6th Conference on Algebra and Coalgebra in Computer Science

CALCO'15, June 24–26, 2015, Nijmegen, The Netherlands

Edited by

Lawrence S. Moss
Paweł Sobociński

LIPICS

Editors

Lawrence S. Moss
Department of Mathematics
Indiana University
Bloomington, IN 47405 USA
lsm@cs.indiana.edu

Paweł Sobociński
Department of Electronics and Computer Science
University of Southampton
Southampton, United Kingdom SO17 1BJ
ps@ecs.soton.ac.uk

ACM Classification 1998
F. Theory of Computation, F.3 Logics and Meanings of Programs, F.3.2 Semantics of Programming Languages

ISBN 978-3-939897-84-2

Published online and open access by
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at http://www.dagstuhl.de/dagpub/978-3-939897-84-2.

Publication date
October, 2015

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 http://dnb.d-nb.de.

Digital Object Identifier: 10.4230/LIPIcs.CALCO.2015.i

ISBN 978-3-939897-84-2 **ISSN 1868-8969** **http://www.dagstuhl.de/lipics**

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

http://www.dagstuhl.de/lipics

Contents

Regular Papers

6th International Conference on Algebra and Coalgebra in Computer Science (CALCO'15).
Editors: Lawrence S. Moss and Pawel Sobocinski
LIPICS Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Preface

This volume contains the Proceedings of the 6th Conference on Algebra and Coalgebra in Computer Science, CALCO'15, held in Nijmegen, the Netherlands, 24–26 June 2015. Previous CALCO conferences have been held in Swansea (Wales, 2005), Bergen (Norway, 2007), Udine (Italy, 2009), Winchester (UK, 2011) and Warsaw (Poland, 2013). CALCO is a high-level, bi-annual conference formed by joining the forces and reputations of CMCS (the International Workshop on Coalgebraic Methods in Computer Science), and WADT (the Workshop on Algebraic Development Techniques). CALCO received 47 submissions. Of these, 21 papers were accepted.

The invited speakers at CALCO were Chris Heunen, Matteo Mio, Daniela Petrisan, and Andy Pitts. We thank them for their stimulating talks. In addition, CALCO has a tradition of Early Ideas talks, allowing presentation of work in progress and original research proposals. CALCO'15 had 11 Early Ideas talks.

We are grateful to the programme committee of CALCO for their hard work:

Andrej Bauer
Filippo Bonchi
Corina Cîrstea
Andrea Corradini
Ross Duncan
Martín Escardó
Dan Ghica
Helle Hansen
Ichiro Hasuo
Bart Jacobs
Bartek Klin
Barbara König
Dexter Kozen
Alexander Kurz
Paul-André Melliés
Stefan Milus
Lawrence S. Moss (co-chair)
Dusko Pavlovic
Daniela Petrisan
Damien Pous
John Power
Jan Rutten
Lutz Schröder
Monika Seisenberger
Alexandra Silva
Paweł Sobociński (co-chair)
Ana Sokolova
Andrzej Tarlecki

We also are pleased to thank the local organizers for their work in bringing such a wonderful conference to fruition: Alexandra Silva, Bart Jacobs, Nicole Messink, and Sam Staton.

Larry Moss and Paweł Sobociński
October 2015

6th International Conference on Algebra and Coalgebra in Computer Science (CALCO'15).
Editors: Lawrence S. Moss and Pawel Sobocinski
LIPICS Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

List of Authors

Jiří Adámek
Institut für Theoretische Informatik
Technische Universität Braunschweig
Germany
j.adamek@tu-bs.de

Adriana Balan
University Politehnica of Bucharest
Romania
adriana.balan@mathem.pub.ro

Paolo Baldan
Dipartimento di Matematica
Universitá di Padova
Italy baldan@math.unipd.it

Filippo Bonchi
CNRS, ENS Lyon, Université de Lyon
France
filippo.bonchi@ens-lyon.fr

Liang-Ting Chen
Institut für Theoretische Informatik
Technische Universität Braunschweig
Germany
l.chen@iti.cs.tu-bs.de

Corina Cîrstea
ECS, University of Southampton
United Kingdom
cc2@ecs.soton.ac.uk

Bob Coecke
University of Oxford
United Kingdom
coecke@cs.ox.ac.uk

Clovis Eberhart
Université Savoie Mont-Blanc
France
ceberhar@ens-cachan.fr

Uli Fahrenberg
IRISA/INRIA, Rennes
France
ulrich.fahrenberg@irisa.fr France

Jeremy Gibbons
University of Oxford
United Kingdom
Jeremy.Gibbons@cs.ox.ac.uk

Ichiro Hasuo
Department of Computer Science
University of Tokyo
Japan
ichiro@is.s.u-tokyo.ac.jp

Tom Hirschowitz
CNRS, Université Savoie Mont-Blanc
France
tom.hirschowitz@univ-savoie.fr

Bart Jacobs
ICIS, Radboud University Nijmegen
The Netherlands
bart@cs.ru.nl

Dimitri Kartsaklis
Queen Mary University of London
United Kingdom
d.kartsaklis@qmul.ac.uk

Toshiki Kataoka
University of Tokyo
Japan
toshikik@is.s.u-tokyo.ac.jp

Shin-Ya Katsumata
Kyoto University
Japan
sinya@kurims.kyoto-u.ac.jp

Henning Kerstan
Universität Duisburg-Essen
Germany
henning.kerstan@uni-due.de

Aleks Kissinger
Oxford University
United Kingdom
aleks.kissinger@cs.ox.ac.uk

Bartek Klin
University of Warsaw
Poland
klin@mimuw.edu.pl

6th International Conference on Algebra and Coalgebra in Computer Science (CALCO'15).
Editors: Lawrence S. Moss and Pawel Sobocinski
LIPICS Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Barbara König
Universität Duisburg-Essen
Germany
barbara_koenig@uni-due.de

Alexander Kurz
University of Leicester
United Kingdom
kurz@mcs.le.ac.uk

Axel Legay
IRISA/INRIA, Rennes
France
Axel.Legay@inria.fr

Paul Blain Levy
University of Birmingham
United Kingdom
p.b.levy@cs.bham.ac.uk

Stefan Milius
FAU Erlangen
Germany
mail@stefan-milius.eu

Beata Nachyła
Institute of Computer Science
Polish Academy of Science
Poland
beatanachyla@gmail.com

Alberto Pardo
Universidad de la República
Uruguay
pardo@fing.edu.uy

Dirk Pattinson
Australian National University
Australia
dirk.pattinson@anu.edu.au

Dusko Pavlovic
University of Hawaii
United States of America
dusko@hawaii.edu

Daniela Petrisan
Radboud University
The Netherlands
daniela.petrisan@gmail.com

Robin Piedeleu
University of Oxford
United Kingdom
robin.piedeleu@cs.ox.ac.uk

Maciej Piróg
University of Oxford
United Kingdom
maciej.adam.pirog@gmail.com

David Quick
Oxford University
United Kingdom
david.quick@cs.ox.ac.uk

Mehrnoosh Sadrzadeh
Queen Mary University of London
United Kingdom
mehrnoosh.sadrzadeh@qmul.ac.uk

Lutz Schröder
Department of Computer Science
FAU Erlangen-Nürnberg
Germany
lutz.schroeder@fau.de

Fatemeh Seifan
ILLC, University of Amsterdam
The Netherlands
fateme.sayfan@gmail.com

Thomas Seiller
PPS, Université Paris VII
France
seiller@ihes.fr

Paula Severi
University of Leicester
United Kingdom
ps330@leicester.ac.uk

Ionuţ Ţuţu
Royal Holloway University of London
United Kingdom
ittutu@gmail.com

Natsuki Urabe
University of Tokyo
Japan
urabenatsuki@is.s.u-tokyo.ac.jp

Henning Urbat
Institut für Theoretische Informatik
Technische Universität Braunschweig
Germany
urbat@iti.cs.tu-bs.de

Jiří Velebil
Czech Technical University
Czech Republic
velebil@math.feld.cvut.cz

Fer-Jan de Vries
University of Leicester
United Kingdom
fdv1@mcs.le.ac.uk

Thorsten Wißmann
Universität Erlangen-Nürnberg
Germany
thorsten.wissmann@fau.de

Nicolas Wu
University of Bristol
United Kingdom
nicolas.wu@bristol.ac.uk

Syntactic Monoids in a Category

Jiří Adámek[1], Stefan Milius[*,2], and Henning Urbat[1]

1 **Institut für Theoretische Informatik**
Technische Universität Braunschweig, Germany
2 **Lehrstuhl für Theoretische Informatik**
Friedrich-Alexander Universität Erlangen-Nürnberg, Germany

Abstract

The syntactic monoid of a language is generalized to the level of a symmetric monoidal closed category $\mathcal{D}$. This allows for a uniform treatment of several notions of syntactic algebras known in the literature, including the syntactic monoids of Rabin and Scott ($\mathcal{D}$ = sets), the syntactic semirings of Polák ($\mathcal{D}$ = semilattices), and the syntactic associative algebras of Reutenauer ($\mathcal{D}$ = vector spaces). Assuming that $\mathcal{D}$ is a commutative variety of algebras, we prove that the syntactic $\mathcal{D}$-monoid of a language L can be constructed as a quotient of a free $\mathcal{D}$-monoid modulo the syntactic congruence of L, and that it is isomorphic to the transition $\mathcal{D}$-monoid of the minimal automaton for L in $\mathcal{D}$. Furthermore, in the case where the variety $\mathcal{D}$ is locally finite, we characterize the regular languages as precisely the languages with finite syntactic $\mathcal{D}$-monoids.

1998 ACM Subject Classification F.4.3 Formal Languages

Keywords and phrases Syntactic monoid, transition monoid, algebraic automata theory, duality, coalgebra, algebra, symmetric monoidal closed category, commutative variety

Digital Object Identifier 10.4230/LIPIcs.CALCO.2015.1

1 Introduction

One of the successes of the theory of coalgebras is that ideas from automata theory can be developed at a level of abstraction where they apply uniformly to many different types of systems. In fact, classical deterministic automata are a standard example of coalgebras for an endofunctor. And that automata theory can be studied with coalgebraic methods rests on the observation that formal languages form the final coalgebra.

The present paper contributes to a new category-theoretic view of *algebraic* automata theory. In this theory one starts with an elegant machine-independent notion of language recognition: a language $L \subseteq X^*$ is recognized by a monoid morphism $e : X^* \to M$ if it is the preimage under e of some subset of M. Regular languages are then characterized as precisely the languages recognized by finite monoids. A key concept, introduced by Rabin and Scott [19] (and earlier in unpublished work of Myhill), is the *syntactic monoid* of a language L. It serves as a canonical algebraic recognizer of L, namely the smallest X-generated monoid recognizing L. Two standard ways to construct the syntactic monoid are:

1. as a quotient of the free monoid X^* modulo the *syntactic congruence* of L, which is a two-sided version of the well-known Myhill-Nerode equivalence, and
2. as the *transition monoid* of the minimal automaton for L.

[*] Stefan Milius acknowledges support by the Deutsche Forschungsgemeinschaft (DFG) under project MI 717/5-1.

6th International Conference on Algebra and Coalgebra in Computer Science (CALCO'15).
Editors: Lawrence S. Moss and Paweł Sobociński; pp. 1–16
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In addition to syntactic monoids there are several related notions of syntactic algebras for (weighted) languages in the literature, most prominently the syntactic idempotent semirings of Polák [18] and the syntactic associative algebras of Reutenauer [20], both of which admit constructions similar to (1) and (2). A crucial observation is that monoids, idempotent semirings and associative algebras are precisely the monoid objects in the categories of sets, semilattices and vector spaces, respectively. Moreover, these three categories are symmetric monoidal closed w.r.t. their usual tensor product.

The main goal of our paper is thus to develop a theory of algebraic recognition in a general symmetric monoidal closed category $\mathcal{D} = (\mathcal{D}, \otimes, I)$. Following Goguen [12], a *language* in $\mathcal{D}$ is a morphism $L : X^{\circledast} \to Y$ where X is a fixed object of inputs, Y is a fixed object of outputs, and $X^{\circledast}$ denotes the free $\mathcal{D}$-monoid on X. And a $\mathcal{D}$*-automaton* is given by the picture below: it consists of an object of states Q, a morphism i representing the initial state, an output morphism f, and a transition morphism δ which may be presented in its curried form $\lambda\delta$.

$$\begin{tikzcd}
& X \otimes Q \arrow[d, "\delta"] & \\
I \arrow[r, "i"] & Q \arrow[r, "f"] \arrow[d, "\lambda\delta"] & Y \\
& {[X, Q]} &
\end{tikzcd} \tag{1}$$

This means that an automaton is at the same time an *algebra* $I + X \otimes Q \xrightarrow{[i,\delta]} Q$ for the functor $FQ = I + X \otimes Q$, and a *coalgebra* $Q \xrightarrow{\langle f,\lambda\delta \rangle} Y \times [X, Q]$ for the functor $TQ = Y \times [X, Q]$. It turns out that much of the classical (co-)algebraic theory of automata in the category of sets extends to this level of generality. Thus Goguen [12] demonstrated that the initial algebra for F coincides with the free $\mathcal{D}$-monoid $X^{\circledast}$, and that every language is accepted by a unique minimal $\mathcal{D}$-automaton. We will add to this picture the observation that the final coalgebra for T is carried by the object of languages $[X^{\circledast}, Y]$, see Proposition 2.21.

In Section 3 we introduce the central concept of our paper, the *syntactic* $\mathcal{D}$*-monoid* of a language $L : X^{\circledast} \to Y$, which by definition is the smallest X-generated $\mathcal{D}$-monoid recognizing L. Assuming that $\mathcal{D}$ is a commutative variety of algebras, we will show that the above constructions (1) and (2) for the classical syntactic monoid adapt to our general setting: the syntactic $\mathcal{D}$-monoid is (1) the quotient of $X^{\circledast}$ modulo the syntactic congruence of L (Theorem 3.14), and (2) the transition $\mathcal{D}$-monoid of the minimal $\mathcal{D}$-automaton for L (Theorem 4.6). As special instances we recover the syntactic monoids of Rabin and Scott ($\mathcal{D}$ = sets), the syntactic semirings of Polák ($\mathcal{D}$ = semilattices) and the syntactic associative algebras of Reutenauer ($\mathcal{D}$ = vector spaces). Furthermore, our categorical setting yields new types of syntactic algebras "for free". For example, we will identify monoids with zero as the algebraic structures representing partial automata (the case $\mathcal{D}$ = pointed sets), which leads to the *syntactic monoid with zero* for a given language. Similarly, by taking as $\mathcal{D}$ the variety of algebras with an involutive unary operation we obtain *syntactic involution monoids.*

Most of the results of our paper apply to arbitrary languages. In Section 5 we will investigate $\mathcal{D}$*-regular languages*, that is, languages accepted by $\mathcal{D}$-automata with a finitely presentable object of states. Under suitable assumptions on $\mathcal{D}$, we will prove that a language is $\mathcal{D}$-regular iff its syntactic $\mathcal{D}$-monoid is carried by a finitely presentable object (Theorem 5.4). We will also derive a dual characterization of the syntactic $\mathcal{D}$-monoid which is new even in the "classical" case $\mathcal{D}$ = sets: if $\mathcal{D}$ is a locally finite variety, and if moreover some other locally finite variety $\mathcal{C}$ is dual to $\mathcal{D}$ on the level of finite objects, the syntactic $\mathcal{D}$-monoid of L dualizes to the local variety of languages in $\mathcal{C}$ generated by the reversed language of L.

Due to space limitations most proofs are omitted or sketched. See [1] for an extended version of this paper.

Related work. Our paper gives a uniform treatment of various notions of syntactic algebras known in the literature [18, 19, 20]. Another categorical approach to (classical) syntactic monoids appears in the work of Ballester-Bolinches, Cosme-Llopez and Rutten [5]. These authors consider automata in the category of sets specified by *equations* or dually by *coequations*, which leads to a construction of the automaton underlying the syntactic monoid of a language. The fact that it forms the transition monoid of a minimal automaton is also interpreted in that setting. In the present paper we take a more general and conceptual approach by studying algebraic recognition in a symmetric monoidal closed category $\mathcal{D}$. One important source of inspiration for our categorical setting was the work of Goguen [12].

In the recent papers [2, 4] we presented a categorical view of *varieties of languages*, another central topic of algebraic automata theory. Building on the duality-based approach of Gehrke, Grigorieff and Pin [11], we generalized Eilenberg's variety theorem and its local version to the level of an abstract (pre-)duality between algebraic categories. The idea to replace monoids by monoid objects in a commutative variety $\mathcal{D}$ originates in this work.

When revising this paper we were made aware of the ongoing work of Bojanczyk [8]. He considers, in lieu of commutative varieties, categories of Eilenberg-Moore algebras for an arbitrary monad on sorted sets, and defines syntactic congruences in this more general setting. Our Theorem 3.14 is a special case of [8, Theorem 3.1].

2 Preliminaries

Throughout this paper we work with deterministic automata in a commutative variety $\mathcal{D}$ of algebras. Recall that a *variety of algebras* is an equational class of algebras over a finitary signature. It is called *commutative* (or *entropic*) if, for any two objects A and B of $\mathcal{D}$, the set $\mathcal{D}(A,B)$ of all homomorphisms from A to B carries a subobject $[A,B] \rightarrowtail B^{|A|}$ of the product of $|A|$ copies of B. Commutative varieties are precisely the categories of Eilenberg-Moore algebras for a commutative finitary monad on the category of sets, see [13, 15]. We fix an object X (of inputs) and an object Y (of outputs) in $\mathcal{D}$.

▶ **Example 2.1.**

1. **Set** is a commutative variety with $[A,B] = B^A$.
2. A *pointed set* $(A, \bot)$ is a set A together with a chosen point $\bot \in A$. The category $\mathbf{Set}_\bot$ of pointed sets and point-preserving functions is a commutative variety. The point of $[(A,\bot_A),(B,\bot_B)]$ is the constant function with value $\bot_B$.
3. An *involution algebra* is a set with an involutive unary operation $x \mapsto \widetilde{x}$, i.e. $\widetilde{\widetilde{x}} = x$. We call $\widetilde{x}$ the *complement* of x. Morphisms are functions f with $f(\widetilde{x}) = \widetilde{f(x)}$. The variety **Inv** of involution algebras is commutative. Indeed, the set $[A,B]$ of all homomorphisms is an involution algebra with pointwise complementation: $\widetilde{f}$ sends x to $\widetilde{f(x)}$.
4. All other examples we treat in our paper are varieties of modules over a semiring. Given a semiring $\mathbb{S}$ (with 0 and 1) we denote by $\mathbf{Mod}(\mathbb{S})$ the category of all $\mathbb{S}$-modules and module homomorphisms (i.e. $\mathbb{S}$-linear maps). Three interesting special cases of $\mathbf{Mod}(\mathbb{S})$ are:
   - **a.** $\mathbb{S} = \{0,1\}$, the boolean semiring with $1+1=1$: the category $\mathbf{JSL}_0$ of join-semilattices with 0, and homomorphisms preserving joins and 0;
   - **b.** $\mathbb{S} = \mathbb{Z}$: the category **Ab** of abelian groups and group homomorphisms;
   - **c.** $\mathbb{S} = \mathbb{K}$ (a field): the category $\mathbf{Vec}(\mathbb{K})$ of vector spaces over $\mathbb{K}$ and linear maps.

▶ **Notation 2.2.** We denote by $\Psi : \mathbf{Set} \to \mathcal{D}$ the left adjoint to the forgetful functor $|{-}| : \mathcal{D} \to \mathbf{Set}$. Thus ΨX_0 is the free object of $\mathcal{D}$ on the set X_0.

▶ **Example 2.3.**

1. We have $\Psi X_0 = X_0$ for $\mathcal{D} = \mathbf{Set}$ and $\Psi X_0 = X_0 + \{\bot\}$ for $\mathcal{D} = \mathbf{Set}_\bot$.
2. For $\mathcal{D} = \mathbf{Inv}$ the free involution algebra on X_0 is $\Psi X_0 = X_0 + \widetilde{X_0}$ where $\widetilde{X_0}$ is a copy of X_0 (whose elements are denoted $\widetilde{x}$ for $x \in X_0$). The involution swaps the copies of X_0, and the universal arrow $X_0 \to X_0 + \widetilde{X_0}$ is the left coproduct injection.
3. For $\mathcal{D} = \mathbf{Mod}(\mathbb{S})$ the free module ΨX_0 is the submodule of $\mathbb{S}^{X_0}$ on all functions $X_0 \to \mathbb{S}$ with finite support. Equivalently, ΨX_0 consists of formal linear combinations $\sum_{i=1}^{n} s_i x_i$ with $s_i \in \mathbb{S}$ and $x_i \in X_0$. In particular, $\Psi X_0 = \mathcal{P}_f X_0$ (finite subsets of X_0) for $\mathcal{D} = \mathbf{JSL}_0$, and ΨX_0 is the vector space with basis X_0 for $\mathcal{D} = \mathbf{Vec}(\mathbb{K})$.

▶ **Definition 2.4.** Given objects A, B and C of $\mathcal{D}$, a *bimorphism* from A, B to C is a function $f : |A| \times |B| \to |C|$ such that the maps $f(a, -) : |B| \to |C|$ and $f(-, b) : |A| \to |C|$ carry morphisms of $\mathcal{D}$ for every $a \in |A|$ and $b \in |B|$. A *tensor product* of A and B is a universal bimorphism $t : |A| \times |B| \to |A \otimes B|$, which means that for every bimorphism $f : |A| \times |B| \to |C|$ there is a unique morphism $f' : A \otimes B \to C$ in $\mathcal{D}$ with $f' \cdot t = f$.

▶ **Theorem 2.5** (Banaschweski and Nelson [6]). *Every commutative variety $\mathcal{D}$ has tensor products, making $\mathcal{D} = (\mathcal{D}, \otimes, I)$ with $I = \Psi 1$ a symmetric monoidal closed category. That is, we have the following bijective correspondence of morphisms, natural in $A, B, C \in \mathcal{D}$:*

$$\begin{array}{rl} f: & A \otimes B \to C \\ \hline \lambda f: & A \to [B, C] \end{array}$$

▶ **Remark 2.6.** Recall that a *monoid* (M, m, i) in a monoidal category $(\mathcal{D}, \otimes, I)$ (with tensor product $\otimes : \mathcal{D} \times \mathcal{D} \to \mathcal{D}$ and tensor unit $I \in \mathcal{D}$) is an object M equipped with a multiplication $m : M \otimes M \to M$ and unit $i : I \to M$ satisfying the usual associative and unit laws. Due to $\otimes$ and $I = \Psi 1$ representing bimorphisms, this categorical definition is equivalent to the following algebraic one in our setting: a *$\mathcal{D}$-monoid* is a triple $(M, \bullet, i)$ where M is an object of $\mathcal{D}$ and $(|M|, \bullet, i)$ is a monoid in $\mathbf{Set}$ with $\bullet : |M| \times |M| \to |M|$ a bimorphism of $\mathcal{D}$. A *morphism* $h : (M, \bullet, i) \to (M', \bullet', i')$ of $\mathcal{D}$-monoids is a morphism $h : M \to M'$ of $\mathcal{D}$ such that $|h| : |M| \to |M'|$ is a monoid morphism in $\mathbf{Set}$. We denote by $\mathbf{Mon}(\mathcal{D})$ the category of $\mathcal{D}$-monoids and their homomorphisms. In the following we will freely work with $\mathcal{D}$-monoids in both categorical and algebraic disguise.

