian operads*, that is one-object cartesian multicategories. The idea of seeing resource calculi as multicategories was first

[1] A restriction on the type of morphisms is needed due to the presence of the monoidal unit.

employed by Mazza *et al.* [26, 25]. We build on their approach, showing that these calculi correspond to appropriate *universal constructions*, namely free linear multicategories. The first resource calculus has been introduced by Boudol [6]. A similar construction was also independently considered by Kfoury [19]. Resource terms have gained special interest thanks to the definition by Ehrhard and Regnier of the *Taylor expansion* for λ-terms [9]. From this perspective, the resource calculus is a *theory of approximation of programs* and has been successfully exploited to study the computational properties of λ-terms [3, 35, 26, 30]. Our syntax is very close to the one of *polyadic calculi* or *rigid resource calculi* [26, 34]. We need to extend the standard operational semantics, adding an η-reduction and a reduction for explicit substitution. Our η-reduction is built from an expansion rule instead of a contraction, since η-expansion naturally fits the adjoint point-of-view, corresponding the the unit of the considered adjunction. In dealing with the technical rewriting issues, we follow [28, 16, 8], obtaining a terminating η-reduction. As already discussed, we handle the explicit substitution following Accattoli and Kesner methodology [18, 2, 1].

The calculi we present are also strongly related to *intuitionistic linear logic* [20, 4]. It is well-known that resource calculi can be seen as fragments of ILL [26, 25]. While ILL is presented *via* sequent calculus, we chose a natural deduction setting, this latter being directly connected to the "adjoint functors" point-of-view. Accattoli and Kesner approach to explicit substitution allows us to bypass the cumbersome commutation rules needed for ILL rewriting. Moreover, resource calculi are closer to the multicategorical definitions (their constructors being *unbiased* [23], *i.e.*, k-ary). Our handling of symmetries is also more canonical and explicit. We use the properties of *shuffle permutations*, in a way similar to Hasegawa [29] and Shulman [33], also inspired by our ongoing work on *bicategorical semantics* [27]. In this way, the type system is *syntax directed* and we are able to prove that, given a term, there exists at most one type derivation for it. The pioneering work of Mints [28] is very close to our perspective. Mints introduced a linear λ-calculus to study the coherence problem of closed category by the means of normalization. We build on that approach, extending it to several different structures and to the multicategorical setting.

Shulman's type theory for (symmetric) monoidal categories [33] does not employ explicit substitutions, being able to handle tensors in way similar to what happens with standard product types. Our proposal differs considerably from Shulman's, both in purpose and in implementation. While Shulman's goal is to start from the categorical structure and define a "practical" type theory to make computations, ours consists of establishing a formal correspondence between two *independent* worlds: resource calculi and linear multicategories and then employ it to prove results about the categorical structure.

Proof-theoretic methods to establish coherence results have been widely exploited and studied also in recent times, see for instance [7, 36]. *Graphical approaches* to monoidal structures [31] have been widely developed. Particularly interesting for our work are the Kelly-Mac Lane graphs [17], This approach has been extended *via* linear logic, thanks to the notion of *proof-net* [5, 13]. However, the handling of monoidal units needs extra care from this perspective, while the terms calculi approach can account for them without any particular complication.

2 Preliminaries

We introduce some concepts, notations and conventions that we will use in the rest of the paper.

Integers, Permutations and Lists. For $n \in \mathbb{N}$, we set $[n] = \{1, \dots, n\}$ and we denote by S_n the symmetric group of order n. The elements of S_n are permutations, that we identify with bijections $[n] \cong [n]$. Given $\sigma, \tau \in S_n$, we denote by $\sigma \circ \tau$ their composition. Given $\sigma \in S_n, \tau \in S_m$ we denote by $\sigma \oplus \tau : [n+m] \cong [n+m]$ the evident induced permutation. We now introduce the notion of shuffle permutation, that is crucial to obtain canonical type derivations for resource terms with permutations (Proposition 32).

▶ **Definition 1** (Shuffles). *Let $n_1, \dots, n_k \in \mathbb{N}$ with $n = \sum_{i=1}^{k} n_i$. A $(n_1, \dots, n_k)$-*shuffle *is a bijection $\sigma : \sum_{i=1}^{k} [n_i] \cong [n]$ such that the composite $[n_i] \hookrightarrow \sum_{i=1}^{k} [n_i] \cong [n]$ is monotone for all $i \in [k]$. We denote the set of all $(n_1, \dots, n_k)$-shuffles as $\mathsf{shu}(n_1, \dots, n_k)$.*

The relevant result on shuffles is the following, that induces canonical decomposition of arbitrary permutations over sums of integers.

▶ **Lemma 2.** *Every permutation $\sigma \in S_{\sum_{i=1}^{k} n_i}$ can be canonically decomposed as $\tau_0 \circ (\bigoplus_{i=1}^{k} \tau_i)$ with $\tau_0 \in \mathsf{shu}(n_1, \dots, n_k)$ and $\tau_i \in S_{n_i}$ for $i \in [k]$.*

Given a set A and a list of its elements $\gamma = a_1, \dots, a_k$ and $\sigma \in S_k$ we set $\gamma \cdot \sigma = a_{\sigma(1)}, \dots, a_{\sigma(k)}$ for the symmetric group right action. We write $\mathsf{len}(\gamma)$ for its length. We denote the *stabilisers* for this action as $\mathsf{Stab}(\gamma) = \{\sigma \in S_k \mid \gamma \cdot \sigma = \gamma\}$. Given lists $\gamma_1, \dots, \gamma_k$, we set $\mathsf{shu}(\gamma_1, \dots, \gamma_k) = \mathsf{shu}(\mathsf{len}(\gamma_1), \dots, \mathsf{len}(\gamma_k))$.

Multicategories. Multicategories constitute the main object of our work. A multicategory is a multigraph that comes equipped with an appropriate composition operation.

▶ **Definition 3.** *A* multigraph *$\mathcal{G}$ is given by the following data:*
- *A collection of* nodes *$\mathcal{G}_0 \ni a, b, c \dots$*
- *For every $a_1, \dots, a_n, b \in \mathcal{G}_0$, a collection of* multiarrows *$\mathcal{G}(a_1, \dots, a_n; b) \ni s, t, u \dots$*

We denote by $arr(\mathcal{G})$ the set of all multiarrows of $\mathcal{G}$.

▶ **Definition 4.** *A* multicategory *is a multigraph $\mathcal{G}$ equipped with the following additional structure:*
- *A* composition operation $- \circ \langle -, \dots, - \rangle : \mathcal{G}(a_1, \dots, a_n; b) \times \prod_{i=1}^{n} \mathcal{G}(\gamma_i, a_i) \to \mathcal{G}(\gamma_1, \dots, \gamma_n; a)$.
- identities $id_a \in \mathcal{G}(a, a)$.

The former data is subjected to evident associativity and identity axioms. We call objects *the nodes of $\mathcal{G}$ and* morphisms *its multiarrows.*

A multicategory can be equipped with structure. We now introduce the notions of *symmetric*, *closed* and *representable* multicategories.

▶ **Definition 5.** *A multicategory $\mathcal{M}$ is* symmetric *if, for $\sigma \in S_k$ we have a family of bijections $- \cdot \sigma : \mathcal{M}(\gamma, a_1, \dots, a_k; a) \cong \mathcal{M}(\gamma, a_{\sigma(1)}, \dots, a_{\sigma(k)}; a)$ that satisfies additional axioms [23].*

▶ **Definition 6.** *A (right)* closed *structure for a multicategory $\mathcal{M}$ is given by a family of objects $(a_1 \otimes \dots \otimes a_k) \multimap a \in \mathcal{M}$ and arrows $ev_{a_1, \dots, a_k, a} : a_1, \dots, a_k, (a_1 \otimes \dots \otimes a_k) \multimap a \to a$, for $a_1, \dots, a_k, a \in \mathcal{M}$, such that the maps*

$$ev \circ \langle -, id_{a_1}, \dots, id_{a_k} \rangle : \mathcal{M}(\gamma; (a_1 \otimes \dots \otimes a_k) \multimap a) \to \mathcal{M}(\gamma, a_1, \dots, a_k; a)$$

induce a bijection, multinatural in γ and natural in a. We write $\lambda(-)$ to denote the inverses to these maps.

▶ **Definition 7.** *A* representable *structure for a multicategory $\mathcal{M}$ is given by a family of objects $(a_1 \otimes \cdots \otimes a_k) \in \mathcal{M}$ and arrows $\mathsf{re}_{a_1,\dots,a_k} : a_1, \dots, a_k \to (a_1 \otimes \cdots \otimes a_k)$, for $a_1, \dots, a_k \in \mathcal{M}$, such that he maps*

$$- \circ \langle id_\gamma, \mathsf{re}, id_\delta \rangle : \mathcal{M}(\gamma, (a_1 \otimes \cdots \otimes a_k), \delta; a) \to \mathcal{M}(\gamma, a_1, \dots, a_k, \delta; a)$$

induce a bijection, multinatural in γ, δ and natural in a. We write $\mathsf{let}(-)$ to denote the inverses to these maps.

We use the name *autonomous multicategories* to denote symmetric representable closed multicategories. We have categories of representable multicategories (RepM), symmetric representable multicategories (RepsM), closed multicategories ($\mathsf{ClosedM}$) and autonomous multicategories (autoM), whose morphisms are functors that preserve the structure on the nose.

Signatures. We introduce signatures for the structures we consider.

▶ **Definition 8.** *A* representable signature *is a pair $\langle \mathsf{At}, \mathcal{R} \rangle$ where At is a set of atoms At and $\mathcal{R}$ is a multigraph with nodes generated by the following inductive grammar:*

$$\mathcal{R}_0 \ni a ::= o \in \mathsf{At} \mid (a_1 \otimes \cdots \otimes a_k) \qquad (k \in \mathbb{N}).$$

▶ **Definition 9.** *A* closed signature *$\mathcal{L}$ is a pair $\langle \mathsf{At}, \mathcal{L} \rangle$ where At is a set of atoms At and $\mathcal{L}$ is a multigraph with with nodes generated by the following inductive grammar:*

$$\mathcal{L}_0 \ni a ::= o \in \mathsf{At} \mid (a_1 \otimes \cdots \otimes a_k) \multimap a \qquad (k \in \mathbb{N}).$$

▶ **Definition 10.** *An* autonomous signature *is a pair $\langle \mathsf{At}, \mathcal{H} \rangle$ where At is a set of atoms At and $\mathcal{H}$ is a multigraph with nodes generated by the following inductive grammar:*

$$\mathcal{H}_0 \ni a ::= o \in \mathsf{At} \mid (a_1 \otimes \cdots \otimes a_k) \mid (a_1 \otimes \cdots \otimes a_k) \multimap a \qquad (k \in \mathbb{N}).$$

A signature is *discrete* whenever the collections of multiarrows are empty. We shall often identify a signature with its graph. There are categories $\mathsf{ClosedSig}, \mathsf{RepSig}$ and $\mathsf{AutoSig}$ for, respectively, closed, representable and autonomous signatures. We have forgetful functors from the categories $\mathsf{ClosedM}, \mathsf{RepM}$ and autoM, which we denote by $(\overline{-})$. One of the main goals of this paper is to build the left adjoints to those functors *via* appropriate resource calculi.

Monoidal Categories vs Representable Multicategories. In order to transport coherence results from (symmetric) representable multicategories to ordinary (symmetric) monoidal categories, we shall employ an equivalence result due to Hermida [12, Theorem 9.8]. Let Mon be the category of monoidal categories and lax monoidal functors.

▶ **Theorem 11** ([12])**.** *There is an equivalence of categories* $\mathsf{RepM} \underset{\mathsf{mon}(-)}{\overset{\mathsf{rep}(-)}{\underset{\simeq}{\rightleftarrows}}} \mathsf{Mon}$.

The representable structure of a monoidal category $(\mathbb{M}, \otimes_{\mathbb{M}}, 1)$ is given by $(a_1 \otimes_{\mathbb{M}} \cdots \otimes_{\mathbb{M}} a_k) = (a_1) \otimes_{\mathbb{M}} (a_2 \otimes_{\mathbb{M}} (\cdots \otimes_{\mathbb{M}} a_k) \dots)$. Then composition needs a choice of structural isomorphisms of $\mathbb{M}$ to be properly defined [12, Definition 9.2][2]. The former equivalence can be extended to the symmetric case in the natural way.

[2] If we assume Mac Lane's Coherence Theorem, the choice is unique. However, we shall not do so, since we are going to exploit Theorem 11 to *transport* an appropriate coherence theorem on representable multicategories to ordinary monoidal categories, thus obtaining the Mac Lane's result as corollary.

$$\frac{f \in \mathcal{R}(a_1, \dots, a_n; b) \qquad \gamma_1 \vdash s_1 : a_1 \dots \gamma_n \vdash s_n : a_n}{(\gamma_1, \dots, \gamma_n) \vdash f(s_1, \dots, s_n) : b} \qquad \frac{\gamma_1 \vdash s_1 : a_1 \dots \gamma_k \vdash s_k : a_k}{\gamma_1, \dots, \gamma_k \vdash \langle s_1, \dots, s_k \rangle : (a_1 \otimes \dots \otimes a_k)}$$

$$\frac{a \in \mathcal{R}_0}{x : a \vdash x : a} \qquad \frac{\gamma \vdash s : (a_1 \otimes \dots \otimes a_k) \qquad \delta, x_1 : a_1, \dots, x_k : a_k, \delta' \vdash t : b}{\delta, \gamma, \delta' \vdash t[x_1^{a_1}, \dots, x_k^{a_k} := s] : b}$$

$$\mathtt{C} ::= [\cdot] \mid \langle s_1, \dots, \mathtt{C}, \dots, s_k \rangle \mid \mathtt{C}[\vec{x} := t] \mid s[\vec{x} := \mathtt{C}] \mid f(s_1, \dots, \mathtt{C}, \dots, s_k).$$
$$\mathtt{E} ::= [\cdot] \mid \langle s_1, \dots, \mathtt{E}, \dots, s_k \rangle \mid \mathtt{E}[\vec{x} := s] \mid s[\vec{x} := \mathtt{E}] \quad (\mathtt{E} \neq [\cdot]) \mid f(s_1, \dots, \mathtt{E}, \dots, s_k).$$
$$\mathtt{L} ::= [\cdot] \mid \mathtt{L}[\vec{x} := t].$$

Figure 1 Representable Type System on a signature $\mathcal{R}$ and contexts with one hole. Types are the elements of $\mathcal{R}_0$.

Notations and Conventions. Given a set of terms A and a reduction relation $\to_\epsilon \subseteq A \times A$, we denote respectively as $\twoheadrightarrow_\epsilon$ and $\to_\epsilon^*$ its transitive closure and its transitive and reflexive closure. We denote by $=_\epsilon \subseteq A \times A$ the smallest equivalence relation generated by $\to_\epsilon$. For a confluent reduction, we denote by $\mathsf{nf}(s)_\epsilon$ the normal form of s, if it exists. Given an equivalence relation $\mathsf{e} \subseteq A \times A$, and $s \in A$, we denote by $[s]_\mathsf{e}$ the corresponding equivalence class. We will often drop the annotation and just write $[s]$. We fix a countable set of variables $\mathcal{V}$, that we will use to define each calculi. Terms are always considered up to renaming of bound variables. Given terms $s, t_1, \dots, t_k$ and variables $x_1, \dots, x_k$ we write $s\{t_1, \dots, t_k / x_1, \dots, x_k\}$ to denote capture-avoiding substitutions. We often use the abbreviation $s\{\vec{t}/\vec{x}\}$. To define reduction relations, we rely on appropriate notions of *contexts with one hole*. Given a context with hole $\mathtt{C}$ and a term s we write $\mathtt{C}[s]$ for the capture-allowing substitution of the holes of $\mathtt{C}$ by s. The *size* of a term $\mathtt{size}(s)$ is the number of syntactic constructors appearing in its body. The calculi we shall introduce are typed *à la Church*, but we will constantly keep the typing implicit, to improve readability. Given $\gamma \vdash s : a$ we write $\mathtt{C}[\delta \vdash p : b] = s$ meaning that $\mathtt{C}[p] = s$ and the type derivation of $\gamma \vdash p : b$ contains a subderivation with conclusion $\delta \vdash p : b$. Given a typing judgment $x_1 : a_1, \dots, x_n : a_n \vdash s : a$ we shall consider variables appearing in the typing context as bound and we will work up to renaming of those variables. We write $\pi \triangleright \gamma \vdash s : a$ meaning that π is a type derivation of conclusion $\gamma \vdash s : a$. For any typing rule with multiple typing contexts, we assume those contexts to be disjoint.

3 A Resource Calculus for Representable Multicategories

We present our calculus for representable multicategories. We begin by introducing its syntax and typing, then we discuss its operational semantics. We prove confluence and strong normalization for its reduction. We show that equivalence classes of terms modulo reduction and a notion of *structural equivalence* define the morphisms of free representable multicategories over a signature. As an application of this result, we give a proof of the coherence theorem for representable multicategories.

Representable Terms. Let $\mathcal{R}$ be a representable signature. The *representable resource terms* over $\mathcal{R}$ are defined by the following inductive grammar:

$$\Lambda_{\mathsf{rep}}(\mathcal{R}) \ni s, t ::= x \in \mathcal{V} \mid \langle s_1, \dots, s_k \rangle \mid s[x_1^{a_1}, \dots, x_k^{a_k} := t] \mid f(s_1, \dots, s_k)$$

for $k \in \mathbb{N}$ and $f \in \mathrm{arr}(\mathcal{R}), a_i \in \mathcal{R}$. A term of the shape $\langle s_1, \dots, s_k \rangle$ is called a *list*. A term of the shape $s[x_1, \dots, x_k := t]$ is called an *(explicit) substitution*. Variables under the scope of an explicit substitution are bound. Given a term s, we denote by $\mathsf{ST}(s)$ the set of its *subterms* defined in the natural way.

▶ Remark 12. Our calculus follows the linear logic tradition of modelling the tensor product structure by means of a let constructor [4]. We opted for the syntactic choice of an explicit substitution $s[x_1, \dots, x_k := t]$, which stands for the more verbose let expression, let $\langle x_{,1}, \dots, x_k \rangle := t$ in s. Terms of the shape $f(s_1, \dots, s_k)$ are needed to capture the multiarrows induced by the signature $\mathcal{R}$.

Typing and contexts with hole for representable terms is defined in Figure 1. A context is *atomic* when it contains just atomic types. We define the following subset of terms $\mathsf{LT} = \{\mathtt{L}[\langle s_1, \dots, s_k \rangle] \mid \text{ for some context } \mathtt{L} \text{ and terms } s_i\}$.

▶ Remark 13. The condition about disjoint contexts grants *linearity*. A term is *linear* when each variable appears at most once in its body. It is easy to check that, by construction, all typed terms are linear. Moreover, given $\gamma \vdash s : a$, the context γ is *relevant*, meaning that it contains just the free variables of s.

A type of the shape $(a_1 \otimes \cdots \otimes a_k)$ is called a k-ary *tensor product*. We use a vector notation to refer to arbitrary tensors, *eg.*, $\vec{a}, \vec{b} \dots$ If $k = 0$, the type $()$ is also called the *unit*. We set $\Lambda_{\mathsf{rep}}(\mathcal{R})(a_1, \dots, a_n; a) = \{s \mid x_1 : a_1, \dots, x_n : a_n \vdash_{\mathsf{rep}} s : a \text{ for some } x_i \in \mathsf{fv}(s).\}$. We observe that, given a representable term $\gamma \vdash s : a$, there exists a unique type derivation for it.

Terms Under Reduction. We now introduce the reduction relation for representable terms. This relation consists of the union of two different subreductions: β and η reductions, defined in Figure 2. The *structural equivalence* on terms is defined as the smallest congruence on terms generated by the rule of Figure 2. We assume that the context $\mathtt{C}$ does not bind any variable of t.

▶ Remark 14. Our η-reduction consists of a restricted version of the standard notion of η-expansion, the restriction is needed to achieve strong normalization. We build on a well-established tradition in term rewriting [28, 16, 8]. Unrestricted η is trivially non-terminating. Indeed, for $x : (a \otimes b) \vdash x : (a \otimes b)$ we have the non-terminating chain $x \to_\eta \langle x, y \rangle[x, y := z] \to_\eta \langle x, y \rangle[x, y := \langle v, w \rangle[v, w := z]] \to_\eta \dots$ Hence, we need to forbid η-reduction on the right side of a substitution term, that is exactly what the restricted η-contexts do. Moreover, there is also a problem of interaction between η and β. Consider $s = \langle x, y \rangle$ well-typed, then we can produce the non-terminating chain $s \to_\eta \langle v, w \rangle[v, w := \langle x, y \rangle] \to_\beta s \to_\eta \dots$ For this reason, the root-step of η has to be restricted too. The presence of a substitution context in the β-rule is an action-at-distance [2], that allows to "free" possible blocked redexes, as the following one $x[x := (\langle y \rangle[y := z])]$. In this way, we can bypass traditional commutation rules and retrieve good rewriting properties. Structural equivalence intuitively says that explicit substitutions can "freely travel" in the body of a term.

We prove that typings are preserved under reduction and structural equivalence.

▶ **Proposition 15** (Subject Reduction and Equivalence). *Let $s \to_{\mathsf{rep}} s'$ or $s \equiv s'$ with $\gamma \vdash s : a$. then $\gamma \vdash s' : a$.*

Proof. The proof is by induction on $s \to_{\mathsf{rep}} s'$ and $s \equiv s'$ and exploits an appropriate substitution lemma. ◀

We now prove that the structural equivalence is a strong bisimulation for the reduction $\to_{\mathsf{rep}}$. Intuitively, this means that the equivalence does not affect terms rewriting.

▶ **Proposition 16.** *If $s' \equiv s$ and $s \to_{\mathsf{rep}} t$ there exists a term t' s.t. $t' \equiv t$ and $s' \to_{\mathsf{rep}} t'$.*

We show that we can associate appropriate measures to terms that decrease under reduction. For β, we just consider the size of terms. For η, we build on Mints approach [28]. We define the *size* of a type by induction: $\mathtt{size}\,(o) = 0, \mathtt{size}\,(\langle a_1, \dots, a_k\rangle) = 1 + \sum \mathtt{size}\,(a_i)$. Given $\gamma \vdash s : a$ we define a set of typed subterms of s: $\mathsf{EST}(s) = \{\delta \vdash p : a \mid p \in \mathsf{ST}(s) \setminus \mathsf{LT}$ s.t. $\mathtt{E}[\delta \vdash p : a] = s$ for some context $\mathtt{E}\}$. We set $\eta(s) = \sum_{\delta \vdash p:a \in \mathsf{EST}(s)} \mathtt{size}\,(a)$.

▶ Remark 17. The size of terms decreases under β-reduction as a consequence of *linearity*. Redexes cannot be copied nor deleted under reduction, since the substitution is linear. This fact is trivially false for standard λ-calculi, where the size of terms can possibly grow during computation. The intuition behind the η measure is that we are counting all subterms of s on which we could perform the η-reduction. The restrictions on the shape of $p \in \mathsf{EST}(s)$ is indeed directly derived from the ones on η-reduction.

▶ **Proposition 18.** *The following statements hold. If $s \to_\beta s'$ then $\mathtt{size}\,(s') < \mathtt{size}\,(s)$; if $s \to_\eta s'$ then $\eta(s') < \eta(s)$.*

▶ **Proposition 19.** *The reductions $\to_\beta$ and $\to_\eta$ are separately strongly normalizing and confluent.*

Proof. Strong normalization is a corollary of the former proposition. For confluence, first one proves local confluence by induction and then apply Newman's Lemma. ◀

We want to extend the result of separate strong normalization and confluence to the whole $\to_{\mathsf{rep}}$-reduction. To do so, we prove that β and η suitably *commutes*.

▶ **Proposition 20.** *If $s \to_\beta^* t \to_\eta^* t'$ there exists s' s.t. $s \to_\eta^* s'$ and $s' \to_\beta^* t'$.*

▶ **Theorem 21.** *The reduction $\to_{\mathsf{rep}}$ is confluent and strongly normalizing.*

Proof. Strong normalization is achieved by observing that any infinite reduction chain of $\to_{\mathsf{rep}}$ would trigger, by Proposition 20, an infinite reduction chain for η, that is strongly normalizing. Confluence is achieved by first proving local confluence and then by applying Newman's Lemma. ◀

Given $s \in \Lambda_{\mathsf{reps}}(\mathcal{R})(\gamma; a)$, we denote by $\mathsf{nf}(s)$ its unique normal form. As a corollary of subject reduction, we get that $\mathsf{nf}(s) \in \Lambda_{\mathsf{reps}}(\mathcal{R})(\gamma; a)$. We shall now present an inductive characterization of $\to_{\mathsf{rep}}$-normal terms for the case where $\mathcal{R}$ is a *discrete* signature.

▶ **Definition 22.** *Consider the following set, inductively defined:*

$$\mathsf{nf}(\Lambda_{\mathsf{rep}}(\mathcal{R})) \ni s ::= v[\vec{x}_1 := x_1] \dots [\vec{x}_n := x_n] \qquad v ::= \langle v_1, \dots, v_k\rangle \mid x$$

where $k, n \in \mathbb{N}, \gamma \vdash p : o$ with o being an atomic type and $\delta \vdash v : a$ with δ being atomic.

▶ **Proposition 23.** *A term $s \in \Lambda_{\mathsf{rep}}(\mathcal{R})$ is a normal form for $\to_{\mathsf{rep}}$ iff there exists $s' \in \mathsf{nf}(\Lambda_{\mathsf{rep}}(\mathcal{R}))$ s.t. $s \equiv s'$.*

Proof. ($\Rightarrow$) By induction on s. If $s = x$ then $s \in \mathsf{nf}(\Lambda_{\mathsf{rep}}(\mathcal{R}))$. If $\langle s_1, \dots, s_k\rangle$ then s_i are normal form. Then we apply the IH and get $s_i' \in \mathsf{nf}(\Lambda_{\mathsf{rep}}(\mathcal{R}))$ s.t. $s_i \equiv s_i'$. By definition $s_i' = v_i[\vec{x}_{i,1} := x_{i,1}] \dots [\vec{x}_{i,1} := x_{i,n_i}]$. We then set $s' = \langle v_1, \dots, v_k\rangle[\vec{x}_{1,1} := x_{1,n_1}] \dots [\vec{x}_{k,1} := x_{k,n_k}]$. If $s = p[\vec{x} := q]$ we have that p is a normal form and q is a β-normal form. We reason by cases on q. If q does not have η-redexes, we apply the IH and conclude in a way similar to the list case. If q has η-redexes, since s is β normal we have that $q \notin \mathsf{LT}$. We can prove that $q = x[\vec{x}_1 := q_1] \dots [\vec{x}_1 := q_n]$ with q_i hereditarely of the same shape. Hence we conclude by pushing out all the substitutions from left to right. ◀

$$\beta \text{ Root step:} \quad s[x_1^{a_1}, \dots, x_k^{a_k} := \mathtt{L}[\langle t_1, \dots, t_k \rangle]] \to_\beta \mathtt{L}[s\{t_1, \dots, t_k / x_1, \dots, x_k\}].$$

$$\eta \text{ Root step:} \quad s \to_\eta \vec{x}[\vec{x}^{\vec{a}} := s] \qquad \text{where } \vec{x} \text{ fresh }, \ \gamma \vdash s : \vec{a}, \ s \notin \mathsf{LT}.$$

$$\text{Contextual extensions:} \quad \frac{s \to_\beta s'}{\mathtt{C}[s] \to_\beta \mathtt{C}[s']} \qquad \frac{s \to_\eta s'}{\mathtt{E}[s] \to_\eta \mathtt{E}[s']} \qquad (\to_{\mathsf{rep}} = \to_\beta \cup \to_\eta).$$

$$\text{Structural equivalence:} \quad \mathtt{C}[s[\vec{x} := t]] \equiv \mathtt{C}[s][\vec{x} := t] \qquad \vec{x} \notin \mathsf{fv}(\mathtt{C}).$$

Figure 2 Representable reduction relations and structural equivalence.

Free Representable Multicategories. Let $\mathcal{R}$ be a representable signature. First, we define a multicategory $\mathsf{RM}(\mathcal{R})$ by setting $\mathrm{ob}(\mathsf{RM}(\mathcal{R})) = \mathcal{R}_0$ and $\mathsf{RM}(\mathcal{R})(\gamma; a) = \Lambda_{\mathsf{rep}}(\mathcal{R})(\gamma; a)/\sim$ where $\sim = (\equiv \cup =_{\mathsf{rep}})$. Composition is given by substitution, identities are given by variables. The operation is well-defined on equivalence classes and satisfies associativity, identity axioms. We also have that if $s \sim s'$, then $\mathsf{nf}(s) \equiv \mathsf{nf}(s')$. We denote by $\eta_{\mathcal{R}} : \mathcal{R} \to \overline{\mathsf{RM}(\mathcal{R})}$ the evident inclusion.

▶ **Proposition 24** (Representability). *We have a bijection* $\mathsf{RM}(\mathcal{R})(\gamma, (a_1 \otimes \cdots \otimes a_k), \delta; a) \cong \mathsf{RM}(\mathcal{R})(\gamma, a_1, \dots, a_k, \delta; a)$ *multinatural in* γ, δ *and natural in* a*, induced by the map* $[s] \mapsto [s\{\langle x_1, \dots, x_k \rangle / x\}]$.

Proof. Naturality follows from basic properties of substitution. Inverses are given by the maps $(-)[\vec{x} := x] : \mathsf{RM}(\mathcal{R})(\gamma, a_1, \dots, a_k, \delta; a) \to \mathsf{RM}(\mathcal{R})(\gamma, (a_1 \otimes \cdots \otimes a_k), \delta; a)$. ◀

▶ **Definition 25.** *Let* $\mathcal{R}$ *be a representable signature and* S *be a representable multicategory. Let* $i : \mathcal{R} \to \overline{\mathsf{S}}$ *be a map of representable signatures. We define a family of maps* $\mathsf{RT}(i)_{\gamma,a} : \Lambda_{\mathsf{rep}}(\mathcal{R})(\gamma; a) \to \mathsf{S}(i(\gamma); i(a))$ *by induction as follows:*

$$\mathsf{RT}(i)_{a,a}(x) = id_{i(a)} \qquad \mathsf{RT}(i)_{\gamma_1,\dots,\gamma_k,(a_1 \otimes \cdots \otimes a_k)}(\langle s_1, \dots, s_k \rangle) = \bigotimes_{i=1}^{k} \mathsf{RT}(i)_{\gamma_i, a_i}(s_i)$$

$$\mathsf{RT}(i)_{\delta_1,\gamma,\delta_2,a}(s[x_1, \dots, x_k := t]) = \mathsf{let}(\mathsf{RT}(i)_{\delta_1,a_1,\dots,a_k,\delta_2,a}(s)) \circ \langle id_{\delta_1}, \mathsf{RT}(i)_{\gamma,(a_1 \otimes \cdots \otimes a_k)}(t), id_{\delta_2} \rangle$$

$$\mathsf{RT}(i)_{\gamma_1,\dots,\gamma_n,a}(f(s_1, \dots, s_n)) = i(f) \circ \langle \mathsf{RT}(i)(s_1), \dots, \mathsf{RT}(i)(s_n) \rangle.$$

▶ **Theorem 26** (Free Construction). *Let* S *be a a representable multicategory and* $i : \mathcal{R} \to \overline{\mathsf{S}}$ *a map of representable signatures. There exists a unique representable functor* $i^* : \mathsf{RM}(\mathcal{R}) \to \mathsf{S}$ *such that* $i = \overline{i^*} \circ \eta_{\mathcal{R}}$.

Proof. The functor is defined exploiting Definition 25. ◀

Coherence Result. We fix a *discrete* representable signature $\mathcal{R}$. We show that if $s, t \in \mathsf{RM}(\mathcal{R})(\gamma; a)$, then $s = t$. Our proof strongly relies on the characterization of normal forms given in Proposition 23.

▶ **Lemma 27.** *Let* γ, γ' *be atomic contexts. If there exists a type* a *and normal terms* s, s' *such that* $s, s' \in \Lambda_{\mathsf{rep}}(\mathcal{R})(\gamma; a)$ *then* $\gamma = \gamma'$ *and* $s \equiv s'$.

▶ **Theorem 28.** *Let* s, s' *be normal terms s.t.* $s, s' \in \Lambda_{\mathsf{rep}}(\mathcal{R})(\gamma; a)$*, then* $s \equiv s'$.

Proof. By Proposition 23, $s \equiv t = (v[\vec{x}_1 := x_1] \dots [\vec{x}_p := x_p])$ and $s' \equiv t' = (v'[\vec{y}_1 := x'_1] \dots [\vec{y}_p := x'_{p'}])$. We prove that $t \equiv t'$ by induction on $p \in \mathbb{N}$. If $p = 0$ then t is

either a list or a variable. We proceed by cases. If $t = x$ then $\gamma = o$ and $a = o$ for some atomic type o. By the former lemma we have that $t \equiv t'$. If $t = \langle v_1, \dots, v_k \rangle$ the result is again a corollary of the former lemma since, by Definition 22, γ is atomic. If $p = n + 1$ then $t = v[\vec{x}_1 := x_1] \dots [\vec{x}_{n+1} := x_{n+1}]$ and, by definition of typing we have

$$\frac{x_{n+1} : \vec{a} \vdash x_{n+1} : \vec{a} \qquad \delta_1, \vec{x}_{n+1} : \vec{a}, \delta_2 \vdash v[\vec{x}_1 := x_1] \dots [\vec{x}_n := x_n] : a}{\delta_1, x_{n+1} : \vec{a}, \delta_2 \vdash s : a}$$

with $\gamma = \delta_1, x_{n+1} : \vec{a}, \delta_2$. Since $t' \in \mathsf{nf}(\Lambda_{\mathsf{rep}})(\gamma; a)$, there exists $i \in \mathbb{N}$ such that $t' = v'[\vec{y}_1 := x'_1] \dots [\vec{y}_i := x'_i] \dots [\vec{y}_p := x'_{p'}]$ and $x'_i = x_{n+1}$. By *structural equivalence* we have that $t' \equiv v'[\vec{y}_1 := x'_1] \dots [\vec{y}_p := x'_{p'}] \dots [\vec{y}_i := x_i]$. By definition of typing and by the hypothesis we have that

$$\frac{x'_i : \vec{a} \vdash x'_i : \vec{a} \qquad \delta_1, \vec{x}_i : \vec{a}, \delta_2 \vdash v'[\vec{y}_1 := x'_1] \dots [\vec{y}_{p'} := x'_{p'}] : a}{\delta_1, x_{n+1} : \vec{a}, \delta_2 \vdash s' : a}$$

By IH we have that $v[\vec{x}_1 := x_1] \dots [\vec{x}_p := x_n] \equiv v'[\vec{y}_1 := x'_1] \dots [\vec{y}_p := x'_{p'}]$. We can then conclude that $t \equiv t'$, by structural equivalence. ◀

▶ **Theorem 29** (Coherence for Representable Multicategories). *Let* $[s], [t] \in \mathsf{RM}(\mathcal{R})(\gamma; a)$. *Then* $[s] = [t]$.

▶ **Theorem 30** (Coherence for Monoidal Categories). *All diagrams in the free monoidal category on a set commute.*

Proof. Corollary of the former theorem and Theorem 11, by noticing that $\mathsf{mon}(\mathsf{RM}(\mathcal{R}))$ is the free monoidal category on the underlying set of $\mathcal{R}$. ◀

4 A Resource Calculus for Symmetric Representable Multicategories

The symmetric representable terms have exactly the same syntax and operational semantics as the representable ones. We first extend the type system in order to account for symmetries. We then study the free constructions establishing an appropriate coherence result.

The typing is defined in Figure 3. It is easy to see that the representable type system consists of a fragment of the symmetric one, where we just consider identity permutations. We write $\gamma \vdash_{\mathsf{srep}} s : a$ when we need to specify that the type judgment refers to the symmetric representable type system. We set $\Lambda_{\mathsf{reps}}(\mathcal{R})(a_1, \dots, a_n; a) = \{s \mid x_1 : a_1, \dots, x_n : a_n \vdash_{\mathsf{srep}} s : a \text{ for some } x_i \in \mathsf{fv}(s).\}$.

▶ Remark 31. The role of permutations in the type system of Figure 3 deserves some commentary. Instead of having an independent permutation rule, variables in contexts can be permuted only when contexts have to be merged. In this way, the system is *syntax directed*. The limitation to the choice of *shuffle permutation* is needed to get uniqueness of type derivations for terms. Indeed, consider $s = \langle \langle x, y \rangle, z \rangle$. If we allow the choice of arbitrary permutations, we could build the following derivations:

$$\frac{\dfrac{x : a \vdash x : a \quad y : b \vdash y : b \quad \sigma}{y : b, x : a \vdash \langle x, y \rangle : (a \otimes b)} \qquad z : a \vdash z : a \quad id}{y : b, x : a, z : a \vdash s : ((a \otimes b) \otimes a)} \qquad \frac{\dfrac{x : a \vdash x : a \quad y : b \vdash y : b \quad id}{x : a, y : b \vdash \langle x, y \rangle : (a \otimes b)} \qquad z : a \vdash z : a \quad \sigma \oplus id}{y : b, x : a, z : a \vdash s : ((a \otimes b) \otimes a)}$$

where σ is the swap. Thanks to the shuffle limitation, only the one on the left is allowed.

▶ **Proposition 32** (Canonicity of Typing). *If* $\pi \triangleright \gamma \vdash s : a$ *and* $\pi' \triangleright \gamma \vdash s : a'$ *then* $a = a'$ *and* $\pi = \pi'$.

$$\frac{a \in \mathcal{R}_0}{x : a \vdash x : a} \qquad \frac{\gamma_1 \vdash s_1 : a_1 \dots \gamma_k \vdash s_k : a_k \qquad \sigma \in \mathsf{shu}(\gamma_1, \dots, \gamma_k)}{(\gamma_1, \dots, \gamma_k) \cdot \sigma \vdash \langle s_1, \dots, s_k \rangle : (a_1 \otimes \dots \otimes a_k)}$$

$$\frac{\gamma \vdash s : (a_1 \otimes \dots \otimes a_k) \qquad \delta, x_1 : a_1, \dots, x_k : a_k, \delta' \vdash t : b \qquad \sigma \in \mathsf{shu}(\delta, \gamma', \delta')}{(\delta, \gamma, \delta') \cdot \sigma \vdash t[x_1^{a_1}, \dots, x_k^{a_k} := s] : b}$$

Figure 3 Symmetric Representable Type System on a signature $\mathcal{R}$. We omit the case $f(\vec{s})$.

Proof. By induction on s. In the cases where a merging of type contexts happens, such as the list case, we rely on the properties of shuffle permutations and on the fact that type contexts are repetitions-free. Hence, the action of non-identity permutations on contexts is always fixedpoint-free. ◀

▶ **Proposition 33.** *The following rule is admissible:* $\dfrac{\gamma \vdash s : a \qquad \sigma \in S_k}{\gamma \cdot \sigma \vdash s : a}$.

Proof. Easy induction on the structure of s, exploiting Lemma 2. ◀

The reduction relation is the same as the representable one, that we know to be strongly normalizing and confluent. We also have preservation of typing under reduction and structural equivalence. Given $s \in \Lambda_{\mathsf{reps}}(\mathcal{R})(\gamma; a)$, we denote by $\mathsf{nf}(s)$ its unique normal form. As a corollary of subject reduction, we get that $\mathsf{nf}(s) \in \Lambda_{\mathsf{reps}}(\mathcal{R})(\gamma; a)$.

Free Symmetric Representable Multicategories. We now characterize the free symmetric representable construction. Given a representable signature $\mathcal{R}$, we define a multicategory by setting $\mathrm{ob}(\mathsf{SRM}(\mathcal{R})) = \mathcal{R}_0$ and $\mathsf{SRM}(\mathcal{R})(\gamma; a) = \Lambda_{\mathsf{reps}}(\mathcal{R})(\gamma; a)/(\equiv \cup =_{\mathsf{rep}})$. Composition is given by substitution, identities are given by variables. The operation is well-defined on equivalence classes and satisfies associativity, identity axioms. One can prove that $\mathsf{SRM}(\mathcal{R})$ is representable, by repeating the argument given for Proposition 24. The proof that $\mathsf{SRM}(\mathcal{R})$ is symmetric is a direct corollary of Proposition 33:

▶ **Proposition 34** (Symmetry). *We have that* $\mathcal{M}(\gamma, a_1, \dots, a_k; a) = \mathcal{M}(\gamma; a_{\sigma(1)}, \dots, a_{\sigma(k)}; a)$.

▶ **Example 35.** An interesting example of structural equivalence is the following. Let $s = \langle\rangle[- := x][- := y]$ and $s' = \langle\rangle[- := y][- := x]$, with $s, s' \in \Lambda_{\mathsf{reps}}(\mathcal{R})((), (); ())$. We have that $\langle\rangle[- := x][- := y] \equiv \langle\rangle[- := y][- := x]$, with $x : (), y : () \vdash s : ()$ and $y : (), x : () \vdash s' : ()$. This is the way our syntax validates the fact that permutations of the unit type *collapse* to the identity permutation, since s corresponds to the identity permutation, while s' to the swapping of x with y.

▶ **Definition 36.** *Let $\mathcal{R}$ be a representable signature and S be a symmetric representable multicategory. Let $i : \mathcal{R} \to \overline{\mathsf{S}}$ be a map of representable signatures. We define a family of maps $\mathsf{RT}(i)_{\gamma,a} : \Lambda_{\mathsf{reps}}(\mathcal{R})(\gamma; a) \to \mathsf{S}(i(\gamma); i(a))$ by induction as follows:*

$$\mathsf{RT}(i)_{a,a}(x) = id_{i(a)} \qquad \mathsf{RT}(i)_{(\gamma_1,\dots,\gamma_k)\cdot\sigma,\langle a_1,\dots,a_k\rangle}(\langle s_1, \dots, s_k\rangle) = \left(\bigotimes_{j=1}^{k} \mathsf{RT}(i)_{\gamma_j,a_j}(s_j)\right) \circ \sigma$$

$$\mathsf{RT}(i)_{\delta_1,\gamma,\delta_2\cdot\sigma,a}(s[x_1^{a_1}, \dots, x_k^{a_k} := t]) = ((\mathsf{RT}(i)_{\delta_1,a_1,\dots,a_k,a}(s))^* \circ \langle id_{\delta_1}, \mathsf{RT}(i)_{\gamma,\vec{a}}(t), id_{\delta_2}\rangle) \circ \sigma.$$

▶ **Theorem 37** (Free Construction). *Let* S *be a a symmetric representable multicategory and* $i : \mathcal{R} \to \overline{\mathsf{S}}$ *a map of representable signatures. There exists a unique symmetric representable functor* i^* *such that* $i = \overline{i^*} \circ \eta_{\mathcal{R}}$.

Coherence Result. Fix a discrete signature $\mathcal{R}$. We shall prove that morphisms in $\mathsf{SRM}(\mathcal{R})$ can by characterized by means of appropriate permutations of their typing context. This will lead the following coherence result for symmetric representable multicategories: two morphisms in $\mathsf{SRM}(\mathcal{R})$ are equal whenever their *underlying permutations* are the same.

We start by defining the *strictification* of a representable type $\mathsf{strict}(a)$, by induction as follows: $\mathsf{strict}(o) = o$, $\mathsf{strict}((a_1 \otimes \cdots \otimes a_k)) = \mathsf{strict}(a_1), \ldots, \mathsf{strict}(a_k)$. $\mathsf{strict}(a)$ is the list of atoms that appear in the type a. We extend the strictification to contexts in the natural way. Let $s \in \mathsf{nf}(\Lambda_{\mathsf{reps}}(\mathcal{R}))(\gamma, a)$ and $\sigma \in \mathsf{Stab}(\mathsf{strict}(\gamma))$. We define the *right action* of σ on s, s^σ by induction as follows:

$$x^{id} = x \qquad \langle s_1, \ldots, s_k \rangle^{\sigma \circ (\bigoplus_{i=1}^{k} \sigma_i)} = \langle s_1^{\sigma_1}, \ldots, s_k^{\sigma_k} \rangle \cdot \sigma$$

$$(s[\vec{x}_1 := x_1] \ldots [\vec{x}_n := x_n])^\sigma = (s^\sigma)[\sigma(\vec{x}_1) := x_1] \ldots [\sigma(\vec{x}_n) := x_n]$$

where $\sigma(x_1, \ldots, x_k)$ stands for the image of $x_1, \ldots, x_k$ along the permutation σ.

▶ **Theorem 38.** *Let* $s \in \mathsf{nf}(\Lambda_{\mathsf{reps}}(\mathcal{R}))(\gamma, a)$. *There exists a unique* $\sigma \in \mathsf{Stab}((\mathsf{strict}(\gamma))$ *and a unique non-symmetric representable normal term* t *such that* $s = t^\sigma$.

Proof. By induction on s, exploiting Proposition 32. If $s = x$ the result is immediate. If $s = \langle s_1, \ldots, s_k \rangle$ with $\gamma = (\gamma_1, \ldots, \gamma_k) \cdot \sigma \vdash \langle s_1, \ldots, s_k \rangle : (a_1 \otimes \cdots \otimes a_k)$ and γ being atomic, by IH we have unique $\sigma_1, \ldots, \sigma_k \in \mathsf{St}(\mathsf{strict}(\gamma_i))$ and $t_1, \ldots, t_k \in \mathsf{nf}(\Lambda_{\mathsf{rep}}(A))$ s.t. $s_i = t_i^{\sigma_i}$ for $i \in [k]$. Then, by definition, $s = \langle t_1, \ldots, t_k \rangle^{\sigma \circ (\sigma_1 \otimes \cdots \otimes \sigma_k)}$. Uniqueness derives by Proposition 32. If $s = p[\vec{x}_1 := x_1] \ldots [\vec{x}_n := x_n]$, By IH there exists unique σ and t s.t. $p = t^\sigma$. Then we can conclude by the fact that the action of non-identity permutations on variables is fixedpoint-free. ◀

Let $s \in \mathsf{nf}(\Lambda_{\mathsf{reps}}(\mathcal{R}))(\gamma; a)$. We denote by $\mathsf{sym}(s)$ the unique permutation given by the former theorem. Given $s \in \Lambda_{\mathsf{reps}} A(\gamma; a)$ we set $\mathsf{sym}(s) = \mathsf{sym}(\mathsf{nf}(s))$. This definition is clearly coherent with the quotient on terms performed in the free construction.

▶ **Theorem 39.** *Let* $s, s' \in \mathsf{nf}(\Lambda_{\mathsf{reps}}(\mathcal{R}))(\gamma; a)$. *If* $\mathsf{sym}(s) = \mathsf{sym}(s')$ *then* $s \equiv s'$.

▶ **Theorem 40** (Coherence). *Let* $[s], [s'] \in \mathsf{SRM}(\mathcal{A})(\gamma; a)$. *If* $\mathsf{sym}([s]) = \mathsf{sym}[s']$ *then* $[s] = [s']$.

▶ **Theorem 41** (Coherence for Symmetric Monoidal Categories). *Two morphisms in the free symmetric monoidal categories are equal if their underlying permutations are equal.*

Proof. Corollary of Theorems 11 and 40. ◀

5 A Resource Calculus for Symmetric Closed Multicategories

We consider the case of symmetric closed multicategories, which is orthogonal to the representable structures we introduced in the previous sections. This calculus corresponds to the resource version of linear λ-calculus, where we have unbiased k-ary λ-abstraction and (linear) application. We begin by defining the terms and their typings, then proceed to introducing their operational semantics. We conclude by characterizing the free construction *via* well-typed equivalence classes of terms.

$$\frac{a \in \mathcal{L}_0}{x : a \vdash x : a} \qquad \frac{\gamma, x_1 : a_1, \dots, x_k : a_k \vdash s : b}{\gamma \vdash \lambda\langle x_1^{a_1}, \dots, x_k^{a_k}\rangle.s : (a_1 \otimes \cdots \otimes a_k) \multimap b}$$

$$\frac{\gamma_0 \vdash s : (a_1 \otimes \cdots \otimes a_k) \multimap b \qquad \gamma_1 \vdash t_1 : a_1 \dots \gamma_k \vdash t_k : a_k \qquad \sigma \in \mathsf{shu}(\gamma_0, \dots, \gamma_k)}{(\gamma, \delta) \cdot \sigma \vdash s\langle t_1, \dots, t_k\rangle : b}$$

$$\mathtt{C} ::= \mathtt{C} ::= [\cdot] \mid s\langle s_1, \dots, \mathtt{C}, \dots, s_k\rangle \mid \mathtt{C}\langle s_1, \dots, s_k\rangle \mid \lambda\langle x_1, \dots, x_k\rangle.\mathtt{C} \mid f(s_1, \dots, \mathtt{C}, \dots, s_k).$$
$$\mathtt{E} ::= [\cdot] \mid s\langle s_1, \dots, \mathtt{E}, \dots, s_k\rangle \mid \mathtt{E}\langle s_1, \dots, s_k\rangle \quad (\mathtt{E} \neq [\cdot]) \mid \lambda\langle x_1, \dots, x_k\rangle.\mathtt{E} \mid f(s_1, \dots, \mathtt{E}, \dots, s_k).$$

Figure 4 Symmetric closed type system on a signature $\mathcal{L}$ and contexts with one hole. Types are the elements of $\mathcal{L}_0$. We omit the case of $f(\vec{s})$.

β *Root-Step*: $(\lambda\langle x_1^{a_1}, \dots, x_k^{a_k}\rangle.s)\langle t_1, \dots, t_k\rangle \to_\beta s\{t_1, \dots, t_k/x_1, \dots, x_k\}$.

η *Root-Step*: $s \to_\eta \lambda\vec{x}^{\vec{a}}.(s\vec{x})$ where $\vec{x}$ fresh $, \gamma \vdash s : \vec{a} \multimap a, s \notin \mathsf{AT}$.

Contextual extensions: $\dfrac{s \to_\beta s'}{\mathtt{C}[s] \to_\beta \mathtt{C}[s']} \qquad \dfrac{s \to_\eta s'}{\mathtt{E}[s] \to_\eta \mathtt{E}[s']} \quad (\to_{\mathsf{sc}} = \to_\beta \cup \to_\eta).$

Figure 5 Symmetric closed reduction relations.

Symmetric Closed Resource Terms. Let $\mathcal{L}$ be a closed signature. The *symmetric closed resource terms* on $\mathcal{L}$ are defined by the following inductive grammar:

$$\Lambda_{\mathsf{sc}}(\mathcal{L}) \ni s ::= x \in \mathcal{V} \mid \lambda\langle x_1^{a_1}, \dots, x_k^{a_k}\rangle.s \mid s\langle s_1, \dots, s_k\rangle \mid f(s_1, \dots, s_k)$$

for $k \in \mathbb{N}$ and $f \in \mathsf{arr}(\mathcal{L}), a_i \in \mathcal{L}$. A term of the shape $s\langle s_1, \dots, s_k\rangle$ is called a *(k-linear) application*. A term of the shape $\lambda\langle x_1, \dots, x_k\rangle.s$ is called a *(k-linear) λ-abstraction*. Variables under the scope of a λ-abstraction are bound. We define the following subset of terms $\mathsf{AT} = \{\mathtt{L}[\lambda\vec{x}.t] \mid \text{ for some substitution context } \mathtt{L} \text{ and term } t.\}$. Typing and contexts with hole are defined in Figure 4. Given a term $\gamma \vdash s : a$, there exists a unique type derivation for it.

Terms under Reduction. The reduction relation is defined in Figure 5.

▶ Remark 42. The definition of the β-reduction follows the standard choices for resource calculi. The novel technicality is the restriction of the η-reduction, that is justified again by the goal of obtaining a strongly normalizing reduction. Indeed, η-reduction is again not normalizing. The situation recalls what happens in the standard λ-calculus and we deal with it adapting to our framework the restrictions introduced in [28, 16].

To study the rewriting, we adapt the method introduced for representable terms. We first prove that typing is preserved under reduction. Then, we introduce a measure that decreases under η. We define the *size* of a type by induction: $\mathtt{size}(o) = 0, \mathtt{size}(\langle a_1, \dots, a_k\rangle \multimap a) = 1 + \sum \mathtt{size}(a_i) + \mathtt{size}(a)$. Given $\gamma \vdash s : a$ we define a set of typed subterms of s: $\mathsf{EST}(s) = \{\delta \vdash p : a \mid p \in \mathsf{ST}(s) \setminus \mathsf{AT} \text{ s.t. } \mathtt{E}[\delta \vdash p : a] = s \text{ for some context } \mathtt{E}\}$. We set $\eta(s) = \sum_{\delta \vdash p : a \in \mathsf{EST}(s)} \mathtt{size}(a)$. The proof of strong normalization and confluence is completely

symmetrical to the representable case. Given $s \in \Lambda_{\mathsf{sc}}(\mathcal{R})(\gamma; a)$, we denote by $\mathsf{nf}(s)$ its unique normal form. As a corollary of subject reduction, we get that $\mathsf{nf}(s) \in \Lambda_{\mathsf{sc}}(\mathcal{R})(\gamma; a)$.

Free Symmetric Closed Multicategories. Let $\mathcal{L}$ be a closed signature, we define a multicategory $\mathsf{SCM}(\mathcal{L})$ by setting $\mathrm{ob}(\mathsf{SCM}(\mathcal{L})) = \mathcal{L}_0$ and $\mathsf{SCM}(\mathcal{L})(\gamma; a) = \Lambda_{\mathsf{sc}}(\mathcal{L})(\gamma; a)/\sim$ where $\sim = =_{\mathsf{sc}}$. Composition is given by substitution, identities are given by variables. The operation is well-defined equivalence classes and satisfies associativity and identity axioms. We also have that if $s \sim s'$, then $\mathsf{nf}(s) = \mathsf{nf}(s')$. We denote by $\eta_{\mathcal{L}} : \mathcal{L} \to \overline{\mathsf{SCM}(\mathcal{L})}$ the evident inclusion. One can prove that $\mathsf{SCM}(\mathcal{R})$ is symmetric, by repeating the argument given in the previous section. This multicategory is also *closed*:

▶ **Theorem 43.** *We have a bijection* $\mathsf{SCM}(\mathcal{L})(\gamma; \langle a_1, \dots, a_k\rangle \multimap a) \cong \mathsf{SCM}(\mathcal{L})(\gamma, a_1, \dots, a_k; a)$ *natural in* a *and multinatural in* γ*, induced by the maps* $[s] \mapsto [s\langle x_1, \dots, x_k\rangle]$.

Proof. Naturality derives from basic properties of substitution. Inverses are given by the maps $[s] \mapsto [\lambda\langle x_1, \dots, x_k\rangle.s]$. ◀

▶ **Definition 44.** *Let* E *be a symmetric closed multicategory and let* $i : \mathcal{L} \to \overline{\mathsf{E}}$ *be a map of closed signatures. We define a family of maps* $\mathsf{RT}_{\gamma,a} : \Lambda_{\mathsf{sc}}(\mathcal{L})(\gamma, a) \to \mathsf{E}(i(\gamma), i(a))$ *by induction as follows:*

$$\mathsf{RT}_{a,a}(x) = 1_{i(a)} \qquad \mathsf{RT}_{\gamma,\vec{a}\multimap a}(\lambda\vec{x}.s) = \lambda(\mathsf{RT}_{\gamma,\vec{a},a}(s))$$

$$\mathsf{RT}_{(\gamma_0,\dots,\gamma_k),a}(s\langle t_1, \dots, t_k\rangle) = (ev \circ \langle \mathsf{RT}_{\gamma_0,\langle a_1,\dots,a_k\rangle\multimap a}(s), \mathsf{RT}_{\gamma_1,a_1}(t_1), \dots, \mathsf{RT}_{\gamma_k,a_1}(t_k)\rangle) \cdot \sigma.$$

▶ **Theorem 45** (Free Construction)**.** *Let* S *be a a symmetric closed multicategory and* $i : \mathcal{L} \to \overline{\mathsf{S}}$ *a map of representable signatures. There exists a unique symmetric closed functor* $i^* : \mathsf{SCM}(\mathcal{L}) \to \mathsf{S}$ *such that* $\overline{i^*} \circ \eta_{\mathcal{L}} = i$.

▶ **Theorem 46** (Coherence)**.** *Let* $[s], [s'] \in \mathsf{SCM}(\mathcal{R})(\gamma; a)$*. Then* $[s] = [s']$ *iff* $\mathsf{nf}([s]) \equiv \mathsf{nf}([s'])$.

6 A Resource Calculus for Autonomous Multicategories

In this section we present our calculus for autonomous multicategories. These structures bring together representability, symmetry and closure. For this reason, the calculus we will present is a proper extension of the ones we introduced before. Again, we follow the same pattern of Sections 3 and 5, first introducing the typing, then studying the operational semantics and finally characterizing the free constructions.

Autonomous Terms. Let $\mathcal{A}$ be an autonomous signature. The *autonomous resource terms* on $\mathcal{A}$ are defined by the following inductive grammar:

$$\Lambda_{\mathsf{aut}}(\mathcal{A}) \ni s, t ::= x \mid \lambda\langle x_1^{a_1}, \dots, x_k^{a_k}\rangle.s \mid st \mid \langle s_1, \dots, s_k\rangle \mid s[x_1^{a_1}, \dots, x_k^{a_k} := t] \mid f(s_1, \dots, s_k)$$

for $k \in \mathbb{N}$ and $f \in \mathsf{arr}(\mathcal{A}), a_i \in \mathcal{A}$. Variables under the scope of a λ-abstraction and of a substitution are bound. The typing is given in Figure 6. The calculi introduced in the previous sections can be seen as subsystems of the autonomous one.

Given a subterm p of s we write $\mathsf{ty}(p)_s$ for the type of p in the type derivation of s. The mapping is functional as corollary of the former proposition.

$$\frac{a \in \mathcal{A}_0}{x : a \vdash x : a} \qquad \frac{\gamma_1 \vdash s_1 : a_1 \dots \gamma_k \vdash s_k : a_k \qquad \sigma \in \mathsf{shu}(\gamma_1, \dots, \gamma_k)}{(\gamma_1, \dots, \gamma_k) \cdot \sigma \vdash \langle s_1, \dots, s_k \rangle : (a_1 \otimes \cdots \otimes a_k)}$$

$$\frac{\gamma, x_1 : a_1, \dots, x_k : a_k \vdash s : b}{\gamma \vdash \lambda\langle x_1^{a_1}, \dots, x_k^{a_k} \rangle . s : (a_1 \otimes \cdots \otimes a_k) \multimap b} \qquad \frac{\gamma \vdash s : \vec{a} \multimap b \qquad \delta \vdash t : \vec{a} \qquad \sigma \in \mathsf{shu}(\gamma, \delta)}{(\gamma, \delta) \cdot \sigma \vdash st : b}$$

$$\frac{\gamma \vdash s : (a_1 \otimes \cdots \otimes a_k) \qquad \delta_1, x_1 : a_1, \dots, x_k : a_k, \delta_2 \vdash t : b \qquad \sigma \in \mathsf{shu}(\gamma, \delta_1, \delta_2)}{(\gamma, \delta_1, \delta_2) \cdot \sigma \vdash t[x_1^{a_1}, \dots, x_k^{a_k} := s] : b}$$

Figure 6 Autonomous type system on a signature $\mathcal{A}$. We omit the case of $f(\vec{s})$.

Terms under Reduction. The reduction relation $\to_{\mathsf{aut}}$, together with its subreductions β and η are defined by putting together the reductions $\to_{\mathsf{rep}}$ (Figure 2) and $\to_{\mathsf{sc}}$ (Figure 5). The same happens with structural equivalence. The reduction satisfies subject reduction, strong normalization and confluence. The proofs build on the results of the previous sections. As decreasing measures, we use the size of a term for β-reduction and the sum of the two η measures we defined in the previous sections for η-reduction.

Free Autonomous Multicategories. Let $\mathcal{A}$ be an autonomous signature, we define a multicategory $\mathsf{AUT}(\mathcal{A})$ by setting $\mathrm{ob}(\mathsf{AUT}(\mathcal{A})) = \mathcal{A}_0$ and $\mathsf{AUT}(\mathcal{A})(\gamma; a) = \Lambda_{\mathsf{aut}}(\mathcal{A})(\gamma; a)/\!\sim$ where $\sim$ is the equivalence $\equiv \cup =_{\mathsf{aut}}$. Composition is given by substitution, identities are given by variables. The operation is well-defined on equivalence classes and satisfies associativity and identity axioms. We also have that if $s \sim s'$ then $\mathsf{nf}(s) \equiv \mathsf{nf}(s')$. We denote by $\eta_{\mathcal{A}} : \mathcal{A} \to \overline{\mathsf{AUT}(\mathcal{A})}$ the evident inclusion. One can prove that this multicategory is symmetric, representable and closed by importing the proofs given in the previous sections.

▶ **Definition 47.** *Let* S *be an autonomous multicategory and let* $i : \mathcal{A} \to \overline{\mathsf{S}}$ *be a map of autonomous signatures. We define a family of maps* $\mathsf{RT}_{\gamma,a} : \Lambda_{\mathsf{aut}}(\mathcal{A})(\gamma, a) \to \mathsf{E}(i(\gamma), i(a))$ *by induction, extending Definitions 36 and 44 in the natural way.*

▶ **Theorem 48** (Free Construction)**.** *Let* S *be a an autonomous multicategory and* $i : \mathcal{A} \to \overline{\mathsf{S}}$ *a map of autonomous signatures. There exists a unique autonomous functor* $i^* : \mathsf{AUT}(\mathcal{A}) \to \mathsf{S}$ *such that* $\overline{i^*} \circ \eta_{\mathcal{A}} = i$.

▶ **Theorem 49** (Coherence)**.** *Let* $[s], [s'] \in \mathsf{AUT}(\mathcal{R})(\gamma; a)$. *Then* $[s] = [s']$ *iff* $\mathsf{nf}([s]) \equiv \mathsf{nf}([s'])$.

7 Conclusion

We established a formal correspondence between resource calculi and appropriate linear multicategories, providing coherence theorems by means of normalization. As future work, we consider two possible perspectives. It is tempting to parameterize our construction over the choice of allowed *structural rules* on typing contexts. For instance, while the choice of permutations (*i.e.*, symmetries) gives *linear* structures, the choice of arbitrary functions between indexes would give *cartesian* structures. In this way, we would achieve a general method to produce type theories for appropriate *algebraic theories*, in the sense of [15]. For this, the the perspective on multicategories of [32] will be a starting point. Another perspective is the passage to the *second dimension*, following the path of [10]. In this way, the rewriting of terms would become visible in the multicategorical structure itself. Coherence by normalization could then be upgraded to a method of *coherence by standardization*, exploiting a rewriting relation on reduction paths.

References

1 Beniamino Accattoli. Exponentials as substitutions and the cost of cut elimination in linear logic. In Christel Baier and Dana Fisman, editors, *LICS '22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2–5, 2022*, pages 49:1–49:15. ACM, 2022. doi:10.1145/3531130.3532445.

2 Beniamino Accattoli, Eduardo Bonelli, Delia Kesner, and Carlos Lombardi. A nonstandard standardization theorem. In Suresh Jagannathan and Peter Sewell, editors, *The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14, San Diego, CA, USA, January 20-21, 2014*, pages 659–670. ACM, 2014. doi:10.1145/2535838.2535886.

3 Davide Barbarossa and Giulio Manzonetto. Taylor subsumes scott, berry, kahn and plotkin. *Proc. ACM Program. Lang.*, 4(POPL):1:1–1:23, 2020. doi:10.1145/3371069.

4 Nick Benton, Gavin Bierman, Valeria de Paiva, and Martin Hyland. A term calculus for intuitionistic linear logic. In *Proceedings of the International Conference on Typed Lambda Calculi and Applications (TLCA)*. Springer, January 1993.

5 R.F. Blute, J.R.B. Cockett, R.A.G. Seely, and T.H. Trimble. Natural deduction and coherence for weakly distributive categories. *Journal of Pure and Applied Algebra*, 113(3):229–296, 1996. doi:10.1016/0022-4049(95)00159-X.

6 Gérard Boudol. The lambda-calculus with multiplicities. In Eike Best, editor, *CONCUR'93*, pages 1–6, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

7 Pierre-Louis Curien, Richard Garner, and Martin Hofmann. Revisiting the categorical interpretation of dependent type theory. *Theoretical Computer Science*, 546:99–119, 2014. Models of Interaction: Essays in Honour of Glynn Winskel. doi:10.1016/j.tcs.2014.03.003.

8 Roberto Di Cosmo and Delia Kesner. Combining algebraic rewriting, extensional lambda calculi, and fixpoints. *Theoretical Computer Science*, 169(2):201–220, 1996. doi:10.1016/S0304-3975(96)00121-1.

9 Thomas Ehrhard and Laurent Regnier. Uniformity and the Taylor expansion of ordinary λ-terms. *Theoretical Computer Science*, 403(2-3), 2008. doi:10.1016/j.tcs.2008.06.001.

10 Marcelo Fiore and Philip Saville. Coherence and normalisation-by-evaluation for bicategorical cartesian closed structure. In Holger Hermanns, Lijun Zhang, Naoki Kobayashi, and Dale Miller, editors, *LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020*, pages 425–439. ACM, 2020. doi:10.1145/3373718.3394769.

11 Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–101, 1987. doi:10.1016/0304-3975(87)90045-4.

12 Claudio Hermida. Representable multicategories. *Advances in Mathematics*, 151(2):164–225, 2000. doi:10.1006/aima.1999.1877.

13 Dominic J.D. Hughes. Simple free star-autonomous categories and full coherence. *Journal of Pure and Applied Algebra*, 216(11):2386–2410, 2012. doi:10.1016/j.jpaa.2012.03.020.

14 J. M. E. Hyland. Classical lambda calculus in modern dress. *Mathematical Structures in Computer Science*, 27(5):762–781, 2017. doi:10.1017/S0960129515000377.

15 J.M.E. Hyland. Elements of a theory of algebraic theories. *Theoretical Computer Science*, 546:132–144, 2014. Models of Interaction: Essays in Honour of Glynn Winskel. doi:10.1016/j.tcs.2014.03.005.

16 C. Barry Jay and Neil Ghani. The virtues of eta-expansion. *Journal of Functional Programming*, 5(2):135–154, 1995. doi:10.1017/S0956796800001301.

17 G.M. Kelly and S. Maclane. Coherence in closed categories. *Journal of Pure and Applied Algebra*, 1(1):97–140, 1971. doi:10.1016/0022-4049(71)90013-2.

18 Delia Kesner. The theory of calculi with explicit substitutions revisited. In Jacques Duparc and Thomas A. Henzinger, editors, *Computer Science Logic*, pages 238–252, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

19 AJ Kfoury. A linearization of the lambda-calculus and consequences. *Journal of Logic and Computation*, 10(3):411–436, 2000. doi:10.1093/logcom/10.3.411.

20 Yves Lafont. *Logiques, Categories & Machines: Implantation de Langages de Programmation guidée par la Logique Catégorique*. Phd thesis, Université Paris VII, 1988.

21 Joachim Lambek. Deductive systems and categories ii. standard constructions and closed categories. In Peter J. Hilton, editor, *Category Theory, Homology Theory and their Applications I*, pages 76–122, Berlin, Heidelberg, 1969. Springer Berlin Heidelberg.

22 Joachim Lambek and Philip J. Scott. *Introduction to Higher Order Categorical Logic*. Cambridge University Press, USA, 1986.

23 Tom Leinster. Higher operads, higher categories, 2003. arXiv:math/0305049.

24 Saunders Maclane. Natural associativity and commutativity. *Rice Institute Pamphlet – Rice University Studies*, 49:28–46, 1963.

25 Damiano Mazza. *Polyadic Approximations in Logic and Computation. Habilitation* thesis, Université Paris 13, 2017.

26 Damiano Mazza, Luc Pellissier, and Pierre Vial. Polyadic approximations, fibrations and intersection types. *PACMPL*, 2018. doi:10.1145/3158094.

27 Paul-André Melliès, Federico Olimpieri, and Lionel Vaux Auclair. An explicit construction of the homotopy span model of differential linear logic.

28 G. E. Mints. Closed categories and the theory of proofs. *Journal of Soviet Mathematics*, 1981. doi:10.1007/BF01404107.

29 Yo Ohta and Masahito Hasegawa. A terminating and confluent linear lambda calculus. In Frank Pfenning, editor, *Term Rewriting and Applications, 17th International Conference, RTA 2006, Seattle, WA, USA, August 12-14, 2006, Proceedings*, volume 4098 of *Lecture Notes in Computer Science*, pages 166–180. Springer, 2006. doi:10.1007/11805618_13.

30 Federico Olimpieri. *Intersection Types and Resource Calculi in the Denotational Semantics of Lambda-Calculus*. PhD thesis, Aix-Marseille Université, 2020.

31 P. Selinger. *A Survey of Graphical Languages for Monoidal Categories*, pages 289–355. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. doi:10.1007/978-3-642-12821-9_4.

32 Michael Shulman. Categorical logic from a categorical point of view. Draft for AARMS Summer School 2016, 2016. URL: https://mikeshulman.github.io/catlog/catlog.pdf.

33 Michael Shulman. A practical type theory for symmetric monoidal categories. *Theory and Applications of Categories*, 37(25):863–907, 2021.

34 Takeshi Tsukada, Kazuyuki Asada, and C.-H. Luke Ong. Generalised species of rigid resource terms. In *Proceedings of the 32rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS 2017, 2017. doi:10.1109/LICS.2017.8005093.

35 Lionel Vaux. Taylor expansion, β-reduction and normalization. In *Ccomputer Science Logic 2017*, 2017. doi:10.4230/LIPIcs.CSL.2017.39.

36 Niccolò Veltri. Coherence via focusing for symmetric skew monoidal categories. In Alexandra Silva, Renata Wassermann, and Ruy J. G. B. de Queiroz, editors, *Logic, Language, Information, and Computation – 27th International Workshop, WoLLIC 2021, Virtual Event, October 5-8, 2021, Proceedings*, volume 13038 of *Lecture Notes in Computer Science*, pages 184–200. Springer, 2021. doi:10.1007/978-3-030-88853-4_12.

Conservativity of Type Theory over Higher-Order Arithmetic

Daniël Otten
ILLC, University of Amsterdam, The Netherlands

Benno van den Berg
ILLC, University of Amsterdam, The Netherlands

Abstract

We investigate how much type theory can prove about the natural numbers. A classical result in this area shows that dependent type theory without any universes is conservative over Heyting Arithmetic (HA). We build on this result by showing that type theories with one level of impredicative universes are conservative over Higher-order Heyting Arithmetic (HAH). This result clearly depends on the specific type theory in question, however, we show that the interpretation of logic also plays a major role. For proof-irrelevant interpretations, we will see that strong versions of type theory prove exactly the same higher-order arithmetical formulas as HAH. Conversely, for proof-relevant interpretations, they prove different second-order arithmetical formulas than HAH, while still proving exactly the same first-order arithmetical formulas. Along the way, we investigate the various interpretations of logic in type theory, and to what extent dependent type theories can be seen as extensions of higher-order logic. We apply our results by proving a De Jongh's theorem for type theory.