▶ **Example 2.7.**

1. In $\mathbf{Set}$ the tensor product is the cartesian product, $I = \{*\}$, and $\mathbf{Set}$-monoids are ordinary monoids.
2. In $\mathbf{Set}_\bot$ we have $I = \{\bot, *\}$, and the tensor product of pointed sets $(A, \bot_A)$ and $(B, \bot_A)$ is $A \otimes B = (A \backslash \{\bot_A\}) \times (B \backslash \{\bot_B\}) + \{\bot\}$. $\mathbf{Set}_\bot$-monoids are precisely monoids with zero. Indeed, given a $\mathbf{Set}_\bot$-monoid structure on $(A, \bot)$ we have $x \bullet \bot = \bot = \bot \bullet x$ for all x because $\bullet$ is a bimorphism, i.e. $\bot$ is a zero element. Morphisms of $\mathbf{Mon}(\mathbf{Set}_\bot)$ are zero-preserving monoid morphisms.
3. An $\mathbf{Inv}$-monoid (also called an *involution monoid*) is a monoid equipped with an involution $x \mapsto \widetilde{x}$ such that $x \bullet \widetilde{y} = \widetilde{x} \bullet y = \widetilde{x \bullet y}$. For example, for any set A the power set $\mathcal{P}A$ naturally carries the structure of an involution monoid: the involution takes complements, $\widetilde{S} = A \backslash S$, and the monoid multiplication is the symmetric difference $S \oplus T = (S \backslash T) \cup (T \backslash S)$.
4. $\mathbf{JSL}_0$-monoids are precisely idempotent semirings (with 0 and 1). Indeed, a $\mathbf{JSL}_0$-monoid on a semilattice (i.e. a commutative idempotent monoid) $(D, +, 0)$ is given by a unit 1 and a monoid multiplication that, being a bimorphism, distributes over $+$ and 0.

5. More generally, a $\mathbf{Mod}(\mathbb{S})$-monoid is precisely an associative algebra over $\mathbb{S}$: it consists of an $\mathbb{S}$-module together with a unit 1 and a monoid multiplication that distributes over $+$ and 0 and moreover preserves scalar multiplication in both components.

▶ **Notation 2.8.** We denote by $X^{\otimes n}$ ($n < \omega$) the n-fold tensor power of X, recursively defined by $X^{\otimes 0} = I$ and $X^{\otimes(n+1)} = X \otimes X^{\otimes n}$.

▶ **Proposition 2.9** (see Mac Lane [14]). *The forgetful functor $\mathbf{Mon}(\mathcal{D}) \to \mathcal{D}$ has a left adjoint assigning to every object X the free $\mathcal{D}$-monoid $X^{\circledast} = \coprod_{n<\omega} X^{\otimes n}$. The monoid structure $(X^{\circledast}, m_X, i_X)$ is given by the coproduct injection $i_X : I = X^{\otimes 0} \to X^{\circledast}$ and $m_X : X^{\circledast} \otimes X^{\circledast} \to X^{\circledast}$, where $X^{\circledast} \otimes X^{\circledast} = \coprod_{n,k<\omega} X^{\otimes n} \otimes X^{\otimes k}$ and m_X has as its (n,k)-component the $(n+k)$-th coproduct injection. The universal arrow $\eta_X : X \to X^{\circledast}$ is the first coproduct injection.*

▶ **Proposition 2.10.** *The free $\mathcal{D}$-monoid on $X = \Psi X_0$ is $X^{\circledast} = \Psi X_0^*$. Its monoid multiplication extends the concatenation of words in X_0^*, and its unit is the empty word ε.*

▶ **Example 2.11.**
1. In $\mathbf{Set}$ we have $X^{\circledast} = X^*$. In $\mathbf{Set}_\perp$ with $X = \Psi X_0 = X_0 + \{\perp\}$ we get $X^{\circledast} = X_0^* + \{\perp\}$. The product $x \bullet y$ is concatenation for $x, y \in X_0^*$, and otherwise $\perp$.
2. In $\mathbf{Inv}$ with $X = \Psi X_0 = X_0 + \widetilde{X_0}$ we have $X^{\circledast} = X_0^* + \widetilde{X_0^*}$. The multiplication restricted to X_0^* is concatenation, and is otherwise determined by $\widetilde{u} \bullet v = \widetilde{uv} = u \bullet \widetilde{v}$ for $u, v \in X_0^*$.
3. In $\mathbf{JSL}_0$ with $X = \Psi X_0 = \mathcal{P}_f X_0$ we have $X^{\circledast} = \mathcal{P}_f X_0^*$, the semiring of all finite languages over X_0. Its addition is union and its multiplication is the concatentation of languages.
4. More generally, in $\mathbf{Mod}(\mathbb{S})$ with $X = \Psi X_0$ we get $X^{\circledast} = \Psi X_0^* = \mathbb{S}[X_0]$, the module of all finite $\mathbb{S}$-weighted languages over the alphabet X_0. Hence the elements of $\mathbb{S}[X_0]$ are functions $c : X_0^* \to \mathbb{S}$ with finite support, which may be expressed as polynomials $\sum_{i=1}^{n} c(w_i) w_i$ with $w_i \in X_0^*$ and $c(w_i) \in \mathbb{S}$. The $\mathbb{S}$-algebraic structure of $\mathbb{S}[X_0]$ is given by the usual addition, scalar multiplication and product of polynomials.

▶ **Definition 2.12** (Goguen [12]). A $\mathcal{D}$-*automaton* (Q, δ, i, f) consists of an object Q (of states) and morphisms $\delta : X \otimes Q \to Q$, $i : I \to Q$ and $f : Q \to Y$; see Diagram (1). An *automata homomorphism* $h : (Q, \delta, i, f) \to (Q', \delta', i', f')$ is a morphism $h : Q \to Q'$ preserving transitions as well as initial states and outputs, i.e. making the following diagrams commute:

$$\begin{tikzcd}
X \otimes Q \arrow[r, "\delta"] \arrow[d, "X\otimes h"'] & Q \arrow[d, "h"] \\
X \otimes Q' \arrow[r, "\delta'"'] & Q'
\end{tikzcd}
\qquad
\begin{tikzcd}
I \arrow[r, "i"] \arrow[dr, "i'"'] & Q \arrow[r, "f"] \arrow[d, "h"] & Y \\
 & Q' \arrow[ur, "f'"'] &
\end{tikzcd}$$

The above definition makes sense in any monoidal category $\mathcal{D}$. In our setting, since $I = \Psi 1$, the morphism i chooses an *initial state* in $|Q|$. Moreover, if $X = \Psi X_0$ for some set X_0 (of inputs), the morphism δ amounts to a choice of endomorphisms $\delta_a : Q \to Q$ for $a \in X_0$, representing transitions. This follows from the bijections

$$\begin{array}{ll}
\Psi X_0 \otimes Q \to Q & \text{in } \mathcal{D} \\ \hline
\Psi X_0 \to [Q, Q] & \text{in } \mathcal{D} \\ \hline
X_0 \to \mathcal{D}(Q, Q) & \text{in } \mathbf{Set}
\end{array}$$

▶ **Example 2.13.**
1. The classical deterministic automata are the case $\mathcal{D} = \mathbf{Set}$ and $Y = \{0, 1\}$. Here $f : Q \to \{0, 1\}$ defines the set $F = f^{-1}[1] \subseteq Q$ of final states. For general Y we get deterministic Moore automata with outputs in Y.

2. The setting $\mathcal{D} = \mathbf{Set}_\perp$ with $X = X_0 + \{\perp\}$ and $Y = \{\perp, 1\}$ gives *partial* deterministic automata. Indeed, the state object $(Q, \perp)$ has transitions $\delta_a : (Q, \perp) \to (Q, \perp)$ for $a \in X_0$ preserving $\perp$, that is, $\perp$ is a sink state. Equivalently, we may consider δ_a as a partial transition map on the state set $Q \backslash \{\perp\}$. The morphism $f : (Q, \perp) \to \{\perp, 1\}$ again determines a set of final states $F = f^{-1}[1]$ (in particular, $\perp$ is non-final). And the morphism $i : \{\perp, *\} \to (Q, \perp)$ determines a partial initial state: either $i(*)$ lies in $Q \backslash \{\perp\}$, or no initial state is defined.
3. In $\mathcal{D} = \mathbf{Inv}$ let us choose $X = X_0 + \widetilde{X_0}$ and $Y = \{0, 1\}$ with $\widetilde{0} = 1$. An $\mathbf{Inv}$-automaton is a deterministic automaton with complementary states $x \mapsto \widetilde{x}$ such that (i) for every transition $p \xrightarrow{a} q$ there is a complementary transition $\widetilde{p} \xrightarrow{a} \widetilde{q}$ and (ii) a state q is final iff $\widetilde{q}$ is non-final.
4. For $\mathcal{D} = \mathbf{JSL}_0$ with $X = \mathcal{P}_f X_0$ and $Y = \{0, 1\}$ (the two-chain) an automaton consists of a semilattice Q of states, transitions $\delta_a : Q \to Q$ for $a \in X_0$ preserving finite joins (including 0), an initial state $i \in Q$ and a homomorphism $f : Q \to \{0, 1\}$ which defines a *prime upset* $F = f^{-1}[1] \subseteq Q$ of final states. The latter means that a finite join of states is final iff one of the states is. In particular, 0 is non-final.
5. More generally, automata in $\mathcal{D} = \mathbf{Mod}(\mathbb{S})$ with $X = \Psi X_0$ and $Y = \mathbb{S}$ are $\mathbb{S}$-*weighted automata*. Such an automaton consists of an $\mathbb{S}$-module Q of states, linear transitions $\delta_a : Q \to Q$ for $a \in X_0$, an initial state $i \in Q$ and a linear output map $f : Q \to \mathbb{S}$.

▶ **Remark 2.14.**

1. An *algebra* for an endofunctor F of $\mathcal{D}$ is a pair (Q, α) of an object Q and a morphism $\alpha : FQ \to Q$. A *homomorphism* $h : (Q, \alpha) \to (Q', \alpha')$ of F-algebras is a morphism $h : Q \to Q'$ with $h \cdot \alpha = \alpha' \cdot Fh$. Throughout this paper we work with the endofunctor $FQ = I + X \otimes Q$; its algebras are denoted as triples (Q, δ, i) with $\delta : X \otimes Q \to Q$ and $i : I \to Q$. Hence $\mathcal{D}$-automata are precisely F-algebras equipped with an output morphism $f : Q \to Y$. Moreover, automata homomorphisms are precisely F-algebra homomorphisms preserving outputs.
2. Analogously, a *coalgebra* for an endofunctor T of $\mathcal{D}$ is a pair (Q, γ) of an object Q and a morphism $\gamma : Q \to TQ$. Throughout this paper we work with the endofunctor $TQ = Y \times [X, Q]$; its coalgebras are denoted as triples (Q, τ, f) with $\tau : Q \to [X, Q]$ and $f : Q \to Y$. Hence $\mathcal{D}$-automata are precisely *pointed* T-coalgebras, i.e. T-coalgebras equipped with a morphism $i : I \to Q$. Indeed, given a pointed coalgebra $I \xrightarrow{i} Q \xrightarrow{\langle f, \tau \rangle} Y \times [X, Q]$, the morphism $Q \xrightarrow{\tau} [X, Q]$ is the curried form of a morphism $Q \otimes X \xrightarrow{\cong} X \otimes Q \xrightarrow{\delta} Q$. Automata homomorphisms are T-coalgebra homomorphisms preserving initial states.

▶ **Definition 2.15.** Given a $\mathcal{D}$-monoid (M, m, i) and a morphism $e : X \to M$ of $\mathcal{D}$, the *F-algebra associated to M and e* has carrier M and structure

$$[i, \delta] = (I + X \otimes M \xrightarrow{I + e \otimes M} I + M \otimes M \xrightarrow{[i, m]} M).$$

In particular, the F-algebra associated to the free monoid $X^{\circledast}$ (and its universal arrow η_X) is

$$[i_X, \delta_X] = (I + X \otimes X^{\circledast} \xrightarrow{I + \eta_X \otimes X^{\circledast}} I + X^{\circledast} \otimes X^{\circledast} \xrightarrow{[i_X, m_X]} X^{\circledast}).$$

▶ **Example 2.16.** In **Set** every monoid M together with an "input" map $e : X \to M$ determines an F-algebra with initial state i and transitions $\delta_a = - \bullet e(a)$ for all $a \in X$. The F-algebra associated to X^* is the usual automaton of words: its initial state is ε and the transitions are given by $w \xrightarrow{a} wa$ for $a \in X$.

▶ **Proposition 2.17** (Goguen [12]). *For any symmetric monoidal closed category $\mathcal{D}$ with countable coproducts, $X^\circledast$ is the initial algebra for F.*

▶ **Remark 2.18.** Given any F-algebra (Q, δ, i) the unique F-algebra homomorphism $e_Q : X^\circledast \to Q$ is constructed as follows: extend the morphism $\lambda\delta : X \to [Q, Q]$ to a $\mathcal{D}$-monoid morphism $(\lambda\delta)^+ : X^\circledast \to [Q, Q]$. Then

$$e_Q = (X^\circledast \cong X^\circledast \otimes I \xrightarrow{(\lambda\delta)^+ \otimes i} [Q, Q] \otimes Q \xrightarrow{\mathsf{ev}} Q), \tag{2}$$

where ev is the 'evaluation morphism', i.e. the counit of the adjunction $- \otimes Q \dashv [Q, -]$.

▶ **Notation 2.19.** $\delta^\circledast : X^\circledast \otimes Q \to Q$ denotes the uncurried form of $(\lambda\delta)^+ : X^\circledast \to [Q, Q]$.

▶ **Remark 2.20.** Recall from Rutten [21] that the final coalgebra for the functor $TQ = \{0, 1\} \times Q^X$ on **Set** is the coalgebra $\mathcal{P}X^* \cong [X^*, \{0, 1\}]$ of all languages over X. Given any coalgebra Q, the unique coalgebra homomorphism from Q to $\mathcal{P}\Sigma^*$ assigns to every state q the language accepted by q (as an initial state). These observations generalize to our present setting. The object $[X^\circledast, Y]$ of $\mathcal{D}$ carries the following T-coalgebra structure: its transition morphism $\tau_{[X^\circledast,Y]} : [X^\circledast, Y] \to [X, [X^\circledast, Y]]$ is the two-fold curryfication of

$$[X^\circledast, Y] \otimes X \otimes X^\circledast \xrightarrow{[X^\circledast,Y]\otimes\eta_X\otimes X^\circledast} [X^\circledast, Y] \otimes X^\circledast \otimes X^\circledast \xrightarrow{[X^\circledast,Y]\otimes m_X} [X^\circledast, Y] \otimes X^\circledast \xrightarrow{\mathsf{ev}} Y,$$

and its output morphism $f_{[X^\circledast,Y]} : [X^\circledast, Y] \to Y$ is

$$f_{[X^\circledast,Y]} = ([X^\circledast, Y] \cong [X^\circledast, Y] \otimes I \xrightarrow{[X^\circledast,Y]\otimes i_X} [X^\circledast, Y] \otimes X^\circledast \xrightarrow{\mathsf{ev}} Y).$$

▶ **Proposition 2.21.** *$[X^\circledast, Y]$ is the final coalgebra for T.*

Proof sketch. Given any coalgebra (Q, τ, f), let $\delta : X \otimes Q \to Q$ be the uncurried version of $\tau : Q \to [X, Q]$, see Remark 2.14. Then the unique coalgebra homomorphism into $[X^\circledast, Y]$ is $\lambda h : Q \to [X^\circledast, Y]$, where $h = (Q \otimes X^\circledast \cong X^\circledast \otimes Q \xrightarrow{\delta^\circledast} Q \xrightarrow{f} Y)$. ◀

▶ **Definition 2.22** (Goguen [12]). A *language* in $\mathcal{D}$ is a morphism $L : X^\circledast \to Y$.

Note that if $X = \Psi X_0$ (and hence $X^\circledast = \Psi X_0^*$) for some set X_0, one can identify a language $L : X^\circledast = \Psi X_0^* \to Y$ in $\mathcal{D}$ with its adjoint transpose $\widetilde{L} : X_0^* \to |Y|$, via the adjunction $\Psi \dashv |-| : \mathcal{D} \to$ **Set**. In the case where $|Y|$ is a two-element set, $\widetilde{L}$ is the characteristic function of a "classical" language $L_0 \subseteq X_0^*$.

▶ **Example 2.23.**

1. In $\mathcal{D} =$ **Set** (with $X^\circledast = X^*$ and $Y = \{0, 1\}$) one represents $L_0 \subseteq X^*$ by its characteristic function $L : X^* \to \{0, 1\}$.
2. In $\mathcal{D} = \mathbf{Set}_\bot$ (with $X = X_0 + \{\bot\}$, $X^\circledast = X_0^* + \{\bot\}$ and $Y = \{\bot, 1\}$) one represents $L_0 \subseteq X_0^*$ by its extended characteristic function $L : X_0^* + \{\bot\} \to \{\bot, 1\}$ where $L(\bot) = \bot$.
3. In $\mathcal{D} =$ **Inv** (with $X = X_0 + \widetilde{X_0}$, $X^\circledast = X_0^* + \widetilde{X_0^*}$ and $Y = \{0, 1\}$) one represents $L_0 \subseteq X_0^*$ by $L : X_0^* + \widetilde{X_0^*} \to \{0, 1\}$ where $L(w) = 1$ iff $w \in L_0$ and $L(\widetilde{w}) = 1$ iff $w \notin L_0$ for all words $w \in X_0^*$.
4. In $\mathcal{D} = \mathbf{JSL}_0$ (with $X = \mathcal{P}_f X_0$, $X^\circledast = \mathcal{P}_f X_0^*$ and $Y = \{0, 1\}$) one represents $L_0 \subseteq X_0^*$ by $L : \mathcal{P}_f X_0^* \to \{0, 1\}$ where $L(U) = 1$ iff $U \cap L_0 \neq \emptyset$.
5. In $\mathcal{D} = \mathbf{Mod}(\mathbb{S})$ (with $X = \Psi X_0$, $X^\circledast = \mathbb{S}[X_0]$ and $Y = \mathbb{S}$) an $\mathbb{S}$-weighted language $L_0 : X_0^* \to \mathbb{S}$ is represented by its free extension to a module homomorphism

$$L : \mathbb{S}[X_0^*] \to \mathbb{S}, \quad L\left(\sum_{i=1}^{n} c(w_i) w_i\right) = \sum_{i=1}^{n} c(w_i) L_0(w_i).$$

▶ **Definition 2.24** (Goguen [12])**.** The language *accepted* by a $\mathcal{D}$-automaton (Q, δ, i, f) is $L_Q = (X^{\circledast} \xrightarrow{e_Q} Q \xrightarrow{f} Y)$, where e_Q is the F-algebra homomorphism of Remark 2.18.

▶ **Example 2.25.**

1. In $\mathcal{D} = \mathbf{Set}$ with $Y = \{0,1\}$, the homomorphism $e_Q : X^* \to Q$ assigns to every word w the state it computes in Q, i.e. the state the automaton reaches on input w. Thus $L_Q(w) = 1$ iff Q terminates in a final state on input w, which is precisely the standard definition of the accepted language of an automaton. For general Y, the function $L_Q : X^* \to Y$ is the behavior of the Moore automaton Q, i.e. $L_Q(w)$ is the output of the last state in the computation of w.
2. For $\mathcal{D} = \mathbf{Set}_\perp$ with $X = X_0 + \{\perp\}$ and $Y = \{\perp, 1\}$, we have $e_Q : X_0^* + \{\perp\} \to (Q, \perp)$ sending $\perp$ to $\perp$, and sending a word in X_0^* to the state it computes (if any), and to $\perp$ otherwise. Hence $L_Q : X_0^* + \{\perp\} \to \{\perp, 1\}$ defines (via the preimage of 1) the usual language accepted by a partial automaton.
3. In $\mathcal{D} = \mathbf{Inv}$ with $X = X_0 + \widetilde{X_0}$ and $Y = \{0,1\}$, the map $L_Q : X_0^* + \widetilde{X_0^*} \to \{0,1\}$ sends $w \in X_0^*$ to 1 iff w computes a final state, and it sends $\widetilde{w} \in \widetilde{X_0^*}$ to 1 iff w computes a non-final state.
4. In $\mathcal{D} = \mathbf{JSL}_0$ with $X = \mathcal{P}_f X_0$ and $Y = \{0,1\}$, the map $L_Q : \mathcal{P}X_0^* \to \{0,1\}$ assigns to $U \in \mathcal{P}_f X_0^*$ the value 1 iff the computation of at least one word in U ends in a final state.
5. In $\mathcal{D} = \mathbf{Mod}(\mathbb{S})$ with $X = \Psi X_0$ and $Y = \mathbb{S}$, the map $L_Q : \mathbb{S}[X_0^*] \to \mathbb{S}$ assigns to $\sum_{i=1}^n c(w_i) w_i$ the value $\sum_{i=1}^n c(w_i) y_i$, where y_i is the output of the state Q reaches on input w_i. Taking $Q = \mathbb{S}^n$ for some natural number n yields a classical n-state weighted automaton, and in this case one can show that the restriction of L_Q to X_0^* is is the usual language of a weighted automaton.

▶ **Remark 2.26.** By Remark 2.14 every $\mathcal{D}$-automaton (Q, δ, i, f) is an F-algebra as well as a T-coalgebra. Our above definition of L_Q was purely algebraic. The corresponding coalgebraic definition uses the unique coalgebra homomorphism $c_Q : Q \to [X^{\circledast}, Y]$ into the final T-coalgebra and precomposes with $i : I \to Q$ to get a morphism $c_Q \cdot i : I \to [X^{\circledast}, Y]$ (choosing a language, i.e. an element of $[X^{\circledast}, Y]$). Unsurprisingly, the results are equal:

▶ **Proposition 2.27.** *The language $L_Q : X^{\circledast} \to Y$ of an automaton (Q, δ, i, f) is the uncurried form of the morphism $c_Q \cdot i : I \to [X^{\circledast}, Y]$.*

3 Algebraic Recognition and Syntactic $\mathcal{D}$-Monoids

In classical algebraic automata theory one considers recognition of languages by (ordinary) monoids in lieu of automata. One key concept is the *syntactic monoid* which is characterized as the smallest monoid recognizing a given language. There are also related concepts of canonical algebraic recognizers in the literature, e.g. the syntactic idempotent semiring and the syntactic associative algebra. In this section we will give a uniform account of algebraic language recognition in our categorical setting. Our main result is the definition and construction of a minimal algebraic recognizer, the *syntactic $\mathcal{D}$-monoid* of a language.

▶ **Definition 3.1.** A $\mathcal{D}$-monoid morphism $e : X^{\circledast} \to M$ *recognizes* the language $L : X^{\circledast} \to Y$ if there exists a morphism $f : M \to Y$ of $\mathcal{D}$ with $L = f \cdot e$.

▶ **Example 3.2.** We use the notation of Example 2.23.

1. $\mathcal{D} = \mathbf{Set}$ with $X^{\circledast} = X^*$ and $Y = \{0,1\}$: given a monoid M, a function $f : M \to \{0,1\}$ defines a subset $F = f^{-1}[1] \subseteq M$. Hence a monoid morphism $e : X^* \to M$ recognizes

L via f (i.e. $L = f \cdot e$) iff $L_0 = e^{-1}[F]$. This is the classical notion of recognition of a language $L_0 \subseteq X^*$ by a monoid, see e.g. Pin [17].

2. $\mathcal{D} = \mathbf{Set}_\bot$ with $X = X_0 + \{\bot\}$, $X^{\circledast} = X_0^* + \{\bot\}$ and $Y = \{\bot, 1\}$: given a monoid with zero M, a $\mathbf{Set}_\bot$-morphism $f : M \to \{\bot, 1\}$ defines a subset $F = f^{-1}[1]$ of $M \backslash \{0\}$. A zero-preserving monoid morphism $e : X_0^* + \{\bot\} \to M$ recognizes L via f iff $L_0 = e^{-1}[F]$.
3. $\mathcal{D} = \mathbf{Inv}$ with $X = X_0 + \widetilde{X_0}$, $X^{\circledast} = X_0^* + \widetilde{X_0^*}$ and $Y = \{0, 1\}$: for an involution monoid M to give a morphism $f : M \to \{0, 1\}$ means to give a subset $F = f^{-1}[1] \subseteq M$ satisfying $m \in F$ iff $\widetilde{m} \notin F$. Then L is recognized by $e : X_0^* + \widetilde{X_0^*} \to M$ via f iff $L_0 = X_0^* \cap e^{-1}[F]$.
4. $\mathcal{D} = \mathbf{JSL}_0$ with $X = \mathcal{P}_f X_0$, $X^{\circledast} = \mathcal{P}_f X_0^*$ and $Y = \{0, 1\}$: for an idempotent semiring M a morphism $f : M \to Y$ defines a prime upset $F = f^{-1}[1]$, see Example 2.13. Hence L is recognized by a semiring homomorphism $e : \mathcal{P}_f X_0^* \to M$ via f iff $L_0 = X_0^* \cap e^{-1}[F]$. Here we identify X_0^* with the set of all singleton languages $\{w\}$, $w \in X_0^*$. This is the concept of language recognition introduced by Polák [18] (except that he puts $F = f^{-1}[0]$, so 0 and 1 must be swapped, as well as F and $M \backslash F$).
5. $\mathcal{D} = \mathbf{Mod}(\mathbb{S})$ with $X = \Psi X_0$, $X^{\circledast} = \mathbb{S}[X_0]$ and $Y = \mathbb{S}$: given an associative algebra M, the language L is recognized by $e : \mathbb{S}[X_0] \to M$ via $f : M \to \mathbb{S}$ iff $L = f \cdot e$. For the case where the semiring $\mathbb{S}$ is a ring, this notion of recognition is due to Reutenauer [20].