2012 ACM Subject Classification Theory of computation → Type theory; Theory of computation → Higher order logic; Theory of computation → Constructive mathematics

Keywords and phrases Conservativity, Arithmetic, Realizability, Calculus of Inductive Constructions

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.44

Related Version *The master's thesis of the first author, supervised by the second author, which proves results for second-order arithmetic*: `https://eprints.illc.uva.nl/id/document/12640` [31]

Funding This publication is part of the project "The Power of Equality" (with project number OCENW.M20.380) of the research programme Open Competition Science M 2 which is (partly) financed by the Dutch Research Council (NWO).

1 Introduction

When studying a theory, we obtain a lot of information by determining the arithmetical statements that it can prove. This data decides its consistency strength, which functions it can prove to be recursive, and which other theories it can prove to be consistent. In this work, we determine this for dependent type theories that have a single level of universes. We are interested in the general picture: in predicative and impredicative versions of type theory, intensional and extensional versions, and a wide array of type constructors. We obtain our results by studying a strong version of type theory and deducing results for weaker theories. Besides the theory itself, we consider proof-irrelevant and proof-relevant interpretations of logic, which we think of as black box (•) and white box (∘) interpretations, respectively.

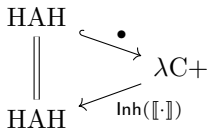
Main Results. Strong versions of type theory with a single level of universes prove:

- the same higher-order arithmetical formulas as HAH for proof-irrelevant interpretations,
- ∘ the same first-order arithmetical formulas as HAH for proof-relevant interpretations.

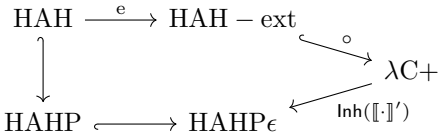
Moreover, the proof-relevant result is maximal: type theories differ from HAH on second-order arithmetical formulas. More precisely, for the proof-relevant interpretation: type theory proves the axiom of choice which is not assumed in HAH while type theory does not prove extensionality of sets which is assumed in HAH. The main type theory that we consider is a version of the Calculus of Inductive Constructions (λC+), specified in Appendix A.

Proof Sketch. We prove the two results as follows:

- The proof-irrelevant result is relatively straightforward. We build a model for λC+ based on subsingletons, partial equivalence relations (PERs), and assemblies. The innovation is that we only use notions that can be defined in HAH. So, for every type A, we get a formula $\mathsf{Inh}([\![A]\!])$ stating that the type is inhabited in the model. Conservativity follows by showing that the diagram commutes up to logical equivalence.



- The proof-relevant result is more involved and is our main contribution. As $\circ$ only interprets HAH $-$ ext, we first interpret HAH in HAH $-$ ext. Now, we extend the model using a choice principle. For this, we conservatively extend HAH: first with primitive notions for partial recursive functions (HAHP), and then with Hilbert-style epsilon-constants, which can be seen as partial choice functions (HAHPϵ). To formulate these extensions, we define a higher-order version of Beeson's logic of partial terms. Conservativity follows by showing for first-order formulas that e behaves as the identity and that the diagram commutes up to logical equivalence.



◀

De Jongh's Theorem. The main application of our results is a proof of De Jongh's theorem for type theories. This theorem says that the only propositional formulas that hold in the theory are those of intuitionistic logic. Here we say that a formula holds when any closed instance of it is provable. De Jongh has shown this for Heyting arithmetic [12], and with Smorynski for second-order Heyting arithmetic [13]. This is not the case for any intuitionistic theory: extensionality, specification, and choice famously imply the law of the excluded middle [15], but the axioms of ZF already cause more formulas to hold [17]. Recently, Robert Passmann has shown that De Jongh's theorem does hold for CZF and IZF (which are equivalent to ZF classically but not intuitionistically) [32, 33]. Our goal was to prove this for type theory despite the proof-relevant interpretation satisfying choice.

Related Work.

- We give a higher-order version of the following classical theorem: type theory without universes is conservative over HA. For a proof, see Beeson [3, Chapter XIII, Theorem 7.5.1]. Note that universes are needed to show $0 \neq 1$ in type theory [37], so, without universes, this should be added as an axiom.
- A related question was independently answered by Berardi [4] and Geuvers [18]. They show that the calculus of constructions is not conservative over higher-order logic. This result hinges on the fact that domains of quantification and propositions are not distinguished in the calculus of constructions. We are in a different setting: we only consider arithmetical formulas, where the only domains are iterated powersets of the natural numbers.
- A history of De Jongh's theorems is given by De Jongh, Verbrugge, and Visser [14].

Structure of the paper. In Section 2 and Section 3 we introduce our arithmetic and type theory respectively. The type theory is motivated in Section 4 where we consider the various interpretations of higher-order logic in type theory. In Section 5, we see the other direction: we build our model of type theory and its interpretation in arithmetic. This is used in Section 6 and Section 7, where we prove proof-irrelevant and proof-relevant conservativity, respectively. De Jongh's theorem is covered in Section 8. Section 9 is the conclusion.

2 Higher-order Arithmetic

We start by stating the various theories of natural numbers. Although they share the same language and axioms, these theories differ in their underlying logic. First, we explain and motivate these logics. Then, we will introduce the various theories of arithmetic.

2.1 Higher-order Logic

Motivation. There are many versions of higher-order logic, and the notation is not standardised. So, before giving a formal introduction, we will first motivate our choices. Our version of higher-order logic quantifies over relations; this directly generalises n-th-order logic. There are also versions that quantify over functions. We observe that in intuitionistic logic, quantifying over relations is more expressive than quantifying over functions:

- We can encode an n-ary function f as an $(n+1)$-ary relation R satisfying $\forall \vec{x}\, \exists! y\, R(\vec{x}, y)$, see for instance [39, Section 2.7]. This means that we can replace $\exists f\ \dots$ and $\forall f\ \dots$ with $\exists R\, (\forall \vec{x}\, \exists! y\, R(\vec{x}, y) \wedge \dots)$ and $\forall R\, (\forall \vec{x}\, \exists! y\, R(\vec{x}, y) \to \dots)$.
- In classical logic, if we have terms $a \neq b$, then it is also possible to encode an n-ary relation R as an n-ary function f satisfying $\forall \vec{x}\, (f(\vec{x}) = a \vee f(\vec{x}) = b)$. However, in intuitionistic logic, this only encodes those R that satisfy $\forall \vec{x}\, (R(\vec{x}) \vee \neg R(\vec{x}))$.

So, for simplicity, we present higher-order logic using only quantifiers over relations. Our second observation is that it is often enough to consider only unary relations:

- If the theory can encode tuples, then, instead of quantifying over an n-ary relation R, we can quantify over a unary relation X: we replace $R(x_0, \dots\ x_{n-1})$ with $X(\langle x_0, \dots, x_{n-1}\rangle)$.

Arithmetic can encode tuples [8, 25, 7, 21]; we can define for instance: $\langle\rangle \coloneqq 0$, $\langle a_0\rangle \coloneqq a_0$, $\langle a_0, a_1\rangle \coloneqq ((a_0 + a_1) \times \mathsf{S}(a_0 + a_1))/2 + a_0$, and $\langle a_0, \dots, a_{n-1}\rangle \coloneqq \langle\langle a_0, \dots, a_{n-2}\rangle, a_{n-1}\rangle$. So, for our purposes, it is sufficient to quantify over subsets (unary relations). Restricting to the unary case gives us "monadic" versions of logic.

▶ **Definition 1.** *A* monadic higher-order logic *is a many-sorted logic with a sort for every numeral $n = 0, 1, \dots$. If we write a^n, then the term a is of the n-th sort, intuitively a member of the n-th power set of the domain. Terms are built using function symbols f, which each have a signature $n_0 \times \dots \times n_{k-1} \to m$. We also allow relation symbols, which each have a signature $n_0 \times \dots \times n_{k-1}$. We always assume that the language has relation symbols $=^n\colon n \times n$ and $\in^n\colon n \times (n+1)$ for every sort n. Formulas are given by:*

$$A, B, \dots \ ::= \ R(a_0^{n_0}, \dots, a_{k-1}^{n_{k-1}}) \mid \bot \mid \top \mid A \vee B \mid A \wedge B \mid A \to B \mid \exists x^n\, B[x^n] \mid \forall x^n\, B[x^n].$$

There are classical and intuitionistic versions of higher-order logic. For both we take standard inference rules for propositional logic, quantifiers, and equality. In addition, we have two axiom schemes for the element relation; for any n and any formula $P[z^n]$ we assume:

$$\exists X^{n+1}\, \forall z^n\, (z \in X \leftrightarrow P[z]), \qquad \text{(specification)}$$

$$\forall X^{n+1}\, \forall Y^{n+1}\, (\forall z^n\, (z \in X \leftrightarrow z \in Y) \to X = Y). \qquad \text{(extensionality)}$$

To reduce clutter, we started omitting the sorts in places where they can easily be inferred.

▶ **Definition 2.** *We define* monadic n-th-order logic *as the restriction of monadic higher-order logic to the first n sorts:* $0, \dots, n-1$.

Defining Logical Connectives. We can also formulate second or higher-order logic in a more minimalistic way, using only $\in$, $\to$, and $\forall$. This is because, by quantifying over a proposition Z (a nullary relation), we can define the other logical connectives:

$$\begin{aligned}
\bot &\coloneqq \forall Z\, Z, && \text{(false)}\\
\top &\coloneqq \forall Z\, (Z \to Z), && \text{(true)}\\
A \vee B &\coloneqq \forall Z\, ((A \to Z) \to (B \to Z) \to Z), && \text{(disjunction)}\\
A \wedge B &\coloneqq \forall Z\, ((A \to (B \to Z)) \to Z), && \text{(conjunction)}\\
\exists x^n\, B[x] &\coloneqq \forall Z\, (\forall x^n\, (B[x] \to Z) \to Z). && \text{(existential quantifier)}
\end{aligned}$$

We can use similar definitions in monadic versions of logic by filling in an arbitrary variable x^0. For example, we could define $\bot$ as $\forall X^1\, (x \in X)$ and $\top$ as $\forall X^1\, (x \in X \to x \in X)$. Similarly, we can define equality using Leibniz's principle:

$$(a =^n b) \coloneqq \forall X^{n+1}\, (a \in X \to b \in X). \qquad \text{(equality)}$$

It is a good exercise to show that these formulas indeed satisfy the correct inference rules. This alternative definition will allow us to simplify our proof for proof-irrelevant conservativity.

2.2 Arithmetic

Language and Axioms. The language of our arithmetical theories consists of a zero constant $0 : 0$ (a nullary function symbol), a successor function $\mathsf{S} : 0 \to 0$, addition $+ : 0 \times 0 \to 0$, and multiplication $\times : 0 \times 0 \to 0$. We have axioms stating that 0 and S are jointly injective:

$$\forall y\, (\mathsf{S}(y) \neq 0), \qquad \forall x\, \forall y\, (\mathsf{S}(x) = \mathsf{S}(y) \to x = y).$$

In addition, we have axioms for addition and multiplication:

$$\begin{aligned}
&\forall y\, (0 + y = y), && \forall x\, \forall y\, (\mathsf{S}(x) + y = \mathsf{S}(x + y)),\\
&\forall y\, (0 \times y = 0), && \forall x\, \forall y\, (\mathsf{S}(x) \times y = (x \times y) + y).
\end{aligned}$$

And we have an axiom scheme for induction; for every formula $A[x]$ we have the axiom:

$$A[0] \wedge \forall x (A[x] \to A[\mathsf{S}(x)]) \to \forall x\, A[x].$$

▶ **Definition 3** (Peano and Heyting arithmetic)**.** *We define the following theories, all with the language and axioms above, and each with different logical inference rules:*

	classical	*intuitionistic*
n-th-order	PAn	HAn
higher-order	PAH	HAH

For PA1 and HA1 we will omit the 1. Note that HAH is stronger than HA^ω (arithmetic in finite types), which quantifies over functions instead of relations [38].

3 Type Theory

We formulate a strong version of type theory that allows all our interpretations of higher-order arithmetic. This theory is impredicative, extensional, and includes inductive types. By proving conservativity for this strong version, we also obtain conservativity results for weaker versions: most notably for more predicative, and intensional versions of type theory. Many of the choices are motivated and explained further in the next section where we discuss these interpretations. So, we will only give a brief overview and cover the details in Appendix A.

Our type theory, which we call λC+, can be seen as a version of the Calculus of Inductive Constructions [11, 5, 34] with only one level of universes. We assume an array of type constructors: $\nvdash, \nVdash, \nvDash, \ldots, \mathbb{N}, \Sigma, \Pi, \mathrm{W}, =, \|\cdot\|$, and quotient types. In addition, we assume two type universes: $\mathsf{Prop}, \mathsf{Set} : \mathsf{Type}$. The universe Prop is used to interpret propositions while Set is used to interpret data types. Prop and Set are impredicative which means that they are closed under products over arbitrary types. So, if we have any type A, and for $x : A$ a type $B[x] : \mathsf{Prop}$, then we always have $\Pi(x : A)\, B[x] : \mathsf{Prop}$, and the same for Set. This allows for self-referential definitions that define a type in the universe by quantifying over all types in the universe; for example, the empty type $\Pi(X : \mathsf{Prop})\, X$ is a term of Prop. Note that the universes are both types themselves, so we can use them to construct new types like $\mathsf{Prop} \times \mathsf{Set}$ and $\mathbb{N} \to \mathsf{Prop}$. Both universes are at the same level, that is, we do not have $\mathsf{Prop} : \mathsf{Set}$ or $\mathsf{Set} : \mathsf{Prop}$. However, we do assume that Prop is a subuniverse of Set, so $A : \mathsf{Prop}$ implies $A : \mathsf{Set}$, and we have that $A : \mathsf{Set}$ implies $A : \mathsf{Type}$. We assume that the theory is extensional, so definitional equality and propositional equality coincide, which implies that we have function extensionality and uniqueness of identity proofs [22]. As we will explain in the next section: we assume that Prop satisfies the axiom of propositional extensionality (types in Prop are equal if they are logically equivalent). In particular, in the terminology of homotopy type theory [41]: all types in Prop are h-propositions (all of their terms are equal) and all types in Set are h-sets (all equalities between terms are h-propositions).

4 Interpreting Higher-order Arithmetic in Type Theory

An interpretation of HAH-formulas in type theory can be divided into three parts: defining natural numbers, logical connectives, and power sets in type theory. For each part there are multiple options, which come with different requirements on the type theory. These requirements make sure that the type theory satisfies the rules and axioms of HAH for the interpretation. We systematically consider the three parts in the following subsections, and, in the end, we will have a clear overview of the various interpretations.

4.1 Interpreting Natural Numbers

To interpret the natural numbers, we use a natural numbers type. That is, that we have a type $\mathbb{N}$ that satisfies the following inference rules:

$$\frac{}{\vdash \mathbb{N} : \mathsf{Set}}\ \mathbb{N}\text{-F}, \qquad \frac{}{\vdash 0 : \mathbb{N}}\ \mathbb{N}\text{-I}_0, \qquad \frac{\Gamma \vdash n : \mathbb{N}}{\Gamma \vdash \mathsf{S}\, n : \mathbb{N}}\ \mathbb{N}\text{-I}_\mathsf{S},$$

$$\frac{\Gamma, n : \mathbb{N} \vdash C[n] : \mathcal{C} \qquad \Gamma \vdash c : C[0] \qquad \Gamma \vdash f : \Pi(n : \mathbb{N})\, (C[n] \to C[\mathsf{S}\, n])}{\Gamma \vdash \mathsf{ind}^{\mathbb{N}}_C\, c\, f : \Pi(n : \mathbb{N})\, C[n]}\ \mathbb{N}\text{-E},$$

$$\frac{}{\mathsf{ind}^{\mathbb{N}}_C\, c\, f\, 0 \equiv c}\ \mathbb{N}\text{-}\beta_0, \qquad \frac{}{\mathsf{ind}^{\mathbb{N}}_C\, c\, f\, (\mathsf{S}\, n) \equiv f\, n\, (\mathsf{ind}^{\mathbb{N}}_C\, c\, f\, n)}\ \mathbb{N}\text{-}\beta_\mathsf{S}.$$

Such a type can be assumed (as we do in λC+) or defined using other type constructors, and it is sufficient for the β-reduction rules to be satisfied propositionally. For example, the definition of natural numbers using W-types [30, 16] satisfies these rules propositionally.

A non-example is the impredicative Church encoding of natural numbers:

$$\mathbb{N} \coloneqq \Pi(C : \mathsf{Set})\,(C \to ((C \to C) \to C)) : \mathsf{Set}.$$

Here we would encode a natural number n as $\lambda C\,\lambda c\,\lambda f\,f^n\,c$. For $0 \coloneqq \lambda C\,\lambda c\,\lambda f\,c$ and $\mathsf{S}\,n \coloneqq \lambda C\,\lambda c\,\lambda f\,f\,(n\,C\,c\,f)$ this indeed satisfies the formation and introduction rules. However, it only satisfies a weak form of the elimination rule. For $\mathsf{rec}^{\mathbb{N}}_C\,c\,f \coloneqq \lambda n\,n\,C\,c\,f$ we have:

$$\frac{\Gamma \vdash C : \mathsf{Set} \qquad \Gamma \vdash c : C \qquad \Gamma \vdash f : C \to C \qquad \Gamma \vdash n : \mathbb{N}}{\Gamma \vdash \mathsf{rec}^{\mathbb{N}}_C\,c\,f : \mathbb{N} \to C}\ \mathbb{N}\text{-E, weak.}$$

This is weaker in two ways: (a) it only gives functions instead of dependent functions, and (b) the codomain must be in Set. From the perspective of category theory this means that we only have a weak natural numbers object [2], and crucially, from a logical perspective this will mean that we cannot prove the axiom scheme of induction.

In the Calculus of Constructions it is not possible to define a type in Prop that satisfies the strong elimination rule [19]. The same story is true for other inductive types: we can define types that satisfy the correct introduction rules, however they only satisfy a weak version of the elimination rules [20]. The work of Awodey, Frey, Speight, and Shulman [1, 36] shows that we can avoid limitation (a) using some additional assumptions ($\mathbb{N}$, Σ, $=$, and function extensionality). However, limitation (b) still applies.

4.2 Interpreting Logical Connectives

Logical connectives are the most influential part: we have a proof-irrelevant ($\bullet$) and a proof-relevant ($\circ$) interpretation. For any many-sorted logic, if we pick for every sort s corresponding types $s_\bullet$ and $s_\circ$, then the interpretations are defined as follows:

$$\begin{aligned}
(a =^s b)_\bullet &\coloneqq (a =_{s_\bullet} b), & (a =^s b)_\circ &\coloneqq (a =_{s_\circ} b),\\
(A \vee B)_\bullet &\coloneqq \|A_\bullet + B_\bullet\|, & (A \vee B)_\circ &\coloneqq A_\circ + B_\circ,\\
(A \wedge B)_\bullet &\coloneqq A_\bullet \times B_\bullet, & (A \wedge B)_\circ &\coloneqq A_\circ \times B_\circ,\\
(A \to B)_\bullet &\coloneqq A_\bullet \to B_\bullet, & (A \to B)_\circ &\coloneqq A_\circ \to B_\circ,\\
(\exists x^s\,B[x])_\bullet &\coloneqq \|\Sigma(x : s_\bullet)\,B[x]_\bullet\|, & (\exists x^s\,B[x])_\circ &\coloneqq \Sigma(x : s_\circ)\,B[x]_\circ,\\
(\forall x^s\,B[x])_\bullet &\coloneqq \Pi(x : s_\bullet)\,B[x]_\bullet, & (\forall x^s\,B[x])_\circ &\coloneqq \Pi(x : s_\circ)\,B[x]_\circ.
\end{aligned}$$

The difference is that the proof-irrelevant interpretation uses propositional-truncation [41, Section 3.7]. The propositional truncation $\|A\| : \mathsf{Prop}$ of a type A removes the distinctions between terms. For every $a : A$ we get a term $|a| : \|A\|$ and if we have $A \to C$ for $C : \mathsf{Prop}$ then we get $\|A\| \to C$. We can define propositional truncation in λC+ by taking $\|A\| \coloneqq \Pi(Z : \mathsf{Prop})\,((A \to Z) \to Z) : \mathsf{Prop}$ and $|a| \coloneqq \lambda Z\,\lambda f\,f\,a$.

To summarise: a term of $A_\bullet$ does not give us any information besides the fact that the formula holds while a term of $A_\circ$ contains a reason that the formula is true.

4.3 Interpreting Power Sets

Now we interpret the sorts of higher-order logic: we define $n_\bullet \coloneqq \mathcal{P}^n_\bullet\,\mathbb{N}$ and $n_\circ \coloneqq \mathcal{P}^n_\circ\,\mathbb{N}$ using the black box powertype $\mathcal{P}_\bullet\,A \coloneqq A \to \mathsf{Prop}$ and the white box powertype $\mathcal{P}_\circ\,A \coloneqq A \to \mathsf{Set}$. The element-relation is simply interpreted by $(x \in X)_\bullet \coloneqq X\,x$ and $(x \in X)_\circ \coloneqq X\,x$.

Recall that the axioms of higher-order logic should hold for a sound interpretation:

$$\exists X^{n+1}\,\forall z^n\,(z \in X \leftrightarrow P[z]), \qquad \text{(specification)}$$
$$\forall X^{n+1}\,\forall Y^{n+1}\,(\forall z^n\,(z \in X \leftrightarrow z \in Y) \to X = Y). \qquad \text{(extensionality)}$$

Specification holds for both interpretations: impredicativity of Prop and Set implies that $P[z]_\bullet : \mathsf{Prop}$ and $P[z]_\circ : \mathsf{Set}$ so in both cases we can take $X \coloneqq \lambda z\,P[z]$. Extensionality holds for $\circ$ because we have the following in λC+:

$$\mathsf{funext} : \Pi(f, f' : A \to \mathsf{Prop})\,(\Pi(x : A)\,(f\,x = f'\,x) \to (f = f')),$$
$$\mathsf{propext} : \Pi(P, P' : \mathsf{Prop})\,((P \leftrightarrow P') \to (P = P')).$$

However, it does not hold for $\bullet$, consider for example $X \coloneqq \lambda z \nVdash$ and $Y \coloneqq \lambda z \nvDash$. These represent the same proposition (the one that holds everywhere) but they are not equal.

So, $\bullet$ interprets HAH while $\circ$ only interprets HAH $-$ ext. In addition, there exists a second-order formula that is not provable in HAH [9], but whose proof-relevant interpretation in type theory is inhabited [41, Section 1.6], namely the axiom of choice:

$$\forall Z^1\,(\forall x^0\,\exists y^0\,\langle x, y\rangle \in Z \to \exists F^1\,(\forall x^0\,\exists! y^0\,\langle x, y\rangle \in F \wedge \forall x^0\,\forall y^0\,(\langle x, y\rangle \in F \to \langle x, y\rangle \in Z)).$$

This means that the second-order formulas that are provable in type theory for $\circ$ are incomparable with those provable in HAH. Therefore, our best hope is to show that type theory still proves the same first-order formulas. To do this, we use an interpretation e of HAH in HAH $-$ ext obtained by redefining $=$ and $\in$:

$$(a =_{\mathrm{e}}^0 b) \coloneqq (a =^0 b),$$
$$(A =_{\mathrm{e}}^{n+1} B) \coloneqq \forall x^n\,(x \in_{\mathrm{e}}^n A \leftrightarrow x \in_{\mathrm{e}}^n B),$$
$$(a \in_{\mathrm{e}}^n A) \coloneqq \exists x^n\,(a =_{\mathrm{e}}^n x \wedge x \in^n A).$$

For a formula A we write A_{e} for the result of replacing $=$ and $\in$ by $=_{\mathrm{e}}$ and $\in_{\mathrm{e}}$ respectively. The definition of $=_{\mathrm{e}}$ ensures that we satisfy extensionality and $\in_{\mathrm{e}}$ ensures that we respect the new equality. Note that this interpretation does not modify first-order formulas.

5 Interpreting Type Theory in Higher-order Arithmetic

To interpret our type theory in arithmetic, we will construct a model of λC+ using only notions that we can express within HAH. Our model can be seen as an modification of Hyland's small complete category [23], which forms a model for the calculus of constructions [35]. Our description can be incorporated into one of the usual categorical frameworks for type theory, such as comprehension categories [24], or categories with families [22]. The main idea is that we interpret our universes using the following categories:

$\mathsf{Prop} \rightsquigarrow \mathsf{Subsing}$ (the category of subsingletons),
$\mathsf{Set} \rightsquigarrow \mathsf{PER}$ (the category of partial equivalence relations),
$\mathsf{Type} \rightsquigarrow \mathsf{Assem}$ (the category of assemblies or $\mathcal{K}_1$-sets).

We will show that these categories have the right structure to interpret λC+, namely: embeddings $\mathsf{Subsing} \hookrightarrow \mathsf{PER} \hookrightarrow \mathsf{Assem}$, encodings of $\mathsf{Subsing}$ and PER as objects of Assem, definitions for $\mathbf{0}, \mathbf{1}, \mathbf{2}, \ldots, \mathbf{N}, \Sigma, \Pi, \mathrm{W}, =, \|\cdot\|, /$, and elements showing that the axioms are satisfied. We use this model to interpret every type in λC+ as a formula in HAH: the formula stating that the type is inhabited in the model. However, first we show why a naive model – of propositions, sets, and types in λC+ as sets in HAH or ZFC – cannot work. The notions we define for this naive approach will be useful to define our actual model.

5.1 Sets in HAH

Conventions. The sets in HAH are all subsets of $\mathcal{P}^n(\mathbb{N})$ for some n. It will be very convenient if we can view $\mathcal{P}^n(\mathbb{N})$ as a subset of $\mathcal{P}^{n+1}(\mathbb{N})$. Our motivation for this is that we want to define notions such as $\Sigma(a \in A)\, B[a]$, $\Pi(a \in A)\, B[a]$, and $\mathrm{W}(a \in A)\, B[a]$. If we view the hierarchy as cumulative, then we only need to define these notions for the case where A and all $B[a]$ are subsets of the same $\mathcal{P}^n(\mathbb{N})$. One way to achieve this is by considering $x \in \mathbb{N}$ to be equal to $\{\cdots\{x\}\cdots\} \in \mathcal{P}^n(\mathbb{N})$. This already gives us a cumulative hierarchy. For example: $\{0, 2\} \in \mathcal{P}(\mathbb{N})$ is viewed as $\{\{0\}, \{2\}\} \in \mathcal{P}^2(\mathbb{N})$, and $\{\{\{0\}\}, \{\{2\}\}\} \in \mathcal{P}^3(\mathbb{N})$, and so on. More formally: we define inclusions $\iota_n : \mathcal{P}^n(\mathbb{N}) \to \mathcal{P}^{n+1}(\mathbb{N})$ by $\iota_0(x) \coloneqq \{x\}$ and $\iota_{n+1}(X) \coloneqq \{\iota_n(x) \ : \ x \in X\}$. These are embeddings because they preserve $\in$-relation: we have $x \in Y$ iff $\iota_n(x) \in \iota_{n+1}(Y)$. From now on, we will use these emdeddings implicitly.

Secondly, it is convenient if we extend our definition of pairs from natural numbers to sets. We do this using the disjoint union, for $A, B \subseteq \mathcal{P}^n(\mathbb{N})$ we inductively define:

$$\langle A, B\rangle \coloneqq \{p \in \mathcal{P}^n(\mathbb{N}) \ : \ \exists(a \in A)\,(p = \langle 0, a\rangle) \vee \exists(b \in B)\,(p = \langle 1, b\rangle)\} \in \mathcal{P}^n(\mathbb{N}).$$

Definitions. Now we can inductively define Σ, Π, W inside HAH. If we have a set $A \subseteq \mathcal{P}^n(\mathbb{N})$ and for every $a \in A$ a set $B[a] \subseteq \mathcal{P}^n(\mathbb{N})$, then we define the dependent Cartesian product and dependent function space as follows:

$$\Sigma(a \in A)\, B[a] \coloneqq \{p \in \mathcal{P}^n(\mathbb{N}) \ : \ \exists(a \in A)\ \exists(b \in B[a])\,(\langle a, b\rangle = p)\} \subseteq \mathcal{P}^n(\mathbb{N}),$$
$$\Pi(a \in A)\, B[a] \coloneqq \{P \subseteq \Sigma(a \in A)\, B[a] \ : \ \forall(a \in A)\,\exists!(b \in B[a])\,(\langle a, b\rangle \in P)\} \subseteq \mathcal{P}^{n+1}(\mathbb{N}).$$

To define $\mathrm{W}(a \in A)\, B[a]$ we have to define labelled trees in HAH. A tree for A and B must satisfy the following: every node has a label $a \in A$, and a child for every $b \in B[a]$. We will encode a tree by describing the set of finite paths starting at the root. So, a tree will be a set $T \subseteq \mathcal{P}^n(\mathbb{N})$ whose elements are of the form $\langle a_0, b_0, a_1, \ldots, a_{n-1}, b_{n-1}, a_n\rangle$ such that for every i we have $a_i \in A$ and $b_i \in B[a_i]$. This set should be:

- inhabited: there exists an $a \in A$ such that $\langle a\rangle \in T$;
- downward-closed: if $\langle a_0, b_0, a_1, \ldots, a_n, b_n, a_{n+1}\rangle \in T$ then $\langle a_0, b_0, a_1, \ldots, a_n\rangle \in T$;
- complete: if we have $\langle a_0, b_0, a_1, \ldots, a_n\rangle \in T$ and $b_n \in B[a_n]$ then there exists an $a_{n+1} \in A$ such that $\langle a_0, b_0, a_1, \ldots, a_n, b_n, a_{n+1}\rangle \in T$;
- consistent: if $\langle a_0, b_0, a_1, \ldots, b_{n-1}, a_n\rangle, \langle a_0, b_0, a_1, \ldots, b_{n-1}, a'_n\rangle \in T$ then $a_n = a'_n$.

For two paths $p, q \in T$, we write $p \sqsubset q$ iff p is a strict subpath of q, that is, iff we can write $p = \langle a_0, b_0, a_1, \ldots, a_n\rangle$ and $q = \langle a_0, b_0, a_1, \ldots, a_m\rangle$ where $n < m$. We call a tree T well-founded iff the inverse relation $\sqsupset$ is well-founded, that is, if we have for any $S \subseteq T$:

$$\forall(p \in T)\,(\forall(q \sqsupset p)\,(q \in S) \to p \in S) \to S = T.$$

Now for $A, B[a \in A] \subseteq \mathcal{P}^n(\mathbb{N})$ we define:

$$\mathrm{W}(a \in A)\, B[a] \coloneqq \{T \subseteq \mathcal{P}^n(\mathbb{N}) \ : \ T \text{ is a well-founded tree for } A \text{ and } B[a]\} \subseteq \mathcal{P}^{n+1}(\mathbb{N}).$$

Problems. It is important to note for $A, B[a \in A] \subseteq \mathcal{P}^n(\mathbb{N})$ that, while $\Sigma(a \in A)\, B[a]$ is still a subset of $\mathcal{P}^n(\mathbb{N})$, we see that $\Pi(a \in A)\, B[a]$ and $\mathrm{W}(a \in A)\, B[a]$ are both subsets of $\mathcal{P}^{n+1}(\mathbb{N})$. So Π and W increase the level. This is a problem: if we interpret Prop and Set as subsets of some $\mathcal{P}^n(\mathbb{N})$ then they cannot be closed under Π and W. This problem exists in general for naive interpretations of impredicative type theory. If we interpret an impredicative universe as a set $\mathcal{U}$ in ZFC, then for $A, B \in \mathcal{U}$ we must have $A \to B \coloneqq \Pi(a \in A)\, B \in \mathcal{U}$. If $\mathcal{U}$ contains a set A with at least two elements, then we get a contradiction for the cardinality:

$$\begin{aligned}
|\Pi(B \in \mathcal{U})\,(B \to A)| &= |(\Sigma(B \in \mathcal{U})\,B) \to A| && \text{(by Currying)}\\
&\geq |\mathcal{P}(\Sigma(B \in \mathcal{U})\,B)| && \text{(because } |A| \geq 2\text{)}\\
&\geq |\mathcal{P}(\Pi(B \in \mathcal{U})\,(B \to A))| && \text{(because } \Pi(B \in \mathcal{U})\,(B \to A) \in \mathcal{U}\text{)}\\
&> |\Pi(B \in \mathcal{U})\,(B \to A)|. && \text{(by Cantor's diagonal argument)}
\end{aligned}$$

This counterexample comes from lectures of Hyland and Streicher, see [28, 35]. If $\mathcal{U}$ only consists of subsingletons, then we have no contradiction, and we obtain a model for simple type theories like ML0 and λC [37]. Indeed, we use this approach to interpret Prop as the set $\mathcal{P}(\{*\})$. The intuitive idea behind our other interpretations, of Set and Type, is that we restrict $\Pi(a \in A)\,B[a]$ and $\mathrm{W}(a \in A)\,B[a]$ to the elements that are in some sense computable.

5.2 Subsingletons, PERs, and Assemblies

In this subsection we define the three categories we use to model λC+. We start simple:

▶ **Definition 4** (subsingleton). *A* subsingleton *is a subset $S \subseteq \{*\}$. A* subsingleton morphism *from S to T is just a function from S to T.*

Because we want our model to satisfy propositional extensionality, we always consider subsets of the same singleton $\{*\}$; by defining for example: $* \coloneqq 0$. Note that we cannot prove intuitionistically for every $S \subseteq \{*\}$ that $S = \emptyset$ or $S = \{*\}$, so $\mathcal{P}(\{*\})$ can be large [29].

The next categories are more interesting and use a notion of computation. We use Kleene's first algebra [26, 27, 10]: the fact that natural numbers can be seen as codes for partial computable functions. For $f, n \in \mathbb{N}$, we will write $f\,n\downarrow$ iff the partial computable function encoded by f is defined on the natural number n, and $f\,n$ for the result. In Section 7 we will consider a conservative extension of HAH where $f\,n$ and $\downarrow$ are primitive notions that satisfy a computational choice principle. This will be needed to show conservativity in the proof-relevant case. For now however, think of $f\,n$ as Kleene-application as described here.

▶ **Definition 5** (PER). *A* partial equivalence relation (PER) *is a relation $R \subseteq \mathbb{N} \times \mathbb{N}$ that is symmetric and transitive. For a PER R we define:*

$$\begin{aligned}
\mathrm{dom}(R) &\coloneqq \{n \in \mathbb{N} \;:\; \langle n, n\rangle \in R\} = \{n \in \mathbb{N} \;:\; \exists (m \in \mathbb{N})\,\langle n, m\rangle \in R\}, && \text{(domain)}\\
[n]_R &\coloneqq \{m \in \mathbb{N} \;:\; \langle n, m\rangle \in R\}, && \text{(equivalence class)}\\
\mathbb{N}/R &\coloneqq \{[n]_R \;:\; n \in \mathrm{dom}(R)\}. && \text{(quotient)}
\end{aligned}$$

A PER morphism from R to S is a function $F : \mathbb{N}/R \to \mathbb{N}/S$ that is "tracked" by some $f \in \mathbb{N}$, meaning that $a \in \mathrm{dom}(R)$ implies $f\,a\downarrow$ and $f\,a \in F([n]_R)$.

The intuition is made clear by the following: suppose that we have a type and and want to define a PER to model it. The idea is that we view natural numbers as potential codes or realizers for terms of the type. Consider the relation that relates natural numbers when they encode the same term. This explains why we consider PER's: the relation is symmetric and transitive but not necessarily reflexive as not every natural number has to encode a term. Using this principle we define PER's that model $\nvdash$,$\nVdash$,$\nvDash$, ..., and $\mathbb{N}$:

$$\mathbf{n} \coloneqq \{\langle i, j\rangle \;:\; i = j \wedge i < n\}, \qquad (\text{for } n = 0, 1, 2, \dots) \qquad\qquad \mathbf{N} \coloneqq \{\langle i, j\rangle \;:\; i = j\}.$$

The PER morphisms are those functions on equivalence classes (which we view as terms) that can be implemented as partial computable functions acting on the codes.

The next category generalises these ideas of modelling types:

▶ **Definition 6** (assembly). *An n-assembly consists of a set $\mathcal{A} \subseteq \mathcal{P}^n(\mathbb{N})$ and a relation $\Vdash \subseteq \mathbb{N} \times \mathcal{A}$ such that for every $A \in \mathcal{A}$ there exists a "realizer" $a \in \mathbb{N}$ such that $a \Vdash_{\mathcal{A}} A$. For an n-assembly $\mathcal{A}$ we will write $|\mathcal{A}|$ for the set and $\Vdash_{\mathcal{A}}$ for the relation. An n-assembly morphism from $\mathcal{A}$ to $\mathcal{B}$ is a function F from $\mathcal{A}$ to $\mathcal{B}$ that is "tracked" by some $f \in \mathbb{N}$, meaning that for every $A \in \mathcal{A}$ and $a \in \mathbb{N}$ with $a \Vdash_{\mathcal{A}} A$ we have $f\,a \downarrow$ and $f\,a \Vdash_{\mathcal{B}} F(A)$.*

The inclusions $\iota_n : \mathcal{P}^n(\mathbb{N}) \to \mathcal{P}^{n+1}(\mathbb{N})$ also give us a cumulative hierarchy of assemblies.

Cumulativity. We can view any subsingleton $S \subseteq \{*\}$ as a PER $\{\langle i, j\rangle \ : \ * \in S\}$ and any PER R as a 1-assembly with domain $\mathbb{N}/R$ and realizability relation $\in$. This gives us full embeddings: $\mathsf{Subsing} \hookrightarrow \mathsf{PER} \hookrightarrow \mathsf{Assem}$ which we use to model cumulativity in λC+. In a similar vein, for any set $A \subseteq \mathcal{P}^n(\mathbb{N})$, we get an n-assembly ∇A with domain A and the total realizability relation $\mathbb{N} \times A$; in this way $\mathcal{P}^n(\mathbb{N})$ also forms a full subcategory of Assem^n.

Universes. We have to show that we can view the sets $\mathsf{Subsing}$ and PER as assemblies to model $\mathsf{Prop} : \mathsf{Type}$ and $\mathsf{Set} : \mathsf{Type}$. For $\mathsf{Subsing}$ this is easy, if we take $* \coloneqq 0$, then we have $\mathsf{Subsing} = \mathcal{P}(\{0\}) \subseteq \mathcal{P}(\mathbb{N})$ so we get an 1-assembly $\nabla\mathsf{Subsing}$. Similarly, we can consider a PER to be a subset of $\mathbb{N} \times \mathbb{N} \coloneqq \Sigma(x \in \mathbb{N})\,\mathbb{N} \subseteq \mathbb{N}$, so we get a 1-assembly $\nabla\mathsf{PER}$.

5.3 Modelling Type Constructors

We define Σ, Π, W for assemblies by restricting the definitions for sets to those elements which are realised. So, for $\mathcal{A}, \mathcal{B}[A \in \mathcal{A}] \in \mathsf{Assem}^n$ and $\mathcal{Q} = \Sigma, \Pi, \mathrm{W}$, we define the assembly $\mathcal{Q}(A \in \mathcal{A})\,\mathcal{B}[A]$ by taking $|\mathcal{Q}(A \in \mathcal{A})\,\mathcal{B}[A]| \coloneqq \{Q \in \mathcal{Q}(A \in |\mathcal{A}|)\,|\mathcal{B}[A]| \ : \ \exists(q \in \mathbb{N})\,(q \Vdash Q)\}$, where $\Vdash \subseteq \mathbb{N} \times |\mathcal{Q}(A \in \mathcal{A})\,\mathcal{B}[A]|$ is defined as follows for $\mathcal{Q} = \Sigma, \Pi, \mathrm{W}$:

- Σ We say $p \Vdash P$ iff we have $\mathsf{pr}_0(p) \Vdash_{\mathcal{A}} \mathsf{pr}_0(P)$ and $\mathsf{pr}_1(p) \Vdash_{\mathcal{B}[A]} \mathsf{pr}_1(P)$.
- Π We say $f \Vdash F$ iff for every $A \in \mathcal{A}$ and $a \Vdash_{\mathcal{A}} A$ we have $f\,a \downarrow$ and $f\,a \Vdash_{\mathcal{B}[A]} F(A)$.
- W We say $t \Vdash T$ iff for every $\langle A_0, B_0, A_1, \ldots, A_{n-1}, B_{n-1}, A_n\rangle \in T$ and $b_0 \Vdash_{\mathcal{B}[A_0]} B_0$, $\ \ldots,$ $b_{n-1} \Vdash_{\mathcal{B}[A_{n-1}]} B_{n-1}$ we have that for $t_0, \ldots, t_{n-1} \in \mathbb{N}$ given inductively by $t_0 \coloneqq t$ and $t_{i+1} \coloneqq (\mathsf{pr}_1(t_i))\,b_i$ we have for every $i < n+1$ that $t_i \downarrow$ and $\mathsf{pr}_0(t_i) \Vdash_{\mathcal{A}} A_i$.

Now that we have defined Σ, Π, W for assemblies, we use this to define these notions also for subsingletons and PER's. This is possible because of the following observation:

▶ **Proposition 7.** *Suppose that $\mathcal{A} \in \mathsf{Assem}^n$ and for every $A \in \mathcal{A}$ that $\mathcal{B}[A] \in \mathsf{Assem}^n$.*

- *If $\mathcal{A}$ and all $\mathcal{B}[A]$ are isomorphic to a subsingleton/PER, then $\Sigma(A \in \mathcal{A})\,\mathcal{B}[A]$ is as well.*
- *If all $\mathcal{B}[A]$ are isomorphic to a subsingleton/PER, then $\Pi(A \in \mathcal{A})\,\mathcal{B}[A]$ is as well.*
- *If $\mathcal{A}$ is isomorphic to a subsingleton/PER, then $\mathrm{W}(A \in \mathcal{A})\,\mathcal{B}[A]$ is as well.*

Note that this is precisely what we need to model our formation rules. In particular, we can model the impredicative rule for products: up to isomorphism, we have that $\Pi(A \in \mathcal{A})\,\mathcal{B}[A]$ always lives in the same category as the $\mathcal{B}[A]$, regardless of $\mathcal{A}$.

For $\mathcal{A} \in \mathsf{Assem}^n$ we define equality and propositional truncation as subsingletons:

$$(A =_{\mathcal{A}} A') \coloneqq \{* \ : \ A = A'\} \in \mathsf{Subsing}, \qquad \|\mathcal{A}\| \coloneqq \{* \ : \ \exists A\,(A \in \mathcal{A})\} \in \mathsf{Subsing}.$$

Lastly, if we have $\mathcal{A} \in \mathsf{Assem}^n$ and for every $A, A' \in \mathcal{A}$ an $\mathcal{R}[A, A'] \in \mathsf{Assem}^n$, then we define $\mathcal{A}/\mathcal{R} \in \mathsf{Assem}^{n+1}$ by taking $|\mathcal{A}/\mathcal{R}| \coloneqq |\mathcal{A}|/\{\langle A, A'\rangle \ : \ \exists(R \in \mathcal{R}[A, A'])\}$ and $q \Vdash Q$ iff there exists an $A \in Q$ such that $q \Vdash A$. We can extend this to PER's and subsingletons using:

▶ **Proposition 8.** *If $\mathcal{A} \in \mathsf{Assem}^n$ is isomorphic to a subsingleton/PER, then $\mathcal{A}/\mathcal{R}$ is as well.*

5.4 Interpretation

Model. Now that we have all of the building blocks, we pack everything together to build our model. Using simultaneous induction on the derivation we define:

- for any well-formed context Γ an n-assembly $[\![\Gamma]\!]$ for some n;
- for any judgement $\Gamma \vdash A : \mathsf{Type}$ a function $[\![\Gamma \vdash A : \mathsf{Type}]\!] : [\![\Gamma]\!] \to \mathsf{Assem}^n$ for some n;
- for any judgement $\Gamma \vdash a : A$ a morphism $[\![\Gamma \vdash a : A]\!] : [\![\Gamma]\!] \to [\![\Gamma \vdash A : \mathsf{Type}]\!]$.

Here contexts are the only part that we have not yet discussed. We define $[\![\Gamma]\!]$ using Σ:

$$[\![]\!] \coloneqq \mathbf{1}, \qquad [\![\Gamma, x : A]\!] \coloneqq \Sigma(G \in [\![\Gamma]\!])\, [\![\Gamma \vdash A : \mathsf{Type}]\!](G).$$

The other two judgements use the structure that we have defined in the previous sections: the embeddings $\mathsf{Subsing} \hookrightarrow \mathsf{PER} \hookrightarrow \mathsf{Assem}$, the assemblies $\nabla\mathsf{Subsing}$ and $\nabla\mathsf{PER}$, and the constructions $\mathbf{n}, \mathbf{N}, \Sigma, \Pi, \mathrm{W}, =, \|\cdot\|, /$. The full model is given in Appendix B.

Realizability. Now that we have defined a model for λC+ within HAH, we consider the two interpretations HAH $\to \lambda$C+ $\to$ HAH. The idea is as follows: for a formula A, we get types $\Gamma^A_\bullet \vdash A_\bullet : \mathsf{Prop}$ and $\Gamma^A_\circ \vdash A_\circ : \mathsf{Set}$. By interpreting these in our model we get a subsingleton $[\![A_\bullet]\!] \coloneqq [\![\Gamma^A_\bullet \vdash A_\bullet : \mathsf{Prop}]\!](G^A_\bullet)$ and a PER $[\![A_\circ]\!] \coloneqq [\![\Gamma_{A_\circ} \vdash A_\circ : \mathsf{Set}]\!](G^A_\circ)$ by defining some canonical $G^A_\bullet \in [\![\Gamma^A_\bullet]\!]$ and $G^A_\circ \in [\![\Gamma^A_\circ]\!]$ that have the same free variables as A. We consider the HAH-formulas: $\mathsf{Inh}([\![A_\bullet]\!])$ and $\mathsf{Inh}([\![A_\circ]\!])$ where $\mathsf{Inh}(A) \coloneqq \exists x\,(x \in A)$. These formulas have the same free variables as A and state the the types are inhabited in the model, that is, that the model satisfies A.

To make this precise, we consider the context of A_* for $* \coloneqq \bullet, \circ$. If A has free variables $x_0^{n_0}, \dots, x_{k-1}^{n_{k-1}}$ then the context is $\Gamma^A_* \coloneqq (x_0 : \mathcal{P}^{n_0}_* \mathbb{N}, \dots, x_{k-1} : \mathcal{P}^{n_{k-1}}_* \mathbb{N})$. This means that $[\![\Gamma^A_*]\!] = [\![\mathcal{P}^{n_0}_* \mathbb{N}]\!] \times \cdots \times [\![\mathcal{P}^{n_{k-1}}_* \mathbb{N}]\!]$ where $[\![\mathbb{N}]\!] \coloneqq (\mathbb{N}/{=}) = \{\{x\} \;:\; x \in \mathbb{N}\}$ and we have $[\![\mathcal{P}^{n+1}_\bullet \mathbb{N}]\!] \coloneqq [\![\mathcal{P}^n_\bullet \mathbb{N}]\!] \to \nabla\mathsf{Subsing}$ and $[\![\mathcal{P}^{n+1}_\circ \mathbb{N}]\!] \coloneqq [\![\mathcal{P}^n_\circ \mathbb{N}]\!] \to \nabla\mathsf{PER}$. We can translate between $\mathcal{P}(\mathbb{N})$ and $[\![\mathcal{P}^n_* \mathbb{N}]\!]$ using $g^n_* : \mathcal{P}^n(\mathbb{N}) \to [\![\mathcal{P}^n_* \mathbb{N}]\!]$ and $h^n_* : [\![\mathcal{P}^n_* \mathbb{N}]\!] \to \mathcal{P}^n(\mathbb{N})$:

$$g^0_*(x) \coloneqq \{x\}, \qquad g^{n+1}_*(X) \coloneqq (f \in [\![\mathcal{P}^n \mathbb{N}]\!]) \mapsto \{z \;:\; h^n_*(f) \in X\},$$
$$h^0_*(\{x\}) \coloneqq x, \qquad h^{n+1}_*(F) \coloneqq \{x \in \mathcal{P}^n(\mathbb{N}) \;:\; \mathsf{Inh}(F(g^n_*(x)))\}.$$

Now we define $G^A_* \coloneqq \langle g^{n_0}_*(x_0), \dots, g^{n_{k-1}}_*(x_{k-1})\rangle$.

6 Proof-irrelevant Conservativity

Now that we have defined our interpretations, the proof-irrelevant result follows quickly:

▶ **Theorem 9.** *For any* HAH*-formula* A*, we have* $\mathrm{HAH} \vdash \mathsf{Inh}([\![A_\bullet]\!]) \leftrightarrow A$.

Proof. We prove this with induction on the formula A. Because the other logical connectives can be defined using $\in$, $\to$, and $\forall$, and because λC+ satisfies the rules and axioms of HAH, we only have to check the following cases:

$$\begin{aligned}\mathsf{Inh}([\![(x \in^n Y)_\bullet]\!]) &\leftrightarrow * \in [\![x : \mathcal{P}^n_\bullet \mathbb{N}, Y : \mathcal{P}^{n+1}_\bullet \mathbb{N} \vdash Y\, x : \mathsf{Prop}]\!](\langle g^n_\bullet(x), g^{n+1}_\bullet(Y)\rangle)\\ &\leftrightarrow * \in g^{n+1}_\bullet(Y)(g^n_\bullet(x))\\ &\leftrightarrow h^n_\bullet(g^n_\bullet(x)) \in Y\\ &\leftrightarrow x \in^n Y,\end{aligned}$$

$$\begin{aligned}
\mathsf{Inh}([\![(A \to B)_\bullet]\!]) &\leftrightarrow * \in [\![\Gamma_\bullet^{A\to B} \vdash A_\bullet \to B_\bullet : \mathsf{Prop}]\!](G_\bullet^{A\to B}) \\
&\leftrightarrow * \in \Pi(h \in [\![\Gamma_\bullet^{A} \vdash A_\bullet : \mathsf{Prop}]\!](G_\bullet^{A}))\, [\![\Gamma_\bullet^{B} \vdash B_\bullet : \mathsf{Prop}]\!](G_\bullet^{B}) \\
&\leftrightarrow * \in [\![\Gamma_\bullet^{A} \vdash A_\bullet : \mathsf{Prop}]\!](G_\bullet^{A}) \to * \in [\![\Gamma_\bullet^{B} \vdash B_\bullet : \mathsf{Prop}]\!](G_\bullet^{B}) \\
&\leftrightarrow A \to B,
\end{aligned}$$

$$\begin{aligned}
\mathsf{Inh}([\![(\forall x^n\, B[x])_\bullet]\!]) &\leftrightarrow * \in [\![\Gamma_\bullet^{\forall x^n B[x]} \vdash \Pi(x : \mathcal{P}_\bullet^n\, \mathbb{N})\, B[x]_\bullet : \mathsf{Prop}]\!](G_\bullet^{\forall x^n B[x]}) \\
&\leftrightarrow * \in \Pi(f \in [\![\mathcal{P}^n\, \mathbb{N}]\!])\, [\![\Gamma_\bullet^{B[x]} \vdash B[x]_\bullet : \mathsf{Prop}]\!](\langle G_\bullet^{\forall x^n B[x]}, f\rangle) \\
&\leftrightarrow \forall (f \in [\![\mathcal{P}_\bullet^n\, \mathbb{N}]\!]) * \in [\![\Gamma_\bullet^{B[x]} \vdash B[x]_\bullet : \mathsf{Prop}]\!](\langle G_\bullet^{\forall x^n B[x]}, f\rangle) \\
&\leftrightarrow \forall (x \in \mathcal{P}_\bullet^n(\mathbb{N})) * \in [\![\Gamma_\bullet^{B[x]} \vdash B[x]_\bullet : \mathsf{Prop}]\!](\langle G_\bullet^{\forall x^n B[x]}, g_\bullet^n(x)\rangle) \\
&\leftrightarrow \forall x^n\, B[x].
\end{aligned}$$

◀

▶ **Corollary 10** (proof-irrelevant conservativity). *For a higher-order arithmetical formula A, we have that* HAH *proves A iff there exists a term a such that* λC+ *proves $\Gamma_\bullet^A \vdash a : A_\bullet$.*

Proof. We have already seen that λC+ satisfies the axioms and inference rules of HAH, so it is an extension of HAH. That this extension is conservative will follow from the previous theorem. Suppose for a formula A in the language of HAH that it is provable in λC+, that is, that we have $\Gamma_\bullet^A \vdash a : A_\bullet$ for some term a. Then we get $[\![\Gamma_\bullet^A \vdash a : A_\bullet]\!](G_\bullet^A) \in [\![\Gamma_\bullet^A \vdash A_\bullet : \mathsf{Prop}]\!](G_\bullet^A)$ so we have $\mathsf{Inh}([\![A_\bullet]\!])$. Using the last theorem we see that A is provable in HAH. ◀

7 Proof-relevant Conservativity

In the proof of Theorem 9, we used the fact that, from second-order logic upwards, we can define every logical connective using $\in$, $\to$, and $\forall$. Because our conservativity will only hold for first-order formulas, we cannot use this shortcut. It turns out that $\vee$ and $\exists$ are the difficult cases; luckily, in HA we can define $A \vee B \coloneqq \exists n^0\, ((n = 0 \to A) \wedge (n \neq 0 \to B))$ so we only have to worry about $\exists$. Similarly, we can define $\bot \coloneqq (0 = 1)$ and $\top \coloneqq (0 = 0)$. First we write out what $\langle z, z'\rangle \in [\![A_\circ]\!]$ means by unrolling the definition:

▶ **Proposition 11.** *In* HAH*, we can prove the following:*

$$\begin{aligned}
&\langle z, z'\rangle \in [\![(\ a =^0 b\ )_\circ]\!] \leftrightarrow a =^0 b, \\
&\langle z, z'\rangle \in [\![(\ A \wedge B\ )_\circ]\!] \leftrightarrow \langle \mathsf{pr}_0\, z,\ \mathsf{pr}_0\, z'\rangle \in [\![A_\circ]\!] \wedge \langle \mathsf{pr}_1\, z,\ \mathsf{pr}_1\, z'\rangle \in [\![B_\circ]\!], \\
&\langle z, z'\rangle \in [\![(\ A \to B\ )_\circ]\!] \leftrightarrow \forall x, x'\, (\langle x, x'\rangle \in [\![A_\circ]\!] \to \langle z\, x, z'\, x'\rangle \in [\![B_\circ]\!]), \\
&\langle z, z'\rangle \in [\![(\exists x^0 B[x])_\circ]\!] \leftrightarrow \mathsf{pr}_0\, z = \mathsf{pr}_0\, z' \wedge \langle \mathsf{pr}_1\, z, \mathsf{pr}_1\, z'\rangle \in [\![B[\mathsf{pr}_0\, z]_\circ]\!], \\
&\langle z, z'\rangle \in [\![(\forall x^0 B[x])_\circ]\!] \leftrightarrow \forall x\, (\langle z\, x, z'\, x\rangle \in [\![B[x]_\circ]\!]).
\end{aligned}$$

Now, we will prove conservativity by using an extra assumption: that we have Hilbert-style epsilon constants. That is, we assume for every first-order formula $A[\vec{x}, y]$, that there exists a choice function $\epsilon_{y.A} \in \mathbb{N}$ sending every $\vec{x}$ to some y such that $A[\vec{x}, y]$ if such a y exists:

$$\forall \vec{x}\, (\exists y\, A[\vec{x}, y] \to \epsilon_{y.A}\, \vec{x}\downarrow), \qquad \forall \vec{x}\, (\epsilon_{y.A}\, \vec{x}\downarrow \to A[\vec{x}, \epsilon_{y.A}\, \vec{x}]).$$

Unfortunately, this assumption is not true for Kleene-application; however, in the next sections, we will see that we can conservatively extend HAH with a notion of application where these constants exist. First, we show how this allows us to prove conservativity:

▶ **Theorem 12.** *Assuming ϵ-constants, for any* HA*-formula A, we have* $\mathsf{Inh}([\![A_\circ]\!]) \leftrightarrow A$.

Proof. For any HA-formula A with free variables $\vec{x}$, we construct a canonical realizer r_A:

$$\begin{aligned}
r_{a=^0 b} &:= \lambda\vec{x}\,0,\\
r_{A\wedge B} &:= \lambda\vec{x}\,\langle r_A\,\vec{x}, r_B\,\vec{x}\rangle,\\
r_{A\to B} &:= \lambda\vec{x}\,\lambda y\,(r_B\,\vec{x}),\\
r_{\exists y^0\,B[y]} &:= \lambda\vec{x}\,\langle \epsilon_{y.B}\,\vec{x}, r_B\,\vec{x}\,(\epsilon_{y.B}\,\vec{x})\rangle,\\
r_{\forall y^0\,B[y]} &:= \lambda\vec{x}\,\lambda y\,(r_B\,\vec{x}\,y).
\end{aligned}$$

With induction on A, we can prove $\mathsf{Inh}([\![A_\circ]\!]) \leftrightarrow (r_A\,\vec{x}\downarrow \wedge \langle r_A\,\vec{x}, r_A\,\vec{x}\rangle \in [\![A_\circ]\!]) \leftrightarrow A$. ◀

What remains is proving that we can make this assumption. Here, we translate the approach of [42] to higher-order logic. First, in Subsection 7.1 we extend our higher-order logic to allow for partial function symbols. Then in Subsection 7.2 we extend HAH to HAHP by adding primitive notions for application. This extension is conservative because these notions can already be defined using Kleene-application. Then in Subsection 7.3, we extend further, to HAHPϵ by adding ϵ-constants, and show that this is still conservative over HAH.

7.1 Higher-order Logic of Partial Terms

We will consider a higher-order version of the logic of partial terms by Beeson [3, Section VI.1]. In this logic, function symbols are allowed to correspond to partial functions. So, if we have a function symbol f then $f(\vec{x})$ is not necessarily defined. For every term a we add a new atomic formula $a\downarrow$, which stands for "a is defined". We add the following inference rules:

$$\frac{}{\Gamma\vdash x^n\downarrow,}\downarrow\text{-var},\qquad \frac{\Gamma\vdash f(a_0^{n_0},\dots,a_{k-1}^{n_{k-1}})\downarrow}{\Gamma\vdash a_i^{n_i}\downarrow}\downarrow\text{-fun},\qquad \frac{\Gamma\vdash R(a_0^{n_0},\dots,a_{k-1}^{n_{k-1}})}{\Gamma\vdash a_i^{n_i}\downarrow}\downarrow\text{-rel}.$$

Note that we view $=^n$ and $\in^n$ as relation symbols so the $\downarrow$-rel rule applies. In addition, we restrict the exists-introduction and forall-elimination rules to terms that are defined:

$$\frac{\Gamma\vdash B[a^n]\qquad \Gamma\vdash a^n\downarrow}{\Gamma\vdash \exists x^n\,B[x^n]}\exists\text{-I},\qquad \frac{\Gamma\vdash \forall x^n\,B[x^n]\qquad \Gamma\vdash a^n\downarrow}{\Gamma\vdash B[a^n]}\forall\text{-E}.$$

The other rules are the same as those of higher-order logic. Any theory in higher-order logic can be seen as a theory in the higher-order logic of partial terms by adding for every function symbol $f : n^0\times\cdots\times n^k\to m$ the axiom $\forall x_0^{n_0}\ \dots\ \forall x_k^{n^k}\,f(x_0,\dots,x_k)\downarrow$. Accordingly, we will view HAH as a theory in this new logic.

In this logic, it is often useful to consider a weaker notion of equality that also holds when both terms are not defined: $a^n\simeq b^n := a\downarrow\vee b\downarrow\to a=b$.

7.2 HAHP: Adding Primitive Application

In HAHP, we extend the language with a binary partial function symbol $\mathsf{app} : (0,0)\rightharpoonup 0$, and constants $\mathsf{k},\mathsf{s},\mathsf{suc},\mathsf{rec} : 0$ which stand for natural numbers encoding basic functions: k and s give a partial combinatory algebra [6], suc computes the successor function, and rec allows us to do recursion. We abbreviate $\mathsf{app}(a,b)$ as $a\,b$. For $\mathsf{k},\mathsf{s},\mathsf{suc},\mathsf{rec}$, we add the axioms:

$$\begin{aligned}
&\forall x\,\forall y\,(\mathsf{k}\,x\,y = x),\qquad && \forall x\,(\mathsf{suc}\,x = \mathsf{S}(x)),\\
&\forall x\,\forall y\,(\mathsf{s}\,x\,y\downarrow),\qquad && \forall x\,\forall y\,(\mathsf{rec}\,x\,y\,0 = x),\\
&\forall x\,\forall y\,\forall z\,(\mathsf{s}\,x\,y\,z\simeq (x\,z)\,(y\,z)),\qquad && \forall x\,\forall y\,\forall z\,(\mathsf{rec}\,x\,y\,(\mathsf{S}\,z)\simeq y\,z\,(\mathsf{rec}\,x\,y\,z)).
\end{aligned}$$

The raison d'être for k and s is that they are used to define $\lambda x\, b[x]$:

$$\begin{aligned}
\lambda x\, x &\coloneqq \mathsf{s}\,\mathsf{k}\,\mathsf{k}, & \lambda x\, \mathsf{S}(b[x]) &\coloneqq \lambda x\, (\mathsf{suc}\ b[x]),\\
\lambda x\, c &\coloneqq \mathsf{k}\, c, \quad (\text{if } c \neq x) & \lambda x\, b[x] + c[x] &\coloneqq \lambda x\, (\mathsf{add}\, b[x]\, c[x]),\\
\lambda x\, (b[x]\, c[x]) &\coloneqq \mathsf{s}\, (\lambda x\, b[x])\, (\lambda x\, c[x]), & \lambda x\, b[x] \times c[x] &\coloneqq \lambda x\, (\mathsf{mul}\, b[x]\, c[x]),
\end{aligned}$$

where $\mathsf{add} \coloneqq \lambda y\, \mathsf{rec}\, y\, (\lambda i\, \lambda r\, \mathsf{suc}\ r)$ and $\mathsf{mul} \coloneqq \lambda y\, \mathsf{rec}\, 1\, (\lambda i\, \lambda r\, \mathsf{add}\, r\, y)$.

These lambda functions are enough to construct our model in HAHP using app as our application. We write $[\![\cdot]\!]' : \lambda\mathrm{C}+ \to \mathrm{HAHP}$ for this modification of $[\![\cdot]\!] : \lambda\mathrm{C}+ \to \mathrm{HAH}$.

▶ **Theorem 13.** HAHP *is conservative over* HAH*.*

Proof. Kleene-application satisfies the axioms of HAHP. Here, $\mathsf{app}(a, b)$ is the application of the partial recursive function encoded by a to b. See [40, Proposition 9.3.12] for more. ◀

7.3 HAHPϵ: Adding Computable Choice

In HAHPϵ, we extend the theory even further by adding a constant $\epsilon_{\exists y A} : 0$ for every HAH-formula $A[x^0_0, \ldots, x^0_{k-1}, y^0]$, and adding the following axioms:

$$\forall \vec{x}\, (\exists y\, A[\vec{x}, y] \to \epsilon_{\exists y\, A}\, \vec{x}\downarrow), \qquad \forall \vec{x}\, (\epsilon_{\exists y\, A}\, \vec{x}\downarrow \to A[\vec{x}, \epsilon_{\exists y\, A}\, \vec{x}]).$$

Such constants do not exist for Kleene-application, so HAHPϵ is not conservative over HAHP. However, HAHPϵ is conservative over HAH as we will prove in the remainder of this section.

▶ **Proposition 14.** *Suppose that* $A[x^0, y^0]$ *is an* HAH*-formula and let* HAHPF *be the extension of* HAHP *with a relation symbol* $F : 0 \times 0$ *and axioms:*

$$\forall x\, !y\, F(x, y), \qquad \forall x\, (\exists y\, A[x, y] \to \exists y\, F(x, y)), \qquad \forall x\, \forall y\, (F(x, y) \to A[x, y]),$$

where $!y$ *means "at most one* y*" which is defined by* $!y\, B[y] \coloneqq \forall y\, \forall y'\, (B[y] \wedge B[y'] \to y = y')$*. Then* HAHP$F$ *is conservative over* HAHP*.*

Proof. We prove this using forcing. We define a forcing condition P to be a finite approximation of the relation F: a finite set of pairs $\{\langle x_0, y_0\rangle, \ldots, \langle x_{n-1}, y_{n-1}\rangle\}$ where the x_i are distinct and for every $i < n$ we have $A(x_i, y_i)$. For a forcing condition P and an HAHPF-formula A we define a HAHP-formula $P \Vdash_R A$ with induction on A:

$$\begin{aligned}
P \Vdash_R A &\coloneqq A, && (\text{if } A \text{ is an atomic HAHP-formula})\\
P \Vdash_R F(x, y) &\coloneqq \forall (P' \supseteq P)\, \exists (P'' \supseteq P')\, (\langle x, y\rangle \in P''),\\
P \Vdash_R A \vee B &\coloneqq \forall (P' \supseteq P)\, \exists (P'' \supseteq P')\, ((P'' \Vdash_R A) \vee (P'' \Vdash_R B)),\\
P \Vdash_R A \wedge B &\coloneqq (P \Vdash_R A) \wedge (P \Vdash_R B),\\
P \Vdash_R A \to B &\coloneqq \forall (P' \supseteq P)\, ((P' \Vdash_R A) \to (P' \Vdash_R B)),\\
P \Vdash_R \exists x^n\, B[x] &\coloneqq \forall (P' \supseteq P)\, \exists (P'' \supseteq P')\, \exists x^n\, (P'' \Vdash_R B[x^n]),\\
P \Vdash_R \forall x^n\, B[x] &\coloneqq \forall (P' \supseteq P)\, \forall x^n\, (P' \Vdash_R B[x]).
\end{aligned}$$

As usual, we can prove with induction for every HAHPF-formula A that we have:

$$\begin{aligned}
&\mathrm{HAHP} \vdash \forall P\, \forall (P' \supseteq P)\, ((P \Vdash_R A) \to (P' \Vdash_R A)),\\
&\mathrm{HAHP} \vdash \forall P\, (\forall (P' \supseteq P)\, \exists (P'' \supseteq P')\, (P'' \Vdash_R A) \to (P \Vdash_R A)).
\end{aligned}$$

Similarly, for every HAHP-formula B we show with induction on B that:

$$\mathsf{HAHP} \vdash \forall P\,((P \Vdash_R B) \leftrightarrow B).$$

With this, we prove for every HAHPF-formula A that HAHP$F \vdash A$ implies $\mathsf{HAHP} \vdash \forall P\,(P \Vdash_R A)$ with induction on the proof of $\mathsf{HAHP}F \vdash A$. This is tedious but straightforward. See also the proof of [42, Proposition 2.5] where they show a similar statement.

This shows that $\mathsf{HAHP}F$ is conservative over HAHP: suppose for a HAHP-formula B that we have $\mathsf{HAHP}F \vdash B$, then we have $\mathsf{HAHP} \vdash \forall P\,(P \Vdash_R B)$ and therefore $\mathsf{HAHP} \vdash B$. ◀

▶ **Proposition 15.** *Suppose that $A[x^0, y^0]$ is an* HAH*-formula and let* HAHf *be the extension of* HAHP *with a partial function symbol $f : 0 \rightharpoonup 0$ and axioms:*

$$\forall x\,(\exists y\,A[x,y] \to f(x)\downarrow), \qquad \forall x\,(f(x)\downarrow \to A[x, f(x)]).$$

Then HAHf *is conservative over* HAHP*.*

Proof. This follows because every n-ary function symbol f can be encoded as an $(n+1)$-ary relation symbol F and an axiom $\forall \vec{x}\,!y\,F(\vec{x}, y)$. The axioms of Proposition 14 are precisely the axioms of this proposition under this encoding. For a similar translation in more detail, see [39, Section 2.7]. ◀

▶ **Proposition 16.** *Suppose that $A[x^0, y^0]$ is an* HAH*-formula and let* HAHPc *be the extension of* HAHP *with a constant $\mathsf{c} : 0$ and axioms:*

$$\forall x\,(\exists y\,A[x,y] \to \mathsf{c}\,x\downarrow), \qquad \forall x\,(\mathsf{c}\,x\downarrow\!\to A[x, \mathsf{c}\,x]).$$

Then HAHPc *is conservative over* HAH*.*

Proof. We work in HAHPf and use our existing evaluation $\mathsf{eval}(a,b)$ to define a new evaluation $\mathsf{eval}^f(a,b)$, which can use the partial function symbol f as an oracle.

The informal idea to calculate $\mathsf{eval}^f(a,b)$ is the following. We start by calculating $\mathsf{eval}(a,b)$. If this returns a value $\langle 0, x_0\rangle$ then that means that the function a wants to ask the oracle for the result of applying f to x_0. So we supply this value and run the function again, now we calculate $\mathsf{eval}(a, \langle b, f(x_0)\rangle)$. If this returns a value $\langle 0, x_1\rangle$ then the function a want another result from the oracle so we calculate $\mathsf{eval}(a, \langle b, f(x_0), f(x_1)\rangle)$. We keep doing this until a eventually returns a value $\langle 1, c\rangle$ in which case we say $\mathsf{eval}^f(a,b) = c$.

More formally, we say that the formula $\mathsf{eval}^f(a,b) = c$ is true if there exists a sequence $\langle x_0, \ldots, x_{n-1}\rangle$ such that:

- for every $i < n$ we have $\mathsf{eval}(a, \langle b, f(x_0), \ldots, f(x_{i-1})\rangle) = \langle 0, x_i\rangle$;
- and $\mathsf{eval}(a, \langle b, f(x_0), \ldots, f(x_{n-1})\rangle) = \langle 1, c\rangle$.

For this new evaluation we can define new constants $\mathsf{k}^f, \mathsf{s}^f, \mathsf{suc}^f, \mathsf{rec}^f$ which leads to a new lambda abstraction λ^f. For the details, see [43, Theorem 2.2].

We can use this to show that HAHPc is conservative over HAH. For any HAHPc-formula A we relativize the evaluation and constants to f to get an HAHf-formula A^f, we can prove with induction that we have HAHP$\mathsf{c} \vdash A$ iff HAH$f \vdash A^f$. For an HAH-formula B we see that B^f is the same as B so HAHP$\mathsf{c} \vdash B$ implies HAH$f \vdash B$ which implies HAH $\vdash B$. ◀

▶ **Theorem 17.** HAHϵ *is conservative over* HAH*.*

Proof. Suppose that we have $\mathsf{HAH}\epsilon \vdash A$ for an HAH-formula A. Note that the proof for A can only use a finite amount of choice functions, say $\epsilon_{\exists y\,B_i}$ for $i < n$. We can modify the proof of A to use only the choice function $\epsilon_{\exists y\,C}$ where $C[z,y] \coloneqq \bigwedge_{i<n} \forall \vec{x}\,(z = \langle i, \vec{x}\rangle \to B_i[\vec{x}, y])$. So, the theorem follows from the previous proposition. ◀

▶ **Corollary 18** (proof-relevant conservativity). *For a first-order arithmetical formula A, we have that* HAH *proves A iff there exists a term a such that* λC+ *proves* $\Gamma_A \vdash a : A_\circ$.

Proof. This follows from Theorem 12 and Theorem 17 in the same way as Corollary 10. ◀

8 De Jongh's Theorem for Type Theory

Before proving it for type theory, let us first state De Jongh's original theorem:

▶ **Theorem 19** (De Jongh [12]). *Let $A[P_0, \ldots, P_{n-1}]$ be a propositional formula with propositional variables $P_0, \ldots, P_{n-1}$. If A is not provable in intuitionistic propositional logic, then we can construct sentences $B_0, \ldots, B_{n-1}$ in the language of* HA *such that $A[B_0, \ldots, B_{n-1}]$ is not provable in* HA.

De Jongh and Smorynski have shown that this also holds for HA2 [13] and Robert Passmann has shown it for CZF and IZF [32, 33]. First we observe that we can use Passman's proof to obtain a new result for HAH:

▶ **Corollary 20.** *De Jongh's Theorem holds for* HAH.

Proof. This theorem follows from Passmann's proof for IZF because of the following two observations: HAH can be seen as a subtheory of IZF, and the sentences $B_0, \ldots, B_{n-1}$ used by Passmann can already be stated in the language of HAH. See Appendix C. ◀

Now, using our conservativity results, we see the following:

▶ **Corollary 21.** *De Jongh's Theorem holds for* λC+ *(and smaller type theories) for both the proof-relevant and proof-irrelevant interpretations of (higher-order) logic.*

In particular, we see that this holds for both predicative and impredicative theories and for both intuitionistic and extensional theories with at most one level of universes.

9 Conclusion and Future Work

The interpretations of higher-order logic in type theory differ greatly on second-order formulas:

- the proof-irrelevant interpretation satisfies specification and extensionality but not choice,
- the proof-relevant interpretation satisfies specification and choice but not extensionality.

However, although having all three of these principles makes the theory classical [15], these interpretations still prove exactly the same first-order arithmetical formulas: those of the intuitionistic theory HAH. These results hold for both intensional and extensional versions of type theory and are sufficient to prove De Jongh's theorem for both predicative and impredicative versions.

We have characterised the arithmetical statements provable in type theories with one level of impredicative universes. This gives two natural directions of future work:

- Can we find a characterisation for predicative type theories? For such a type theory both interpretations do not satisfy specification, so, can we find a corresponding weaker arithmetical theory?
- Can we find a characterisation for type theories with more universes?

References

1 Steve Awodey, Jonas Frey, and Sam Speight. Impredicative encodings of (higher) inductive types. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '18, pages 76–85, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3209108.3209130.

2 Steve Awodey, Nicola Gambino, and Kristina Sojakova. Inductive types in homotopy type theory. In *2012 27th Annual IEEE Symposium on Logic in Computer Science*, pages 95–104, 2012. doi:10.1109/LICS.2012.21.

3 Michael Beeson. *Foundations of Constructive Mathematics*. A Series of Modern Surveys in Mathematics. Springer, Berlin, 1985.

4 Stefano Berardi. Encoding of data types in pure construction calculus: a semantic justification. In G. Huet and G. Plotkin, editors, *Logical Environments*, pages 30–60, 1993.

5 Yves Bertot and Pierre Castéran. *Interactive theorem proving and program development, Coq'Art: the calculus of inductive constructions*. Springer Science & Business Media, 2013.

6 Ingemarie Bethke. *Notes on partial combinatory algebras*. PhD thesis, University of Amsterdam, 1988.

7 Andrés Caicedo. How is exponentiation defined in Peano arithmetic? Mathematics Stack Exchange, 2013. (version: 2017-04-13). URL: https://math.stackexchange.com/q/313049.

8 Georg Cantor. Ein Beitrag zur Mannigfaltigkeitslehre. *Journal für die reine und angewandte Mathematik*, 84:242–258, 1877. URL: http://eudml.org/doc/148353.

9 Ray-Ming Chen and Michael Rathjen. Lifschitz realizability for intuitionistic Zermelo–Fraenkel set theory. *Archive for Mathematical Logic*, 51(7):789–818, 2012.

10 J.H. Conway. *Regular Algebra and Finite Machines*. Chapman and Hall mathematics series. Dover Publications, Incorporated, 2012. URL: https://books.google.nl/books?id=1KAXc5TpEV8C.

11 Thierry Coquand and Gérard Huet. The calculus of constructions. *Information and Computation*, 76(2):95–120, 1988. doi:10.1016/0890-5401(88)90005-3.

12 Dick de Jongh. The maximality of the intuitionistic predicate calculus with respect to Heyting's arithmetic. *The Journal of Symbolic Logic*, 1970.

13 Dick de Jongh and Craig Smorynski. Kripke models and the intuitionistic theory of species. *Annals of Mathematical Logic*, 9(1):157, 1976. doi:10.1016/0003-4843(76)90008-5.

14 Dick de Jongh, Rineke Verbrugge, and Albert Visser. Intermediate logics and the de jongh property. *Archive for Mathematical Logic*, 50(1-2):197–213, 2011.

15 Radu Diaconescu. Axiom of choice and complementation. *Proceedings of the American Mathematical Society*, 51(1):176–178, 1975. URL: http://www.jstor.org/stable/2039868.

16 Peter Dybjer. Representing inductively defined sets by wellorderings in Martin-Löf's type theory. *Theoretical Computer Science*, 176(1):329–335, 1997. doi:10.1016/S0304-3975(96)00145-4.

17 Harvey Friedman and Andrej Ščedrov. On the quantificational logic of intuitionistic set theory. *Mathematical proceedings of the Cambridge Philosophical Society*, 99(1):5–10, 1986.

18 Herman Geuvers. *The calculus of constructions and higher order logic*, pages 139–191. Cahiers du centre de logique. Katholieke Universiteit Leuven, Belgium, 1994.

19 Herman Geuvers. Induction is not derivable in second order dependent type theory. In Samson Abramsky, editor, *Typed Lambda Calculi and Applications*, pages 166–181, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

20 Jean-Yves Girard. *Proofs and types*, volume 7. Cambridge university press Cambridge, 1989.

21 Petr Hájek and Pavel Pudlák. *Metamathematics of first-order arithmetic*, volume 3. Cambridge University Press, 2017.

22 Martin Hofmann. *Syntax and Semantics of Dependent Types*, pages 79–130. Publications of the Newton Institute. Cambridge University Press, 1997. doi:10.1017/CBO9780511526619.004.

23 Martin Hyland. A small complete category. *Annals of pure and applied logic*, 40(2):135–165, 1988.

24 Bart Jacobs. Comprehension categories and the semantics of type dependency. *Theoretical Computer Science*, 107(2):169–207, 1993. doi:10.1016/0304-3975(93)90169-T.

25 Richard Kaye. *Models of Peano arithmetic*. Clarendon Press, 1991.

26 S. C. Kleene. *Representation of Events in Nerve Nets and Finite Automata*, pages 3–42. Princeton University Press, Princeton, 1956. doi:10.1515/9781400882618-002.

27 D. Kozen. A completeness theorem for kleene algebras and the algebra of regular events. *Information and Computation*, 110(2):366–390, 1994. doi:10.1006/inco.1994.1037.

28 Giuseppe Longo and Eugenio Moggi. Constructive natural deduction and its 'ω-set' interpretation. *Mathematical Structures in Computer Science*, 1(2):215–254, 1991. doi:10.1017/S0960129500001298.

29 Robert S. Lubarsky. Independence results around constructive ZF. *Annals of Pure and Applied Logic*, 132(2):209–225, 2005. doi:10.1016/j.apal.2004.08.002.

30 Per Martin-Löf. *Intuitionistic type theory*. Studies in proof theory. Lecture notes; 1 861180607. Bibliopolis, Napoli, 1984.

31 Daniël Otten. De Jongh's theorem for type theory. Master's thesis, University of Amsterdam, 2022. URL: https://eprints.illc.uva.nl/id/document/12640.

32 Robert Passmann. De Jongh's theorem for intuitionistic Zermelo-Fraenkel set theory. In Maribel Fernández and Anca Muscholl, editors, *28th EACSL Annual Conference on Computer Science Logic (CSL 2020)*, volume 152 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 33:1–33:16, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.CSL.2020.33.

33 Robert Passmann. The first-order logic of CZF is intuitionistic first-order logic. *The Journal of Symbolic Logic*, pages 1–23, 2022. doi:10.1017/jsl.2022.51.

34 Christine Paulin-Mohring. Introduction to the calculus of inductive constructions, 2015.

35 Bernhard Reus. Realizability models for type theories. *Electronic Notes in Theoretical Computer Science*, 23(1):128–158, 1999. Tutorial Workshop on Realizability Semantics and Applications (associated to FLoC'99, the 1999 Federated Logic Conference). doi:10.1016/S1571-0661(04)00108-2.

36 Mike Shulman. Impredicative encodings, part 3, November 2018. URL: https://homotopytypetheory.org/2018/11/26/impredicative-encodings-part-3/.

37 Jan M. Smith. The independence of Peano's fourth axiom from Martin-Löf's type theory without universes. *The Journal of Symbolic Logic*, 53(3):840–845, 1988. URL: http://www.jstor.org/stable/2274575.

38 Anne S. Troelstra. *Metamathematical Investigation of Intuitionistic Arithmetic and Analysis*. New York,: Springer, 1973.

39 Anne Sjerp Troelstra and Dirk van Dalen. *Constructivism in Mathematics*, volume I. North-Holland Publishing Co., 1988.

40 Anne Sjerp Troelstra and Dirk van Dalen. *Constructivism in Mathematics*, volume II. North-Holland Publishing Co., 1988.

41 The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. https://homotopytypetheory.org/book, Institute for Advanced Study, 2013.

42 Benno van den Berg and Lotte van Slooten. Arithmetical conservation results. *Indagationes Mathematicae*, 29:260–275, 2018.

43 Jaap van Oosten. A general form of relative recursion. *Notre Dame Journal of Formal Logic*, 47(3):311–318, 2006.

A Type Theory

$$x \text{ not free in } \Gamma\ \frac{\Gamma \vdash A : \mathcal{A}}{\Gamma, x : A \vdash x : A}\ \text{start}, \qquad x \text{ not free in } \Gamma\ \frac{\Gamma \vdash A : \mathcal{A} \qquad \Gamma \vdash b : B}{\Gamma, x : A \vdash b : B}\ \text{weakening},$$

$$\frac{}{\vdash \mathsf{Prop} : \mathsf{Type}}\ \text{axiom}_\mathsf{P}, \qquad \frac{\Gamma \vdash A : \mathsf{Prop}}{\Gamma \vdash A : \mathsf{Set}}\ \text{cumul}_\mathsf{P}, \qquad \frac{\Gamma \vdash A \equiv A' : \mathcal{A} \qquad \Gamma \vdash a : A}{\Gamma \vdash a : A'}\ \text{convers},$$

$$\frac{}{\vdash \mathsf{Set} : \mathsf{Type}}\ \text{axiom}_\mathsf{S}, \qquad \frac{\Gamma \vdash A : \mathsf{Set}}{\Gamma \vdash A : \mathsf{Type}}\ \text{cumul}_\mathsf{S}, \qquad \frac{\Gamma \vdash p : a =_A a'}{\Gamma \vdash a \equiv a' : A}\ \text{reflection},$$

Rules stating that $\Gamma \vdash \cdot \equiv \cdot : A$ is a congruence relation have been omitted for brevity.

*

$$n \text{ numeral}\ \frac{}{\vdash \ltimes : \mathsf{Set}}\ \ltimes\text{-F}, \qquad k < n\ \frac{}{\vdash k_\ltimes : \ltimes}\ \ltimes\text{-I},$$

$$\frac{\Gamma, i : \ltimes \vdash C[i] : \mathcal{C} \qquad \Gamma \vdash c_0 : C[0_\ltimes] \qquad \dots \qquad \Gamma \vdash c_{n-1} : C[(n-1)_\ltimes]}{\Gamma \vdash \mathsf{ind}^\ltimes_C\, c_0\ \dots\ c_{n-1} : \Pi(i : \ltimes)\, C[i]}\ \ltimes\text{-E},$$

$$k < n\ \frac{\Gamma \vdash \mathsf{ind}^\ltimes_C\, c_0\ \dots\ c_{n-1}\, k_\ltimes : C[k_\ltimes]}{\Gamma \vdash \mathsf{ind}^\ltimes_C\, c_0\ \dots\ c_{n-1}\, k_\ltimes \equiv c_k : C[k_\ltimes]}\ \ltimes\text{-}\beta,$$

$$\frac{}{\vdash \mathbb{N} : \mathsf{Set}}\ \mathbb{N}\text{-F}, \qquad \frac{}{\vdash 0 : \mathbb{N}}\ \mathbb{N}\text{-I}_0, \qquad \frac{\Gamma \vdash n : \mathbb{N}}{\Gamma \vdash \mathsf{S}\, n : \mathbb{N}}\ \mathbb{N}\text{-I}_\mathsf{S},$$

$$\frac{\Gamma, n : \mathbb{N} \vdash C[n] : \mathcal{C} \qquad \Gamma \vdash c : C[0] \qquad \Gamma \vdash f : \Pi(n : \mathbb{N})\, (C[n] \to C[\mathsf{S}\, n])}{\Gamma \vdash \mathsf{ind}^\mathbb{N}_C\, c\, f : \Pi(n : \mathbb{N})\, C[n]}\ \mathbb{N}\text{-E},$$

$$\frac{\Gamma \vdash \mathsf{ind}^\mathbb{N}_C\, c\, f\, 0 : C[0]}{\Gamma \vdash \mathsf{ind}^\mathbb{N}_C\, c\, f\, 0 \equiv c : C[0]}\ \mathbb{N}\text{-}\beta_0, \qquad \frac{\Gamma \vdash \mathsf{ind}^\mathbb{N}_C\, c\, f\, (\mathsf{S}\, n) : C[\mathsf{S}\, n]}{\Gamma \vdash \mathsf{ind}^\mathbb{N}_C\, c\, f\, (\mathsf{S}\, n) \equiv f\, n\, (\mathsf{ind}^\mathbb{N}_C\, c\, f\, n) : C[\mathsf{S}\, n]}\ \mathbb{N}\text{-}\beta_\mathsf{S},$$

$$\frac{\Gamma \vdash A : \mathcal{C} \quad \Gamma, x : A \vdash B[x] : \mathcal{C}}{\Gamma \vdash \Sigma(x : A)\, B[x] : \mathcal{C}}\ \Sigma\text{-F}, \qquad \frac{\Gamma \vdash \Sigma(x : A)\, B[x] : \mathcal{C} \quad \Gamma \vdash a : A \quad \Gamma \vdash b : B[a]}{\Gamma \vdash \langle a, b\rangle : \Sigma(x : A)\, B[x]}\ \Sigma\text{-I},$$

$$\frac{\Gamma, p : \Sigma(x : A)\, B[x] \vdash C[p] : \mathcal{C} \qquad \Gamma \vdash f : \Pi(x : A)\, \Pi(y : B[a])\, C[\langle x, y\rangle]}{\Gamma \vdash \mathsf{ind}^\Sigma_C\, f : \Pi(p : \Sigma(x : A)\, B[x])\, C[p]}\ \Sigma\text{-E},$$

$$\frac{\Gamma \vdash \mathsf{ind}^\Sigma_C\, f\, \langle a, b\rangle : C[\langle a, b\rangle]}{\Gamma \vdash \mathsf{ind}^\Sigma_C\, f\, \langle a, b\rangle \equiv f\, a\, b : C[\langle a, b\rangle]}\ \Sigma\text{-}\beta,$$

$$\frac{\Gamma \vdash A : \mathcal{A} \qquad \Gamma, x : A \vdash B[x] : \mathcal{B}}{\Gamma \vdash \Pi(x : A)\, B[x] : \mathcal{B}}\ \Pi\text{-F}, \qquad \frac{\Gamma \vdash \Pi(x : A)\, B[x] : \mathcal{B} \qquad \Gamma, x : A \vdash b[x] : B[x]}{\Gamma \vdash \lambda(x : A)\, b[x] : \Pi(x : A)\, B[x]}\ \Pi\text{-I},$$

$$\frac{\Gamma \vdash f : \Pi(x : A)\, B[x] \qquad \Gamma \vdash a : A}{\Gamma \vdash f\, a : B[a]}\ \Pi\text{-E}, \qquad \frac{\Gamma \vdash (\lambda(x : A)\, b[x])\, a : B[a]}{\Gamma \vdash (\lambda(x : A)\, b[x])\, a \equiv b[a] : B[a]}\ \Pi\text{-}\beta,$$

$$\frac{\Gamma \vdash A : \mathcal{A} \qquad \Gamma, x : A \vdash B[x] : \mathcal{B}}{\Gamma \vdash \mathrm{W}(x : A)\, B[x] : \mathcal{A}}\ \text{W-F},$$

$$\frac{\Gamma \vdash \mathrm{W}(x : A)\, B[x] : \mathcal{A} \qquad \Gamma \vdash a : A \qquad \Gamma \vdash d : B[a] \to \mathrm{W}(x : A)\, B[x]}{\Gamma \vdash \mathsf{tree}\, a\, d : \mathrm{W}(x : A)\, B[x]}\ \text{W-I},$$

$$\frac{\begin{array}{c}\Gamma, t : \mathrm{W}(x : A)\, B[x] \vdash C[t] : \mathcal{C} \\ \Gamma \vdash f : \Pi(a : A)\, \Pi(d : B[a] \to \mathrm{W}(x : A)\, B[x])\, ((\Pi(b : B[a])\, C[d\, b]) \to C[\mathsf{tree}\, a\, d])\end{array}}{\mathsf{ind}^{\mathrm{W}}_C\, f : \Pi(t : \mathrm{W}(x : A)\, B[x])\, C[t]}\ \text{W-E},$$

$$\frac{\Gamma \vdash \mathsf{ind}^{\mathrm{W}}_C\, f\, (\mathsf{tree}\, a\, d) : C[\mathsf{tree}\, a\, d]}{\Gamma \vdash \mathsf{ind}^{\mathrm{W}}_C\, f\, (\mathsf{tree}\, a\, d) \equiv f\, a\, d\, (\lambda(b : B\, a)\, \mathsf{ind}^{\mathrm{W}}_C\, f\, (d\, b)) : C[\mathsf{tree}\, a\, d]}\ \text{W-}\beta,$$

$$\frac{\Gamma \vdash A : \mathcal{A} \qquad \Gamma \vdash a : A \qquad \Gamma \vdash a' : A}{\Gamma \vdash a =_A a' : \mathsf{Prop}}\ =\text{-F}, \qquad \frac{\Gamma \vdash A : \mathcal{A} \qquad \Gamma \vdash a : A}{\Gamma \vdash \mathsf{refl}\, a : a =_A a}\ =\text{-I},$$

$$\frac{\Gamma, a : A, a' : A, e : a =_A a' \vdash C[a, a', e] : \mathcal{C} \qquad \Gamma \vdash f : \Pi(x : A)\, C[x, x, \mathsf{refl}\, x]}{\Gamma \vdash \mathsf{ind}^{=}_C\, f : \Pi(x, x' : A)\, \Pi(e : x =_A x')\, C[x, x', e]}\ =\text{-E},$$

$$\frac{\Gamma \vdash \mathsf{ind}^{=}_C\, f\, a\, a\, (\mathsf{refl}\, a) : C[a, a, \mathsf{refl}\, a]}{\Gamma \vdash \mathsf{ind}^{=}_C\, f\, a\, a\, (\mathsf{refl}\, a) \equiv f\, a : C[a, a, \mathsf{refl}\, a]}\ =\text{-}\beta,$$

$$\frac{\Gamma \vdash A : \mathcal{A}}{\Gamma \vdash \|A\| : \mathsf{Prop}}\ \|\cdot\|\text{-F}, \qquad \frac{\Gamma \vdash \|A\| : \mathsf{Prop} \qquad \Gamma \vdash a : A}{\Gamma \vdash |a| : \|A\|}\ \|\cdot\|\text{-I},$$

$$\frac{\Gamma, t : \|A\| \vdash C[t] : \mathsf{Prop} \qquad \Gamma \vdash f : \Pi(x : A)\, C[|x|]}{\Gamma \vdash \mathsf{ind}^{\|\cdot\|}_C\, f : \Pi(t : \|A\|)\, C[t]}\ \|\cdot\|\text{-E},$$

$$\frac{\Gamma \vdash \mathsf{ind}^{\|\cdot\|}_C\, f\, h\, |a| : C[|a|]}{\Gamma \vdash \mathsf{ind}^{\|\cdot\|}_C\, f\, h\, |a| \equiv f\, a : C[|a|]}\ \|\cdot\|\text{-}\beta,$$

$$\frac{\Gamma \vdash A : \mathcal{A} \qquad \Gamma, x : A, x' : A \vdash R[x, x'] : \mathcal{B}}{\Gamma \vdash A/R : \mathcal{A}}\ /\text{-F}, \qquad \frac{\Gamma \vdash A/R : \mathcal{A} \qquad \Gamma \vdash a : A}{\Gamma \vdash [a]_R : A/R}\ /\text{-I},$$

$$\frac{\begin{array}{c}\Gamma, q : A/R \vdash C[q] : \mathcal{C} \qquad \Gamma \vdash f : \Pi(x : A)\, C[[x]_R] \\ \Gamma \vdash h : \Pi(x, x' : A)\, \Pi(r : R[x, x'])\, ((\mathsf{ax}_{/}\, x\, x'\, r)_*\, (f\, x) = f\, x')\end{array}}{\Gamma \vdash \mathsf{ind}^{/}_C\, f\, h : \Pi(q : A/R)\, C[q]}\ /\text{-E},$$

$$\frac{\Gamma \vdash A/R : \mathcal{A}}{\Gamma \vdash \mathsf{ax}_{/} : \Pi(a, a' : A)\, (R[a, a'] \to [a]_R = [a']_R)}\ /\text{-I}_{=}, \qquad \frac{\Gamma \vdash \mathsf{ind}^{/}_C\, f\, h\, [a]_R : C[[a]_R]}{\Gamma \vdash \mathsf{ind}^{/}_C\, f\, h\, [a]_R \equiv f\, a : C[[a]_R]}\ /\text{-}\beta,$$

$$\frac{}{\vdash \mathsf{propext} : \Pi(P, P' : \mathsf{Prop})\, ((P \leftrightarrow P) \to (P =_{\mathsf{Prop}} P'))}\ \text{propext}.$$

B Model

Using simultaneous induction on the derivation we define:

- for any well-formed context Γ an n-assembly $[\![\Gamma]\!]$ for some n;
- for any judgement $\Gamma \vdash A : \mathsf{Type}$ a function $[\![\Gamma \vdash A : \mathsf{Type}]\!] : [\![\Gamma]\!] \to \mathsf{Assem}^n$ for some n;
- for any judgement $\Gamma \vdash a : A$ a morphism $[\![\Gamma \vdash a : A]\!] : [\![\Gamma]\!] \to [\![\Gamma \vdash A : \mathsf{Type}]\!]$.

For contexts, we define:

$$[\![]\!] \coloneqq \mathbf{1}, \qquad [\![\Gamma, x : A]\!] \coloneqq \Sigma(G \in [\![\Gamma]\!])\, [\![\Gamma \vdash A : \mathsf{Type}]\!](G).$$

For the start and weakening laws, we define:

$$[\![\Gamma, x : A \vdash x : A]\!](G) \coloneqq \mathsf{pr}_1(G), \quad [\![\Gamma, x : A \vdash b : B]\!](G) \coloneqq [\![\Gamma \vdash b : B]\!](G).$$

For the β-conversion law, if $\Gamma \vdash A \equiv A' : \mathcal{A}$, then we define:

$$[\![\Gamma \vdash a : A']\!](G) \coloneqq [\![\Gamma \vdash a : A]\!](G),$$

For the axioms and cumulativity laws, we define:

$$[\![\vdash \mathsf{Prop} : \mathsf{Type}]\!] \coloneqq \nabla\mathsf{Subsing}, \qquad [\![\Gamma \vdash A : \ \mathsf{Set}\ ]\!](G) \coloneqq \{\langle z, z' \rangle \ : \ * \in [\![\Gamma : \mathsf{Prop}]\!](G)\},$$
$$[\![\vdash \ \mathsf{Set}\ : \mathsf{Type}]\!] \coloneqq \nabla\mathsf{PER}, \qquad [\![\Gamma \vdash A : \mathsf{Type}]\!](G) \coloneqq \mathbb{N}/[\![\Gamma \vdash A : \mathsf{Set}]\!](G).$$

For the finite types, we define:

$$[\![\vdash \ltimes : \mathsf{Set}]\!] \coloneqq \mathbf{n}, \qquad [\![\vdash k_n : \ltimes]\!] \coloneqq \{k\},$$
$$[\![\Gamma \vdash \mathsf{ind}^{\ltimes}_C\, c_0 \ \ldots \ c_{n-1} : \Pi(k : \ltimes)\, C[k]]\!](G)(\{k\}) \coloneqq [\![\Gamma \vdash c_k : C[k]]\!](G).$$

For the natural numbers, we define:

$$[\![\vdash \mathbb{N} : \mathsf{Set}]\!] \coloneqq \mathbf{N}, \qquad [\![\vdash 0 : \mathbb{N}]\!] \coloneqq \{0\} \qquad [\![\Gamma \vdash \mathsf{S}\, n : \mathbb{N}]\!](G) \coloneqq \mathsf{S}([\![\Gamma \vdash n : \mathbb{N}]\!](G)),$$
$$[\![\Gamma \vdash \mathsf{ind}^{\mathbb{N}}_C\, c\, f : \Pi(n : \mathbb{N})\, C[n]]\!](G)(\{n\}) \coloneqq$$
$$[\![\Gamma \vdash f : \Pi(n : \mathbb{N})\, (C[n] \to C[\mathsf{S}\, n])]\!](G)(\{n-1\})(\ldots$$
$$[\![\Gamma \vdash f : \Pi(n : \mathbb{N})\, (C[n] \to C[\mathsf{S}\, n])]\!](G)(\{0\})([\![\Gamma \vdash c : C[0]]\!](G))).$$

For Σ-types, we define:

$$[\![\Gamma \vdash \ \Sigma\, (x : A)\, B[x] : \mathcal{C}]\!](G) \coloneqq \ \Sigma\, (X \in [\![\Gamma \vdash A : \mathcal{C}]\!](G))\, [\![\Gamma, x : A \vdash B[x] : \mathcal{C}]\!](\langle G, X \rangle),$$
$$[\![\Gamma \vdash \langle a, b \rangle : \Sigma(x : A)\, B[x]]\!](G) \coloneqq \langle [\![\Gamma \vdash a : A]\!](G), [\![\Gamma \vdash b : B[a]]\!](G) \rangle,$$
$$[\![\Gamma \vdash \mathsf{ind}^{\Sigma}_C\, f : \Pi(p : \Sigma(x : A)\, B[x])\, C[p]]\!](G)(\langle A, B \rangle) \coloneqq$$
$$[\![\Gamma \vdash \ f : \Pi(x : A)\, \Pi(y : B[x])\, C[\langle x, y \rangle]]\!](G)(A)(B).$$

For Π-types, we define:

$$[\![\Gamma \vdash \ \Pi\, (x : A)\, B[x] : \mathcal{B}]\!](G) \coloneqq \ \Pi\, (X \in [\![\Gamma \vdash A : \mathcal{A}]\!](G))\, [\![\Gamma, x : A \vdash B[x] : \mathcal{B}]\!](\langle G, X \rangle),$$
$$[\![\Gamma \vdash \lambda(x : A)\, b[x] : \Pi(x : A)\, B[x]]\!](G)(A) \coloneqq [\![\Gamma, x : A \vdash b[x] : B[x]]\!](\langle G, A \rangle),$$
$$[\![\Gamma \vdash f\, a : B[a]]\!](G) \coloneqq [\![\Gamma \vdash f : \Pi(x : A)\, B[x]]\!](G)([\![\Gamma \vdash a : A]\!](G)).$$

For W-types, we define:

$$\begin{aligned}
&[\![\Gamma \vdash \mathrm{W}(x : A)\, B[x] : \mathcal{A}]\!](G) \coloneqq \mathrm{W}(X \in [\![\Gamma \vdash A : \mathcal{A}]\!](G))\, [\![\Gamma, x : A \vdash B[x] : \mathcal{B}]\!](\langle G, X\rangle),\\
&[\![\Gamma \vdash \mathsf{tree}\, a\, d : \mathrm{W}(x : A)\, B[x]]\!](G) \coloneqq \{\langle A_0, B_0, A_1, \dots, A_n\rangle \;:\; A_0 = [\![\Gamma \vdash a : A]\!](G)\\
&\qquad \wedge \langle A_1, B_1, A_2, \dots, A_n\rangle \in [\![\Gamma \vdash d : B[a] \to \mathrm{W}(x : A)\, B[x]]\!](G)(B_0)\},\\
&[\![\Gamma \vdash \mathsf{ind}^{\mathrm{W}}_C\, f : \Pi(t : \mathrm{W}(x : A)\, B[x])\, C[t]]\!](G)(T) \coloneqq\\
&\qquad [\![\Gamma \vdash f : \Pi(a : A)\, \Pi(d : B[a] \to \mathrm{W}(x : A)\, B[x])\, ((\Pi(b : B[a])\, C[d\, b]) \to C[\mathsf{tree}\, a\, d])]\!]\\
&\qquad (G)(\mathsf{root}(T))(B_0 \mapsto \{\langle A_1, B_1, A_2, \dots, A_n\rangle \;:\; \langle \mathsf{root}(T), B_0, A_1, \dots, A_n\rangle \in T\})(\dots).
\end{aligned}$$

For propositional equality, we define:

$$\begin{aligned}
&[\![\Gamma \vdash a =_A a' : \mathsf{Prop}]\!](G) \coloneqq ([\![\Gamma \vdash a : A]\!](G) =_{[\![\Gamma \vdash A : \mathsf{Type}]\!](G)} [\![\Gamma \vdash a' : A]\!](G)),\\
&[\![\Gamma \vdash \mathsf{refl}\, a : a =_A a]\!](G) \coloneqq *,\\
&[\![\Gamma \vdash \mathsf{ind}^{=}_C\, f : \Pi(x, x' : A)\, \Pi(e : x =_A x')\, C[x, x', e]]\!](G)(A)(A')(E) \coloneqq\\
&\qquad [\![\Gamma \vdash f : \Pi(x : A)\, C[x, x, \mathsf{refl}\, x]]\!](G)(A).
\end{aligned}$$

For propositional truncation, we define:

$$\begin{aligned}
&[\![\Gamma \vdash \|A\| : \mathsf{Prop}]\!](G) \coloneqq \|[\![\Gamma \vdash A : \mathcal{A}]\!](G)\|,\\
&[\![\Gamma \vdash |a| : \|A\|]\!](G) \coloneqq *,\\
&[\![\Gamma \vdash \mathsf{ind}^{\|\cdot\|}_C\, f\, h : \Pi(t : \|A\|)\, C[t]]\!](G)(T) \coloneqq [\![\Gamma \vdash f : \Pi(x : A)\, C[|x|]]\!](G)(A)\\
&\qquad \text{for any } A \in [\![\Gamma \vdash A : \mathsf{Type}]\!](G).
\end{aligned}$$

For quotient types, we define:

$$\begin{aligned}
&[\![\Gamma \vdash A/R : \mathcal{A}]\!](G) \coloneqq [\![\Gamma \vdash A : \mathcal{A}]\!](G)/[\![\Gamma, x : A, x' : A \vdash R[x, x'] : \mathcal{B}]\!](G),\\
&[\![\Gamma \vdash [a]_R : A/R]\!](G) \coloneqq [[\![\Gamma \vdash a : A]\!](G)],\\
&[\![\Gamma \vdash \mathsf{ind}^{\|\cdot\|}_C\, f\, h : \Pi(t : \|A\|)\, C[t]]\!](G)(Q) \coloneqq [\![\Gamma \vdash f : \Pi(x : A)\, C[[x]_R]]\!](G)(A)\\
&\qquad \text{for any } A \in Q.
\end{aligned}$$

For propositional extensionality, we define:

$$[\![\vdash \mathsf{propext} : \Pi(P, P' : \mathsf{Prop})\, ((P \leftrightarrow P) \to (P =_{\mathsf{Prop}} P'))]\!](S)(S')(F) \coloneqq *.$$

We can see with induction that these interpretations are well-defined, so in particular that every function $[\![\Gamma \vdash a : A]\!]$ is indeed tracked by a natural number and therefore a morphism.

C De Jongh's Theorem for HAH

▶ **Corollary** (De Jongh's theorem for HAH). *Let $A[P_0, \dots, P_{n-1}]$ be a propositional formula with propositional variables $P_0, \dots, P_{n-1}$. If A is not provable in intuitionistic propositional logic then we can construct sentences $B_0, \dots, B_{n-1}$ in the language of* HAH *such that $A[B_0, \dots, B_{n-1}]$ is not provable in* HAH.

Proof. Firstly, every higher-order arithmetical formula can be seen as a first-order formula in the language of set theory by interpreting $\exists x^n$ as $\exists(x \in \mathcal{P}^n(\omega))$ and $\forall x^n$ as $\forall(x \in \mathcal{P}^n(\omega))$. IZF proves the axioms of HAH so we can view HAH as a subtheory of IZF. Now, in the proof of De Jongh's theorem for IZF, Passmann [32] constructs suitable $B_0, \dots, B_{n-1}$ of the form:

$$\bigvee_k (\Gamma_k \wedge \bigwedge_l \neg(\neg\Delta_l \wedge \Delta_{l+1})),$$

where Γ_k and Δ_l are roughly the following set theoretic formulas:

$$\Gamma_k \coloneqq (|\mathcal{P}(1)| < k), \qquad \Delta_l \coloneqq (|\mathcal{P}(\aleph_0)| < \aleph_l).$$

More precisely, the formula Γ_k can be stated in the language of HAH as follows:

$$\Gamma_k \coloneqq \forall X_0^1 \cdots \forall X_{k-1}^1 \, (\textstyle\bigwedge_{i<k}(\forall y^0(y \in X_i \to y = 0)) \to \bigvee_{i<j<k}(x_i = x_j)).$$

Note that Γ_k is not trivial in constructive set theory because we cannot prove for every set of the form $\{x \in 1 \mid A\}$ that it equal to $0 = \emptyset$ or $1 = \{\emptyset\}$. For Δ_l we can take any of the equivalent definitions for the statement $|\mathcal{P}(\aleph_0)| < \aleph_l$ in ZFC. One possible definition of Δ_l in the language of HAH is the following:

$$\Delta_l \coloneqq \forall \mathcal{X}_0^2 \cdots \forall \mathcal{X}_l^2 \, (\textstyle\bigwedge_{i<l+1} \mathsf{is\text{-}infinite}(\mathcal{X}_i) \to \bigvee_{i<j<l+1}(|\mathcal{X}_i| = |\mathcal{X}_j|)).$$

Note that the $\mathcal{X}_i$ are of level 2 so in IZF they will be interpreted as elements of $\mathcal{P}^2(\omega)$ which are subsets of $\mathcal{P}(\omega)$. So Δ_l states that for any $l+1$ infinite subsets of $\mathcal{P}(\mathbb{N})$ there must be two that have the same cardinality. This means that we have at most l infinite subsets of $\mathcal{P}(\mathbb{N})$ with distinct cardinalities, in which case we would have $\omega = \aleph_0, \dots, \aleph_{l-1} = \mathcal{P}(\omega)$. Here we make use of the following definitions:

$$\begin{aligned} |X^{n+1}| = |Y^{n+1}| &\coloneqq \exists Z^{n+1} \, (\forall(x \in X) \, \exists!(y \in Y) \, (\langle x, y\rangle \in Z) \wedge \\ &\qquad \forall(y \in Y) \, \exists!(x \in X) \, (\langle x, y\rangle \in Z)), \\ \mathsf{is\text{-}infinite}(X^{n+1}) &\coloneqq \exists Y^{n+1} \, (\exists(x \in X) \, (x \notin Y) \wedge \forall(y \in Y) \, (y \in X) \wedge |X| = |Y|). \end{aligned}$$

Note that we use Dedekinds definition of infinity because it is easier to state in the language of HAH. It is equivalent to the usual notion of infinity in ZFC. Now, suppose that we have a propositional formula $A[P_0, \dots, P_{n-1}]$ that is not provable in intuitionistic logic. Passmann shows that there are $B_0, \dots, B_{n-1}$ such that $A[B_0, \dots, B_{n-1}]$ is not provable in IZF. But we can view $B_0, \dots, B_{n-1}$ as HAH-formulas and then $A[B_0, \dots, B_{n-1}]$ is certainly not provable in HAH because we can view HAH as a subtheory of IZF. ◀

A Generic Characterization of Generalized Unary Temporal Logic and Two-Variable First-Order Logic

Thomas Place
Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, F-33400, Talence, France

Marc Zeitoun
Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, F-33400, Talence, France

Abstract

We study an operator on classes of languages. For each class $\mathcal{C}$, it produces a new class $\mathrm{FO}^2(\mathbb{I}_{\mathcal{C}})$ associated with a variant of two-variable first-order logic equipped with a signature $\mathbb{I}_{\mathcal{C}}$ built from $\mathcal{C}$. For $\mathcal{C} = \{\emptyset, A^*\}$, we obtain the usual $\mathrm{FO}^2(<)$ logic, equipped with linear order. For $\mathcal{C} = \{\emptyset, \{\varepsilon\}, A^+, A^*\}$, we get the variant $\mathrm{FO}^2(<, +1)$, which also includes the successor predicate. If $\mathcal{C}$ consists of all Boolean combinations of languages A^*aA^*, where a is a letter, we get the variant $\mathrm{FO}^2(<, Bet)$, which includes "between" relations. We prove a *generic* algebraic characterization of the classes $\mathrm{FO}^2(\mathbb{I}_{\mathcal{C}})$. It elegantly generalizes those known for all the cases mentioned above. Moreover, it implies that if $\mathcal{C}$ has decidable separation (plus some standard properties), then $\mathrm{FO}^2(\mathbb{I}_{\mathcal{C}})$ has a decidable membership problem.

We actually work with an equivalent definition of $\mathrm{FO}^2(\mathbb{I}_{\mathcal{C}})$ in terms of *unary temporal logic*. For each class $\mathcal{C}$, we consider a variant $\mathrm{TL}(\mathcal{C})$ of unary temporal logic whose future/past modalities depend on $\mathcal{C}$ and such that $\mathrm{TL}(\mathcal{C}) = \mathrm{FO}^2(\mathbb{I}_{\mathcal{C}})$. Finally, we also characterize $\mathrm{FL}(\mathcal{C})$ and $\mathrm{PL}(\mathcal{C})$, the pure-future and pure-past restrictions of $\mathrm{TL}(\mathcal{C})$. Like for $\mathrm{TL}(\mathcal{C})$, these characterizations imply that if $\mathcal{C}$ is a class with decidable separation, then $\mathrm{FL}(\mathcal{C})$ and $\mathrm{PL}(\mathcal{C})$ have decidable membership.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory

Keywords and phrases Classes of regular languages, Generalized unary temporal logic, Generalized two-variable first-order logic, Generic decidable characterizations, Membership, Separation

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.45

Related Version *Extended Version*: `https://arxiv.org/abs/2307.09349` [29]

1 Introduction

Context. Regular languages of finite words form a robust class: they admit a wide variety of equivalent definitions, whether by regular expressions, finite automata, finite monoids or monadic second-order logic. It is therefore natural to study the *fragments* of regular languages obtained by restricting the syntax of one of the above-mentioned formalisms. For each particular fragment, we seek to prove that it has a decidable *membership problem*: given a regular language as input, decide whether it belongs to the fragment. Intuitively, doing so requires a thorough knowledge of the fragment and the languages it can describe.

This approach was initiated by Schützenberger [30] for the class SF of star-free languages. These are the languages defined by a *star-free expression*: a regular expression without Kleene star but with complement instead. Equivalently, these are the languages that can be defined in first-order logic with the linear order [16] ($\mathrm{FO}(<)$) or in linear temporal logic [11] (LTL). Schützenberger established an algebraic characterization of SF: a regular language is star-free if and only if its *syntactic monoid is aperiodic*. This yields a membership procedure for SF because the syntactic monoid can be computed and aperiodicity is a decidable property.

Operators. This seminal result prompted researchers to look at other natural classes, spawning a fruitful line of research (see *e.g.*, [4, 33, 12, 18, 36, 24]). Although there are numerous classes, they can be grouped into families based on "variants" of the same syntax. Let us

use logic to clarify this point. Each logical fragment can use several *signatures* (*i.e.*, sets of predicates allowed in formulas), each giving rise to a class. For instance, first-order logic is commonly equipped with predicates such as the linear order "$<$" [16, 30], the successor "$+1$" [3] or the modular predicates "MOD" [2]. While it is worth looking at multiple variants of prominent classes, doing so individually for each of them has an obvious disadvantage: the proof has to be systematically modified to accommodate each change. This can be tedious, difficult, and not necessarily enlightening. To overcome this drawback, a natural approach is to capture a whole family of variants with an *operator*. An operator "Op" takes a class $\mathcal{C}$ as input, and outputs a larger one $\mathrm{Op}(\mathcal{C})$. Thus, we can study all classes $\mathrm{Op}(\mathcal{C})$ *simultaneously*: the question becomes: "what hypotheses about $\mathcal{C}$ guarantee the decidability of $\mathrm{Op}(\mathcal{C})$-membership?". For example, one can generalize the three definitions of star-free languages through operators:

1. The *star-free closure* $\mathcal{C} \mapsto \mathrm{SF}(\mathcal{C})$ has been introduced in [31, 34]. Languages in $\mathrm{SF}(\mathcal{C})$ are defined by "extended" star-free expressions, which can freely use languages from $\mathcal{C}$.
2. A construction associating a signature $\mathbb{I}_{\mathcal{C}}$ to a class $\mathcal{C}$ has been given in [23]. For each $L \in \mathcal{C}$, the set $\mathbb{I}_{\mathcal{C}}$ contains a binary predicate $I_L(x, y)$: for a word w and two positions i, j in w, $I_L(i, j)$ holds if and only if $i < j$ and the infix of w between i and j belongs to L. We get an operator $\mathcal{C} \mapsto \mathrm{FO}(\mathbb{I}_{\mathcal{C}})$ based on first-order logic. It captures many choices of signature.
3. Similarly, an operator $\mathcal{C} \mapsto \mathrm{LTL}(\mathcal{C})$ that generalizes LTL has been defined in [28].

It is shown in [23, 28] that $\mathrm{SF}(\mathcal{C}) = \mathrm{FO}(\mathbb{I}_{\mathcal{C}}) = \mathrm{LTL}(\mathcal{C})$ for any class $\mathcal{C}$ (with mild hypotheses). Moreover, a *generic* algebraic characterization is proved in [25, 28]. Given a regular language L, it relies on a construction that identifies monoids inside its syntactic monoid, called the *$\mathcal{C}$-orbits*: $L \in \mathrm{SF}(\mathcal{C})$ if and only if its $\mathcal{C}$-orbits are all *aperiodic*. This elegantly generalizes Schützenberger's theorem and gives a *transfer theorem* for membership. Indeed, the $\mathcal{C}$-orbits are connected with a decision problem that strengthens membership: *$\mathcal{C}$-separation*. Given *two* input regular languages L_1 and L_2, $\mathcal{C}$-separation asks whether there is $K \in \mathcal{C}$ such that $L_1 \subseteq K$ and $L_2 \cap K = \emptyset$. The crucial point is that $\mathcal{C}$-orbits are computable if $\mathcal{C}$-separation is decidable. Thus, $\mathrm{SF}(\mathcal{C})$-membership is also decidable in this case. Similar results are known for other operators such as *polynomial closure* [23] or its *unambiguous* restriction [22, 27].

Unary temporal logic and two-variable first-order logic. The operator we investigate generalizes another important class admitting multiple definitions [35, 6] (see [8, 7] for extensions). We are interested in two of them. It consists of languages that can be defined in *two-variable first-order logic* with the linear order ($\mathrm{FO}^2(<)$) or equivalently in *unary temporal logic* (TL) with the modalities F (sometimes in the future) and P (sometimes in the past). Etessami, Vardi and Wilke [9] have shown that $\mathrm{FO}^2(<) = \mathrm{TL}$. Its algebraic characterization by Thérien and Wilke [36] is one of the famous results of this type: a regular language belongs to $\mathrm{FO}^2(<) = \mathrm{TL}$ if and only if its syntactic monoid belongs to the variety of monoids DA.

Both definitions extend to natural operators. First, the generic signatures $\mathbb{I}_{\mathcal{C}}$ yield an operator $\mathcal{C} \mapsto \mathrm{FO}^2(\mathbb{I}_{\mathcal{C}})$ based on FO^2. Second, an operator $\mathcal{C} \mapsto \mathrm{TL}(\mathcal{C})$ has been defined in [27]. It enriches TL with new modalities F_L and P_L, both depending on the languages $L \in \mathcal{C}$. For example, the formula $\mathrm{F}_L\, \varphi$ holds at a position i in a word w if there is a position $j > i$ in w such that φ holds at j and the infix between i and j belongs to L. We know that $\mathrm{FO}^2(\mathbb{I}_{\mathcal{C}}) = \mathrm{TL}(\mathcal{C})$ when $\mathcal{C}$ is closed under Boolean operations [27]. Here, we work with the $\mathrm{TL}(\cdot)$ operator, which encompasses all classic classes based on two-variable first-order logic or unary temporal logic. This includes the original variants $\mathrm{FO}^2(<) = \mathrm{TL}$ and $\mathrm{FO}^2(<, +1) = \mathrm{TLX}$, both of which were studied by Thérien and Wilke [36] (here, "$+1$" is the successor predicate and TLX is defined by enriching TL with "next" and "yesterday" modalities). Another example is the variant

$\mathrm{FO}^2(<, MOD)$ endowed with *modular predicates*, investigated by Dartois and Paperman [5]. Finally, we capture the variant $\mathrm{FO}^2(<, Bet) = \mathrm{BInvTL}$ equipped with "*between*" relations, defined and characterized by Krebs, Lodaya, Pandya and Straubing [13, 14, 15].

Contributions. We prove a *generic* algebraic characterization of the classes $\mathrm{FO}^2(\mathbb{I}_{\mathcal{C}}) = \mathrm{TL}(\mathcal{C})$. We reuse the $\mathcal{C}$-orbits introduced for star-free closure: for any class $\mathcal{C}$ (having mild properties) we show that a regular language belongs to $\mathrm{TL}(\mathcal{C})$ if and only if all $\mathcal{C}$-orbits of its syntactic monoid belong to DA. In particular, this yields a *transfer theorem* for membership: if $\mathcal{C}$ has decidable *separation*, then $\mathrm{FO}^2(\mathbb{I}_{\mathcal{C}}) = \mathrm{TL}(\mathcal{C})$ has decidable membership. Moreover, this characterization generalizes the characterizations known for all the above instances.

A key feature of our proof is that we use a third auxiliary operator. It combines two other operators: Boolean polynomial closure (BPol) and unambiguous polynomial closure (UPol). We have $\mathrm{UPol}(\mathrm{BPol}(\mathcal{C})) \subseteq \mathrm{TL}(\mathcal{C})$ if $\mathcal{C}$ has mild properties [27]. In fact, for many natural classes, $\mathrm{UPol}(\mathrm{BPol}(\mathcal{C})) = \mathrm{TL}(\mathcal{C})$. For example, $\mathrm{UPol}(\mathrm{BPol}(\{\emptyset, A^*\}))$ is the class UL of *unambiguous languages* defined by Schützenberger [32]. It is known [36] that $\mathrm{UL} = \mathrm{TL} = \mathrm{FO}^2(<)$. More generally, $\mathrm{UPol}(\mathrm{BPol}(\mathcal{C})) = \mathrm{TL}(\mathcal{C})$ for every class $\mathcal{C}$ consisting of *group languages* [27]. Yet, this is a strong hypothesis and the inclusion $\mathrm{UPol}(\mathrm{BPol}(\mathcal{C})) \subseteq \mathrm{TL}(\mathcal{C})$ is *strict* in general. For example, the results of [15] imply that $\mathrm{UPol}(\mathrm{BPol}(\mathrm{AT})) \neq \mathrm{TL}(\mathrm{AT})$, where AT consists of all Boolean combinations of languages A^*aA^* (with $a \in A$). Nevertheless, the classes $\mathrm{UPol}(\mathrm{BPol}(\mathcal{C}))$ serve as a central ingredient in the most difficult direction of our proof: "*If a language satisfies our characterization on $\mathcal{C}$-orbits, prove that it belongs to* $\mathrm{TL}(\mathcal{C})$". More precisely, we exploit the known characterization of $\mathrm{UPol}(\mathrm{BPol}(\mathcal{C}))$ to prove that auxiliary languages belong to this class, and we then conclude using the inclusion $\mathrm{UPol}(\mathrm{BPol}(\mathcal{C})) \subseteq \mathrm{TL}(\mathcal{C})$.

Finally, we look at two additional operators: $\mathcal{C} \mapsto \mathrm{FL}(\mathcal{C})$ and $\mathcal{C} \mapsto \mathrm{PL}(\mathcal{C})$. They are also defined in terms of unary temporal logic, as the pure-future and the pure-past restrictions of $\mathcal{C} \mapsto \mathrm{TL}(\mathcal{C})$. We present generic algebraic characterizations for these two operators as well. Again, they are based on $\mathcal{C}$-orbits. For every class $\mathcal{C}$ (with mild hypotheses), we show that a regular language belongs to $\mathrm{FL}(\mathcal{C})$ (resp. $\mathrm{PL}(\mathcal{C})$) if and only if all the $\mathcal{C}$-orbits inside its syntactic monoid are $\mathcal{L}$-trivial (resp. $\mathcal{R}$-trivial) monoids. As before, these results yield *transfer theorems*: if $\mathcal{C}$ has decidable *separation*, then $\mathrm{FL}(\mathcal{C})$ and $\mathrm{PL}(\mathcal{C})$ have decidable membership.

Organization of the paper. We recall the notation and background in Section 2. In Section 3, we present the $\mathcal{C}$-orbits and their properties. In Section 4, we define the operator $\mathcal{C} \mapsto \mathrm{TL}(\mathcal{C})$. Section 5 is devoted to the generic characterization of $\mathrm{TL}(\mathcal{C})$ and to its proof. In Section 6, finally, we state the characterizations of the pure-future and pure-past restrictions of $\mathrm{TL}(\mathcal{C})$.

2 Preliminaries

We fix a finite *alphabet* A for the paper. As usual, A^* denotes the set of all finite words over A, including the empty word ε. A *language* is a subset of A^*. We let $A^+ = A^* \setminus \{\varepsilon\}$. For $u, v \in A^*$, we write uv for the word obtained by concatenating u and v. We lift the concatenation to languages as follows: if $K, L \subseteq A^*$, we let $KL = \{uv \mid u \in K, v \in L\}$. If $w \in A^*$, we write $|w| \in \mathbb{N}$ for its length. A word $w = a_1 \cdots a_{|w|} \in A^*$ is viewed as an *ordered set* $\mathsf{Pos}(w) = \{0, 1, \ldots, |w|, |w|+1\}$ *of* $|w|+2$ *positions*. In addition, we let $\mathsf{Pos}_c(w) = \{1, \ldots, |w|\} \subsetneq \mathsf{Pos}(w)$. Position $i \in \mathsf{Pos}_c(w)$ carries label $a_i \in A$, which we write $w[i] = a_i$. On the other hand, positions 0 and $|w|+1$ carry *no label*. We write $w[0] = min$ and $w[|w|+1] = max$. For $v, w \in A^*$, we say that v is an *infix* (resp. *prefix*, *suffix*) of w when there exist $x, y \in A^*$ such that $w = xvy$ (resp. $w = vy$, $w = xv$). Given a word $w = a_1 \cdots a_{|w|} \in A^*$ and $i, j \in \mathsf{Pos}(w)$ such that $i < j$, we write $w(i, j) = a_{i+1} \cdots a_{j-1} \in A^*$ (*i.e.*, the infix obtained by keeping the letters carried by positions *strictly* between i and j).

Classes. A *class* of languages $\mathcal{C}$ is simply a set of languages. Such a class $\mathcal{C}$ is a *lattice* when $\emptyset \in \mathcal{C}$, $A^* \in \mathcal{C}$ and $\mathcal{C}$ is closed under both union and intersection: for all $K, L \in \mathcal{C}$, we have $K \cup L \in \mathcal{C}$ and $K \cap L \in \mathcal{C}$. Moreover, a class of languages $\mathcal{C}$ is a *Boolean algebra* if it is a lattice closed under complement: for all $L \in \mathcal{C}$, we have $A^* \setminus L \in \mathcal{C}$. Finally, the class $\mathcal{C}$ is *closed under quotients* if for all $L \in \mathcal{C}$ and $u \in A^*$, we have $u^{-1}L \stackrel{\text{def}}{=} \{w \in A^* \mid uw \in L\} \in \mathcal{C}$ and $Lu^{-1} \stackrel{\text{def}}{=} \{w \in A^* \mid wu \in L\} \in \mathcal{C}$. A *prevariety* is a Boolean algebra closed under quotients and containing only *regular languages*. Regular languages are those which can be equivalently defined by finite automata, finite monoids or monadic second-order logic. We work with the definition by monoids, which we now recall.

Monoids. A *monoid* is a set M endowed with an associative multiplication $(s,t) \mapsto st$ having an identity element 1_M (*i.e.*, such that $1_M s = s1_M = s$ for every $s \in M$). An *idempotent* of a monoid M is an element $e \in M$ such that $ee = e$. We write $E(M) \subseteq M$ for the set of all idempotents in M. It is folklore that for every *finite* monoid M, there exists a natural number $\omega(M)$ (denoted by ω when M is understood) such that for every $s \in M$, the element s^ω is an idempotent. Finally, we shall use the following Green relations [10] defined on monoids. Given a monoid M and $s, t \in M$, we write:

$s \leqslant_{\mathcal{J}} t$ when there exist $x, y \in M$ such that $s = xty$,
$s \leqslant_{\mathcal{L}} t$ when there exists $x \in M$ such that $s = xt$,
$s \leqslant_{\mathcal{R}} t$ when there exists $y \in M$ such that $s = ty$.

Clearly, $\leqslant_{\mathcal{J}}$, $\leqslant_{\mathcal{L}}$ and $\leqslant_{\mathcal{R}}$ are preorders (*i.e.*, they are reflexive and transitive). We write $<_{\mathcal{J}}$, $<_{\mathcal{L}}$ and $<_{\mathcal{R}}$ for their strict variants (for example, $s <_{\mathcal{J}} t$ when $s \leqslant_{\mathcal{J}} t$ but $t \not\leqslant_{\mathcal{J}} s$). Finally, we write $\mathcal{J}$, $\mathcal{L}$ and $\mathcal{R}$ for the corresponding equivalence relations (for example, $s \mathcal{J} t$ when $s \leqslant_{\mathcal{J}} t$ and $t \leqslant_{\mathcal{J}} s$). There are many technical results about Green relations. We will just need the following easy and standard lemma, which applies to *finite* monoids (see *e.g.*, [17, 20]).

▶ **Lemma 1.** *Let M be a finite monoid and let $s, t \in M$. If $s \mathcal{J} t$ and $s \leqslant_{\mathcal{R}} t$, then $s \mathcal{R} t$.*

Regular languages and syntactic morphisms. Since A^* is a monoid whose multiplication is concatenation (the identity element is ε), we may consider monoid morphisms $\alpha : A^* \to M$ where M is an arbitrary monoid. That is, $\alpha : A^* \to M$ is a map satisfying $\alpha(\varepsilon) = 1_M$ and $\alpha(uv) = \alpha(u)\alpha(v)$ for all $u, v \in A^*$. We say that a language $L \subseteq A^*$ is *recognized* by α when there exists a set $F \subseteq M$ such that $L = \alpha^{-1}(F)$.

It is well known that regular languages are exactly those recognized by a morphism into a *finite* monoid. Moreover, every language L is recognized by a canonical morphism, which we briefly recall. One can associate to L an equivalence $\equiv_L$ over A^*: the *syntactic congruence of L*. Given $u, v \in A^*$, we let $u \equiv_L v$ if and only if $xuy \in L \Leftrightarrow xvy \in L$ for every $x, y \in A^*$. One can check that "$\equiv_L$" is indeed a congruence on A^*: it is an equivalence compatible with word concatenation. Thus, the set of equivalence classes $M_L = A^*/\equiv_L$ is a monoid. It is called the *syntactic monoid of L*. Finally, the map $\alpha_L : A^* \to M_L$ sending every word to its equivalence class is a morphism recognizing L, called the *syntactic morphism of L*. It is known that a language L is regular if and only if M_L is finite (*i.e.*, $\equiv_L$ has finite index): this is the Myhill-Nerode theorem. In this case, one can compute the syntactic morphism $\alpha_L : A^* \to M_L$ from any representation of L (such as an automaton or a monoid morphism).

Decision problems. We consider two decision problems, both depending on an arbitrary class $\mathcal{C}$. They serve as mathematical tools for analyzing it, as obtaining an algorithm for one of these problems requires a solid understanding of that class $\mathcal{C}$. The *$\mathcal{C}$-membership problem*

is the simplest: it takes as input a single regular language L and simply asks whether $L \in \mathcal{C}$. The second problem, *$\mathcal{C}$-separation*, is more general. Given three languages K, L_1, L_2, we say that K *separates* L_1 from L_2 if $L_1 \subseteq K$ and $L_2 \cap K = \emptyset$. Given a class $\mathcal{C}$, we say that L_1 is *$\mathcal{C}$-separable* from L_2 if some language of $\mathcal{C}$ separates L_1 from L_2. The *$\mathcal{C}$-separation problem* takes as input two regular languages L_1, L_2 and asks whether L_1 is $\mathcal{C}$-separable from L_2.

▶ Remark 2. The $\mathcal{C}$-separation problem generalizes $\mathcal{C}$-membership. Indeed, a regular language belongs to $\mathcal{C}$ if and only if it is $\mathcal{C}$-separable from its complement, which is regular.

3 Orbits

Instead of looking at single classes, we consider *operators*. These are correspondences $\mathcal{C} \mapsto \mathrm{Op}(\mathcal{C})$ that take as input a class $\mathcal{C}$ to build a new one $\mathrm{Op}(\mathcal{C})$. We investigate three operators in Sections 4 to 6. For now, we present general tools for handling such operators. Given a class $\mathcal{C}$ and a morphism $\alpha : A^* \to M$, we define special subsets of M: the $\mathcal{C}$-orbits for α. This notion was introduced in [28]. We shall use it to formulate *generic characterizations* of the operators $\mathcal{C} \mapsto \mathrm{Op}(\mathcal{C})$ that we consider: for each input prevariety $\mathcal{C}$, the languages in $\mathrm{Op}(\mathcal{C})$ are characterized by a property of the $\mathcal{C}$-orbits for their syntactic morphisms.

$\mathcal{C}$-pairs. Consider a class $\mathcal{C}$ and a morphism $\alpha : A^* \to M$. We say that a pair $(s,t) \in M^2$ is a *$\mathcal{C}$-pair for α* if and only if $\alpha^{-1}(s)$ is *not* $\mathcal{C}$-separable from $\alpha^{-1}(t)$. Note that if $\mathcal{C}$-separation is decidable, then one can compute all $\mathcal{C}$-pairs for an input morphism.

We turn to a useful technical result, which characterizes the $\mathcal{C}$-pairs using morphisms. Consider *two* morphisms $\alpha : A^* \to M$ and $\eta : A^* \to N$. For every pair $(s,t) \in M^2$, we say that (s,t) is an *η-pair for α* when there exist $u, v \in A^*$ such that $\eta(u) = \eta(v)$, $\alpha(u) = s$ and $\alpha(v) = t$. In addition, for each class $\mathcal{C}$, we define the *$\mathcal{C}$-morphisms* as the *surjective* morphisms $\eta : A^* \to N$ into a finite monoid N such that all languages recognized by η belong to $\mathcal{C}$. We have the following elementary lemma, proved in [27, Lemma 5.11].

▶ **Lemma 3.** *Let $\mathcal{C}$ be a prevariety and $\alpha : A^* \to M$ be a morphism. Then,*

1. *For every $\mathcal{C}$-morphism $\eta : A^* \to N$, all $\mathcal{C}$-pairs for α are also η-pairs for α.*
2. *There exists a $\mathcal{C}$-morphism $\eta : A^* \to N$ such that all η-pairs for α are also $\mathcal{C}$-pairs for α.*

$\mathcal{C}$-orbits. Consider a class $\mathcal{C}$ and a morphism $\alpha : A^* \to M$. For every *idempotent* $e \in E(M)$, the *$\mathcal{C}$-orbit of e for α* is the set $M_e \subseteq M$ consisting of all elements $ete \in M$ such that $(e,t) \in M^2$ is a $\mathcal{C}$-pair. If $\mathcal{C}$ is a prevariety and α is surjective, it is proved in [28, Lemma 5.5] that M_e is a monoid in M: it is closed under multiplication and $e \in M_e$ is its identity. On the other hand, M_e is not a "*sub*monoid" of M (this is because 1_M needs not belong to M_e).

▶ **Lemma 4.** *Let $\mathcal{C}$ be a prevariety and $\alpha : A^* \to M$ be a surjective morphism into a finite monoid. For all $e \in E(M)$, the $\mathcal{C}$-orbit of e for α is a monoid in M whose identity is e.*

As seen above, when $\mathcal{C}$ has decidable separation, one can compute the $\mathcal{C}$-pairs associated with an input morphism. Hence, one can also compute the $\mathcal{C}$-orbits in this case.

▶ **Lemma 5.** *Let $\mathcal{C}$ be a class with decidable separation. Given as input a morphism $\alpha : A^* \to M$ into a finite monoid and $e \in E(M)$, one can compute the $\mathcal{C}$-orbit of e for α.*

Finally, the following lemma connects $\mathcal{C}$-orbits with $\mathcal{C}$-morphisms.

▶ **Lemma 6.** *Let $\mathcal{C}$ be a prevariety and $\alpha : A^* \to M$ be a morphism. Moreover, let $\eta : A^* \to N$ be a $\mathcal{C}$-morphism. For every $e \in E(M)$, there exists $f \in E(N)$ such that the $\mathcal{C}$-orbit of e for α is contained in the set $\alpha(\eta^{-1}(f))$.*

Proof. Let $t_1, \ldots, t_n \in M$ be all elements of the set $\{t \in M \mid (e,t) \text{ is a } \mathcal{C}\text{-pair}\}$. By definition, the $\mathcal{C}$-orbit of e for α is $M_e = \{et_1e, \ldots, et_ne\}$. Since η is a $\mathcal{C}$-morphism, Lemma 3 implies that (e, t_i) is an η-pair for all $i \leq n$. This yields $x_i, y_i \in A^*$ such that $\eta(x_i) = \eta(y_i)$, $\alpha(x_i) = e$ and $\alpha(y_i) = t_i$. Let $p = \omega(N)$, $w = (x_1 \cdots x_n)^p$ and $f = \eta(w)$. Note that f is idempotent by choice of p. We show that $et_ie \in \alpha(\eta^{-1}(f))$ for $i \leq n$. We define $w_i = (x_1 \cdots x_n)^p x_1 \cdots x_{i-1} y_i x_{i+1} \cdots x_n (x_1 \cdots x_n)^{2p-1}$. By definition, we have $\alpha(w_i) = et_ie$. Now, since $\eta(x_i) = \eta(y_i)$, we get $\eta(w_i) = \eta(w) = f$. Hence, $et_ie \in \alpha(\eta^{-1}(f))$, as desired. ◀

4 Generalized unary temporal logic

In this section, we define generalized unary temporal logic. We introduce an *operator* $\mathcal{C} \mapsto \mathrm{TL}(\mathcal{C})$ that associates a new class of languages $\mathrm{TL}(\mathcal{C})$ with every input class $\mathcal{C}$. We first recall its definition (taken from [27]), and we then complete it with useful properties.

4.1 Definition

Syntax. We associate with any class $\mathcal{C}$ a set of temporal formulas denoted by $\mathrm{TL}[\mathcal{C}]$ as follows. A $\mathrm{TL}[\mathcal{C}]$ formula is built from atomic formulas using Boolean connectives and temporal operators. The atomic formulas are $\top$, $\bot$, min, max and "a" for every letter $a \in A$. All Boolean connectives are allowed: if ψ_1 and ψ_2 are $\mathrm{TL}[\mathcal{C}]$ formulas, then so are $(\psi_1 \vee \psi_2)$, $(\psi_1 \wedge \psi_2)$ and $(\neg\psi_1)$. We associate *two temporal modalities* with every language $L \in \mathcal{C}$, which we denote by F_L and P_L: if ψ is a $\mathrm{TL}[\mathcal{C}]$ formula, then so are $(\mathrm{F}_L\, \psi)$ and $(\mathrm{P}_L\, \psi)$. For the sake of improved readability, we omit parentheses when there is no ambiguity.

Semantics. Evaluating a $\mathrm{TL}[\mathcal{C}]$ formula φ requires a word $w \in A^*$ and a position $i \in \mathsf{Pos}(w)$. We define by induction what it means for (w, i) *to satisfy* φ, which one denotes by $w, i \models \varphi$.

- **Atomic formulas:** $w, i \models \top$ always holds, $w, i \models \bot$ never holds and for every symbol $\ell \in A \cup \{min, max\}$, $w, i \models \ell$ holds when $\ell = w[i]$.
- **Disjunction:** $w, i \models \psi_1 \vee \psi_2$ when $w, i \models \psi_1$ or $w, i \models \psi_2$.
- **Conjunction:** $w, i \models \psi_1 \wedge \psi_2$ when $w, i \models \psi_1$ and $w, i \models \psi_2$.
- **Negation:** $w, i \models \neg\psi$ when $w, i \models \psi$ does not hold.
- **Finally:** for $L \in \mathcal{C}$, we let $w, i \models \mathrm{F}_L\, \psi$ when there exists $j \in \mathsf{Pos}(w)$ such that $i < j$, $w(i, j) \in L$ and $w, j \models \psi$.
- **Previously:** for $L \in \mathcal{C}$, we let $w, i \models \mathrm{P}_L\, \psi$ when there exists $j \in \mathsf{Pos}(w)$ such that $j < i$, $w(j, i) \in L$ and $w, j \models \psi$.

When no distinguished position is specified, it is customary to evaluate formulas at the *leftmost* unlabeled position. One could also consider the symmetrical convention of evaluating formulas at the *rightmost* unlabeled position. The convention chosen does not matter: we end-up with the same class of languages. However, we shall consider restrictions of $\mathrm{TL}[\mathcal{C}]$ for which this choice *does* matter. This is why we introduce notations for both conventions. Given a formula $\varphi \in \mathrm{TL}[\mathcal{C}]$ we let $L_{min}(\varphi) = \{w \in A^* \mid w, 0 \models \varphi\}$ and $L_{max}(\varphi) = \{w \in A^* \mid w, |w| + 1 \models \varphi\}$.

We are now ready to define the operator $\mathcal{C} \mapsto \mathrm{TL}(\mathcal{C})$. Consider an arbitrary class $\mathcal{C}$. We write $\mathrm{TL}(\mathcal{C})$ for the class consisting of all languages $L_{min}(\varphi)$ where $\varphi \in \mathrm{TL}[\mathcal{C}]$. Observe that by definition, $\mathrm{TL}(\mathcal{C})$ is a Boolean algebra. Actually, the results of [27] imply that when $\mathcal{C}$ is a prevariety, then so is $\mathrm{TL}(\mathcal{C})$ (we do not need this fact in the present paper).

Classic unary temporal logic. Let $\mathrm{ST} = \{\emptyset, A^*\}$ and $\mathrm{DD} = \{\emptyset, \{\varepsilon\}, A^+, A^*\}$. The modalities F_{A^*} and P_{A^*} have the same semantics as the modalities F and P of standard unary temporal logic – *e.g.*, $w, i \models \mathrm{F}\,\varphi$ when there exists $j \in \mathsf{Pos}(w)$ such that $i < j$ and $w, j \models \varphi$. Similarly, the modalities $\mathrm{F}_{\{\varepsilon\}}$ and $\mathrm{P}_{\{\varepsilon\}}$ have the same semantics as the modalities X (next) and Y (yesterday) – *e.g.*, $w, i \models \mathrm{X}\,\varphi$ when $i+1 \in \mathsf{Pos}(w)$ and $w, i+1 \models \varphi$. Using these facts, one can check that the classes TL(ST) and TL(DD) correspond exactly to the two original standard variants of unary temporal logic (see *e.g.*, [9]): we have $\mathrm{TL} = \mathrm{TL}(\mathrm{ST})$ and $\mathrm{TLX} = \mathrm{TL}(\mathrm{DD})$.

▶ Remark 7 (Robustness of classes to which TL is applied). Note that including $\emptyset$ in an input class does not bring any new modality in unary temporal logic. Similarly, the classes TL(DD) and $\mathrm{TL}(\mathrm{DD} \setminus \{A^+\})$ are identical. However, in order to use generic results such as those from Section 3, we require the classes to which the operator $\mathcal{C} \mapsto \mathrm{TL}(\mathcal{C})$ is applied to have robust properties: they should be prevarieties (hence, they should be closed under complement).

▶ Remark 8 (Connection with FO^2). Etessami, Vardi and Wilke [9] have shown that the variant TL corresponds to the class $\mathrm{FO}^2(<)$ (two-variable first-order logic equipped with the linear order), and that TLX corresponds to $\mathrm{FO}^2(<, +1)$ (which also allows the successor). In [27], these results are generalized to all classes $\mathrm{TL}(\mathcal{C})$ where $\mathcal{C}$ is a *Boolean algebra*. In this case, we can construct from $\mathcal{C}$ a set of predicates $\mathbb{I}_{\mathcal{C}}$ such that $\mathrm{TL}(\mathcal{C}) = \mathrm{FO}^2(\mathbb{I}_{\mathcal{C}})$.

▶ Remark 9. Another important input is the class AT of alphabet testable languages. It consists of all Boolean combinations of languages A^*aA^*, where $a \in A$ is a letter. The class TL(AT) has been studied by Krebs, Lodaya, Pandya and Straubing [13, 14, 15], who worked with the definition based on two-variable first-order logic (*i.e.*, with the class $\mathrm{FO}^2(\mathbb{I}_{\mathrm{AT}})$, see Remark 8). In particular, they proved that TL(AT) has decidable membership. We shall obtain this result as a corollary of our generic characterization of the classes $\mathrm{TL}(\mathcal{C})$.

4.2 Connection with unambiguous polynomial closure

It is shown in [27] that $\mathcal{C} \mapsto \mathrm{TL}(\mathcal{C})$ can be expressed by other operators for very specific inputs: prevarieties of *group languages*. If $\mathcal{G}$ is such a class, then $\mathrm{TL}(\mathcal{G})$ coincides with $\mathrm{UPol}(\mathrm{BPol}(\mathcal{G}))$, a class built on top of $\mathcal{G}$ with the two standard operators UPol and BPol. We do not use this result here, since we are tackling *arbitrary* input prevarieties, and in general, $\mathrm{UPol}(\mathrm{BPol}(\mathcal{C}))$ is *strictly* included in $\mathrm{TL}(\mathcal{C})$ (it follows from [15] that the inclusion is strict for the class AT of Remark 9). However, the operators UPol and BPol remain key tools in the paper: we use two results of [27] about them. Let us first briefly recall their definitions.