▶ **Remark 3.3.**

1. Since $\mathcal{D}$ and $\mathbf{Mon}(\mathcal{D})$ are varieties, we have the usual factorization system of regular epimorphisms (= surjective homomorphisms) and monomorphisms (= injective homomorphisms). *Quotients* and *subobjects* are understood w.r.t. this system.
2. By an *X-generated $\mathcal{D}$-monoid* we mean a quotient $e : X^{\circledast} \twoheadrightarrow M$ in $\mathbf{Mon}(\mathcal{D})$. For two such quotients $e_i : X^{\circledast} \twoheadrightarrow M_i$, $i = 1, 2$, we say, as usual, that e_1 is *smaller or equal to e_2* (notation: $e_1 \leqslant e_2$) if e_1 factorizes through e_2. Note that if $X = \Psi X_0$, the free $\mathcal{D}$-monoid $X^{\circledast} = \Psi X_0^*$ on X is also the free $\mathcal{D}$-monoid on the set X_0 (w.r.t. the forgetful functor $\mathbf{Mon}(\mathcal{D}) \to \mathbf{Set}$), see Proposition 2.10. In this case, to give a quotient $e : X^{\circledast} \twoheadrightarrow M$ is equivalent to giving a set of generators for the $\mathcal{D}$-monoid M indexed by X_0 – which is why M may also be called an *X_0-generated $\mathcal{D}$-monoid*.
3. Let $e : X^{\circledast} \twoheadrightarrow M$ be an X-generated $\mathcal{D}$-monoid with unit $i : I \to M$ and multiplication $m : M \otimes M \to M$. Recall that $\eta_X : X \to X^{\circledast}$ denotes the universal morphism of the free $\mathcal{D}$-monoid on X and consider the F-algebra associated to M and $X \xrightarrow{\eta_X} X^{\circledast} \xrightarrow{e} M$ (see Definition 2.15). Thus, together with a given $f : M \to Y$ an X-generated $\mathcal{D}$-monoid induces an automaton (M, δ, i, f) called the *derived* automaton.

▶ **Lemma 3.4.** *The language recognized by an X-generated $\mathcal{D}$-monoid $e : X^{\circledast} \twoheadrightarrow M$ via $f : M \to Y$ is the language accepted by its derived automaton.*

We are now ready to give an abstract account of syntactic algebras in our setting. In classical algebraic automata theory the syntactic monoid of a language is characterized as the smallest monoid recognizing that language. We will use this property as our definition of the syntactic $\mathcal{D}$-monoid.

▶ **Definition 3.5.** The *syntactic $\mathcal{D}$-monoid* of language $L : X^{\circledast} \to Y$, denoted by $\mathsf{Syn}(L)$, is the smallest X-generated monoid recognizing L.

In more detail, the syntactic $\mathcal{D}$-monoid is an X-generated $\mathcal{D}$-monoid $e_L : X^{\circledast} \twoheadrightarrow \mathsf{Syn}(L)$ together with a morphism $f_L : \mathsf{Syn}(L) \to Y$ of $\mathcal{D}$ such that (i) e_L recognizes L via f_L, and (ii) for every X-generated $\mathcal{D}$-monoid $e : X^{\circledast} \twoheadrightarrow M$ recognizing L via $f : M \to Y$ we have

$e_L \leqslant e$, that is, the left-hand triangle below commutes for some $\mathcal{D}$-monoid morphism h:

$$\begin{tikzcd}
X^{\circledast} \arrow[r, two heads, "e"] \arrow[dr, two heads, "e_L"'] & M \arrow[r, "f"] \arrow[d, two heads, "h"] & Y \\
& \mathsf{Syn}(L) \arrow[ur, "f_L"'] &
\end{tikzcd}$$

Note that the right-hand triangle also commutes since e is epimorphic and $f \cdot e = L = f_L \cdot e_L$. The universal property determines $\mathsf{Syn}(L)$, e_L and f_L uniquely up to isomorphism. A construction of $\mathsf{Syn}(L)$ is given below (Construction 3.13). We first consider a special case:

▶ **Example 3.6.** In $\mathcal{D} = \mathbf{Set}$ with $Y = \{0, 1\}$, the syntactic monoid of a language $L \subseteq X^*$ can be constructed as the quotient of X^* modulo the *syntactic congruence*, see e.g. [17]:

$$\mathsf{Syn}(L) = X^*/{\sim}, \qquad \text{where } u \sim v \text{ iff for all } x, y \in X^*\text{: } xuy \in L \iff xvy \in L.$$

We aim to generalize this construction to our categorical setting. First note the following

▶ **Lemma 3.7.** *Let $\mathcal{D}$ be any symmetric monoidal closed category with countable coproducts. Then the forgetful functor $\mathbf{Mon}(\mathcal{D}) \to \mathcal{D}$ preserves reflexive coequalizers.*

▶ **Notation 3.8.** Let (M, m, i) be a $\mathcal{D}$-monoid and $x : I \to M$. We write $x \bullet -$ and $- \bullet x$ for the following morphisms, respectively:

$$M \cong I \otimes M \xrightarrow{x \otimes M} M \otimes M \xrightarrow{m} M \qquad \text{and} \qquad M \cong M \otimes I \xrightarrow{M \otimes x} M \otimes M \xrightarrow{m} M.$$

Recall that in our setting, where $\mathcal{D}$ is a commutative variety, we have $I = \Psi 1$ and so the morphism x is the adjoint transpose of an element of M (see Remark 2.6). In the following we shall often write $x \bullet y$, identifying $x, y : I \to M$ with their corresponding elements of M.

▶ **Definition 3.9.** The *syntactic congruence* of a language $L : X^{\circledast} \to Y$ is the following relation on the underlying set of $X^{\circledast}$:

$$E = \{(u, v) \in X^{\circledast} \times X^{\circledast} \mid \forall x, y \in X^{\circledast} : L(x \bullet u \bullet y) = L(x \bullet v \bullet y)\}$$

The projection maps are denoted by $l, r : E \to X^{\circledast}$.

▶ **Lemma 3.10.** *The set E carries a canonical $\mathcal{D}$-algebraic structure making it a $\mathcal{D}$-object.*

Proof sketch. Just observe that $E = \bigcap E_{x,y}$ where for fixed $x, y \in X^{\circledast}$ the object $E_{x,y}$ is the kernel of the $\mathcal{D}$-morphism $X^{\circledast} \xrightarrow{x \bullet -} X^{\circledast} \xrightarrow{- \bullet y} X^{\circledast} \xrightarrow{L} Y$. ◀

That the name syntactic *congruence* makes sense follows from Lemma 3.11 below. First recall that a $\mathcal{D}$-*monoid congruence* on a given $\mathcal{D}$-monoid M is an equivalence relation in $\mathbf{Mon}(\mathcal{D})$, that is, a jointly monic pair $c_1, c_2 : C \to M$ of $\mathcal{D}$-monoid morphisms (equivalently a $\mathcal{D}$-submonoid $\langle c_1, c_2 \rangle : C \rightarrowtail M \times M$) which is reflexive, symmetric and transitive. Congruences on M are ordered as subobjects of $M \times M$, i.e. via inclusion.

▶ **Lemma 3.11.** *E is a $\mathcal{D}$-monoid congruence on $X^{\circledast}$.*

We can give an alternative, more conceptual, description of E:

▶ **Lemma 3.12.** *Let $l_0, r_0 : K \to X^{\circledast}$ be the kernel pair of $L : X^{\circledast} \to Y$ in $\mathcal{D}$. Then $l, r : E \to X^{\circledast}$ is the* largest *$\mathcal{D}$-monoid congruence contained in K.*

▶ **Construction 3.13.** Let $L : X^{\circledast} \to Y$ be a language and $l, r : E \to X^{\circledast}$ its syntactic congruence. We construct the $\mathcal{D}$-monoid $\mathsf{Syn}(L)$ as the coequalizer of l and r in $\mathbf{Mon}(\mathcal{D})$:

$$\begin{tikzcd}
E \arrow[r, shift left, "l"] \arrow[r, shift right, "r"'] & X^{\circledast} \arrow[r, two heads, "e_L"] & \mathsf{Syn}(L).
\end{tikzcd}$$

We need to show that $\mathsf{Syn}(L)$ has the universal property of Definition 3.5, which first requires to define the morphism $f_L : \mathsf{Syn}(L) \to Y$ with $L = f_L \cdot e_L$. To this end consider the diagram below, where l_0, r_0 is the kernel pair of L and m witnesses that E is contained in K, i.e. $l = l_0 \cdot m$ and $r = r_0 \cdot m$ (see Lemma 3.12).

$$\begin{tikzcd}
K \arrow[r, shift left, "l_0"] \arrow[r, shift right, "r_0"'] & X^{\circledast} \arrow[r, "L"] \arrow[dr, two heads, "e_L"'] & Y \\
E \arrow[u, "m"] \arrow[ur, shift left, "l"] \arrow[ur, shift right, "r"'] & & \mathsf{Syn}(L) \arrow[u, dashed, "f_L"']
\end{tikzcd}$$

By Lemma 3.7 the morphism e_L is also a coequalizer of l and r in $\mathcal{D}$. Since $L \cdot l = L \cdot r$ by the above diagram, this yields a unique $f_L : \mathsf{Syn}(L) \to Y$ with $L = f_L \cdot e_L$. In other words, $\mathsf{Syn}(L)$ recognizes L via f_L.

▶ **Theorem 3.14.** *$\mathsf{Syn}(L)$ together with e_L and f_L forms the syntactic $\mathcal{D}$-monoid of L.*

Proof sketch. This follows from the correspondence between kernel pairs and regular quotients: since $l, r : E \to X^{\circledast}$ is the largest congruence contained in the kernel pair of L by Lemma 3.12, the coequalizer e_L of l, r is the smallest quotient of $X^{\circledast}$ recognizing L. ◀

▶ **Remark 3.15.** Our proof of Theorem 3.14 is quite conceptual and works in a general symmetric monoidal closed category $\mathcal{D}$ with enough structure. On this level of generality one would use Lemma 3.12 to *define* the syntactic congruence E as the largest $\mathcal{D}$-monoid congruence contained in the kernel of $L : X^{\circledast} \to Y$. However, it is unclear whether such a congruence exists in this generality and so its existence might have to be taken as an assumption. Hence we restricted ourselves to the setting of a commutative variety $\mathcal{D}$.

▶ **Example 3.16.** Using the notation of Example 2.23 we obtain the following concrete syntactic algebras:

1. In $\mathbf{Set}_\perp$ with $X = X_0 + \{\perp\}$ and $Y = \{\perp, 1\}$ the *syntactic monoid with zero* of a language $L_0 \subseteq X_0^*$ is $(X_0^* + \{\perp\})/{\sim}$ where, for all $u, v \in X_0^* + \{\perp\}$,

   $$u \sim v \quad \text{iff} \quad \text{for all } x, y \in X_0^* : xuy \in L_0 \Leftrightarrow xvy \in L_0.$$

   The zero element is the congruence class of $\perp$.
2. In $\mathbf{Inv}$ with $X = X_0 + \widetilde{X_0}$ and $Y = \{0, 1\}$ the *syntactic involution monoid* of a language $L_0 \subseteq X_0^*$ is the quotient of $X_0 + \widetilde{X_0^*}$ modulo the congruence $\sim$ defined for words $u, v \in X_0^*$ as follows:

   (i) $u \sim v$ iff $\widetilde{u} \sim \widetilde{v}$ iff for all $x, y \in X_0^* : xuy \in L_0 \iff xvy \in L_0$;

   (ii) $u \sim \widetilde{v}$ iff $\widetilde{u} \sim v$ iff for all $x, y \in X_0^* : xuy \in L_0 \iff xvy \notin L_0$.
3. In $\mathbf{Mod}(\mathbb{S})$ with $X = \Psi X_0$ and $Y = \mathbb{S}$ the *syntactic associative $\mathbb{S}$-algebra* of a weighted language $L_0 : X_0^* \to \mathbb{S}$ is the quotient of $\mathbb{S}[X_0]$ modulo the congruence defined for $U, V \in \mathbb{S}[X_0]$ as follows:

   $$U \sim V \quad \text{iff} \quad \text{for all } x, y \in X_0^* : L(xUy) = L(xVy) \tag{3}$$

   Indeed, since $L : \mathbb{S}[X_0] \to \mathbb{S}$ is linear, (3) implies $L(PUQ) = L(PVQ)$ for all $P, Q \in \mathbb{S}[X_0]$, which is the syntactic congruence of Definition 3.9.

4. In particular, for $\mathcal{D} = \mathbf{JSL}_0$ with $X = \mathcal{P}_f X_0$ and $Y = \{0, 1\}$, we get the *syntactic (idempotent) semiring* of a language $L_0 \subseteq X_0^*$ introduced by Polák [18]: it is the quotient $\mathcal{P}_f X_0^*/{\sim}$ where for $U, V \in \mathcal{P}_f X_0^*$ we have

$$U \sim V \quad \text{iff} \quad \text{for all } x, y \in X_0^* : (xUy) \cap L_0 \neq \varnothing \iff xVy \cap L_0 \neq \varnothing.$$

5. For $\mathcal{D} = \mathbf{Vec}(\mathbb{K})$ with $X = \Psi X_0$ and $Y = \mathbb{K}$, the *syntactic* $\mathbb{K}$*-algebra* of a $\mathbb{K}$-weighted language $L_0 : X_0^* \to \mathbb{K}$ is the quotient $\mathbb{K}[X_0]/I$ of the $\mathbb{K}$-algebra of finite weighted languages modulo the ideal

$$I = \{V \in \mathbb{K}[X_0] \mid \text{for all } x, y \in X_0^* : L(xVy) = 0\}.$$

Indeed, the congruence this ideal I generates ($U \sim V$ iff $U - V \in I$) is precisely (3). Syntactic $\mathbb{K}$-algebras were studied by Reutenauer [20].

6. Analogously, for $\mathcal{D} = \mathbf{Ab}$ with $X = \Psi X_0$ and $Y = \mathbb{Z}$, the *syntactic ring* of a $\mathbb{Z}$-weighted language $L_0 : X_0^* \to \mathbb{Z}$ is the quotient $\mathbb{Z}[X_0]/I$, where I is the ideal of all $V \in \mathbb{Z}[X_0]$ with $L(xVy) = 0$ for all $x, y \in X_0^*$.

4 Transition $\mathcal{D}$-Monoids

Here we present another construction of the syntactic $\mathcal{D}$-monoid of a language: it is the transition $\mathcal{D}$-monoid of the minimal $\mathcal{D}$-automaton for this language. Recall that for any object Q of a closed monoidal category $\mathcal{D}$, the object $[Q, Q]$ forms a $\mathcal{D}$-monoid w.r.t. composition.

▶ **Definition 4.1.** The *transition* $\mathcal{D}$*-monoid* $\mathsf{T}(Q)$ of an F-algebra (Q, δ, i) is the image of the $\mathcal{D}$-monoid morphism $(\lambda\delta)^+ : X^\circledast \to [Q, Q]$ extending $\lambda\delta : X \to [Q, Q]$:

$$\begin{tikzcd}
X^{\circledast} \arrow[rr, "(\lambda\delta)^+"] \arrow[dr, two heads, "e_{\mathsf{T}(Q)}"'] & & {[Q,Q]} \\
& \mathsf{T}(Q) \arrow[ur, hook, "m_{\mathsf{T}(Q)}"'] &
\end{tikzcd}$$

▶ **Example 4.2.**

1. In **Set** the transition monoid of an F-algebra Q (i.e. an automaton without final states) is the monoid of all extended transition maps $\delta_w = \delta_{a_n} \cdot \cdots \cdot \delta_{a_1} : Q \to Q$ for $w = a_1 \cdots a_n \in X^*$, with unit $\mathsf{id}_Q = \delta_\varepsilon$ and composition as multiplication.
2. In $\mathbf{Set}_\perp$ with $X = X_0 + \{\perp\}$ (the setting for partial automata) this is completely analogous, except that we add the constant endomap of Q with value $\perp$.
3. In **Inv** with $X = X_0 + \widetilde{X_0}$ we get the involution monoid of all δ_w and $\widetilde{\delta_w}$. Again the unit is δ_ε, and the multiplication is determined by composition plus the equations $x\widetilde{y} = \widetilde{xy} = \widetilde{x}y$.
4. In $\mathbf{JSL}_0$ with $X = \mathcal{P}_f X_0$ the transition semiring consists of all finite joins of extended transitions, i.e. all semilattice homomorphisms of the form $\delta_{w_1} \vee \cdots \vee \delta_{w_n}$ for $\{w_1, \ldots, w_n\} \in \mathcal{P}_f X_0^*$. The transition semiring was introduced by Polák [18].
5. In $\mathbf{Mod}(\mathbb{S})$ with $X = \Psi X_0$ the associative transition algebra consists of all linear maps of the form $\sum_{i=1}^n s_i \delta_{w_i}$ with $s_i \in \mathbb{S}$ and $w_i \in X_0^*$.

Recall from Definition 2.12 that a $\mathcal{D}$-automaton is an F-algebra Q together with an output morphism $f : Q \to Y$. Hence we can speak of the transition $\mathcal{D}$-monoid of a $\mathcal{D}$-automaton.

▶ **Proposition 4.3.** *The language accepted by a* $\mathcal{D}$*-automaton* (Q, δ, f, i) *is recognized by the* $\mathcal{D}$*-monoid morphism* $e_{\mathsf{T}(Q)} : X^\circledast \twoheadrightarrow \mathsf{T}(Q)$.

Proof sketch. The desired morphism $f_{\mathsf{T}(Q)} : \mathsf{T}(Q) \to Y$ with $L_Q = f_{\mathsf{T}(Q)} \cdot e_{\mathsf{T}(Q)}$ is

$$f_{\mathsf{T}(Q)} = (\mathsf{T}(Q) \xrightarrow{m_{\mathsf{T}(Q)}} [Q,Q] \cong [Q,Q] \otimes I \xrightarrow{[Q,Q]\otimes i} [Q,Q] \otimes Q \xrightarrow{\mathsf{ev}} Q \xrightarrow{f} Y). \qquad ◀$$

▶ **Definition 4.4.** A $\mathcal{D}$-automaton (Q, δ, i, f) is called *minimal* iff it is

(a) *reachable*: the unique F-algebra homomorphism $X^{\circledast} \to Q$ is surjective;

(b) *simple*: the unique T-coalgebra homomorphism $Q \to [X^{\circledast}, Y]$ is injective.

▶ **Theorem 4.5** (Goguen [12]). *Every language $L : X^{\circledast} \to Y$ is accepted by a minimal $\mathcal{D}$-automaton* $\mathsf{Min}(L)$*, unique up to isomorphism. Given any reachable automaton Q accepting L, there is a unique surjective automata homomorphism from Q into* $\mathsf{Min}(L)$.

This leads to the announced construction of syntactic $\mathcal{D}$-monoids via transition $\mathcal{D}$-monoids. The case $\mathcal{D} = \mathbf{Set}$ is a standard result of algebraic automata theory (see e.g. Pin [17]), and the case $\mathcal{D} = \mathbf{JSL}_0$ is due to Polák [18].

▶ **Theorem 4.6.** *The syntactic $\mathcal{D}$-monoid of a language $L : X^{\circledast} \to Y$ is isomorphic to the transition $\mathcal{D}$-monoid of its minimal $\mathcal{D}$-automaton:*

$$\mathsf{Syn}(L) \cong \mathsf{T}(\mathsf{Min}(L)).$$

Proof sketch. Using reachability and simplicity of $\mathsf{Min}(L)$, one proves that the quotients $e_L : X^{\circledast} \twoheadrightarrow \mathsf{Syn}(L)$ and $e_{\mathsf{T}(\mathsf{Min}(L))} : X^{\circledast} \twoheadrightarrow \mathsf{T}(\mathsf{Min}(L))$ have the same kernel pair, namely the syntactic congruence of L. This implies the statement of the theorem. ◀

5 $\mathcal{D}$-Regular Languages

Our results so far apply to arbitrary languages in $\mathcal{D}$. In the present section we focus on *regular languages*, which in $\mathcal{D} = \mathbf{Set}$ are the languages accepted by finite automata, or equivalently the languages recognized by finite monoids. For arbitrary $\mathcal{D}$ the role of finite sets is taken over by finitely presentable objects. Recall that an object D of $\mathcal{D}$ is *finitely presentable* if the hom-functor $\mathcal{D}(D, -) : \mathcal{D} \to \mathbf{Set}$ preserves filtered colimits. Equivalently, D is an algebra presentable with finitely many generators and relations.

▶ **Definition 5.1.** A language $L : X^{\circledast} \to Y$ is called *$\mathcal{D}$-regular* if it is accepted by some $\mathcal{D}$-automaton with a finitely presentable object of states.

To work with this definition, we need the following

▶ **Assumptions 5.2.** We assume that the full subcategory $\mathcal{D}_f$ of finitely presentable objects of $\mathcal{D}$ is closed under subobjects, strong quotients and finite products.

▶ **Example 5.3.**

1. Recall that a variety is *locally finite* if all finitely presentable algebras (equivalently all finitely generated free algebras) are finite. Every locally finite variety satisfies the above assumptions. This includes our examples $\mathbf{Set}$, $\mathbf{Set}_\perp$, $\mathbf{Inv}$ and $\mathbf{JSL}_0$.
2. A semiring $\mathbb{S}$ is called *Noetherian* if all submodules of finitely generated $\mathbb{S}$-modules are finitely generated. In this case, as shown in [10], the category $\mathbf{Mod}(\mathbb{S})$ satisfies our assumptions. Every field is Noetherian, as is every finitely generated commutative ring, so $\mathbf{Vec}(\mathbb{K})$ and $\mathbf{Ab} = \mathbf{Mod}(\mathbb{Z})$ are special instances.

▶ **Theorem 5.4.** *For any language $L : X^{\circledast} \to Y$ the following statements are equivalent:*

(a) *L is $\mathcal{D}$-regular.*

(b) *The minimal $\mathcal{D}$-automaton $\mathsf{Min}(L)$ has finitely presentable carrier.*

(c) *L is recognized by some $\mathcal{D}$-monoid with finitely presentable carrier.*

(d) *The syntactic $\mathcal{D}$-monoid $\mathsf{Syn}(L)$ has finitely presentable carrier.*

Proof sketch. This follows immediately from the universal properties of $\mathsf{Syn}(L)$ and $\mathsf{Min}(L)$ and the assumed closure properties of $\mathcal{D}_f$. ◀

Just as the collection of all languages is internalized by the final coalgebra $[X^{\circledast}, Y]$, see Proposition 2.21, we can internalize the regular languages by means of the *rational coalgebra*.

▶ **Definition 5.5.** The *rational coalgebra* ϱT for T is the colimit (taken in the category of T-coalgebras and homomorphisms) of all T-coalgebras with finitely presentable carrier.