Given finitely many languages $L_0, \ldots, L_n \subseteq A^*$, a *marked product* of $L_0, \ldots, L_n$ is a product of the form $L_0a_1L_1 \cdots a_nL_n$ where $a_1, \ldots, a_n \in A$. A single language L_0 is a marked product (this is the case $n = 0$). The *polynomial closure* of a class $\mathcal{C}$, denoted by $\mathrm{Pol}(\mathcal{C})$, consists of all *finite unions* of marked products $L_0a_1L_1 \cdots a_nL_n$ such that $L_0, \ldots, L_n \in \mathcal{C}$. If $\mathcal{C}$ is a prevariety, then $\mathrm{Pol}(\mathcal{C})$ is a lattice (this is due to Arfi [1], see also [19, 23] for recent proofs). However, $\mathrm{Pol}(\mathcal{C})$ need not be closed under complement. This is why it is often combined with another operator: the Boolean closure of a class $\mathcal{D}$, denoted by $\mathrm{Bool}(\mathcal{D})$, is the least Boolean algebra containing $\mathcal{D}$. We write $\mathrm{BPol}(\mathcal{C})$ for $\mathrm{Bool}(\mathrm{Pol}(\mathcal{C}))$. It is standard that if $\mathcal{C}$ is a prevariety, then so is $\mathrm{BPol}(\mathcal{C})$ (see [23] for example). Finally, UPol is the *unambiguous* restriction of Pol. A marked product $L_0a_1L_1 \cdots a_nL_n$ is *unambiguous* when every word $w \in L_0a_1L_1 \cdots a_nL_n$ has a *unique* decomposition $w = w_0a_1w_1 \cdots a_nw_n$ where $w_i \in L_i$ for $0 \leq i \leq n$. The *unambiguous polynomial closure* of a class $\mathcal{C}$, written $\mathrm{UPol}(\mathcal{C})$, consists of all *finite disjoint unions* of *unambiguous marked products* $L_0a_1L_1 \cdots a_nL_n$ such that $L_0, \ldots, L_n \in \mathcal{C}$ (by "disjoint" we mean that the languages in the union must be pairwise disjoint). While this is not apparent on the definition, it is known [27] that if the input class $\mathcal{C}$ is a prevariety, then so is $\mathrm{UPol}(\mathcal{C})$. Thus, UPol preserves closure under complement.

In the paper, we are interested in the "combined" operator $\mathcal{C} \mapsto \mathrm{UPol}(\mathrm{BPol}(\mathcal{C}))$. Indeed, it is connected to the classes $\mathrm{TL}(\mathcal{C})$ by the following proposition proved in [27, Proposition 9.12].

▶ **Proposition 10.** *For every prevariety $\mathcal{C}$, we have* $\mathrm{UPol}(\mathrm{BPol}(\mathcal{C})) \subseteq \mathrm{TL}(\mathcal{C})$.

Although the inclusion of Proposition 10 is *strict* in general, it is essential for proving that *particular* languages belong to $\mathrm{TL}(\mathcal{C})$. Indeed, we will combine it with the next result [27, Theorem 6.7] to prove that languages belong to $\mathrm{UPol}(\mathrm{BPol}(\mathcal{C}))$ – and therefore to $\mathrm{TL}(\mathcal{C})$.

▶ **Theorem 11.** *Let $\mathcal{C}$ be a prevariety, $L \subseteq A^*$ be a regular language and $\alpha : A^* \to M$ be its syntactic morphism. Then, $L \in \mathrm{UPol}(\mathrm{BPol}(\mathcal{C}))$ if and only if α satisfies the following property:*

$$(esete)^{\omega+1} = (esete)^{\omega} ete (esete)^{\omega} \quad \text{for every } \mathcal{C}\text{-pair } (e,s) \in M^2 \text{ and every } t \in M. \tag{1}$$

5 Algebraic characterization of TL($\mathcal{C}$)

We present a generic characterization of $\mathrm{TL}(\mathcal{C})$ when $\mathcal{C}$ is a prevariety. It elegantly generalizes the characterizations of $\mathrm{TL} = \mathrm{FO}^2(<)$ and $\mathrm{TLX} = \mathrm{FO}^2(<, +1)$ by Thérien and Wilke [36] and that of $\mathrm{TL}(\mathrm{AT}) = \mathrm{FO}^2(\mathbb{I}_{\mathrm{AT}})$ by Krebs, Lodaya, Pandya and Straubing [13, 14, 15].

5.1 Statement

The characterization is based on the well-known variety of finite monoids DA (see [35] for a survey on this class). A finite monoid M belongs to DA if it satisfies the following equation:

$$(st)^{\omega} = (st)^{\omega} t (st)^{\omega} \quad \text{for every } s,t \in M. \tag{2}$$

Thérien and Wilke [36] showed that a regular language belongs to TL if and only if its syntactic monoid is in DA (strictly speaking, they considered two-variable first-order logic, the equality $\mathrm{FO}^2(<) = \mathrm{TL}$ is due to Etessami, Vardi and Wilke [9]). We extend this result in the following generic characterization of $\mathrm{TL}(\mathcal{C})$, based on $\mathcal{C}$-orbits introduced in Section 3.

▶ **Theorem 12.** *Let $\mathcal{C}$ be a prevariety, $L \subseteq A^*$ be a regular language and $\alpha : A^* \to M$ be its syntactic morphism. The two following properties are equivalent:*

1. *$L \in \mathrm{TL}(\mathcal{C})$.*
2. *For every idempotent $e \in E(M)$, the $\mathcal{C}$-orbit of e for α belongs to* DA.

Given as input a regular language $L \subseteq A^*$, one can compute its syntactic morphism $\alpha : A^* \to M$. In view of Theorem 12, $L \in \mathrm{TL}(\mathcal{C})$ if and only if for every $e \in E(M)$, the $\mathcal{C}$-orbit of e for α belongs to DA. The latter condition can be decided by checking all $\mathcal{C}$-orbits, provided that we are able to compute them. By Lemma 5, this is possible when $\mathcal{C}$-separation is decidable. Altogether, we obtain the following corollary of Theorem 12.

▶ **Corollary 13.** *If a prevariety $\mathcal{C}$ has decidable separation,* $\mathrm{TL}(\mathcal{C})$ *has decidable membership.*

▶ Remark 14. Let $L \subseteq A^*$ be a regular language and $\alpha : A^* \to M$ be its syntactic morphism. The fact that the $\mathcal{C}$-orbit of $e \in E(M)$ for α belongs to DA means that we have,

$$(esete)^{\omega} = (esete)^{\omega} ete (esete)^{\omega} \quad \text{for all } s,t \in M \text{ such that } (e,s) \text{ and } (e,t) \text{ are } \mathcal{C}\text{-pairs.} \tag{3}$$

One can check that (3) follows from (1), which characterizes $\mathrm{UPol}(\mathrm{BPol}(\mathcal{C}))$ (this is consistent with Proposition 10 asserting that $\mathrm{UPol}(\mathrm{BPol}(\mathcal{C})) \subseteq \mathrm{TL}(\mathcal{C})$). Indeed, choosing $t = s$ in (1) shows that the $\mathcal{C}$-orbit of e is aperiodic, *i.e.*, $(ese)^{\omega+1} = (ese)^{\omega}$ if (e,s) is a $\mathcal{C}$-pair. However, note that the element t is "free" in (1), whereas it must be part of a $\mathcal{C}$-pair (e,t) in (3).

Before proving Theorem 12, we first explain why it generalizes the original characterizations of the classes TL, TLX and TL(AT), as mentioned at the beginning of the section.

5.2 Application to historical classes

We first deduce the original characterizations of the classes TL = TL(ST) and TLX = TL(DD) by Thérien and Wilke [36] as simple corollaries of Theorem 12. We start with the former.

▶ **Theorem 15** (Thérien and Wilke [36]). *Let $L \subseteq A^*$ be a regular language and let M be its syntactic monoid. The two following properties are equivalent:*
1. *L belongs to* TL.
2. *M belongs to* DA.

Proof. Let $\alpha : A^* \to M$ be the syntactic morphism of L. Since TL = TL(ST), Theorem 12 implies that $L \in$ TL if and only if every ST-orbit for α belongs to DA. Since ST $= \{\emptyset, A^*\}$, every pair $(e, s) \in E(M) \times M$ is a $\mathcal{C}$-pair, so that the ST-orbit of $e \in E(M)$ for α is eMe. In particular the ST-orbit of 1_M is the whole monoid M. Hence, every ST-orbit for α belongs to DA if and only if M belongs to DA, which completes the proof. ◀

We turn to the characterization of TLX = TL(DD), also due to Thérien and Wilke [36]. In order to state it, we need an additional definition. Consider a regular language L and let $\alpha : A^* \to M$ be its syntactic morphism. The *syntactic semigroup of* L is the set $S = \alpha(A^+)$. Note that for every idempotent $e \in E(S)$, the set eSe is a monoid whose neutral element is e.

▶ **Theorem 16** (Thérien and Wilke [36]). *Let $L \subseteq A^*$ be a regular language and S be its syntactic semigroup. The two following properties are equivalent:*
1. *L belongs to* TLX.
2. *For every $e \in E(S)$, the monoid eSe belongs to* DA.

Proof. Let $\alpha : A^* \to M$ be the syntactic morphism of L. For $e \in E(M)$, let $M_e \subseteq M$ be the DD-orbit of e for α. Since DD $= \{\emptyset, \{\varepsilon\}, A^+, A^*\}$, for $(e, s) \in E(S) \times S$, the language $\alpha^{-1}(e)$ is not DD-separable from $\alpha^{-1}(s)$. Hence, (e, s) is a $\mathcal{C}$-pair, so that $M_e = eSe$ for all $e \in E(S)$. Moreover, if $1_M \notin E(S)$ (which means that $\alpha^{-1}(1_M) = \{\varepsilon\}$), then we have $M_{1_M} = \{1_M\}$ (which clearly belongs to DA). Hence, every DD-orbit for α belongs to DA if and only if $eSe \in$ DA for every $e \in E(S)$. In view of Theorem 12, this implies Theorem 16. ◀

Finally, we consider the class TL(AT), defined and characterized by Krebs, Lodaya, Pandya and Straubing [13, 14, 15]. Let us first present their characterization. It is based on a variety of finite monoids called M_eDA. Let M be a finite monoid. For each $e \in E(M)$, let $N_e \subseteq M$ be the submonoid of M generated by the set $\{s \in M \mid e \leqslant_{\mathcal{J}} s\}$. We say that M belongs to M_eDA if and only if for every idempotent $e \in E(M)$, the monoid of eN_ee belongs to DA.

▶ **Theorem 17** (Krebs, Lodaya, Pandya and Straubing [15]). *Let $L \subseteq A^*$ be a regular language and M be its syntactic monoid. The two following properties are equivalent:*
1. *$L \in$* TL(AT).
2. *M belongs to* M_eDA.

Proof. For $w \in A^*$, let $alph(w) \subseteq A$ be the set of letters occurring in w (*i.e.*, the least set $B \subseteq A$ such that $w \in B^*$). For $e \in E(M)$, let M_e be the AT-orbit of e for α. We prove that $M_e = eN_ee$ for every $e \in E(M)$. It will follows that M belongs to M_eDA if and only if every AT-orbit for α belongs to DA. In view of Theorem 12 this implies Theorem 17.

We first consider $s' \in eN_ee$ and prove that $s' \in M_e$. We have $s \in N_e$ such that $s' = ese$. By definition, $s = s_1 \cdots s_n$ where $e \leqslant_{\mathcal{J}} s_i$ for every $i \leq n$. If $n = 0$, then $s = 1_M$ and $ese = e \in M_e$. Assume now that we have $n \geq 1$. Since $e \leqslant_{\mathcal{J}} s_i$, we have $q_i, r_i \in M$ such that $e = q_i s_i r_i$ for every $i \leq n$. Hence, since $e \in E(M)$, we have $e = q_1 s_1 r_1 \cdots q_n s_n r_n$. For every

$i \leq n$, let $x_i \in \alpha^{-1}(q_i)$, $y_i \in \alpha^{-1}(r_i)$ and $u_i \in \alpha^{-1}(s_i)$. Finally, let $w = x_1u_1y_1 \cdots x_nu_ny_n$ and $w' = wu_1 \cdots u_nw$. By definition, we have $e = \alpha(w)$ and $ese = \alpha(w')$. Moreover, it is clear that $alph(w) = alph(w')$. By definition of AT, it follows that $\alpha^{-1}(e)$ is not AT-separable from $\alpha^{-1}(ese)$. Thus, (e, ese) is an AT-pair for α, which yields $s' = ese \in M_e$, as desired.

Conversely, let $s' \in M_e$. By definition, there exists an AT-pair $(e, s) \in M^2$ with $e \in E(M)$ such that $s' = ese$. Therefore, by definition of AT, there exist $u, v \in A^*$ such that $alph(u) = alph(v)$, $\alpha(u) = e$ and $\alpha(v) = s$. Let $a_1 \ldots, a_n \in A$ be the letters such that $v = a_1 \cdots a_n$. Since $alph(u) = alph(v)$, it is immediate that for each $i \leq n$, there are $x_i, y_i \in A^*$ such that $u = x_ia_iy_i$. Hence $e = \alpha(u) \leqslant_{\mathcal{J}} \alpha(a_i)$ and we conclude that $s = \alpha(a_1 \cdots a_n) \in N_e$. Consequently, $s' = ese \in eN_ee$, as desired. ◀

5.3 Proof of Theorem 12

We fix a prevariety $\mathcal{C}$, a regular language $L \subseteq A^*$ and its syntactic morphism $\alpha : A^* \to M$ for the proof. We prove that $L \in \mathrm{TL}(\mathcal{C})$ if and only if all $\mathcal{C}$-orbits for α belong to DA. We start with the left-to-right implication.

From $\mathsf{TL}(\mathcal{C})$ to DA. This direction follows from results of [27]. To use them, we need some preliminary terminology. We introduce equivalence relations connected to the class $\mathrm{TL}(\mathcal{C})$ when $\mathcal{C}$ is a prevariety. Given a morphism $\eta : A^* \to N$ into a finite monoid N, denote by $\mathcal{C}_\eta$ be the class of all languages recognized by η. The following fact is easy (see [27, Fact 9.3]).

▶ **Fact 18.** *Let $\mathcal{C}$ be a prevariety. For every $\mathrm{TL}[\mathcal{C}]$ formula φ, there exists a $\mathcal{C}$-morphism $\eta : A^* \to N$ such that φ is a $\mathrm{TL}[\mathcal{C}_\eta]$ formula.*

We use the standard notion of rank of a $\mathrm{TL}[\mathcal{C}_\eta]$ formula: the *rank* of φ is defined as the length of the longest sequence of nested temporal operators within its parse tree. Formally:

- Any atomic formula has rank 0.
- The rank of $\neg\varphi$ is the same as the rank of φ.
- The rank of $\varphi \vee \psi$ and $\varphi \wedge \psi$ is the maximum between the ranks of φ and ψ.
- For every language $L \subseteq A^*$, the rank of $\mathrm{F}_L\,\varphi$ and $\mathrm{P}_L\,\varphi$ is the rank of φ plus 1.

Two $\mathrm{TL}[\mathcal{C}_\eta]$ formulas φ and ψ are *equivalent* if they have the same semantics. That is, for every $w \in A^*$ and every position $i \in \mathsf{Pos}(w)$, we have $w, i \models \varphi \Leftrightarrow w, i \models \psi$. The following key lemma is immediate from a simple induction on the rank of TL formulas.

▶ **Lemma 19.** *Let $\eta : A^* \to N$ be a morphism into a finite monoid and let $k \in \mathbb{N}$. There are only finitely many non-equivalent $\mathrm{TL}[\mathcal{C}_\eta]$ formulas of rank at most k.*

We now define equivalence relations. Let $\eta : A^* \to N$ be a morphism into a finite monoid and let $k \in \mathbb{N}$. Given $w, w' \in A^*$, $i \in \mathsf{Pos}(w)$ and $i' \in \mathsf{Pos}(w')$, we write, $w, i \cong_{\eta,k} w', i'$ when:

$$\text{For every } \mathrm{TL}[\mathcal{C}_\eta] \text{ formula } \varphi \text{ of rank at most } k, \quad w, i \models \varphi \Longleftrightarrow w', i' \models \varphi.$$

It is straightforward that $\cong_{\eta,k}$ is an equivalence relation. Moreover, it is immediate from the definition and Lemma 19, that $\cong_{\eta,k}$ has finite index. We lift each relation $\cong_{\eta,k}$ to A^* (abusing terminology, we also denote by $\cong_{\eta,k}$ the new relation): given $w, w' \in A^*$, we write $w \cong_{\eta,k} w'$ when $w, 0 \cong_{\eta,k} w', 0$. Clearly, $\cong_{\eta,k}$ is an equivalence relation of finite index over A^*. Moreover, we have the following connection between $\mathrm{TL}(\mathcal{C})$ and the relations $\cong_{\eta,k}$.

▶ **Lemma 20.** *Let $\mathcal{C}$ be a prevariety and $L \subseteq A^*$. If $L \in \mathrm{TL}(\mathcal{C})$, then there exists a $\mathcal{C}$-morphism $\eta : A^* \to N$ and $k \in \mathbb{N}$ such that L is a union of $\cong_{\eta,k}$-classes.*

Proof. Let $L \in \mathrm{TL}(\mathcal{C})$. There exists a $\mathrm{TL}[\mathcal{C}]$ formula φ such that $w \in L \Leftrightarrow w, 0 \models \varphi$ for all $w \in A^*$. By Fact 18, there exists a $\mathcal{C}$-morphism $\eta : A^* \to N$ such that φ is a $\mathrm{TL}[\mathcal{C}_\eta]$ formula. Let $k \in \mathbb{N}$ be the rank of φ. We prove that L is a union of $\cong_{\eta,k}$-classes. Given $w, w' \in A^*$ such that $w \cong_{\eta,k} w'$, we have to prove that $w \in L \Leftrightarrow w' \in L$. By symmetry, we only prove the left to right implication. Thus, we assume that $w \in L$. By definition of φ, it follows that $w, 0 \models \varphi$. Moreover, since $w \cong_{\eta,k} w'$ (*i.e.*, $w, 0 \cong_{\eta,k} w', 0$) and φ is a $\mathrm{TL}[\mathcal{C}_\eta]$ formula of rank k, we have $w', 0 \models \varphi$ by definition of $\cong_{\eta,k}$. Hence, $w' \in L$ by definition of φ, as desired. ◀

In addition to the link stated in Lemma 20 between $\mathrm{TL}(\mathcal{C})$ and the equivalence relations $\cong_{\eta,k}$, we use a property of $\cong_{\eta,k}$ that follows from [27, Lemma 9.6 and Proposition 9.7].

▶ **Proposition 21.** *Consider a morphism $\eta : A^* \to N$ into a finite monoid, let $f \in E(N)$ be an idempotent, let $u, v, z \in \eta^{-1}(f)$ and let $x, y \in A^*$. For every $k \in \mathbb{N}$, we have:*

$$x(z^k u z^{2k} v z^k)^k (z^k u z^{2k} v z^k)^k y \cong_{\eta,k} x(z^k u z^{2k} v z^k)^k z^k v z^k (z^k u z^{2k} v z^k)^k y.$$

We are ready to conclude this direction of the proof: assuming that $L \in \mathrm{TL}(\mathcal{C})$, we show that all $\mathcal{C}$-orbits for its syntactic monoid belong to DA. Let $e \in E(M)$ and M_e be its $\mathcal{C}$-orbit. Proving that $M_e \in \mathsf{DA}$ amounts to proving that any elements $s, t \in M_e$ satisfy (2). Fix $e, s, t \in E(M) \times M_e \times M_e$. Lemma 20 yields a $\mathcal{C}$-morphism $\eta : A^* \to N$ and $k \in \mathbb{N}$ such that L is a union of $\cong_{\eta,k}$-classes. Since η is a $\mathcal{C}$-morphism, Lemma 6 yields $f \in E(N)$ such that $M_e \subseteq \alpha(\eta^{-1}(f))$. Since $e, s, t \in M_e$, we get $z, u, v \in A^*$ such that $z, u, v \in \eta^{-1}(f)$, $\alpha(z) = e$, $\alpha(u) = s$ and $\alpha(v) = t$. Let $x, y \in A^*$ be two arbitrary words. By Proposition 21, we obtain,

$$x(z^k u z^{2k} v z^k)^k (z^k u z^{2k} v z^k)^k y \cong_{\eta,k} x(z^k u z^{2k} v z^k)^k z^k v z^k (z^k u z^{2k} v z^k)^k y.$$

Since L is a union of $\cong_{\eta,k}$-classes, the words $(z^k u z^k v z^k)^{2k}$ and $(z^k u z^k v z^k)^k z^k v z^k (z^k u z^k v z^k)^k$ are equivalent for the syntactic congruence of L, so they have the same image under its syntactic morphism α. Since $e \in E(M)$, this yields $(esete)^{2k} = (esete)^k ete (esete)^k$. Hence, $(st)^{2k} = (st)^k t (st)^k$ since $e, s, t \in M_e$ and e is neutral in M_e by Lemma 4. It now suffices to multiply by enough copies of st on both sides to get $(st)^\omega = (st)^\omega t (st)^\omega$. Therefore, (2) holds.

From DA to TL($\mathcal{C}$). Assuming that every $\mathcal{C}$-orbit for the syntactic morphism $\alpha : A^* \to M$ of L belongs to DA, we have to show that $L \in \mathrm{TL}(\mathcal{C})$, *i.e.*, to build a $\mathrm{TL}[\mathcal{C}]$ formula defining L. Let us start by giving a high-level overview of the proof for this direction.

Since $\mathrm{TL}(\mathcal{C})$ is closed under union, it suffices to prove that for all $s \in M$, the language $\alpha^{-1}(s)$ is in $\mathrm{TL}(\mathcal{C})$. We achieve this by inductively constructing a $\mathrm{TL}[\mathcal{C}]$ formula defining $\alpha^{-1}(s)$. According to Lemma 3, there exists a $\mathcal{C}$-morphism $\eta : A^* \to N$ such that the $\mathcal{C}$-pairs for α are exactly the η-pairs for α. We use η to leverage the assumption that all $\mathcal{C}$-orbits for α belong to DA. More precisely, η recognizes all the basic languages in $\mathcal{C}$ that we shall use in our $\mathrm{TL}[\mathcal{C}]$ formulas. The induction proceeds as follows: using η, we define a sequence of languages $K_0 \supseteq K_1 \supseteq \cdots \supseteq K_{|N|}$ and show by induction on $|N| - \ell$ that $K_\ell \cap \alpha^{-1}(s)$ can be defined by a $\mathrm{TL}[\mathcal{C}]$ formula for each $\ell \leq |N|$. The induction basis is the case $\ell = |N|$, which is simple because $K_{|N|}$ is a finite language. Furthermore, the case $\ell = 0$ gives the desired result since K_0 contains all words. The induction step consists in building a $\mathrm{TL}[\mathcal{C}]$ formula describing $K_\ell \cap \alpha^{-1}(s)$ from several $\mathrm{TL}[\mathcal{C}]$ formulas that describe the languages $K_{\ell+1} \cap \alpha^{-1}(t)$ for all $t \in M$. However, the actual argument is slightly more involved. Indeed, in order to perform the induction step, we must abstract each word in $K_\ell \cap \alpha^{-1}(s)$ by considering a specific decomposition of this word and viewing each infix as a new letter. We then argue that the resulting word belongs to $K_{\ell+1} \cap \alpha^{-1}(s)$, which allows us to apply induction. Yet,

for this process to work, the letter that we use to abstract an infix must have the same images as this original infix under both α and η. This is problematic, because such a letter does not necessarily exist. We solve this issue by considering an extended alphabet B, replacing $\alpha : A^* \to M$ and $\eta : A^* \to N$ with two new morphisms $\beta : B^* \to M$ and $\delta : B^* \to N$ that have the required property. Of course, this involves some preliminary work: we must reformulate both our objective (proving that all languages $\alpha^{-1}(s)$ can be defined in $\mathrm{TL}(\mathcal{C})$) and our hypothesis (that every $\mathcal{C}$-orbit for α belongs to DA) on the new morphisms β and η.

We now start the proof by first defining β and δ. Recall that $\eta : A^* \to N$ is the $\mathcal{C}$-morphism provided by Lemma 3: it is such that the $\mathcal{C}$-pairs for α are exactly the η-pairs for α. We fix η for the entire proof. We define an auxiliary alphabet B. Let $P \subseteq M \times N$ be the set of all pairs $(\alpha(w), \eta(w)) \in M \times N$ where $w \in A^+$ is a nonempty word. For each pair $(s, r) \in P$, we create a fresh letter $b_{s,r} \notin A$ and we define $B = \{b_{s,r} \mid (s, r) \in P\}$.

Let $\beta : B^* \to M$ and $\delta : B^* \to N$ be the morphisms defined by $\beta(b_{s,r}) = s$ and $\delta(b_{s,r}) = r$ for $(s, r) \in P$. By definition, we have $(\beta(w), \delta(w)) \in P$ for all $w \in B^+$. Let $\mathcal{C}_\delta$ be the class of all languages over B recognized by δ. One can check that $\mathcal{C}_\delta$ is a prevariety. We now reduce membership of inverse images under α to $\mathrm{TL}(\mathcal{C})$ to that of inverse images under β to $\mathrm{TL}(\mathcal{C}_\delta)$.

▶ **Lemma 22.** *For every $F \subseteq M$, if $\beta^{-1}(F) \in \mathrm{TL}(\mathcal{C}_\delta)$, then $\alpha^{-1}(F) \in \mathrm{TL}(\mathcal{C})$.*

Proof. We first define a morphism $\gamma : A^* \to B^*$. Consider a letter $a \in A$. By definition, $(\alpha(a), \eta(a)) \in P$. Hence, we may define $\gamma(a) = b_{\alpha(a),\eta(a)} \in B$. By definition, we have $\alpha(w) = \beta(\gamma(w)) \in M$ and $\eta(w) = \delta(\gamma(w))$ for every $w \in A^*$. It follows that for every $F \subseteq M$, we have $\alpha^{-1}(F) = \gamma^{-1}(\beta^{-1}(F)) \subseteq A^*$. Consequently, it now suffices to prove that for every $K \subseteq B^*$ such that $K \in \mathrm{TL}(\mathcal{C}_\delta)$, we have $\gamma^{-1}(K) \in \mathrm{TL}(\mathcal{C})$. We fix K for the proof. Since $K \in \mathrm{TL}(\mathcal{C}_\delta)$, it is defined by a formula $\psi \in \mathrm{TL}[\mathcal{C}_\delta]$. We apply two kinds of modifications to ψ in order to build a new formula $\psi' \in \mathrm{TL}[\mathcal{C}]$ defining $\gamma^{-1}(K)$:

1. We replace every atomic subformula "b" for $b \in B$ by the $\mathrm{TL}[\mathcal{C}]$-formula $\bigvee_{\{a \in A \mid \gamma(a) = b\}} a$.
2. For every temporal modality F_H (resp. P_H) occurring in ψ, we have $H \in \mathcal{C}_\delta$ by hypothesis. Hence, H is recognized by δ and there exists $G \subseteq N$ such that $H = \delta^{-1}(G)$. Note that $\eta^{-1}(G) \in \mathcal{C}$ since η is a $\mathcal{C}$-morphism. We replace the temporal modality F_H (resp. P_H) by $\mathrm{F}_{\eta^{-1}(G)}$ (resp. $\mathrm{P}_{\eta^{-1}(G)}$).

By definition the resulting formula ψ' belongs to $\mathrm{TL}[\mathcal{C}]$ and one can verify that for every $w \in A^*$, we have $w, 0 \models \psi' \Leftrightarrow \gamma(w), 0 \models \psi$. Since $L_{min}(\psi) = K$, we get $L_{min}(\psi') = \gamma^{-1}(K)$, which implies that $K \in \mathrm{TL}(\mathcal{C})$. This completes the proof. ◀

In view of Lemma 22, it suffices to prove that any language recognized by β belongs to $\mathrm{TL}(\mathcal{C}_\delta)$. Since L is recognized by α, this will imply $L \in \mathrm{TL}(\mathcal{C})$, which is our goal. In the next lemma, we reformulate on β and δ the assumption that every $\mathcal{C}$-orbit for α belongs to DA.

▶ **Lemma 23.** *For every $e \in E(M)$ and every $s, t \in M$, if (e, s) and (e, t) are δ-pairs for β, then $(esete)^\omega = (esete)^\omega ete(esete)^\omega$.*

Proof. By hypothesis, there exist $u, v, x, y \in B^*$ such that $\delta(u) = \delta(v)$, $\delta(x) = \delta(y)$, $\beta(u) = \beta(x) = e$, $\beta(v) = s$ and $\beta(y) = t$. The definitions of β and δ imply that for any $w \in B^*$, there exists $w' \in A^*$ such that $\delta(w) = \eta(w')$ and $\beta(w) = \alpha(w')$. Therefore, we obtain $u', v', x', y' \in A^*$ such that $\eta(u') = \eta(v')$, $\eta(x') = \eta(y')$, $\alpha(u') = \alpha(x') = e$, $\alpha(v') = s$ and $\alpha(y') = t$. Thus, $(e, s) \in M^2$ and $(e, t) \in M^2$ are η-pairs for α. By definition of η, it follows that they are $\mathcal{C}$-pairs for α. Hence, ese and ete both belong to the $\mathcal{C}$-orbit of e for α. Since all $\mathcal{C}$-orbits for α belong to DA by hypothesis, this gives $(esete)^\omega = (esete)^\omega ete(esete)^\omega$. ◀

We now use the Green relation $\mathcal{J}$ over N to associate a number $d_{\mathcal{J}}(r) \in \mathbb{N}$ with every element $r \in N$. We let $d_{\mathcal{J}}(r)$ be the maximal number $n \in \mathbb{N}$ such that there exist n elements $r_1, \ldots, r_n \in N$ satisfying $r <_{\mathcal{J}} r_1 <_{\mathcal{J}} \cdots <_{\mathcal{J}} r_n$. By definition, $0 \leq d_{\mathcal{J}}(r) \leq |N| - 1$. In particular, we have $d_{\mathcal{J}}(r) = 0$ if and only if r is maximal for $\leqslant_{\mathcal{J}}$ (*i.e.*, if and only if $r \mathrel{\mathcal{J}} 1_N$). Finally, given a word $w \in B^*$, we write $d_{\mathcal{J}}(w) \in \mathbb{N}$ for $d_{\mathcal{J}}(\delta(w))$. Observe that for all $x, y, z \in B^*$, we have $d_{\mathcal{J}}(y) \leq d_{\mathcal{J}}(xyz)$ (as $xyz \leqslant_{\mathcal{J}} y$), a fact that we shall use frequently.

In order to argue inductively, we define a family of languages $K_\ell \subseteq B^*$ for $\ell \in \mathbb{N}$ as follows:

$$K_\ell = \big\{w \in B^* \mid \text{for all } k \leq \ell \text{ and } x, y, z \in B^*, \text{ if } w = xyz \text{ and } |y| = k, \text{ then } d_{\mathcal{J}}(y) \geq k\big\}.$$

Note that $K_0 = B^*$ as $d_{\mathcal{J}}(y) \geq 0$ for all $y \in B^*$. Also, if $\ell \geq |N|$, then K_ℓ is *finite* (it contains words of length at most $|N| - 1$ as $d_{\mathcal{J}}(y) < |N|$ for all $y \in B^*$). We now have the next lemma.

▶ **Lemma 24.** *Let $\ell \in \mathbb{N}$ and $w \in K_\ell$. Then $d_{\mathcal{J}}(w) \leq \ell$ if and only if for all $x, y, z \in B^*$ such that $w = xyz$ and $|y| \leq \ell + 1$, we have $d_{\mathcal{J}}(y) \leq \ell$.*

Proof. The "only if" direction is immediate since $d_{\mathcal{J}}(y) \leq d_{\mathcal{J}}(w)$ for every infix y of w. Conversely, assume that for all $x, y, z \in B^*$ such that $w = xyz$ and $|y| \leq \ell + 1$, we have $d_{\mathcal{J}}(y) \leq \ell$. We prove that $d_{\mathcal{J}}(w) \leq \ell$. If $|w| \leq \ell + 1$, this is immediate. Assume now that $|w| > \ell + 1$. We get $b_1, \ldots, b_n \in B$ and $v \in B^*$ such that $|v| = \ell + 1$ and $w = vb_1 \cdots b_n$. We use induction on i to prove that $\delta(v) \mathrel{\mathcal{J}} \delta(vb_1 \cdots b_i)$ for all $i \leq n$. Since $d_{\mathcal{J}}(v) \leq \ell$ by hypothesis, the case $i = n$ yields $d_{\mathcal{J}}(w) \leq \ell$. The case $i = 0$ is trivial: we have $\delta(v) \mathrel{\mathcal{J}} \delta(v)$. Assume now that $i \geq 1$. By induction hypothesis, we know that $\delta(v) \mathrel{\mathcal{J}} \delta(vb_1 \cdots b_{i-1})$. Let $x, y \in B^*$ such that $|y| = \ell$ and $xy = vb_1 \cdots b_{i-1}$ (the words x and y exist because $|v| = \ell + 1$). Since $w \in K_\ell$, and y is an infix of w such that $|y| = \ell$, we know that $d_{\mathcal{J}}(y) \geq \ell$. Moreover, yb_i is an infix of w such that $|yb_i| = \ell + 1$, which yields $d_{\mathcal{J}}(yb_i) \leq \ell$ by hypothesis. Since $d_{\mathcal{J}}(y) \leq d_{\mathcal{J}}(yb_i)$, we get $d_{\mathcal{J}}(yb_i) = d_{\mathcal{J}}(y) = \ell$, which implies that $\delta(yb_i) \mathrel{\mathcal{J}} \delta(y)$. Moreover, we have $\delta(yb_i) \leqslant_{\mathcal{R}} \delta(y)$. Thus, Lemma 1 yields $\delta(yb_i) \mathrel{\mathcal{R}} \delta(y)$. This implies that $\delta(xyb_i) \mathrel{\mathcal{R}} \delta(xy)$. Hence, $\delta(vb_1 \cdots b_i) \mathrel{\mathcal{J}} \delta(vb_1 \cdots b_{i-1}) \mathrel{\mathcal{J}} \delta(v)$. This completes the proof. ◀

We now prove that for all $s \in M$ and $\ell \in \mathbb{N}$, we have $K_\ell \cap \beta^{-1}(s) \in \mathrm{TL}(\mathcal{C}_\delta)$. Our objective (every language recognized by β belongs to $\mathrm{TL}(\mathcal{C}_\delta)$) follows from the case $\ell = 0$, since $K_0 = B^*$. The proof involves two steps. The first settles the case of elements of $K_\ell \cap \beta^{-1}(s)$ whose image under δ has a $d_{\mathcal{J}}$ value at most ℓ. We do not use induction for this case, which relies on the inclusion $\mathrm{UPol}(\mathrm{BPol}(\mathcal{C}_\delta)) \subseteq \mathrm{TL}(\mathcal{C}_\delta)$. It is also the place where we use Lemma 23, *i.e.*, the hypothesis that all $\mathcal{C}$-orbits for α are in DA.

▶ **Proposition 25.** *Let $(\ell, s, r) \in \mathbb{N} \times M \times N$. If $d_{\mathcal{J}}(r) \leq \ell$ then $K_\ell \cap \beta^{-1}(s) \cap \delta^{-1}(r) \in \mathrm{TL}(\mathcal{C}_\delta)$.*

Proof. We prove that $K_\ell \cap \beta^{-1}(s) \cap \delta^{-1}(r) \in \mathrm{UPol}(\mathrm{BPol}(\mathcal{C}_\delta))$, which, by Proposition 10, will give the desired result $K_\ell \cap \beta^{-1}(s) \cap \delta^{-1}(r) \in \mathrm{TL}(\mathcal{C}_\delta)$. Let $\gamma : B^* \to Q$ be the syntactic morphism of $K_\ell \cap \beta^{-1}(s) \cap \delta^{-1}(r)$. By Theorem 11, it suffices to show that given $q_1, q_2 \in Q$ and $f \in E(Q)$ such that $(f, q_1) \in Q^2$ is a $\mathcal{C}_\delta$-pair for γ, the following equation holds:

$$(fq_1fq_2f)^{\omega+1} = (fq_1fq_2f)^{\omega} fq_2f(fq_1fq_2f)^{\omega}. \tag{4}$$

Let $q_1, q_2, f \in Q$ be such elements. By definition of $\mathcal{C}_\delta$, we know that δ is a $\mathcal{C}_\delta$-morphism. Therefore, Lemma 3 implies that (f, q_1) is a δ-pair for γ. We get $u', v'_1 \in B^*$, such that $\delta(u') = \delta(v'_1)$, $\gamma(u') = f$ and $\gamma(v'_1) = q_1$. Note that if $v'_1 = \varepsilon$, then $q_1 = 1_Q$ and (4) holds since it is clear that $(fq_2f)^{\omega+1} = (fq_2f)^{2\omega+1}$. Therefore, we assume from now on that $v'_1 \in B^+$. Let us also choose $v'_2 \in B^*$ such that $\gamma(v'_2) = q_2$. We now define $p = \ell \times \omega(N) \times \omega(M) \times \omega(Q)$,

$u = (u')^p$, $v_1 = (u')^{p-1}v_1'$ and $v_2 = uv_2'u(uv_1uv_2'u)^{p-1}$. We compute $\gamma(u) = f$, $\gamma(v_1) = fq_1$ and $\delta(u) = \delta(v_1)$. Moreover, since p is a multiple of $\omega(N)$, the element $\delta(u) = \delta(v_1)$ is an idempotent $g \in E(N)$. Finally, we have $\gamma(v_2) = fq_2f(fq_1fq_2f)^{p-1}$ and $\delta(v_2) = (g\delta(v_2')g)^p$. In particular, it follows that $\delta(v_2)$ is an idempotent $h \in E(N)$ such that $gh = hg = h$.

We prove that $(uv_1uv_2u)^p$ and $(uv_1uv_2u)^puv_2u(uv_1uv_2u)^p$ are equivalent for the syntactic congruence of $K_\ell \cap \beta^{-1}(s) \cap \delta^{-1}(r)$. This will imply that they have the same image under γ, which yields $(fq_1fq_2f)^\omega = (fq_1fq_2f)^\omega fq_2f(fq_1fq_2f)^{2\omega-1}$. One may then multiply by fq_1fq_2f on the right to get (4), as desired. For $x, y \in A^*$, let $z_1 = x(uv_1uv_2u)^py$ and $z_2 = x(uv_1uv_2u)^puv_2u(uv_1uv_2u)^py$. We have to show that $z_1 \in K_\ell \cap \beta^{-1}(s) \cap \delta^{-1}(r)$ if and only if $z_2 \in K_\ell \cap \beta^{-1}(s) \cap \delta^{-1}(r)$. We first treat the special case where $|u| < \ell$.

Assume that $|u| < \ell$. We show that in this case $z_1 \notin K_\ell$ and $z_2 \notin K_\ell$ (which implies the desired result). Since $u = (u')^p$ and $p \geq \ell$, the hypothesis that $|u| < \ell$ yields $u = u' = \varepsilon$. Since $\delta(u) = \delta(v_1)$, we get $\delta(v_1) = 1_N$. Recall that $v_1 = (u')^{p-1}v_1'$ and $v_1' \in B^+$ by hypothesis. Thus, $v_1 \in B^+$, which means that it contains a letter $b \in B$ such that $\delta(b) \mathcal{J} 1_N$. In particular $d_{\mathcal{J}}(b) = 0$. Hence, b is an infix of length 1 of both z_1 and z_2 such that $d_{\mathcal{J}}(b) < 1$. Now $\ell > |u| = 0$, so that $\ell \geq 1$. This implies $z_1 \notin K_\ell$ and $z_2 \notin K_\ell$. This completes the special case.

From now on, we assume that $|u| \geq \ell$. Since $\delta(u) = \delta(v_1) = g \in E(N)$, $\delta(v_2) = h \in E(N)$ and $gh = hg = h$, we have $\delta(z_1) = \delta(z_2) = \delta(x)h\delta(y)$. Therefore, $z_1 \in \delta^{-1}(r)$ if and only if $z_2 \in \delta^{-1}(r)$. Let us prove that $z_1 \in K_\ell \Leftrightarrow z_2 \in K_\ell$. This is trivial if $\ell = 0$ since $K_0 = B^*$. Assume now that $\ell \geq 1$. Since $|u| \geq \ell$ by hypothesis, it follows that for every $k \leq \ell$, z_1 and z_2 have the same infixes of length k. This implies that $z_1 \in K_\ell \Leftrightarrow z_2 \in K_\ell$, as desired.

It remains to prove that if $z_1, z_2 \in K_\ell \cap \delta^{-1}(r)$, then $\beta(z_1) = \beta(z_2)$. We first show that our assumptions imply $g \mathcal{J} h$. Again, there are two cases. First, assume that $\ell = 0$. Since $d_{\mathcal{J}}(r) \leq \ell$ by hypothesis, we get $r \mathcal{J} 1_N$. Thus, since u and v_2 are infixes of $z_1 \in \delta^{-1}(r)$, we have $\delta(u) \mathcal{J} \delta(v_2) \mathcal{J} 1_N$, which exactly says that $g \mathcal{J} h \mathcal{J} 1_N$. Assume now that $\ell \geq 1$. Recall that $|u| \geq \ell$. Since u is an infix of v_2, this also implies that $|v_2| \geq \ell$. Hence, since u and v_2 are infixes of $z_2 \in K_\ell \cap \delta^{-1}(r)$, we get $d_{\mathcal{J}}(u) \geq \ell$ and $d_{\mathcal{J}}(v_2) \geq \ell$, $r \leqslant_{\mathcal{J}} \delta(u)$ and $r \leqslant_{\mathcal{J}} \delta(v_2)$. In particular, it follows that $d_{\mathcal{J}}(r) \geq d_{\mathcal{J}}(u) \geq \ell$ and $d_{\mathcal{J}}(r) \geq d_{\mathcal{J}}(v_2) \geq \ell$. Since $d_{\mathcal{J}}(r) \leq \ell$ by hypothesis on r, we get $d_{\mathcal{J}}(r) = d_{\mathcal{J}}(u) = d_{\mathcal{J}}(v_2) = \ell$. Together with $r \leqslant_{\mathcal{J}} \delta(u)$ and $r \leqslant_{\mathcal{J}} \delta(v_2)$, this yields $r \mathcal{J} \delta(u) \mathcal{J} \delta(v_2)$, *i.e.*, $r \mathcal{J} g \mathcal{J} h$. This completes the proof that $g \mathcal{J} h$. Since we also know that $hg = gh = h$, we have $h \leqslant_{\mathcal{R}} g$ and Lemma 1 yields $g \mathcal{R} h$. We get $z \in N$ such that $g = hz$. Thus, we have $h = hg = hhz = hz = g$.

Altogether, we obtain $\delta(u) = \delta(v_1) = \delta(v_2) = g \in E(N)$. This implies that $(\beta(u), \beta(v_1))$ and $(\beta(u), \beta(v_2))$ are δ-pairs for β. Moreover, recall that $u = (u')^p$ where p is a multiple of $\omega(M)$. Hence, we have $\beta(u) \in E(M)$. Consequently, it follows from Lemma 23 that $\beta((uv_1uv_2u)^p) = \beta((uv_1uv_2u)^puv_2u(uv_1uv_2u)^p)$. It now suffices to multiply by $\beta(x)$ on the left and $\beta(y)$ on the right to obtain $\beta(z_1) = \beta(z_2)$, as desired. ◀

We now turn to the second step of the proof, which is formalized in the following statement.

▶ **Proposition 26.** *Let $\ell \leq |N|$ and $s \in M$. There exists a $\mathrm{TL}[\mathcal{C}_\delta]$ formula $\varphi_{\ell,s}$ such that for every $w \in K_\ell$, we have $w, 0 \models \varphi_{\ell,s} \Leftrightarrow \beta(w) = s$.*

Let us first use Proposition 26 to complete the main proof: we have to show that every language recognized by β belongs to $\mathrm{TL}(\mathcal{C}_\delta)$. Clearly, it suffices to show that $\beta^{-1}(s) \in \mathrm{TL}(\mathcal{C}_\delta)$ for each $s \in M$. We apply Proposition 26 for $\ell = 0$. Since $K_0 = B^*$, this yields a formula $\varphi_{0,s} \in \mathrm{TL}[\mathcal{C}_\delta]$ such that $L_{min}(\varphi_{0,s}) = \beta^{-1}(s)$. Thus, $\beta^{-1}(s) \in \mathrm{TL}(\mathcal{C}_\delta)$, as desired.

It remains to prove Proposition 26. We construct $\varphi_{\ell,s} \in \mathrm{TL}[\mathcal{C}_\delta]$ by induction on $|N| - \ell$. If $\ell = |N|$, we define $\varphi_{\ell,s}$ so that $L_{min}(\varphi_{\ell,s}) = K_\ell \cap \beta^{-1}(s)$. Since $K_{|N|} \cap \beta^{-1}(s)$ is finite and $\mathrm{TL}[\mathcal{C}_\delta]$ is closed under disjunction, it suffices to build for every word $w \in B^*$ a $\mathrm{TL}[\mathcal{C}_\delta]$ formula φ_w defining $\{w\}$. Since $B^* \in \mathcal{C}_\delta$, one may use the "F" modality. For $w = b_1 \cdots b_n$, let

$\psi_w = \mathrm{F}\,(b_1 \wedge \mathrm{F}\,(b_2 \wedge \mathrm{F}\,(b_3 \wedge \cdots \wedge \mathrm{F}\,b_n)))$.

One may then choose $\varphi_w = \psi_w \wedge \bigwedge_{u \in B^*, |u|=|w|+1} \neg\psi_u$.

Assume now that $\ell < |N|$. We present a construction for splitting the words in K_ℓ into two parts: a prefix mapped to an element $r \in N$ such that $d_{\mathcal{J}}(r) \leq \ell$ (we handle it with Proposition 25) and a suffix that we abstract as a word in $K_{\ell+1}$ (we handle it by induction).

Let $w \in K_\ell$. For each position $i \in \mathsf{Pos}(w) \setminus \{0\}$ and $k \in \mathbb{N}$, we write $\sigma_k(w,i) \in B^*$ for the infix $w(i-1,j)$ where $j = \min(i+k, |w|+1)$. In other words, $\sigma_k(w,i) = w[i] \cdots w[i+k-1]$ if $i+k-1 \leq |w|$ and $\sigma_k(w,i) = w[i] \cdots w[|w|]$ otherwise. In particular, we have $|\sigma_k(w,i)| \leq k$.

▶ **Lemma 27.** *Let $k \leq \ell+1$ and $u \in B^*$ be such that $|u| \leq k$. There exists a formula $\pi_{k,u} \in \mathrm{TL}[\mathcal{C}_\delta]$ such that for all $w \in K_\ell$ and $i \in \mathsf{Pos}(w) \setminus \{0\}$, $w,i \models \pi_{k,u} \Leftrightarrow \sigma_k(w,i) = u$.*

Proof. If $u = \varepsilon$, it suffices to define $\pi_{k,u} = \top$ when $k = 0$ and $\pi_{k,u} = \mathit{max}$ when $k \geq 1$. Assume now that $|u| \geq 1$. If $k \leq 1$, it follows that $|u| = 1 = k$. Hence, u is a letter $b \in B$ and it suffices to define $\pi_{k,u} = b$. Assume now that $k \geq 2$. Let $C \subseteq B$ be the set of letters mapped to 1_N under δ, so that $H \stackrel{\text{def}}{=} \delta^{-1}(1_N) = C^*$. By definition, $H \in \mathcal{C}_\delta$. Since $2 \leq k \leq \ell+1$, we have $\ell \geq 1$, which implies, by definition of K_ℓ, that no word of K_ℓ can contain a letter b with $d_{\mathcal{J}}(b) = 0$. In particular, words of K_ℓ cannot contain letters of C. Therefore, if $w \in K_\ell$, $i \in \mathsf{Pos}(w)$ and $\psi \in \mathrm{TL}[\mathcal{C}_\delta]$, we have $w,i \models \mathrm{F}_H\,\psi$ if and only if $w,i+1 \models \psi$. Let $u = b_1 \cdots b_n$ (with $b_i \in B$). We have $n = |u| \leq k$ by hypothesis. We consider two cases for defining $\pi_{k,u}$:

- If $n = k$, we let $\pi_{k,u} = (b_1 \wedge \mathrm{F}_H\,(b_2 \wedge \mathrm{F}_H\,(b_3 \wedge \cdots \mathrm{F}_H\,b_n)))$.
- If $n < k$, we let $\pi_{k,u} = (b_1 \wedge \mathrm{F}_H\,(b_2 \wedge \mathrm{F}_H\,(b_3 \wedge \cdots \mathrm{F}_H\,(b_n \wedge \mathrm{F}_H\,\mathit{max}))))$.

The above fact on F_H implies that this definition fulfills the desired property. ◀

Pointed positions. Consider $w \in K_\ell$. We say that an arbitrary position $i \in \mathsf{Pos}(w)$ is *pointed* when either $i \in \{0, |w|+1\}$, or $i \in \mathsf{Pos}_c(w)$ and $d_{\mathcal{J}}(\sigma_{\ell+1}(w,i)) \geq \ell+1$.

▶ **Definition 28** (Detection of pointed positions in $\mathrm{TL}[\mathcal{C}_\delta]$). *Let $\pi = \mathit{min} \vee \mathit{max} \vee \bigvee_{u \in U} \pi_{\ell+1,u}$ where $U = \{u \in B^* \mid |u| \leq \ell+1 \text{ and } d_{\mathcal{J}}(u) \geq \ell+1\}$. By definition of $\pi_{\ell+1,u}$ in Lemma 27, we know that for $w \in K_\ell$ and $i \in \mathsf{Pos}(w)$, we have $w,i \models \pi$ if and only if position i is pointed.*

A position $i \in \mathsf{Pos}(w)$ which is *not* pointed is said to be *safe*. We now prove that we may constrain the evaluation of $\mathrm{TL}[\mathcal{C}_\delta]$ formulas to infixes that only contain *safe* positions.

▶ **Lemma 29.** *Let $\psi \in \mathrm{TL}[\mathcal{C}_\delta]$ and $H \in \mathcal{C}_\delta$. There exist two formulas $\mathrm{F}^{sa}_H\,\psi$ and $\mathrm{P}^{sa}_H\,\psi$ of $\mathrm{TL}[\mathcal{C}_\delta]$ such that for all $w \in K_\ell$ and all $i \in \mathsf{Pos}(w)$, the two following properties hold:*

- *$w,i \models \mathrm{F}^{sa}_H\,\psi$ if and only if there exists $j \in \mathsf{Pos}(w)$ such that $j > i$, $w,j \models \psi$, $w(i,j) \in H$ and all positions $h \in \mathsf{Pos}(w)$ such that $i < h < j$ are safe.*
- *$w,i \models \mathrm{P}^{sa}_H\,\psi$ if and only if there exists $j \in \mathsf{Pos}(w)$ such that $j < i$, $w,j \models \psi$, $w(j,i) \in H$ and all positions $h \in \mathsf{Pos}(w)$ such that $j < h < i$ are safe.*

Proof. We begin by characterizing infixes containing only safe positions. Let $w \in K_\ell$ and $i,j \in \mathsf{Pos}(w)$ be such that $i < j$. We prove that the following two properties are equivalent:

1. All positions $h \in \mathsf{Pos}(w)$ such that $i < h < j$ are safe.
2. Either $\delta(w(i,j)) = 1_N$ or $d_{\mathcal{J}}(w(i,j)\sigma_\ell(w,j)) \leq \ell$.

Assume first that all positions $h \in \mathsf{Pos}(w)$ such that $i < h < j$ are safe. If $i+1 = j$, then $w(i,j) = \varepsilon$, whence $\delta(w(i,j)) = 1_N$. Assume now that $i+1 < j$. Observe that $w(i,j)\sigma_\ell(w,j)$ belongs to K_ℓ since it is an infix of $w \in K_\ell$. Moreover, since $i+1 < j$, there exists at least one $h \in \mathsf{Pos}(w)$ such that $i < h < j$. Combined with the assumption that all such positions h are safe, this implies that for every $x,y,z \in B^*$ such that $xyz = w(i,j)\sigma_\ell(w,j)$ and $|y| \leq \ell+1$, we have $d_{\mathcal{J}}(y) \leq \ell$. Therefore, Lemma 24 entails that $d_{\mathcal{J}}(w(i,j)\sigma_\ell(w,j)) \leq \ell$, as desired.

Conversely, assume that either $\delta(w(i,j)) = 1_N$ or $d_{\mathcal{J}}(w(i,j)\sigma_\ell(w,j)) \leq \ell$. We start with the latter case. Since $w(i,j)\sigma_\ell(w,j) \in K_\ell$, Lemma 24 implies that for every $x,y,z \in B^*$ such that $xyz = w(i,j)\sigma_\ell(w,j)$ and $|y| \leq \ell+1$, we have $d_{\mathcal{J}}(y) \leq \ell$. In particular, it follows that every $h \in \mathsf{Pos}(w)$ such that $i < h < j$ is safe. Assume now that $\delta(w(i,j)) = 1_N$. If $\ell = 0$, then $\sigma_\ell(w,j) = \varepsilon$ and we are back to the previous case. Otherwise, $\ell \geq 1$ and since $w \in K_\ell$, the fact that $\delta(w(i,j)) = 1_N$ yields $w(i,j) = \varepsilon$, which completes the argument.

We are now ready to complete the proof of the lemma. Let $\psi \in \mathrm{TL}[\mathcal{C}_\delta]$ and $H \in \mathcal{C}_\delta$. For every $r \in N$, we let $H_r = H \cap \delta^{-1}(r)$ and $U_r = \{u \in B^* \mid |u| \leq \ell \text{ and } d_{\mathcal{J}}(r\delta(u)) \leq \ell\}$. Observe that $H_r \in \mathcal{C}_\delta$. Now, in view of the preliminary result, it suffices to define,

$$\mathrm{F}^{sa}_H\,\psi = \mathrm{F}_{H_{1_N}}\,\psi \vee \bigvee_{r\in N}\bigvee_{u\in U_r} \mathrm{F}_{H_r}\,(\pi_{\ell,u} \wedge \psi) \quad\text{and}\quad \mathrm{P}^{sa}_H\,\psi = \mathrm{P}_{H_{1_N}}\,\psi \vee \bigvee_{r\in N}\bigvee_{u\in U_r} (\pi_{\ell,u} \wedge \mathrm{P}_{H_r}\,\psi)\,.$$

This completes the proof. ◀

Pointed decomposition. Let $w \in K_\ell$ and let $0 = i_0 < i_1 < \cdots < i_n < i_{n+1} = |w|+1$ be all the pointed positions of w. The *pointed decomposition* of w is the decomposition $w = w_0 b_1 w_1 \cdots b_n w_n$ where the highlighted letters $b_1,\ldots,b_n \in B$ are those carried by the pointed positions $i_1,\ldots,i_n$. For $0 \leq j \leq n$, we associate the word $f(w,i_j) = w_j$ to the pointed position i_j. Moreover, we define a new word $\widehat{w} \in B^*$ built from the suffix $b_1 w_1 \cdots b_n w_n$. For $1 \leq j \leq n$, let $(t_j,q_j) = (\beta(b_j w_j), \delta(b_j w_j)) \in P$. By definition of β and δ, we know that there is a letter $b_{t_j,q_j} \in B$ such that $(\beta(b_{t_j,q_j}), \delta(b_{t_j,q_j})) = (t_j,q_j)$. We let $\widehat{w} = b_{t_1,q_1}\cdots b_{t_n,q_n}$. Note that by definition, $\beta(b_1 w_1 \cdots b_n w_n) = \beta(\widehat{w})$ and $\delta(b_1 w_1 \cdots b_n w_n) = \delta(\widehat{w})$. Finally, we define a surjective map $i \mapsto \mu(i)$ associating a position $\mu(i) \in \mathsf{Pos}(\widehat{w})$ to each *pointed* position $i \in \mathsf{Pos}(w)$: for $0 \leq j \leq n+1$, we let $\mu(i_j) = j$. We complete this definition with a key property. For every pointed position $i \in \{0\} \cup \mathsf{Pos}_c(w)$, one can compute the images of the word $f(w,i)$ under β and δ with a $\mathrm{TL}[\mathcal{C}_\delta]$ formula. This is where we use Proposition 25.

▶ **Lemma 30.** *Let $(t,r) \in M \times N$. There exists $\Gamma_{t,r} \in \mathrm{TL}[\mathcal{C}_\delta]$ such that for all $w \in K_\ell$ and all pointed positions $i \in \{0\} \cup \mathsf{Pos}_c(w)$, we have $w,i \models \Gamma_{t,r} \Leftrightarrow \beta(f(w,i)) = t$ and $\delta(f(w,i)) = r$.*

Proof. First observe that by definition, if $w \in K_\ell$ and $i \in \{0\} \cup \mathsf{Pos}_c(w)$ is pointed, the infix $f(w,i)$ contains only *safe* positions. Hence, for every $x,y,z \in B^*$ such that $f(w,i) = xyz$ and $|y| \leq \ell+1$, we have $d_{\mathcal{J}}(y) \leq \ell$. By Lemma 24, it follows that $d_{\mathcal{J}}(f(w,i)) \leq \ell$. Therefore, if $d_{\mathcal{J}}(r) > \ell$, then $\delta(f(w,i))$ cannot be equal to r, and it suffices to define $\Gamma_{t,r} = \bot$.

We now assume that $d_{\mathcal{J}}(r) \leq \ell$. Proposition 25 implies that $K_\ell \cap \beta^{-1}(t) \cap \delta^{-1}(r) \in \mathrm{TL}(\mathcal{C}_\delta)$. We get a formula $\psi \in \mathrm{TL}[\mathcal{C}_\delta]$ such that for every $u \in K_\ell$, we have $u,0 \models \psi$ if and only if $\beta(u) = t$ and $\delta(u) = r$. Using Lemma 29, we modify ψ so that given $w \in K_\ell$, the evaluation of ψ at a pointed position i is constrained to the infix $f(w,i)$. More precisely, we use structural induction to build two formulas $\langle\psi\rangle_{min}$ and $\langle\psi\rangle_{max}$ such that given $w \in K_\ell$, a pointed position $i \in \{0\} \cup \mathsf{Pos}_c(w)$ and $j \in \mathsf{Pos}(f(w,i))$, the two following properties hold:

- If $j \leq |f(w,i)|$, then $w, i+j \models \langle\psi\rangle_{min} \Leftrightarrow f(w,i), j \models \psi$.
- If $1 \leq j$, then $w, i+j \models \langle\psi\rangle_{max} \Leftrightarrow f(w,i), j \models \psi$.

It will then suffice to define $\Gamma_{t,r} = \langle\psi\rangle_{min}$. We only describe the construction, and leave it to the reader to check that it satisfies the above properties. Note that we use the formula $\pi \in \mathrm{TL}[\mathcal{C}_\delta]$ of Definition 28 that detects pointed positions.

For $\psi \in B \cup \{\top,\bot\}$, we let $\langle\psi\rangle_{min} = \langle\psi\rangle_{max} = \psi$. If $\psi = min$, we let $\langle\psi\rangle_{min} = \pi$ and $\langle\psi\rangle_{max} = \bot$. If $\psi = max$, we let $\langle\psi\rangle_{min} = \bot$ and $\langle\psi\rangle_{max} = \pi$. We handle Boolean operators in the expected way. For instance, we define $\langle\psi' \vee \psi''\rangle_{min} = \langle\psi'\rangle_{min} \vee \langle\psi''\rangle_{min}$, $\langle\psi' \wedge \psi''\rangle_{min} = \langle\psi'\rangle_{min} \wedge \langle\psi''\rangle_{min}$ and $\langle\neg\psi'\rangle_{min} = \neg\langle\psi'\rangle_{min}$, and similarly for $\langle\cdot\rangle_{max}$.

If $\psi = \mathrm{F}_H\,\psi'$ for $H \in \mathcal{C}_\delta$, we let $\langle\psi\rangle_{min} = \mathrm{F}^{sa}_H\,\langle\psi'\rangle_{max}$ and $\langle\psi\rangle_{max} = \neg\pi \wedge \mathrm{F}^{sa}_H\,\langle\psi'\rangle_{max}$. Symmetrically, if $\psi = \mathrm{P}_H\,\psi'$ for some $H \in \mathcal{C}_\delta$, we define $\langle\psi\rangle_{min} = \neg\pi \wedge \mathrm{P}^{sa}_H\,\langle\psi'\rangle_{min}$ and $\langle\psi\rangle_{max} = \mathrm{P}^{sa}_H\,\langle\psi'\rangle_{min}$. This concludes the inductive construction of $\langle\psi\rangle_{min}$ and $\langle\psi\rangle_{max}$ and the proof of the proposition. ◀

Construction of the formulas $\varphi_{\ell,s}$. We are ready to complete the proof of Proposition 26. For every $s \in M$, we build a formula $\zeta_s \in \mathrm{TL}[\mathcal{C}_\delta]$ such that for every $w \in K_\ell$, we have $w, 0 \models \zeta_s \Leftrightarrow \beta(\widehat{w}) = s$. Given $s \in M$, it will then suffice to define $\varphi_{\ell,s} \in \mathrm{TL}[\mathcal{C}_\delta]$ as follows:

$$\varphi_{\ell,s} = \bigvee_{\{(s_1,s_2)\in M^2 \mid s_1 s_2 = s\}} \Big(\Big(\bigvee_{r\in N} \Gamma_{s_1,r}\Big) \wedge \zeta_{s_2}\Big).$$

Indeed, it is straightforward that for every word $w \in K_\ell$, we have $\beta(w) = \beta(f(w,0))\beta(\widehat{w})$. Consequently, by definition of $\varphi_{\ell,s}$, we get $w, 0 \models \varphi_{\ell,s} \Leftrightarrow \beta(f(w,0))\beta(\widehat{w}) = s \Leftrightarrow \beta(w) = s$ for all $w \in K_\ell$, which concludes the proof of Proposition 26. We now concentrate on building ζ_s. This is where we use induction in Proposition 26. Indeed, we have the following lemma.

▶ **Lemma 31.** *For every $w \in K_\ell$, we have $\widehat{w} \in K_{\ell+1}$.*

Proof. Let $k \leq \ell + 1$ and $x, y, z \in B^*$ such that $\widehat{w} = xyz$ and $|y| = k$. We have to prove that $d_{\mathcal{J}}(y) \geq k$. Let $w = w_0 b_1 w_1 \cdots b_n w_n$ be the pointed decomposition of w. By definition of $\widehat{w}$, we have $\delta(y) = \delta(b_h w_h \cdots b_{h+k-1} w_{h+k-1})$ for some $h \leq n$. Let $u = b_h w_h \cdots b_{h+k-1} w_{h+k-1}$. We have to show that $d_{\mathcal{J}}(y) = d_{\mathcal{J}}(u) \geq k$. Clearly, $|u| \geq k$. Hence, if $k \leq \ell$, the hypothesis that $w \in K_\ell$ yields $d_{\mathcal{J}}(u) \geq k$. Otherwise, $k = \ell + 1$. Thus, $|u| \geq \ell + 1$ and since the position labeled by b_h in w is pointed, this yields $d_{\mathcal{J}}(u) \geq \ell + 1$. In both cases, we get $d_{\mathcal{J}}(y) \geq k$. ◀

Let $s \in M$. In view of Lemma 31, induction on $|N| - \ell$ in Proposition 26 yields a $\mathrm{TL}[\mathcal{C}_\delta]$ formula ψ_s such that for every $w \in K_\ell$, we have $\widehat{w}, 0 \models \psi_s \Leftrightarrow \beta(\widehat{w}) = s$. Thus, it now suffices to prove that for every $\psi \in \mathrm{TL}[\mathcal{C}_\delta]$, there exists a formula $\lfloor\psi\rfloor \in \mathrm{TL}[\mathcal{C}_\delta]$ such that for every $w \in K_\ell$ and every pointed position $i \in \mathsf{Pos}(w)$, we have $w, i \models \lfloor\psi\rfloor \Leftrightarrow \widehat{w}, \mu(i) \models \psi$. It will then follow, for $i = 0$, that $w, 0 \models \lfloor\psi_s\rfloor \Leftrightarrow \beta(\widehat{w}) = s$, meaning that we can define $\zeta_s = \lfloor\psi_s\rfloor$.

We construct $\lfloor\psi\rfloor$ by structural induction on ψ. If $\psi \in \{min, max, \top, \bot\}$, we let $\lfloor\psi\rfloor = \psi$. Suppose now that $\psi = b_{t,q} \in B$ for $(t,q) \in P$. Thus, when evaluated in w at a pointed position i carrying a "b", we want $\lfloor\psi\rfloor$ to check that $\beta(b)\beta(f(w,i)) = t$ and $\delta(b)\delta(f(w,i)) = q$. Let $T = \big\{(b,t',q') \in B \times M \times N \mid \beta(b)t' = t \text{ and } \delta(b)q' = q\big\}$. Using the formulas $\Gamma_{t',q'}$ from Lemma 30, we define $\psi = \bigvee_{(b,t',q')\in T} (b \wedge \Gamma_{t',q'})$. Boolean operators are handled as expected. It remains to deal with temporal modalities, *i.e.*, the case where there exists $H \in \mathcal{C}_\delta$ such that $\psi = \mathrm{F}_H\,\psi'$ or $\psi = \mathrm{P}_H\,\psi'$. For every $b \in B$, let $F_b = \big\{r \in N \mid \delta(b)r \in \delta(H)\big\}$. We define:

$$\lfloor \mathrm{F}_H\,\psi' \rfloor \stackrel{\text{def}}{=} \begin{cases} \mathrm{F}^{sa}_{B^*}\,(\pi \wedge (\bigvee_{b\in B} (b \wedge \mathrm{F}_{\delta^{-1}(F_b)}\,(\pi \wedge \lfloor\psi'\rfloor)))) & \text{if } \varepsilon \notin H, \\ \mathrm{F}^{sa}_{B^*}\,(\pi \wedge (\bigvee_{b\in B} (b \wedge \mathrm{F}_{\delta^{-1}(F_b)}\,(\pi \wedge \lfloor\psi'\rfloor)) \vee \lfloor\psi'\rfloor)) & \text{if } \varepsilon \in H. \end{cases}$$

$$\lfloor \mathrm{P}_H\,\psi' \rfloor \stackrel{\text{def}}{=} \begin{cases} \bigvee_{b\in B} \mathrm{P}_{\delta^{-1}(F_b)}\,(\pi \wedge b \wedge \mathrm{P}^{sa}_{B^*}\,(\pi \wedge \lfloor\psi'\rfloor)) & \text{if } \varepsilon \notin H, \\ \mathrm{P}^{sa}_{B^*}\,(\pi \wedge \lfloor\psi'\rfloor) \vee \bigvee_{b\in B} \mathrm{P}_{\delta^{-1}(F_b)}\,(\pi \wedge b \wedge \mathrm{P}^{sa}_{B^*}\,(\pi \wedge \lfloor\psi'\rfloor)) & \text{if } \varepsilon \in H. \end{cases}$$

We give an intuition when $\psi = \mathrm{F}_H\,\psi'$ and $\varepsilon \notin H$. Let $w_0 b_1 w_1 \cdots b_n w_n$ be the pointed decomposition of w and $\widehat{w} = b'_1 \cdots b'_n$. Let $i_k \in \mathsf{Pos}(w)$ be the position of the distinguished b_k, so that $\mu(i_k) = k$. Now, $\widehat{w}, k \models \mathrm{F}_H\,\psi'$ when there exists $m > k$ such that $\widehat{w}, m \models \psi'$ and $b'_{k+1} \cdots b'_{m-1} \in H$. The construction ensures that $w, i_k \models \lfloor \mathrm{F}_H\,\psi' \rfloor$ when there exists $m > k$ such that $w, i_m \models \lfloor\psi'\rfloor$ and $b_{k+1} w_{k+1} \cdots b_{m-1} w_{m-1} \in H$. The purpose of using $\mathrm{F}^{sa}_{B^*}\,(\pi \wedge \dots)$ is to "jump" to b_{k+1}. The remainder checks that the next jump, to a pointed position, determines a word of $\delta^{-1}(\delta(H)) = H$. More generally, one can check that $w, i \models \lfloor\psi\rfloor \Leftrightarrow \widehat{w}, \mu(i) \models \psi$ for all $w \in K_\ell$ and all pointed positions $i \in \mathsf{Pos}(w)$. This concludes the proof.

6 Natural restrictions of generalized unary temporal logic

We turn to two natural restrictions of the classes $\mathrm{TL}(\mathcal{C})$, which were defined in [26]: the pure-future and pure-past fragments. For a class $\mathcal{C}$, we write $\mathrm{FL}[\mathcal{C}] \subseteq \mathrm{TL}[\mathcal{C}]$ for the set of all formulas that contain only *future* modalities (*i.e.*, the modalities P_L are disallowed). Symmetrically, $\mathrm{PL}[\mathcal{C}] \subseteq \mathrm{TL}[\mathcal{C}]$ is the set of all formulas in $\mathrm{TL}[\mathcal{C}]$ that contain only *past* modalities (*i.e.*, the modalities F_L are disallowed).

We now define the two associated operators $\mathcal{C} \mapsto \mathrm{FL}(\mathcal{C})$ and $\mathcal{C} \mapsto \mathrm{PL}(\mathcal{C})$. For every class $\mathcal{C}$, let $\mathrm{FL}(\mathcal{C})$ be the class consisting of all languages $L_{min}(\varphi)$ where $\varphi \in \mathrm{FL}[\mathcal{C}]$. Symmetrically, we write $\mathrm{PL}(\mathcal{C})$ for the class consisting of all languages $L_{max}(\varphi)$, with $\varphi \in \mathrm{PL}[\mathcal{C}]$.

▶ Remark 32. Note that $\mathrm{FL}[\mathcal{C}]$ formulas are evaluated at the leftmost unlabeled position whereas $\mathrm{PL}[\mathcal{C}]$ formulas are evaluated at the rightmost unlabeled position.

6.1 Connection with left and right polynomial closure

The main ideas to establish decidable characterizations for $\mathrm{FL}(\mathcal{C})$ and $\mathrm{PL}(\mathcal{C})$ follow the lines of the proof of Theorem 12. However, there are some differences. First, for the easy direction (proving that some property on $\mathcal{C}$-orbits is necessary), we have to adapt Lemma 20 to the operators $\mathcal{C} \mapsto \mathrm{FL}(\mathcal{C})$ and $\mathcal{C} \mapsto \mathrm{PL}(\mathcal{C})$. We prove these adapted properties in the extended version of this paper [29] as corollaries of results presented in [26].

The proof of the difficult direction is mostly identical to that in Theorem 12. However, there is a key difference: we have to find a substitute for Proposition 25, whose proof relied the inclusion $\mathrm{UPol}(\mathrm{BPol}(\mathcal{C})) \subseteq \mathrm{TL}(\mathcal{C})$ from Proposition 10. We replace unambiguous polynomial closure (UPol) by two variants, called *right* and *left* polynomial closure (RPol and LPol). It is shown [26] that $\mathrm{RPol}(\mathrm{BPol}(\mathcal{C})) \subseteq \mathrm{FL}(\mathcal{C})$ and $\mathrm{LPol}(\mathrm{BPol}(\mathcal{C})) \subseteq \mathrm{PL}(\mathcal{C})$ for every prevariety $\mathcal{C}$: this serves as a substitute for Proposition 10. Finally, while no simple generic characterization of the classes $\mathrm{RPol}(\mathrm{BPol}(\mathcal{C}))$ and $\mathrm{LPol}(\mathrm{BPol}(\mathcal{C}))$ are known, we are able to replace Theorem 11 by combining independent characterizations of the operators Pol and RPol (resp. Pol and LPol) from [23, 21].

We now establish a connection between the operators $\mathcal{C} \mapsto \mathrm{FL}(\mathcal{C})$ and $\mathcal{C} \mapsto \mathrm{PL}(\mathcal{C})$ and the two weaker variants RPol and LPol of unambiguous polynomial closure. Consider a marked product $L_0a_1L_1 \cdots a_nL_n$. For $1 \leq i \leq n$, we write $H_i = L_1a_1L_2 \cdots a_{i-1}L_{i-1}$ and $K_i = L_ia_{i+1}L_{i+1} \cdots a_nL_n$. We say that $L_0a_1L_1 \cdots a_nL_n$ is *right deterministic* (resp. *left deterministic*) when we have $A^*a_iK_i \cap K_i = \emptyset$ (resp. $H_ia_iA^* \cap H_i = \emptyset$) for every $i \leq n$. The *right polynomial closure* of a class $\mathcal{C}$, written $\mathrm{RPol}(\mathcal{C})$, consists of all *finite disjoint unions* of *right deterministic marked products* $L_0a_1L_1 \cdots a_nL_n$ such that $L_0, \ldots, L_n \in \mathcal{C}$ (by "disjoint" we mean that the languages in the union must be pairwise disjoint). Similarly, the *left polynomial closure* $\mathrm{LPol}(\mathcal{C})$ of $\mathcal{C}$ consists of all finite disjoint unions of *left* deterministic marked products $L_0a_1L_1 \cdots a_nL_n$ such that $L_0, \ldots, L_n \in \mathcal{C}$. While this is not immediate, it is known [21] that when the input class $\mathcal{C}$ is a prevariety, then so are $\mathrm{RPol}(\mathcal{C})$ and $\mathrm{LPol}(\mathcal{C})$.

As expected, we are interested in the "combined" operators $\mathcal{C} \mapsto \mathrm{RPol}(\mathrm{BPol}(\mathcal{C}))$ and $\mathcal{C} \mapsto \mathrm{LPol}(\mathrm{BPol}(\mathcal{C}))$. Indeed, the first one is connected to the classes $\mathrm{FL}(\mathcal{C})$ by the following result proved in [26, Proposition 5].

▶ **Proposition 33.** *For every prevariety $\mathcal{C}$, we have* $\mathrm{RPol}(\mathrm{BPol}(\mathcal{C})) \subseteq \mathrm{FL}(\mathcal{C})$.

We have the following symmetrical statement for $\mathrm{PL}(\mathcal{C})$.

▶ **Proposition 34.** *For every prevariety $\mathcal{C}$, we have* $\mathrm{LPol}(\mathrm{BPol}(\mathcal{C})) \subseteq \mathrm{PL}(\mathcal{C})$.

Propositions 33 and 34 serve as the replacement of Proposition 10 when dealing with the classes $\mathrm{FL}(\mathcal{C})$ and $\mathrm{PL}(\mathcal{C})$, respectively. It now remains to replace the generic algebraic characterization of the classes $\mathrm{UPol}(\mathrm{BPol}(\mathcal{C}))$ presented in Theorem 11. This is more tricky as no such characterization is known for the classes $\mathrm{RPol}(\mathrm{BPol}(\mathcal{C}))$ (nor for the classes $\mathrm{LPol}(\mathrm{BPol}(\mathcal{C}))$). Yet, we manage to prove a *sufficient condition* for a language to belong to $\mathrm{RPol}(\mathrm{BPol}(\mathcal{C}))$ or $\mathrm{LPol}(\mathrm{BPol}(\mathcal{C}))$ by combining results of [27] and [21]. While it does *not* characterize these classes in general, it suffices for our needs: proving that particular languages belong to $\mathrm{RPol}(\mathrm{BPol}(\mathcal{C}))$ (and therefore to $\mathrm{FL}(\mathcal{C})$ by Proposition 33) or to $\mathrm{LPol}(\mathrm{BPol}(\mathcal{C}))$ (and therefore to $\mathrm{FL}(\mathcal{C})$ by Proposition 34).

▶ **Proposition 35.** *Let $\mathcal{C}$ be a prevariety, $L \subseteq A^*$ be a regular language and $\alpha : A^* \to M$ be its syntactic morphism. Assume that α satisfies the following property:*

$$(esete)^{\omega+1} = ete(esete)^{\omega} \quad \text{for every } \mathcal{C}\text{-pair } (e,s) \in M^2 \text{ and every } t \in M. \tag{5}$$

Then, $L \in \mathrm{RPol}(\mathrm{BPol}(\mathcal{C}))$.

▶ **Proposition 36.** *Let $\mathcal{C}$ be a prevariety, $L \subseteq A^*$ be a regular language and $\alpha : A^* \to M$ be its syntactic morphism. Assume that α satisfies the following property:*

$$(esete)^{\omega+1} = (esete)^{\omega}ese \quad \text{for every } \mathcal{C}\text{-pair } (e,t) \in M^2 \text{ and every } s \in M.$$

Then, $L \in \mathrm{LPol}(\mathrm{BPol}(\mathcal{C}))$.

Since Propositions 35 and 36 are symmetrical, we only prove the first one and leave the second to the reader.

Proof of Proposition 35. We use a generic characterization of the classes $\mathrm{RPol}(\mathcal{D})$ proved in [21]. Let us first present it. For every class $\mathcal{D}$, we define a preorder $\preceq_{\mathcal{D}}$ and an equivalence $\sim_{\mathcal{D}}$ over M. Given $s,t \in M$, we let,

$$\begin{aligned} s \sim_{\mathcal{D}} t \quad &\text{if and only if} \quad s \in F \Leftrightarrow t \in F \text{ for every } F \subseteq M \text{ such that } \alpha^{-1}(F) \in \mathcal{D}, \\ s \preceq_{\mathcal{D}} t \quad &\text{if and only if} \quad s \in F \Rightarrow t \in F \text{ for every } F \subseteq M \text{ such that } \alpha^{-1}(F) \in \mathcal{D}. \end{aligned}$$

Clearly, $\preceq_{\mathcal{D}}$ is a preorder on M and $\sim_{\mathcal{D}}$ is the equivalence generated by $\preceq_{\mathcal{D}}$. When $\alpha : A^* \to M$ is the syntactic morphism of L, it is shown in [21, Theorem 4.1] that for every prevariety $\mathcal{D}$, we have $L \in \mathrm{RPol}(\mathcal{D})$ if and only if $s^{\omega+1} = ts^{\omega}$ for all $s,t \in M$ such that $s \sim_{\mathcal{D}} t$.

Hence, since $\mathrm{BPol}(\mathcal{C})$ is a prevariety, it suffices to prove that for every $s,t \in M$ such that $s \sim_{\mathrm{BPol}(\mathcal{C})} t$, we have $s^{\omega+1} = ts^{\omega}$. We fix s,t for the proof. Since $s \sim_{\mathrm{BPol}(\mathcal{C})} t$, we have $s \preceq_{\mathrm{BPol}(\mathcal{C})} t$. Moreover, let $co\text{-}\mathrm{Pol}(\mathcal{C})$ be the class consisting of all complements of languages in $\mathrm{Pol}(\mathcal{C})$ (*i.e.*, $L \in co\text{-}\mathrm{Pol}(\mathcal{C})$ if and only if $A^* \setminus L \in co\text{-}\mathrm{Pol}(\mathcal{C})$). Clearly, we have $co\text{-}\mathrm{Pol}(\mathcal{C}) \subseteq \mathrm{BPol}(\mathcal{C})$. Hence, the definition implies that $s \preceq_{co\text{-}\mathrm{Pol}(\mathcal{C})} t$

Moreover, it is shown in [27, Lemma 6.6] that $\preceq_{co\text{-}\mathrm{Pol}(\mathcal{C})}$ is the least preorder on M such that for every $x,y,q \in M$ and $e \in E(M)$, if $(e,q) \in M^2$ is a $\mathcal{C}$-pair, then $xeqey \preceq_{co\text{-}\mathrm{Pol}(\mathcal{C})} xey$ (the proof is based on the algebraic characterization of $\mathrm{Pol}(\mathcal{C})$, see [23]). This yields $s_0, \ldots, s_n \in M$ such that $s = s_0$, $t = s_n$ and, for every $i \leq n$, there exist $x,y,q \in M$ and $e \in E(M)$ such that $(e,q) \in M^2$ is a $\mathcal{C}$-pair, $s_{i-1} = xeqey$ and $s_i = xey$. We use induction on i to prove that $s^{\omega+1} = s_i s^{\omega}$ for every $i \leq n$. Since $s_n = t$, the case $i = n$ yields the desired result. When $i = 0$, it is immediate that $s^{\omega+1} = s_0 s^{\omega}$ since $s_0 = s$. Assume now that $i \geq 1$.

By induction hypothesis, we know that $s^{\omega+1} = s_{i-1}s^\omega$. Moreover, we have $x, y, q \in M$ and $e \in E(M)$ such that $(e, q) \in M^2$ is a $\mathcal{C}$-pair, $s_{i-1} = xeqey$ and $s_i = xey$. Since $(s^{\omega+1})^{\omega+2} = s^{\omega+2}$, we get $s^{\omega+2} = (xeqeys^\omega)^{\omega+2}$. Hence, we get

$$\begin{aligned} s^{\omega+2} &= x\ (eqeys^\omega xe)^{\omega+1}\ eqeys^\omega \\ &= x\ eys^\omega xe(eqeys^\omega xe)^\omega\ eqeys^\omega \quad \text{by (5) since } (e,q) \text{ is a } \mathcal{C}\text{-pair} \\ &= xeys^\omega(xeqeys^\omega)^{\omega+1}. \end{aligned}$$

This yields, $s^{\omega+2} = s_i s^\omega (s_{i-1}s^\omega)^{\omega+1} = s_i s^\omega (s^{\omega+1})^{\omega+1} = s_i s^{\omega+1}$. It now remains to multiply by $s^{\omega-1}$ on the right to get $s^{\omega+1} = s_i s^\omega$, as desired. ◀

6.2 Statements

The classes $\mathrm{FL}(\mathcal{C})$ and $\mathrm{PL}(\mathcal{C})$ admit algebraic characterizations similar to that of $\mathrm{TL}(\mathcal{C})$. We reuse the $\mathcal{C}$-orbits introduced in Section 3. Let $\mathcal{X} \in \{\mathcal{L}, \mathcal{R}, \mathcal{J}\}$ be one the Green relations defined in Section 2. A monoid M is *$\mathcal{X}$-trivial* when $s\ \mathcal{X}\ t$ implies $s = t$ for all $s, t \in M$. It is standard and simple to verify that a finite monoid M is $\mathcal{R}$-trivial (resp. $\mathcal{L}$-trivial) if and only if for all $s, t \in M$, we have $(st)^\omega s = (st)^\omega$ (resp. $t(st)^\omega = (st)^\omega$), see [17, 20] for a proof. We are now able to present the two symmetrical characterizations of $\mathrm{FL}(\mathcal{C})$ and $\mathrm{PL}(\mathcal{C})$.

▶ **Theorem 37.** *Let $\mathcal{C}$ be a prevariety, $L \subseteq A^*$ be a regular language and $\alpha : A^* \to M$ be its syntactic morphism. The two following properties are equivalent:*

1. $L \in \mathrm{FL}(\mathcal{C})$.
2. *Every $\mathcal{C}$-orbit for α is $\mathcal{L}$-trivial.*

▶ **Theorem 38.** *Let $\mathcal{C}$ be a prevariety, $L \subseteq A^*$ be a regular language and $\alpha : A^* \to M$ be its syntactic morphism. The two following properties are equivalent:*

1. $L \in \mathrm{PL}(\mathcal{C})$.
2. *Every $\mathcal{C}$-orbit for α is $\mathcal{R}$-trivial.*

Since $\mathrm{FL}(\mathcal{C})$ and $\mathrm{PL}(\mathcal{C})$ are symmetrical, it is natural to consider a third class denoted $\mathrm{FL}(\mathcal{C}) \cap \mathrm{PL}(\mathcal{C})$. It consists of all languages belonging simultaneously to $\mathrm{FL}(\mathcal{C})$ and $\mathrm{PL}(\mathcal{C})$. It is standard that the finite monoids which are both $\mathcal{L}$-trivial and $\mathcal{R}$-trivial are exactly the $\mathcal{J}$-trivial monoids (see [17, 20]). This yields the following corollary of Theorems 37 and 38.

▶ **Corollary 39.** *Let $\mathcal{C}$ be a prevariety, $L \subseteq A^*$ be a regular language and $\alpha : A^* \to M$ be its syntactic morphism. The two following properties are equivalent:*

1. $L \in \mathrm{FL}(\mathcal{C}) \cap \mathrm{PL}(\mathcal{C})$.
2. *Every $\mathcal{C}$-orbit for α is $\mathcal{J}$-trivial.*

Recall that given a regular language $L \subseteq A^*$ as input, its syntactic morphism $\alpha : A^* \to M$ can be computed. Moreover, Lemma 5 implies that all $\mathcal{C}$-orbits for α can be computed when $\mathcal{C}$-separation is decidable. Thus, the three above characterizations yield the following corollary.

▶ **Corollary 40.** *Let $\mathcal{C}$ be a prevariety with decidable separation. Then, the classes* $\mathrm{FL}(\mathcal{C})$, $\mathrm{PL}(\mathcal{C})$ *and* $\mathrm{FL}(\mathcal{C}) \cap \mathrm{PL}(\mathcal{C})$ *have decidable membership.*

We leave the proof of Theorem 37 for the extended version of this paper [29] (it omits the proof of Theorem 38, which is symmetrical).

7 Conclusion

We presented generic characterizations of the classes $\mathrm{TL}(\mathcal{C})$, $\mathrm{FL}(\mathcal{C})$ and $\mathrm{PL}(\mathcal{C})$. While the proofs are complex, the statements are simple and elegant. They generalize in a natural way all known characterizations of classes built with these operators. As a corollary, we obtained that if $\mathcal{C}$ is a prevariety with decidable *separation*, then all classes $\mathrm{TL}(\mathcal{C})$, $\mathrm{FL}(\mathcal{C})$ and $\mathrm{PL}(\mathcal{C})$ have decidable *membership*.

The next step is to tackle *separation*. This question is difficult in general, but it is worth looking at *particular* input classes. For instance, one can define the TL-hierarchy of basis $\mathcal{C}$: level 0 is $\mathrm{TL}_0(\mathcal{C}) = \mathcal{C}$ and level $n \geq 1$ is $\mathrm{TL}_n(\mathcal{C}) = \mathrm{TL}(\mathrm{TL}_{n-1}(\mathcal{C}))$. It can be shown that the hierarchies of bases $\mathrm{ST} = \{\emptyset, A^*\}$ and $\mathrm{DD} = \{\emptyset, \{\varepsilon\}, A^+, A^*\}$ are strict. Thus, since $\mathrm{BPol}(\mathcal{C}) \subseteq \mathrm{TL}(\mathcal{C})$, they both classify the star-free languages (or equivalently the languages definable in full linear temporal logic). We already know that in both hierarchies, membership is decidable for levels 1 (*i.e.*, the variants TL and TLX of unary temporal logic) and 2 (which were studied in [15]). The results of the present paper show that if $\mathrm{TL}_2(\mathrm{ST})$ and $\mathrm{TL}_2(\mathrm{DD})$ have decidable separation, then $\mathrm{TL}_3(\mathrm{ST})$ and $\mathrm{TL}_3(\mathrm{DD})$ would have decidable membership.

Finally, all other major operators have language-theoretic counterparts. Another possible follow-up is to look for such a definition for all three operators $\mathcal{C} \mapsto \mathrm{TL}(\mathcal{C}), \mathrm{FL}(\mathcal{C})$ and $\mathrm{PL}(\mathcal{C})$.

References

1 Mustapha Arfi. Polynomial operations on rational languages. In *Proceedings of the 4th Annual Symposium on Theoretical Aspects of Computer Science, STACS'87*, Lecture Notes in Computer Science, pages 198–206. Springer, 1987.

2 David A. Mix Barrington, Kevin Compton, Howard Straubing, and Denis Thérien. Regular languages in NC^1. *Journal of Computer and System Sciences*, 44(3):478–499, 1992.

3 Danièle Beauquier and Jean-Éric Pin. Languages and scanners. *Theoretical Computer Science*, 84(1):3–21, 1991.

4 Janusz A. Brzozowski and Imre Simon. Characterizations of locally testable events. *Discrete Mathematics*, 4(3):243–271, 1973.

5 Luc Dartois and Charles Paperman. Two-variable first order logic with modular predicates over words. In *Proceedings of the 30th International Symposium on Theoretical Aspects of Computer Science, STACS'13*, Leibniz International Proceedings in Informatics (LIPIcs), pages 329–340. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2013.

6 Volker Diekert, Paul Gastin, and Manfred Kufleitner. A survey on small fragments of first-order logic over finite words. *International Journal of Foundations of Computer Science*, 19(3):513–548, 2008.

7 Volker Diekert, Martin Horsch, and Manfred Kufleitner. On first-order fragments for Mazurkiewicz traces. *Fundamenta Informaticae*, 80(1-3):1–29, 2007.

8 Volker Diekert and Manfred Kufleitner. Fragments of first-order logic over infinite words. *Theory of Computing Systems (ToCS)*, 48(3):486–516, 2011.

9 Kousha Etessami, Moshe Y. Vardi, and Thomas Wilke. First-order logic with two variables and unary temporal logic. *Information and Computation*, 179(2):279–295, 2002.

10 James Alexander Green. On the structure of semigroups. *Annals of Mathematics*, 54(1):163–172, 1951.

11 Hans W. Kamp. *Tense Logic and the Theory of Linear Order*. Phd thesis, Computer Science Department, University of California at Los Angeles, USA, 1968.

12 Robert Knast. A semigroup characterization of dot-depth one languages. *RAIRO – Theoretical Informatics and Applications*, 17(4):321–330, 1983.

13 Andreas Krebs, Kamal Lodaya, Paritosh K. Pandya, and Howard Straubing. Two-variable logic with a between relation. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS'16*, pages 106–115, 2016.

14 Andreas Krebs, Kamal Lodaya, Paritosh K. Pandya, and Howard Straubing. An algebraic decision procedure for two-variable logic with a between relation. In *27th EACSL Annual Conference on Computer Science Logic, CSL'18*, Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018.

15 Andreas Krebs, Kamal Lodaya, Paritosh K. Pandya, and Howard Straubing. Two-variable logics with some betweenness relations: Expressiveness, satisfiability and membership. *Logical Methods in Computer Science*, Volume 16, Issue 3, 2020.

16 Robert McNaughton and Seymour A. Papert. *Counter-Free Automata*. MIT Press, 1971.

17 Jean-Éric Pin. *Varieties of Formal Languages*. North Oxford Academic, 1986.

18 Jean-Éric Pin and Pascal Weil. Polynomial closure and unambiguous product. *Theory of Computing Systems*, 30(4):383–422, 1997.

19 Jean-Éric Pin. An explicit formula for the intersection of two polynomials of regular languages. In *Proceedings of the 17th International Conference on Developments in Language Theory, DLT'13*, volume 7907 of *Lecture Notes in Computer Science*, pages 31–45. Springer, 2013.

20 Jean-Éric Pin. Mathematical foundations of automata theory. Lecture notes, in preparation, 2022. URL: `https://www.irif.fr/~jep/PDF/MPRI/MPRI.pdf`.

21 Thomas Place. The amazing mixed polynomial closure and its applications to two-variable first-order logic. In *Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS'22*, 2022.

22 Thomas Place and Marc Zeitoun. Separating without any ambiguity. In *45th International Colloquium on Automata, Languages, and Programming, ICALP'18*, Leibniz International Proceedings in Informatics (LIPIcs), pages 137:1–137:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018.

23 Thomas Place and Marc Zeitoun. Generic results for concatenation hierarchies. *Theory of Computing Systems (ToCS)*, 63(4):849–901, 2019. Selected papers from CSR'17.

24 Thomas Place and Marc Zeitoun. Going higher in first-order quantifier alternation hierarchies on words. *Journal of the ACM*, 66(2):12:1–12:65, 2019.

25 Thomas Place and Marc Zeitoun. On all things star-free. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming, ICALP'19*, Leibniz International Proceedings in Informatics (LIPIcs), pages 126:1–126:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019.

26 Thomas Place and Marc Zeitoun. How many times do you need to go back to the future in unary temporal logic? In *Proceedings of the 15th Latin American Symposium on Theoretical Informatics, LATIN'22*, Lecture Notes in Computer Science. Springer, 2022.

27 Thomas Place and Marc Zeitoun. All about unambiguous polynomial closure. *TheoretiCS*, 2(11):1–74, 2023. `doi:10.46298/theoretics.23.11`.

28 Thomas Place and Marc Zeitoun. Closing star-free closure, 2023. `arXiv:2307.09376`.

29 Thomas Place and Marc Zeitoun. A generic characterization of generalized unary temporal logic and two-variable first-order logic, 2023. `arXiv:2307.09349`.

30 Marcel Paul Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8(2):190–194, 1965.

31 Marcel Paul Schützenberger. Sur certaines opérations de fermeture dans les langages rationnels. *Symposia Mathematica*, XV:245–253, 1975.

32 Marcel Paul Schützenberger. Sur le produit de concaténation non ambigu. *Semigroup Forum*, 13:47–75, 1976.

33 Imre Simon. Piecewise testable events. In *Proceedings of the 2nd GI Conference on Automata Theory and Formal Languages*, pages 214–222. Springer, 1975.

34 Howard Straubing. Aperiodic homomorphisms and the concatenation product of recognizable sets. *Journal of Pure and Applied Algebra*, 15(3):319–327, 1979.

35 Pascal Tesson and Denis Thérien. Diamonds are forever: The variety DA. In *Semigroups, Algorithms, Automata and Languages*, pages 475–500. World Scientific, 2002.

36 Denis Thérien and Thomas Wilke. Over words, two variables are as powerful as one quantifier alternation. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing, STOC'98*, pages 234–240. ACM, 1998.

Concurrent Stochastic Lossy Channel Games

Daniel Stan
EPITA, Le Kremlin-Bicêtre, France

Muhammad Najib
Heriot-Watt University, Edinburgh, UK

Anthony Widjaja Lin
University of Kaiserslautern-Landau, Germany
Max-Planck Institute for Software Systems, Kaiserslautern, Germany

Parosh Aziz Abdulla
Uppsala University, Sweden

Abstract

Concurrent stochastic games are an important formalism for the *rational verification* of probabilistic multi-agent systems, which involves verifying whether a temporal logic property is satisfied in some or all game-theoretic equilibria of such systems. In this work, we study the rational verification of probabilistic multi-agent systems where agents can cooperate by communicating over *unbounded lossy channels*. To model such systems, we present *concurrent stochastic lossy channel games* (CSLCG) and employ an equilibrium concept from cooperative game theory known as the *core*, which is the most fundamental and widely studied cooperative equilibrium concept. Our main contribution is twofold. First, we show that the rational verification problem is undecidable for systems whose agents have almost-sure LTL objectives. Second, we provide a decidable fragment of such a class of objectives that subsumes almost-sure reachability and safety. Our techniques involve reductions to solving infinite-state zero-sum games with conjunctions of qualitative objectives. To the best of our knowledge, our result represents the first decidability result on the rational verification of stochastic multi-agent systems on infinite arenas.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory; Theory of computation → Verification by model checking; Theory of computation → Concurrency; Theory of computation → Solution concepts in game theory

Keywords and phrases concurrent, games, stochastic, lossy channels, wqo, finite attractor property, cooperative, core, Nash equilibrium

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.46

Related Version *Full Version*: `http://arxiv.org/abs/2311.17037`

Funding *Anthony Widjaja Lin*: supported by European Research Council under European Union's Horizon research and innovation programme (grant agreement no 501100000781).

Acknowledgements We wish to thank Richard Mayr and all anonymous reviewers for their useful feedback.

1 Introduction

Rational verification concerns the problem of checking which temporal logic properties will be satisfied in game-theoretic equilibria of a multi-agent system, that is, the stable collective behaviours that arise assuming that agents choose strategies/policies rationally in order to achieve their goals [27, 1]. The usual approach to rational verification is to model multi-agent systems as concurrent games [27, 31]. This involves converting a multi-agent system into a game where agents are represented by a collection of independent, self-interested players in a finite-state environment. The game is played over an infinite number of rounds, with each

player/agent (we use these terms interchangeably throughout the paper) choosing an action to perform in each round. Each player's goal is typically given by a temporal logic formula, which the player aims to satisfy. The temporal logic formula may or may not be satisfied by the infinite plays generated from the game, assuming that the players act rationally to achieve their goals.

In this paper, unlike much of previous work in rational verification, we consider systems that give rise to games with *probabilistic transitions* and *infinitely many states*. In particular, we focus on systems that can naturally be modelled by *stochastic lossy channel games* [8] in the *multi-player* and *concurrent* setting, where there are $n \geq 2$ players who can make concurrent moves. Our setting generalises the two-player turn-based framework presented in [8]. We call this model *Concurrent Stochastic Lossy Channel Games* (CSLCG). This model can be used to analyse a wide class of systems that communicate through potentially unreliable FIFO channels, such as communication networks, timed protocols, distributed systems, and memory systems [2, 9, 10, 4]. Stochasticity can be used to represent uncertainty in both the environment (e.g., branching and message losses) and the behaviour of agents. Incorporating such uncertainty is desirable from a practical standpoint, as real-world systems are expected to operate correctly even when communication is not perfect and agents' behaviour is not deterministic. In the context of memory systems, stochasticity is used as a fairness condition that prevents unrealistic scenarios where the shared memory is never updated by the processes [3, 34].

Given the possibility of communication, albeit imperfect, among agents, it seems quite natural to assume that some form of cooperation may arise in games. Thus, a relevant and fundamental question within the rational verification framework is: *"What temporal logic property is satisfied by the rational cooperation that emerges in such a setting?"* To address this question, we consider an equilibrium concept from cooperative game theory called the *core* [14, 41, 30], which is the most fundamental and widely-studied cooperative equilibrium concept. With this concept, the standard assumption is that there exists some mechanism[1] that the players in a game can use to make *binding agreements*. These binding agreements enable players to cooperate and work in teams/coalitions, providing a way to eliminate undesirable equilibria that may arise in non-cooperative settings [30, 29]. We illustrate that this is also true in our setting in Example 14. Despite using a cooperative equilibrium concept, we emphasize that players are still self-interested, meaning they rationally pursue their individual goals. As such, the games we consider in this work are general-sum games instead of strictly positive-sum games, which are purely cooperative.

Contributions. We study the rational verification problem in CSLCG with the core as the key equilibrium concept. It is shown in [30] that the core of a game with *qualitative objectives* can never be empty, which also applies to the setting considered in this paper. Thus, two relevant decision problems pertaining to rational verification in the present work are E-Core and A-Core. E-Core asks whether there *exists* a strategy profile in the core satisfying a given property Γ, whereas A-Core asks whether *all* profiles in the core satisfy Γ. We first show that these problems are undecidable for games in which the players' objectives and property Γ are almost-sure LTL formulae, i.e., of the form $\mathrm{AS}(\varphi)$, where φ is a LTL formula. We consider LTL *with regular valuations* [24] where the set of *states/configurations* satisfying an atomic proposition is represented by a regular language. Then, for our main

[1] Such mechanism is assumed to be exogenous (e.g., via contracts) and beyond the scope of the present paper.

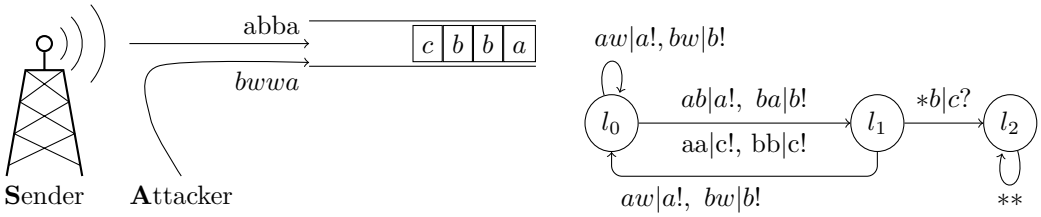


Figure 1 A simple adversarial transmission system from Example 1 (left) and its graphical representation as a 2-player arena (right). The special action $*$ is a shorthand for any possible action. The resulting operation f on the channel is written after $|$ and omitted if $f = nop$.

contribution, we show the following: when players' goals are given as *almost-sure reachability* or *almost-sure safety* objectives, and the property Γ is given as *almost-sure reachability*, *almost-sure safety*, or *almost-sure Büchi*, the problems of E-Core and A-Core become decidable. Our decidability proof is obtained via a reduction to *concurrent* 2.5-player[2] lossy channel games with conjunction of objectives. This approach differs from previous work in two ways. First, our reduction considers *concurrent* plays, in contrast to *turn-based* 2.5-player games considered by [8]. Second, we do not assume finite-memory strategies, as opposed to finite-memory assumption in [7, 16]. This is because finite-memory strategies do not ensure determinacy [35]: it is possible that none of the players has a winning strategy in a given concurrent 2.5-player game, even in the finite state case with simple objectives [23]. Therefore, general strategies (which may require infinite memory) are required in equilibrium concepts such as the core, where players (or coalitions) may try to satisfy their objectives while simultaneously preventing other players from achieving theirs. However, the main challenge in using these strategies in the infinite state case is the issue of representation. To address this, we provide a novel encoding of strategies in our proof of decidability. To our knowledge, this is the first decidability result on the rational verification of stochastic multi-player games with infinite-state arenas.

▶ **Example 1.** To illustrate the model, we consider a simple transmission system depicted in Figure 1 throughout the article. In this example, *Sender* (player 1) tries to emit some message of either type a or b. *Attacker* (player 2) is trying to scramble the communication by concurrently choosing the same message type (action a or b). Moreover, Attacker cannot scramble the communication two times in a row and has to wait (action w) otherwise. The CSLCG arena is depicted on the right with the corresponding transitions and an extra location, reachable by a unilateral decision of player 1 by reading a c-letter from the channel, which is possible only in case of a successful scrambling. Note that although the game structure is deterministic in this example, some stochastic behaviour will still appear both from message losses and from players' strategies, which are played concurrently. As a more concrete example of Attacker's objective, one could specify the condition "reaching l_2 almost-surely, while not having more than 3 queued messages with positive probability". Note that this is a conjunction of reachability and safety conditions over locations and regular sets of channel configurations.

[2] Henceforth, we use the usual terms 1.5-player and 2.5-player games for, respectively, one-player and two-player stochastic games.

Related Work. As already mentioned above, the most relevant work w.r.t. verification of CSLCG is [8], which shows decidability of *two-player turn-based* stochastic lossy channel games with almost-sure reachability or almost-sure Büchi objectives. This work was extended to parity conditions in [7], where decidability can be shown *assuming finite-memory strategies*; otherwise, it is already undecidable for 1.5-player games over lossy channel systems with almost-sure co-Büchi objectives [16]. We are not aware of any work on rational verification of concurrent stochastic games over infinite arenas. [26] studies verification of the core in a probabilistic setting, while [11] presents *Probabilistic Strategy Logic*, which can be used to characterize the core. However, both of these works are in a finite state setting only. Without probability, we mention the work [37, 22] on concurrent (deterministic) pushdown games with multiple players. In particular, ATL* model checking is decidable in such games, which allows one to reason about the core. Additionally, there has been work on pushdown *module-checking*, which provides some element of non-determinism through an (external) environment. [12] examines the imperfect information setting, while [21] studies multi-agent systems with ATL* specifications. Note that lossy channel systems, which are the focus of our work, are inherently different from the models considered in these studies.

Organization. Section 2 introduces preliminary definitions and notations. Section 3 describes concurrent lossy channel games and the special case of 2.5-player zero-sum games. Section 4 presents a characterization of the core, the problems E-Core and A-Core, a procedure to solve them, and an undecidability result of E-Core and A-Core. Section 5 addresses the computability of winning regions for concurrent 2.5-player zero-sum games and provides algorithms to compute such regions. Section 6 studies the conjunction of objectives, while Section 7 presents our main result on the decidability of E-Core and A-Core. Finally, Section 8 concludes with a discussion and future work.

2 Preliminaries

For a finite alphabet Σ, the set of finite sequences, called *words*, is written Σ^*. Given two words $u, v \in \Sigma^*$, we write $u \cdot v$ for their concatenation and extend this notation to sets of words. Given $L \subseteq \Sigma^*$, L^+ denotes the smallest set containing L and closed under concatenation and $L^* = \{\epsilon\} \cup L^+$ with ϵ the empty word. The class of *regular languages* is the smallest class containing Σ, closed under difference, union, Kleene star and concatenation. We refer to [33] for further references about regular expressions and their link to automata theory.

Let S denote a countable set, for example $S = \Sigma^*$. A *well-quasi-ordering* [25] (wqo) $\preceq$ over S is a quasi-ordering (i.e. reflexive and transitive binary relation) such that any infinite sequence $(s_i)_{i\in\mathbb{N}}$ of elements of S contains an increasing pair $i < j$ such that $s_i \preceq s_j$. As an example, Higman's lemma [32] states that the *sub-word ordering* $\preceq$ defined below is a wqo over Σ^*:

▶ **Definition 2.** *For any $w, w' \in \Sigma^*$, $w \sqsubseteq w'$ if w can be written $w = w_0 \cdot w_1 \cdots w_n$ and $w' \in \Sigma^* \cdot \{w_0\} \cdot \Sigma^* \cdots \Sigma^* \cdot \{w_n\} \cdot \Sigma^*$.*

A subset $U \subseteq S$ is *upward-closed* (UC) if for every $s \preceq t$ such that $s \in U$, we also have $t \in U$. Any UC set can be uniquely represented by its set of minimal elements, which is finite (wqo property). A *downward-closed* (DC) set is defined in a similar manner. In particular, any UC or DC set w.r.t. the sub-word ordering is a regular language.

A *distribution over* S is an array $\delta \in \mathbb{R}_{\geq 0}^{S}$ of values $\delta[s] \in \mathbb{R}_{\geq 0}$ for any $s \in S$, such that $\sum_{s \in S} \delta[s] = 1$. If X is finite and non-empty, we write $\mathcal{U}(X)$ for the *uniform distribution over* X, namely $\forall x \in X, \mathcal{U}(X)[x] = 1/|X|$. We use the notation $\mathrm{Dist}(S)$ to denote the set of distributions over S.

Let S^ω denote the set of infinite sequences over S, called *paths*. A set $X \subseteq S^\omega$ is a *cylinder* if it is of the form $s_0 \cdots s_n \cdot S^\omega$ for some $s_0 \cdots s_n \in S^*$. Such a set is written $\mathrm{Cyl}(s_0 \cdots s_n)$. We introduce $\mathcal{F}(S)$ as the smallest family of sets of paths containing all cylinders, closed under complementation, and countable union. Such sets of $\mathcal{F}(S)$ are called *measurable*.

Given an initial state $s_0 \in S$ and a mapping $\eta : S^+ \to \mathrm{Dist}(S)$ from *histories* to distributions over S, we define a partial function $\mathbb{P} : \mathcal{F}(S) \to \mathbb{R}_{\geq 0}$ such that $\mathbb{P}(\mathrm{Cyl}(s_0)) = 1$, $\forall s \neq s_0, \mathbb{P}(\mathrm{Cyl}(s)) = 0$, and for all $h \cdot s \in S^+ \cdot S$, $\mathbb{P}(\mathrm{Cyl}(h \cdot s)) = \mathbb{P}(\mathrm{Cyl}(h)) \cdot \eta(h)[s]$. Carathéodory's criterion [40] ensures this definition is well and uniquely defined on all $\mathcal{F}(S)$, and $\mathbb{P}$ is therefore called a *probability measure*.

We describe infinite paths with the logic LTL, whose usual semantics over infinite words in S^ω, leads to measurable sets [17, Remark 10.57]. More precisely, we consider LTL with regular valuations [24], where atomic propositions are represented by regular languages. We focus on the fragment without the "until" operator:

▶ **Definition 3** (LTL [39]). *An LTL($\bigcirc, \Diamond$) formula is any φ in the following grammar, where ν ranges over regular languages over S:*

$$\varphi ::= \nu \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \bigcirc \varphi \mid \Diamond \varphi \mid \Box \varphi$$

Here, $\Box\varphi$, $\Diamond\varphi$ and $\Box\Diamond\varphi$ denote *always* φ, *eventually* φ and *infinitely often* φ, respectively.

Let I be a set of indices and V be a set of values. A *profile* $\vec{v}$ is a mapping from I to V, where v_i is the value assigned to i. In particular if $I = \{1 \dots n\}$, then $\vec{v} = (v_1 \dots v_n)$. We introduce a fresh symbol $\perp \notin V$, and define the notation $\vec{v}_{-i}$ as the profile where v_i has been replaced by $\perp$. Given any other value $w \in V$, we write $(\vec{v}_{-i}, w)$ for the profile where the value assigned to i has been replaced by w. We extend these notations to a subset $Y \subseteq I$ in the usual way, i.e., $\vec{v}_{-Y}$ denotes $\vec{v}$ where each $v_i, i \in Y$ is replaced by $\perp$ and $(\vec{v}_{-Y}, \vec{v}'_Y)$ where v_i is replaced by v'_i for each $i \in Y$.

3 Lossy Channel Games

In this section, we provide formal definitions for the game model and the equilibrium concept that we use. While the model definitions are direct generalizations of those in [8], the concurrent setting requires extra care. In this setting, all players choose their actions concurrently and independently. The resulting action profile is evaluated on the game graph, which provides a (distribution of) channel operation to apply and a successor control state. Message losses are then processed.

3.1 Lossy Channel

For simplicity, this article focuses on a single channel system. The *channel configuration* is represented by a word $\mu \in M^*$, where M is a finite alphabet of *messages*. This channel is subject to stochastic *message losses*, meaning that every message has a fixed probability $\lambda \in (0, 1)$ of being lost at every round, independently of other messages.

We can derive the following probability values:

▶ **Example 4** (Message Losses). For any $\mu, \mu' \in M^*$, let us write $P_\lambda(\mu, \mu')$ for the probability of transitioning from channel configuration μ to μ' after random message losses. For example, $P_\lambda(\mu, \mu) = (1 - \lambda)^{|\mu|}$ (no message loss) and whenever $\mu' \not\preceq \mu$ (not a sub-word), we have $P_\lambda(\mu, \mu') = 0$. Moreover, for a single message letter $a \in M$, $P_\lambda(a^n, a^m) = \binom{n}{m} \lambda^n (1 - \lambda)^{n-m}$.

As we will see later in Section 5, the exact value of λ is not relevant for the qualitative probabilistic objectives considered in Definition 10.

3.2 Channel Operations

▶ **Definition 5.** *A* channel operation f *is defined as one of these three type of partial functions:*

- *If* $f = nop$ *then* $f(\mu) = \mu$*;*
- *If* $f = !m$ *then* $f(\mu) = m \cdot \mu$*;*
- *If* $f = ?m$ *and* $\mu = w \cdot m$*, then* $f(\mu) = w$ *and* $f(\mu) = \bot \notin M^*$ *otherwise.*

The set of all such partial functions is denoted op_M*.*

Intuitively, *nop*, $!m$, and $?m$ denote "no action", "enqueue the message m", and "dequeue the message m", respectively. If the channel configuration does not end with the message m, then the effect of $f = ?m$ is the fresh symbol $\bot$, indicating that the operation is not allowed.

3.3 Lossy Channel Arena

Players choose channel operations through an *arena*, which specifies a control state (*location*) that defines the actions allowed by the players and the resulting effects on the channel:

▶ **Definition 6.** *A* n*-player* concurrent stochastic lossy channel arena *(CSLCG arena) is a tuple* $\mathcal{A} = (\mathrm{Agt}, L, M, \{\mathrm{Act}_i\}_{i \in \mathrm{Agt}}, \mathrm{Tab}, l_0)$ *where:*

- $\mathrm{Agt} = \{1 \dots n\}$ *is the set of agents;*
- L *is the set of locations;*
- $l_0 \in L$ *is the initial location;*
- M *is the message alphabet;*
- *For each* $i \in \mathrm{Agt}$*,* Act_i *is a finite set of actions available to agent* i *and require these sets to be pairwise disjoint. We write* Act *for* $\prod_{i \in \mathrm{Agt}} \mathrm{Act}_i$*;*
- $\mathrm{Tab} : L \times \mathrm{Act} \to \mathrm{Dist}(L \times \mathrm{op}_M)$.

A configuration*, or* state *of a CSLCG arena* $\mathcal{A}$ *is a word* $s = l \cdot \mu$ *composed of a location and a channel configuration. The* state space *is denoted by* $S = L \cdot M^*$ *and the* initial state *is* $s_0 = l_0 \cdot \epsilon \in S$*.*

In the rest of the section we assume $\mathcal{A}$ to be fixed.

3.4 Concurrent Actions and Strategies

Since actions are taken concurrently, players must be prevented from taking certain actions that could result in illegal channel operations. In general, the set of allowed actions and strategy decisions will depend on the state (location *and* channel) of the arena, as formalized below:

▶ **Definition 7** (Allowed Actions and Strategies)**.** *For a configuration* $s = l \cdot \mu$ *and a player* $i \in \mathrm{Agt}$*, we define* $\mathrm{Act}_i(s)$ *as the set of* allowed actions α *such that if* $\exists \vec{\beta} \in \mathrm{Act}^n : \mathrm{Tab}(l, (\vec{\beta}_{-i}, \alpha))[l', f] > 0$*, then* $f(\mu) \neq \bot$*. A* strategy for player i *is a mapping* σ_i *from sequences of states (*histories*), to distributions of allowed actions. Namely, for all* $h \cdot s \in S^+$*, we have* $\sigma_i(h \cdot s) \in \mathrm{Dist}(\mathrm{Act}_i(s))$*. We write* $\sigma_i(\alpha \mid h \cdot s)$ *as a shorthand for* $\sigma_i(h \cdot s)[\alpha]$*. The set of strategies of* i *is written* $\mathfrak{S}_i$ *and* $\vec{\mathfrak{S}} = \prod_{i \in \mathrm{Agt}} \mathfrak{S}_i$ *is the set of* strategy profiles*.*

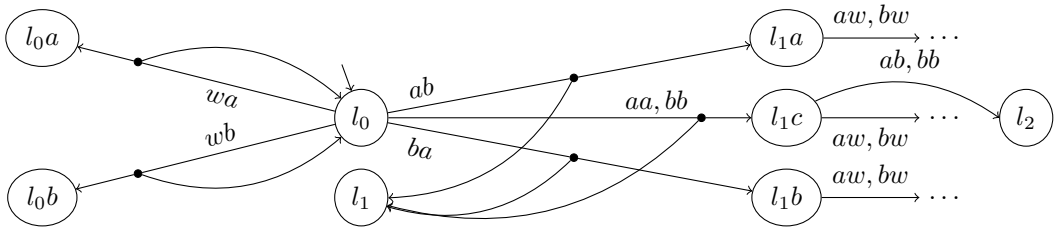


Figure 2 Fragment of the semantics of the CSLCG arena of Figure 1 as a concurrent arena. Stochastic behaviours such as message losses are represented by the • nodes and probability values are omitted.

▶ Remark 8. A strategy profile $\vec{\sigma}$ can be seen as a strategy of a single player taking any (distribution of) action(s) $\vec{\alpha} \in \mathrm{Act}$ from history $h \cdot s \in S^+$ with probability

$$\vec{\sigma}(\vec{\alpha} \mid h \cdot s) = \prod_{i \in \mathrm{Agt}} \sigma_i(\alpha_i \mid h \cdot s)$$

However the converse does not always hold: for example if $\mathrm{Act}_i(s) = \{a, b\}$ for $i \in \{1, 2\}$, there exist no pair of strategies σ_1, σ_2 such that $\vec{\sigma}(aa \mid s) = \vec{\sigma}(bb \mid s) = 1/2$ and $\vec{\sigma}(ab \mid s) = \vec{\sigma}(ba \mid s) = 0$.

3.5 Semantics

The semantics of a CSLCG arena can be understood as an n-player, infinite-state, concurrent stochastic game arena $(S, \mathrm{Agt}, \{\mathrm{Act}_i\}_{i \in \mathrm{Agt}}, s_0, p)$ where $p : S \times \mathrm{Act} \to \mathrm{Dist}(S)$. Intuitively, from any state $s = l \cdot \mu$: (1) Every player picks an allowed action from $\mathrm{Act}_i(s)$ based on the probability given by its strategy on the current history; (2) A new location l' and a channel operation are then sampled according to $\mathrm{Tab}(l, \vec{\alpha})$. Since players only play allowed actions, the new channel configuration $f(\mu)$ is guaranteed to be defined; (3) Message losses finally occur from $f(\mu)$ to some channel configuration μ', resulting in state $s' = l' \cdot \mu'$. This is formally defined below. An example of such a semantics is provided in Figure 2.

▶ **Definition 9** (Semantics of CSLCG). *For an initial state $s_0 \in L \cdot M^*$ and a strategy profile $\vec{\sigma} = (\sigma_i)_{i \in \mathrm{Agt}}$ in the CSLCG arena $\mathcal{A} = (\mathrm{Agt}, L, M, \{\mathrm{Act}_i\}_{i \in \mathrm{Agt}}, \mathrm{Tab}, l_0)$, we define $\mathbb{P}^{\vec{\sigma}}_{s_0}$ as the probability measure on $\mathcal{F}(S)$ uniquely defined by its values on cylinders:*

- $\mathbb{P}^{\vec{\sigma}}_{s_0}(\mathrm{Cyl}(s_0)) = 1$;
- $\forall s \neq s_0, \mathbb{P}^{\vec{\sigma}}_{s_0}(\mathrm{Cyl}(s)) = 0$;
- *For every pair of states $s = l \cdot \mu \in S$ and $s' = l' \cdot \mu' \in S$, and any history $h \in S^*$,*

$$\mathbb{P}^{\vec{\sigma}}_{s_0}(\mathrm{Cyl}(h \cdot s \cdot s')) = \mathbb{P}^{\vec{\sigma}}_{s_0}(\mathrm{Cyl}(h \cdot s)) \sum_{\vec{\alpha}, f} \underbrace{\vec{\sigma}(\vec{\alpha} \mid h \cdot s)}_{(1)} \times \underbrace{\mathrm{Tab}(l, \vec{\alpha})[(l', f)]}_{(2)} \times \underbrace{P_\lambda(f(\mu), \mu')}_{(3)}.$$

When $s_0 = l_0 \cdot \epsilon$, we simply write $\mathbb{P}^{\vec{\sigma}} = \mathbb{P}^{\vec{\sigma}}_{s_0}$.

We now define qualitative objectives, that will be used to express players' goals and the (global) properties to be checked:

▶ **Definition 10.** *Let φ be a measurable set of paths.For any strategy profile $\vec{\sigma}$ and state s, we define the* property $\mathrm{NZ}(\varphi)$ *and* $\mathrm{AS}(\varphi)$ *by:*

$$\mathcal{A}, \vec{\sigma}, s \vDash \mathrm{NZ}(\varphi) \text{ for } \mathbb{P}^{\vec{\sigma}}_s(\varphi) > 0 \quad \text{and} \quad \mathcal{A}, \vec{\sigma}, s \vDash \mathrm{AS}(\varphi) \text{ for } \mathbb{P}^{\vec{\sigma}}_s(\varphi) = 1.$$

We omit $\mathcal{A}$ or s when they are clear from context.

We extend the definition to any conjunction of objectives: For a conjunction of NZ *and* AS *objectives $\Psi_1 \wedge \ldots \wedge \Psi_k$, we have*

$\mathcal{A}, \vec{\sigma}, s \vDash \Psi_1 \wedge \ldots \wedge \Psi_k$ *if and only if for all i* $\mathcal{A}, \vec{\sigma}, s \vDash \Psi_i$.

We consider for φ reachability or safety conditions of regular sets of states, namely LTL *with regular valuations* [24], and identify a formula φ with its semantics $\mathcal{L}(\varphi) \subseteq (L \cdot M^*)^\omega$. For example, to specify the objective "reaching l_2 almost-surely, while not having more than 3 queued messages with positive probability", one would write:

$\Gamma = \mathrm{AS}(\Diamond l_2 \cdot M^*) \wedge \mathrm{NZ}(\Box L \cdot M^{\leq 3})$

▶ **Definition 11.** *A CSLCG is defined as an n-player arena together with such properties, called* objectives*, for every players: $\mathcal{G} = (\mathcal{A}, \Phi_1 \ldots \Phi_n)$. In particular, when $n = 2$, and $\Phi_1 = \neg\Phi_2$, we say that $\mathcal{G}$ is a* 2.5 player zero-sum game*. A strategy $\sigma_i \in \mathfrak{S}_i$ such that $\forall \vec{\sigma}', (\vec{\sigma}'_{-i}, \sigma_i), s \vDash \Phi$ is called a* winning strategy of Φ for i.

▶ **Definition 12.** *For a game $\mathcal{G}$, strategy profile $\vec{\sigma}$, and state s, we define the set of winners and losers by $\mathrm{W}_\mathcal{G}(\vec{\sigma}, s) = \{i \in \mathrm{Agt} : (\vec{\sigma}, s) \models \Phi_i\}$ and $\mathrm{L}_\mathcal{G}(\vec{\sigma}, s) = \mathrm{Agt} \setminus \mathrm{W}_\mathcal{G}(\vec{\sigma}, s)$. When s is the initial state we simply write $\mathrm{W}_\mathcal{G}(\vec{\sigma})$ and $\mathrm{L}_\mathcal{G}(\vec{\sigma})$.*

4 Verifying the Core

We consider a cooperative equilibrium concept called the core [14, 41, 30]. Analogous to a Nash equilibrium (NE) [38], a member of the core can be characterized by (the absence of) *beneficial deviations*. However, unlike a Nash equilibrium where only one player can deviate, with the core, a *group* or *coalition* of players can deviate together. The notion of beneficial coalitional deviation is formally defined as follows.

▶ **Definition 13.** *For a strategy profile $\vec{\sigma}$, we say that a joint strategy $\vec{\sigma}'_C, C \subseteq \mathrm{Agt}, C \neq \varnothing$, is a beneficial coalitional deviation from $\vec{\sigma}$ if $C \subseteq \mathrm{L}_\mathcal{G}(\vec{\sigma})$ and for all $\vec{\sigma}'_{-C}$, we have $C \subseteq \mathrm{W}_\mathcal{G}((\vec{\sigma}'_C, \vec{\sigma}'_{-C}))$.*

The core of a game $\mathcal{G}$ is defined to be *the set of strategy profiles that admit no beneficial coalitional deviation*[3]. We write $\mathrm{Core}(\mathcal{G})$ to denote the set of strategy profiles in the core of $\mathcal{G}$. We focus on two decision problems related to the core: E-Core and A-Core. These problems are formally defined below.

Given: $(\mathcal{G}, \Gamma)$ with game $\mathcal{G}$ and property Γ.
E-Core: Does there exists some $\vec{\sigma} \in \mathrm{Core}(\mathcal{G})$ such that $\mathcal{G}, \vec{\sigma} \models \Gamma$?
A-Core: Is it the case that for all $\vec{\sigma} \in \mathrm{Core}(\mathcal{G})$ we have $\mathcal{G}, \vec{\sigma} \models \Gamma$?

Note that, due to the duality of these problems, it is enough to provide a procedure for E-Core.

▶ **Example 14.** Consider a transmission system with one channel similar to the one discussed in Example 1. Now, suppose there are three players, S (sender), R (receiver), and A (attacker). S/R decides to send/request a message, a or b. A message is delivered successfully if R reads

[3] An alert reader may notice the similarities between the core and *strong NE* [13] and *coalition-proof NE* [18]. The main difference is that these non-cooperative equilibrium concepts do not assume the existence of binding agreements. We refer to [29] for a more detailed discussion on this matter.

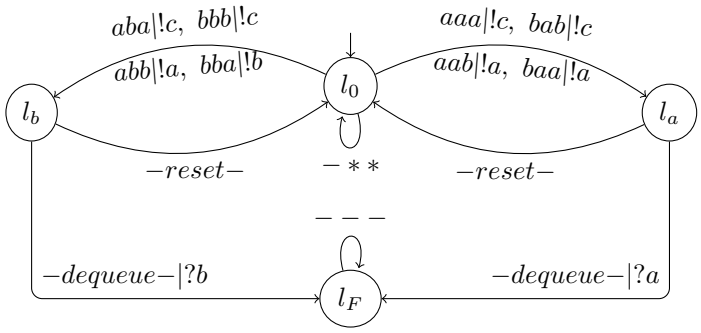


Figure 3 A graphical representation of a 3-player arena. Joint actions are ordered by SRA, for example, xyz indicates that players S, R, A choose actions x, y, z respectively.

(dequeues) the same type of message as she requested. An attack is successful if A chooses the same type of message being sent, and this turns the message into a c. We model this game as a CSLCG where $\mathrm{Act}_S = \mathrm{Act}_A = \{a, b, -\}$, $\mathrm{Act}_R = \{a, b, dequeue, reset\}$. The arena is depicted in Figure 3. The goals of S and R are given as $\Phi_S = \mathrm{AS}(\Box(L \cdot M^{\leq k}))$ for some $k \in \mathbb{N}$ (almost surely the channel never exceeds size k), and $\Phi_R = \mathrm{AS}(\Diamond(l_F \cdot M^*))$ (almost surely a correct message is delivered). The goal of A is $\Phi_A = \neg(\Phi_S \wedge \Phi_R)$.

Consider the following strategy profile: if the channel contains fewer than k messages, S and R play aa and bb uniformly at random. Otherwise, S plays action $-$. This profile satisfies both Φ_S and Φ_R, and is therefore in the core. In fact, *all* strategy profiles in the core satisfy both Φ_S and Φ_R. As a result, if we query E-Core with $\Gamma = \Phi_A = \neg(\Phi_S \wedge \Phi_R)$, it returns a negative answer. This is in contrast to NE: since with NE S and R do not act as a coalition, there exists a (undesirable) NE in which there is a non-zero probability that the channel will exceed size k or a correct message will never be delivered (i.e., Γ is satisfied). ⌟

To address E-Core, we first introduce the notion of *concurrent two-player coalition game* (TPCG) as follows.

▶ **Definition 15.** *Let $\mathcal{G} = (\mathcal{A}, (\Phi_i)_{i \in \mathrm{Agt}})$ be a CSLCG and $C \subseteq \mathrm{Agt}$, with the underlying arena $\mathcal{A} = (\mathrm{Agt}, L, M, \{\mathrm{Act}_i\}_{i \in \mathrm{Agt}}, \mathrm{Tab}, l_0)$. The concurrent two-player coalition game arena is defined as $\mathcal{A}^C = ((1, 2), L, M, \{\mathrm{Act}_i\}_{i \in \mathrm{Agt}}, \mathrm{Tab}', l_0)$ where for all actions $\vec{\alpha}, \vec{\beta} \in \mathrm{Act}$, the transition is determined by the projections on C and $-C = \mathrm{Agt} \setminus C$, respectively for player 1 and 2: $\mathrm{Tab}'(l, (\vec{\alpha}, \vec{\beta})) = \mathrm{Tab}(l, (\vec{\alpha}_C, \vec{\beta}_{-C}))$. The TPCG with respect to $\mathcal{G}, C$, and objective Ψ_C is thus defined as $\mathcal{G}^{C, \Psi_C} = (\mathcal{A}^C, (\Psi_C, \neg\Psi_C))$.*

Observe that the ability of a coalition C to satisfy an objective Ψ_C depends on whether it has a winning strategy in TPCG $\mathcal{G}^{C, \Psi_C}$ from the initial state of the game (thus, the existence of beneficial deviation). With this observation, we restate the characterization of E-Core from [26] as follows.

▶ **Proposition 16** ([26]). *A pair $(\mathcal{G}, \Gamma)$ is a yes-instance of E-Core if and only if there exists $W \subseteq \mathrm{Agt}$ such that*
(a) *there exists some $\vec{\sigma}$ such that $\mathcal{G}, \vec{\sigma} \models \Phi_W$, and*
(b) *for all $C \subseteq \mathrm{Agt} \setminus W$, C has no winning strategy in $\mathcal{G}^{C, \Psi_C}$,*
where $\Phi_W = \bigwedge_{i \in W} \Phi_i \wedge \bigwedge_{i \notin W} \neg\Phi_i \wedge \Gamma$ and $\Psi_C = \bigwedge_{i \in C} \Phi_i$.

Using Proposition 16, we provide the following procedure for determining whether some $(\mathcal{G}, \Gamma)$ is a yes-instance of E-Core.

1. Guess a set of winning players $W \subseteq \mathrm{Agt}$;
2. Check if there is a winning strategy $\vec{\sigma}$ in TPCG $\mathcal{G}^{\mathrm{Agt},\Phi_W} = (\mathcal{A}, (\Phi_W, \neg\Phi_W))$;
3. Check if there is a coalition $C \subseteq \mathrm{Agt} \setminus W$ with a winning strategy $\vec{\sigma}'_C$ in TPCG $\mathcal{G}^{C,\Psi_C} = (\mathcal{A}^C, (\Psi_C, \neg\Psi_C))$;
4. If the answers to Step 2 is "Yes" and Step 3 is "No", then $(\mathcal{G}, \Gamma)$ is a yes-instance of E-Core. Otherwise, it is not.

Observe that above procedure corresponds to Proposition 16. In particular, Steps 2 and 3 correspond precisely to (a) and (b) in Proposition 16, respectively. Thanks to this procedure, the problem of checking whether $(\mathcal{G}, \Gamma)$ is a yes-instance of E-Core can be reduced to solving a collection of concurrent 2.5-player zero-sum games.

Note that if we consider almost-sure LTL objectives, we can construct a CSLCG $\mathcal{G}$ with one player whose goal is $\Phi = \mathrm{AS}((\Box\Diamond R_1) \wedge (\Diamond\Box R_2))$. Then, we can set $\Gamma = \Phi$ and query E-Core. This reduces to a 1.5-player game over a lossy channel with objective Φ, which is already undecidable [16, Lemma 5.12]. Therefore, we obtain the following result.

▶ **Proposition 17.** *For a pair $(\mathcal{G}, \Gamma)$ where players' objectives and Γ are given by almost-sure LTL formulae, the problems of E-Core and A-Core are undecidable.*

In the following sections, we study the technical machineries required to solve concurrent 2.5-player zero-sum games with almost-sure safety and almost-sure reachability objectives. As motivated by Proposition 16 and its corresponding procedure for E-Core, we further solve concurrent 2.5-player zero-sum games with a conjunction of objectives. These provide the necessary foundation for the main decidability result presented later in Section 7.

5 The Zero-Sum case is Effective

In this section, we focus on solving concurrent 2.5-player zero-sum CSLCG as a crucial step towards our main decidability result for E-Core. We assume that the CSLCG $\mathcal{G}$ with arena $\mathcal{A}$ is fixed, and R represents a set of states, which may be infinite. While [23] provides an approach for solving concurrent stochastic games with $\mathrm{AS}(\Diamond R)$, $\mathrm{AS}(\Box R)$, $\mathrm{NZ}(\Diamond R)$ and $\mathrm{NZ}(\Box R)$ objectives in the *finite-state* setting, here we extend their approach to the *infinite-state* setting. We note that the infinite-state setting has been previously examined for *turn-based* games in [19, 5, 15].

Our approach can be summarized as follows: First, we provide algorithms to symbolically compute one-step reachability for regular set of states. Then, we prove that the algorithms from [23] remain valid (in terms of termination and correctness) for this class. More precisely, we fix a regular set $R \subseteq (L \cdot M^*)^*$ and a 2.5-player zero-sum CSLCG $\mathcal{G}$ with $\mathrm{Agt} = \{1, 2\}$. For $i \in \mathrm{Agt}$, the objective Φ_i is of the form $\mathrm{NZ}(\Diamond R)$ or $\mathrm{AS}(\Diamond R)$. Furthermore, by instantiating games for the opponent $-i$, setting $R' = S \setminus R$, and by determinacy of such games [35], we also solve the game for objectives of the form $\mathrm{AS}(\Box R')$ or $\mathrm{NZ}(\Box R')$. To this end, we focus on the existence of winning strategies (Definition 11) and the computation of *winning regions*.

▶ **Definition 18.** *For an objective Φ of player i, the* winning region of Φ for player i *in $\mathcal{G}$ is given by:* $[\![\Phi]\!]_i(\mathcal{G}) = \{s \in S \mid \exists \sigma_i \in \mathfrak{S}_i, \forall \vec{\sigma}' \in \vec{\mathfrak{S}}, (\vec{\sigma}'_{-i}, \sigma_i), s \vDash \Phi\}$.

As previously mentioned, moving from a finite to an infinite state setting presents a challenge in the representation of winning strategies. To address this, in what follows, we provide a novel encoding of different classes of winning strategies necessary for ensuring a win.

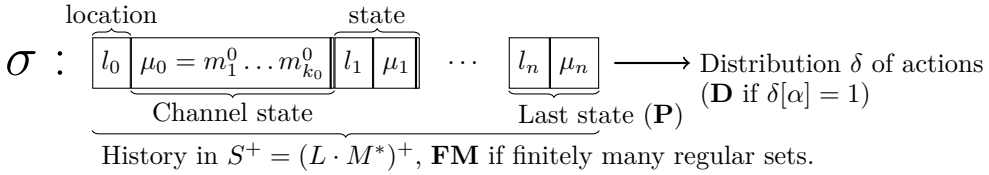


Figure 4 Summary of strategy classes.

5.1 Regularity Properties of CSLCG

We begin by revisiting the regularity results of lossy channel systems from [19, 5] to later design algorithms for computing winning regions. In this context, the *predecessor function* and the *finite attractor* [15] are essential notions that need to be adapted to the concurrent case.

▶ **Definition 19** (Predecessor Function). *For a given player $i \in \mathrm{Agt}$ and a set of states $B \subseteq S$, we write $\mathrm{Pre}_i(B)$ the set of states from which player i can enforce reaching B with **positive probability**, no matter the actions of the other players:*

$$\mathrm{Pre}_i(B) = \{s \in S \mid \exists \sigma_i \forall \vec{\sigma}' \mathbb{P}_s^{(\vec{\sigma}'_{-i}, \sigma_i)}(\bigcirc B) > 0\}$$

We summarize the properties below, which are all consequences of the lossy nature of the system.

▶ **Proposition 20.** *The order $\sqsubseteq$ defined for all $s = (l \cdot \mu), s' = (l' \cdot \mu')$ by $s \sqsubseteq s'$ if $s \preceq s'$ (sub-word) and μ, μ' are equal or have the same last letter, is a wqo. Further, given $X \subseteq S$,*

1. *$\mathrm{Pre}_i(X)$ is upward-closed for $\sqsubseteq$, hence regular;*
2. *If X is regular, then $\mathrm{Pre}_i(X)$ can be computed;*
3. *(*Finite Attractor *[15]) There exists a finite set $A \subseteq S$, such that $\forall \vec{\sigma} \in \vec{\mathfrak{S}}\ \mathbb{P}^{\vec{\sigma}}(\Box\Diamond A) = 1$.*

Proof sketch.

1. If $s \sqsubseteq t$, and $s \in \mathrm{Pre}_i(X)$, then an action from s leads to X with positive probability. Since $s \sqsubseteq t$ entails the same last letter in s and t's channel, $\mathrm{Act}_i(s) = \mathrm{Act}_i(t)$ and the same action can be played by i from t as with positive probability the extra messages in t and not in s can then be dropped.
2. We refer to the computability section of [7], and argue that the concurrent setting can be simulated by a turn-based game where the first player i commits to an action α, then moves to a new state where the rest of the players provide their action, and move to a stochastic state where messages are actually dropped. Pre_i can therefore be simulated by three calls to the predecessor function of [7].
3. We refer to Corollary 5.3 in [19] or Lemma 5.3 in [5]. Intuitively, having more messages on the channel increases the likelihood of a decrease in message count, hence the set $L \cdot \epsilon$ (any location, empty channel) is shown to a finite attractor. ◀

We provide the following regular encoding of strategies, which allows for a more precise description of classes of strategies and enables their finite representation:

▶ **Definition 21.** *Let $\sigma_i \in \mathfrak{S}_i$ be a strategy for player i. σ_i is* positional *(P) if it depends only on the last state (location and channel): $\forall h, h' \in S^*, \forall s \in S, \sigma_i(hs) = \sigma_i(h's)$.*

σ_i is finite memory *(FM) if*

- *The set $\{\sigma_i(h) \mid h \in S^+\} \subseteq \mathrm{Dist}(\mathrm{Act}_i)$ is finite;*
- *For every $\delta \in \mathrm{Dist}(\mathrm{Act}_i)$, the set of histories $\sigma^{-1}(\delta) \subseteq (L \cdot M^*)^*$ is regular.*

Figure 4 illustrates how a FM strategy can be represented finitely. It determines which action to play by running a finite number of automata on the history, reading every location and channel configuration. In contrast, a strategy that depends only on the last state may not be representable in the infinite state case, as infinitely many different decisions might be taken. Therefore, we avoid using the usually synonymous term *memoryless* and instead refer to these strategies as *positional*. A convenient representation is therefore the positional *and* finite memory (PFM) strategies, meaning that the action depends only on the last state, which must belong to one of finitely many regular sets.

5.2 Positive Reachability

In the positive reachability case, player i tries to enforce that *some* finite prefix reaches the target set R. This can be achieved by a backward-reachability algorithm instantiated on well-quasi-ordered sets [6]:

▶ **Lemma 22.** *Given a regular set $R \subseteq S^*$, the algorithm that computes $\bigcup_{k\geq 0} \mathrm{Pre}_i^k(R)$ converges in a finite number of steps and returns a set $R \cup V$, where V is upward-closed and $R \cup V = [\![\mathrm{NZ}(\Diamond R)]\!]_i$. Moreover, PFM winning strategies are sufficient for both players.*

▶ **Example 23.** Consider the example from Figure 1. A winning strategy for player 2/Attacker to achieve the objective $\mathrm{NZ}(\Diamond l_2 \cdot c)$ (eventually reaching state $l_2 \cdot c$ with positive probability) consists in playing $\mathcal{U}(\{a, b\})$ from any state in $l_0 \cdot M^*$, then b from any state in $l_1 \cdot M^* \cdot c^2$. With positive probability, the state $l_1 \cdot cc$ is reached after 3 steps, regardless of Sender's strategy, and then $l_2 \cdot c$ is reached.

This algorithm can also be used to compute almost-sure safety of R by applying the determinacy result: $[\![\mathrm{AS}(\Box R)]\!]_i = \overline{[\![\mathrm{NZ}(\Diamond \overline{R})]\!]_{-i}}$. As shown in Lemma 22, the optimal strategy is PFM, and in the safety case, it can be further restricted without compromising the safety property. More precisely, we define the most general action restriction for player i that preserves safety as follows:

▶ **Definition 24.** *For any set $R \subseteq S$, $\mathrm{Stay}_i^{\mathcal{G}}(R)$ is the mapping that restricts the allowed actions on R to stay in R:*

$$s \in R \mapsto \{\alpha \in \mathrm{Act}_i(s) \mid \forall \vec{\sigma}, \mathbb{P}^{(\vec{\sigma}_{-i}, \alpha)}(\bigcirc R) = 1\}$$

5.3 Almost-Sure Reachability

In contrast to the turn-based case in [8], concurrent actions require a careful analysis of the allowed actions. The intuition is as follows: As the player with reachability objective tries to avoid "being trapped" in bad states, he may choose not to play some actions based on the previously defined Stay operator (Definition 24). This limits the available actions and gives more power to his opponent, which then reduces the winning region. This idea was proposed in the algorithm for almost-sure reachability in finite state games described in [23]. We follow here this approach by implementing the algorithm symbolically as depicted in Algorithm 1 which boils down to providing effective procedures for Pre_i and Stay_i, when the input is a regular set of states. As the algorithm in [23] actually solves almost-sure *repeated* reachability, the authors assumed that the target set R is *absorbing*, meaning that once a state in R is reached, the game cannot exit R, regardless of the players' actions. This assumption is also made here as adjusting the Pre_i operator is sufficient to guarantee this property in our setting. Termination of the algorithm can be shown using the wqo property, similar to that described in [8], resulting in the following lemma:

Algorithm 1 Almost-Sure Reachability.

Input: An arena $\mathcal{A}$, $i \in \mathrm{Agt}$ and a regular set R
Output: $D_k = [\![\mathrm{AS}(\Box\Diamond R)]\!]_i$

$k \leftarrow 0$; $D_0 \leftarrow S$; $\mathcal{A}_0 \leftarrow \mathcal{A}$
repeat
$C_k = [\![\mathrm{AS}(\Box(D_k \backslash R))]\!]_{-i}(\mathcal{A}_k)$
$Y_k = [\![\mathrm{NZ}(\Diamond R)]\!]_i(\mathcal{A}_k)$
$D_{k+1} = [\![\mathrm{AS}(\Box(Y_k))]\!]_i(\mathcal{A}_k)$
$\mathcal{A}_{k+1} \leftarrow \mathcal{A}_k$ where $\mathrm{Act}_i^{\mathcal{A}_{k+1}} = \mathrm{Stay}_i^{\mathcal{A}_k}(D_{k+1})$
until $D_k = D_{k+1}$

▶ **Lemma 25.** *The almost-sure reachability algorithm of [23], instantiated symbolically on 2.5-player CSLCG in Algorithm 1, terminates and returns a downward-closed set.*

To prove the correctness of Algorithm 1, we must provide a winning strategy from every state in the computed set D_k, and a strategy for the opponent from every state in the *complement* set $\overline{D_k}$. At this point, it is important to note that positional strategies may not be sufficient, especially for the opponent player. As illustrated by the *Hide-or-Run game* presented in [23], non-positional may be needed. More precisely, the authors noticed that Markov strategies –where decisions depend on the current state and on the clock value only– are sufficient for winning with positive safety objectives. They further introduced the sufficient subclass of so-called *counting strategies*, where a sequence of probability values is fixed, so that the strategy can be finitely represented. We generalise this notion of counting strategies to the infinite state case as follows:

▶ **Definition 26.** *For any k, let $p_k = 2^{-1/(2^k)}$. A strategy $\sigma_i \in \mathfrak{S}_i$ is* counting *(C) if there exist two PFM strategies $\sigma_i^v, \sigma_i^u \in \mathfrak{S}_i$ such that for every $k \in \mathbb{N}$, $h \cdot s \in S^k$ and any $\alpha \in \mathrm{Act}_i(s)$,*

$$\sigma_i(\alpha \mid h \cdot s) = p_k \sigma_i^u(\alpha \mid h \cdot s) + (1 - p_k)\sigma_i^v(\alpha \mid h \cdot s)$$

Note that p_k is fixed a sequence of reals between 0 and 1, such that the infinite product $\prod_{i=1}^{\infty} p_i = p$ is between 0 and 1. This means that the strategy σ_i^u always has some positive probability of being played, but overall cannot be played forever. Since the sequence $(p_k)_k$ is fixed, a counting strategy requires infinite memory but can be finitely represented by two PFM strategies.