▶ **Proposition 5.6.** *There is a one-to-one correspondence between $\mathcal{D}$-regular languages and elements $I \to \varrho T$ of the rational coalgebra.*

We conclude this section with an interesting dual perspective on syntactic monoids, based on our previous work [2, 4]. For lack of space we restrict to the case $\mathcal{D} = \mathbf{Set}$. This category is *predual* to the category $\mathbf{BA}$ of boolean algebras in the sense that the full subcategories of finite sets and finite boolean algebras are dually equivalent. Indeed, this is a restriction of the well-known Stone duality: the dual equivalence functor assigns to a finite boolean algebra B the set $\mathsf{At}(B)$ of its atoms, and to a boolean homomorphism $h : A \to B$ the map $\mathsf{At}(h) : \mathsf{At}(B) \to \mathsf{At}(A)$ sending $b \in \mathsf{At}(B)$ to the unique atom $a \in \mathsf{At}(A)$ with $ha \geqslant b$.

How do the concepts we investigated in $\mathbf{Set}$ – languages, automata and monoids – dualize to $\mathbf{BA}$? Observe that $\mathsf{Reg}(X)$, the boolean algebra of regular languages over the alphabet X, can be viewed as a deterministic automaton: its final states are the regular languages containing the empty word, and the transitions are given by $L \xrightarrow{a} a^{-1}L$ for $a \in X$, where $a^{-1}L = \{w \in X^* : aw \in L\}$ is the *left derivative* of L w.r.t. the letter a. (Similarly, the *right derivative* of L w.r.t. a is $La^{-1} = \{w \in X^* : wa \in L\}$.) This makes $\mathsf{Reg}(X)$ a coalgebra for the endofunctor $\overline{T} = \{0,1\} \times \mathsf{Id}^X$ on $\mathbf{BA}$. Since the two-chain $\{0,1\}$ is dual to the singleton set 1, finite coalgebras for $\overline{T}$ dualize to finite *algebras* for the functor $F = 1 + X \times \mathsf{Id} \cong 1 + \coprod_X \mathsf{Id}$ on $\mathbf{Set}$. Based on this, we proved in [2] that further (i) finite $\overline{T}$-subcoalgebras of $\mathsf{Reg}(X)$ dualize to finite quotient algebras of the initial F-algebra X^*, and (ii) finite *local varieties of languages* (i.e. finite $\overline{T}$-subcoalgebras of $\mathsf{Reg}(X)$ closed under right derivatives) dualize to those F-algebras associated to X-generated monoids, see Definition 2.15. For a regular language $L \subseteq X^*$ the F-algebras associated to the minimal automaton $\mathsf{Min}(L)$ and the syntactic monoid $\mathsf{Syn}(L)$ are finite. Their dual $\overline{T}$-coalgebras are characterized as follows:

▶ **Theorem 5.7.** *Let $L \subseteq X^*$ be a regular language, and L^{rev} its reversed language.*

(a) *$\mathsf{Min}(L)$ is dual to the smallest subcoalgebra of $\mathsf{Reg}(X)$ containing L^{rev}.*

(b) *$\mathsf{Syn}(L)$ is dual to the smallest local variety of languages containing L^{rev}.*

Part (a) of this theorem adds to the recently developed dual view of minimal automata, see [7] and also [16, 3]. All the above considerations generalize from $\mathbf{BA}/\mathbf{Set}$ to arbitrary pairs $\mathcal{C}/\mathcal{D}$ of predual locally finite varieties of algebras. Examples include the self-predual varieties $\mathcal{C} = \mathcal{D} = \mathbf{JSL}_0$ and $\mathcal{C} = \mathcal{D} = \mathbf{Vec}(\mathbb{K})$ for a finite field $\mathbb{K}$.

6 Conclusions and Future Work

We proposed the first steps of a categorical theory of algebraic language recognition. Despite our assumption that $\mathcal{D}$ is a commutative variety, the bulk of our definitions, constructions and proofs works in any symmetric monoidal closed category with enough structure. However, the construction of the syntactic monoid via the syntactic congruence, and the proof that it coincides with a transition monoid, required the concrete algebraic setting. It remains an open problem to develop a genuinely abstract framework for our theory. In particular, such a generalized setting should provide the means for incorporating *ordered* algebras, e.g. the syntactic ordered monoids of Pin [17]. We expect this can be achieved by working with (order-)enriched categories, where the coequalizer in our construction of the syntactic monoid is replaced by a coinserter. A more general theory of recognition might also open the door to treating algebraic recognizers for additional types of behaviors, including Wilke algebras [22] (representing ω-languages) and forest algebras [9] (representing tree and forest languages).

One of the leading themes of algebraic automata theory is the classification of languages in terms of their syntactic algebras. For instance, by Schützenberger's theorem a language is star-free iff its syntactic monoid is aperiodic. We hope that our conceptual view of syntactic monoids (notably their dual characterization in Theorem 5.7) can contribute to a duality-based approach to such results, leading to generalizations and new proof techniques.

References

1 Jiří Adámek, Stefan Milius, , and Henning Urbat. Syntactic monoids in a category. Extended version. http://arxiv.org/abs/1504.02694, 2015.

2 Jiří Adámek, Stefan Milius, Robert S. R. Myers, and Henning Urbat. Generalized Eilenberg Theorem I: Local Varieties of Languages. In Anca Muscholl, editor, *Proc. Foundations of Software Science and Computation Structures (FoSSaCS)*, volume 8412 of *Lecture Notes Comput. Sci.*, pages 366–380. Springer, 2014.

3 Jiří Adámek, Stefan Milius, Robert S. R. Myers, and Henning Urbat. On continuous nondeterminism and state minimality. In Bart Jacobs, Alexandra Silva, and Sam Staton, editors, *Proc. Mathematical Foundations of Programming Science (MFPS XXX)*, volume 308 of *Electron. Notes Theor. Comput. Sci.*, pages 3–23. Elsevier, 2014.

4 Jiří Adámek, Stefan Milius, Robert S. R. Myers, and Henning Urbat. Varieties of Languages in a Category. Accepted for LICS 2015. http://arxiv.org/abs/1501.05180, 2015.

5 A. Ballester-Bolinches, E. Cosme-Llopez, and J.J.M.M. Rutten. The dual equivalence of equations and coequations for automata. Technical report, CWI, 2014.

6 Bernhard Banaschweski and Evelyn Nelson. Tensor products and bimorphisms. *Canad. Math. Bull.*, 19:385–402, 1976.

7 Nick Bezhanishvili, Clemens Kupke, and Prakash Panangaden. Minimization via duality. In Luke Ong and Ruy de Queiroz, editors, *Logic, Language, Information and Computation*, volume 7456 of *Lecture Notes in Computer Science*, pages 191–205. Springer Berlin Heidelberg, 2012.

8 Mikołaj Bojánczyk. Recognisable languages over monads. Preprint: http://arxiv.org/abs/1502.04898, 2015.

9 Mikołaj Bojánczyk and Igor Walukiewicz. Forest algebras. In *Automata and Logic: History and Perspectives*, pages 107–132, 2006.

10 Marcello M. Bonsangue, Stefan Milius, and Alexandra Silva. Sound and complete axiomatizations of coalgebraic language equivalence. *ACM Trans. Comput. Log.*, 14(1:7), 2013.

11 Mai Gehrke, Serge Grigorieff, and Jean-Éric Pin. Duality and equational theory of regular languages. In *Proc. ICALP 2008, Part II*, volume 5126 of *Lecture Notes Comput. Sci.*, pages 246–257. Springer, 2008.

12 Joseph A. Goguen. Discrete-time machines in closed monoidal categories. I. *J. Comput. Syst. Sci.*, 10(1):1–43, 1975.

13 Anders Kock. Monads on symmetric monoidal closed categories. *Arch. Math.*, 21:1–10, 1970.

14 Saunders Mac Lane. *Categories for the working mathematician.* Springer, 2nd edition, 1998.

15 Fred Linton. Autonomous equational categories. *J. Math. Mech.*, 15:637–642, 166.

16 Robert S. R. Myers, Jiří Adámek, Stefan Milius, and Henning Urbat. Canonical nondeterministic automata. In Marcello M. Bonsangue, editor, *Proc. Coalgebraic Methods in Computer Science (CMCS'14)*, volume 8446 of *Lecture Notes Comput. Sci.*, pages 189–210. Springer, 2014.

17 Jean-Éric Pin. Mathematical foundations of automata theory. available at `http://www.liafa.jussieu.fr/~jep/PDF/MPRI/MPRI.pdf`, January 2015.

18 Libor Polák. Syntactic semiring of a language. In Jiří Sgall, Aleš Pultr, and Petr Kolman, editors, *Proc. International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 2136 of *Lecture Notes Comput. Sci.*, pages 611–620. Springer, 2001.

19 Michael O. Rabin and Dana S. Scott. Finite automata and their decision problems. *IBM J. Res. Dev.*, 3(2):114–125, April 1959.

20 Christophe Reutenauer. Séries formelles et algèbres syntactiques. *J. Algebra*, 66:448–483, 1980.

21 Jan J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theoret. Comput. Sci.*, 249(1):3–80, 2000.

22 Thomas Wilke. An Eilenberg Theorem for Infinity-Languages. In *Proc. ICALP 91*, pages 588–599. Springer, 1991.

Extensions of Functors From Set to $\mathscr{V}$-cat[*]

Adriana Balan[1], Alexander Kurz[2], and Jiří Velebil[2]

1 **Department of Mathematical Methods and Models,**
University Politehnica of Bucharest, Romania
adriana.balan@mathem.pub.ro

2 **Department of Computer Science,**
University of Leicester, United Kingdom
kurz@mcs.le.ac.uk

3 **Department of Mathematics, Faculty of Electrical Engineering,**
Czech Technical University in Prague, Czech Republic
velebil@math.feld.cvut.cz

Abstract

We show that for a commutative quantale $\mathscr{V}$ every functor $\mathsf{Set} \longrightarrow \mathscr{V}\text{-}\mathsf{cat}$ has an enriched left-Kan extension. As a consequence, coalgebras over Set are subsumed by coalgebras over $\mathscr{V}$-cat. Moreover, one can build functors on $\mathscr{V}$-cat by equipping Set-functors with a metric.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases enriched category, quantale, final coalgebra

Digital Object Identifier 10.4230/LIPIcs.CALCO.2015.17

1 Introduction

Coalgebras for a functor $T : \mathsf{Set} \longrightarrow \mathsf{Set}$ capture a wide variety of dynamic systems [18]. Moreover, the category $\mathsf{Coalg}(T)$ of coalgebras has a rich structure, which dualizes to some extent the theory of universal algebra. For example, an important role is played by final (or cofree) coalgebras, which give rise to a notion of behavioural equivalence and coinduction. One says that two elements of two coalgebras are behaviourally equivalent (or bisimilar), if they are identified by the morphisms into the final coalgebra. The coinduction principle states that on the final coalgebra two bisimilar elements are equal.

Rutten [17] and Worrell [20, 21] investigate how to account for richer notions of behaviour. For example, we might want to say that one behaviour is smaller than (or, is simulated by) another behaviour. Or we might want to measure distances between behaviours by real numbers. As proposed by Rutten [17], the right framework to develop a theory of metric coalgebras that parallels the theory of coalgebras over Set is given by coalgebras over $\mathscr{V}$-cat, in the sense we are going to explain now.

It was Lawvere [14] who discovered that metric spaces are categories enriched over the category

$$(([0,\infty], \geq_{\mathbb{R}}), +, 0).$$

That an enriched category $\mathscr{X}$ with homs $\mathscr{X}(x, y) \in [0,\infty]$ has identities means $0 = \mathscr{X}(x, x)$ and composition becomes the triangle inequality $\mathscr{X}(x, y) + \mathscr{X}(y, z) \geq_{\mathbb{R}} \mathscr{X}(x, z)$. Thus, enriched categories are nothing but generalized metric spaces, generalized in the sense that distances need not be symmetric and that $\mathscr{X}(x, y) = \mathscr{X}(y, x) = 0$ is not equality but merely

[*] J. Velebil acknowledges the support by grant No. P202/11/1632 of the Czech Science Foundation.

6th International Conference on Algebra and Coalgebra in Computer Science (CALCO'15).
Editors: Lawrence S. Moss and Paweł Sobociński; pp. 17–34
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

an equivalence relation. This interpretation of enriched categories is meaningful not only for $\mathscr{V} = (([0,\infty], \geq_{\mathbb{R}}), +, 0)$, but for any *commutative quantale* $\mathscr{V}$. A category enriched over $\mathscr{V}$ is then called a $\mathscr{V}$-category.

For a detailed discussion of examples showing the relevance of this approach to the denotational semantics of programmming languages we refer to Worrell [21, Chapter 4].

In this paper, we contribute a theorem about the category $\mathscr{V}$-cat of categories enriched over a commutative quantale $\mathscr{V}$. The theorem states that any functor $H : \mathsf{Set} \longrightarrow \mathscr{V}\text{-}\mathsf{cat}$ has an enriched left Kan extension along the 'discrete' functor $D^{\mathscr{V}} : \mathsf{Set} \longrightarrow \mathscr{V}\text{-}\mathsf{cat}$. Moreover, the proof of the theorem shows how to compute the Kan extension $H^\sharp$ on a $\mathscr{V}$-category $\mathscr{X}$ by applying H to the '$\mathscr{V}$-nerve' of $\mathscr{X}$ and then taking an appropriate colimit in $\mathscr{V}$-cat. For example, the extension of $D^{\mathscr{V}}P : \mathsf{Set} \longrightarrow \mathscr{V}\text{-}\mathsf{cat}$, where $P : \mathsf{Set} \longrightarrow \mathsf{Set}$ is the powerset functor, yields the familiar Pompeiu-Hausdorff metric, if the quantale is assumed to be constructively completely distributive.

Apart from allowing us to construct functors on $\mathscr{V}$-cat, the theorem also allows us to establish that for any commutative quantale $\mathscr{V}$ (satisfying some mild properties) the setting of coalgebras enriched over $\mathscr{V}$-cat is indeed richer than the setting of Set-coalgebras in the following sense. For any functor $T : \mathsf{Set} \longrightarrow \mathsf{Set}$ we can define its $\mathscr{V}$-cat-ification $T_{\mathscr{V}}$ to be the left Kan extension of $D^{\mathscr{V}}T$ along $D^{\mathscr{V}}$. Then there is a functor $\widetilde{D}^{\mathscr{V}} : \mathsf{Coalg}(T) \longrightarrow \mathsf{Coalg}(T_{\mathscr{V}})$ which is *right* adjoint and therefore preserves behaviours. In other words, in the world of $\mathscr{V}$-categories all functors $T : \mathsf{Set} \longrightarrow \mathsf{Set}$ are still available via their $\mathscr{V}$-cat-ifications. On the other hand, it happens often for an endofunctor T on Set to carry an interesting $\mathscr{V}$-metric, which in turn determines a lifting $\overline{T}$ of T to $\mathscr{V}$-cat. In such case the discrete $\mathscr{V}$-cat-functor has as *ordinary* right adjoint the forgetful functor $\widetilde{V}^{\mathscr{V}} : \mathsf{Coalg}(\overline{T}) \longrightarrow \mathsf{Coalg}(T)$, which consequently preserves behaviors.

2 Preliminaries

In this section we gather all the necessary technicalities and notation from category theory enriched in a complete and cocomplete symmetric monoidal category that we shall use later. For the standard notions of enriched categories, enriched functors and enriched natural transformations we refer to Kelly's book [12].

We shall mainly use two prominent enrichments: that in a quantale $\mathscr{V}$ and that in the category $\mathscr{V}$-cat of *small* $\mathscr{V}$-categories and $\mathscr{V}$-functors for a quantale $\mathscr{V}$. We spell out in more details how the relevant notions look like, and carefully write all the enrichment-prefixes. In particular, the underlying category of an enriched category will be denoted by the same symbol, followed by the subscript "o" as usual.

2.1 Categories and functors enriched in a quantale

Suppose $\mathscr{V} = (\mathscr{V}_o, \otimes, e, [-,-])$ is a *quantale*. More in detail: $\mathscr{V}_o$ is a complete lattice, equipped with the commutative and associative monotone binary operation $\otimes$, called the *tensor*. We require the element e to be a *unit* of tensor. Furthermore, we require every monotone map $- \otimes r : \mathscr{V}_o \longrightarrow \mathscr{V}_o$ to have a right adjoint $[r,-] : \mathscr{V}_o \longrightarrow \mathscr{V}_o$. We call $[-,-]$ the *internal hom* of $\mathscr{V}_o$.

Quantales are the "simplest" complete and cocomplete symmetric monoidal closed categories. Therefore, one can define $\mathscr{V}$-categories, $\mathscr{V}$-functors, and $\mathscr{V}$-natural transformations. Before we say what these are, let us mention several examples of quantales.

▶ **Examples 2.1.**

1. The two-element chain $\mathbb{2} = \{0, 1\}$ with the usual order, and tensor $r \otimes s = r \wedge s$.
2. The real half line $([0, \infty], \geq_{\mathbb{R}})$, with (extended) addition as tensor product.
3. The unit interval $([0, 1], \geq_{\mathbb{R}})$ with tensor product $r \otimes s = \max(r, s)$.
4. The poset of all monotone functions $f : [0, \infty] \longrightarrow [0, 1]$ such that the equality $f(x) = \bigvee_{y<x} f(y)$ holds, with the pointwise order. It becomes a quantale with the tensor product
$$f \otimes g(z) = \bigvee_{x+y\leq z} f(x) \cdot g(y)$$
having as unit the function mapping all nonzero elements to 1, and 0 to itself [10].
5. The three-element chain $\mathbb{3} = \{0, 1, 2\}$ with usual order, and the (unique!) commutative tensor product with unit 1, which necessarily satisfies $2 \otimes 2 = 2$ (which can be seen by tensoring both sides of $1 \leq 2$ with 2). ◀

A *(small)* $\mathscr{V}$*-category* $\mathscr{X}$ consists of a (small) set of objects, together with an object $\mathscr{X}(x', x)$ in $\mathscr{V}_o$ for each pair x', x of objects, subject to the following axioms

$$e \leq \mathscr{X}(x, x), \qquad \mathscr{X}(x', x) \otimes \mathscr{X}(x'', x') \leq \mathscr{X}(x'', x)$$

for all objects x'', x' and x in $\mathscr{X}$. A $\mathscr{V}$-category $\mathscr{X}$ is called *discrete* if $\mathscr{X}(x', x) = e$ for $x' = x$, and $\perp$ otherwise.

A $\mathscr{V}$*-functor* $f : \mathscr{X} \longrightarrow \mathscr{Y}$ is given by the object-assignment $x \mapsto fx$, such that

$$\mathscr{X}(x', x) \leq \mathscr{Y}(fx', fx)$$

holds for all x', x.

A $\mathscr{V}$*-natural transformation* $f \longrightarrow g$ is given whenever

$$e \leq \mathscr{Y}(fx, gx)$$

holds for all x. Thus, there is at most one $\mathscr{V}$-natural transformation between f and g.

▶ **Example 2.2.** The two-element chain $\mathbb{2}$ is a quantale. A small $\mathbb{2}$-category[1] $\mathscr{X}$ is precisely a *preorder*, where $x' \leq x$ iff $\mathscr{X}(x', x) = 1$, while a $\mathbb{2}$-functor $f : \mathscr{X} \longrightarrow \mathscr{Y}$ is a monotone map. A $\mathbb{2}$-natural transformation $f \to g$ expresses that $fx \leq gx$ holds for every x. Thus $\mathbb{2}$-cat is the category Preord of preorders and monotone maps.

A good intution is that $\mathscr{V}$-categories are (rather general) metric spaces and $\mathscr{V}$-functors are nonexpanding maps. This intuition goes back to Lawvere [14]. We show next some examples that explain this intuition. For more details, see also [16].

▶ **Examples 2.3.**

1. Let $\mathscr{V}$ be the real half line $([0, \infty], \geq_{\mathbb{R}}, +, 0)$ as in Example 2.1.2. It is easy to see that a small $\mathscr{V}$-category can be identified with a set X and a mapping $d_X : X \times X \longrightarrow [0, \infty]$ such that $\langle X, d_X\rangle$ is a *generalized metric space*. The slight generalization of the usual notion lies in the fact that the distance function d is not necessarily symmetric and $d_X(x', x) = 0$ does not necessarily entail $x' = x$.
A $\mathscr{V}$-functor $f : (X, d_X) \longrightarrow (Y, d_Y)$ is then a exactly a *nonexpanding mapping*, i.e., one satisfying the inequality $d_Y(fx', fx) \leq d_X(x', x)$ for every $x, x' \in X$.
The existence of a $\mathscr{V}$-natural transformation $f \longrightarrow g$ means that $\bigvee_x d_Y(fx, gx) = 0$, i.e., the distance $d_Y(fx, gx)$ is 0, for every $x \in X$.

[1] To not be confounded with the notion of a 2-category, that is, a Cat-enriched category.

2. For the unit interval $\mathscr{V} = ([0,1], \geq_{\mathbb{R}}, \max, 0)$ from Example 2.1.3, a $\mathscr{V}$-category is a *generalized ultrametric space* $\langle X, d_X : X \times X \longrightarrow [0,1]\rangle$ [16, 20]. Again, the slight generalization of the usual notion lies in the fact that the distance function d is not necessarily symmetric and $d_X(x', x) = 0$ does not necessarily entail $x = x'$. Similarly, $\mathscr{V}$-functors are precisely the nonexpanding maps, and the existence of a $\mathscr{V}$-natural transformation $f \longrightarrow g : \langle X, d_X\rangle \longrightarrow \langle Y, d_Y\rangle$ means, again, that $\bigvee_x d_Y(fx, gx) = 0$, i.e., the distance $d_Y(fx, gx)$ is 0, for every $x \in X$.
3. Using the quantale $\mathscr{V}$ from Example 2.1.4 leads to *probabilistic metric spaces*: for a $\mathscr{V}$-category $\mathscr{X}$, and for every pair x, x' of objects of $\mathscr{X}$, the hom-object is a function $\mathscr{X}(x', x) : [0, \infty] \longrightarrow [0,1]$ with the intuitive meaning $\mathscr{X}(x', x)(r) = s$ holds iff s is the probability that the distance from x' to x is smaller than r. See [6, 10].
4. Finally, for the three-element quantale from Example 2.1.5, $\mathscr{V}$-enriched categories arose in the model of concurrency proposed by Gaifman and Pratt [8] under the name of *prossets*. Explicitly, the objects of a $\mathscr{V}$-category can be seen as events subject to a schedule, endowed with a preorder $\leq$ and a binary relation $\prec$, where $x \leq y$ iff $\mathscr{X}(x, y) \geq 1$ (with the interpretation that "y cannot begin before x begins, and cannot complete before x completes"), and $x \prec y$ iff $\mathscr{X}(x, y) = 2$ (which is intended to mean "y cannot begin until x has completed").

2.2 Categories, functors and natural transformations, enriched in $\mathscr{V}$-cat

Suppose that $\mathscr{V} = (\mathscr{V}_o, \otimes, e, [-,-])$ is a quantale. We denote by $\mathscr{V}\text{-}\mathsf{cat}_o$ the *ordinary* category of all small $\mathscr{V}$-categories and all $\mathscr{V}$-functors between them.

We recall (see for example [21]) that the ordinary category $\mathscr{V}\text{-}\mathsf{cat}_o$ has a monoidal closed structure. The *tensor product* $\mathscr{X} \otimes \mathscr{Y}$ is inherited from $\mathscr{V}$. Namely, $\mathscr{X} \otimes \mathscr{Y}$ has as objects the corresponding pairs of objects and we put

$$(\mathscr{X} \otimes \mathscr{Y})((x', y'), (x, y)) = \mathscr{X}(x', x) \otimes \mathscr{Y}(y', y)$$

The *unit* for the tensor product is the $\mathscr{V}$-category $\mathbb{1}$, with one object 0 and $\mathscr{V}$-hom $\mathbb{1}(0,0) = e$.

The $\mathscr{V}$-functor $- \otimes \mathscr{Y} : \mathscr{V}\text{-}\mathsf{cat}_o \longrightarrow \mathscr{V}\text{-}\mathsf{cat}_o$ has a right adjoint $[\mathscr{Y}, -]$. Explicitly, $[\mathscr{Y}, \mathscr{Z}]$ is the following $\mathscr{V}$-category:

1. Objects of $[\mathscr{Y}, \mathscr{Z}]$ are $\mathscr{V}$-functors from $\mathscr{Y}$ to $\mathscr{Z}$.
2. The "distance" $[\mathscr{Y}, \mathscr{Z}](f, g)$ is $\bigwedge_y \mathscr{Z}(fy, gy)$.