▶ **Example 27.** Consider again the example shown in Figure 1. We provide two examples of winning strategies in the AS case:

- If $\Phi_1 = \mathrm{AS}(\Diamond(l_1 \cdot \{a\}^3 \cup l_2 \cdot M^*))$ –namely Sender can almost-surely eventually force a valid transmission of 3 consecutive a's, assuming that the game continues forever– Sender has a winning strategy by playing $\mathcal{U}(\{a, b\})$ from all states. For any strategy σ_2 of Attacker, either the game eventually reaches l_2, or there is a state $s = l \cdot \mu \in \{l_0, l_1\} \cdot M^*$ visited infinitely often. From this state, there is a fixed probability $p > 0$ to produce three consecutive a's and that all b's are dropped, so this event eventually happens almost-surely.
- If $\Phi_1 = \mathrm{AS}(\Diamond L \cdot M^*\{a\}M^*)$ –namely a message a is eventually sent almost-surely– we argue that Sender cannot achieve her objective. To observe this, one can exhibit a winning strategy for Attacker, whose objective is then $\Phi_2 = \mathrm{NZ}(\Box L \cdot \{b, c\}^*)$. Such a strategy consists of the counting strategy playing action w with probability p_k at round k and

$\mathcal{U}(\{a,b\})$ otherwise. At any round still in l_0, Sender cannot risk playing a since there is a small probability for Attacker to scramble the communication and then reach l_2. The overall probability to stay in $l_0 \cdot b^*$ is therefore $\prod_k p_k > 0$.
On the other hand, any PFM strategy by Attacker can be defeated, which proves that counting strategies are required. Indeed by playing b from all states in $l_0 \cdot b^*$, Sender ensures that she will never lose since either σ_A eventually plays w with probability 1, and Sender can then play a and win, or there is a fixed positive probability (FM) that the game moves to $l_1 \cdot b^*$, which happens almost-surely.

This allows us to conclude on the almost-sure reachability case by adapting the correctness proof of the almost-sure reachability algorithm in [23] mentioned in Lemma 25. As a matter of fact, the finite attractor property seen in Proposition 20 allows us to refer back to the finite state case and derive sufficient strategies for both players. We conclude this subsection with the following:

▶ **Lemma 28.** *Using Algorithm 1, one can compute:*
- *The winning set* $W = [\![\mathrm{AS}(\Diamond R)]\!]_i$*;*
- *A PFM winning strategy for* i*,* $\sigma_i : h \cdot s \in S^* \cdot W \mapsto \mathcal{U}(\mathrm{Stay}_i(W))$*;*
- *A counting (C) winning strategy for his opponent* $-i$ *(objective* $\mathrm{NZ}(\Box\overline{R})$*).*

6 Conjunction of Objectives

In this section, we stay in the 2.5-player zero-sum setting, and address the computation of winning regions for objectives composed as a *conjunction* of qualitative objectives. Note that contrary to the previous section, games are in general not determined for such conjunctive objectives [43], so we focus now on the winning strategies for the player whose objective is a conjunction of objectives.

As discussed in Section 5, positive reachability and almost-sure safety/reachability can be achieved using PFM strategies, while positive safety may require the use of counting strategies. The rest of this section is dedicated to proving the following theorem by combining PFM and counting strategy classes.

▶ **Theorem 29.** *Let* Φ *be a conjunction of* NZ *and* AS *objectives for safety and reachability path specifications. Then the winning region* $[\![\Phi]\!]_i(\mathcal{G})$ *is computable.*

We first notice that conjunction of objectives usually enjoys some convexity properties, namely randomization between individual optimal strategies is sometimes sufficient to achieve the conjunctive objective. This is the case when the conjunction Φ does not contain a positive safety objective. Following this observation, we get:

▶ **Lemma 30.** *If* Φ *is a conjunctive objective containing only* $\mathrm{NZ}(\Diamond\cdot), \mathrm{AS}(\Box\cdot)$ *and* $\mathrm{AS}(\Diamond\cdot)$ *objectives of regular sets, then* $[\![\Phi]\!]_i(\mathcal{G})$ *is computable, and there exists a winning FM strategy for player* i*.*

Sketch.
- If all reachability objective sets are absorbing –that is to say cannot be left once satisfied– we prove that a PFM winning strategy exists for player i.
  For every objective $\Psi = \mathrm{AS}(\Box R)$ or $\Psi = \mathrm{AS}(\Diamond R)$ appearing in Φ, we restrict the set of available actions to $\mathrm{Stay}_i([\![\Psi]\!]_i)$, as a winning strategy necessarily plays actions from this set only. States where no actions are available are removed. Finally, we solve the game on the restricted arena for each $\mathrm{NZ}(\Diamond\cdot)$ objective, and take the intersection of winning regions. A sufficient winning strategy consists in playing all actions uniformly at random.

- In the general case, we build a new game $\mathcal{G}'$ keeping track of the already satisfied reachability objectives. Each reachability objective can therefore be rewritten to be absorbing by referring to the extra information bit stored in every state. A PFM winning strategy in $\mathcal{G}'$ is then translated back to a FM strategy in $\mathcal{G}$, at the expense of a 1-bit memory per objective. ◀

The treatment of positive safety objectives requires more attention, as winning strategies may require infinite memory, as seen in Lemma 28. We notice however that any positive safety objective can be replaced by a simple positive reachability property as it is sufficient for a winning strategy to wait for all almost-sure reachability objectives to be met, then play the counting strategy for $\mathrm{NZ}(\Box\cdot)$. It is therefore sufficient and necessary, for at least one state in $[\![\mathrm{NZ}(\Box\cdot)]\!]_i$ to be reachable with positive probability. This is summarized by the final lemma below:

▶ **Lemma 31** (From $\mathrm{NZ}(\Box)$ to $\mathrm{NZ}(\Diamond)$)**.** *If* $\Phi \equiv \Theta \wedge \mathrm{NZ}(\Box\overline{R})$*, then* $[\![\Phi]\!]_i = [\![\Theta \wedge \mathrm{NZ}(\Diamond R')]\!]_i$ *where* Θ *is some conjunctive condition (as defined in Definition 10) and*

$$R' = [\![\mathrm{NZ}(\Box\overline{R})]\!]_i \cap \bigcap_{\mathrm{AS}(\Diamond R'')\in\Phi} R''.$$

Theorem 29 is then obtained by applying Lemma 31 for every $\mathrm{NZ}(\Box\cdot)$ objective, then applying Lemma 30.

7 Decidability of E-Core and A-Core

In this section, we present our main decidability result. We begin by recalling that E-Core and A-Core are undecidable when players' goals are given by almost-sure LTL formulae (Proposition 17). To obtain decidability, we focus on two types of objectives: almost-sure reachability and almost-sure safety objectives. Consider a CSLCG $\mathcal{G} = (\mathcal{A}, (\Phi_i)_{i\in\mathrm{Agt}})$ in which for each $\Phi_i = \mathrm{AS}(\varphi_i)$, either $\varphi_i \equiv \Diamond R_i$ or $\varphi_i \equiv \Box R_i$. Then the formula used in Step 2 of the procedure for solving E-Core in Section 4 is given as:

$$\Phi_W = \bigwedge_{i\in W} \mathrm{AS}(\varphi_i) \wedge \bigwedge_{i\notin W} \mathrm{NZ}(\neg\varphi_i) \wedge \Gamma.$$

The formula used in Step 3 is given as $\Psi_C = \bigwedge_{i\in C} \mathrm{AS}(\varphi_i)$.

Observe that the TPCG $\mathcal{G}^{\mathrm{Agt},\Phi_W}$ in Step 2 is, in fact, a 1.5-player game, since player 1 consists of all players in the original game. Thus, solving Step 2 amounts to finding a scheduler that satisfies Φ_W. This problem is decidable if $\Gamma = \mathrm{AS}(\varphi)$ and φ is of the form $\bigwedge_i \Diamond R_i$, $\bigwedge_i \Box R_i$, or $\bigwedge_i \Box\Diamond R_i$ [16]. Furthermore, if such a scheduler exists, then there is one that is deterministic and has finite memory. To solve Step 3, we reduce it to solving, for each $C \subseteq \mathrm{Agt}$, a 2.5-player zero-sum game in which the goal of player 1 is Ψ_C. As shown in Theorem 29, this problem is decidable. Therefore, we obtain the following theorem.

▶ **Theorem 32.** *For a pair* $(\mathcal{G}, \Gamma)$ *where players' objectives are almost-sure reachability or almost-sure safety objectives, and property* $\Gamma = \mathrm{AS}(\varphi)$ *with* φ *of the form* $\bigwedge_i \Diamond R_i$*,* $\bigwedge_i \Box R_i$*, or* $\bigwedge_i \Box\Diamond R_i$*, the problems of E-Core and A-Core are decidable.*

8 Concluding Remarks

This paper presents the first result on rational verification of concurrent stochastic games on infinite arenas, specifically for lossy channel games. Our focus is on the decidability of verifying the core of multi-player concurrent stochastic lossy channel games. To this end, we

provide an approach for solving concurrent 2.5-player zero-sum games with a conjunction of almost-sure safety and almost-sure reachability objectives, and infinite state arenas. Our approach extends previous methods, which were limited to either finite state arenas or turn-based games, to work on concurrent settings and infinite state arenas.

It is worth mentioning that most of the algorithms described in this work have non-elementary complexity in the worst case, as they address problems that can be reduced to the reachability problem in lossy channel systems [42]. One might wonder if it is possible to trade stochasticity for non-determinism in the model to obtain an easier problem. Surprisingly, although the winning region for non-deterministic reachability (i.e., equivalent to $\mathrm{NZ}(\Diamond\cdot)$) is computable, the winning region for non-deterministic safety ("does there exists an infinite path ...") is non-computable, due to undecidability results by [36].

Finally, an obvious direction for future work is to consider the NE concept and its related variants, such as strong NE and coalition-proof NE. However, addressing questions related to NE in concurrent games with probabilistic qualitative objectives is challenging, even in the finite state case. There are two main challenges: Firstly, because NE are strategy profiles where players do not behave as a coalition, limiting their ability to prevent deviations (see Remark 8), finding the appropriate joint strategies is not straightforward. Secondly, encoding stability against any deviation in the NE setting produces more complex conjunctions of objectives, requiring significant extensions of the techniques presented in Section 6. Reductions from concurrent to turn-based two-players games, have been used in previous work. For instance, in *suspect games* [20] one player proposes a NE while the second attempts to disprove it. Another approach is by sequentializing games in order to compute *punishing strategies/regions* to characterize NE [28, 31]. However, these approaches encounter similar challenges above when stochastic behaviours are introduced. We conjecture that in order to capture the NE concept in its generality, the reduction to 2.5-player games should feature concurrent actions to account for randomized actions and even counting strategies in the NE.

References

1 Alessandro Abate, Julian Gutierrez, Lewis Hammond, Paul Harrenstein, Marta Kwiatkowska, Muhammad Najib, Giuseppe Perelli, Thomas Steeples, and Michael Wooldridge. Rational verification: game-theoretic verification of multi-agent systems. *Applied Intelligence*, 51(9):6569–6584, 2021.

2 Parosh Aziz Abdulla, C. Aiswarya, and Mohamed Faouzi Atig. Data communicating processes with unreliable channels. In Martin Grohe, Eric Koskinen, and Natarajan Shankar, editors, *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 166–175. ACM, 2016. doi:10.1145/2933575.2934535.

3 Parosh Aziz Abdulla, Mohamed Faouzi Atig, Raj Aryan Agarwal, Adwait Godbole, and Shankara Narayanan Krishna. Probabilistic total store ordering. In Ilya Sergey, editor, *Programming Languages and Systems – 31st European Symposium on Programming, ESOP 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings*, volume 13240 of *Lecture Notes in Computer Science*, pages 317–345. Springer, 2022. doi:10.1007/978-3-030-99336-8_12.

4 Parosh Aziz Abdulla, Mohamed Faouzi Atig, Ahmed Bouajjani, and Tuan Phong Ngo. A load-buffer semantics for total store ordering. *Logical Methods in Computer Science*, 14(1), 2018. doi:10.23638/LMCS-14(1:9)2018.

5 Parosh Aziz Abdulla, Nathalie Bertrand, Alexander Moshe Rabinovich, and Philippe Schnoebelen. Verification of probabilistic systems with faulty communication. *Inf. Comput.*, 202(2):141–165, 2005. doi:10.1016/j.ic.2005.05.008.

6 Parosh Aziz Abdulla, Karlis Cerans, Bengt Jonsson, and Yih-Kuen Tsay. General decidability theorems for infinite-state systems. In *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996*, pages 313–321. IEEE Computer Society, 1996. doi:10.1109/LICS.1996.561359.

7 Parosh Aziz Abdulla, Lorenzo Clemente, Richard Mayr, and Sven Sandberg. Stochastic parity games on lossy channel systems. *Log. Methods Comput. Sci.*, 10(4), 2014. doi:10.2168/LMCS-10(4:21)2014.

8 Parosh Aziz Abdulla, Noomene Ben Henda, Luca de Alfaro, Richard Mayr, and Sven Sandberg. Stochastic games with lossy channels. In Roberto M. Amadio, editor, *Foundations of Software Science and Computational Structures, 11th International Conference, FOSSACS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29 – April 6, 2008. Proceedings*, volume 4962 of *Lecture Notes in Computer Science*, pages 35–49. Springer, 2008. doi:10.1007/978-3-540-78499-9_4.

9 Parosh Aziz Abdulla and Bengt Jonsson. Verifying programs with unreliable channels. In *Proceedings of the Eighth Annual Symposium on Logic in Computer Science (LICS '93), Montreal, Canada, June 19-23, 1993*, pages 160–170. IEEE Computer Society, 1993. doi:10.1109/LICS.1993.287591.

10 Parosh Aziz Abdulla and Bengt Jonsson. Model checking of systems with many identical timed processes. *Theor. Comput. Sci.*, 290(1):241–264, 2003. doi:10.1016/S0304-3975(01)00330-9.

11 Benjamin Aminof, Marta Kwiatkowska, Bastien Maubert, Aniello Murano, and Sasha Rubin. Probabilistic strategy logic. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 32–38, 2019.

12 Benjamin Aminof, Axel Legay, Aniello Murano, Olivier Serre, and Moshe Y Vardi. Pushdown module checking with imperfect information. *Information and Computation*, 223:1–17, 2013.

13 Robert J Aumann. Acceptable points in general cooperative n-person games. *Contributions to the Theory of Games*, 4(40):287–324, 1959.

14 Robert J. Aumann. The core of a cooperative game without side payments. *Transactions of the American Mathematical Society*, 98(3):539–552, 1961. URL: http://www.jstor.org/stable/1993348.

15 Christel Baier, Nathalie Bertrand, and Philippe Schnoebelen. A note on the attractor-property of infinite-state Markov chains. *Inf. Process. Lett.*, 97(2):58–63, 2006. doi:10.1016/j.ipl.2005.09.011.

16 Christel Baier, Nathalie Bertrand, and Philippe Schnoebelen. Verifying nondeterministic probabilistic channel systems against ω-regular linear-time properties. *ACM Trans. Comput. Log.*, 9(1):5, 2007. doi:10.1145/1297658.1297663.

17 Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.

18 B Douglas Bernheim, Bezalel Peleg, and Michael D Whinston. Coalition-proof Nash equilibria i. concepts. *Journal of economic theory*, 42(1):1–12, 1987.

19 Nathalie Bertrand and Philippe Schnoebelen. Model checking lossy channels systems is probably decidable. In Andrew D. Gordon, editor, *Foundations of Software Science and Computational Structures, 6th International Conference, FOSSACS 2003 Held as Part of the Joint European Conference on Theory and Practice of Software, ETAPS 2003, Warsaw, Poland, April 7-11, 2003, Proceedings*, volume 2620 of *Lecture Notes in Computer Science*, pages 120–135. Springer, 2003. doi:10.1007/3-540-36576-1_8.

20 Patricia Bouyer, Romain Brenguier, Nicolas Markey, and Michael Ummels. Pure Nash equilibria in concurrent deterministic games. *Log. Methods Comput. Sci.*, 11(2), 2015. doi:10.2168/LMCS-11(2:9)2015.

21 Laura Bozzelli, Aniello Murano, and Adriano Peron. Module checking of pushdown multi-agent systems. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, volume 17, pages 162–171, 2020.

22 Taolue Chen, Fu Song, and Zhilin Wu. Global model checking on pushdown multi-agent systems. In Dale Schuurmans and Michael P. Wellman, editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 2459–2465. AAAI Press, 2016. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/11984, doi:10.1609/AAAI.V30I1.10124.

23 Luca de Alfaro, Thomas A. Henzinger, and Orna Kupferman. Concurrent reachability games. *Theor. Comput. Sci.*, 386(3):188–217, 2007. doi:10.1016/j.tcs.2007.07.008.

24 Javier Esparza, Antonín Kucera, and Stefan Schwoon. Model checking LTL with regular valuations for pushdown systems. *Inf. Comput.*, 186(2):355–376, 2003. doi:10.1016/S0890-5401(03)00139-1.

25 Alain Finkel and Philippe Schnoebelen. Well-structured transition systems everywhere! *Theor. Comput. Sci.*, 256(1-2):63–92, 2001. doi:10.1016/S0304-3975(00)00102-X.

26 Julian Gutierrez, Lewis Hammond, Anthony W. Lin, Muhammad Najib, and Michael J. Wooldridge. Rational verification for probabilistic systems. In Meghyn Bienvenu, Gerhard Lakemeyer, and Esra Erdem, editors, *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021, Online event, November 3-12, 2021*, pages 312–322, 2021. doi:10.24963/kr.2021/30.

27 Julian Gutierrez, Paul Harrenstein, and Michael Wooldridge. From model checking to equilibrium checking: Reactive modules for rational verification. *Artificial Intelligence*, 248:123–157, 2017.

28 Julian Gutierrez, Paul Harrenstein, and Michael J. Wooldridge. Expresiveness and complexity results for strategic reasoning. In Luca Aceto and David de Frutos-Escrig, editors, *26th International Conference on Concurrency Theory, CONCUR 2015, Madrid, Spain, September 1.4, 2015*, volume 42 of *LIPIcs*, pages 268–282. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.CONCUR.2015.268.

29 Julian Gutierrez, Szymon Kowara, Sarit Kraus, Thomas Steeples, and Michael Wooldridge. Cooperative concurrent games. *Artificial Intelligence*, 314:103806, 2023.

30 Julian Gutierrez, Sarit Kraus, and Michael J. Wooldridge. Cooperative concurrent games. In Edith Elkind, Manuela Veloso, Noa Agmon, and Matthew E. Taylor, editors, *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, Montreal, QC, Canada, May 13-17, 2019*, pages 1198–1206. International Foundation for Autonomous Agents and Multiagent Systems, 2019. URL: http://dl.acm.org/citation.cfm?id=3331822.

31 Julian Gutierrez, Muhammad Najib, Giuseppe Perelli, and Michael J. Wooldridge. Automated temporal equilibrium analysis: Verification and synthesis of multi-player games. *Artif. Intell.*, 287:103353, 2020. doi:10.1016/j.artint.2020.103353.

32 Graham Higman. Ordering by divisibility in abstract algebras. In *Proc. London Math. Soc.*, volume 2, pages 326–336, 1952.

33 John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation, 2nd Edition.* Addison-Wesley series in computer science. Addison-Wesley-Longman, 2001.

34 Ori Lahav, Egor Namakonov, Jonas Oberhauser, Anton Podkopaev, and Viktor Vafeiadis. Making weak memory models fair. *Proc. ACM Program. Lang.*, 5(OOPSLA):1–27, 2021. doi:10.1145/3485475.

35 Donald A. Martin. Borel determinacy. *Annals of Mathematics*, 102(2):363–371, 1975. URL: http://www.jstor.org/stable/1971035.

36 Richard Mayr. Undecidable problems in unreliable computations. *Theor. Comput. Sci.*, 297(1-3):337–354, 2003. doi:10.1016/S0304-3975(02)00646-1.

37 Aniello Murano and Giuseppe Perelli. Pushdown multi-agent system verification. In Qiang Yang and Michael J. Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1090–1097. AAAI Press, 2015. URL: `http://ijcai.org/Abstract/15/158`.

38 John F. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences of the United States of America*, 36(1):48–49, 1950.

39 Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October – 1 November 1977*, pages 46–57. IEEE Computer Society, 1977. `doi:10.1109/SFCS.1977.32`.

40 Halsey L. Royden and Patrick M. Fitzpatrick. *Real Analysis*. Prentice Hall, 2010. URL: `https://books.google.fr/books?id=H65bQgAACAAJ`.

41 Herbert E Scarf. The core of an n person game. *Econometrica: Journal of the Econometric Society*, pages 50–69, 1967.

42 Philippe Schnoebelen. Verifying lossy channel systems has nonprimitive recursive complexity. *Inf. Process. Lett.*, 83(5):251–261, 2002. `doi:10.1016/S0020-0190(01)00337-4`.

43 Tobias Winkler and Maximilian Weininger. Stochastic games with disjunctions of multiple objectives. In Pierre Ganty and Davide Bresolin, editors, *Proceedings 12th International Symposium on Games, Automata, Logics, and Formal Verification, GandALF 2021, Padua, Italy, 20-22 September 2021*, volume 346 of *EPTCS*, pages 83–100, 2021. `doi:10.4204/EPTCS.346.6`.

Towards Univalent Reference Types

The Impact of Univalence on Denotational Semantics

Jonathan Sterling
University of Cambridge, UK

Daniel Gratzer
Aarhus University, Denmark

Lars Birkedal
Aarhus University, Denmark

Abstract

We develop a denotational semantics for general reference types in an impredicative version of **guarded homotopy type theory**, an adaptation of synthetic guarded domain theory to Voevodsky's univalent foundations. We observe for the first time the profound impact of univalence on the denotational semantics of mutable state. Univalence automatically ensures that all computations are invariant under symmetries of the heap – a bountiful source of program equivalences. In particular, even the most simplistic univalent model enjoys many new equations that do not hold when the same constructions are carried out in the universes of traditional set-level (extensional) type theory.

2012 ACM Subject Classification Theory of computation → Denotational semantics; Theory of computation → Categorical semantics; Theory of computation → Type structures; Theory of computation → Type theory

Keywords and phrases univalent foundations, homotopy type theory, impredicative encodings, synthetic guarded domain theory, guarded recursion, higher-order store, reference types

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.47

Related Version *Full Version*: `https://arxiv.org/abs/2307.16608`

Funding This work was supported in part by a Villum Investigator grant (no. 25804), Center for Basic Research in Program Verification (CPV), from the VILLUM Foundation.
Jonathan Sterling: Jonathan Sterling was funded in part by the European Union under the Marie Skłodowska-Curie Actions Postdoctoral Fellowship project *TypeSynth: synthetic methods in program verification*, and in part by AFOSR under grant FA9550-23-1-0728, *New Spaces for Denotational Semantics* (Tristan Nguyen, program manager). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union, the European Commission, nor AFOSR. Neither the European Union nor the granting authority nor AFOSR can be held responsible for them.

Acknowledgements We are thankful to Carlo Angiuli, Steve Awodey, and Robert Harper for teaching us the importance of realizability methods in homotopy type theory. We thank Zhixuan Yang for proof-reading. Finally, we are grateful to the anonymous referees for their comments and suggestions.

1 Introduction

Moggi [32] famously distinguished three semantics-based approaches to proving equivalences between programs: **operational**, **denotational**, and **logical**. Operational semantics studies programs *indirectly* by investigating the properties of a transition function that executes programs *qua* code on a highly specific idealized computer; in contrast, denotational semantics views programs *directly* as functions on highly specialized kinds of spaces, without making any detour through transition functions. Moggi's departure is to advance a *logical* approach to program equivalence, in which a programming language is an equational theory equipped with a *category* of denotational models for which it is both sound and complete.

Moggi's logical approach to program equivalence therefore subsumes traditional denotational semantics: both the general and the particular necessarily exude their own depth and sophistication, but they are now correctly situated in relation to each other so that workers in semantics can reap the greatest benefits from the *theory–model* dialectic:

1. Even if there is a distinguished "standard" model of a given programming language (*e.g.* the Scott model of PCF), any non-trivial investigation of the syntax of that language necessarily involves non-standard models – if only because induction can always be seen as a model construction. These non-standard models include the *generic* model, built from the theory itself, as well as models based on logical relations; thus the need for clear thinking about many models via logical semantics cannot be bypassed.
2. Conversely, the discovery of a new model of a programming language can inspire and justify the refinement of its equational theory: for instance, *parametric models* have been used to justify equational theories for data abstraction and local store. In the other direction, the discovery of a "non-model" that nonetheless has desirable properties can open up new semantic vistas by motivating a relaxed equational theory.

1.1 State and reference types: static and dynamic allocation

One of the oldest programming constructs is *state*: the ability to read from and write to the computer's memory as a side effect. Theories of state delineate themselves along two axes: (1) the kinds of data that can be stored, and (2) the kinds of allocations allowed. On the first axis, languages range from being able to store integers and strings (**first-order store**) all the way to being able to store elements of arbitrary types, including closures (**higher-order store**). On the second axis, we have **static allocation** on one end, where the type of a function specifies exactly what kind of state it uses, and **dynamic allocation** on the other end, where the types and quantity of memory cells allocated are revealed only during execution. Under dynamic allocation, one has **reference types** whose elements are *pointers* to memory cells storing elements of a given type.

1.2 Equational theories of dynamic storage: between local and global

The semantics of state are only difficult under dynamic allocation; indeed, computations that interact with a statically known heap configuration $\overline{\ell_i : \sigma_i}$ can be classified by Moggi's state monad $\sigma \to \sigma \times -$ where $\sigma :\equiv \prod_i \sigma_i$, and it is reasonable to *define* the equational theory of static allocation by means of this interpretation. The equational theory of *dynamic* storage is by contrast far from solidified: the introduction of dynamic allocation opens up a spectrum of abstraction between what may be called **local store** and **global store**.

Global store is the least abstract theory of dynamic allocation: in a model of global store, it is permitted that allocations be globally observable regardless of their impact on the results of computations. For instance, global store models are allowed to distinguish the program $(\ell \leftarrow \mathsf{alloc}\ \texttt{"hello"}; \mathsf{ret}\ 10)$ from the simpler program $\mathsf{ret}\ 10$. By contrast, models of *local store* validate equations resembling an idealized garbage collector, in which the heap is only observable through its abstract read/write interface; in a model of local store, we necessarily have $(\ell \leftarrow \mathsf{alloc}\ \texttt{"hello"}; \mathsf{ret}\ 10) = \mathsf{ret}\ 10$ as well as many other equations.

The abstraction offered by local store is highly desirable. Moreover, Staton [42] has shown that Plotkin and Power's algebraic theory of *first-order* local store [38] is complete in the extremely strong sense that it derives any consistent equation. Beyond first-order references, the very definition of the local store theory becomes less clear, and so a landscape

of intermediate theories has emerged in the search for well-behaved models. For example, Kammar et al. [27] have constructed a compelling model of local **full ground store**, going beyond first-order store by allowing pointers to pointers. On the other hand, Levy [28, 29, 30] has given a domain theoretic model of the *global* allocation theory of higher-order store.

1.3 Semantic worlds and guarded models of higher-order store

Denotational models of full dynamic allocation, such as those of Plotkin and Power [38], Levy [28], and Kammar et al. [27], tend to share an important limitation: in the model, a semantic program can only allocate a memory cell with a *syntactic* type. This restriction is quite unnatural and impractical in the context of higher-order store, where many important program equivalences actually follow from the presence of exotic semantic types lying outside the image of the interpretation function (*e.g.* in relational models à la Girard and Reynolds).

The search for models of general references closed under allocation of cells with *semantic* types has been major motivation of current work in *guarded domain theory*, expressed in operational semantics by *step-indexing* [8, 3] and in denotational semantics by means of various generalizations of metric space [10, 6, 21, 17]. The problem solved by guarded domain theory is the following famous circularity described in several prior works [3, 9, 17]:

1. A semantic type needs to be some kind of covariant family of predomains indexed in the possible configurations of the heap ("worlds"); a single predomain won't do, because the elements of type $\mathsf{IORef}\,\sigma$ vary depending on what cells have been allocated.
2. A semantic world should be a finite mapping from memory locations to semantic types.

Guarded domain theory approximates a solution to the domain equation evoked above by decreasing precision at every recursive occurrence. Although it may be possible to find a fully precise solution to this domain equation using traditional domain theory, Birkedal et al. [18, §5] have presented evidence that such a fixed point will *not* brook the interpretation of reference types by a continuous function on the domain of all types, ruling out semantics for recursive types. Thus guarded domain theory or step-indexing would seem to be mandatory for *functional* models of general reference types with semantic worlds.[1]

Models of guarded domain theory can be embedded into topoi whose internal language is referred to as **synthetic guarded domain theory** or *SGDT*; the most famous of these topoi is the **topos of trees** [17] given by presheaves on ω. The idea of using synthetic guarded domain theory as a setting for the naïve denotational semantics of programming languages with general recursion was first explored by Paviotti, Møgelberg, and Birkedal [36, 31, 35].

Sterling, Gratzer, and Birkedal [43] have recently extended the program of Paviotti et al. to the general case of full higher-order store with polymorphism and recursive types: in particular, *op. cit.* have shown how to model general reference types in synthetic guarded domain theory assuming an impredicative universe (as can be found in realizability models [25, 26]). This model is the starting point of the present paper: by adapting the construction of Sterling et al. to the setting of univalent foundations, we obtain a new suite of equational reasoning principles that we refer to as the *theory of univalent reference types*.

[1] A non-functional approach to compositionality for reference types would be the expression of Reynolds' *capability interpretation* of references [39] in game semantics by Abramsky, Honda, and McCusker [2].

1.4 Univalent reference types and data abstraction in the heap

The thesis of this paper is that Voevodsky's *univalence* principle leads to simpler models of general reference types that nonetheless validate extraordinarily strong equations between stateful programs. To examine this claim, we consider the type of object-oriented counters in a Haskell-like language:

$$\mathsf{Counter} :\equiv \{\mathsf{incr} : \mathsf{IO}\,(); \mathsf{read} : \mathsf{IO}\,\mathsf{Int}\}$$

The most obvious implementation of the Counter interface simply allocates an integer and increments it in memory as follows:

$$\begin{aligned}&\mathsf{posCounter} : \mathsf{IO}\,\mathsf{Counter}\\ &\mathsf{posCounter} :\equiv \ell \leftarrow \mathsf{alloc}\,0; \mathsf{ret}\,\{\mathsf{incr} \hookrightarrow i \leftarrow \mathsf{get}\,\ell; \mathsf{set}\,\ell\,(i+1), \mathsf{read} \hookrightarrow \mathsf{get}\,\ell\}\end{aligned}$$

Another implementation might count *backwards* and then negate the stored value on read using the functorial action map of IO on $\mathsf{neg} : \mathsf{Int} \to \mathsf{Int}$:

$$\mathsf{negCounter} :\equiv \ell \leftarrow \mathsf{alloc}\,0; \mathsf{ret}\,\{\mathsf{incr} \hookrightarrow i \leftarrow \mathsf{get}\,\ell; \mathsf{set}\,\ell\,(i-1), \mathsf{read} \hookrightarrow \mathsf{map}\,\mathsf{neg}\,(\mathsf{get}\,\ell)\}$$

By intuition, the posCounter and negCounter implementations of the counter interface should be "observationally equivalent" in the sense that no context of ground type should be able to distinguish them: indeed, even though negCounter is writing negative numbers to the heap instead of positive numbers, the only way a context can observe the allocated cell is using the read method. The observational equivalence $\mathsf{posCounter} \simeq \mathsf{negCounter}$ is typically proved using a *relational model*, as both Birkedal et al. [18, §6.3] and Sterling et al. [43] did.

Observational equivalence is not the same as equality, neither in syntax nor semantics. Indeed, typical equational theories of (local or global) dynamic allocation do *not* derive the equation $\vdash \mathsf{posCounter} \equiv \mathsf{negCounter}$, as can be seen easily by means of a countermodel: we have $[\![\mathsf{posCounter}]\!] \neq [\![\mathsf{negCounter}]\!]$ in both the relational models of *op. cit.*, although it is true that $[\![\mathsf{posCounter}]\!]\ R_{\mathsf{IO\,Counter}}\ [\![\mathsf{negCounter}]\!]$ holds. The observational equivalence $\mathsf{posCounter} \simeq \mathsf{negCounter}$ is deduced in the relational models because the relation on *observations* is discrete.

What distinguishes our **univalent reference types** from ordinary reference types is that the former actually derive equations like $\vdash \mathsf{posCounter} \equiv \mathsf{negCounter}$, as shown in Theorem 2.1. We substantiate this equational theory by constructing a model (Theorem 3.27) in univalent foundations [46] in which the equation $\vdash \mathsf{posCounter} \equiv \mathsf{negCounter}$ follows immediately from the univalence principle of the metalanguage. Although it may be possible to validate this equation using non-standard parametric models (or less scrupulously by an extensional collapse), our contribution is to show that it also holds in a "standard" model, provided that this standard model is constructed in a univalent metatheory.

2 A higher-order language with (univalent) reference types

We begin by giving a description of the syntax and the equational theory of a simple language with references. The language is meant to be *as simple as possible, but no simpler*. In particular, it contains several problematic constructs (higher-order store, dynamic allocations, *etc.*) that have been historically difficult to model in denotational semantics.

$$\begin{array}{lrcl} \textit{types} & \tau, \sigma & ::= & \sigma \to \tau \mid \mathsf{IO}\,\tau \mid \mathsf{IORef}\,\tau \mid \dots \\ \textit{terms} & e, e' & ::= & x \mid \mathsf{rec}\ f\ x\ \mathsf{in}\ e \mid \mathsf{ret}\,e \mid x \leftarrow e;\ e' \mid \mathsf{alloc}\,e \mid \mathsf{get}\,e \mid \mathsf{set}\,e\,e' \mid \mathsf{step} \mid \dots \end{array}$$

$$\frac{\Gamma \vdash e : \sigma}{\Gamma \vdash \mathsf{alloc}\,e : \mathsf{IO}\,(\mathsf{IORef}\,\sigma)} \qquad \frac{\Gamma \vdash e : \mathsf{IORef}\,\sigma}{\Gamma \vdash \mathsf{get}\,e : \mathsf{IO}\,\sigma} \qquad \frac{\Gamma \vdash e : \mathsf{IORef}\,\sigma \qquad \Gamma \vdash e' : \sigma}{\Gamma \vdash \mathsf{set}\,e\,e' : \mathsf{IO}\,()} \qquad \frac{}{\Gamma \vdash \mathsf{step} : \mathsf{IO}\,()}$$

$$\frac{\Gamma, f : \sigma \to \mathsf{IO}\,\tau, x : \sigma \vdash e : \mathsf{IO}\,\tau}{\Gamma \vdash \mathsf{rec}\ f\ x\ \mathsf{in}\ e : \sigma \to \mathsf{IO}\,\tau} \qquad \frac{\Gamma, f : \sigma \to \mathsf{IO}\,\tau, x : \sigma \vdash e : \mathsf{IO}\,\tau \qquad \Gamma \vdash e' : \sigma}{\Gamma \vdash (\mathsf{rec}\ f\ x\ \mathsf{in}\ e)\,e' \equiv \mathsf{step}; [(\mathsf{rec}\ f\ x\ \mathsf{in}\ e)/f, e'/x]e : \mathsf{IO}\,\tau}$$

$$e : \mathsf{IORef}\,\sigma, e' : \sigma \vdash \mathsf{set}\,e\,e'; \mathsf{get}\,e \equiv \mathsf{step}; \mathsf{set}\,e\,e'; \mathsf{ret}\,e' : \mathsf{IO}\,\sigma$$

$$e : \sigma, e' : \sigma \vdash (x \leftarrow \mathsf{alloc}\,e; \mathsf{set}\,x\,e'; \mathsf{ret}\,x) \equiv \mathsf{alloc}\,e' : \mathsf{IO}\,(\mathsf{IORef}\,\sigma)$$

$$e : \mathsf{IORef}\,\sigma, e' : \sigma, e'' : \sigma \vdash \mathsf{set}\,e\,e'; \mathsf{set}\,e\,e'' \equiv \mathsf{set}\,e\,e'' : \mathsf{IO}\,()$$

$$e : \mathsf{IORef}\,\sigma, e' : \mathsf{IORef}\,\tau \vdash (x \leftarrow \mathsf{get}\,e; y \leftarrow \mathsf{get}\,e'; \mathsf{ret}\,\langle x, y\rangle) \equiv (y \leftarrow \mathsf{get}\,e'; x \leftarrow \mathsf{get}\,e; \mathsf{ret}\,\langle x, y\rangle) : \mathsf{IO}\,(\sigma \times \tau)$$

$$e : \mathsf{IORef}\,\sigma \vdash (x \leftarrow \mathsf{get}\,e; \mathsf{set}\,e\,x; \mathsf{ret}\,x) \equiv \mathsf{get}\,e : \mathsf{IO}\,\sigma$$

$$e : \mathsf{IORef}\,\sigma, e' : \mathsf{IO}\,\tau \vdash \mathsf{get}\,e; e' \equiv \mathsf{step}; e'$$

Figure 1 Syntax and selected typing and equational rules for a higher-order monadic language with general reference types. We assume standard notational conventions for monadic programming, *e.g.* writing $e; e'$ for $\_ \leftarrow e; e'$. We assume the standard β/η-equational theory of function and product types, as well as the monadic laws. We also assume that step lies in the *center* [22] of the monad IO, *i.e.* commutes with all monadic operations.

2.1 The equational theory of monadic general reference types

Although there are many different ways to present programming languages with side effects, for the sake of familiarity we have chosen to focus on a variant of Moggi's *monadic metalanguage* [32].[2] Essentially, this is a simply-typed lambda calculus supplemented with a *strong monad* IO and further equipped with a type of references $\mathsf{IORef}\,\tau$ along with a suite of effectful operations for interacting with references. Like in Haskell, all side effects are confined to the monad; unlike Haskell, general recursion is treated as a side effect.

One non-standard aspect of our language bears special attention, namely the nullary side effect $\mathsf{step} : \mathsf{IO}\,()$. This effect can be thought of as the "exhaust" left behind in the equational theory by unfolding any kind of recursively defined construct, including not only the unfolding of recursive functions but also accesses to the heap. In particular, for a given recursive function $g :\equiv \mathsf{rec}\ f\ x\ \mathsf{in}\ e$, we do not have $\vdash g\,e' \equiv [g/f, e'/x]e$ but rather only $\vdash g\,e' \equiv \mathsf{step}; [g/f, e'/x]e$. Likewise, our equational theory does not equate $\vdash (\ell \leftarrow \mathsf{alloc}\,e; \mathsf{get}\,\ell) \equiv \mathsf{ret}\,e$ but rather only $\vdash (\ell \leftarrow \mathsf{alloc}\,e; \mathsf{get}\,\ell) \equiv \mathsf{step}; \mathsf{ret}\,e$. The presence of step in our equational theory is forced by the *guarded* denotational semantics that we will later employ in Section 3 and Theorem 3.27.

2.2 The equational theory of univalent reference types

The equational theory of **univalent reference types** strengthens Figure 1 by quotienting under symmetries of the heap, expressed in the two rules depicted in Figure 2.

[2] When developing our denotational semantics in Section 3, we will refine the monadic point of view by passing to an adjoint call-by-push-value resolution of the computational monad [29].

ALLOCATION PERMUTATION

$$\frac{\Gamma \vdash e : \sigma \qquad \Gamma \vdash e' : \tau}{\Gamma \vdash \ell \leftarrow \mathsf{alloc}\, e; \ell' \leftarrow \mathsf{alloc}\, e'; \mathsf{ret}\, \langle \ell, \ell' \rangle \equiv \ell' \leftarrow \mathsf{alloc}\, e'; \ell \leftarrow \mathsf{alloc}\, e; \mathsf{ret}\, \langle \ell, \ell' \rangle : \mathsf{IO}\, (\mathsf{IORef}\, \sigma \times \mathsf{IORef}\, \tau)}$$

REPRESENTATION INDEPENDENCE

$$\frac{\begin{array}{c}\Gamma \vdash e : \sigma \qquad \Gamma \vdash f^+ : \sigma \to \tau \qquad \Gamma \vdash f^- : \tau \to \sigma \\ \Gamma, x : \tau \vdash f^+(f^- x) \equiv x : \tau \qquad \Gamma, x : \sigma \vdash f^-(f^+ x) \equiv x : \sigma\end{array}}{\Gamma \vdash \ell \leftarrow \mathsf{alloc}\, e; \mathsf{ret}\, \langle \mathsf{get}\, \ell, \mathsf{set}\, \ell \rangle \equiv \ell \leftarrow \mathsf{alloc}\, (f^+ e); \mathsf{ret}\, \langle \mathsf{map}\, f^- (\mathsf{get}\, \ell), \mathsf{set}\, \ell \circ f^+ \rangle : \mathsf{IO}\, (\mathsf{Cell}\, \sigma)}$$

Figure 2 The equational theory of **univalent reference types**, extending that of Figure 1; we define $\mathsf{Cell}\, \sigma :\equiv \mathsf{IO}\, \sigma \times (\sigma \to \mathsf{IO}\, ())$ to be the "abstract interface" of a reference cell. Here we write $\mathsf{map}\, f : \mathsf{IO}\, A \to \mathsf{IO}\, B$ for the functorial action of IO on a function $f : A \to B$.

1. The ALLOCATION PERMUTATION rule states that the order in which references are allocated does not matter; this is a kind of *nominal symmetry* built into the theory of univalent reference types, expressing that the *layout* of the heap is viewed up to isomorphism.
2. The REPRESENTATION INDEPENDENCE rule states that the *observable interface* of a reference cell is invariant under isomorphisms of that cell's contents.

The ALLOCATION PERMUTATION rule is common to theories of local dynamic allocation, but less common in theories of global dynamic allocation. The REPRESENTATION INDEPENDENCE rule is, however, a new feature of univalent reference types that goes beyond existing local theories of dynamic allocation: as we have discussed in Section 1.4, such a law typically holds up to observational equivalence but almost never "on the nose" at higher type. It is therefore worth going into more detail.

The idea of REPRESENTATION INDEPENDENCE is that allocating a cell of type σ and then only interacting with it by means of its $(\mathsf{get}, \mathsf{set})$ methods should be the same as allocating a cell of a different type τ and interacting with it by conjugating its $(\mathsf{get}, \mathsf{set})$ interface by an isomorphism $e : \sigma \cong \tau$. In particular, it is allowed that $\sigma \equiv \tau$ and $e : \sigma \cong \sigma$ be nonetheless a non-trivial automorphism: and so we may derive from REPRESENTATION INDEPENDENCE our case study involving imperative counters that count forward and backwards.

▶ **Theorem 2.1.** *Let* Counter, posCounter, *and* negCounter *be as in Section 1.4:*

$\mathsf{Counter} :\equiv \{\mathsf{incr} : \mathsf{IO}\, (); \mathsf{read} : \mathsf{IO}\, \mathsf{Int}\}$
$\mathsf{posCounter} :\equiv \ell \leftarrow \mathsf{alloc}\, 0; \mathsf{ret}\, \{\mathsf{incr} \hookrightarrow i \leftarrow \mathsf{get}\, \ell; \mathsf{set}\, \ell\, (i+1), \mathsf{read} \hookrightarrow \mathsf{get}\, \ell\}$
$\mathsf{negCounter} :\equiv \ell \leftarrow \mathsf{alloc}\, 0; \mathsf{ret}\, \{\mathsf{incr} \hookrightarrow i \leftarrow \mathsf{get}\, \ell; \mathsf{set}\, \ell\, (i-1), \mathsf{read} \hookrightarrow \mathsf{map}\, \mathsf{neg}\, (\mathsf{get}\, \ell)\}$

We may derive $\vdash \mathsf{posCounter} \equiv \mathsf{negCounter} : \mathsf{Counter}$.

Proof. The function $\mathsf{neg} : \mathsf{Int} \to \mathsf{Int}$ sending an integer to its negation is a self-dual automorphism; we therefore calculate from left to right.

$\ell \leftarrow \mathsf{alloc}\, 0; \mathsf{ret}\, \{\mathsf{incr} \hookrightarrow i \leftarrow \mathsf{get}\, \ell; \mathsf{set}\, \ell\, (i+1), \mathsf{read} \hookrightarrow \mathsf{get}\, \ell\}$

by REPRESENTATION INDEPENDENCE
$\equiv \ell \leftarrow \mathsf{alloc}\, (\mathsf{neg}\, 0); \mathsf{ret}\, \{\mathsf{incr} \hookrightarrow i \leftarrow \mathsf{map}\, \mathsf{neg}\, (\mathsf{get}\, \ell); \mathsf{set}\, \ell\, (\mathsf{neg}\, (i+1)), \mathsf{read} \hookrightarrow \mathsf{map}\, \mathsf{neg}\, (\mathsf{get}\, \ell)\}$
by simplification and $\mathsf{neg}\, 0 \equiv 0$
$\equiv \ell \leftarrow \mathsf{alloc}\, 0; \mathsf{ret}\, \{\mathsf{incr} \hookrightarrow i \leftarrow \mathsf{get}\, \ell; \mathsf{set}\, \ell\, (\mathsf{neg}\, (\mathsf{neg}\, i + 1)), \mathsf{read} \hookrightarrow \mathsf{map}\, \mathsf{neg}\, (\mathsf{get}\, \ell)\}$
by $\mathsf{neg}\, (\mathsf{neg}\, i + 1) \equiv \mathsf{neg}\, (\mathsf{neg}\, i) + \mathsf{neg}\, 1 \equiv i - 1$
$\equiv \ell \leftarrow \mathsf{alloc}\, 0; \mathsf{ret}\, \{\mathsf{incr} \hookrightarrow i \leftarrow \mathsf{get}\, \ell; \mathsf{set}\, \ell\, (i-1), \mathsf{read} \hookrightarrow \mathsf{map}\, \mathsf{neg}\, (\mathsf{get}\, \ell)\}$

Thus we have $\mathsf{posCounter} \equiv \mathsf{negCounter}$. ◀

3 Denotational semantics in univalent foundations

We now turn to the construction of a model of univalent reference types. At the coarsest level, this model follows the standard template for a model with mutable state: types are interpreted by covariant presheaves on a certain category of *worlds* with each world describing the collection of references available and the (semantic) type associated to each. The type of references $\mathsf{IORef}\,\tau$ assigns each world to the collection of locations of appropriate type while the monad IO is then interpreted by a certain *store-passing* monad.

This simple picture is quickly complicated by the need to model general store: semantic types must reference worlds which in turn reference semantic types. This naturally leads us to synthetic guarded domain theory (SGDT) in order to cope with the circularity. This alone, however, is insufficient. While SGDT allows us to define the category of worlds, the resulting solution is a *large type* – at least the size of the universe of semantic types. This becomes a problem when it comes time to model the state monad, which must quantify over all possible worlds for its input and return a new world for its output. To model these large products and sums of worlds, we will base our model on an *impredicative* universe: impredicativity implies that the category of covariant presheaves on our large category of worlds is (locally) cartesian closed and supports all the structure of our language.

We present some of the prerequisites for our model in Section 3.1. In Section 3.2 we construct the model of univalent reference types.

3.1 Univalent impredicative synthetic guarded domain theory

We work informally in the language of homotopy type theory [40, 46]; in this section, we briefly describe some of our preferred conventions. When we speak of "existence", we shall always mean *mere* existence. Categories are always assumed to be univalent 1-categories; given a category $\mathcal{X}$, we will write $|\mathcal{X}|$ for its underlying 1-type of objects. Rather than fixing a global hierarchy of universes, we assume universes locally where needed. In this paper, all universes are assumed to be univalent; when we wish to assume that a universe is closed under the connectives of Martin-Löf type theory (dependent products, dependent sums, finite coproducts, W-types, *etc.*) we will refer to it as a **Martin-Löf universe**. We will not belabor the difference between codes and types.

3.1.1 Impredicative subuniverses in univalent foundations

Recall that a type A is called **$\mathcal{U}$-small** if and only if there exists a (necessarily unique) code $\hat{A} : \mathcal{U}$ together with an equivalence $[\hat{A}] \simeq A$, and a family is $\mathcal{U}$-small when each of its fibers are. A **reflection** of A in a universe $\mathcal{U}$ is, by contrast, defined to be a (necessarily unique) function $\eta : A \to A_{\mathcal{U}}$ with $A_{\mathcal{U}} \in \mathcal{U}$ such for any type $C \in \mathcal{U}$, the precomposition map $C^{\eta} : C^{A_{\mathcal{U}}} \to C^{A}$ is an equivalence. When a reflection of A in $\mathcal{U}$ exists (necessarily uniquely), we shall say that A is reflected in $\mathcal{U}$.

A **subuniverse** of a universe $\mathcal{U}$ is defined to be a dependent type $A : \mathcal{U} \vdash PA\ \mathit{type}$ such that each PA is a proposition. We may write $\mathcal{U}_P$ for the universe $\sum_{A:\mathcal{U}} PA$ obtained by restricting $\mathcal{U}$ to the elements satisfying P. We will frequently abuse notation implicitly identifying the predicate coding a subuniverse with its comprehension as an actual type. A subuniverse $\mathcal{S} \subseteq \mathcal{U}$ is said to be **reflective** if every $A : \mathcal{U}$ is reflected in $\mathcal{S}$. A subuniverse of $\mathcal{U}$ is said to be "small" when its comprehension as a type is $\mathcal{U}$-small.

Let $\mathcal{U}$ be a universe closed under dependent products. A subuniverse $\mathcal{S} \subseteq \mathcal{U}$ is said to be a **dependent exponential ideal** if for every $A : \mathcal{U}$ and $B : A \to \mathcal{S}$, the dependent product $\prod_{x:A} Bx$ lies in $\mathcal{S}$. An **impredicative subuniverse** of $\mathcal{U}$ is defined to be a small,

dependent exponential ideal $\mathcal{S} \subseteq \mathcal{U}$. It is proved by Rijke, Shulman, and Spitters [41] that any reflective subuniverse of $\mathcal{U}$ is a dependent exponential ideal of $\mathcal{U}$. We will refer to a subuniverse $\mathcal{S}$ such that $\mathsf{Set}_{\mathcal{S}}$ is impredicative in $\mathcal{U}$ as **set-impredicative**; we will refer to $\mathcal{S} \subseteq \mathcal{U}$ as **set-reflective** when $\mathsf{Set}_{\mathcal{S}}$ is reflective in $\mathcal{U}$. Under suitable assumptions, these two conditions are in fact equivalent:

▶ **Theorem 3.1.** *A small Σ-closed subuniverse $\mathcal{S}$ of a Martin-Löf universe $\mathcal{U}$ is set-impredicative and closed under identity types if and only if it is set-reflective.*

Proof. This can be shown using the methods of Awodey, Frey, and Speight [13]. ◀

By virtue of Theorem 3.1, we see that small reflective subuniverses are just another presentation of the impredicative universes that appear in the Calculus of Constructions.

3.1.2 The Hofmann–Streicher universe

Let $\mathcal{S}$ be a small subuniverse of a Martin-Löf universe $\mathcal{U}$, and let $\mathcal{X}$ be a $\mathcal{U}$-small category; we can define the **Hofmann–Streicher lifting** [24, 12] of $\mathsf{Set}_{\mathcal{S}}$ as co-presheaf of 1-types on $\mathcal{X}$. Formally, this means constructing a functor from the 1-category $\mathcal{X}$ to the (2,1)-category $1\mathsf{Type}_{\mathcal{U}}$ of 1-types in $\mathcal{U}$; thus we depend technically on the account of bicategories in univalent foundations due to Ahrens et al. [4].[3]

▶ Remark 3.2. The purpose of introducing the Hofmann–Streicher lifting in such detail is give some structure to the otherwise bewildering Construction 3.4, which plays a crucial technical role in the definition of univalent reference types.

▶ **Construction 3.3** (The Hofmann–Streicher lifting)**.** Let $\mathcal{S}$ be a small subuniverse of a Martin-Löf universe $\mathcal{U}$, and let $\mathcal{X}$ be a $\mathcal{U}$-small category. We may define a 2-functor $\lfloor \mathsf{Set}_{\mathcal{S}} \rfloor : \mathcal{X} \to 1\mathsf{Type}_{\mathcal{U}}$ called the **Hofmann–Streicher lifting** of $\mathsf{Set}_{\mathcal{S}}$ as follows:

$$\lfloor \mathsf{Set}_{\mathcal{S}} \rfloor U :\equiv |\mathsf{Fun}(U/\mathcal{X}, \mathsf{Set}_{\mathcal{S}})|$$
$$\lfloor \mathsf{Set}_{\mathcal{S}} \rfloor (f : U \to V)\,(E : U/\mathcal{X} \to \mathsf{Set}_{\mathcal{S}}) :\equiv E \circ (f/\mathcal{X})$$

When $\mathcal{X}$ is viewed as a 2-category, the 2-cells are given by identifications. Thus the 2-functoriality of $\lfloor \mathsf{Set}_{\mathcal{S}} \rfloor$ and all related coherences are defined by path induction.

▶ **Construction 3.4** (Restricting co-presheaves)**.** Let $\mathcal{S}$ be a small subuniverse of a Martin-Löf universe $\mathcal{U}$, and let $\mathcal{X}$ be a $\mathcal{U}$-small category. Every co-presheaf $E : \mathcal{X} \to \mathsf{Set}_{\mathcal{S}}$ determines a global element $\mathbf{1} \to \lfloor \mathsf{Set}_{\mathcal{S}} \rfloor$ of the Hofmann–Streicher universe; in particular, we may define $\lceil E \rceil_U : \lfloor \mathsf{Set}_{\mathcal{S}} \rfloor U$ natural in $U \in \mathcal{X}$ by setting $\lceil E \rceil_U\,(f : U \to V) :\equiv EV$.

3.1.3 (Higher) synthetic guarded domain theory

We adapt Birkedal et al.'s formulation [19] of dependently typed guarded recursion to the setting of homotopy type theory. In particular, we introduce a new syntactic sort of *delayed substitutions* $\vdash \xi \rightsquigarrow \Xi$ simultaneously with a new type former $\vdash \blacktriangleright[\xi].A$ *type* called the **later**

[3] We differ from the conventions of Ahrens et al. [4]: we will say "2-category" to mean *univalent bicategory* in the sense of *op. cit.*, as we are not at all concerned with the strict notions considered there. Therefore, a "(2,1)-category" in our sense refers to a 2-category whose 2-cells are given by identifications.

$$\frac{\xi \rightsquigarrow \Xi \qquad A\ \mathit{type}}{\blacktriangleright[\xi].A\ \mathit{type}} \qquad \frac{\xi \rightsquigarrow \Xi \qquad a : A}{\mathsf{next}[\xi].a : \blacktriangleright[\xi].A} \qquad \frac{}{\cdot \rightsquigarrow \cdot} \qquad \frac{\xi \rightsquigarrow \Xi \qquad a : \blacktriangleright[\xi].A}{(\xi, x \leftarrow a) \rightsquigarrow \Xi, x : A}$$

$$\frac{f : \blacktriangleright A \to A}{\mathsf{fix}_{\blacktriangleright}\ f : A} \qquad \frac{f : \blacktriangleright A \to A}{\mathsf{fix}_{\blacktriangleright}\ f \equiv f\ (\mathsf{next}\ (\mathsf{fix}_{\blacktriangleright}\ f))}$$

Figure 3 Summary of delayed substitutions and the later modality; there are a number of equational rules governing the delayed substitutions, *e.g.* $\blacktriangleright[\xi, x \leftarrow a].A \equiv \blacktriangleright[\xi].A$ for any A in which x does not appear; we also assume $(\blacktriangleright[\xi].a = b) \simeq (\mathsf{next}[\xi].a = \mathsf{next}[\xi].b)$, making $\blacktriangleright$ left exact. We will write $\blacktriangleright A$ and $\mathsf{next}\, a$ for $\blacktriangleright[\cdot].A$ and $\mathsf{next}[\cdot].a$ respectively. For the remaining rules, we refer the reader to the description of Bizjak and Møgelberg [20].

modality,[4] whose introduction form is written $\mathsf{next}[\xi].a$; we summarize the rules for the later modality in Figure 3. The *raison d'être* for the later modality is to form **guarded fixed points**: in particular, if we have $f : \blacktriangleright A \to A$, there is a *unique* element $\mathsf{fix}_{\blacktriangleright} f : A$ such that $f\ (\mathsf{next}\ (\mathsf{fix}_{\blacktriangleright} f)) = \mathsf{fix}_{\blacktriangleright} f$. In particular, this gives unique fixed points for any function $f : A \to A$ factoring on the left through $\mathsf{next} : A \to \blacktriangleright A$.

▶ **Definition 3.5.** *A **guarded (n-)domain** is an (n-)type A equipped with the structure of a $\blacktriangleright$-algebra,* i.e. *a function $\vartheta_A : \blacktriangleright A \to A$.*

We will refer to a (sub)universe closed under later modalities as a **guarded (sub)universe**. For any universe $\mathcal{S}$, we may consider the category $\mathsf{0Dom}_{\mathcal{S}}$ of **guarded 0-domains** in $\mathcal{S}$, *i.e.* sets $A : \mathsf{Set}_{\mathcal{S}}$ equipped with a mapping $\vartheta_A : \blacktriangleright A \to A$.

▶ **Lemma 3.6.** *If $\mathcal{S}$ is a guarded universe closed under binary coproducts, then the forgetful functor $\mathsf{R} : \mathsf{0Dom}_{\mathcal{S}} \to \mathsf{Set}_{\mathcal{S}}$ has a left adjoint $\mathsf{L} : \mathsf{Set}_{\mathcal{S}} \to \mathsf{0Dom}_{\mathcal{S}}$.*

Proof. We define $\mathsf{L}A$ by solving the domain equation $\mathsf{L}A \cong A + \blacktriangleright \mathsf{L}A$ via the following guarded fixed point construction in $\mathsf{Set}_{\mathcal{S}}$, using both guarded structure and binary coproducts:

$$\mathsf{L}A :\equiv \mathsf{fix}_{\blacktriangleright}(\lambda X : \blacktriangleright \mathsf{Set}_{\mathcal{S}}.A + \blacktriangleright[Y \leftarrow X].Y)$$ ◀

▶ **Notation 3.7.** We will write $\mathsf{now} : A \to \mathsf{R}\mathsf{L}A$ for the unit of the adjunction $\mathsf{L} \dashv \mathsf{R}$.

▶ **Lemma 3.8** (Later modality in presheaves)**.** *Given a guarded set-reflective small subuniverse $\mathcal{S} \subseteq \mathcal{U}$ and a $\mathcal{U}$-small category $\mathcal{X}$, the later modality from $\mathcal{S}$ lifts (with all its operations) into $\mathsf{Fun}(\mathcal{X}, \mathsf{Set}_{\mathcal{S}})$ pointwise,* i.e. *for any $A \in \mathsf{Fun}(\mathcal{X}, \mathsf{Set}_{\mathcal{S}})$ we may define $(\blacktriangleright A)U :\equiv \blacktriangleright(AU)$.*

3.2 Models of univalent general reference types

To construct our model of higher-order store (Section 3.2.3), we must construct a suitable category of recursively defined semantic worlds (Section 3.2.1) whose co-presheaves admit reference types (Construction 3.13) and a strong monad for higher-order store (Section 3.2.2).

[4] The "later modality" is *not* a modality in the sense of Rijke, Shulman, and Spitters [41], but rather in the older and more general sense of modality in type theory or logic.

3.2.1 Worlds as univalent heap configurations

Let Inj be the category of finite sets and embeddings; by univalence, any two equipollent finite sets are identified. We now define the basic elements of worlds *qua* heap configurations.

▶ **Definition 3.9** (The displayed category of families)**.** *For any 1-type X, we define the displayed category [5] IFam_X of* Inj***-families in*** *X over Inj as follows:*

1. *over a finite set $I : \mathsf{Inj}$, a displayed object of IFam_X is a function $\partial_I : I \to X$;*
2. *over a function $f : I \to J$ between finite sets, a displayed morphism from ∂_I to ∂_J is a path $\partial_f : \partial_J \circ f = \partial_I$ in $I \to X$.*

▶ **Definition 3.10** (The category of bags)**.** *For any 1-type X, the category of* Inj***-bags in*** *X is defined to be the total category $\mathsf{IBag}_X :\equiv \int_{\mathsf{Inj}} \mathsf{IFam}_X$ of the displayed category of finite families in X. We will write $U \equiv (|U|, \partial_U)$ for an object of IBag_X.*

▶ **Definition 3.11.** *For a universe $\mathcal{S}$, we define the category $\mathcal{C}_\mathcal{S}$ of* ***worlds*** *simultaneously with its category of $\mathsf{Set}_\mathcal{S}$-valued co-presheaves on $\mathcal{C}_\mathcal{S}$ to be the unique solution to the guarded recursive domain equation $\mathcal{C}_\mathcal{S} = \mathsf{IBag}_{\blacktriangleright|\mathsf{Fun}(\mathcal{C}_\mathcal{S},\mathsf{Set}_\mathcal{S})|}$.*

Construction. The system of equations above is solved internally [16] by Löb induction in any guarded Martin-Löf universe $\mathcal{S}^+$ containing $\mathcal{S}$.

$$E :\equiv \mathsf{fix}_\blacktriangleright(\lambda R : \blacktriangleright\mathcal{S}^+.|\mathsf{Fun}(\mathsf{IBag}_{\vartheta_{\mathcal{S}^+} R}, \mathsf{Set}_\mathcal{S})|) \qquad \mathcal{C}_\mathcal{S} :\equiv \mathsf{IBag}_{\blacktriangleright E}$$

Of course, E is the 1-type of objects of the functor category $\mathsf{Fun}(\mathcal{C}_\mathcal{S}, \mathsf{Set}_\mathcal{S})$. ◀

We shall require the following technical observation:

▶ **Lemma 3.12** (Structure identity principle for presheaves)**.** *Let $\mathcal{S}$ be a universe and let $\mathcal{X}$ be a category; for any $A, B : \mathsf{Fun}(\mathcal{X}, \mathsf{Set}_\mathcal{S})$, let $A \cong B$ be the type of natural isomorphisms between presheaves. Then the canonical map $A = B \to A \cong B$ is an equivalence.*

We now come to the construction of the univalent reference type constructor.

▶ **Construction 3.13** (Univalent references)**.** Let $\mathcal{S}$ be a small, guarded, Σ-closed set-reflective subuniverse of a guarded Martin-Löf universe $\mathcal{U}$ containing Inj. We define the **univalent reference type constructor** as a mapping $\mathsf{IORef} : |\mathsf{Fun}(\mathcal{C}_\mathcal{S}, \mathsf{Set}_\mathcal{S})| \to |\mathsf{Fun}(\mathcal{C}_\mathcal{S}, \mathsf{Set}_\mathcal{S})|$:

$$\mathsf{IORef} : |\mathsf{Fun}(\mathcal{C}_\mathcal{S}, \mathsf{Set}_\mathcal{S})| \to |\mathsf{Fun}(\mathcal{C}_\mathcal{S}, \mathsf{Set}_\mathcal{S})|$$
$$\mathsf{IORef}\, A\, U :\equiv \textstyle\sum_{\ell:|U|} \blacktriangleright[X \leftarrow \partial_U \ell].\, \lceil X \rceil_U = \lceil A \rceil_U$$

Above, we have used the $\lceil - \rceil$ operator from Construction 3.4. We define the functorial action of $\mathsf{IORef}\, A$ on $f : U \to V$ by path induction on the identification $\partial_f : \partial_V \circ |f| = \partial_U$:

$$\mathsf{IORef}\, A\, (|f|, \mathsf{refl})\, (\ell, \phi) :\equiv (|f|\ell, \mathsf{next}[X \leftarrow \partial_V(|f|\ell), \psi \leftarrow \phi].\, \mathsf{ap}_{\lfloor \mathsf{Set}_\mathcal{S} \rfloor f} \psi)$$

Proof. That the identification $\lceil X \rceil_U = \lceil A \rceil_U$ is $\mathcal{S}$-small follows from Lemma 3.12, using the fact that $U/\mathcal{C}_\mathcal{S}$ is $\mathcal{U}$-small because $\mathcal{S}$ is assumed $\mathcal{U}$-small. ◀

3.2.2 A strong monad for general store

Rather than constructing the monad for general store all at once by hand, we take a more bite-sized approach by decomposing it into a simpler call-by-push-value adjunction following Levy [29]. In fact, we go quite a bit further than this and decompose the call-by-push-value adjunction itself into three separate and simpler adjunctions; the advantage of our decomposition is that it reveals the simple and elegant source of the admittedly complex explicit constructions of *op. cit.* All these adjunctions will be suitably *enriched* so as to give rise to a strong monad.[5] To get started, we will first require the concept of a **heaplet**, which is the valuation of a heap configuration at a particular world, assigning each specified location to an element of the prescribed semantic type at that world.

▶ **Construction 3.14** (The heaplet distributor)**.** Let $\mathcal{S}$ be guarded universe closed under finite products. We may define a distributor $\mathcal{H}_{\mathcal{S}} : \mathcal{C}_{\mathcal{S}}^{\mathsf{op}} \times \mathcal{C}_{\mathcal{S}} \to \mathsf{Set}_{\mathcal{S}}$ like so:

$$\mathcal{H}_{\mathcal{S}} : \mathcal{C}_{\mathcal{S}}^{\mathsf{op}} \times \mathcal{C}_{\mathcal{S}} \to \mathsf{Set}_{\mathcal{S}}$$
$$\mathcal{H}_{\mathcal{S}}\,(U, V) :\equiv \textstyle\prod_{\ell : |U|} \vartheta_{|\mathsf{Fun}(\mathcal{C}_{\mathcal{S}}, \mathsf{Set}_{\mathcal{S}})|}\,(\partial_U \ell)\,V$$

Then we will write $\widetilde{\mathcal{H}_{\mathcal{S}}}$ for the dependent sum $\sum_{U : |\mathcal{C}_{\mathcal{S}}|} \mathcal{H}_{\mathcal{S}}\,U\,U$ classifying **heaps**. We will write $\pi_{\mathcal{H}} : \widetilde{\mathcal{H}_{\mathcal{S}}} \to |\mathcal{C}_{\mathcal{S}}|$ for the first projection of a packed heap; given $H : \widetilde{\mathcal{H}_{\mathcal{S}}}$ and $\ell : |\pi_{\mathcal{H}} H|$, we will write $H \mathbin{@} \ell : \blacktriangleright[X \leftarrow \partial_{\pi_{\mathcal{H}} H} \ell] X(\pi_{\mathcal{H}} H)$ for the element stored by H at location ℓ.

Presheaf categories and unenriched adjunctions

Let $\mathcal{S}$ be a guarded universe closed under finite products. As $\widetilde{\mathcal{H}_{\mathcal{S}}}$ is a 1-type, we can equally well view it as a category whose hom sets are given by identity types, *i.e.* a groupoid. From this point of view, the projection $\pi_{\mathcal{H}} : \widetilde{\mathcal{H}_{\mathcal{S}}} \to |\mathcal{C}_{\mathcal{S}}|$ extends to functors $\pi_{\mathcal{H}} : \widetilde{\mathcal{H}_{\mathcal{S}}} \to \mathcal{C}_{\mathcal{S}}$ and $\bar{\pi}_{\mathcal{H}} : \widetilde{\mathcal{H}_{\mathcal{S}}} \cong \widetilde{\mathcal{H}_{\mathcal{S}}}^{\mathsf{op}} \to \mathcal{C}_{\mathcal{S}}^{\mathsf{op}}$. We will use these projections to construct a network of adjunctions between the following presheaf categories:

$$\mathcal{P}_{\mathcal{S}} :\equiv \mathsf{Fun}(\mathcal{C}_{\mathcal{S}}, \mathsf{Set}_{\mathcal{S}}) \qquad \mathcal{N}_{\mathcal{S}} :\equiv \mathsf{Fun}(\mathcal{C}_{\mathcal{S}}^{\mathsf{op}}, \mathsf{0Dom}_{\mathcal{S}})$$
$$\bar{\mathcal{P}}_{\mathcal{S}} :\equiv \mathsf{Fun}(\mathcal{C}_{\mathcal{S}}^{\mathsf{op}}, \mathsf{Set}_{\mathcal{S}}) \qquad \mathcal{Q}_{\mathcal{S}} :\equiv \mathsf{Fun}(\widetilde{\mathcal{H}_{\mathcal{S}}}, \mathsf{Set}_{\mathcal{S}})$$

▶ **Exegesis 3.15.** $\mathcal{P}_{\mathcal{S}}$ is the category on which our higher-order state monad is defined; this monad arises from a call-by-push-value adjunction [29] in which $\mathcal{P}_{\mathcal{S}}$ is the category of "value types and pure functions" and $\mathcal{N}_{\mathcal{S}}$ is the category of "computation types and stacks". A computation type differs from a value type in two ways, as it is both *contravariantly* indexed in worlds and valued in 0-domains rather than sets. We will treat these differences modularly by factoring the adjunction $\mathsf{F} \dashv \mathsf{U} : \mathcal{N}_{\mathcal{S}} \to \mathcal{P}_{\mathcal{S}}$ through further adjunctions. Our first adjoint resolution, to deal strictly with variance, is described in Lemma 3.16; later on in Lemma 3.19, we will lift the adjunction between sets and 0-domains to the world of presheaves.

▶ **Lemma 3.16.** *Let $\mathcal{S}$ be a small, set-reflective, guarded subuniverse of a guarded Martin-Löf universe $\mathcal{U}$ containing* Inj*. Then the unenriched base change functors $\Delta_{\pi_{\mathcal{H}}} : \mathcal{P}_{\mathcal{S}} \to \mathcal{Q}_{\mathcal{S}}$ and $\Delta_{\bar{\pi}_{\mathcal{H}}} : \bar{\mathcal{P}}_{\mathcal{S}} \to \mathcal{Q}_{\mathcal{S}}$ has left and right adjoints $\exists_{\bar{\pi}_{\mathcal{H}}} \dashv \Delta_{\bar{\pi}_{\mathcal{H}}}$ and $\Delta_{\pi_{\mathcal{H}}} \dashv \forall_{\pi_{\mathcal{H}}}$ respectively.*

Proof. As $\widetilde{\mathcal{H}_{\mathcal{S}}}$ is both discrete and $\mathcal{U}$-small, the Kan extensions exist because $\mathsf{Set}_{\mathcal{S}}$ has all $\mathcal{U}$-small coproducts and products as a reflective subuniverse of $\mathcal{U}$. ◀

[5] The purpose of strength, as ever, to transform the global Kleisli extension operation of the monad into a *binding*-operation that applies in arbitrary contexts.

The unenriched adjunctions of Lemma 3.16 can be computed on objects as follows, where $\bigcirc : \mathcal{U} \to \mathsf{Set}_{\mathcal{S}}$ is the assumed reflection:

$$\exists_{\bar{\pi}_{\mathcal{H}}} A\, U = \bigcirc \sum_{H : \widetilde{\mathcal{H}_{\mathcal{S}}}} \sum_{f : \hom_{\mathcal{C}_{\mathcal{S}}^{\mathsf{op}}}(\pi_{\mathcal{H}} H, U)} AH$$
$$\forall_{\pi_{\mathcal{H}}} A\, U = \prod_{H : \widetilde{\mathcal{H}_{\mathcal{S}}}} \prod_{f : \hom_{\mathcal{C}_{\mathcal{S}}}(U, \pi_{\mathcal{H}} H)} AH$$

We draw attention to the fact that codomain of $\exists_{\bar{\pi}_{\mathcal{H}}}$ is $\bar{\mathcal{P}}_{\mathcal{S}}$ while the codomain of $\forall_{\bar{\pi}_{\mathcal{H}}}$ is $\mathcal{P}_{\mathcal{S}}$, hence the appearance of $\hom_{\mathcal{C}_{\mathcal{S}}^{\mathsf{op}}}$ in the former and $\hom_{\mathcal{C}_{\mathcal{S}}}$ in the latter.

Enrichments and enriched adjunctions

Let $\mathcal{S}$ be a guarded universe closed under finite products. We now impose a common enrichment on $\mathcal{P}_{\mathcal{S}}, \bar{\mathcal{P}}_{\mathcal{S}}, \mathcal{N}_{\mathcal{S}}, \mathcal{Q}_{\mathcal{S}}$ so as to lift Lemma 3.16 to the enriched level, making all these categories *locally indexed* in $\mathcal{P}_{\mathcal{S}}$ in the sense of [29]. Given a co-presheaf $\Gamma \in \mathcal{P}_{\mathcal{S}}$, we will write $\pi_\Gamma : \tilde{\Gamma} \to \mathcal{C}_{\mathcal{S}}^{\mathsf{op}}$ for the discrete cartesian fibration corresponding to Γ. With this in hand, we impose the following additional notations:

$$\mathcal{P}_{\mathcal{S}}^{\Gamma} :\equiv \mathsf{Fun}(\tilde{\Gamma}^{\mathsf{op}}, \mathsf{Set}_{\mathcal{S}}) \qquad \mathcal{N}_{\mathcal{S}}^{\Gamma} :\equiv \mathsf{Fun}(\tilde{\Gamma}, \mathsf{0Dom}_{\mathcal{S}})$$
$$\bar{\mathcal{P}}_{\mathcal{S}}^{\Gamma} :\equiv \mathsf{Fun}(\tilde{\Gamma}, \mathsf{Set}_{\mathcal{S}}) \qquad \mathcal{Q}_{\mathcal{S}}^{\Gamma} :\equiv \mathsf{Fun}(\tilde{\Gamma}^{\mathsf{op}} \times_{\mathcal{C}_{\mathcal{S}}} \widetilde{\mathcal{H}_{\mathcal{S}}}, \mathsf{Set}_{\mathcal{S}})$$

Above, we note that $\mathcal{P}_{\mathcal{S}}^{\Gamma}$ is equivalent to the slice $\mathcal{P}_{\mathcal{S}}/\Gamma$; in the definition of $\mathcal{Q}_{\mathcal{S}}^{\Gamma}$, the expression $\tilde{\Gamma}^{\mathsf{op}} \times_{\mathcal{C}_{\mathcal{S}}} \widetilde{\mathcal{H}_{\mathcal{S}}}$ refers to the pullback of the span $\{\tilde{\Gamma}^{\mathsf{op}} \xrightarrow{\pi_\Gamma^{\mathsf{op}}} \mathcal{C}_{\mathcal{S}} \xleftarrow{\pi_{\mathcal{H}}} \widetilde{\mathcal{H}_{\mathcal{S}}}\}$. We have the following base change functors for any co-presheaf $\Gamma \in \mathcal{P}_{\mathcal{S}}$:

$$\Delta_\Gamma : \mathcal{P}_{\mathcal{S}} \to \mathcal{P}_{\mathcal{S}}^{\Gamma} \qquad \Delta_\Gamma : \mathcal{N}_{\mathcal{S}} \to \mathcal{N}_{\mathcal{S}}^{\Gamma}$$
$$\Delta_\Gamma A\, (U, \gamma) :\equiv AU \qquad \Delta_\Gamma\, X\, (U, \Gamma) :\equiv XU$$

$$\Delta_\Gamma : \bar{\mathcal{P}}_{\mathcal{S}} \to \bar{\mathcal{P}}_{\mathcal{S}}^{\Gamma} \qquad \Delta_\Gamma : \mathcal{Q}_{\mathcal{S}} \to \mathcal{Q}_{\mathcal{S}}^{\Gamma}$$
$$\Delta_\Gamma X\, (U, \gamma) :\equiv XU \qquad \Delta_\Gamma\, A\, (U, \Gamma, H) :\equiv A(U, H)$$

▶ **Construction 3.17** (Enrichments). We extend $\mathcal{P}_{\mathcal{S}}$, $\bar{\mathcal{P}}_{\mathcal{S}}$, $\mathcal{N}_{\mathcal{S}}$, and $\mathcal{Q}_{\mathcal{S}}$ to $\hat{\mathcal{P}}_{\mathcal{S}}$-enriched categories $\underline{\mathcal{P}_{\mathcal{S}}}$, $\underline{\bar{\mathcal{P}}_{\mathcal{S}}}$, $\underline{\mathcal{N}_{\mathcal{S}}}$, and $\underline{\mathcal{Q}_{\mathcal{S}}}$ respectively, regarding $\hat{\mathcal{P}}_{\mathcal{S}}$ with its *cartesian* monoidal structure:

$$\hom_{\underline{\mathcal{P}_{\mathcal{S}}}}(A, B)_\Gamma :\equiv \hom_{\mathcal{P}_{\mathcal{S}}^{\Gamma}}(\Delta_\Gamma A, \Delta_\Gamma B) \qquad \hom_{\underline{\mathcal{N}_{\mathcal{S}}}}(X, Y)_\Gamma :\equiv \hom_{\mathcal{N}_{\mathcal{S}}^{\Gamma}}(\Delta_\Gamma X, \Delta_\Gamma Y)$$
$$\hom_{\underline{\bar{\mathcal{P}}_{\mathcal{S}}}}(X, Y)_\Gamma :\equiv \hom_{\bar{\mathcal{P}}_{\mathcal{S}}^{\Gamma}}(\Delta_\Gamma X, \Delta_\Gamma Y) \qquad \hom_{\underline{\mathcal{Q}_{\mathcal{S}}}}(A, B)_\Gamma :\equiv \hom_{\mathcal{Q}_{\mathcal{S}}^{\Gamma}}(\Delta_\Gamma A, \Delta_\Gamma B)$$

These enrichments agree with those given by Levy [29] in terms of dinatural transformations as one can see using the formula for a natural transformation as an end. The purpose of imposing these enrichments was to be able to state Lemmas 3.18 and 3.19 below.

▶ **Lemma 3.18.** *Under the assumptions of Lemma 3.16, the unenriched adjunctions* $\Delta_{\pi_{\mathcal{H}}} \dashv \forall_{\pi_{\mathcal{H}}}$ *and* $\exists_{\bar{\pi}_{\mathcal{H}}} \dashv \Delta_{\bar{\pi}_{\mathcal{H}}}$ *extend to* $\hat{\mathcal{P}}_{\mathcal{S}}$*-enriched adjunctions* $\underline{\Delta}_{\pi_{\mathcal{H}}} \dashv \underline{\forall}_{\pi_{\mathcal{H}}}$ *and* $\underline{\exists}_{\bar{\pi}_{\mathcal{H}}} \dashv \underline{\Delta}_{\bar{\pi}_{\mathcal{H}}}$.

▶ **Lemma 3.19.** *Let* $\mathcal{S}$ *be a guarded universe closed under binary coproducts. Then the adjunction* $\mathsf{L} \dashv \mathsf{R} : \mathsf{0Dom}_{\mathcal{S}} \to \mathsf{Set}_{\mathcal{S}}$ *between sets and guarded 0-domains (Lemma 3.6) can be lifted pointwise to an enriched adjunction* $\mathsf{L} \dashv \mathsf{R} : \underline{\mathcal{N}_{\mathcal{S}}} \to \underline{\bar{\mathcal{P}}_{\mathcal{S}}}$.

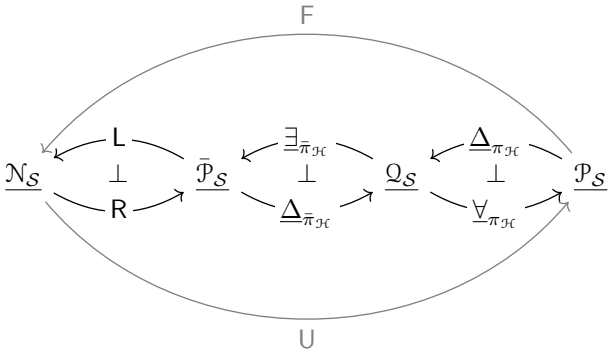


Figure 4 A diagram of $\hat{\mathcal{P}}_{\mathcal{S}}$-enriched adjunctions, together comprising a call-by-push-value adjunction $\mathsf{F} \dashv \mathsf{U} : \underline{\mathcal{N}_{\mathcal{S}}} \to \underline{\mathcal{P}_{\mathcal{S}}}$ resolving an enriched (and thus strong) monad $\mathsf{IO} = \mathsf{U} \circ \mathsf{F}$ on $\underline{\mathcal{P}_{\mathcal{S}}}$.

The call-by-push-value adjunction and resulting strong monad

Let $\mathcal{S}$ be a small, set-reflective, guarded subuniverse of a guarded Martin-Löf universe $\mathcal{U}$ containing Inj. We can compose the enriched adjunctions obtained in Lemma 3.18 to obtain a single enriched adjunction $\mathsf{F} \dashv \mathsf{U} : \underline{\mathcal{N}_{\mathcal{S}}} \to \underline{\mathcal{P}_{\mathcal{S}}}$, setting $\mathsf{F} :\equiv \mathsf{L} \circ \underline{\exists}_{\bar{\pi}_{\mathcal{H}}} \circ \underline{\Delta}_{\pi_{\mathcal{H}}}$ and $\mathsf{U} :\equiv \underline{\forall}_{\pi_{\mathcal{H}}} \circ \underline{\Delta}_{\bar{\pi}_{\mathcal{H}}} \circ \mathsf{R}$ as depicted in Figure 4. We will write ret for the unit of this adjunction. Our adjunction is an adjoint decomposition of Levy's possible worlds model of general storage [29], with Levy's syntactic Kripke worlds replaced by **recursively defined univalent semantic worlds**, as can be seen from Computation 3.20 below.

▶ **Computation 3.20** (Description of adjoints and monad)**.** For the sake of concreteness, we compute the action of the left and right adjoints on objects as follows:

$$\mathsf{F}A\,U = \mathsf{L} \bigcirc \textstyle\sum_{H : \widetilde{\mathcal{H}_{\mathcal{S}}}} \sum_{f : \hom_{\mathcal{S}}(U, \pi_{\mathcal{H}} H)} A(\pi_{\mathcal{H}} H)$$
$$\mathsf{U}X\,U = \textstyle\prod_{H : \widetilde{\mathcal{H}_{\mathcal{S}}}} \prod_{f : \hom_{\mathcal{S}}(U, \pi_{\mathcal{H}} H)} \mathsf{R}(X(\pi_{\mathcal{H}} H))$$

Composing the above, we describe the action of the monad $\mathsf{IO} = \mathsf{U} \circ \mathsf{F}$ on objects:

$$\mathsf{IO}\,A\,U = \textstyle\prod_{H : \widetilde{\mathcal{H}_{\mathcal{S}}}} \prod_{f : \hom_{\mathcal{S}}(U, \pi_{\mathcal{H}} H)} \mathsf{RL} \bigcirc \sum_{H' : \widetilde{\mathcal{H}_{\mathcal{S}}}} \sum_{f : \hom_{\mathcal{S}}(\pi_{\mathcal{H}} H, \pi_{\mathcal{H}} H')} A(\pi_{\mathcal{H}} H')$$

▶ **Lemma 3.21.** *Under the assumptions of Lemma 3.8, each* $\mathsf{U}X$ *is a guarded domain.*

3.2.3 The model of univalent reference types

We have now defined all the basic elements of our model.

3.2.3.1 General recursion and stepping

Let $\mathcal{S}$ be a small, set-reflective, guarded subuniverse of a guarded Martin-Löf universe $\mathcal{U}$ containing Inj. Abstract steps in our model are encoded in terms of a global element $\mathsf{step} : \mathsf{IO}\,\mathbf{1}$, defined using the guarded domain structure of $\mathsf{IO}\,\mathbf{1}$ as $\mathsf{step} :\equiv \vartheta_{\mathsf{IO}\,\mathbf{1}}(\mathsf{next}\,(\mathsf{ret}\,*))$.

▶ **Lemma 3.22.** *For any* $u : \mathsf{IO}\,A$*, we have* $\mathsf{step}; u = \vartheta_{\mathsf{IO}\,A}(\mathsf{next}\,u)$.

Proof. By unfolding the definition of the ▶-algebra structure pointwise in the model. ◀

$$
\begin{aligned}
&\mathsf{get} : \mathsf{IORef}\ A \to \mathsf{IO}\ A \\
&\mathsf{get}_U(\ell : |U|, \phi : \blacktriangleright[X \leftarrow \partial_U \ell].\lceil X \rceil_U = \lceil A \rceil_U)\ H\ (f \equiv (|f|, \mathsf{refl})) :\equiv \\
&\quad \vartheta_{\mathsf{F}\,A\,(\pi_{\mathcal{H}}\,H)}\mathsf{next}[X \leftarrow \partial_U \ell, \psi \leftarrow \phi, x \leftarrow H \mathbin{@} |f|\,\ell]. \\
&\quad \eta^{\bigcirc}(H, 1_{\pi_{\mathcal{H}}\,H}, \psi_* x)
\end{aligned}
$$

$$
\begin{aligned}
&\mathsf{set} : \mathsf{IORef}\ A \times A \to \mathsf{IO}\ \mathbf{1} \\
&\mathsf{set}_U((\ell, \phi), a)\ H\ (f \equiv (|f|, \mathsf{refl})) :\equiv \\
&\quad \mathbf{let}\ H_{a/\ell} :\equiv H[|f|\,\ell \hookrightarrow \mathsf{next}[X \leftarrow \partial_U \ell, \psi \leftarrow \phi].\psi_*^{-1}(A f\ a)]\ \mathbf{in} \\
&\quad \mathsf{now}\ (\eta^{\bigcirc}(H_{a/\ell}, 1_{\pi_{\mathcal{H}}\,H}, *))
\end{aligned}
$$

$$
\begin{aligned}
&\mathsf{alloc} : A \to \mathsf{IO}\ (\mathsf{IORef}\ A) \\
&\mathsf{alloc}_U\ a\ H\ (f \equiv (|f|, \mathsf{refl})) :\equiv \\
&\quad \mathbf{let}\ H_a :\equiv \begin{cases} \mathsf{inl}\,\ell \hookrightarrow \mathsf{next}[X \leftarrow \partial_{\pi_{\mathcal{H}}\,H}\ell, x \leftarrow H \mathbin{@} \ell].\ X\ \mathsf{inl}\ x \\ \mathsf{inr}\,* \hookrightarrow \mathsf{next}\ (A\ \mathsf{inr}\ a) \end{cases} \quad \mathbf{in} \\
&\quad \mathsf{now}\ (\eta^{\bigcirc}(H_a, \mathsf{inl}, (\mathsf{inr}\,*, \mathsf{next}\ \mathsf{refl})))
\end{aligned}
$$

Figure 5 Summary of the store operations in $\mathcal{P}_{\mathcal{S}}$ when $\mathcal{S}$ is a small, Σ-closed, set-reflective, guarded subuniverse of a guarded Martin-Löf universe $\mathcal{U}$ containing Inj.

Using $\mathsf{fix}_{\blacktriangleright}$, we can define a monadic fixed point combinator satisfying the equation $\mathsf{rec}\,h\,a = \mathsf{step}; h\,(\mathsf{rec}\,h)\,a$.

$$
\begin{aligned}
&\mathsf{rec} : ((A \to \mathsf{IO}\ B) \to A \to \mathsf{IO}\ B) \to A \to \mathsf{IO}\ B \\
&\mathsf{rec}\,h :\equiv \mathsf{fix}_{\blacktriangleright}\ (\lambda f.\,\lambda x.\,\vartheta_{\mathsf{IO}\ B}(\mathsf{next}[g \leftarrow f].h\,g\,x))
\end{aligned}
$$

▶ **Lemma 3.23.** *We have* $\mathsf{rec}\,h\,a = \mathsf{step}; h\,(\mathsf{rec}\,h)\,a$.

Proof. We compute as follows:

$\mathsf{rec}\,h\,a$

by unfolding definitions
$\equiv \mathsf{fix}_{\blacktriangleright}\ (\lambda f.\,\lambda x.\,\vartheta_{\mathsf{IO}\ B}(\mathsf{next}[g \leftarrow f].h\,g\,x))\,a$
by $\mathsf{fix}_{\blacktriangleright}$ computation rule
$\equiv \vartheta_{\mathsf{IO}\ B}(\mathsf{next}[g \leftarrow \mathsf{next}\ (\mathsf{rec}\,h)].h\,g\,a)$
by rules of delayed substitutions
$\equiv \vartheta_{\mathsf{IO}\ B}(\mathsf{next}\ (h\ (\mathsf{rec}\,h)\ a))$
by Lemma 3.22
$= \mathsf{step}; h\,(\mathsf{rec}\,h)\,a$

We are done. ◀

3.2.3.2 Store operations: getting, setting, and allocation

Let $\mathcal{S}$ be a small, Σ-closed, set-reflective, guarded subuniverse of a guarded Martin-Löf universe $\mathcal{U}$ containing Inj. In this section, we explicitly construct the store operations in $\mathcal{P}_{\mathcal{S}}$, which we summarize in Figure 5.

▶ **Construction 3.24** (Detailed construction of the getter)**.** For any $A \in \mathcal{P}_{\mathcal{S}}$, we can interpret the getter as a natural transformation $\mathsf{get} : \mathsf{IORef}\ A \to \mathsf{IO}\ A$ in $\mathcal{P}_{\mathcal{S}}$. Because the definition is a little subtle, we will do it step-by-step.

$$\mathsf{get} : \mathsf{IORef}\ A \to \mathsf{IO}\ A$$
$$\mathsf{get}_U(\ell : |U|, \phi : \blacktriangleright[X \leftarrow \partial_U \ell].\lceil X \rceil_U = \lceil A \rceil_U)\ H\ f :\equiv \ ?_{} : \mathsf{F}A\,(\pi_{\mathcal{H}} H)$$

We proceed by based path induction on the singleton $(\partial_U, \partial_f : \partial_{\pi_{\mathcal{H}} H} \circ |f| = \partial_U)$, setting $U :\equiv (|U|, \partial_{\pi_{\mathcal{H}} H} \circ |f|)$ and $f :\equiv (|f|, \mathsf{refl})$:

$$\mathsf{get}_U(\ell : |U|, \phi : \blacktriangleright[X \leftarrow \partial_U \ell].\lceil X \rceil_U = \lceil A \rceil_U)\ V\ (f \equiv (|f|, \mathsf{refl})) :\equiv \ ?_{} : \mathsf{F}A\,(\pi_{\mathcal{H}} H)$$

Next, we use the guarded domain structure of the goal:

$$\mathsf{get}_U(\ell : |U|, \phi : \blacktriangleright[X \leftarrow \partial_U \ell].\lceil X \rceil_U = \lceil A \rceil_U)\ H\ (f \equiv (|f|, \mathsf{refl})) :\equiv$$
$$\vartheta_{\mathsf{F}A\,(\pi_{\mathcal{H}} H)}\ ?_{} : \blacktriangleright \mathsf{F}A\,(\pi_{\mathcal{H}} H)$$

Using the introduction rule for the later modality, we may unwrap the delayed identification ϕ to assume $\psi : \lceil X \rceil_U = \lceil A \rceil_U$, as well as the delayed element $H\,@\,|f|\,\ell$ to assume $x : X(\pi_{\mathcal{H}} H)$:

$$\mathsf{get}_U(\ell : |U|, \phi : \blacktriangleright[X \leftarrow \partial_U \ell].\lceil X \rceil_U = \lceil A \rceil_U)\ H\ (f \equiv (|f|, \mathsf{refl})) :\equiv$$
$$\vartheta_{\mathsf{F}A\,(\pi_{\mathcal{H}} H)}\mathsf{next}[X \leftarrow \partial_U \ell, \psi \leftarrow \phi, x \leftarrow H\ @\ |f|\,\ell].\ ?_{} : \mathsf{F}A\,(\pi_{\mathcal{H}} H)$$

Applying the unit of the reflection $\bigcirc : \mathcal{U} \to \mathsf{Set}_{\mathcal{S}}$ and splitting the resulting goal, we have three holes:

$$\mathsf{get}_U(\ell : |U|, \phi : \blacktriangleright[X \leftarrow \partial_U \ell].\lceil X \rceil_U = \lceil A \rceil_U)\ H\ (f \equiv (|f|, \mathsf{refl})) :\equiv$$
$$\vartheta_{\mathsf{F}A\,(\pi_{\mathcal{H}} H)}\mathsf{next}[X \leftarrow \partial_U \ell, \psi \leftarrow \phi, x \leftarrow H\ @\ |f|\,\ell].$$
$$\eta^{\bigcirc}(\ ?_0 : \widetilde{\mathcal{H}_{\mathcal{S}}}\ ,\ ?_1 : \hom_{\mathcal{C}_{\mathcal{S}}}(\pi_{\mathcal{H}} H, \pi_{\mathcal{H}} ?_0)\ ,\ ?_2 : A(\pi_{\mathcal{H}} ?_0)\ )$$

A read-operation does not change the heap; therefore, we fill in the first hole with the existing heap H and the second hole with the identity map.

$$\mathsf{get}_U(\ell : |U|, \phi : \blacktriangleright[X \leftarrow \partial_U \ell].\lceil X \rceil_U = \lceil A \rceil_U)\ H\ (f \equiv (|f|, \mathsf{refl})) :\equiv$$
$$\vartheta_{\mathsf{F}A\,(\pi_{\mathcal{H}} H)}\mathsf{next}[X \leftarrow \partial_U \ell, \psi \leftarrow \phi, x \leftarrow H\ @\ |f|\,\ell].$$
$$\eta^{\bigcirc}(H, 1_{\pi_{\mathcal{H}} H},\ ?_{} : A(\pi_{\mathcal{H}} H)\ )$$

Recall that we have an identification $\psi : \lceil X \rceil_U = \lceil A \rceil_U$ in the type $\lfloor \mathsf{Set}_{\mathcal{S}} \rfloor_U$ of copresheaves on $U/\mathcal{C}_{\mathcal{S}}$; transporting by this identification in the family $Z : \lfloor \mathsf{Set}_{\mathcal{S}} \rfloor_U \vdash Z\,(\pi_{\mathcal{H}} H)\,f : \mathsf{Set}_{\mathcal{S}}$, we have a mapping from $\lceil X \rceil_U\,(\pi_{\mathcal{H}} H)\,f \equiv X(\pi_{\mathcal{H}} H)$ to $\lceil A \rceil_U\,(\pi_{\mathcal{H}} H)\,f \equiv A(\pi_{\mathcal{H}} H)$, which we use to fill the final hole:

$$\mathsf{get}_U(\ell : |U|, \phi : \blacktriangleright[X \leftarrow \partial_U \ell].\lceil X \rceil_U = \lceil A \rceil_U)\ H\ (f \equiv (|f|, \mathsf{refl})) :\equiv$$
$$\vartheta_{\mathsf{F}A\,(\pi_{\mathcal{H}} H)}\mathsf{next}[X \leftarrow \partial_U \ell, \psi \leftarrow \phi, x \leftarrow H\ @\ |f|\,\ell].$$
$$\eta^{\bigcirc}(H, 1_{\pi_{\mathcal{H}} H}, \psi_* x)$$

This completes the definition of the getter.

▶ **Construction 3.25** (Detailed construction of the setter)**.** For each $A \in \mathcal{P}_{\mathcal{S}}$, we define the setter as a natural transformation $\mathsf{IORef}\ A \times A \to \mathsf{IO}\ \mathbf{1}$ in $\mathcal{P}_{\mathcal{S}}$.

$$\mathsf{set} : \mathsf{IORef}\ A \times A \to \mathsf{IO}\ \mathbf{1}$$
$$\mathsf{set}_U((\ell, \phi), a)\ H\ (f \equiv (|f|, \mathsf{refl})) :\equiv \ ?_{} : \mathsf{F}\mathbf{1}\,(\pi_{\mathcal{H}} H)$$

We apply the unit now of the lifting monad, followed by the unit of the reflection $\bigcirc : \mathcal{U} \to \mathsf{Set}_{\mathcal{S}}$, and then split the goal:

$$\begin{aligned}
&\mathsf{set} : \mathsf{IORef}\, A \times A \to \mathsf{IO}\, \mathbf{1} \\
&\mathsf{set}_U((\ell, \phi), a)\, H\, (f \equiv (|f|, \mathsf{refl})) :\equiv \\
&\quad \mathsf{now}\, (\eta^{\bigcirc}(\ ?_0 : \widetilde{\mathcal{H}_{\mathcal{S}}}\ ,\ ?_1 : \mathrm{hom}_{\mathcal{C}_{\mathcal{S}}}(\pi_{\mathcal{H}} H, \pi_{\mathcal{H}} ?_0)\ , *))
\end{aligned}$$

We want to replace the contents of H at the location $|f|\,\ell$ with the reindexed element $\mathsf{next}(A f\, a) : \blacktriangleright A(\pi_{\mathcal{H}} H)$; for this to make sense, we need to transport along the (delayed) identification $\phi : \blacktriangleright[X \leftarrow \partial_U \ell].\lceil X \rceil_U = \lceil A \rceil_U$. We define the updated heap as follows, noting that $\partial_U \ell \equiv \partial_{\pi_{\mathcal{H}} H} |f|\, \ell$:

$$?_0 :\equiv H[|f|\,\ell \hookrightarrow \mathsf{next}[X \leftarrow \partial_U \ell, \psi \leftarrow \phi].\psi_*^{-1}(A f\, a)]$$

The updated heap can be so-defined because its set of locations is finite, and thus has decidable equality. Because the updated heap has the same underlying configuration, we can fill our remaining hole $?_1 :\equiv 1_{\pi_{\mathcal{H}} H}$, completing the definition of the setter as follows:

$$\begin{aligned}
&\mathsf{set} : \mathsf{IORef}\, A \times A \to \mathsf{IO}\, \mathbf{1} \\
&\mathsf{set}_U((\ell, \phi), a)\, H\, (f \equiv (|f|, \mathsf{refl})) :\equiv \\
&\quad \textbf{let}\ H_{a/\ell} :\equiv H[|f|\,\ell \hookrightarrow \mathsf{next}[X \leftarrow \partial_U \ell, \psi \leftarrow \phi].\psi_*^{-1}(A f\, a)]\ \textbf{in} \\
&\quad \mathsf{now}\, (\eta^{\bigcirc}(H_{a/\ell}, 1_{\pi_{\mathcal{H}} H}, *))
\end{aligned}$$

▶ **Construction 3.26** (The allocator)**.** For each $A \in \mathcal{P}_{\mathcal{S}}$, we define the allocator as a natural transformation $A \to \mathsf{IO}\,(\mathsf{IORef}\, A)$ in $\mathcal{P}_{\mathcal{S}}$.

$$\begin{aligned}
&\mathsf{alloc} : A \to \mathsf{IO}\,(\mathsf{IORef}\, A) \\
&\mathsf{alloc}_U\, a\, H\, (f \equiv (|f|, \mathsf{refl})) :\equiv \\
&\quad \textbf{let}\ H_a :\equiv \begin{cases} \mathsf{inl}\, \ell \hookrightarrow \mathsf{next}[X \leftarrow \partial_{\pi_{\mathcal{H}} H} \ell, x \leftarrow H \mathbin{@} \ell].\, X\, \mathsf{inl}\, x \\ \mathsf{inr}\, * \hookrightarrow \mathsf{next}\, (A\, \mathsf{inr}\, a) \end{cases} \quad \textbf{in} \\
&\quad \mathsf{now}\, (\eta^{\bigcirc}(H_a, \mathsf{inl}, (\mathsf{inr}\, *, \mathsf{next}\, \mathsf{refl})))
\end{aligned}$$

Above, we have defined a new heap H_a whose underlying finite set of locations is the coproduct $|\pi_{\mathcal{H}} H| + \mathbf{1}$, filling the new location with a and return the pointer to this location.

3.2.3.3 The main theorem

We now come to the main result of this paper, which obtains a model of univalent reference types from a suitably structured small set-reflective subuniverse.

▶ **Theorem 3.27.** *Let $\mathcal{S}$ be a small, Σ-closed, set-reflective, guarded subuniverse of a guarded Martin-Löf universe $\mathcal{U}$ containing* Inj *such that $\mathcal{S}$ is additionally closed under the type of natural numbers. Then there is a model of the monadic language from Section 2 satisfying the equational theory of univalent reference types (Figures 1 and 2), in which:*

1. *contexts, types, and terms are interpreted in the category $\mathcal{P}_{\mathcal{S}} = \mathsf{Fun}(\mathcal{C}_{\mathcal{S}}, \mathsf{Set}_{\mathcal{S}})$;*
2. *the reference type connective is interpreted as in Construction 3.13;*
3. *the computational monad is given by $\mathsf{IO} = \mathsf{U} \circ \mathsf{F}$ as defined in Figure 4;*
4. *general recursion and the store operations are interpreted as in Sections 3.2.3.1 and 3.2.3.2.*

Proof. We note that $\mathcal{P}_{\mathcal{S}} = \mathsf{Fun}(\mathcal{C}_{\mathcal{S}}, \mathsf{Set}_{\mathcal{S}})$ is locally cartesian closed in spite of the fact that $\mathcal{C}_{\mathcal{S}}$ is as large as $\mathsf{Set}_{\mathcal{S}}$ is: local cartesian closure nonetheless follows because $\mathsf{Set}_{\mathcal{S}}$ is reflective in $\mathcal{U}$ and $\mathcal{C}_{\mathcal{S}}$ is $\mathcal{U}$-small. Everything except the two laws of *univalent* reference types (Figure 2) follows in the same way as in the non-univalent model given by Sterling et al. [43]. The ALLOCATION PERMUTATION law holds under the interpretations given because the two heaps resulting from allocations in different orders are identified under univalence. The REPRESENTATION INDEPENDENCE law holds for similar reasons, considering the effect of *transporting* along an identification between equivalent heaps on the getter and the setter. ◀

4 Models of guarded HoTT with impredicative universes

Our main result (Theorem 3.27) is contingent on there existing a model of guarded homotopy type theory in which there can be found a suitably small, Σ-closed, set-reflective, guarded subuniverse of a guarded Martin-Löf universe containing Inj. It is by no means obvious that such a model exists, but in this section we will provide some preliminary evidence.

1. Sterling, Gratzer, and Birkedal [43] have constructed models of **impredicative guarded dependent type theory** (**iGDTT**), a non-univalent version of our metalanguage.
2. Awodey [11] has constructed a model of impredicative homotopy type theory in **cubical assemblies**, *i.e.* internal cubical sets in the category of assemblies. Uemura [45] subsequently described a variant of this model in the style of Orton and Pitts [34].
3. Birkedal et al. [15, 14] have constructed an Orton–Pitts model of **guarded cubical type theory** in presheaves on the product of a cube category with the ordinal ω. This model was revisited in the context of multi-modal type theory by Aagaard et al. [1].

The methods of the papers above are essentially modular, and are furthermore not particularly sensitive to the choice of cube category or ordinal, so long as these can be defined in assemblies without resorting to quotients.

▶ **Conjecture 4.1** (Soundness). *There is a non-trivial model of guarded homotopy type theory in guarded cubical assemblies in which there is a small, set-reflective, guarded Martin-Löf subuniverse $\mathcal{S} \subseteq \mathcal{U}$ of a guarded Martin-Löf universe $\mathcal{U}$ containing* Inj.

5 Conclusions and future work

We have demonstrated the impact of a univalent metalanguage on the denotational semantics of higher-order store, extending the guarded global allocation model of Sterling et al. [43] with new program equivalences: invariance under permutation and representation independence in the heap. We believe that we have only scratched the surface of the potential for univalent denotational semantics in general, and univalent reference types in particular; we describe a few potential areas for further development beyond substantiating Conjecture 4.1.

1. Sterling et al. [43] have given non-univalent denotational semantics of *polymorphic* λ-calculus with recursive types and general reference types. It is within reach to adapt this model to the univalent setting, obtaining even more program equivalences than before. In particular, many data abstraction theorems for existential packages that typically hold only up to observational equivalence are expected to hold on the nose.

2. Our case study, an equation between two object-oriented counters, involves invariance of the heap under *isomorphisms* between data representations – whereas parametricity is often employed in cases of correspondences that are not isomorphisms. Angiuli et al. [7] have shown that many such applications of parametricity are nonetheless subsumed by univalence in the presence of quotient types, and thus many more observational equivalences can be replaced with honest equations in univalent denotational semantics. We are eager to put the wisdom of *op. cit.* into practice in the context of imperative and object-oriented programming by incorporating quotients into our theory and model.
3. Although our theory validates many more desirable equations than the global store theory of Sterling et al. [43], we do not come close to modeling full local store: for example, two programs that allocate different numbers of cells cannot be equal. We hope that it will be possible to adapt the methods of Kammar et al. [27] to the guarded, univalent, and impredicative setting in order to develop even more abstract models of mutable state.
4. Our language does not allow for references to be directly compared (*nominal references*) and no such equality testing function exists in our model. Prior work [44, 33] has given models of such references using the theory of nominal sets [23, 37]. We hope that these methods may be adapted to our model in order to support nominal univalent references.

References

1 Frederik Lerbjerg Aagaard, Magnus Baunsgaard Kristensen, Daniel Gratzer, and Lars Birkedal. Unifying cubical and multimodal type theory. Unpublished manuscript, 2022. doi:10.48550/arXiv.2203.13000.

2 S. Abramsky, K. Honda, and G. McCusker. A fully abstract game semantics for general references. In *Proceedings of the 13th Annual IEEE Symposium on Logic in Computer Science*, pages 334–344, USA, 1998. IEEE Computer Society. doi:10.1109/LICS.1998.705669.

3 Amal Jamil Ahmed. *Semantics of Types for Mutable State*. PhD thesis, Princeton University, 2004. URL: http://www.ccs.neu.edu/home/amal/ahmedthesis.pdf.

4 Benedikt Ahrens, Dan Frumin, Marco Maggesi, Niccolò Veltri, and Niels van der Weide. Bicategories in univalent foundations. *Mathematical Structures in Computer Science*, 31(10):1232–1269, 2021. doi:10.1017/S0960129522000032.

5 Benedikt Ahrens and Peter LeFanu Lumsdaine. Displayed categories. *Logical Methods in Computer Science*, 15, March 2019. doi:10.23638/LMCS-15(1:20)2019.

6 Pierre America and Jan J. M. M. Rutten. Solving reflexive domain equations in a category of complete metric spaces. In *Proceedings of the 3rd Workshop on Mathematical Foundations of Programming Language Semantics*, pages 254–288, Berlin, Heidelberg, 1987. Springer-Verlag.

7 Carlo Angiuli, Evan Cavallo, Anders Mörtberg, and Max Zeuner. Internalizing representation independence with univalence. *Proceedings of the ACM on Programming Languages*, 5(POPL):1–30, January 2021. doi:10.1145/3434293.

8 Andrew W. Appel and David McAllester. An indexed model of recursive types for foundational proof-carrying code. *ACM Transactions on Programming Languages and Systems*, 23(5):657–683, September 2001. doi:10.1145/504709.504712.

9 Andrew W. Appel, Paul-André Melliès, Christopher D. Richards, and Jérôme Vouillon. A very modal model of a modern, major, general type system. In *Proceedings of the 34th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 109–122, Nice, France, 2007. Association for Computing Machinery.

10 A. Arnold and M. Nivat. Metric interpretations of infinite trees and semantics of non deterministic recursive programs. *Theoretical Computer Science*, 11(2):181–205, 1980. doi:10.1016/0304-3975(80)90045-6.

11 Steve Awodey. Impredicative encodings in HoTT (or: Towards a realizability ∞-topos). Slides from a talk given the *Big Proof* meeting, Isaac Newton Institute, Cambridge. URL: https://www.andrew.cmu.edu/user/awodey/talks/BigProofs.pdf.

12 Steve Awodey. On Hofmann–Streicher universes. Unpublished manuscript, 2022. doi:10.48550/arXiv.2205.10917.

13 Steve Awodey, Jonas Frey, and Sam Speight. Impredicative encodings of (higher) inductive types. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 76–85, Oxford, United Kingdom, 2018. Association for Computing Machinery. doi:10.1145/3209108.3209130.

14 Lars Birkedal, Aleš Bizjak, Ranald Clouston, Hans Bugge Grathwohl, Bas Spitters, and Andrea Vezzosi. Guarded Cubical Type Theory: Path Equality for Guarded Recursion. In Jean-Marc Talbot and Laurent Regnier, editors, *25th EACSL Annual Conference on Computer Science Logic (CSL 2016)*, volume 62 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 23:1–23:17, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.CSL.2016.23.

15 Lars Birkedal, Aleš Bizjak, Ranald Clouston, Hans Bugge Grathwohl, Bas Spitters, and Andrea Vezzosi. Guarded cubical type theory. *Journal of Automated Reasoning*, 63(2):211–253, 2019. doi:10.1007/s10817-018-9471-7.

16 Lars Birkedal and Rasmus Ejlers Møgelberg. Intensional type theory with guarded recursive types qua fixed points on universes. In *Proceedings of the 2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 213–222, Washington, DC, USA, 2013. IEEE Computer Society. doi:10.1109/LICS.2013.27.

17 Lars Birkedal, Rasmus Ejlers Møgelberg, Jan Schwinghammer, and Kristian Støvring. First steps in synthetic guarded domain theory: Step-indexing in the topos of trees. In *Proceedings of the 2011 IEEE 26th Annual Symposium on Logic in Computer Science*, pages 55–64, Washington, DC, USA, 2011. IEEE Computer Society. doi:10.1109/LICS.2011.16.

18 Lars Birkedal, Kristian Støvring, and Jacob Thamsborg. Realisability semantics of parametric polymorphism, general references and recursive types. *Mathematical Structures in Computer Science*, 20(4):655–703, 2010. doi:10.1017/S0960129510000162.

19 Aleš Bizjak, Hans Bugge Grathwohl, Ranald Clouston, Rasmus Ejlers Møgelberg, and Lars Birkedal. Guarded dependent type theory with coinductive types. In Bart Jacobs and Christof Löding, editors, *Foundations of Software Science and Computation Structures: 19th International Conference, FOSSACS 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2–8, 2016, Proceedings*, pages 20–35, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg. doi:10.1007/978-3-662-49630-5_2.

20 Aleš Bizjak and Rasmus Ejlers Møgelberg. Denotational semantics for guarded dependent type theory. *Mathematical Structures in Computer Science*, 30(4):342–378, 2020. doi:10.1017/S0960129520000080.

21 Franck Breugel and Jeroen Warmerdam. Solving domain equations in a category of compact metric spaces. Technical report, CWI (Centre for Mathematics and Computer Science), NLD, 1994.

22 Titouan Carette, Louis Lemonnier, and Vladimir Zamdzhiev. Central submonads and notions of computation: Soundness, completeness and internal languages. In *2023 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–13, Los Alamitos, CA, USA, June 2023. IEEE Computer Society. doi:10.1109/LICS56636.2023.10175687.

23 Murdoch J. Gabbay and Andrew M. Pitts. A new approach to abstract syntax with variable binding. *Formal Aspects of Computing*, 13(3):341–363, July 2002. doi:10.1007/s001650200016.

24 Martin Hofmann and Thomas Streicher. Lifting Grothendieck universes. Unpublished note, 1997. URL: https://www2.mathematik.tu-darmstadt.de/~streicher/NOTES/lift.pdf.

25 J. M. E. Hyland. The effective topos. In A. S. Troelstra and D. Van Dalen, editors, *The L.E.J. Brouwer Centenary Symposium*, pages 165–216. North Holland Publishing Company, 1982.

26 J. M. E. Hyland, E. P. Robinson, and G. Rosolini. The Discrete Objects in the Effective Topos. *Proceedings of the London Mathematical Society*, s3-60(1):1–36, January 1990. doi:10.1112/plms/s3-60.1.1.

27 Ohad Kammar, Paul B. Levy, Sean K. Moss, and Sam Staton. A monad for full ground reference cells. In *Proceedings of the 32nd Annual ACM/IEEE Symposium on Logic in Computer Science*, Reykjavik, Iceland, June 2017. IEEE Press. doi:10.1109/LICS.2017.8005109.

28 Paul Blain Levy. Possible world semantics for general storage in call-by-value. In Julian Bradfield, editor, *Computer Science Logic*, pages 232–246, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

29 Paul Blain Levy. Adjunction models for call-by-push-value with stacks. *Electronic Notes in Theoretical Computer Science*, 69:248–271, 2003. CTCS'02, Category Theory and Computer Science. doi:10.1016/S1571-0661(04)80568-1.

30 Paul Blain Levy. *Call-by-Push-Value: A Functional/Imperative Synthesis*. Kluwer, Semantic Structures in Computation, 2, January 2003.

31 Rasmus Ejlers Møgelberg and Marco Paviotti. Denotational semantics of recursive types in synthetic guarded domain theory. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 317–326, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2933575.2934516.

32 Eugenio Moggi. Notions of computation and monads. *Information and Computation*, 93(1):55–92, 1991. Selections from 1989 IEEE Symposium on Logic in Computer Science. doi:10.1016/0890-5401(91)90052-4.

33 Andrzej Murawski and Nikos Tzevelekos. Nominal game semantics. *Foundations and Trends in Programming Languages*, 2(4):191–269, 2016. doi:10.1561/2500000017.

34 Ian Orton and Andrew M. Pitts. Axioms for modelling cubical type theory in a topos. In Jean-Marc Talbot and Laurent Regnier, editors, *25th EACSL Annual Conference on Computer Science Logic (CSL 2016)*, volume 62 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 24:1–24:19, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.CSL.2016.24.

35 Marco Paviotti. *Denotational semantics in Synthetic Guarded Domain Theory*. PhD thesis, IT-Universitetet i København, Denmark, 2016.

36 Marco Paviotti, Rasmus Ejlers Møgelberg, and Lars Birkedal. A model of PCF in Guarded Type Theory. *Electronic Notes in Theoretical Computer Science*, 319(Supplement C):333–349, 2015. The 31st Conference on the Mathematical Foundations of Programming Semantics (MFPS XXXI). doi:10.1016/j.entcs.2015.12.020.

37 Andrew M. Pitts. *Nominal Sets: Names and Symmetry in Computer Science*. Cambridge University Press, New York, NY, USA, 2013.

38 Gordon D. Plotkin and John Power. Notions of computation determine monads. In *Proceedings of the 5th International Conference on Foundations of Software Science and Computation Structures*, pages 342–356, Berlin, Heidelberg, 2002. Springer-Verlag.

39 John C. Reynolds. The essence of algol. In Peter W. O'Hearn and Robert D. Tennent, editors, *Algol-like Languages*, pages 67–88. Birkhäuser Boston, Boston, MA, 1997. doi:10.1007/978-1-4612-4118-8_4.

40 Egbert Rijke. Introduction to homotopy type theory. To appear, Cambridge University Press, 2022. doi:10.48550/arXiv.2212.11082.

41 Egbert Rijke, Michael Shulman, and Bas Spitters. Modalities in homotopy type theory. *Logical Methods in Computer Science*, 16, January 2020. doi:10.23638/LMCS-16(1:2)2020.

42 Sam Staton. Completeness for algebraic theories of local state. In Luke Ong, editor, *Foundations of Software Science and Computational Structures*, pages 48–63, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

43 Jonathan Sterling, Daniel Gratzer, and Lars Birkedal. Denotational semantics of general store and polymorphism. Unpublished manuscript, July 2022. doi:10.48550/arXiv.2210.02169.

44 Nikos Tzevelekos. Full abstraction for nominal general references. *Logical Methods in Computer Science*, Volume 5, Issue 3, September 2009. doi:10.2168/LMCS-5(3:8)2009.

45 Taichi Uemura. Cubical Assemblies, a Univalent and Impredicative Universe and a Failure of Propositional Resizing. In Peter Dybjer, José Espírito Santo, and Luís Pinto, editors, *24th International Conference on Types for Proofs and Programs (TYPES 2018)*, volume 130 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 7:1–7:20, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.TYPES.2018.7.

46 The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. https://homotopytypetheory.org/book, Institute for Advanced Study, 2013.

Guarded Hybrid Team Logics

Marius Tritschler
Technische Universität Darmstadt, Germany

Abstract

Team logics are extensions of first-order logic where formulae are not evaluated over assignments, but over sets ("*teams*") of assignments. In its most basic form, this does not increase the expressiveness of the logic because we can only form statements about the common properties of all assignments ("*flatness*"). Therefore, additional "*team atoms*" are introduced to allow for assertions about interdependencies between the assignments like dependence or inclusion. We propose to consider *binders* known from hybrid logic to increase the expressiveness, where the bound teams may then be referenced as regular relations. We call this *hybrid team logic* (HTL). Additionally, we define the positive and negative fragments of HTL (HTL^+ and HTL^-) by requiring that relations that arise from binding only occur positively or negatively, respectively.

We find that HTL and its positive and negative fragments are equivalent to prominent team logics: HTL^+ is eqivalent to inclusion logic, HTL^- is equivalent to exclusion/dependence logic and HTL itself is equivalent to independence or inclusion/exclusion logic. This classifies HTL as equivalent to existential second order logic and HTL^+ as equivalent to the positive fragment of greatest fixpoint logic.

Binders also enhance the expressiveness of guarded team logics because they enable access to information that normally is obscured by the built-in limitations of these logics. We will take a closer look at guarded hybrid team logics and establish a finite model property for the guarded fragment of HTL using model checking games. More precisely, we encode winning strategies of model checking games as relations, a process that is a natural fit for binders. Further, we notice that the hierarchy of guarded team logics is more complex than the hierarchy of non-guarded team logics, and we establish a hierarchy of prominent union-closed guarded team logics.

2012 ACM Subject Classification Theory of computation → Logic

Keywords and phrases Team semantics, guarded logics, expressiveness, model checking games

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.48

1 Introduction

Team semantics is a generalization of Tarski semantics in which logical formulae are not evaluated for single assignments, but for *sets of assignments* called *teams*. This opens new avenues to reason about interdependencies between assignments, which are relevant e.g. for large sets of data. In fact, most prominent team logics feature notions that have been studied in database theory like dependence [3], independence [14], inclusion [9] or exclusion [10].

Team semantics was originally conceived by Hodges [22] to provide a compositional, model theoretic semantics for independence-friendly logic [21]. Since then, it has been established as a basis for logics of imperfect information. Here, it is prevalent to view the aforementioned interdependencies as atomic properties of teams, an approach that emerged with Väänänen's dependence logic [29] and includes (conditional) independence logic [20] and inclusion/exclusion logic [12]. The expressiveness of these logics is well understood. On sentences, dependence, independence and exclusion logic are all equivalent to existential second order logic Σ^1_1 and inclusion logic is eqivalent to positive greatest fixpoint logic $\nu\mathrm{FO}^+$. On formulae, independence and inclusion/exclusion logic are again equivalent to Σ^1_1, while dependence logic, exclusion logic and inclusion logic are equivalent to specific fragments of

Σ^1_1 and $\nu\mathrm{FO}^+$, respectively (see Proposition 8 for more details). When including additional propositional connectives like strong negation, even dependence logic reaches the expressive power of full second order logic [24].

However, the expressive power comes at the cost of a comparatively high complexity. Therefore, it seems natural to explore variants of the mentioned team logics that are inspired by logics with desirable algorithmic and model-theoretic properties. One promising direction is the study of guarded logics with team semantics. The basic guarded logic, i.e. the guarded fragment GF of first-order logic, was introduced by Andréka, van Benthem and Németi [1] to explain and generalise the good model theoretic properties of modal logics (for an overview over modal logics, see [7, 8]). GF is defined by restricting first-order quantification in such a way that formulae can only be evaluated with respect to *guarded tuples*, which are tuples of elements that occur together in some atomic fact. This yields a logic that is decidable [1] and where every formula has a finite model [16], amongst other convenient properties. For a more in-depth survey, see e.g. [18]. Many variations and extensions of GF have been studied, for example guarded fixpoint logic, guarded second order logic or the guarded fragment over finite models (see e.g. [4, 5, 6, 17, 27]).

In particular, there exist explorations of guarded team logics by Grädel and Otto [19] and Lück [26]. These focus mostly on the analysis of guarded team logics with additional propositional connectives like strong negation, and establish guarded bisimulation as a suitable tool to analyse the expressiveness of these logics. More specifically, a core feature commonly found with guarded logics is invariance under guarded bisimulation. However, the addition of team atoms will, in general, interfere with bisimulation invariance.

This work aims to provide further insight into guarded team logics that cannot be characterised by guarded bisimulation invariance. For this, a novel team logic called "*hybrid team logic*" (HTL), as well as its positive and negative fragment, are introduced. We show that in their non-guarded version, these are equivalent to independence logic, inclusion logic and dependence logic, respectively. Further, we notice that guarded hybrid team logics are uniquely suited for a type of reduction that encodes winning strategies in model checking games as formulae in the basic guarded fragment, thus reducing the satisfiability problem to the satisfiability problem of GF and thereby providing an easy proof for decidability and a finite model property for guarded fragments of HTL. We then establish a partial expressive hierarchy of guarded team logics, which is more complex than the hierarchy of non-guarded team logics. Due to space limitations, we mainly focus on union closed logics.

2 First-Order Team Logic

In this section, we lay the foundation for the coming sections by providing basic definitions and recalling relevant results from the study of first-order team logic and its extensions.

2.1 Basic definitions

▶ **Definition 1.** *We use the following conventions throughout the paper.*

- *Let A be a set and $\overline{a} = (a_1, \dots, a_n) \in A^n$ be a tuple. We write $[\overline{a}] \coloneqq \{a_1, \dots, a_n\}$ for the set of components, $|\overline{x}| \coloneqq n$ for the length, and $\overline{a} \in A^*$ if n is irrelevant.*
- *An* assignment *to variables $\overline{x}$ into a set $A \neq \emptyset$ is a map $t\colon [\overline{x}] \to A$. We write $\mathrm{dom}(t) = [\overline{x}]$. For any tuple $\overline{y} = (y_1, \dots, y_k) \in \mathrm{dom}(t)^k$, we write $t(\overline{y}) \coloneqq (t(y_1), \dots, t(y_k))$, and similarily $t(X) \coloneqq \{t(x) \mid x \in X\}$ for $X \subseteq \mathrm{dom}(t)$.*
- *An* update $t\left[\begin{smallmatrix} a \\ x \end{smallmatrix}\right]\colon \mathrm{dom}(t) \cup \{x\} \to A$ *of an assignment t is the assignment that maps x to a, and agrees with t everywhere else. Further, $t\left[\begin{smallmatrix} \overline{a} \\ \overline{x} \end{smallmatrix}\right] = t\left[\begin{smallmatrix} a_1 \\ x_1 \end{smallmatrix}\right] \dots \left[\begin{smallmatrix} a_n \\ x_n \end{smallmatrix}\right]$.*

▶ **Definition 2.** *We use the following conventions with regard to teams.*

- *A* team *T is a set of assignments with a shared domain* $\mathrm{dom}(T)$*. A team may be empty. If* $\mathrm{dom}(T) = \emptyset$*, it may only contain the empty assignment.*
- *For tuples* $\overline{x} \in \mathrm{dom}(T)^*$ *and sets* $X \subseteq \mathrm{dom}(T)$*, we write* $T(\overline{x}) \coloneqq \{t(\overline{x}) \mid t \in T\}$ *and* $T(X) = \{t(X) \mid t \in T\}$*.*
- *Let* $F \colon T \to \mathcal{P}(A) \setminus \emptyset$*. The* update *of T (along F) is the team* $T\binom{F}{x} \coloneqq \{t\binom{a}{x} \mid t \in T, a \in F(t)\}$*. For any set* $B \subseteq A$*, we write* $T\binom{B}{x}$ *for the special case where* $F(t) = B$ *for all* $t \in T$*.*

Now, we can inductively provide team semantics for formulae of first-order logic in negation normal form. We are only using relational signatures σ.

▶ **Definition 3** (Team semantics for FO)**.** *Let* $\mathfrak{A}$ *be a* σ*-structure with universe* A*. Let* T *be a team in* $\mathfrak{A}$ *and* $\varphi \in \mathrm{FO}_\sigma$*. We assume that* $\mathrm{free}(\varphi) \subseteq \mathrm{dom}(T)$*.*

- *If* φ *is a literal then* $\mathfrak{A}, T \models \varphi$ *iff* $\mathfrak{A}, t \models \varphi$ *for all* $t \in T$*.*
- *If* $\varphi = \varphi_1 \wedge \varphi_2$ *then* $\mathfrak{A}, T \models \varphi$ *iff* $\mathfrak{A}, T \models \varphi_1$ *and* $\mathfrak{A}, T \models \varphi_2$*.*
- *If* $\varphi = \varphi_1 \vee \varphi_2$ *then* $\mathfrak{A}, T \models \varphi$ *iff there are* $T_1, T_2 \subseteq T$ *with* $T_1 \cup T_2 = T$ *so that* $\mathfrak{A}, T_1 \models \varphi_1$ *and* $\mathfrak{A}, T_2 \models \varphi_2$*. We call* T_1, T_2 *a* split *of* T*.*
- *If* $\varphi = \exists x \psi$ *then* $\mathfrak{A}, T \models \varphi$ *iff there is an update* $T\binom{F}{x}$ *so that* $\mathfrak{A}, T\binom{F}{x} \models \psi$*.*
- *If If* $\varphi = \forall x \psi$ *then* $\mathfrak{A}, T \models \varphi$ *iff* $\mathfrak{A}, T\binom{A}{x} \models \psi$*. We call* $T\binom{A}{x}$ *the* universal update *of* T *in* $\mathfrak{A}$*.*

In addition to the syntax and semantics presented here, other propositional connectives can be considered. However, most of them (like intuitionistic disjunction, implication or strong negation) do not preserve some of the following convenient properties:

▶ **Proposition 4.** FO *with team semantics satisfies the following properties:*

- ***Flatness****:* $\mathfrak{A}, T \models \varphi$ *if and only if for all* $t \in T, \mathfrak{A}, \{t\} \models \varphi$*.*
- ***Union-closure****:* $\mathfrak{A}, T_1 \models \varphi$ *and* $\mathfrak{A}, T_2 \models \varphi$ *implies* $\mathfrak{A}, T_1 \cup T_2 \models \varphi$*.*
- ***Downward closure****:* $\mathfrak{A}, T \models \varphi$ *and* $T' \subseteq T$ *implies* $\mathfrak{A}, T' \models \varphi$
- ***Locality****:* $\mathfrak{A}, T \models \varphi$ *if and only if* $\mathfrak{A}, T{\restriction_{\mathrm{free}(\varphi)}} \models \varphi$*.*
- ***Empty team property****:* $\mathfrak{A}, \emptyset \models \varphi$*.*

One could argue whether flatness is a desirable property, as it essentially reduces FO with team semantics to classical FO. Still, this version of FO with team semantics provides a clear base for the addition of team atoms.

2.2 Team Atoms

Team atoms are added to first-order team logic to describe atomic team properties that correspond to interdependencies between the assignments in a team. The first and probably best-known resulting logic was dependence logic [29]. Other notable team logics include independence logics [20], inclusion logic and exclusion logic [12] and its combinations, which extend FO with team semantics by one or more of the following team atoms:

▶ **Definition 5** (Team atoms)**.** *Let* $\mathfrak{A}$ *be a structure, let* T *be a team in* $\mathfrak{A}$ *and let* $\overline{x}, \overline{y}, \overline{z} \in \mathrm{dom}(T)^*$ *be tuples of variables, where* $\overline{x}$ *and* $\overline{y}$ *have the same length.*

- ***Dependence****:* $\mathfrak{A}, T \models \ =(\overline{x}, \overline{z})$ *if and only if the assignments to* $\overline{z}$ *depend on the assignments to* $\overline{y}$*, in the sense that for all* $t, t' \in T$*,* $t(\overline{x}) = t'(\overline{x})$ *implies* $t(\overline{z}) = t'(\overline{z})$*.*
- ***Independence****:* $\mathfrak{A}, T \models (\overline{x} \perp_{\overline{z}} \overline{y})$ *if and only if for all* $t, t' \in T$ *with* $t(\overline{z}) = t'(\overline{z})$*, there is a* $t'' \in T$ *with* $t(\overline{x}) = t''(\overline{x})$ *and* $t'(\overline{y}) = t''(\overline{y})$*.*
- ***Inclusion****:* $\mathfrak{A}, T \models (\overline{x} \subseteq \overline{y})$ *if and only if for all* $t \in T$ *there is a* $t' \in T$ *with* $t(\overline{x}) = t'(\overline{y})$*.*
- ***Exclusion****:* $\mathfrak{A}, T \models (\overline{x} | \overline{y})$ *if and only if for all* $t, t' \in T$ *we have* $t(\overline{x}) \neq t'(\overline{y})$*.*

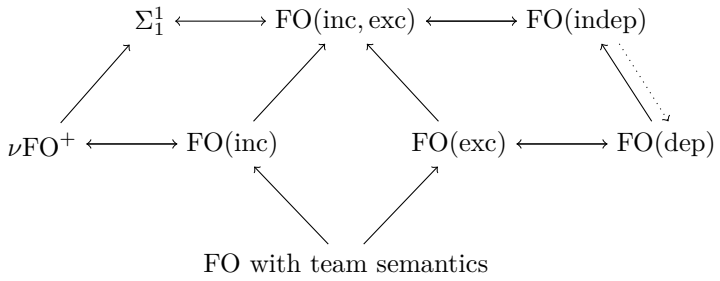


Figure 1 An overview over the hierarchy of team logics. $L \to L'$ means that L' is at least as expressive as L, with dotted arrows for sentences.

All of these are proper extensions of FO with team semantics, which can be seen by analysing which of the properties in Proposition 4 are preserved.

▶ **Proposition 6.** FO(dep, indep, inc, exc) *satisfies locality and has the empty team property.* FO(dep) *and* FO(exc) *are downward closed, but not union closed.* FO(inc) *is union-closed, but not downward closed.* FO(indep) *is neither downward nor union-closed. (cf. [12, 20, 29])*

To further describe the expressive power of these logics, we can, on one hand, compare them amongst themselves.

▶ **Proposition 7.**

- FO(dep) $\equiv$ FO(exc) *[12].*
- FO(dep) $\equiv$ FO(indep) *for sentences [29],[20].*
- FO(inc, exc) $\equiv$ FO(indep) *[12].*

On the other hand, it may be desirable to compare a team logic L to another logic L' that is not designed to handle teams. This can be achieved by interpreting the evaluation of a team over given variables as a relation. For example, for any given formula $\varphi(\overline{x}) \in L$, we can then ask whether there is a corresponding $\varphi'(R) \in L'$ with a new relation symbol R so that for every structure $\mathfrak{A}$ and team T, we have that $\mathfrak{A}, T \models \varphi(\overline{x})$ if an only if $(\mathfrak{A}, T(\overline{x})) \models \varphi'(R)$.

In particular, all standard variations of first-order team logic are fragments of existential second order Σ^1_1 in this sense. To be more precise:

▶ **Proposition 8.**

- *For every formula* $\varphi \in$ FO(indep)*, there is a corresponding sentence* $\varphi(R) \in \Sigma^1_1$ *and vice versa [12].*
- *For every formula* $\varphi \in$ FO(dep)*, there is a corresponding sentence* $\varphi(R) \in \Sigma^1_1$ *where* R *only appears negatively, and vice versa [25].*
- *For every formula* $\varphi \in$ FO(inc)*, there is a corresponding sentence* $\forall\overline{x}(R\overline{x} \to \psi(R, \overline{x})) \in \nu\text{FO}^+$ *(positive greatest fixpoint logic) and vice versa [13].*

▶ Note. Positive greatest fixpoint logic νFO$^+$ is an extension of first-order logic by positive occurrences of *greatest fixpoint operators* $[\text{gfp}_{S,\overline{x}}\, \psi(S, \overline{x})]\overline{y}$, thus being a fragment of least fixed point logic. More details can be found e.g. in [23] for fixed point logics in general, and in [13] for νFO$^+$ in particular.

See Figure 1 for a summary of this section.

3 Hybrid Team Logics

We introduce a new team logic called hybrid team logic. The name is inspired by hybrid modal logics, a collection of extensions of modal logic by first-order machinery that was first introduced in 1967 by Prior in [28] to deal with specific issues in temporal logics. For a detailed account of the fundamentals of hybrid logics, see e.g. [2].

One of the main features of modal hybrid logics is the $\downarrow$ *binder*, which was introduced by Goranko in [15] to "bind" the current world as the interpretation of a constant. This concept can be transferred to team logics in the sense that teams can be bound as interpretations of new relational variables.

▶ **Definition 9.** Hybrid team logic (HTL) *is an extension of first-order logic with team semantics by* binders $\downarrow$ *with the following semantics: for all structures* $\mathfrak{A}$*, teams* T*, variables* $\overline{x} \in \mathrm{dom}(T)^*$ *and formulae* $\varphi(X) \in \mathrm{HTL}$ *where* X *is a new relation symbol of arity* $|\overline{x}|$*,*

$$\mathfrak{A}, T \models \downarrow_{\overline{x}} X\varphi(X) \quad \Leftrightarrow \quad (\mathfrak{A}, T(\overline{x})), T \models \varphi(X).$$

The variables in $\overline{x}$ *are considered free variables, i.e.* $\mathrm{free}(\downarrow_{\overline{x}} X\varphi(X)) = \mathrm{free}(\varphi) \cup [\overline{x}]$.
The positive (negative) fragment HTL^+ *(*HTL^-*) is the fragment of* HTL *where bound relations may only occur positively (negatively).*

We immediately notice that inclusion and exclusion atoms can be expressed in hybrid team logic on an elementary level.

▶ **Lemma 10.** $\mathrm{FO(inc)} \subseteq \mathrm{HTL}^+$, $\mathrm{FO(exc)} \subseteq \mathrm{HTL}^-$ *and* $\mathrm{FO(inc, exc)} \subseteq \mathrm{HTL}$.

Proof. We need to show that for every $\varphi \in \mathrm{FO(inc)}$ ($\mathrm{FO(exc)}$, $\mathrm{FO(inc, exc)}$), there is a $\varphi' \in \mathrm{HTL}^+$ (HTL^-, HTL) so that for all structures $\mathfrak{A}$ and teams T,

$$\mathfrak{A}, T \models \varphi \quad \Leftrightarrow \quad \mathfrak{A}, T \models \varphi'.$$

Recall that X is a relational variable, i.e. $\mathfrak{A}, T \models X\overline{x}$ if and only if $T(\overline{x}) \subseteq X^{\mathfrak{A}}$ and $\mathfrak{A}, T \models \neg X\overline{x}$ if and only if $T(\overline{x}) \cap X^{\mathfrak{A}} = \emptyset$. With that, it is straightforward to verify that

$$\mathfrak{A}, T \models (\overline{x} \subseteq \overline{y}) \quad \Leftrightarrow \quad \mathfrak{A}, T \models \downarrow_{\overline{y}} X(X\overline{x})$$

and

$$\mathfrak{A}, T \models (\overline{x}|\overline{y}) \quad \Leftrightarrow \quad \mathfrak{A}, T \models \downarrow_{\overline{y}} X(\neg X\overline{x}).$$

With that, we can replace every occurrence of an inclusion or exclusion atom in φ to get the desired φ'. ◀

Taking a closer look at the positive fragment HTL^+ of hybrid team logic, we shall see that it is equivalent to inclusion logic FO(inc) and positive greatest fixpoint logic $\nu\mathrm{FO}^+$. Considering Lemma 10, it is enough to show that HTL^+ is a fragment of $\nu\mathrm{FO}^+$ in the sense of Proposition 8.

One way to think about this equivalence is that each of the logics in question provides its own tools (independence atoms, binders, teams, fixpoints), which can be *simulated* by the other logics. For example, in the proof of $\mathrm{FO(inc)} \equiv \nu\mathrm{FO}^+$ in [13], fixpoints are simulated in FO(inc) by expanding any given team to a cartesian product, i.e. introducing a second team (over fresh free variables) that represents the fixpoint and can be handled independently.

Another example already occurred in Proposition 8, where $\nu\mathrm{FO}^+$ could be used to simulate teams via an additional relation. In general, this team will be manipulated when evaluating a formula, e.g. by splitting or updating. This can be simulated with the help of fixpoints.

▶ **Theorem 11.** *For every* $\varphi(\overline{x}) \in \mathrm{HTL}^+$ *there is a* $\varphi^*(R, \overline{x}) \in \nu\mathrm{FO}^+$ *so that* R *only appears positively and*

$$\mathfrak{A}, T \models \varphi(\overline{x}) \quad \Leftrightarrow \quad (\mathfrak{A}, T(\overline{x})), t \models \varphi^*(R, \overline{x}) \text{ for all } t \in T.$$

Proof. We use syntactic induction and retrace the proof of Theorem 15 in [13] regarding everything except binders. If $\varphi = {\downarrow_{\overline{x}}} S\psi(S, \overline{xy})$, by induction we have $\psi^*(R, S, \overline{xy})$ and use

$$\varphi^* = \psi^*(R, \exists \overline{y} R\_\overline{y}, \overline{xy}),$$

i.e. $\exists \overline{y} R\_\overline{y}$ is supposed to replace S in the sense that every instance of $S\overline{x}$ is replaced by $\exists \overline{y} R\overline{xy}$. This way, we have

$$\begin{aligned}
& \mathfrak{A}, T \models \varphi \\
\Leftrightarrow \quad & (\mathfrak{A}, T(\overline{x})), T \models \psi(S, \overline{xy}) \\
\Leftrightarrow \quad & (\mathfrak{A}, T(\overline{xy}), T(\overline{x})), t \models \psi^*(R, S, \overline{xy}) \text{ for all } t \in T \\
\Leftrightarrow \quad & (\mathfrak{A}, T(\overline{xy})), t \models \varphi^* \text{ for all } t \in T,
\end{aligned}$$

because $(\mathfrak{A}, T(\overline{x})), t \models S\overline{x}$ if and only if $(\mathfrak{A}, T(\overline{xy})), t \models \exists \overline{y} R\overline{xy}$. ◀

▶ **Corollary 12.** $\mathrm{HTL}^+ \equiv \mathrm{FO}(\mathrm{inc}) \equiv \nu\mathrm{FO}^+$

The question of whether the unique features available in some logic can be simulated in another logic will be revisited in Section 5.

For the sake of completeness, we mention that the other team logics are also equivalent to their respective counterparts from Lemma 10. A proof can be found in Appendix A.

▶ **Proposition 13.** $\mathrm{HTL}^- \equiv \mathrm{FO}(\mathrm{exc})$ *and* $\mathrm{HTL} \equiv \mathrm{FO}(\mathrm{inc}, \mathrm{exc})$.

4 Guarded Team Logics

In classical first-order logic, there are several equivalent ways to define the guarded fragment. In particular, there are syntactic and semantic definitions. In the former case, we add guards to quantification in the sense that, if quantification appears in a formula, it has to have the form $\exists \overline{x}(\alpha(\overline{xy}) \wedge \psi(\overline{xy}))$ or $\forall \overline{x}(\alpha(\overline{xy}) \to \psi(\overline{xy}))$ for some guard α. In the latter case, we require the images of all assignments to be guarded.

Here, we work with one of the least restrictive variations of guarded logics, similar to [18].

▶ **Definition 14.** *Let* σ *be a relational signature and* $\mathfrak{A}$ *be a* σ*-structure with universe* A*. The* set of guards $\mathbb{G}(\mathfrak{A})$ *is defined as*

$$\mathbb{G}(\mathfrak{A}) \coloneqq \{G \subseteq A \mid G \subseteq [\overline{a}], \overline{a} \in R^{\mathfrak{A}}, R \in \sigma \cup \{=\}\}.$$

As we can see, the set of guards in $\mathfrak{A}$ contains all sets that are completely "covered" by a tuple that occurs in a relation. The inclusion of "=" in the selection of guards entails that all singleton sets are guarded.

▶ **Definition 15.** *Let* $\mathfrak{A}$ *be a relational structure.*

- *A tuple* $\overline{a}$ *is guarded in* $\mathfrak{A}$ *if* $[\overline{a}] \in \mathbb{G}(\mathfrak{A})$.
- *An assignment* t *is guarded if* $t(\mathrm{dom}(t)) \in \mathbb{G}(\mathfrak{A})$.
- *A team* T *is guarded if* $T(\mathrm{dom}(T)) \subseteq \mathbb{G}(\mathfrak{A})$.

It is clear that, in general, not all updates of guarded teams are guarded. However, there always is a *universal guarded update*, i.e. a unique maximal guarded update that can take the place of universal updates in guarded semantics. With this, we can define a guarded variant of team logic by replacing the classical quantification of FO with *guarded quantification*.

▶ **Definition 16.** Guarded team logic GTL *is defined analogously to standard* FO *with team semantics, where quantification is replaced by guarded quantification in the following sense:*

- $\mathfrak{A}, T \models \forall_g x\psi$ *iff* $A, T' \models \psi$ *for the universal guarded update* T' *of* $T{\restriction_{\mathrm{free}(\psi)\setminus\{x\}}}$.
- $\mathfrak{A}, T \models \exists_g x\psi$ *iff* $A, T' \models \psi$ *for some guarded update* T' *of* $T{\restriction_{\mathrm{free}(\psi)\setminus\{x\}}}$.

For tuples $\overline{x} = (x_1, \dots, x_n)$*, we write* $\exists_g \overline{x}\psi$ *instead of* $\exists_g x_1 \dots \exists_g x_n \psi$ *(similarly for* $\forall$*).*

Extending GTL *by adding binders, inclusion atoms etc. yields guarded hybrid team logic* GHTL*, guarded inclusion logic* GTL(inc) *etc. respectively. Binding teams with the* $\downarrow$*-operator does not change the set of guards.*

One of the features of guarded logics is the fact that quantification can be thought of as moving from guarded patch to guarded patch. This is reflected by the evaluation of guarded quantification through the restriction of T to those variables that are relevant for the inner formula.

▶ Note. In general, there are several options to design guarded team logics. One obvious candidate would be to use team semantics with the standard guarded fragment of FO, which would entail guarding each quantification with a relational atom. However, this would prohibit mixed teams, i.e. teams that contain assignments that are guarded by different relations. In the non-team case, there is no tangible difference between both versions as all assignments are considered independently, anyway. With teams, however, there are cases where extensions of GF with team semantics are strictly weaker than extensions of GTL as defined above (see Appendix B.2 for details).

▶ Note. GTL is technically not a fragment of FO with team semantics. The reason is that guarded quantification implicitly introduces a disjunction over all relations in the signature. For infinite signatures, this is not reproducible in FO. For any finite signature however, we can find a translation between GTL and FO with team semantics (and their extensions). For the rest of this paper, we therefore assume that all signatures are finite if not explicitly stated otherwise (see Appendix B for details).