It follows from [13] that the symmetric monoidal closed category $(\mathscr{V}\text{-}\mathsf{cat}_o, \otimes, \mathbb{1}, [-,-])$ is complete and cocomplete, with generator consisting of $\mathscr{V}$-categories of the form $\mathbb{2}_r$, $r \in \mathscr{V}_o$. Here, every $\mathbb{2}_r$ has two objects 0 and 1, with $\mathscr{V}$-homs

$$\mathbb{2}_r(0,0) = \mathbb{2}_r(1,1) = e\,,\ \mathbb{2}_r(0,1) = r\,,\ \mathbb{2}_r(1,0) = \bot \tag{1}$$

Thus we can define $\mathscr{V}\text{-}\mathsf{cat}$-enriched categories, $\mathscr{V}\text{-}\mathsf{cat}$-functors and $\mathscr{V}\text{-}\mathsf{cat}$-natural transformations.

A (*small*) $\mathscr{V}\text{-}\mathsf{cat}$-*category* $\mathbb{X}$ consists of a (small) set of objects X, Y, Z, ..., a small $\mathscr{V}$-category $\mathbb{X}(X, Y)$ for every pair X, Y of objects, and $\mathscr{V}$-functors

$$u_X : \mathbb{1} \longrightarrow \mathbb{X}(X, X), \quad c_{X,Y,Z} : \mathbb{X}(Y, Z) \otimes \mathbb{X}(X, Y) \longrightarrow \mathbb{X}(X, Z)$$

that represent the identity and composition and satisfy the usual axioms [12]:

$$\begin{tikzcd}
\mathbb{X}(Z,W) \otimes \mathbb{X}(Y,Z) \otimes \mathbb{X}(X,Y) \arrow[r, "1 \otimes c_{X,Y,Z}"] \arrow[d, "c_{Y,Z,W} \otimes 1"'] & \mathbb{X}(Z,W) \otimes \mathbb{X}(X,Z) \arrow[d, "c_{X,Z,W}"] \\
\mathbb{X}(Y,W) \otimes \mathbb{X}(X,Y) \arrow[r, "c_{X,Y,W}"] & \mathbb{X}(X,W)
\end{tikzcd}$$

$$\begin{tikzcd}
\mathbb{1} \otimes \mathbb{X}(X,Y) \arrow[r, "u_Y \otimes 1"] \arrow[dr, "\cong"'] & \mathbb{X}(Y,Y) \otimes \mathbb{X}(X,Y) \arrow[d, "c_{X,Y,Y}"] \\
& \mathbb{X}(X,Y)
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathbb{X}(X,Y) \otimes \mathbb{X}(X,X) \arrow[d, "c_{X,X,Y}"'] & \mathbb{X}(X,Y) \otimes \mathbb{1} \arrow[l, "1 \otimes u_X"'] \arrow[dl, "\cong"] \\
\mathbb{X}(X,Y) &
\end{tikzcd}$$

Objects of $\mathbb{X}(X,Y)$ will be sometimes denoted by $f : X \longrightarrow Y$ and their "distance" by $\mathbb{X}(X,Y)(f,g)$ in $\mathscr{V}$. The action of $c_{X,Y,Z}$ at objects (f',f) in $\mathbb{X}(Y,Z) \otimes \mathbb{X}(X,Y)$ is denoted simply by $f' \cdot f$, and for their distances the inequality below (expressing that $c_{X,Y,Z}$ is a $\mathscr{V}$-functor) holds:

$$(\mathbb{X}(Y,Z) \otimes \mathbb{X}(X,Y))\,((f',g'),(f,g)) \leq \mathbb{X}(X,Z)(f' \cdot f, g' \cdot g)$$

A $\mathscr{V}$-cat-*functor* $F : \mathbb{X} \longrightarrow \mathbb{Y}$ is given by:

1. The assignment $X \mapsto FX$ on objects.
2. For each pair of objects X, X' in $\mathbb{X}$, a $\mathscr{V}$-functor $F_{X',X} : \mathbb{X}(X',X) \longrightarrow \mathbb{Y}(FX',FX)$, whose action on objects $f : X' \longrightarrow X$ is denoted by $Ff : FX' \longrightarrow FX$. For the distances we have the inequality

$$\mathbb{X}(X',X)(f',f) \leq \mathbb{Y}(FX',FX)(Ff',Ff)$$

Of course, the diagrams of $\mathscr{V}$-functors below, expressing the preservation of unit and composition, should commute:

$$\begin{tikzcd}
\mathbb{X}(X,X) \arrow[rr, "F_{X,X}"] & & \mathbb{Y}(FX,FX) \\
& \mathbb{1} \arrow[ul, "u_X"] \arrow[ur, "u_{FX}"'] &
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathbb{X}(Y,Z) \otimes \mathbb{X}(X,Y) \arrow[r, "F_{Y,Z} \otimes F_{X,Y}"] \arrow[d, "c_{X,Y,Z}"'] & \mathbb{Y}(FY,FZ) \otimes \mathbb{Y}(FX,FY) \arrow[d, "c_{X,Y,Z}"] \\
\mathbb{X}(X,Z) \arrow[r, "F_{X,Z}"'] & \mathbb{Y}(FX,FZ)
\end{tikzcd}$$

Given $F, G : \mathbb{X} \longrightarrow \mathbb{Y}$, a $\mathscr{V}$-cat-natural transformation $\tau : F \longrightarrow G$ is given by a collection of $\mathscr{V}$-cat-functors $\tau_X : \mathbb{1} \longrightarrow \mathbb{Y}(FX,GX)$, such that the diagram

$$\begin{tikzcd}
& \mathbb{1} \otimes \mathbb{X}(X',X) \arrow[r, "\tau_X \otimes F_{X',X}"] & \mathbb{Y}(FX,GX) \otimes \mathbb{Y}(FX',FX) \arrow[dr, "c_{FX',FX,GX}"] & \\
\mathbb{X}(X',X) \arrow[ur, "\cong"] \arrow[dr, "\cong"'] & & & \mathbb{Y}(FX',GX) \\
& \mathbb{X}(X',X) \otimes \mathbb{1} \arrow[r, "G_{X',X} \otimes \tau_{X'}"'] & \mathbb{Y}(FX',GX) \otimes \mathbb{Y}(FX',GX') \arrow[ur, "c_{FX',GX',GX}"'] &
\end{tikzcd}$$

of $\mathscr{V}$-functors commutes. We shall abuse the notation and denote by $\tau_X : FX \longrightarrow GX$ the image in $\mathbb{Y}(FX,GX)$ of 0 in $\mathbb{1}$ under $\tau_X : \mathbb{1} \longrightarrow \mathbb{Y}(FX,GX)$. The above diagram (when read at the object-assignments of the ambient $\mathscr{V}$-functors) then translates as the equality

$$Gf \cdot \tau_{X'} = \tau_X \cdot Ff$$

of objects of the $\mathscr{V}$-category $\mathbb{Y}(FX',GX)$, for every object $f : X' \longrightarrow X$. On hom-objects, the above diagram says nothing[2] (recall that $\mathscr{V}_o$ is a poset, hence there are no parallel pairs of morphisms in $\mathscr{V}_o$).

Since $\mathscr{V}$-categories are "generalized metric spaces" (as seen in Examples 2.3), $\mathscr{V}$-cat-categories are "locally" metric spaces and $\mathscr{V}$-cat-functors are "locally" nonexpanding.

The last bit of notation standard from enriched category theory concerns colimits. We introduce it for $\mathscr{V}$-cat-categories.

[2] This is well-known for Preord-natural transformations: one only needs to verify ordinary naturality.

▶ **Definition 2.4.** A *colimit* of a diagram $D : \mathbb{D} \longrightarrow \mathbb{X}$ weighted by a $\mathscr{V}$-cat-functor $\varphi : \mathbb{D}^{op} \longrightarrow \mathscr{V}$-$\mathsf{cat}$ consists of an object $\varphi * D$ of $\mathbb{X}$, together with an isomorphism

$$\mathbb{X}(\varphi * D, X) \cong [\mathbb{D}^{op}, \mathscr{V}\text{-}\mathsf{cat}](\varphi, \mathbb{X}(D-, X))$$

which is $\mathscr{V}$-cat-natural in X.

In case $\mathbb{D}$ is the one-object $\mathscr{V}$-cat-category, we can identify the $\mathscr{V}$-cat-functor D with an object P of $\mathbb{X}$ and φ with a $\mathscr{V}$-category $\mathscr{C}$. We write then $\mathscr{C} \bullet P$ instead of $\varphi * D$.

▶ **Example 2.5.** Let Set denote in the sequel the free $\mathscr{V}$-cat-category on the ordinary category of sets and functions Set_o. This means that $\mathsf{Set}(X', X) = \mathsf{Set}_o(X', X) \bullet \mathbb{1}$, hence the homs of Set are copowers of the one-element "metric" space, indexed by set-theoretical maps from X' to X (that is, $\mathsf{Set}(X', X)$ is a discrete $\mathscr{V}$-category). Observe that ordinary functors $\mathsf{Set}_o \longrightarrow \mathsf{Set}_o$ automatically induce $\mathscr{V}$-cat-enriched functors $\mathsf{Set} \longrightarrow \mathsf{Set}$, and similarly for natural transformations between such ordinary functors.

3 Extensions from Set to $\mathscr{V}$-cat

From now on, we fix a quantale $\mathscr{V}$. We consider $\mathscr{V}$-cat enriched over itself as usual, using its internal hom described in Section 2.2, and Set as free $\mathscr{V}$-cat-category (Example 2.5).
Denote by $D^{\mathscr{V}} : \mathsf{Set} \longrightarrow \mathscr{V}$-$\mathsf{cat}$ the corresponding $\mathscr{V}$-cat-enriched embedding. Explicitly, $D^{\mathscr{V}}$ maps a set X to the discrete $\mathscr{V}$-category having X as set of objects.
Notice that there is an *ordinary* adjunction $D^{\mathscr{V}}_o \dashv V^{\mathscr{V}} : \mathscr{V}$-$\mathsf{cat}_o \longrightarrow \mathsf{Set}_o$ where the (ordinary) functor $V^{\mathscr{V}}$ maps a $\mathscr{V}$-category $\mathscr{X}$ to its set of objects of $\mathscr{X}$.

▶ **Definition 3.1.** Let $T : \mathsf{Set} \longrightarrow \mathsf{Set}$, $\overline{T} : \mathscr{V}$-$\mathsf{cat} \longrightarrow \mathscr{V}$-$\mathsf{cat}$ be $\mathscr{V}$-cat-functors.

- We say that a $\mathscr{V}$-cat-natural isomorphism

$$\begin{tikzcd}
\mathscr{V}\text{-}\mathsf{cat} \arrow[r, "\overline{T}"] & \mathscr{V}\text{-}\mathsf{cat} \\
\mathsf{Set} \arrow[u, "D^{\mathscr{V}}"] \arrow[r, "T"'] \arrow[ur, phantom, "\nwarrow\alpha"] & \mathsf{Set} \arrow[u, "D^{\mathscr{V}}"']
\end{tikzcd}$$

of $\mathscr{V}$-cat-functors exhibits $\overline{T}$ as an *extension* of T. If additionally the above isomorphism α is the unit of a left Kan extension, i.e., if $\overline{T} = \mathrm{Lan}_{D^{\mathscr{V}}}(D^{\mathscr{V}}T)$ holds, then we say that α exhibits $\overline{T}$ as the $\mathscr{V}$-cat-*ification* of T, and we shall denote it by $T_{\mathscr{V}}$.

- We say that a natural isomorphism

$$\begin{tikzcd}
\mathscr{V}\text{-}\mathsf{cat}_o \arrow[r, "\overline{T}_o"] \arrow[d, "V^{\mathscr{V}}"'] & \mathscr{V}\text{-}\mathsf{cat}_o \arrow[d, "V^{\mathscr{V}}"] \\
\mathsf{Set}_o \arrow[r, "T_o"'] \arrow[ur, phantom, "\nwarrow\beta"] & \mathsf{Set}_o
\end{tikzcd}$$

of ordinary functors exhibits $\overline{T}$ as a *lifting* of T.

▶ **Examples 3.2.**

1. The identity $\mathscr{V}$-cat-functor $\mathsf{Id} : \mathscr{V}$-$\mathsf{cat} \longrightarrow \mathscr{V}$-$\mathsf{cat}$ is *always* an extension and a lifting of the identity ($\mathscr{V}$-cat-)functor on Set.
In case the quantale has an element r satisfying $e \leq r$ and $r \otimes r \leq r$ (consequently, $r \otimes r = r$), then the identity on Set has another lifting, namely $\mathsf{Id}_r : \mathscr{V}$-$\mathsf{cat} \longrightarrow \mathscr{V}$-$\mathsf{cat}$, mapping a $\mathscr{V}$-category $\mathscr{X}$ to the $\mathscr{V}$-category with same objects, and $\mathscr{V}$-homs $(\mathsf{Id}_r \mathscr{X})(x', x) = \mathscr{X}(x', x) \otimes r$ "shrinked" by r, and acting as identity on $\mathscr{V}$-functors.

2. Extensions and liftings need not be unique. We have seen above an example for liftings, now we give one for extensions. Suppose $\mathscr{V} = \mathbb{2}$ (thus $\mathscr{V}$-cat is Preord). We shall then denote simply by $D : \mathsf{Set} \longrightarrow \mathsf{Preord}$ the discrete functor, omitting the superscript $\mathbb{2}$. It has as ($\mathbb{2}$-enriched!) left adjoint the functor $C : \mathsf{Preord} \longrightarrow \mathsf{Set}$ assigning to any preorder X the set of its connected components. The composite $\pi = DC : \mathsf{Preord} \longrightarrow \mathsf{Preord}$ is an extension of $\mathsf{Id} : \mathsf{Set} \longrightarrow \mathsf{Set}$. The latter follows from the fact that $\pi D \cong DCD \cong D$ holds by virtue of the counit of $C \dashv D$. Hence both Id and π are extensions of $\mathsf{Id} : \mathsf{Set} \longrightarrow \mathsf{Set}$.

   We shall later show (Examples 3.7) that $\mathsf{Id} : \mathscr{V}\text{-}\mathsf{cat} \longrightarrow \mathscr{V}\text{-}\mathsf{cat}$ is, in fact, a $\mathscr{V}$-cat-ification of the identity functor on Set, for an *arbitrary* quantale $\mathscr{V}$.
3. A $\mathscr{V}$-cat-ification $T_{\mathscr{V}}$ exists for every *accessible* functor $T : \mathsf{Set} \longrightarrow \mathsf{Set}$ for rather trivial reasons. More in detail, if T is λ-accessible for a regular cardinal, then $T = \mathrm{Lan}_{J_\lambda}(TJ_\lambda)$, where $J_\lambda : \mathsf{Set}_\lambda \longrightarrow \mathsf{Set}$ is the inclusion of the full subcategory Set_λ spanned by λ-small sets. Consequently,

$$T_{\mathscr{V}} = \mathrm{Lan}_{D^{\mathscr{V}} J_\lambda}(D^{\mathscr{V}} T J_\lambda)$$

   exhibits $T_{\mathscr{V}}$ as $\mathrm{Lan}_{D^{\mathscr{V}}}(D^{\mathscr{V}} T)$ by [12, Theorem 4.47]. In particular, the $\mathscr{V}$-cat-ification $(T_\Sigma)_{\mathscr{V}}$ exists for every polynomial functor

$$T_\Sigma X = \coprod_n \mathsf{Set}(n, X) \bullet \Sigma n$$

   where $\Sigma : |\mathsf{Set}_\lambda| \longrightarrow \mathsf{Set}$ is a λ-ary signature. We shall give an explicit formula for the $\mathscr{V}$-cat-ification $(T_\Sigma)_{\mathscr{V}}$ later. ◀

We plan to show that for each endofunctor T on Set, its $\mathscr{V}$-cat-ification exists. We shall obtain this from the more general result below, which also will provide examples of liftings.

▶ **Theorem 3.3.** *Every functor $H : \mathsf{Set} \longrightarrow \mathscr{V}\text{-}\mathsf{cat}$ has a $\mathscr{V}$-cat-enriched left Kan extension $H^\sharp : \mathscr{V}\text{-}\mathsf{cat} \longrightarrow \mathscr{V}\text{-}\mathsf{cat}$ along $D^{\mathscr{V}} : \mathsf{Set} \longrightarrow \mathscr{V}\text{-}\mathsf{cat}$.*

Proof. We first introduce a $\mathscr{V}$-cat-functor $N : \mathbb{N}^{op} \longrightarrow \mathscr{V}\text{-}\mathsf{cat}$. Its domain $\mathbb{N}$ is the *free* $\mathscr{V}$-cat-category built upon the following ordinary category N: the objects are all r in $\mathscr{V}_o$, together with an extra symbol Ω, with arrows $\delta_0^r : r \longrightarrow \Omega$ and $\delta_1^r : r \longrightarrow \Omega$, for all r in $\mathscr{V}_o$.

We define N to be the $\mathscr{V}$-cat-functor sending Ω to $\mathbb{1}$, and r to $\mathbb{2}_r$. Recall that $\mathbb{1}$ is the unit one-object $\mathscr{V}$-category with $\mathbb{1}(0,0) = e$, and $\mathbb{2}_r$ is the $\mathscr{V}$-category on two objects 0 and 1, with the only non-trivial "distance" $\mathbb{2}_r(0,1) = r$, as introduced in Equation (1). The action of N on arrows is defined as follows: $N\delta_0^r : \mathbb{1} \longrightarrow \mathbb{2}_r$ sends 0 to 0, while $N\delta_1^r : \mathbb{1} \longrightarrow \mathbb{2}_r$ sends 0 to 1.

Then, for every $\mathscr{V}$-category $\mathscr{X}$, we consider the following $\mathscr{V}$-cat-functor $D_{\mathscr{X}} : \mathbb{N} \longrightarrow \mathsf{Set}$. Since $\mathbb{N}$ is a free $\mathscr{V}$-cat-category, it suffices to define an ordinary functor $\mathsf{N} \longrightarrow \mathsf{Set}_o$. We put $D_{\mathscr{X}}\Omega$ to be the set of objects of $\mathscr{X}$. Every r is sent to the set $D_{\mathscr{X}} r$ of pairs (x', x) of objects such that $r \leq \mathscr{X}(x', x)$ holds. The mapping $D_{\mathscr{X}}\delta_0^r$ sends (x', x) to x' and $D_{\mathscr{X}}\delta_1^r$ sends (x', x) to x.

We prove the following facts:

1. The colimit $N * (D^{\mathscr{V}} D_{\mathscr{X}})$ in $\mathscr{V}$-cat is isomorphic to $\mathscr{X}$.
2. If we define $H^\sharp \mathscr{X}$ as the colimit $N * (HD_{\mathscr{X}})$, then the assignment $\mathscr{X} \mapsto H^\sharp \mathscr{X}$ can be extended to a $\mathscr{V}$-cat-functor that is a left Kan extension of H along $D^{\mathscr{V}}$.

Let us proceed:

1. The colimit $N * (D^{\mathscr{V}} D_{\mathscr{X}})$ exists in $\mathscr{V}$-cat, since the $\mathscr{V}$-cat-category $\mathbb{N}$ is small.
To ease the notation, we put $D^{\mathscr{V}} D_{\mathscr{X}} \Omega = \mathscr{X}_\Omega$, $D^{\mathscr{V}} D_{\mathscr{X}} r = \mathscr{X}_r$, $D^{\mathscr{V}} D_{\mathscr{X}} \delta_0^r = \partial_0^r$, and $D D_{\mathscr{X}} \delta_1^r = \partial_1^r$.
Let us analyze the defining isomorphism

$$\mathscr{V}\text{-}\mathsf{cat}(N * (D^{\mathscr{V}} D_{\mathscr{X}}), \mathscr{Y}) \cong [\mathbb{N}^{op}, \mathscr{V}\text{-}\mathsf{cat}](N, \mathscr{V}\text{-}\mathsf{cat}(D^{\mathscr{V}} D_{\mathscr{X}} -, \mathscr{Y}))$$

of $\mathscr{V}$-categories, natural in $\mathscr{Y}$.
The $\mathscr{V}$-category $[\mathbb{N}^{op}, \mathscr{V}\text{-}\mathsf{cat}](N, \mathscr{V}\text{-}\mathsf{cat}(D^{\mathscr{V}} D_{\mathscr{X}} -, \mathscr{Y}))$ of N-weighted "cocones" for $D^{\mathscr{V}} D_{\mathscr{X}}$ is described as follows:

a. The objects are $\mathscr{V}$-cat-natural transformations $\tau : N \longrightarrow \mathscr{V}\text{-}\mathsf{cat}(D^{\mathscr{V}} D_{\mathscr{X}} -, \mathscr{Y})$. Each such τ consists of $\mathscr{V}$-functors

i. $\tau_\Omega : N\Omega \longrightarrow \mathscr{V}\text{-}\mathsf{cat}(\mathscr{X}_\Omega, \mathscr{Y})$. Since $N\Omega = \mathbb{1}$, τ_Ω picks up a $\mathscr{V}$-functor $f_\Omega : \mathscr{X}_\Omega \longrightarrow \mathscr{Y}$. No other restrictions are imposed since $\mathbb{1}(0,0) = e$.

ii. $\tau_r : Nr \longrightarrow \mathscr{V}\text{-}\mathsf{cat}(\mathscr{X}_r, \mathscr{Y})$. This $\mathscr{V}$-functor picks up two $\mathscr{V}$-functors $f_0^r : \mathscr{X}_r \longrightarrow \mathscr{Y}$ and $f_1^r : \mathscr{X}_r \longrightarrow \mathscr{Y}$. Since $\mathscr{X}_r$ is discrete, both f_0 and f_1 are defined by their object-assignments only. There is, however, the constraint below, because $Nr = \mathbb{2}_r$:

$$r \leq \bigwedge_{r \leq \mathscr{X}(x',x)} \mathscr{Y}(f_0^r(x',x), f_1^r(x',x))$$

In addition to the above, there are various commutativity conditions since τ is natural. Explicitly, for $\delta_0^r : r \longrightarrow \Omega$, we have the commutative square

$$\begin{tikzcd}
N\Omega \arrow[r, "\tau_\Omega"] \arrow[d, "N\delta_0^r"'] & \mathscr{V}\text{-}\mathsf{cat}(\mathscr{X}_\Omega, \mathscr{Y}) \arrow[d, "\mathscr{V}\text{-}\mathsf{cat}(\partial_0^r, \mathscr{Y})"] \\
Nr \arrow[r, "\tau_r"'] & \mathscr{V}\text{-}\mathsf{cat}(\mathscr{X}_r, \mathscr{Y})
\end{tikzcd}$$

that, on the level of objects, is the requirement $f_\Omega \cdot \partial_0^r = f_0^r$. Analogously, the requirement $f_\Omega \cdot \partial_1^r = f_1^r$ holds.
We conclude that to give τ reduces to a $\mathscr{V}$-functor $f_\Omega : \mathscr{X}_\Omega \longrightarrow \mathscr{Y}$ (and, recall, this $\mathscr{V}$-functor is given just by the object-assignment $x \mapsto f_\Omega x$, since $\mathscr{X}_\Omega$ is discrete) such that $r \leq \mathscr{Y}(f_\Omega x', f_\Omega x)$ holds for every object (x', x) in $\mathscr{X}_r$ and every r.
This means precisely that $\mathscr{X}(x', x) \leq \mathscr{Y}(f_\Omega x', f_\Omega x)$ holds.

b. Given τ and τ', then

$$[\mathbb{N}^{op}, \mathscr{V}\text{-}\mathsf{cat}](N, \mathscr{V}\text{-}\mathsf{cat}(D^{\mathscr{V}} D_{\mathscr{X}} -, \mathscr{Y}))(\tau, \tau') = \bigwedge_x \mathscr{Y}(f_\Omega x, f'_\Omega x)$$

where f_Ω corresponds to τ and f'_Ω corresponds to τ'.

From the above, it follows that the $\mathscr{V}$-functor $q_{\mathscr{X}} : \mathscr{X}_\Omega \longrightarrow \mathscr{X}$ that sends each object x to itself is the couniversal such "cocone". More precisely, $r \leq \mathscr{X}(q_{\mathscr{X}} x', q_{\mathscr{X}} x)$ holds for every (x', x) in $\mathscr{X}_r$ and every r.
Furthermore, given any $\mathscr{V}$-functor $f_\Omega : \mathscr{X}_\Omega \longrightarrow \mathscr{Y}$ with the above properties, then there is a unique $\mathscr{V}$-functor $f_\Omega^\sharp : \mathscr{X} \longrightarrow \mathscr{Y}$ such that $f_\Omega^\sharp q_{\mathscr{X}} = f_\Omega$ holds.
The "2-dimensional aspect" of the colimit says that

$$\bigwedge_x \mathscr{Y}(f_\Omega^\sharp x, f_\Omega'^\sharp x) = \bigwedge_x \mathscr{Y}(f_\Omega x, f'_\Omega x)$$

Hence we have proved that $\mathscr{X}$ is isomorphic to $N * (D^{\mathscr{V}} D_{\mathscr{X}})$.

2. Suppose $H : \mathsf{Set} \longrightarrow \mathscr{V}\text{-}\mathsf{cat}$ is given.

a. We first define a $\mathscr{V}$-cat-functor $H^\sharp : \mathscr{V}\text{-}\mathsf{cat} \longrightarrow \mathscr{V}\text{-}\mathsf{cat}$.

To make the notation less heavy, for every small $\mathscr{V}$-category $\mathscr{X}$ and every $r \in \mathscr{V}_o$, we denote by X_r the *set* of pairs (x', x) such that $r \le \mathscr{X}(x', x)$ and by X_Ω the *set* of objects of $\mathscr{X}$. Analogously, for a $\mathscr{V}$-functor $f : \mathscr{X} \longrightarrow \mathscr{Y}$, we denote by $f_r : X_r \longrightarrow Y_r$ and $f_\Omega : X_\Omega \longrightarrow Y_\Omega$ the maps corresponding to $(x', x) \mapsto (fx', fx)$ and the object assignment of f, respectively. Let also denote $d_0^r = D_{\mathscr{X}}\delta_0^r$ and $d_1^r = D_{\mathscr{X}}\delta_1^r$.

For every small $\mathscr{V}$-category $\mathscr{X}$, we put $H^\sharp\mathscr{X}$ to be the colimit $N * (HD_{\mathscr{X}})$.

Unravelling the definition of the weighted colimit, the 1-dimensional aspect says that to give a $\mathscr{V}$-functor $f^\sharp : H^\sharp\mathscr{X} \longrightarrow \mathscr{Y}$ is the same as to give a $\mathscr{V}$-functor $f : HX_\Omega \longrightarrow \mathscr{Y}$ such that

$$r \le \bigwedge_{C \in HX_r} \mathscr{Y}(fHd_0^r(C), fHd_1^r(C)) \tag{2}$$

holds for all r.[3] In particular, there is a "quotient" $\mathscr{V}$-functor $c_{\mathscr{X}} : HX_\Omega \longrightarrow H^\sharp\mathscr{X}$ such that

$$r \le \bigwedge_{C \in HX_r} H^\sharp\mathscr{X}(c_{\mathscr{X}}Hd_0^r(C), c_{\mathscr{X}}Hd_1^r(C)) \tag{3}$$

holds for all r, with the property that any $\mathscr{V}$-functor $HX_\Omega \longrightarrow \mathscr{Y}$ satisfying (2) uniquely factorizes through $c_{\mathscr{X}}$.

The 2-dimensional aspect of the colimit says that given any $f, g : HX_\Omega \longrightarrow \mathscr{Y}$, the relation

$$\bigwedge_{B \in HX_\Omega} \mathscr{Y}(f(B), g(B)) = \bigwedge_{A \in H^\sharp\mathscr{X}} \mathscr{Y}(f^\sharp(A), g^\sharp(A)) \tag{4}$$

holds.

For a $\mathscr{V}$-functor $f : \mathscr{X} \longrightarrow \mathscr{Y}$ we recall that the diagram

$$\begin{tikzcd}
X_r \arrow[r, shift left, "d_1^r"] \arrow[r, shift right, "d_0^r"'] \arrow[d, "f_r"'] & X_\Omega \arrow[d, "f_\Omega"] \\
Y_r \arrow[r, shift left, "d_1^r"] \arrow[r, shift right, "d_0^r"'] & Y_\Omega
\end{tikzcd}$$

commutes serially. Hence f induces a $\mathscr{V}$-cat-natural transformation $D_f : D_{\mathscr{X}} \longrightarrow D_{\mathscr{Y}}$. Therefore we can define $H^\sharp f : H^\sharp\mathscr{X} \longrightarrow H^\sharp\mathscr{Y}$ as the unique mediating $\mathscr{V}$-functor

$$N * (HD_f) : N * (HD_{\mathscr{X}}) \longrightarrow N * (HD_{\mathscr{Y}})$$

In particular, we have the commutative diagram below:

$$\begin{tikzcd}
HX_\Omega \arrow[r, "c_{\mathscr{X}}"] \arrow[d, "Hf_\Omega"'] & H^\sharp\mathscr{X} \arrow[d, "H^\sharp f"] \\
HY_\Omega \arrow[r, "c_{\mathscr{Y}}"] & H^\sharp\mathscr{Y}
\end{tikzcd}$$

Also, from the 2-dimensional aspect of the colimit (see Eq. (4)), we have that for any $f, g : \mathscr{X} \longrightarrow \mathscr{Y}$, the equality below holds:

$$\bigwedge_{B \in HX_\Omega} H^\sharp\mathscr{Y}(c_{\mathscr{Y}}Hf_\Omega(B), c_{\mathscr{Y}}Hg_\Omega(B)) = \bigwedge_{A \in H^\sharp\mathscr{X}} H^\sharp\mathscr{Y}(H^\sharp f(A), H^\sharp g(A)) \tag{5}$$

[3] By slight abuse of language, we shall use here and subsequently notation like $C \in HX_r$ to mean that C runs through all objects in *the $\mathscr{V}$-category HX_r*.

It remains to prove that the inequality

$$\mathscr{V}\text{-}\mathsf{cat}(\mathscr{X},\mathscr{Y})(f,g) \leq \mathscr{V}\text{-}\mathsf{cat}(H^\sharp\mathscr{X}, H^\sharp\mathscr{Y})(H^\sharp f, H^\sharp g)$$

is satisfied. To that end, suppose that $r \leq \mathscr{V}\text{-}\mathsf{cat}(\mathscr{X},\mathscr{Y})(f,g)$ holds. This is equivalent to the fact that there is a mapping $t : X_\Omega \longrightarrow Y_r$ such that the triangles

$$\begin{tikzcd} X_\Omega \arrow[r, "t"] \arrow[dr, "f_\Omega"'] & Y_r \arrow[d, "d_0^r"] \\ & Y_\Omega \end{tikzcd} \qquad \begin{tikzcd} X_\Omega \arrow[r, "t"] \arrow[dr, "g_\Omega"'] & Y_r \arrow[d, "d_1^r"] \\ & Y_\Omega \end{tikzcd} \tag{6}$$

commute. In fact, $t(x) = (f(x), g(x))$. To prove that $r \leq \mathscr{V}\text{-}\mathsf{cat}(H^\sharp\mathscr{X}, H^\sharp\mathscr{Y})(H^\sharp f, H^\sharp g)$ holds, we need to prove the inequality

$$r \leq \bigwedge_{A \in H^\sharp\mathscr{X}} H^\sharp\mathscr{Y}(H^\sharp f(A), H^\sharp g(A))$$

This follows from:

$$\begin{aligned} r &\leq \bigwedge_{C \in HY_r} H^\sharp\mathscr{Y}(c_\mathscr{Y} Hd_0^r(C), c_\mathscr{Y} Hd_1^r(C)) && \text{by (3)} \\ &\leq \bigwedge_{B \in HX_\Omega} H^\sharp\mathscr{Y}(c_\mathscr{Y} Hd_0^r Ht(B), c_\mathscr{Y} Hd_1^r Ht(B)) \\ &= \bigwedge_{B \in HX_\Omega} H^\sharp\mathscr{Y}(c_\mathscr{Y} Hf_\Omega(B), c_\mathscr{Y} Hg_\Omega(B)) && \text{by (6)} \\ &= \bigwedge_{A \in H^\sharp\mathscr{X}} H^\sharp\mathscr{Y}(H^\sharp f(A), H^\sharp g(A)) && \text{by (5)} \end{aligned}$$

We proved that $\mathscr{X} \mapsto H^\sharp\mathscr{X}$ can be extended to a $\mathscr{V}$-cat-functor $H^\sharp : \mathscr{V}\text{-}\mathsf{cat} \longrightarrow \mathscr{V}\text{-}\mathsf{cat}$.

b. We prove now that $H^\sharp \cong \mathrm{Lan}_{D^\mathscr{V}} H$ holds.

Due to the definition of $H^\sharp$, there is a $\mathscr{V}$-cat-natural isomorphism $\alpha : H \longrightarrow H^\sharp D^\mathscr{V}$. We prove that α is the unit of a left Kan extension.

Suppose that $K : \mathscr{V}\text{-}\mathsf{cat} \longrightarrow \mathscr{V}\text{-}\mathsf{cat}$ is any $\mathscr{V}$-cat-functor. To give a $\mathscr{V}$-cat-natural transformation $\tau : H^\sharp \longrightarrow K$ is to give a collection $\tau_\mathscr{X} : H^\sharp\mathscr{X} \longrightarrow K\mathscr{X}$ of $\mathscr{V}$-functors such that the square

$$\begin{tikzcd} H^\sharp\mathscr{X} \arrow[r, "\tau_\mathscr{X}"] \arrow[d, "H^\sharp f"'] & K\mathscr{X} \arrow[d, "Kf"] \\ H^\sharp\mathscr{Y} \arrow[r, "\tau_\mathscr{Y}"'] & K\mathscr{Y} \end{tikzcd}$$

commutes for every $\mathscr{V}$-functor $f : \mathscr{X} \longrightarrow \mathscr{Y}$.

The composite

$$H \xrightarrow{\alpha} H^\sharp D^\mathscr{V} \xrightarrow{\tau D^\mathscr{V}} KD^\mathscr{V}$$

yields a natural transformation $\tau^\flat : H^\sharp \longrightarrow KD^\mathscr{V}$.

Conversely, for every natural transformation $\sigma : H \longrightarrow KD^\mathscr{V}$, we define $\sigma^\sharp : H^\sharp \longrightarrow K$ at a $\mathscr{V}$-category $\mathscr{X}$ by considering first the composite

$$HD_\mathscr{X} \xrightarrow{\sigma D_\mathscr{X}} KD^\mathscr{V} D_\mathscr{X} \xrightarrow{Kc_\mathscr{X}} K\mathscr{X}$$

which yields $\sigma^\sharp_\mathscr{X} : H^\sharp\mathscr{X} \longrightarrow K\mathscr{X}$ by the passage to colimit (where $c_\mathscr{X} : D^\mathscr{V} D_\mathscr{X} \longrightarrow \mathscr{X}$ is the colimiting cocone).

The processes $\tau \mapsto \tau^\flat$ and $\sigma \mapsto \sigma^\sharp$ are inverses to each other. ◀

▶ **Remark 3.4.** The proof of the above theorem also provides a recipe on how to compute the left Kan extension of a $\mathscr{V}$-cat-functor $H : \mathsf{Set} \longrightarrow \mathscr{V}\text{-}\mathsf{cat}$ along $D^{\mathscr{V}}$. Recall the notation such as X_Ω and X_r from item 2.a of the proof. For a $\mathscr{V}$-category $\mathscr{X}$, $H^\sharp\mathscr{X}$ is the $\mathscr{V}$-category having the same objects as HX_Ω (that is, the underlying set of objects of the $\mathscr{V}$-category obtained by applying H to the set of objects of $\mathscr{X}$). The couniversal cocone $c_{\mathscr{X}} : HX_\Omega \longrightarrow H^\sharp\mathscr{X}$ is the identity on objects. The $\mathscr{V}$-homs are, for any two objects A', A, given by $H^\sharp\mathscr{X}(A', A) =$

$$\bigvee\{HX_\Omega(A', A_0) \otimes r_1 \otimes HX_\Omega(A'_1, A_1) \otimes r_2 \otimes \ldots \otimes HX_\Omega(A'_{n-1}, A_{n-1}) \otimes r_n \otimes HX_\Omega(A_n, A)\}$$

where the join is computed over all (possibly empty) paths $(A_0, A'_1, A_1, \ldots, A'_n, A_n)$ and all (possibly empty) tuples of elements $(r_1, \ldots, r_n)$ such that there are $C_i \in HX_{r_i}$ with $Hd_0^{r_i}(C_i) = A_{i-1}$, $Hd_1^{r_i}(C_i) = A'_i$, for all $i = 1, n$:

$$\begin{tikzcd}
 & C_1 \in HX_{r_1} \arrow[dl, "Hd_0^{r1}"'] \arrow[dr, "Hd_1^{r1}"] & & C_2 \in HX_{r_2} \arrow[dl, "Hd_0^{r2}"'] \arrow[dr, "Hd_1^{r2}"] & & \cdots \arrow[dl, "\ldots"'] \arrow[dr, "\ldots"] & & C_n \in HX_{r_n} \arrow[dl, "Hd_0^{rn}"'] \arrow[dr, "Hd_1^{rn}"] & \\
A' \,,\, A_0 & & A'_1 \,,\, A_1 & & A'_2 \,,\, A_2 & & A'_{n-1} \,,\, A_{n-1} & & A_n \,,\, A
\end{tikzcd}$$

▶ **Corollary 3.5.** *Every $T : \mathsf{Set} \longrightarrow \mathsf{Set}$ has a $\mathscr{V}$-cat-ification.*

Proof. Apply Theorem 3.3 to the composite $H = D^{\mathscr{V}}T : \mathsf{Set} \longrightarrow \mathscr{V}\text{-}\mathsf{cat}$. ◀

In particular, we obtain from the above that $\mathsf{Id} : \mathscr{V}\text{-}\mathsf{cat} \longrightarrow \mathscr{V}\text{-}\mathsf{cat}$ is the $\mathscr{V}$-cat-ification of $\mathsf{Id} : \mathsf{Set} \longrightarrow \mathsf{Set}$. Thus by [12, Theorem 5.1],

▶ **Proposition 3.6.** The $\mathscr{V}$-cat-functor $D^{\mathscr{V}} : \mathsf{Set} \longrightarrow \mathscr{V}\text{-}\mathsf{cat}$ is dense.

Corollary 3.5, together with the proof of Theorem 3.3 (see the above remark), give us a recipe of how to compute various $\mathscr{V}$-cat-ifications.

▶ **Examples 3.7** (The $\mathscr{V}$-cat-ification of polynomial functors).

1. Let $T : \mathsf{Set} \longrightarrow \mathsf{Set}$, $TX = S$ be a constant functor. Then $T_{\mathscr{V}}$ is again constant, where $T_{\mathscr{V}}\mathscr{X} = D^{\mathscr{V}}S$ for any $\mathscr{V}$-category $\mathscr{X}$.
2. Let $T : \mathsf{Set} \longrightarrow \mathsf{Set}$ be the functor $TX = X^n$, for n a natural number. Then $T_{\mathscr{V}}$ maps a $\mathscr{V}$-category $\mathscr{X}$ to its n-th power $\mathscr{X}^n$, where an easy computation shows
$$\mathscr{X}^n((x'_0, \ldots, x'_{n-1}), (x_0, \ldots, x_{n-1})) = \mathscr{X}(x'_0, x_0) \wedge \cdots \wedge \mathscr{X}(x'_{n-1}, x_{n-1}).$$
3. If n is an *arbitrary* cardinal number, the $\mathscr{V}$-cat-ification $T_{\mathscr{V}}$ of $T : \mathsf{Set} \longrightarrow \mathsf{Set}$, $TX = X^n$ also exists and $T_{\mathscr{V}}\mathscr{X}((x'_i), (x_i)) = \bigwedge_i \mathscr{X}(x'_i, x_i)$. That is, $T_{\mathscr{V}}\mathscr{X} = \mathscr{X}^n$.
4. The $\mathscr{V}$-cat-ification of a finitary polynomial functor $X \mapsto \coprod_n X^n \bullet \Sigma n$ is the "strongly polynomial" $\mathscr{V}$-cat-functor $\mathscr{X} \mapsto \coprod_n \mathscr{X}^n \otimes D^{\mathscr{V}}\Sigma n$, where n ranges through finite sets.

▶ **Example 3.8** (The $\mathscr{V}$-cat-ification of the powerset). Let $P : \mathsf{Set} \longrightarrow \mathsf{Set}$ be the powerset functor. By Theorem 3.3 and Corollary 3.5, its $\mathscr{V}$-cat-ification $P_{\mathscr{V}}$ is defined as follows. Let $\mathscr{X}$ be any small $\mathscr{V}$-category. Then the objects of $P_{\mathscr{V}}\mathscr{X}$ are subsets of the set of objects of $\mathscr{X}$, while the $\mathscr{V}$-"distances" in $P_{\mathscr{V}}\mathscr{X}$ are computed as follows:

$$\begin{aligned} P_{\mathscr{V}}\mathscr{X}(A', A) &= \bigvee_s \{s \mid \text{ there is } B \text{ in } PX_s \text{ s.t. } Pd_0^s(B) = A' \text{ and } Pd_1^s(B) = A\} \\ &= \bigvee_s \{s \mid \ \forall x' \in A'\ \exists x \in A.\, s \le \mathscr{X}(x', x) \text{ and } \forall x \in A\ \exists x' \in A'.\, s \le \mathscr{X}(x', x)\ \} \end{aligned}$$

If the quantale $\mathscr{V}$ is constructively completely distributive [7, 19], as it is the case with $\mathscr{V} = [0,1]$ and $\mathscr{V} = [0,\infty]$, then the above is equivalent to the following:

$$\sup\{ \sup_{x' \in A'} \inf_{x \in A} \mathscr{X}(x',x) \ , \ \sup_{x \in A} \inf_{x' \in A'} \mathscr{X}(x',x)\} \tag{7}$$

where we switched notation to the dual order (that is, the natural "less-or-equal" order in case of reals). So we write inf for $\bigvee$ and sup for $\bigwedge$, in order to emphasise the interpretation of $\mathscr{V}$-cat as metric spaces.

Recall that this metric is known as the Pompeiu-Hausdorff metric ([9, §28], [15, §21]).

We should mention also the connection with the work of [1]. Finally, observe that in case $\mathscr{V} = \mathbb{2}$ (ie $\mathscr{V}$-cat = Preord), the above specializes to the locally monotone functor $P_{\mathbb{2}}$: Preord $\longrightarrow$ Preord which sends a preorder $(X, \leq)$ to the Egli-Milner preorder

$$A' \sqsubseteq A \quad \text{iff} \quad \forall\, x' \in A' \ \exists\, x \in A.\, x' \leq x \text{ and } \forall\, x' \in A \ \exists\, x \in A'.\, x' \leq x$$

on the powerset PX.

▶ **Remark 3.9.** The $\mathscr{V}$-cat-functor $D^{\mathscr{V}}$: Set $\longrightarrow$ $\mathscr{V}$-cat preserves conical colimits. This follows from the $D^{\mathscr{V}}_o$ being an *ordinary* left adjoint. However, the $\mathscr{V}$-cat-functor $D^{\mathscr{V}}$: Set $\longrightarrow$ $\mathscr{V}$-cat is not a left $\mathscr{V}$-cat-*adjoint*, as its ordinary right adjoint functor $V^{\mathscr{V}}$ cannot be extended to a $\mathscr{V}$-cat-functor.

▶ **Proposition 3.10.** *The assignment* $(-)_{\mathscr{V}}$: [Set, Set] $\longrightarrow$ [$\mathscr{V}$-cat, $\mathscr{V}$-cat], $T \mapsto T_{\mathscr{V}}$ *of the* $\mathscr{V}$-cat-*ification preserves all colimits preserved by* $D^{\mathscr{V}}$: Set $\longrightarrow$ $\mathscr{V}$-cat. *In particular,* $T \mapsto T_{\mathscr{V}}$ *preserves conical colimits.*

Proof. Any natural transformation $\tau : T \longrightarrow S$ induces a $\mathscr{V}$-cat-natural transformation

$$(\tau_{\mathscr{V}})_{\mathscr{X}} = N * (D^{\mathscr{V}} \tau D_{\mathscr{X}}) : N * (D^{\mathscr{V}} T D_{\mathscr{X}}) \longrightarrow N * (D^{\mathscr{V}} S D_{\mathscr{X}})$$

Since any colimit is cocontinuous in its weight and since

$$N * (D^{\mathscr{V}} T D_{\mathscr{X}}) \cong (D^{\mathscr{V}} T D_{\mathscr{X}}) * N$$

holds, the assignment $T \mapsto T_{\mathscr{V}}$ preserves all colimits that are preserved by $D^{\mathscr{V}}$: Set $\longrightarrow$ $\mathscr{V}$-cat. The last statement follows from Remark 3.9. ◀

▶ **Corollary 3.11.** *Suppose that the coequalizer*

$$\begin{tikzcd} T_\Gamma \arrow[r, shift left, "\lambda"] \arrow[r, shift right, "\rho"'] & T_\Sigma \arrow[r, "\gamma"] & T \end{tikzcd}$$

is the equational presentation of a λ*-accessible functor* T : Set $\longrightarrow$ Set. *Then the* $\mathscr{V}$-cat-*ification* $T_{\mathscr{V}}$ *can be obtained as the coequalizer*

$$\begin{tikzcd} (T_\Gamma)_{\mathscr{V}} \arrow[r, shift left, "\lambda_{\mathscr{V}}"] \arrow[r, shift right, "\rho_{\mathscr{V}}"'] & (T_\Sigma)_{\mathscr{V}} \arrow[r, "\gamma_{\mathscr{V}}"] & T_{\mathscr{V}} \end{tikzcd}$$

in [$\mathscr{V}$-cat, $\mathscr{V}$-cat].

Proof. A coequalizer is a conical colimit. Now use Proposition 3.10. ◀

▶ **Remark 3.12** (The $\mathscr{V}$-cat-ification of finitary functors)**.** Corollary 3.11 allows us to say that the $\mathscr{V}$-cat-ification $T_{\mathscr{V}}$ of a finitary functor T is given by imposing the "same" operations and equations in $\mathscr{V}$-cat.

Intuitively, the endofunctors on $\mathscr{V}$-cat that arise as left Kan extensions along the discrete functor D^V are the $\mathscr{V}$-cat-endofunctors definable in "discrete arities". This statement will be made formal in future work, here we restrict ourselves to a basic example.

▶ **Example 3.13.** Consider a set A and the associate stream functor $T : \mathsf{Set} \longrightarrow \mathsf{Set}$, $TX = X \times A$. If A carries the additional structure of a $\mathscr{V}$-category (that, is, there is a $\mathscr{V}$-category $\mathscr{A}$ with underlying set of objects A), then T_o can be written as the composite $V^{\mathscr{V}} H$, where $H : \mathsf{Set} \longrightarrow \mathscr{V}\text{-}\mathsf{cat}$ is the $\mathscr{V}$-cat-functor $HX = D^{\mathscr{V}} X \otimes \mathscr{A}$. Now it is immediate to see that the latter extends to the stream functor $H^\sharp$ on $\mathscr{V}$-cat over the "generalized metric space" $\mathscr{A}$, mapping a $\mathscr{V}$-category $\mathscr{X}$ to the tensor product of $\mathscr{V}$-categories $H^\sharp \mathscr{X} = \mathscr{X} \otimes \mathscr{A}$.

The above example is typical. It happens quite often for endofunctors on Set to carry an interesting $\mathscr{V}$-metric where TX is a $\mathscr{V}$-category rather than a mere set, for every X, and this structure is compatible with substitution. The following generalizes the notion of an order on a functor [11] from $\mathscr{V} = \mathbb{2}$.