4.1 Properties of Guarded Team Logics

In Sections 2 and 3, we introduced several team logics and mentioned their properties in Proposition 4. It is straightforward to verify that these properties translate to the respective guarded variants of these logics. (See Appendix B for more details.)

This leaves the question whether we can lift desirable properties of classical guarded logic to the team setting. We take a closer look at two specific properties.

▶ **Proposition 17** ([1, 16])**.** *The guarded fragment of classical first-order logic is* decidable *since it has the* finite model property*, i.e. every satisfiable formula has a finite model.*

▶ Remark 18. All logics with team semantics that are examined here have the empty team property, i.e. all formulae are satisfied by the empty team across all structures. A sensible notion of decidability and finite model property for team logics would therefore only regard satisfaction by non-empty teams.

We can immediately show that guarded dependence logic GTL(dep) cannot have the finite model property:

▶ **Lemma 19.** *Let E be a binary relation symbol. The sentence*

$$\begin{aligned}\varphi \coloneqq \exists_g x(\forall_g y \neg Eyx) \wedge \\ \forall_g z(\exists_g w(z \neq w \wedge Ezw) \wedge \\ \forall_g w(\neg Ezw \vee (Ezw \wedge\ =(z,w) \wedge\ =(w,z))))\end{aligned}$$

is a satisfiable formula in GTL(dep) *that is satisfied by a structure $\mathfrak{A}$ only if $\mathfrak{A}$ contains at least one infinite simple E-path.*

Proof. If $\mathfrak{A} \models \varphi$, then there is at least one element a without any E-predecessor. We also know that every element has exactly one successor and at most one predecessor. This implies that $\mathfrak{A}$ consists of only cycles or infinite paths with or without starting point. The vertex a cannot lie on a circle, so it has to be the starting point of an infinite path. Clearly, $(\mathbb{N}, \{(n, n+1) \mid n \in \mathbb{N}\})$ satisfies φ. ◀

In contrast, guarded hybrid team logic GHTL has the finite model property and is decidable. Both can be shown by reducing the satisfiability of formulae in GHTL to the satisfiability of the classical guarded fragment.

▶ **Theorem 20.** *Let σ be a relational signature and $\varphi(\overline{x}) \in \mathrm{GHTL}_\sigma$. There is a signature $\tau \supseteq \sigma \cup \{R_\varphi\}$ and a formula $\varphi^*(R_\varphi) \in \mathrm{GF}_\tau$ so that all σ-structures $\mathfrak{A}$ and guarded teams T satisfy φ if and only if there is a expansion of $\mathfrak{A}$ to a τ-structure $\mathfrak{A}^*$ so that $(\mathfrak{A}^*, T(\overline{x})) \models \varphi^*(R_\varphi)$.*

Proof. The general strategy is to expand σ by relation symbols R_ψ for all instances of subformulae ψ of φ. Then, we include clauses in φ^* that are only satisfied if the interpretations of the R_ψ provide a winning strategy for a basic model checking game that is similar to the one presented in [29, section 5.2].

Let $S(\varphi)$ be the set of (instances of) subformulae of φ, including φ itself, and let $\mathcal{X}(\varphi)$ be the set of relational variables that are bound in φ. We define $\tau = \sigma \cup \{R_\psi\}_{\psi \in S(\varphi)} \cup \mathcal{X}(\varphi)$ and formulae $\varphi^*(\psi)$ according to Appendix C.1 and define

$$\varphi^* \coloneqq \bigwedge_{\psi \in S(\varphi)} \varphi^*(\psi).$$

There are a few technical details that have to be considered (see Appendix C.2), but overall, it is straightforward to verify by syntactic induction that this is as required. ◀

We see that guarded hybrid team logic is uniquely suited for this type of translation, because the concept of interchangeability between teams and relations that is at the core of the proof is already included in the logic itself.

Moreover, we get a glimpse of why guarded dependence logic loses the finite model property: the dependence atom is fundamentally non-guarded in the sense that it cannot be defined by a formula in classical guarded logic. Another explanation would be the similarity of GF(dep) to guarded logic with counting quantifiers, which also does not have the finite model property and is undecidable [16].

▶ **Corollary 21.** GHTL *has the finite model property and is decidable in the sense of Remark 18.*

Proof. From Theorem 20, we know that any $\varphi \in$ GHTL is satisfied by some (possibly infinite) structure $\mathfrak{A}$ and non-empty team T with domain $\overline{x}$ if and only if $\varphi^* \wedge \exists \overline{x} R_\varphi \overline{x}$ is satisfied by an appropriate expansion $\mathfrak{A}^*$ of $\mathfrak{A}$ with $R_\varphi^{\mathfrak{A}^*} = T(\overline{x})$.

- Due to the finite model property of GF, we find a finite model $(\mathfrak{B}^*, T'(\overline{x}))$ satisfying $\varphi^* \wedge \exists \overline{x} R_\varphi \overline{x}$. Using Theorem 20, we get a finite model $\mathfrak{B}, T'$ of φ with non-empty T' which proves the finite model property.
- The translation from φ to $\varphi^* \wedge \exists \overline{x} R_\varphi \overline{x} \in \mathrm{GF}$ is computable in linear time, so decidability of GHTL follows directly from the decidability of GF. ◀

5 A Hierarchy of Union-Closed Team Logics

One takeaway from Lemma 19 and Corollary 21 is that the hierarchy of team logics in the guarded case differs from the hierarchy in the non-guarded case.

▶ **Corollary 22.** $\mathrm{GTL}(\mathrm{dep}) \not\subseteq \mathrm{GHTL}^-$.

Of course, all the obvious inclusions like $\mathrm{GTL}(\mathrm{exc}) \subseteq \mathrm{GTL}(\mathrm{inc}, \mathrm{exc})$ still hold, and the translations of Lemma 10 work on an atomic level and are therefore unaffected by changes to quantification.

▶ **Corollary 23.** $\mathrm{GTL}(\mathrm{inc}) \subseteq \mathrm{GHTL}^+$, $\mathrm{GTL}(\mathrm{exc}) \subseteq \mathrm{GHTL}^-$, $\mathrm{GTL}(\mathrm{inc}, \mathrm{exc}) \subseteq \mathrm{GHTL}$.

This also implies that there are formulae in guarded dependence logic that cannot be expressed in guarded exclusion logic, contrary to the non-guarded case.

In this section, we further investigate the relative expressiveness of union-closed guarded team logics, i.e. guarded inclusion logic $\mathrm{GTL}(\mathrm{inc})$, positive guarded hybrid team logic GHTL^+ and guarded positive greatest fixpoint logic $\nu\mathrm{GF}^+$.

An upper bound for the expressiveness of these logics is GHTL. It is clear that $\mathrm{GTL}(\mathrm{inc}) \subseteq \mathrm{GHTL}^+ \subseteq \mathrm{GHTL}$. The proof for $\nu\mathrm{GF}^+ \subseteq \mathrm{GHTL}$ can roughly be outlined as follows:

1. As mentioned towards the end of Section 3, in the non-guarded case, we could simulate fixpoints by introducing them as new teams over fresh variables. This way, we could effectively handle more than one team simultaneously. However, this strategy is not available any more because in general, it would require a shared guard for all teams.
2. Let L be a team logic and R a relation symbol that may occur both positively and negatively. For all tuples $\overline{x}$ of appropriate length and $\psi \in L$ with $\mathrm{free}(\psi) \subseteq [\overline{x}]$, we can define a sentence $\varphi(R, \psi)$ that is satisfied in a structure $\mathfrak{A}$ if and only if $R^{\mathfrak{A}}$, interpreted as a team with domain $[\overline{x}]$, satisfies ψ.
3. We can bind a team, introduce a fixpoint, bind the fixpoint, and then "recover" the team (using the sentence in 2.) and check whether it is in the fixpoint. This circumvents the problem in 1.

However, the same strategy cannot be employed in GHTL^+ or GHTL^-, because the sentence in 2. uses both positive and negative instances of R. Neither $\nu\mathrm{GF}^+$ nor GHTL^- contains the other because the former is union-closed and the latter is downward closed, but not vice versa. By contrast, Theorem 11 can be easily adapted.

▶ **Lemma 24.** *For every $\varphi(\overline{x}) \in \mathrm{GHTL}^+$ there is a formula $\varphi^*(R, \overline{x}) \in \nu\mathrm{GF}^+$ in which R only appears positively and*

$$\mathfrak{A}, T \models \varphi(\overline{x}) \quad \Leftrightarrow \quad (\mathfrak{A}, T(\overline{x})), t \models \varphi^*(R, \overline{x}) \text{ for all } t \in T.$$

Proof. The proof is very similar to the one of Theorem 11. In most steps, quantification does not matter, so we focus on the ones where it does:

- If $\varphi = \exists_g \overline{y} \psi(\overline{xy})$, let

  $$\varphi^* = \exists_g \overline{y} [\mathrm{gfp}_{S,\overline{xy}} \exists_g \overline{z} R\overline{xz} \wedge \psi^*(S, \overline{xy})]\overline{xy},$$

  where the length of $\overline{xz}$ is equal to the arity of R. The tuple $\overline{z}$ is supposed to represent the variables that are dropped from the domain of the team when evaluating the quantification. An identical argument can be made for universal quantification.
- If $\varphi = {\downarrow_{\overline{x}}} S\psi(S, \overline{xy})$, let

  $$\varphi^* = \psi^*(R, \exists_g \overline{y} R\_\overline{y}, \overline{xy}),$$

  matching the non-guarded case. ◀

We shall see that we have $\mathrm{GHTL}^+ \equiv \nu\mathrm{GF}^+$ for sentences but $\mathrm{GHTL}^+ \subsetneq \nu\mathrm{GF}^+$ for arbitrary formulae.

5.1 $\mathrm{GHTL}^+ \equiv \nu\mathrm{GF}^+$ on Sentences

To simplify notation, we introduce some abbreviations:
- We write $[\mathrm{gfp}\, \psi]$ instead of $[\mathrm{gfp}_{S,\overline{x}}\, \psi(S, \overline{x})]$ whenever the variables are not in focus.
- For all structures $\mathfrak{A}$ and fixpoints $[\mathrm{gfp}\, \psi]$, we write $[\mathrm{gfp}\, \psi]^{\mathfrak{A}}$ for the interpretation of $[\mathrm{gfp}\, \psi]$ in $\mathfrak{A}$.

One direction of $\mathrm{GHTL}^+ \equiv \nu\mathrm{GF}^+$ is already included in Lemma 24. To show $\nu\mathrm{GF}^+ \subseteq \mathrm{GHTL}^+$, we provide a translation $\varphi^* \in \mathrm{GHTL}^+$ for any sentence $\varphi \in \nu\mathrm{GF}^+$. To do that, we design φ^* so that first, all necessary fixpoints are simulated and bound, and then the sentence is evaluated as usual, with the bound pseudo-fixpoints replacing the actual fixpoints.

In the process of applying the translation, we transition from $\nu\mathrm{GF}^+$ to GHTL^+ fixpoint by fixpoint. This leads to intermediate steps that involve the syntax of both logics. To handle this combined logic, we need team semantics for $\nu\mathrm{GF}^+$, which we get by extending the semantics of guarded team logic GTL by a clause for fixpoints:

▶ **Definition 25.** *For all structures $\mathfrak{A}$, teams T and fixpoints $[\mathrm{gfp}\, \psi]$,*

$$\mathfrak{A}, T \models [\mathrm{gfp}\, \psi]\overline{x} \quad :\Leftrightarrow \quad T(\overline{x}) \subseteq [\mathrm{gfp}\, \psi]^{\mathfrak{A}}.$$

▶ **Lemma 26.** *Let $\psi \in \nu\mathrm{GF}^+$ and let R be a new relation symbol.*
1. *$\mathfrak{A}, T \models [\mathrm{gfp}\, \psi]\overline{x}$ iff $(\mathfrak{A}, [\mathrm{gfp}\, \psi]^{\mathfrak{A}}), T \models R\overline{x}$.*
2. *$\nu\mathrm{GF}^+$ with team semantics has the flatness property.*
3. *$\mathfrak{A}, \{t\} \models \psi$ iff $\mathfrak{A}, t \models \psi$.*
4. *If $(\mathfrak{A}, T(\overline{x})), T \models \psi(S, \overline{x})$, then $\mathfrak{A}, T \models [\mathrm{gfp}\, \psi]\overline{x}$.*
5. *If T is maximal so that $\mathfrak{A}, T \models [\mathrm{gfp}\, \psi]\overline{x}$, then $(\mathfrak{A}, T), T \models \psi(S, \overline{x})$.*

Proof. 1. follows directly from Definition 25. The proofs for 2. and 3. are identical to the proofs for non-guarded team logic, using 1. The proofs of 4. and 5. can then be reduced to the respective proofs of [13, Lemma 14]. ◀

▶ Note. Lemma 26, part 2, does not imply flatness for GHTL^+, even though it is a fragment in the sense of Lemma 24. For this, the translations φ^* would have to be flat with regard to the relational encoding of the team, i.e.

$$\text{for all } t \in T : (\mathfrak{A}, T(\overline{x})), t \models \varphi^* \Leftrightarrow \text{for all } t' \in T : (\mathfrak{A}, \{t'(\overline{x})\}), t' \models \varphi^*.$$

As is, fixpoints may contain non-guarded tuples because the inner formulae are satisfiable by non-guarded tuples (e.g. $[\mathrm{gfp}_{S,x_1x_2}\ \top]^{\mathfrak{A}}$ contains every pair in the universe of $\mathfrak{A}$). However, there are ways to replace these fixpoints without changing the guarded parts.

▶ **Lemma 27.** *Let $\varphi \in \nu\mathrm{GF}^+$, let $\mathfrak{A}$ be a structure and $\overline{a}$ be a tuple in $\mathfrak{A}$.*

1. *There is a formula $\varphi^g \in \nu\mathrm{GF}^+$ such that $\mathfrak{A}, \overline{a} \models \varphi^g$ if and only if $\mathfrak{A}, \overline{a} \models \varphi$ and $\overline{a} \in \mathbb{G}(\mathfrak{A})$.*
2. *$[\mathrm{gfp}\,\varphi^g]^{\mathfrak{A}} = [\mathrm{gfp}\,\varphi]^{\mathfrak{A}} \cap \mathbb{G}(\mathfrak{A})$.*

Proof.

1. We can construct φ^g using trivial quantification, e.g. $\varphi^g(\overline{x}) \coloneqq \exists_g \overline{x}'(\overline{x}' = \overline{x} \wedge \varphi(\overline{x}'))$.
2. To show $[\mathrm{gfp}\,\varphi^g]^{\mathfrak{A}} \subseteq [\mathrm{gfp}\,\varphi]^{\mathfrak{A}} \cap \mathbb{G}(\mathfrak{A})$, we use the first part of this lemma twice. First, we immediately get $[\mathrm{gfp}\,\varphi^g]^{\mathfrak{A}} \subseteq \mathbb{G}(\mathfrak{A})$ because φ^g is only satisfiable by guarded tuples. Second, for all $\overline{a} \in [\mathrm{gfp}\,\varphi^g]^{\mathfrak{A}}$, we have $(\mathfrak{A}, [\mathrm{gfp}\,\varphi^g]^{\mathfrak{A}}), \overline{a} \models \varphi^g(\overline{x}, S)$ and therefore $(\mathfrak{A}, [\mathrm{gfp}\,\varphi^g]^{\mathfrak{A}}), \overline{a} \models \varphi(\overline{x}, S)$ and with an argument similar to [13, Lemma 14] we get $\mathfrak{A}, \overline{a} \models [\mathrm{gfp}\,\varphi]\overline{x}$.
   For the other direction, let $\overline{a} \in [\mathrm{gfp}\,\varphi]^{\mathfrak{A}} \cap \mathbb{G}(\mathfrak{A})$, so we have $(\mathfrak{A}, [\mathrm{gfp}\,\varphi]^{\mathfrak{A}}), \overline{a} \models \varphi(\overline{x}, S)$ and therefore $(\mathfrak{A}, [\mathrm{gfp}\,\varphi]^{\mathfrak{A}}), \overline{a} \models \varphi^g(\overline{x}, S)$ because $\overline{a}$ is guarded. When evaluating $\varphi^g(\overline{x}, S)$, every occurrence of $S\overline{y}$ will be evaluated by a guarded tuple (because $\overline{a}$ and every update of $\overline{a}$ due to quantification is guarded). The non-guarded part of $[\mathrm{gfp}\,\varphi]^{\mathfrak{A}}$ is therefore irrelevant for the evaluation and can be omitted, which yields $(\mathfrak{A}, [\mathrm{gfp}\,\varphi]^{\mathfrak{A}} \cap \mathbb{G}(\mathfrak{A})), \overline{a} \models \varphi^g(\overline{x}, S)$. Again, we refer to [13, Lemma 14] to get $\mathfrak{A}, \overline{a} \models [\mathrm{gfp}\,\varphi^g]\overline{x}$ and are done. ◀

We can now provide a tool to replace fixpoints by bound teams. To this end we allow the creation of formulae that may contain both binders and fixpoints, as long as no fixpoint contains a binder (bound relations are allowed). This way, we can always find a fixpoint with a first-order inner formula, which is key for the translation to work.

▶ **Lemma 28.** *Let $\varphi([\mathrm{gfp}\,\psi])$ be a sentence in $\nu\mathrm{GF}^+$ so that $[\mathrm{gfp}\,\psi]$ appears in φ and ψ itself does not contain fixpoints. Then*

$$\varphi([\mathrm{gfp}\,\psi]) \equiv \exists_g \overline{x} \downarrow_{\overline{x}} X(\psi(X, \overline{x}) \wedge \varphi(X)).$$

Proof. We show the equivalence for an arbitrary structure $\mathfrak{A}$. Without loss of generality, we can assume that fixpoints are guarded because they are always evaluated by guarded teams. As such, only the guarded part of the fixpoint matters, and we can always find a corresponding fixpoint due to Lemma 27.

For one direction, assume $\mathfrak{A} \models \varphi([\mathrm{gfp}\,\psi])$. Because we can assume that fixpoints are guarded, it suffices to show

$$(\mathfrak{A}, [\mathrm{gfp}\,\psi]^{\mathfrak{A}}), [\mathrm{gfp}\,\psi]^{\mathfrak{A}} \models \psi(X, \overline{x}) \wedge \varphi(X).$$

For $(\mathfrak{A}, [\mathrm{gfp}\,\psi]^{\mathfrak{A}}), [\mathrm{gfp}\,\psi]^{\mathfrak{A}} \models \psi(X, \overline{x})$, we use Lemma 26, part 5, as $[\mathrm{gfp}\,\psi]^{\mathfrak{A}}$ obviously is maximal. Also, we can use the assumption and Lemma 26, part 1, to replace every instance of $[\mathrm{gfp}\,\psi]\overline{x}$ in φ with $X\overline{x}$ to get $(\mathfrak{A}, [\mathrm{gfp}\,\psi]^{\mathfrak{A}}), [\mathrm{gfp}\,\psi]^{\mathfrak{A}} \models \varphi(X)$.

For the other direction, let T be a witness for the existential claim. From $(\mathfrak{A}, T(\overline{x})), T \models \psi(X, \overline{x})$, it follows that $T \subseteq [\mathrm{gfp}\,\psi]^{\mathfrak{A}}$ according to Lemma 26, part 4, and Definition 25. We also have $(\mathfrak{A}, T(\overline{x})) \models \varphi(X)$, and because X occurs only positively in φ, satisfaction of φ is preserved whenever the interpretation of X is replaced by a superset. This yields $(\mathfrak{A}, [\mathrm{gfp}\,\psi]^{\mathfrak{A}}) \models \varphi(X)$. We replace all instances of $X\overline{x}$ with $[\mathrm{gfp}\,\psi]\overline{x}$ using Lemma 26, part 1, and are done. ◀

▶ **Theorem 29.** *$\nu\mathrm{GF}^+ \equiv \mathrm{GHTL}^+$ for sentences.*

Proof. $\mathrm{GHTL}^+ \subseteq \nu\mathrm{GF}^+$ is already established for formulae in Lemma 24. Towards $\nu\mathrm{GF}^+ \subseteq \mathrm{GHTL}^+$, let σ be a signature and $\varphi \in \nu\mathrm{GF}^+$ be a sentence. We start by applying Lemma 28 on one of the "innermost" fixpoints $[\mathrm{gfp}\,\psi_1]$ of φ, i.e. one that does not contain other fixpoints, to get

$$\varphi_1 \coloneqq \exists_g \overline{x}\, {\downarrow_{\overline{x}}} X_1(\psi_1(X_1, \overline{x}) \wedge \varphi(X_1)).$$

We notice that $\varphi(X_1)$ is again a sentence in $\nu\mathrm{GF}^+$ over the signature $\sigma \cup \{X_1\}$ that now contains one fixpoint less. This allows for repeated application of Lemma 28 until all fixpoints are eliminated and we get a formula of the form

$$\begin{aligned}\varphi_n \coloneqq &\exists_g \overline{x}_1\, {\downarrow_{\overline{x}}} X_1(\psi_1(X_1, \overline{x}_1)\\ &\wedge \exists_g \overline{x}_2\, {\downarrow_{\overline{x}}} X_2(\psi_2(X_2, \overline{x}_2)\\ &\dots\\ &\wedge \exists_g \overline{x}_n\, {\downarrow_{\overline{x}}} X_n(\psi_n(X_n, \overline{x}_n) \wedge \varphi(X_1, \dots, X_n))\dots)),\end{aligned}$$

where $\varphi(X_1, \dots, X_n)$ is a first-order sentence. All instances of bound relations are positive, because they only replace fixpoints or relation variables in fixpoints, which only appear positive in $\nu\mathrm{GF}^+$. Therefore, $\varphi_n \in \mathrm{GHTL}^+$ and we are done. ◀

5.2 $\mathbf{GF(inc) \subsetneq GHTL^+ \subsetneq \nu GF^+}$ on Formulae

The aim of this section is to show that on formulae, guarded positive greatest fixpoint logic is more expressive than positive guarded hybrid team logic, which in turn is more expressive than guarded inclusion logic. For this, we take a closer look at the nature of quantification in guarded logics.

Quantification in guarded logics can either be *local* or *global*, which means that the new assignment or team either has to be guarded together with (parts of) the old assignment or team, or the previous team can be "forgotten". More specifically, we say that a subformula $Q_g\overline{x}\psi$ (where $Q \in \{\forall, \exists\}$) corresponds to a global move if $\mathrm{free}(\psi) \subseteq [\overline{x}]$, i.e. $Q_g\overline{x}\psi$ is a sentence. This motivates the following definitions :

▶ **Definition 30.**

- *For all formulae φ, any subformula of φ that is a sentence on its own is called a* subsentence *of φ. In general, there will be subsentences that do contain other subsentences.*
- *Gaifman-neighbourhoods [11]: For any σ-structure $\mathfrak{A}$ with universe A and subset $B \subseteq A$, we inductively define the l-*neighbourhood *$\mathfrak{n}(B, l)$ according to $\mathfrak{n}(B, 0) = B$ and*

$$\mathfrak{n}(B, l+1) = \left\{ a \in A \;\middle|\; \exists \overline{a} \in \bigcup_{R \in \sigma} R^{\mathfrak{A}} \text{ s.t. } a \in [\overline{a}] \text{ and } [\overline{a}] \cap \mathfrak{n}(B, l) \neq \emptyset \right\}.$$

- *For any assignment t with domain $\mathrm{dom}(t) = X$, we write $\mathfrak{n}(t, l) \coloneqq \mathfrak{n}(t(X), l)$. For any team T with domain $\mathrm{dom}(T) = X$, we define*

$$\mathfrak{n}(T, l) \coloneqq \mathfrak{n}\left(\bigcup_{t \in T} t(X), l\right), \quad \text{or equivalently} \quad \mathfrak{n}(T, l) \coloneqq \bigcup_{t \in T} \mathfrak{n}\,(t, l)\,.$$

- *Let T be a team in $\mathfrak{A}$ with $\mathrm{dom}(T) = X$ and $T' \subseteq T$. We say that T' is an l-*local cluster *in T if, for all $t \in T$, we have $t \in T'$ or*

$$\mathfrak{n}(t, l) \cap \mathfrak{n}(T', l) = \emptyset.$$

- *Let $\mathfrak{A}, \mathfrak{B}$ be structures, $T_\mathfrak{A}, T_\mathfrak{B}$ be teams with domain X and $C_\mathfrak{A}, C_\mathfrak{B}$ be l-local clusters in $T_\mathfrak{A}, T_\mathfrak{B}$ respectively. We say that $C_\mathfrak{A}$ and $C_\mathfrak{B}$ are* l-locally isomorphic *($\mathfrak{A}, C_\mathfrak{A} \simeq^l \mathfrak{B}, C_\mathfrak{B}$) if there is a bijection $\pi\colon C_\mathfrak{A} \to C_\mathfrak{B}$ that can be extended to an isomorphism $\iota\colon \mathfrak{n}(C_\mathfrak{A}, l) \to \mathfrak{n}(C_\mathfrak{B}, l)$ on the induced substructures in the sense that for all $x \in X$ and $t_\mathfrak{A} \in T_\mathfrak{A}$, we have $\iota(t_\mathfrak{A}(x)) = \pi(t_\mathfrak{A})(x)$.*
- *The local rank $\mathfrak{lr}(\varphi)$ is defined inductively, identically to the standard quantifier rank for guarded logics, with one essential difference: $\mathfrak{lr}(\varphi) = 0$ if φ is a (sub)sentence.*

The concept of local clusters and local isomorphism gives us a strong criterion for local indistinguishability.

5.2.1 GHTL$^+ \not\subseteq$ GF(inc)

If there are two teams in the same structure that cannot be distinguished locally, both teams satisfy the same GF(inc)-formulae.

▶ **Lemma 31.** *Let $\mathfrak{A}$ be a σ-structure and $l \in \mathbb{N}$. Let T_1 and T_2 be teams in $\mathfrak{A}$ such that for every $(l+1)$-local cluster C_2 in T_2, there is an $(l+1)$-local cluster C_1 in T_1 such that $\mathfrak{A}, C_1 \simeq^{l+1} \mathfrak{A}, C_2$. Let $\varphi \in$ GF(inc) be a formula with locality rank $\mathfrak{lr}(\varphi) \leq l$. Then $\mathfrak{A}, T_1 \models \varphi$ implies $\mathfrak{A}, T_2 \models \varphi$.*

Proof. The full proof can be found in Appendix D. In short, we use syntactic induction. For (sub)sentences, the current teams are irrelevant and we have $\mathfrak{A}, T_1 \models \varphi$ if and only if $\mathfrak{A} \models \varphi$ if and only if $\mathfrak{A}, T_2 \models \varphi$. For everything else, we mainly have to make sure that we can match every local move (or split) in T_1 by a local move (or split) in T_2 that preserves the preconditions. This is always provided by the local isomorphism. ◀

It is clear that the precision of this invariance is not comparable to results like bisimulation invariance of GF or GTL. Still, it is a strong enough tool so show our intended result by providing an example of an inexpressible property using a specific class of structures.

▶ **Definition 32.** *For $n \geq 3$, let $\mathfrak{B}_n$ consist of an n-cycle and an n-line, i.e. $\mathfrak{B}_n$ is an $\{E\}$-structure with universe $B_n := \{b_1, \dots, b_n, c_1, \dots, c_n\}$ and*

$$E^{\mathfrak{B}_n} = \{(b_i, b_{i+1}) \mid 1 \leq i < n)\} \cup \{(c_i, c_{i+1}) \mid 1 \leq i < n\} \cup \{(c_n, c_1)\}.$$

▶ **Lemma 33.** *Let $\ell \in \mathbb{N}$ and $\varphi \in$ GF(inc) with $\mathfrak{lr}(\varphi) \leq \ell$. Let $\mathfrak{B}_{2\ell+4}$ be as defined in Definition 32 and let $T_C := \{x \mapsto c_{\ell+2}\}$ and $T_L := \{x \mapsto b_{\ell+2}\}$ be two teams with domain $\{x\}$ consisting of just one assignment each. Then*

$$\mathfrak{B}_{2\ell+4}, T_C \models \varphi \qquad \Leftrightarrow \qquad \mathfrak{B}_{2\ell+4}, T_L \models \varphi.$$

Proof. If we can show that the prerequisites of Lemma 31 are fulfilled in both directions, we are done. Obviously, both T_C and T_L consist of only one $(\ell+1)$-local cluster, for which we only have to show that there is an isomorphism $\iota\colon \mathfrak{n}(T_C, \ell+1) \to \mathfrak{n}(T_L, \ell+1)$ so that $\iota(c_{\ell+2}) = b_{\ell+2}$. We notice that $\mathfrak{n}(T_C, \ell+1) = \{c_1, \dots, c_{2\ell+3}\}$ and $\mathfrak{n}(T_L, \ell+1) = \{b_1, \dots, b_{2\ell+3}\}$. It is then straightforward to check that $\iota(c_i) = b_i$ fulfils the requirements. ◀

▶ **Theorem 34.** GHTL$^+ \not\subseteq$ GF(inc).

Proof. In Lemma 33, we saw that there cannot be a formula separating $\mathfrak{B}_n, T_L$ from $\mathfrak{B}_n, T_C$ for all $n \in \mathbb{N}$. However, there is such a formula in GHTL$^+$, namely

$$\varphi(x) := \downarrow_x X(\exists_g yz(Eyz \wedge \downarrow_{yz} Y(\exists_g w(Ezw \wedge Yzw) \wedge \exists_g uv(Euv \wedge Yuv \wedge Xu)))),$$

which is always satisfied by T_C but not by T_L. After binding the starting team to X, we move to a binary team T_{yz} with the property that for each edge in T_{yz}, we can move one step further on the graph and still be in T_{yz}. This can only be the case if T_{yz} consists exactly of all edges of the cycle. We bind this team and then move to another team T_{uv} that may only consist of edges on the cycle where the first node is in X. This can only be possible if the vertex in the original team was on the cycle. ◀

5.2.2 $\nu\mathbf{GF}^+ \not\subseteq \mathbf{GHTL}^+$

This section builds upon the previous one. In particular, we want to provide an inexpressibility result for GHTL^+ similar to Lemma 31. All the observations about local behaviour in guarded logics still apply, but we now have to account for the fact that teams can be bound with $\downarrow$ and then "carried" through global moves. To handle this, we can make use of the fact that these bound teams appear only positively, which means that a subsentence is still satisfied when replacing bound teams by supersets.

▶ **Lemma 35.** *Let $\tau = \sigma \cup \sigma^+$ be a signature and $l \in \mathbb{N}$. Let $\mathfrak{A}$ be a σ-structure and let $\mathfrak{A}_1$, $\mathfrak{A}_2$ be expansions of $\mathfrak{A}$ to τ so that for all $R \in \sigma^+$, we have $R^{\mathfrak{A}_1} \subseteq R^{\mathfrak{A}_2}$. Let $T_1 \subseteq T_2$ be teams in $\mathfrak{A}_2$ such that T_1 is an $(l+1)$-local cluster in T_2 (with respect to $\mathfrak{A}_2$) and for every $(l+1)$-local cluster C_2 in T_2, there is a $(l+1)$-local cluster C_1 in $T_1 \subseteq T_2$ so that $\mathfrak{A}_1, C_1 \simeq^{l+1} \mathfrak{A}_2, C_2$.*
Let $\varphi \in \mathrm{GHTL}^+$ be a formula with locality rank $\mathfrak{lr}(\varphi) \leq l$ such that all relations in σ^+ appear only positively. Then $\mathfrak{A}_1, T_1 \models \varphi$ implies $\mathfrak{A}_2, T_2 \models \varphi$.

▶ Note. Even though local clusters are supposed to capture the intuition of locally isolated partitions, the union of two or more disjoint local clusters is still a local cluster and a union of local isomorphisms is still a local isomorphism.

Proof of Lemma 35. Overall, the proof is very similar to the proof of Lemma 31. Concerning local behaviour, it is identical except for the case $\varphi = {\downarrow_{\overline{x}}} R\psi$. Here we need to show that $(\mathfrak{A}_1, T_1(\overline{x})), T_1 \models \psi$ implies $(\mathfrak{A}_2, T_2(\overline{x})), T_2 \models \psi$. If we can show that $(\mathfrak{A}_1, T_1(\overline{x})), T_1$ and $(\mathfrak{A}_2, T_2(\overline{x})), T_2$ satisfy the preconditions of the lemma, we are done by induction.

- First, we notice that $T_1(\overline{x}) \subseteq T_2(\overline{x})$ because $T_1 \subseteq T_2$.
- T_1 is still an $(l+1)$-local cluster in T_2: else, the neighbourhoods of T_1 and $T_2 \setminus T_1$ in $(\mathfrak{A}_2, T_2(\overline{x}))$ would overlap. This means there is a short path from T_1 to T_2, i.e. a tuple of guarded sets $(G_1, \ldots, G_n)$ so that $n \leq 2l+2$, $G_1 \in T_1(X)$ and $G_n \in T_2(X) \setminus T_1(X)$ with $\mathrm{dom}(T_2) = \mathrm{dom}(T_1) = X$. We can assume that this path is minimal, which in particular means that no G_i is guarded by $T_2(\overline{x})$ for $1 \leq i < n$. But this means this path would already exist in $\mathfrak{A}_2$ without the expansion by $T_2(\overline{x})$, and T_1 would not have been a cluster in T_2 in the first place.
- The same argument can be extended to all other clusters in T_2 and T_1 in the sense that they still are clusters after the expansion, so we can keep the correspondence of clusters between T_1 and T_2.
- For every pair of corresponding cluster $\mathfrak{A}_1, C_1 \simeq^{l+1} \mathfrak{A}_2, C_2$, with local isomorphism (ι, π), we have to show $(\mathfrak{A}_1, T_1(\overline{x})), C_1 \simeq^{l+1} (\mathfrak{A}_2, T_2(\overline{x})), C_2$. For this, it suffices to show that for all tuples $\overline{a}$ with $[\overline{a}] \subseteq \mathfrak{n}(C_1, l+1)$ we have $\overline{a} \in T_1(\overline{x})$ if and only if $\iota(\overline{a}) \in T_2(\overline{x})$. But $\overline{a} \in T_1(\overline{x})$ if and only if there is a $t_1 \in T_1$ with $\overline{a} = t(\overline{x})$, so $\iota(\overline{a}) = \iota(t_1(\overline{x})) = \pi(t_1)(\overline{x}) \in C_2(\overline{x}) \subseteq T_2(\overline{x})$ (the other direction is similar).

As soon as we reach a subsentence ψ, we can ignore the team, which leaves the interpretations of relations in σ^+ as the only remaining difference between $\mathfrak{A}_1$ and $\mathfrak{A}_2$. But these relations only appear positively, and all interpretations of these relations in $\mathfrak{A}_2$ are supersets of the interpretations in $\mathfrak{A}_1$. So any evaluation that satisfies ψ in $\mathfrak{A}_1$ can be applied to $\mathfrak{A}_2$ as well. ◀

▶ **Lemma 36.** *Let $\ell \in \mathbb{N}$ and $\varphi \in \mathrm{GHTL}^+$ with $\mathfrak{tr}(\varphi) \leq \ell$. Let $\mathfrak{B}_{2\ell+4}$ be as defined in Definition 32 and let $T_B \coloneqq \{x \mapsto b_{\ell+2}, x \mapsto c_{\ell+2}\}$ and $T_C \coloneqq \{x \mapsto c_{\ell+2}\}$ be two teams with domain $\{x\}$. Then*

$$\mathfrak{B}_{2\ell+4}, T_C \models \varphi \qquad \Rightarrow \qquad \mathfrak{B}_{2\ell+4}, T_B \models \varphi.$$

Proof. Again, it is enough to show that Lemma 35 is applicable. We assume $\sigma^+ = \emptyset$. We have $T_C \subseteq T_B$ and each assignment in T_B is its own $(\ell+1)$-local cluster, both of which are locally isomorphic to the cluster in T_C. ◀

▶ **Theorem 37.** $\nu\mathrm{GF}^+ \not\subseteq \mathrm{GHTL}^+$.

Proof. Similar to the proof of Theorem 34, we provide a formula $\varphi \in \nu\mathrm{GF}^+$ in the required format (see Proposition 8) that is satisfied by all $(\mathfrak{B}_n, T_C)$ but not by (B_n, T_B):

$$\varphi \coloneqq \forall_g x(Rx \to [\mathrm{gfp}_{S,x}\, \exists_g y(Exy \wedge Sy)]x)$$

is satisfied in $\mathfrak{B}_n$ if and only if all elements in the teams are part of the cycle, so exactly as required. ◀

6 Closing Remarks

We have seen that guarded team logics bring together many desirable properties of guarded logics and team logics. Hybrid team logics in particular not only offer a new perspective on well-known team logics as seen in Section 3, we could also easily show in Corollary 21 that the guarded variants are decidable and have the finite model property. In Section 5 we also showed that positive guarded hybrid team logic provides an intermediate step in the hierarchy of expressiveness between guarded inclusion logic and guarded positive greatest fixpoint logic. An important ingredient for this was the separation of local and global behaviour of guarded formulae, which we used to establish a strict hierarchy through Lemma 31 and Lemma 35.

It remains an open question whether $\mathrm{GF(inc)} \equiv \mathrm{GHTL}^+$ for sentences. Further, the expressiveness of downward closed guarded team logics like guarded dependence logic, was largely set aside for this paper. This also extends to related questions, e.g. which fragments of existential second order logic correspond to guarded independence logic and guarded hybrid team logic, respectively. It might also be worthwhile to revisit basic design questions for guarded team logics such as which notion of guardedness is appropriate in which contexts, or what effect the addition of further propositional connectives would have.

References

1 Hajnal Andréka, István Németi, and Johan van Benthem. Modal languages and bounded fragments of predicate logic. *J. Philos. Log.*, 27(3):217–274, 1998. doi:10.1023/A:1004275029985.

2 Carlos Areces and Balder ten Cate. Hybrid logics. In Patrick Blackburn, J. F. A. K. van Benthem, and Frank Wolter, editors, *Handbook of Modal Logic*, volume 3 of *Studies in logic and practical reasoning*, pages 821–868. North-Holland, 2007. doi:10.1016/s1570-2464(07)80017-6.

3 William Ward Armstrong. Dependency structures of data base relationships. In Jack L. Rosenfeld, editor, *Information Processing, Proceedings of the 6th IFIP Congress 1974, Stockholm, Sweden, August 5-10, 1974*, pages 580–583. North-Holland, 1974.

4 Vince Bárány, Georg Gottlob, and Martin Otto. Querying the guarded fragment. In *Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science, LICS 2010, 11-14 July 2010, Edinburgh, United Kingdom*, pages 1–10. IEEE Computer Society, 2010. doi:10.1109/LICS.2010.26.

5 Vince Bárány, Balder ten Cate, and Luc Segoufin. Guarded negation. In Luca Aceto, Monika Henzinger, and Jirí Sgall, editors, *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part II*, volume 6756 of *Lecture Notes in Computer Science*, pages 356–367. Springer, 2011. doi:10.1007/978-3-642-22012-8_28.

6 Michael Benedikt, Pierre Bourhis, and Michael Vanden Boom. Characterizing definability in decidable fixpoint logics. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 107:1–107:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.107.

7 Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2001. doi:10.1017/CBO9781107050884.

8 Patrick Blackburn, Johan van Benthem, and Frank Wolter, editors. *Handbook of Modal Logic*, volume 3 of *Studies in logic and practical reasoning*. North-Holland, 2007. URL: https://www.sciencedirect.com/bookseries/studies-in-logic-and-practical-reasoning/vol/3/suppl/C.

9 Marco A. Casanova, Ronald Fagin, and Christos H. Papadimitriou. Inclusion dependencies and their interaction with functional dependencies. In Jeffrey D. Ullman and Alfred V. Aho, editors, *Proceedings of the ACM Symposium on Principles of Database Systems, March 29-31, 1982, Los Angeles, California, USA*, pages 171–176. ACM, 1982. doi:10.1145/588111.588141.

10 Marco A. Casanova and Vânia Maria Ponte Vidal. Towards a sound view integration methodology. In Ronald Fagin and Philip A. Bernstein, editors, *Proceedings of the Second ACM SIGACT-SIGMOD Symposium on Principles of Database Systems, March 21-23, 1983, Colony Square Hotel, Atlanta, Georgia, USA*, pages 36–47. ACM, 1983. doi:10.1145/588058.588065.

11 Haim Gaifman. On local and non-local properties. In J. Stern, editor, *Proceedings of the Herbrand Symposium*, volume 107 of *Studies in Logic and the Foundations of Mathematics*, pages 105–135. Elsevier, 1982. doi:10.1016/S0049-237X(08)71879-2.

12 Pietro Galliani. Inclusion and exclusion dependencies in team semantics - on some logics of imperfect information. *Ann. Pure Appl. Log.*, 163(1):68–84, 2012. doi:10.1016/j.apal.2011.08.005.

13 Pietro Galliani and Lauri Hella. Inclusion logic and fixed point logic. In Simona Ronchi Della Rocca, editor, *Computer Science Logic 2013 (CSL 2013), CSL 2013, September 2-5, 2013, Torino, Italy*, volume 23 of *LIPIcs*, pages 281–295. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2013. doi:10.4230/LIPIcs.CSL.2013.281.

14 Dan Geiger, Azaria Paz, and Judea Pearl. Axioms and algorithms for inferences involving probabilistic independence. *Inf. Comput.*, 91(1):128–141, 1991. doi:10.1016/0890-5401(91)90077-F.

15 Valentin Goranko. Temporal logic with reference pointers. In Dov M. Gabbay and Hans Jürgen Ohlbach, editors, *Temporal Logic, First International Conference, ICTL '94, Bonn, Germany, July 11-14, 1994, Proceedings*, volume 827 of *Lecture Notes in Computer Science*, pages 133–148. Springer, 1994. doi:10.1007/BFb0013985.

16 Erich Grädel. On the restraining power of guards. *J. Symb. Log.*, 64(4):1719–1742, 1999. doi:10.2307/2586808.

17 Erich Grädel, Colin Hirsch, and Martin Otto. Back and forth between guarded and modal logics. *ACM Trans. Comput. Log.*, 3(3):418–463, 2002. doi:10.1145/507382.507388.

18 Erich Grädel and Martin Otto. The freedoms of (guarded) bisimulation. In Alexandru Baltag and Sonja Smets, editors, *Johan van Benthem on Logic and Information Dynamics*, pages 3–31. Springer, 2014. doi:10.1007/978-3-319-06025-5_1.

19 Erich Grädel and Martin Otto. Guarded teams: The horizontally guarded case. In Maribel Fernández and Anca Muscholl, editors, *28th EACSL Annual Conference on Computer Science Logic, CSL 2020, January 13-16, 2020, Barcelona, Spain*, volume 152 of *LIPIcs*, pages 22:1–22:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.CSL.2020.22.

20 Erich Grädel and Jouko A. Väänänen. Dependence and independence. *Stud Logica*, 101(2):399–410, 2013. doi:10.1007/s11225-013-9479-2.

21 Jaakko Hintikka and Gabriel Sandu. Informational independence as a semantical phenomenon. In Jens Erik Fenstad, Ivan T. Frolov, and Risto Hilpinen, editors, *Logic, Methodology and Philosophy of Science VIII*, volume 126 of *Studies in Logic and the Foundations of Mathematics*, pages 571–589. Elsevier, 1989. doi:10.1016/S0049-237X(08)70066-1.

22 W. Hodges. Compositional semantics for a language of imperfect information. *Logic Journal of the IGPL*, 5(4):539–563, 1997. doi:10.1093/jigpal/5.4.539.

23 Neil Immerman. Relational queries computable in polynomial time. *Inf. Control.*, 68(1-3):86–104, 1986. doi:10.1016/S0019-9958(86)80029-8.

24 Juha Kontinen and Ville Nurmi. Team logic and second-order logic. In Hiroakira Ono, Makoto Kanazawa, and Ruy J. G. B. de Queiroz, editors, *Logic, Language, Information and Computation, 16th International Workshop, WoLLIC 2009, Tokyo, Japan, June 21-24, 2009. Proceedings*, volume 5514 of *Lecture Notes in Computer Science*, pages 230–241. Springer, 2009. doi:10.1007/978-3-642-02261-6_19.

25 Juha Kontinen and Jouko A. Väänänen. On definability in dependence logic. *J. Log. Lang. Inf.*, 18(3):317–332, 2009. doi:10.1007/s10849-009-9082-0.

26 Martin Lück. *Team logic: axioms, expressiveness, complexity*. PhD thesis, University of Hanover, Hannover, Germany, 2020. URL: https://www.repo.uni-hannover.de/handle/123456789/9430.

27 Martin Otto. Highly acyclic groups, hypergraph covers, and the guarded fragment. *J. ACM*, 59(1):5:1–5:40, 2012. doi:10.1145/2108242.2108247.

28 Arthur Prior. *Past, Present and Future*. Oxford,: Clarendon P., 1967.

29 Jouko A. Väänänen. *Dependence Logic - A New Approach to Independence Friendly Logic*, volume 70 of *London Mathematical Society student texts*. Cambridge University Press, 2007. URL: http://www.cambridge.org/de/knowledge/isbn/item1164246/?site_locale=de_DE.

A Proposition 13: Negative and Full Hybrid Team Logic

We outline a proof for Proposition 13. In both cases, one part of the equivalence is already shown in Lemma 10. The other part uses syntactic induction, which poses one particular challenge: when we evaluate a formula that contains binders, we may accumulate bound relations over the course of this evaluation. In the proof of Theorem 11, the translation from φ to φ^* "records" each change of the team using fixpoints, which automatically makes the bound teams available if needed. Therefore, we do not have to deal with the afromentioned accumulation in the induction.

In the proofs of Proposition 13 (and also if we wanted to prove $\mathrm{HTL}^+ \subseteq \mathrm{FO(inc)}$ directly), this strategy is not available. Instead, we want to save bound teams over fresh free variables and refer to them using exclusion (or inclusion) atoms. This results in a slightly more general statement.

▶ **Theorem 38.** *Let σ be a signature and Γ be a set of relational variables with $\sigma \cap \Gamma = \emptyset$. Let $(\overline{x}_X)_{X\in\Gamma}$ be a family of pairwise disjoint tuples of variables such that $|\overline{x}_X|$ is equal to the arity of X, and let $\overline{y}$ be a tuple of additional fresh variables.*

For every $\varphi(\overline{x},\Gamma) \in \mathrm{HTL}^-$ so that all $X \in \Gamma$ appear only negatively, there is a $\varphi^+(\overline{x},(\overline{x}_X)_{X\in\Gamma}) \in \mathrm{FO}(\mathrm{exc})$ such that

$$(\mathfrak{A},(X^{\mathfrak{A}})_{X\in\Gamma}),T \models \varphi(\overline{y},\Gamma) \quad \Leftrightarrow \quad \mathfrak{A},T\begin{bmatrix}X^{\mathfrak{A}}\\ \overline{x}_X\end{bmatrix}_{X\in\Gamma} \models \varphi^+(\overline{y},(\overline{x}_X)_{X\in\Gamma}).$$

Proof. Literals in σ, $\wedge$, and quantification are straightforward. If $\varphi = \neg X\overline{y}$ for some $X \in \Gamma$, we use $\varphi^+ = (\overline{x}_X \perp \overline{y})$ as both formulae are satisfied if and only if $T(\overline{y}) \cap X^{\mathfrak{A}} = \emptyset$. This leaves disjunctions and binders. In both cases, we make use of the fact that we may use dependence atoms because $\mathrm{FO}(\mathrm{exc}) \equiv \mathrm{FO}(\mathrm{exc},\mathrm{dep})$.

- If $\varphi = \alpha \vee \beta$, let

$$\begin{aligned}\varphi^+ = \exists zz'ww'(\ &=(\overline{y},zz') \wedge\ =(\overline{y},ww') \wedge (z = z' \vee w = w')\\ &\wedge (z \neq z' \vee (z = z' \wedge \alpha^+)) \wedge (w \neq w' \vee (w = w' \wedge \beta^+))).\end{aligned}$$

  In short, a team S satisfies φ^+ if and only if it can be split into two subteams $S_\alpha \cup S_\beta = S$ so that S_α satisfies α^* and S_β satisfies β^*. In particular, this split only depends on the assignments to $\overline{y}$, so if S has the form $S{\restriction_{\overline{y}}} \times S'$ with $\mathrm{dom}(S') = \mathrm{dom}(S) \setminus [\overline{y}]$, the subteams have the form $S_\alpha{\restriction_{\overline{y}}} \times S'$ and $S_\beta{\restriction_{\overline{y}}} \times S'$.

  Because $\overline{y}$ and all $\overline{x}_X$ are supposed to be disjoint, $T\left[\begin{smallmatrix}X^{\mathfrak{A}}\\ \overline{x}_X\end{smallmatrix}\right]_{X\in\Gamma}$ has this form and we have $\mathfrak{A},T\left[\begin{smallmatrix}X^{\mathfrak{A}}\\ \overline{x}_X\end{smallmatrix}\right]_{X\in\Gamma} \models \varphi^+$ if and only if there is some split $T_\alpha \cup T_\beta = T$ such that $\mathfrak{A},T_\alpha\left[\begin{smallmatrix}X^{\mathfrak{A}}\\ \overline{x}_X\end{smallmatrix}\right]_{X\in\Gamma} \models \alpha^+$ and $\mathfrak{A},T_\beta\left[\begin{smallmatrix}X^{\mathfrak{A}}\\ \overline{x}_X\end{smallmatrix}\right]_{X\in\Gamma} \models \beta^+$. By induction, this is equivalent to $\mathfrak{A},T_\alpha \models \alpha$ and $\mathfrak{A},T_\beta \models \beta$, which is equivalent to $\mathfrak{A},T \models \varphi$ as required.
- If $\varphi = {\downarrow_{\overline{y}'}} Y\psi(\overline{y},\Gamma,Y)$ for some $[\overline{y}'] \subseteq [\overline{y}]$, let

$$\varphi^+ = \forall\overline{x}_Y\exists zz'(\ =(\overline{x}_Y,zz') \wedge ((z = z' \wedge \psi^+(\overline{y},(\overline{x}_X)_{X\in\Gamma},\overline{x}_Y)) \vee (z \neq z' \wedge (\overline{x}_Y|\overline{y}'))))).$$

  Similar to the above case, a team S satisfies φ^+ if and only if there is a superset S' of $S(\overline{y}')$ so that $S\left[\begin{smallmatrix}S'\\ \overline{x}_Y\end{smallmatrix}\right]$ satisfies ψ^+. (†)

  Applying (†) in one direction, if $\mathfrak{A},T\left[\begin{smallmatrix}X^{\mathfrak{A}}\\ \overline{x}_X\end{smallmatrix}\right]_{X\in\Gamma} \models \varphi^+$, there is a superset T' of $T(\overline{y}')$ so that $\mathfrak{A},T\left[\begin{smallmatrix}X^{\mathfrak{A}}\\ \overline{x}_X\end{smallmatrix}\right]_{X\in\Gamma}\left[\begin{smallmatrix}T'\\ \overline{x}_Y\end{smallmatrix}\right] \models \psi^+$. Because exclusion logic is downward closed, this implies $\mathfrak{A},T\left[\begin{smallmatrix}X^{\mathfrak{A}}\\ \overline{x}_X\end{smallmatrix}\right]_{X\in\Gamma}\left[\begin{smallmatrix}T(\overline{y}')\\ \overline{x}_Y\end{smallmatrix}\right] \models \psi^+$, which by induction is equivalent to $(\mathfrak{A},(X^{\mathfrak{A}})_{X\in\Gamma},T(\overline{y}')),T \models \psi$. Using the definition of binders, we get $(\mathfrak{A},(X^{\mathfrak{A}})_{X\in\Gamma}),T \models \varphi$ as required.

  For the other direction, we can trace the same steps backwards, except for the one using downward closure. Considering (†), we know that $\mathfrak{A},T\left[\begin{smallmatrix}X^{\mathfrak{A}}\\ \overline{x}_X\end{smallmatrix}\right]_{X\in\Gamma}\left[\begin{smallmatrix}T(\overline{y}')\\ \overline{x}_Y\end{smallmatrix}\right] \models \psi^+$ directly implies $\mathfrak{A},T\left[\begin{smallmatrix}X^{\mathfrak{A}}\\ \overline{x}_X\end{smallmatrix}\right]_{X\in\Gamma} \models \varphi^+$ and we are done. ◀

▶ **Theorem 39.** *Let σ be a signature and Γ be a set of relational variables with $\sigma \cap \Gamma = \emptyset$. Let $(\overline{x}_X)_{X\in\Gamma}$ be a family of pairwise disjoint tuples of variables such that $|\overline{x}_X|$ is equal to the arity of X, and let $\overline{y}$ be a tuple of additional fresh variables.*

For every $\varphi(\overline{x},\Gamma) \in \mathrm{HTL}$ there is a $\varphi^{\#}(\overline{x},(\overline{y}_X)_{X\in\Gamma}) \in \mathrm{FO}(\mathrm{inc},\mathrm{exc})$ such that

$$(\mathfrak{A},(X^{\mathfrak{A}})_{X\in\Gamma}),T \models \varphi(\overline{y},\Gamma) \quad \Leftrightarrow \quad \mathfrak{A},T\begin{bmatrix}X^{\mathfrak{A}}\\ \overline{x}_X\end{bmatrix}_{X\in\Gamma} \models \varphi^{\#}(\overline{y},(\overline{x}_X)_{X\in\Gamma}).$$

Proof. Again, ordinary literals, $\wedge$, and quantification is straightforward. Negative literals in Γ and $\vee$ are exactly as in the proof above, which does not rely on any properties that are specific to FO(exc). This leaves binders.

In the proof above, we approximated the bound team with a superset, which was sufficient because of downward closure. Here, we can identify the team directly using both inclusion and exclusion atoms. So if $\varphi = \downarrow_{\overline{y}'} Y\psi(\overline{y}, \Gamma, Y)$ for some $[\overline{y}'] \subseteq [\overline{y}]$, let

$$\varphi^{\#} = \forall \overline{x}_Y \exists z z'(=(\overline{x}_Y, zz') \wedge ((z = z' \wedge (\overline{x}_Y \subseteq \overline{y}') \wedge \psi^{\#}) \vee (z \neq z' \wedge (\overline{x}_Y | \overline{y}')))).$$

A team S satisfies $\varphi^{\#}$ if and only if $S\left[\frac{S(\overline{y}')}{\overline{x}_Y}\right]$ satisfies $\psi^{\#}$. With that, we can adapt the above proof to not require downward closure and are done. ◀

Proposition 13 follows from these theorems for $\Gamma = \emptyset$ and Lemma 10.

B Syntactic Definitions of Guarded Team Logics

Proof that for a given finite signature, every formula in guarded logics is expressible as a formula in non-guarded logics:

▶ **Lemma 40.** *Let σ be a finite signature. Let $\mathcal{L}$ be a first-order team logic and let $\mathcal{GL}$ be the guarded variant in the sense of Definition 16, i.e. quantification is replaced by guarded quantification. Then for all $\varphi \in \mathcal{GL}$ there is a $\varphi^{\sigma} \in \mathcal{L}$ so that for all σ-structures $\mathfrak{A}$ with team T we have*

$$\mathfrak{A}, T \models \varphi \quad \Leftrightarrow \quad \mathfrak{A}, T \models \varphi^{\sigma}.$$

Proof. It suffices to show that quantification is replaceable. For this, we use an auxilliary formula $\mathcal{G}^{\sigma}(\overline{x}) \in \mathrm{FO}$ that is satisfied by a team T if and only if $T(\overline{x})$ is guarded. With this, for all formulae ψ with $\mathrm{free}(\psi) = [\overline{xy}]$, we have

$$\exists_g \overline{x}\psi \equiv \exists \overline{x}(\mathcal{G}^{\sigma}(\overline{xy}) \wedge \psi^{\sigma}) \quad \text{and} \quad \forall_g \overline{x}\psi \equiv \forall \overline{x}(\neg\mathcal{G}^{\sigma}(\overline{xy}) \vee (\mathcal{G}^{\sigma}(\overline{xy}) \wedge \psi^{\sigma}),$$

where $\neg\mathcal{G}^{\sigma}(\overline{xy})$ is the negated formula in negation normal form. As $\mathcal{G}^{\sigma}$ is in FO and therefore flat, any formula with the same purpose for GF can be used (see for example [17, Section 3]). ◀

In particular, this proves that the properties of Proposition 6 transfer to their guarded variants, which we show for union-closed logics as an example.

▶ **Lemma 41.** *Let $\mathcal{L}$ be a union-closed team logic and $\mathcal{GL}$ be the guarded variant of $\mathcal{L}$. Then $\mathcal{GL}$ is also union-closed.*

Proof. Let $\varphi \in \mathcal{GL}$, and let $\mathfrak{A}$ be a σ-structure with teams T_1 and T_2 such that $\mathfrak{A}, T_1 \models \varphi$ and $\mathfrak{A}, T_2 \models \varphi$. By Lemma 40, there exists a $\varphi^{\sigma} \in \mathcal{L}$ that is equivalent to φ on σ-structures, so $\mathfrak{A}, T_1 \models \varphi^{\sigma}$ and $\mathfrak{A}, T_2 \models \varphi^{\sigma}$. The union-closure of $\mathcal{L}$ yields $\mathfrak{A}, T_1 \cup T_2 \models \varphi^{\sigma}$ and therefore $\mathfrak{A}, T_1 \cup T_2 \models \varphi$ as required. ◀

We see that this same strategy can be applied to all properties in Proposition 4.

B.1 A Note on Structures with Infinite Signature

Over infinite signatures, there are formulae in GTL that cannot have an equivalent formula in FO with team semantics, which we prove by giving a counterexample:

Let $\sigma = \{E_1, E_2, \ldots\}$ consist of infinitely many binary relations, and let $\mathfrak{A}$ be a σ-structure with universe $\mathbb{N} \times \{0,1\}$ and $((n,b),(m,c)) \in E_i$ if and only if $n = m = i$ and $b \neq c$, i.e. each relation consist of exactly one (symmetric) edge and each element in the universe has exactly one partner. The sentence $\varphi \coloneqq \forall_g x \exists_g y(x \neq y)$ is satisfied if and only if each element has a partner, so it is satisfied in $\mathfrak{A}$, but not in any reduct of $\mathfrak{A}$. But as we know, any formula $\psi \in \mathrm{FO}$ can only contain a finite set of relations $\sigma_\psi \subset \sigma$, and is therefore satisfied in $\mathfrak{A}$ if and only if it is satisfied in the reduct of $\mathfrak{A}$ to σ_ψ. Therefore, ψ cannot be equivalent to φ.

B.2 The Standard Guarded Fragment with Team Semantics

We take a closer look at GF with team semantics, i.e. FO with team semantics where each quantification in a formula is accompanied by a guard that only consists of one atom (see [1]). We provide an example for a property that is expressible in GTL(inc), but not in GF(inc). In GTL(inc), the sentence $\varphi \coloneqq \exists_g xy((Exy \vee Fxy) \wedge (y \subseteq x))$ is satisfied by a (directed) graph with coloured edges if and only if there is an infinite walk along E- *and* F-edges. This is impossible to express in GF(inc), as can be seen by comparing the following two structures.

Let $\mathfrak{A}$ be a structure with universe $A = \{(n,m) \in \mathbb{N} \times \mathbb{N} \mid m \leq n\}$ and relations $E^{\mathfrak{A}} \coloneqq \{((n,i),(n,i+1)) \mid i \equiv 0 \mod 2\}$ and $F^{\mathfrak{A}} \coloneqq \{((n,i),(n,i+1)) \mid i \equiv 1 \mod 2\}$, i.e. $\mathfrak{A}$ consists of finite paths of increasing length that alternate between E- and F-edges. Let $\mathfrak{B}$ be a structure with universe $B \coloneqq A \uplus \{\infty\} \times \mathbb{N}$ and relations similar to $\mathfrak{A}$ so that we again have an infinite number of alternating paths, but now including a path of infinite length. We have $\mathfrak{B} \models \varphi$, but $\mathfrak{A} \not\models \varphi$ in GTL.

In both $\mathfrak{A}$ and $\mathfrak{B}$, every $E(F)$-edge is disjoint from every other $E(F)$-edge. This means that for every GF-guarded team T in one of the structures, $T \models (\overline{x} \subseteq \overline{y})$ if and only if $T \models \overline{x} = \overline{y}$. Therefore, $\mathfrak{A}$ and $\mathfrak{B}$ can only be distinguished by a sentence in GF(inc) if they can be distinguished by a sentence in GF. Due to flatness, this is only possible if they can be distinguished by a sentence in the standard GF, which is known to be impossible.

C The Proof of Theorem 20 in Detail

C.1 Translations

In Table 1, we provide a detailed account of all translations used in the proof of Theorem 20. For better readability, we use a few abbreviations:

Let $\overline{x}, \overline{y}, \overline{z}$ be tuples. If $[\overline{y}] \subseteq [\overline{x}]$, we use $\alpha(\overline{y}) \subseteq \beta(\overline{x})$ as an abbreviation for $\forall_g \overline{y}(\alpha(\overline{y}) \to \exists_g \overline{z}(\beta(\overline{x})))$ with $[\overline{x}] = [\overline{y}] \uplus [\overline{z}]$. If $[\overline{x}] \subseteq [\overline{y}]$, we use $\alpha(\overline{y}) \subseteq \beta(\overline{x})$ as an abbreviation for $\forall_g \overline{y}(\alpha(\overline{y}) \to \beta(\overline{x}))$. Correspondingly $(\alpha(\overline{y}) = \beta(\overline{x})) \coloneqq (\alpha(\overline{y}) \subseteq \beta(\overline{x})) \wedge (\beta(\overline{x}) \subseteq \alpha(\overline{y}))$.

C.2 Further Details

We provide a proper account of the model checking game that is referenced in the proof.

For all structures $\mathfrak{A}$, team T and $\varphi \in \mathrm{GHTL}$, we can inductively define a model checking game $\mathfrak{G}(\mathfrak{A}, T, \varphi)$ as a two player game with perfect information. Player **II** (*"Verifier"*) wants to show that $\mathfrak{A}, T \models \varphi$, while player **I** (*"Falsifier"*) tries to spoil it for **II**. During the game, both players may move to *positions* $(S, \psi)_{\mathfrak{A}}$ with S being a team in $\mathfrak{A}$ and $\psi \in \mathrm{GHTL}$. The starting position for the game $\mathfrak{G}(\mathfrak{A}, T, \varphi)$ is $(T, \varphi)_{\mathfrak{A}}$.

Table 1 Translations used in the proof of Theorem 20.

$\psi =$	$\varphi^*(\psi) :=$
$\alpha(\overline{x})$ for some literal α in $\sigma \cup \mathcal{X}(\varphi)$	$R_\psi \overline{x} \subseteq \alpha(\overline{x})$
$\psi_1(\overline{y}_1) \wedge \psi_2(\overline{y}_2)$ with $[\overline{x}] = [\overline{y}_1] \cup [\overline{y}_2]$	$(R_\psi \overline{x} = R_{\psi_1} \overline{y}_1) \wedge (R_\psi \overline{x} = R_{\psi_2} \overline{y}_2)$
$\psi_1(\overline{y}_1) \vee \psi_2(\overline{y}_2)$ with $[\overline{x}] = [\overline{y}_1] \cup [\overline{y}_2]$	$(R_\psi \overline{x} \subseteq (R_{\psi_1} \overline{y}_1 \vee R_{\psi_2} \overline{y}_2))$ $\wedge (R_{\psi_1} \overline{y}_1 \subseteq R_\psi \overline{x}) \wedge (R_{\psi_2} \overline{y}_2 \subseteq R_\psi \overline{x})$
$\exists_g \overline{y} \phi(\overline{xy})$ with $[\overline{x}] = \text{free}(\phi) \setminus [\overline{y}]$	$\forall_g \overline{x}(R_\psi \overline{x} \leftrightarrow \exists_g \overline{y} R_\phi \overline{xy})$
$\forall_g \overline{y} \phi(\overline{xy})$ with $[\overline{x}] = \text{free}(\phi) \setminus [\overline{y}]$	$\forall_g \overline{x}(R_\psi \overline{x} \leftrightarrow \forall_g \overline{y} R_\phi \overline{xy})$
$\downarrow_{\overline{x}} X \phi(\overline{xy})$	$(X\overline{x} = R_\psi \overline{xy}) \wedge (R_\psi \overline{xy} = R_\phi \overline{xy})$

In any position $(S, \psi)_\mathfrak{A}$, the rules of the game are as follows:

- If ψ is a literal, **II** wins if $\mathfrak{A}, S \models \psi$, else **I** wins.
- If $\psi = \psi_1 \wedge \psi_2$, then **I** decides whether the game proceeds from $(S, \psi_1)_\mathfrak{A}$ or $(S, \psi_2)_\mathfrak{A}$.
- If $\psi = \psi_1 \vee \psi_2$, then **II** chooses a split $S_1 \cup S_2 = S$ and **I** decides whether the game proceeds from $(S_1, \psi_1)_\mathfrak{A}$ or $(S_2, \psi_2)_\mathfrak{A}$.
- If $\psi = \exists_g \overline{x} \phi$, then **II** chooses a guarded update S' of S and the game proceeds from $(S', \phi)_\mathfrak{A}$.
- If $\psi = \forall_g \overline{x} \phi$, then let S' be the universal guarded update of S and the game proceeds from $(S', \phi)_\mathfrak{A}$.
- If $\psi = \downarrow_{\overline{x}} X \phi$, then the game proceeds from $(S, \phi)_{(\mathfrak{A}, S(\overline{x}))}$.

As we can see, the definition of the game mirrors the semantics of GHTL. As such, it is clear that a winning strategy for **II** in the game $\mathfrak{G}(\mathfrak{A}, T, \varphi)$ is equivalent to $\mathfrak{A}, T \models \varphi$.

In the proof, we use the fact that **II** has a winning strategy in the game $\mathfrak{G}(\mathfrak{A}, T, \varphi(\overline{x}))$ if and only if there is an expansion $\mathfrak{A}^*$ of $\mathfrak{A}$ with $R_\varphi^{\mathfrak{A}^*} = T(\overline{x})$ so that $\mathfrak{A}^* \models \varphi^*$. This strategy is provided by the R_ψ in the sense that each of the translations in Appendix C.1 corresponds to a winning strategy in one of the positions of the game. For example, if **II** has a winning strategy in position $(R_\phi^{\mathfrak{A}^*}, \phi)_\mathfrak{A}$, then $\mathfrak{A}^* \models \forall_g \overline{x}(R_\psi \overline{x} \leftrightarrow \exists_g \overline{y} R_\phi \overline{xy})$ if and only if **II** has a winning strategy in position $(R_\psi^{\mathfrak{A}^*}, \psi)_\mathfrak{A}$ for $\psi = \exists_g \overline{y} \phi(\overline{xy})$.

As already stated at the end of the proof, there are a few technical details that have to be considered:

1. In general, the arity of R_ψ should correspond to the number of free variables in ψ. If ψ is a sentence, then R_ψ would be 0-ary. We circumvent this by choosing R_ψ to be unary. This way, the winning strategy should include the position $(\emptyset, \psi)_\mathfrak{A}$ if $R_\psi^{\mathfrak{A}^*}$ is empty, and $(\{\emptyset\}, \psi)_\mathfrak{A}$ else. In the latter case, the specific interpretation of R_ψ does not matter, as it only serves as a distinction between the empty assignment $\{\emptyset\}$ and the empty team.
2. This also requires a modification of the translation in the case that ψ is a sentence that starts with a quantification.
   If $\psi = \exists_g \overline{x} \phi(\overline{x})$, then $\varphi^*(\psi) = \exists_g \overline{y} R_\psi \overline{y} \to \exists_g \overline{x} R_\phi \overline{x}$.
   If $\psi = \forall_g \overline{x} \phi(\overline{x})$, then $\varphi^*(\psi) = \exists_g \overline{y} R_\psi \overline{y} \to \forall_g \overline{x} R_\phi \overline{x}$.
   This prevents false positives by making sure that the interpretation of R_ϕ is non-empty if the interpretation of R_ψ is non-empty.

D Proof of Lemma 31

Recall Lemma 31:

Let $\mathfrak{A}$ be a σ-structure and $l \in \mathbb{N}$. Let T_1 and T_2 be teams in $\mathfrak{A}$ such that for every $(l+1)$-local cluster C_2 in T_2, there is a $(l+1)$-local cluster C_1 in T_1 such that $\mathfrak{A}, C_1 \simeq^{l+1} \mathfrak{A}, C_2$. Let $\varphi \in \mathrm{GF}(\mathrm{inc})$ be a formula with locality rank $\mathfrak{lr}(\varphi) \leq l$. Then $\mathfrak{A}, T_1 \models \varphi$ implies $\mathfrak{A}, T_2 \models \varphi$.

Proof. We use syntactic induction.

- For classical literals, 1-local isomorphisms imply atomic equivalence.
- Inclusion atoms are also handled by 1-local isomorphism: we assume $\mathfrak{A}, T_1 \models (\overline{x} \subseteq \overline{y})$ and $t \in T_2$. Then there is a 1-local cluster C_2 in T_2 containing t and a corresponding 1-local cluster C_1 in T_1 with bijection π from C_1 to C_2. We then find some $t' \in T_1$ so that $t'(\overline{y}) = \pi^{-1}(t)(\overline{x})$ by assumption. Because C_1 is a 1-local cluster, we also have $t' \in C_1$ and by commutativity of the maps this yields $\pi(t')(\overline{y}) = t(\overline{x})$.
- Conjunctions are straightforward. For disjunctions, we show that every split $T_1^1 \cup T_1^2 = T_1$ can be used to find a split $T_2^1 \cup T_2^2 = T_2$ that preserves the preconditions of the lemma. For this, let C_2 be a cluster in T_2 and C_1 be the corresponding cluster in T_1 with local isomorphism (ι, π). Clearly, $C_1^1 = C_1 \cap T_1^1$ and $C_1^2 = C_1 \cap T_1^2$ form a split of C_1, and thus $C_2^1 = \pi(C_1^1)$ and $C_2^2 = \pi(C_1^2)$ form a split of C_2. Let T_2^1 be the union of all C_2^1 and T_2^2 be the union of all C_2^2. Then $T_2^1 \cup T_2^2 = T_2$. For all $i, j \in \{1, 2\}$, C_i^j is a cluster in T_i^j and $\mathfrak{A}, C_1^j \simeq^{l+1} \mathfrak{A}, C_2^j$. By construction, we find a corresponding C_1^j for all C_2^j and are done.
- For local universal quantification, the guarded universal update of each $(l+1)$-local cluster is still a l-local cluster, and the guarded universal update of two $(l+1)$-locally isomorphic clusters are still l-locally isomorphic.
- For local existential quantification, we can argue similarly to the local universal case. For any cluster C_2 in T_2, there might be several $l+1$-locally isomorphic clusters $C_1^1, C_1^2, \ldots$ in T_1 which might not be l-locally isomorphic in the update of T_1 anymore. However, we just have to update C_2 in a way so that there is *some* corresponding cluster in the update of T_1, for which we can choose any of the possible candidates, e.g. C_1^1, and update C_2 according to the image of C_1^1 under the local isomorphism.
- For global quantification, when evaluating a (sub)sentence, the current teams are irrelevant and we have $\mathfrak{A}, T_1 \models \varphi$ if and only if $\mathfrak{A} \models \varphi$ if and only if $\mathfrak{A}, T_2 \models \varphi$. ◀