▶ **Definition 3.14.** Let $T : \mathsf{Set} \longrightarrow \mathsf{Set}$ be a functor. We say that T *carries a $\mathscr{V}$-metric* if there is a $\mathscr{V}$-cat-functor $H : \mathsf{Set} \longrightarrow \mathscr{V}\text{-}\mathsf{cat}$ such that T coincides with the composite

$$\mathsf{Set}_o \xrightarrow{H_o} \mathscr{V}\text{-}\mathsf{cat}_o \xrightarrow{V^{\mathscr{V}}} \mathsf{Set}_o \ .$$

Let T and H be as in the above definition. How are T and $H^\sharp$, the left Kan extension of H along $D^{\mathscr{V}}$ as provided by Theorem 3.3, related? As $D^{\mathscr{V}}$ is fully faithful, the unit $H \longrightarrow H^\sharp D^{\mathscr{V}}$ of the left Kan extension is a $\mathscr{V}$-cat-natural isomorphism. Hence $T_o = V^{\mathscr{V}} H_o \cong V^{\mathscr{V}} H^\sharp D^{\mathscr{V}}$; using now the counit of the ordinary adjunction $D_o^{\mathscr{V}} \dashv V^{\mathscr{V}}$, we obtain an ordinary natural transformation

$$\beta : T_o V^{\mathscr{V}} \longrightarrow V^{\mathscr{V}} H_o^\sharp : \mathscr{V}\text{-}\mathsf{cat}_o \longrightarrow \mathsf{Set}_o.$$

▶ **Proposition 3.15.** *The natural transformation β is component-wise bijective.*

Consequently, $H^\sharp$ is a lifting of T to $\mathscr{V}$-cat.

▶ **Example 3.16** (The Kantorovich lifting)**.** Let $T : \mathsf{Set} \longrightarrow \mathsf{Set}$ be a functor and let $\heartsuit : T\mathscr{V} \longrightarrow \mathscr{V}$ be a map (a $\mathscr{V}$-valued predicate lifting), where by slight abuse we identify the quantale with its underlying set of elements. We ask for $\heartsuit$ to be $\mathscr{V}$-monotone, in the following sense: for every set X and maps $h, k : X \longrightarrow \mathscr{V}$, the inequality

$$\bigwedge_{x \in X} [h(x), k(x)] \leq \bigwedge_{A \in TX} [\heartsuit(T(h)(A)), \heartsuit(T(k)(A))]$$

should hold.[4] Using the $\mathscr{V}$-valued predicate lifting $\heartsuit$, we can endow T with a $\mathscr{V}$-metric as follows: for each set X, put HX to be the $\mathscr{V}$-category with set of objects TX, and $\mathscr{V}$-distances

$$(HX)(A', A) = \bigwedge_{h : X \longrightarrow \mathscr{V}} [\heartsuit(T(h)(A')), \heartsuit(T(h)(A))]$$

where A', A are elements of TX. For a function $f : X \longrightarrow Y$, we let Hf act as Tf on objects. It is easy to see that the above defines indeed a $\mathscr{V}$-metric for T, that is, a $\mathscr{V}$-cat-functor $H : \mathsf{Set} \longrightarrow \mathscr{V}\text{-}\mathsf{cat}$ (the $\mathscr{V}$-cat-enrichment being a consequence of Set being free as a $\mathscr{V}$-cat-category) with $V^{\mathscr{V}} H_o = T$. The corresponding lifting $H^\sharp$ specializes to the Kantorovich

[4] This generalizes the notion of a monotone predicate lifting from the two-elements quantale to arbitrary $\mathscr{V}$, see [3, Section 7].

lifting as defined in [4] in case $\mathscr{V} = [0, \infty]$. Explicitly, a $\mathscr{V}$-category $\mathscr{X}$ gets mapped to the small $\mathscr{V}$-category $H^\sharp \mathscr{X}$ with set of objects TX_Ω and $\mathscr{V}$-homs

$$H^\sharp \mathscr{X}(A', A) = \bigwedge_{h:\mathscr{X} \longrightarrow \mathscr{V}} [\heartsuit(T(h_\Omega)(A')), \heartsuit(T(h_\Omega)(A))]$$

for every A', A in TX_Ω, where this time h ranges over $\mathscr{V}$-functors.

4 Relating behaviours across different base categories

In the previous section, we have shown that every $\mathscr{V}$-cat-functor $H : \mathsf{Set} \longrightarrow \mathscr{V}\text{-}\mathsf{cat}$ has a left Kan extension along $D^{\mathscr{V}}$, denoted $H^\sharp$. Now, each such functor induces a set-endofunctor simply by forgetting the $\mathscr{V}$-cat-structure

$$\begin{tikzcd} \mathsf{Set}_o \arrow[r, "H_o"] & \mathscr{V}\text{-}\mathsf{cat} \arrow[r, "V^{\mathscr{V}}"] & \mathsf{Set}_o \end{tikzcd}$$

In the special case when H is $D^{\mathscr{V}}T$, the above composite gives back T, and $H^\sharp$ is $T_{\mathscr{V}}$, the $\mathscr{V}$-cat-ification of T.

We plan to see how the corresponding behaviors are related. In particular, we show that if $T_{\mathscr{V}}$ is the $\mathscr{V}$-cat-ification of $T : \mathsf{Set} \longrightarrow \mathsf{Set}$, then $T_{\mathscr{V}}$-behaviour and T-behaviour coincide under some conditions imposed on the base quantale $\mathscr{V}$. This requires comparing behaviours across different base categories.

▶ **Remark 4.1.** For each quantale $\mathscr{V}$, the inclusion (quantale morphism) $\mathsf{d} : \mathbb{2} \longrightarrow \mathscr{V}$ given by $0 \mapsto 0, 1 \mapsto e$ has a right adjoint (as it preserves suprema), denoted $\mathsf{v} : \mathscr{V} \longrightarrow \mathbb{2}$ which maps an element r of $\mathscr{V}$ to 1 if $e \leq r$, and to 0 otherwise.[5]

This induces as usual the *change-of-base* adjunction (even a 2-adjunction, see [5])

$$\begin{tikzcd} \mathbb{2} \arrow[r, bend left, "\mathsf{d}"] \arrow[r, phantom, "\perp"] & \mathscr{V} \arrow[l, bend left, "\mathsf{v}"] \end{tikzcd} \qquad \mapsto \qquad \begin{tikzcd} \mathsf{Preord} \arrow[r, bend left, "\mathsf{d}_*"] \arrow[r, phantom, "\perp"] & \mathscr{V}\text{-}\mathsf{cat} \arrow[l, bend left, "\mathsf{v}_*"] \end{tikzcd}$$

Explicitly, the functor d_* maps a preordered set X to the $\mathscr{V}$-category d_*X with same set of objects, and $\mathscr{V}$-homs given by $\mathsf{d}_*X(x', x) = e$ if $x' \leq x$, and $\perp$ otherwise. Its right adjoint transforms a $\mathscr{V}$-category $\mathscr{X}$ into the preorder $\mathsf{v}_*\mathscr{X}$ with same objects again, and order $x' \leq x$ iff $e \leq \mathscr{X}(x', x)$ holds. Hence d_*X is the free $\mathscr{V}$-category on the preorder X, while $v_*\mathscr{X}$ is the underlying ordinary category (which happens to be a preorder, due to simple nature of quantales) of the $\mathscr{V}$-category $\mathscr{X}$.

Note that d_* is both a $\mathscr{V}$-cat-functor and a Preord-functor, while its right adjoint v_* (in fact, the whole adjunction $\mathsf{d}_* \dashv \mathsf{v}_*$) is only Preord-enriched.

In case $\mathscr{V}$ is nontrivial, and e and $\top$ coincide (the quantale is *integral*), the embedding $\mathsf{d} : \mathbb{2} \longrightarrow \mathscr{V}$ has also a left adjoint $\mathsf{c} : \mathscr{V} \longrightarrow \mathbb{2}$, given by $\mathsf{c}(r) = 0$ iff $r = \perp$, otherwise $\mathsf{c}(r) = 1$. Notice that c is only a colax morphism of quantales, in the sense that $\mathsf{c}(e) \leq 1$ (in fact, here we have equality!) and $\mathsf{c}(r \otimes s) \leq \mathsf{c}(r) \wedge \mathsf{c}(s)$, for all r, s in $\mathscr{V}$.

We shall in the sequel assume that c is actually a morphism of quantales. The reader can check that this boils down to the requirement that $r \otimes s = \perp$ in $\mathscr{V}$ implies $r = \perp$ or $s = \perp$. That is, the quantale has no *zero divisors*. All our examples satisfy this assumption.

[5] Notice that v is only a *lax* morphism of quantales, being right adjoint.

If this is the case, d_* also has a left adjoint c_* mapping a $\mathscr{V}$-category $\mathscr{X}$ to the preorder $\mathsf{c}_*\mathscr{X}$ with same objects, such that $x' \leq x$ iff $\mathscr{X}(x', x) \neq \bot$, and the adjunction $\mathsf{c}_* \dashv d_*$ is $\mathscr{V}$-cat-enriched:

$$\begin{tikzcd}
2 \arrow[r, <-, bend left, "c"] \arrow[r, phantom, "\perp"] \arrow[r, bend right, "d"'] & \mathscr{V} & \mapsto & \mathsf{Preord} \arrow[r, <-, bend left, "c_*"] \arrow[r, phantom, "\perp"] \arrow[r, bend right, "d_*"'] & \mathscr{V}\text{-}\mathsf{cat}
\end{tikzcd}$$

From the above remark we obtain the following:

▶ **Proposition 4.2.** *Let $\mathscr{V}$ be an arbitrary quantale and let $\widehat{T} : \mathsf{Preord} \longrightarrow \mathsf{Preord}$ be a locally monotone functor (that is, Preord-enriched) and $\overline{T} : \mathscr{V}\text{-}\mathsf{cat} \longrightarrow \mathscr{V}\text{-}\mathsf{cat}$ be a lifting of $\widehat{T}$ to $\mathscr{V}$-cat (meaning that $\overline{T}$ is $\mathscr{V}$-cat-functor such that $\mathsf{v}_*\overline{T} \cong \widehat{T}\mathsf{v}_*$ holds). Then the locally monotone adjunction $\mathsf{d}_* \dashv \mathsf{v}_*$ lifts to a locally monotone adjunction $\widetilde{\mathsf{d}}_* \dashv \widetilde{\mathsf{v}}_*$ between the associated Preord-categories of coalgebras.*

$$\begin{tikzcd}
\mathsf{Coalg}(\widehat{T}) \arrow[r, bend left, "\widetilde{\mathsf{d}}_*"] \arrow[r, phantom, "\perp"] \arrow[r, <-, bend right, "\widetilde{\mathsf{v}}_*"'] \arrow[d] & \mathsf{Coalg}(\overline{T}) \arrow[d] \\
\mathsf{Preord} \arrow[r, bend left, "\mathsf{d}_*"] \arrow[r, phantom, "\perp"] \arrow[r, <-, bend right, "\mathsf{v}_*"'] & \mathscr{V}\text{-}\mathsf{cat}
\end{tikzcd}$$

▶ **Proposition 4.3.** *Assume now that $\mathscr{V}$ is a non-trivial integral quantale without zero divisors. Let again $\widehat{T} : \mathsf{Preord} \longrightarrow \mathsf{Preord}$ be a locally monotone functor, but this time consider $\overline{T} : \mathscr{V}\text{-}\mathsf{cat} \longrightarrow \mathscr{V}\text{-}\mathsf{cat}$ be an extension of $\widehat{T}$ to $\mathscr{V}$-cat (meaning that $\overline{T}$ is a $\mathscr{V}$-cat-functor, such that $\overline{T}\mathsf{d}_* \cong \widehat{T}\mathsf{d}_*$ holds). Then the $\mathscr{V}$-cat-adjunction $\mathsf{c}_* \dashv \mathsf{d}_*$ lifts to a $\mathscr{V}$-cat-adjunction $\widetilde{\mathsf{c}}_* \dashv \widetilde{\mathsf{d}}_*$ between the associated $\mathscr{V}$-cat-categories of coalgebras.*

$$\begin{tikzcd}
\mathsf{Coalg}(\widehat{T}) \arrow[r, <-, bend left, "\widetilde{\mathsf{c}}_*"] \arrow[r, phantom, "\perp"] \arrow[r, bend right, "\widetilde{\mathsf{d}}_*"'] \arrow[d] & \mathsf{Coalg}(\overline{T}) \arrow[d] \\
\mathsf{Preord} \arrow[r, <-, bend left, "\mathsf{c}_*"] \arrow[r, phantom, "\perp"] \arrow[r, bend right, "\mathsf{d}_*"'] & \mathscr{V}\text{-}\mathsf{cat}
\end{tikzcd}$$

We come back now to the discrete functor $D^{\mathscr{V}} : \mathsf{Set} \longrightarrow \mathscr{V}\text{-}\mathsf{cat}$. It is easy to see that it decomposes as $\mathsf{d}_*D : \mathsf{Set} \to \mathsf{Preord} \to \mathscr{V}\text{-}\mathsf{cat}$. Additionally, recall the following (see also Example 3.2.2):

1. There are locally monotone functors $D : \mathsf{Set} \longrightarrow \mathsf{Preord}$, $C : \mathsf{Preord} \longrightarrow \mathsf{Set}$, where D maps a set to its discrete preorder and C maps a preorder to its set of connected components.

2. There is a chain $C_o \dashv D_o \dashv V : \mathsf{Preord} \longrightarrow \mathsf{Set}$ of ordinary adjunctions where V is the underlying-set forgetful functor.

3. The locally monotone adjunction $C \dashv D$ is $\mathscr{V}$-cat-enriched.

▶ **Lemma 4.4** ([2]). *Let $T : \mathsf{Set} \longrightarrow \mathsf{Set}$ and $\widehat{T} : \mathsf{Preord} \longrightarrow \mathsf{Preord}$ an extension of T (a locally monotone functor such that $DT \cong \widehat{T}D$). Then the locally monotone adjunction $C \dashv D$ lifts to a locally monotone adjunction $\widetilde{C} \dashv \widetilde{D}$ between the associated categories of coalgebras:*

$$\begin{tikzcd}
\mathsf{Coalg}(T) \arrow[r, <-, bend left, "\widetilde{C}"] \arrow[r, phantom, "\perp"] \arrow[r, bend right, "\widetilde{D}"'] \arrow[d] & \mathsf{Coalg}(\widehat{T}) \arrow[d] \\
\mathsf{Set} \arrow[r, <-, bend left, "C"] \arrow[r, phantom, "\perp"] \arrow[r, bend right, "D"'] & \mathsf{Preord}
\end{tikzcd}$$

Consequently, $\widetilde{D}$ will preserve limits, in particular, the final coalgebra (if it exists).

▶ **Lemma 4.5** ([2]). *Let T : Set $\longrightarrow$ Set and $\widehat{T}$: Preord $\longrightarrow$ Preord a lifting of T (an ordinary functor such that $TV \cong V\widehat{T}$). Then the ordinary adjunction $D_o \dashv V$ lifts to an ordinary adjunction $\widetilde{D}_o \dashv \widetilde{V}$ between the associated categories of coalgebras.*

$$\begin{tikzcd}
\mathsf{Coalg}(\widehat{T}) \arrow[r, bend left, "\widetilde{D}_o"] \arrow[r, phantom, "\perp"] \arrow[d] & \mathsf{Coalg}(T_{\mathcal{V}}) \arrow[l, bend left, "\widetilde{V}"] \arrow[d] \\
\mathsf{Set} \arrow[r, bend left, "D_o"] \arrow[r, phantom, "\perp"] & \mathsf{Preord} \arrow[l, bend left, "V"]
\end{tikzcd}$$

Consequently, $\widetilde{V}$ will preserve limits; in particular, the underlying set of the final $\widehat{T}$-coalgebra (if it exists) will be the final T-coalgebra.

▶ **Remark 4.6.** We have shown in the previous section that $D^{\mathcal{V}} = \mathsf{d}_* D$ is $\mathcal{V}$-cat-dense. Using that D is fully faithful, it follows from [12, Theorem 5.13] that also d_* is $\mathcal{V}$-cat-dense and that $\mathsf{d}_* = \mathsf{Lan}_D(D^{\mathcal{V}})$ holds.

Let T : Set $\longrightarrow$ Set and denote by T_2 is 2-cat-ification, that is, its Preord-ification [3]. Then the $\mathcal{V}$-cat-ification $T_{\mathcal{V}}$ of T can be computed in two stages, as follows:

$$\begin{aligned}
T_{\mathcal{V}} &= \mathsf{Lan}_{D^{\mathcal{V}}}(D^{\mathcal{V}}T) \\
&= \mathsf{Lan}_{(\mathsf{d}_* D)}(\mathsf{d}_* DT) = \mathsf{Lan}_{\mathsf{d}_*}(\mathsf{Lan}_D(\mathsf{d}_* DT)) \text{ by } \quad [12, \text{Theorem } 4.47] \\
&\cong \mathsf{Lan}_{\mathsf{d}_*}(\mathsf{Lan}_D(\mathsf{d}_* T_2 D)) \text{ (because } DT \cong T_2 D) \\
&\cong \mathsf{Lan}_{\mathsf{d}_*}(\mathsf{d}_* T_2) \text{ by } \quad [12, \text{Theorem } 5.29]
\end{aligned}$$

where the last isomorphism holds because the composite $\mathsf{d}_* T_2$ preserves all colimits $\mathsf{Preord}(D-, X) * D$, for X in Preord. To see this, notice first that T_2 does so by construction, while for d_* it follows from being $\mathsf{Lan}_D(D^{\mathcal{V}}) = \mathsf{Lan}_D(\mathsf{d}_* D)$, again using [12, Theorem 5.29].

The above simply says that

> *The $\mathcal{V}$-cat-ification of an endofunctor T of Set can be obtained as taking first the Preord-ification T_2 : Preord $\longrightarrow$ Preord,*[6] *then computing the left Kan extension along d_* : Preord $\longrightarrow$ $\mathcal{V}$-cat of the composite Preord $\xrightarrow{T_2}$ Preord $\xrightarrow{\mathsf{d}_*}$ $\mathcal{V}$-cat .*

Putting things together we now obtain

▶ **Theorem 4.7.** *Let $\mathcal{V}$ be a non-trivial integral quantale without zero divisors, and T : Set $\longrightarrow$ Set an arbitrary endofunctor, with $\mathcal{V}$-cat-ification $T_{\mathcal{V}}$: $\mathcal{V}$-cat $\longrightarrow$ $\mathcal{V}$-cat. Then the $\mathcal{V}$-cat-adjunctions $C \dashv D$: Set $\longrightarrow$ Preord, $\mathsf{c}_* \dashv \mathsf{d}_*$: Preord $\longrightarrow$ $\mathcal{V}$-cat lift to $\mathcal{V}$-cat-adjunctions between the associated $\mathcal{V}$-cat-categories of coalgebras:*

$$\begin{tikzcd}
\mathsf{Coalg}(T) \arrow[r, bend right, "\widetilde{D}"'] \arrow[r, phantom, "\perp"] \arrow[d] & \mathsf{Coalg}(T_2) \arrow[l, bend right, "\widetilde{C}"'] \arrow[r, bend right, "\widetilde{\mathsf{d}_*}"'] \arrow[r, phantom, "\perp"] \arrow[d] & \mathsf{Coalg}(T_{\mathcal{V}}) \arrow[l, bend right, "\widetilde{\mathsf{c}_*}"'] \arrow[d] \\
\mathsf{Set} \arrow[r, bend right, "D"'] \arrow[r, phantom, "\perp"] & \mathsf{Preord} \arrow[l, bend right, "C"'] \arrow[r, bend right, "\mathsf{d}_*"'] \arrow[r, phantom, "\perp"] & \mathcal{V}\text{-}\mathsf{cat} \arrow[l, bend right, "\mathsf{c}_*"']
\end{tikzcd}$$

Since the $\mathcal{V}$-cat-ification $T_{\mathcal{V}}$ of an endofunctor T on Set is supposed to be "T in the world of $\mathcal{V}$-categories", the theorem above confirms the expectation that final $T_{\mathcal{V}}$-coalgebras have a

[6] Which has been considered in [3]; note in particular that T_2 is also a lifting of T to Preord.

discrete metric. In fact, we can say that the final T-coalgebra is the final $T_{\mathscr{V}}$-coalgebra, if we consider $\mathsf{Coalg}(T)$ as a full (enriched-reflective) subcategory of $\mathsf{Coalg}(T_{\mathscr{V}})$.

The next theorem deals with a more general situation where the final metric-coalgebra is the final set-coalgebra with an additional metric. This includes in particular the case where $\overline{T}$ is $H^\sharp$ for some $H : \mathsf{Set} \longrightarrow \mathscr{V}\text{-}\mathsf{cat}$ with $V^{\mathscr{V}} H_o = T_o$.

▶ **Theorem 4.8.** *Let $\mathscr{V}$ be a quantale, $T : \mathsf{Set} \longrightarrow \mathsf{Set}$ be an arbitrary endofunctor, $\widehat{T} : \mathsf{Preord} \longrightarrow \mathsf{Preord}$ a lifting of T to Preord, and $\overline{T} : \mathscr{V}\text{-}\mathsf{cat} \longrightarrow \mathscr{V}\text{-}\mathsf{cat}$ be a lifting of $\widehat{T}$ to $\mathscr{V}\text{-}\mathsf{cat}$. Then the ordinary adjunction $D_o \dashv V : \mathsf{Set} \longrightarrow \mathsf{Preord}$, respectively the Preord-adjunction $\mathsf{d}_* \dashv \mathsf{v}_* : \mathsf{Preord} \longrightarrow \mathscr{V}\text{-}\mathsf{cat}$ lift to adjunctions between the associated $\mathscr{V}\text{-}\mathsf{cat}$-categories of coalgebras:*

$$\begin{tikzcd}
\mathsf{Coalg}(T) \arrow[r, bend left, "\widetilde{D_o}"] \arrow[r, phantom, "\perp"] \arrow[d] & \mathsf{Coalg}(\widehat{T}) \arrow[l, bend left, "\widetilde{V}"] \arrow[r, bend left, "\widetilde{\mathsf{d}_*}"] \arrow[r, phantom, "\perp"] \arrow[d] & \mathsf{Coalg}(\overline{T}) \arrow[l, bend left, "\widetilde{\mathsf{v}_*}"] \arrow[d] \\
\mathsf{Set} \arrow[r, bend left, "D_o"] \arrow[r, phantom, "\perp"] & \mathsf{Preord} \arrow[l, bend left, "V"] \arrow[r, bend left, "\mathsf{d}_*"] \arrow[r, phantom, "\perp"] & \mathscr{V}\text{-}\mathsf{cat} \arrow[l, bend left, "\mathsf{v}_*"]
\end{tikzcd}$$

▶ **Example 4.9.** Recall from Example 3.13 the stream functor $T : \mathsf{Set} \longrightarrow \mathsf{Set}$, $TX = X \times A$, and its lifting $H^\sharp : \mathscr{V}\text{-}\mathsf{cat} \longrightarrow \mathscr{V}\text{-}\mathsf{cat}$, $H^\sharp \mathscr{X} = \mathscr{X} \otimes \mathscr{A}$. Assume that the quantale is integral. Then the final coalgebra is the $\mathscr{V}$-category $\mathscr{A}^{\otimes\infty}$ having streams over A as objects, with $\mathscr{V}$-distances

$$\mathscr{A}^{\otimes\infty}((a_n)_n, (b_n)_n) = \bigwedge_n \{\mathscr{A}(a_0, b_0) \otimes \mathscr{A}(a_1, b_1) \otimes \ldots \otimes \mathscr{A}(a_n, b_n)\}$$

If $\mathscr{V}$ is the real half-line from Example 2.1.2, and $\mathscr{A}$ is the two-elements metric space $\{0, 1\}$ with $\mathscr{V}$-distances $\mathscr{A}(0, 1) = \mathscr{A}(1, 0) = 1$, $\mathscr{A}(0, 0) = \mathscr{A}(1, 1) = 0$, we obtain that the $\mathscr{V}$-distance between two streams is n iff they are different on at most n positions.

5 Conclusions

We showed that every functor $H : \mathsf{Set} \longrightarrow \mathscr{V}\text{-}\mathsf{cat}$ has a left-Kan extension $H^\sharp$, and that the final $H^\sharp$-coalgebra is the final $V^{\mathscr{V}} H_o$-coalgebra equipped with a $\mathscr{V}$-metric. In the case where H takes only discrete values, the final coalgebra is discrete as well.

Acknowledgements. We thank the anonymous referees for valuable comments that improved the presentation of our results.

References

1 A. Akhvlediani, M. M. Clementino and W. Tholen, On the categorical meaning of the Hausdorff and Gromov distances I, *Topology and its Applic.* 157(8) (2010), pp. 1275–1295

2 A. Balan and A. Kurz, Finitary functors: from Set to Preord and Poset. In: A. Corradini et al. (eds.), *CALCO 2011*, LNCS 6859, Springer, Heidelberg (2011), pp. 85–99

3 A. Balan, A. Kurz and J. Velebil, Positive fragments of coalgebraic logics, accepted for publication in *Logic. Meth. Comput. Sci.* (2015), available at `http://arxiv.org/pdf/1402.5922v1.pdf`

4 P. Baldan, F. Bonchi, H. Kerstan and B. König, Behavioral metrics via functor lifting. In: V. Raman and S. P. Suresh (eds.), *FSTTCS2014*, LIPIcs 29 (2014), pp. 403–415

5 S. Eilenberg, and G. M. Kelly, Closed categories. In: S. Eilenberg et al. (eds.), *Proceedings of the Conference on Categorical Algebra*, Springer, Berlin Heidelberg (1966), pp. 421–562

6 B. Flagg and R. Kopperman, Continuity spaces: reconciling domains and metric spaces, *Theoret. Comput. Sci.* 177(1) (1997), pp. 111–138

7 P. Freyd and A. Scedrov. *Categories, Allegories*, North Holland (1990)

8 H. Gaifman, V. Pratt, Partial order models of concurrency and the computation of functions. In: *Proceedings of the Symposium on Logic in Computer Science (LICS'87)*, Ithaca, NY (1987), pp. 72–85

9 F. Hausdorff, *Mengenlehre.* 3rd edition, de Gruyter (1935)

10 D. Hofmann and C. D. Reis, Probabilistic metric spaces as enriched categories, *Fuzzy Sets and Systems* 210 (2013), pp. 1–21

11 J. Hughes and B. Jacobs, Simulations in coalgebra, *Theor. Comput. Sci.* 327(1–2):71–108 (2004)

12 G. M. Kelly, *Basic concepts of enriched category theory*, London Math. Soc. Lecture Notes Series 64, Cambridge Univ. Press (1982) also available as *Repr. Theory Appl. Categ.* 10 (2005)

13 G. M. Kelly and S. Lack, $\mathscr{V}$-cat is locally presentable or bounded if $\mathscr{V}$ is so, *Theory Appl. Categ.* 8(23) (2001), pp. 555–575

14 F. W. Lawvere, Metric spaces, generalized logic, and closed categories, *Rendiconti del Seminario Matematico e Fisico di Milano* XLIII (1973), pp. 135–166, also available as *Repr. Theory Appl. Categ.* 1 (2002), pp. 1–37

15 D. Pompeiu, Sur la continuité des fonctions des variables complexes, *Ann. Fac. Sci. Toulouse* 2(7) (1905), 265–315, available at `http://www.numdam.org/item?id=AFST_1905_2_7_3_265_0`

16 J. J. M. M. Rutten, Elements of generalized ultrametric domain theory, *Theoret. Comput. Sci.* 170 (1996), pp. 349–381

17 J. J. M. M. Rutten. Relators and metric bisimulations (extended abstract). In: B. Jacobs et al. (eds.), *CMCS'98, Electr. Notes Theor. Comput. Sci.* 11 (1998), pp. 1–7

18 J. J. M. M. Rutten, Universal coalgebra: a theory of systems, *Theoret. Comput. Sci.* 249 (2000), pp. 3–80

19 R. Wood, Ordered sets via adjunctions. In: M.-C. Pedicchio and W. Tholen (eds.), *Categorical Foundations.* Cambridge Univ. Press (2004), 5–47

20 J. Worrell, Coinduction for recursive data types: partial order, metric spaces and Ω-categories. In: H. Reichel (ed.), *CMCS'2000, Electr. Notes Theor. Comput. Sci.* 33 (2000), pp. 337–356

21 J. Worrell, On coalgebras and final semantics. PhD thesis, University of Oxford (2000), available at `http://www.cs.ox.ac.uk/people/james.worrell/thesis.ps`

Towards Trace Metrics via Functor Lifting*

Paolo Baldan[1], **Filippo Bonchi**[2], **Henning Kerstan**[3], **and Barbara König**[3]

1 **Dipartimento di Matematica, Università di Padova, Italy**
baldan@math.unipd.it
2 **CNRS, ENS Lyon, Université de Lyon, France**
filippo.bonchi@ens-lyon.fr
3 **Universität Duisburg-Essen, Germany**
{henning.kerstan,barbara_koenig}@uni-due.de

Abstract

We investigate the possibility of deriving metric trace semantics in a coalgebraic framework. First, we generalize a technique for systematically lifting functors from the category **Set** of sets to the category **PMet** of pseudometric spaces, by identifying conditions under which also natural transformations, monads and distributive laws can be lifted. By exploiting some recent work on an abstract determinization, these results enable the derivation of trace metrics starting from coalgebras in **Set**. More precisely, for a coalgebra in **Set** we determinize it, thus obtaining a coalgebra in the Eilenberg-Moore category of a monad. When the monad can be lifted to **PMet**, we can equip the final coalgebra with a behavioral distance. The trace distance between two states of the original coalgebra is the distance between their images in the determinized coalgebra through the unit of the monad. We show how our framework applies to nondeterministic automata and probabilistic automata.

1998 ACM Subject Classification F.3.1 Specifying and Verifying and Reasoning about Programs, D.2.4 Software/Program Verification

Keywords and phrases trace metric, monad lifting, pseudometric, coalgebra

Digital Object Identifier 10.4230/LIPIcs.CALCO.2015.35

1 Introduction

When considering the behavior of state-based system models embodying quantitative information, such as probabilities, time or cost, the interest normally shifts from behavioral equivalences to behavioral distances. In fact, in a quantitative setting, it is often quite unnatural to ask that two systems exhibit exactly the same behavior, while it can be more reasonable to require that the distance between their behaviors is sufficiently small (see, e.g., [10, 7, 23, 1, 5, 6, 8]).

Coalgebras [18] are a well-established abstract framework where a canonical notion of behavioral equivalence can be uniformly derived. The behavior of a system is represented as a coalgebra, namely a map of the form $X \to HX$, where X is a state space and H is a functor that describes the type of computation performed. For instance nondeterministic automata can be seen as coalgebras $X \to 2 \times \mathcal{P}(X)^A$: for any state we specify whether it is final or not, and the set of successors for any given input in A. Under suitable conditions

* Partially supported by the University of Padova project ANCORE and the DFG project BEMEGA. The second author acknowledges the support by project ANR 12IS0 2001 PACE.

6th International Conference on Algebra and Coalgebra in Computer Science (CALCO'15).
Editors: Lawrence S. Moss and Paweł Sobociński; pp. 35–49
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

a final coalgebra exists which can be seen as minimized version of the system, so that two states are deemed equivalent when they correspond to the same state in the final coalgebra.

In a recent paper [2] we faced the problem of devising a framework where, given a coalgebra for an endofunctor H on **Set**, one can systematically derive pseudometrics which measure the behavioral distance of states. A first crucial step is the lifting of H to a functor $\overline{H}$ on **PMet**, the category of pseudometric spaces. In particular, we presented two different approaches which can be viewed as generalizations of the Kantorovich and Wasserstein pseudometrics for probability measures. One can prove that the final coalgebra in **Set** can be endowed with a metric, arising as a solution of a fixpoint equation, turning it into the final coalgebra for the lifting $\overline{H}$. Since any coalgebra $X \to HX$ can be seen as a coalgebra in **PMet** by endowing X with the discrete metric, the unique mapping into the final coalgebra provides a behavioral distance on X.

The canonical notion of equivalence for coalgebras, in a sense, fully captures the behavior of the system as expressed by the functor H. As such, it naturally corresponds to bisimulation equivalences already defined for various concrete formalisms. Sometimes one is interested in coarser equivalences, ignoring some aspects of a computation, a notable example being trace equivalence where the computational effect which is ignored is branching.

In this paper, relying on recent work on an abstract determinization construction for coalgebras in [20, 13, 14], we extend the above framework in order to systematically derive trace metrics. The mentioned work starts from the observation that the distinction between the behavior to be observed and the computational effects that are intended to be hidden from the observer, is sometimes formally captured by splitting the functor H characterizing system computations in two components, a functor F for the observable behavior and a monad T describing the computational effects, e.g., lifting $1 + -$, the powerset functor $\mathcal{P}$ or the distribution functor $\mathcal{D}$ provides partial, nondeterministic or probabilistic computations, respectively. For instance, the functor for nondeterministic automata $2 \times \mathcal{P}(X)^A$ can be seen as the composition of the functor $FX = 2 \times X^A$, describing the transitions, with the powerset monad $T = \mathcal{P}$, capturing nondeterminism. Trace semantics can be derived by viewing a coalgebra $X \to 2 \times \mathcal{P}(X)^A$ as a coalgebra $\mathcal{P}(X) \to 2 \times \mathcal{P}(X)^A$, via a determinization construction. Similarly probabilistic automata can be seen as coalgebras of the form $X \to [0,1] \times \mathcal{D}(X)^A$, yielding coalgebras $\mathcal{D}(X) \to [0,1] \times \mathcal{D}(X)^A$ via determinization.

On this basis, [14] develops a framework for deriving behavioral equivalences which only considers the visible behavior, ignoring the computational effects. The core idea consists in “incorporating” the effect of the monad also in the set of states X, which thus becomes TX, by means of a construction that can be seen as an abstract form of determinization. For functors of the shape FT, this can be done by lifting F to a functor $\widehat{F}$ in $\mathcal{EM}(T)$, the Eilenberg-Moore category of T, using a distributive law between F and T. In fact, the final F-coalgebra lifts to the final $\widehat{F}$-coalgebra in $\mathcal{EM}(T)$. The technique works, at the price of some complications, also for functors of the shape TF [14].

Here, we exploit the results in [14] for systematically deriving metric trace semantics for **Set**-based coalgebras. The situation is summarized in the diagram at the end of Subsection 5.1. As a first step, building on our technique for lifting functors from the category **Set** of sets to the category **PMet** of pseudometric spaces, we identify conditions under which also natural transformations, monads and distributive laws can be lifted. In this way we obtain an adjunction between **PMet** and $\mathcal{EM}(\overline{T})$, where $\overline{T}$ is the lifted monad. Via the lifted distributive law we can transfer a functor $\overline{F}$: **PMet** $\to$ **PMet** to an endofunctor $\widehat{\overline{F}}$ on $\mathcal{EM}(\overline{T})$. By using the trivial discrete distance, coalgebras of the form $TX \to FTX$ can now live in $\mathcal{EM}(\overline{T})$ and can be equipped with a trace distance via a map into the final coalgebra. This final coalgebra is again obtained by lifting the final $\overline{F}$-coalgebra, i.e. a coalgebra equipped with a behavioral distance, to $\mathcal{EM}(\overline{T})$.

The trace distance between two states of the original coalgebra can then be defined as the distance between their images in the determinized coalgebra through the unit of the monad. We illustrate our framework by thoroughly discussing two running examples, namely nondeterministic automata and probabilistic automata. We show that it allows us to recover known or meaningful trace distances such as the standard ultrametric on word languages for nondeterministic automata or the total variation distance on distributions for probabilistic automata.

The paper is structured as follows. In Section 2 we will introduce our notation and quickly recall the basics of our lifting framework from [2]. Then, in Section 3, we tackle the question of compositionality, i.e. we investigate whether based on liftings of two functors we can obtain a lifting of the composed functor. The lifting of natural transformations and monads is treated in Section 4. Equipped with these tools, we show as main result in Section 5 how to obtain trace pseudometrics in the Eilenberg-Moore category of a lifted monad. We conclude our paper with a discussion on related and future work (Section 6). Proofs can be found in the extended version [`arXiv:1505.08105`].

2 Preliminaries

In this section we recap some basic notions and fix the corresponding notation. We also briefly recall the results in [2] which will be exploited in the paper.

We assume that the reader is familiar with the basic notions of category theory, especially with the definitions of functor, product, coproduct and weak pullbacks.

For a function $f\colon X \to Y$ and sets $A \subseteq X$, $B \subseteq Y$ we write $f[A] := \{f(a) \mid a \in A\}$ for the *image* of A and $f^{-1}[B] = \{x \in X \mid f(x) \in B\}$ for the *preimage* of B. Finally, if $Y \subseteq [0,\infty]$ and $f, g\colon X \to Y$ are functions we write $f \leq g$ if $f(x) \leq g(x)$ for all $x \in X$.

A *probability distribution* on a given set X is a function $P\colon X \to [0,1]$ satisfying $\sum_{x\in X} P(x) = 1$. For any set $B \subseteq X$ we define $P(B) = \sum_{x \in B} P(x)$. The *support* of P is the set $\mathrm{supp}(P) := \{x \in X \mid P(x) > 0\}$.

Given a natural number $n \in \mathbb{N}$ and a family $(X_i)_{i=1}^n$ of sets X_i we denote the projections of the (cartesian) product of the X_i by $\pi_i\colon \prod_{i=1}^n X_i \to X_i$. For a source $(f_i\colon X \to X_i)_{i=1}^n$ we denote the unique mediating arrow to the product by $\langle f_1, \dots, f_n\rangle\colon X \to \prod_{i=1}^n X_i$. Similarly, given a family of arrows $(f_i\colon X_i \to Y_i)_{i=1}^n$, we write $f_1 \times \cdots \times f_n = \langle f_1 \circ \pi_1, \dots, f_n \circ \pi_n\rangle\colon \prod_{i=1}^n X_i \to \prod_{i=1}^n Y_i$.

For $\top \in (0,\infty]$ and a set X we call any function $d\colon X^2 \to [0,\top]$ a ($\top$-)*distance* on X (for our examples we will use $\top = 1$ or $\top = \infty$). Whenever d satisfies, for all $x, y, z \in X$, $d(x,x) = 0$ (reflexivity), $d(x,y) = d(y,x)$ (symmetry) and $d(x,y) \leq d(x,z) + d(z,y)$ (triangle inequality) we call it a *pseudometric* and if it additionally satisfies $d(x,y) = 0 \implies x = y$ we call it a *metric*. Given such a function d on a set X, we say that (X,d) is a pseudometric/metric space. By $d_e\colon [0,\top]^2 \to [0,\top]$ we denote the ordinary Euclidean distance on $[0,\top]$, i.e., $d_e(x,y) = |x-y|$ for $x, y \in [0,\top] \setminus \{\infty\}$, and – where appropriate – $d_e(x,\infty) = \infty$ if $x \neq \infty$ and $d_e(\infty,\infty) = 0$. Addition is defined in the usual way, in particular $x + \infty = \infty$ for $x \in [0,\infty]$. We call a function $f\colon X \to Y$ between pseudometric spaces (X, d_X) and (Y, d_Y) *nonexpansive* and write $f\colon (X, d_X) \xrightarrow{1} (Y, d_Y)$ if $d_Y \circ (f \times f) \leq d_X$. If equality holds we call f an *isometry*.

By choosing a fixed maximal element $\top$ in our definition of distances, we ensure that the set of pseudometrics over a fixed set with pointwise order is a complete lattice (since

$[0, \top]$ is) and we obtain a complete and cocomplete category of pseudometric spaces and nonexpansive functions, which we denote by **PMet**. Given a functor F on **Set**, we aim at constructing a functor $\overline{F}$ on **PMet** which is a lifting of F in the following sense.

▶ **Definition 2.1** (Lifting). Let $U\colon \mathbf{PMet} \to \mathbf{Set}$ be the forgetful functor which maps every pseudometric space to its underlying set. A functor $\overline{F}\colon \mathbf{PMet} \to \mathbf{PMet}$ is called a *lifting* of a functor $F\colon \mathbf{Set} \to \mathbf{Set}$ if it satisfies $U\overline{F} = FU$.

Similarly to predicate lifting of coalgebraic modal logic [19], lifting to **PMet** can be conveniently defined once a suitable (evaluation) function from $F[0, \top]$ to $[0, \top]$ is fixed.

▶ **Definition 2.2** (Evaluation Function & Evaluation Functor). Let F be an endofunctor on **Set**. An *evaluation function* for F is a function $ev_F\colon F[0, \top] \to [0, \top]$. Given such a function, we define the *evaluation functor* to be the endofunctor $\widetilde{F}$ on $\mathbf{Set}/[0, \top]$, the slice category[1] over $[0, \top]$, via $\widetilde{F}(g) = ev_F \circ Fg$ for all $g \in \mathbf{Set}/[0, \top]$. On arrows $\widetilde{F}$ is defined as F.

A first lifting technique leads to what we called the Kantorovich pseudometric, which is the smallest possible pseudometric d^F on FX such that, for all nonexpansive functions $f\colon (X, d) \xrightarrow{1} ([0, \top], d_e)$, also $\widetilde{F}f\colon (FX, d^F) \xrightarrow{1} ([0, \top], d_e)$ is again nonexpansive.

▶ **Definition 2.3** (Kantorovich Pseudometric & Kantorovich Lifting). Let $F\colon \mathbf{Set} \to \mathbf{Set}$ be a functor with an evaluation function ev_F. For every pseudometric space (X, d) the *Kantorovich pseudometric* on FX is the function $d^{\uparrow F}\colon FX \times FX \to [0, \top]$, where for all $t_1, t_2 \in FX$:

$$d^{\uparrow F}(t_1, t_2) := \sup \left\{ d_e \left(\widetilde{F}f(t_1), \widetilde{F}f(t_2) \right) \mid f\colon (X, d) \xrightarrow{1} ([0, \top], d_e) \right\} .$$

The *Kantorovich lifting* of the functor F is the functor $\overline{F}\colon \mathbf{PMet} \to \mathbf{PMet}$ defined as $\overline{F}(X, d) = (FX, d^{\uparrow F})$ and $\overline{F}f = Ff$.

This definition is sound, i.e., $d^{\uparrow F}$ is guaranteed to be a pseudometric so that we indeed obtain a lifting of the functor. A dual way for obtaining a pseudometric on FX relies on ideas from probability and transportation theory. It is based on the notion of couplings, which can be understood as a generalization of joint probability measures.

▶ **Definition 2.4** (Coupling). Let $F\colon \mathbf{Set} \to \mathbf{Set}$ be a functor and $n \in \mathbb{N}$. Given a set X and $t_i \in FX$ for $1 \leq i \leq n$ we call an element $t \in F(X^n)$ such that $F\pi_i(t) = t_i$ a *coupling* of the t_i (with respect to F). We write $\Gamma_F(t_1, t_2, \ldots, t_n)$ for the set of all these couplings.

Based on these couplings we are now able to define an alternative distance on FX.

▶ **Definition 2.5** (Wasserstein Distance & Wasserstein Lifting). Let $F\colon \mathbf{Set} \to \mathbf{Set}$ be a functor with evaluation function ev_F. For every pseudometric space (X, d) the *Wasserstein distance* on FX is the function $d^{\downarrow F}\colon FX \times FX \to [0, \top]$ given by, for all $t_1, t_2 \in FX$,

$$d^{\downarrow F}(t_1, t_2) := \inf \left\{ \widetilde{F}d(t) \mid t \in \Gamma_F(t_1, t_2) \right\} .$$

If $d^{\downarrow F}$ is a pseudometric for all pseudometric spaces (X, d), we define the *Wasserstein lifting* of F to be the functor $\overline{F}\colon \mathbf{PMet} \to \mathbf{PMet}$, $\overline{F}(X, d) = (FX, d^{\downarrow F})$, $\overline{F}f = Ff$.

[1] The slice category $\mathbf{Set}/[0, \top]$ has as objects all functions $g\colon X \to [0, \top]$ where X is an arbitrary set. Given g as before and $h\colon Y \to [0, \top]$, an arrow from g to h is a function $f\colon X \to Y$ satisfying $h \circ f = g$.

The names Kantorovich and Wasserstein used for the liftings derive from transportation theory [25]. Indeed we obtain a transport problem if we instantiate F with the distribution functor $\mathcal{D}$ (see also Theorem 2.9 below). In order to measure the distance between two probability distributions $s, t\colon X \to [0,1]$ it is useful to think of the following analogy: assume that X is a collection of cities (with distance function d between them) and s, t represent supply and demand (in units of mass). The distance between s, t can be measured in two ways: the first is to set up an optimal transportation plan with minimal costs (also called coupling) to transport goods from cities with excess supply to cities with excess demand. The cost of transport is determined by the product of mass and distance. In this way we obtain the Wasserstein distance. A different view is to imagine a logistics firm that is commissioned to handle the transport. It sets prices for each city and buys and sells for this price at every location. However, it has to ensure that the price function (here, f) is nonexpansive, i.e., the difference of prices between two cities is smaller than the distance of the cities, otherwise it will not be worthwhile to outsource this task. This firm will attempt to maximize its profit, which can be considered as the Kantorovich distance of s, t. The Kantorovich-Rubinstein duality informs us that these two views lead to the exactly same result.

In Theorem 2.5 we are not guaranteed, in general, that $d^{\downarrow F}$ is a pseudometric. This is the case if we require F to preserve weak-pullbacks and impose the following restrictions on the evaluation function.

▶ **Definition 2.6** (Well-Behaved). Let F be a functor with an evaluation function ev_F. We call ev_F *well-behaved* if it satisfies the following conditions:
W1. $\widetilde{F}$ is monotone, i.e., for $f, g\colon X \to [0,\top]$ with $f \le g$, we have $\widetilde{F}f \le \widetilde{F}g$.
W2. For each $t \in F([0,\top]^2)$ it holds that $d_e(ev_F(t_1), ev_F(t_2)) \le \widetilde{F}d_e(t)$ for $t_i := F\pi_i(t)$.
W3. $ev_F^{-1}[\{0\}] = Fi[F\{0\}]$ where $i\colon \{0\} \hookrightarrow [0,\top]$ is the inclusion map.

While condition W1 is quite natural, for W2 and W3 some explanations are in order. Condition W2 ensures that $\widetilde{F}\mathrm{id}_{[0,\top]} = ev_F\colon F[0,\top] \to [0,\top]$ is nonexpansive once d_e is lifted to $F[0,\top]$ (recall that for the Kantorovich lifting we require $\widetilde{F}f$ to be nonexpansive for any nonexpansive f). Condition W3 requires that exactly the elements of $F\{0\}$ are mapped to 0 via ev_F. This is necessary for reflexivity of the Wasserstein pseudometric. Indeed, with this definition at hand we were able to prove the desired result.

▶ **Proposition 2.7** ([2]). *If F preserves weak pullbacks and ev_F is well-behaved, then $d^{\downarrow F}$ is a pseudometric for any pseudometric space (X, d).*

From now on, whenever we use the Wasserstein lifting $d^{\downarrow F}$, we implicitly assume to be in the hypotheses of Theorem 2.7. It can be shown that, in general, $d^{\uparrow F} \le d^{\downarrow F}$. Whenever equality holds we say that the functor and the evaluation function satisfy the *Kantorovich-Rubinstein duality.* This is helpful in many situations (e.g., in [24] it allowed to reuse an efficient linear programming algorithm to compute behavioral distance) but it is usually difficult to obtain.

We now recall two examples which will play an important role in this paper. First, we consider the following finitary variant of the powerset functor.

▶ **Example 2.8** (Finite Powerset). The finite powerset functor $\mathcal{P}_{\mathit{fin}}$ assigns to each set X the set $\mathcal{P}_{\mathit{fin}}X = \{S \subseteq X \mid |S| < \infty\}$ and to each function $f\colon X \to Y$ the function $\mathcal{P}_{\mathit{fin}}f\colon \mathcal{P}_{\mathit{fin}}X \to \mathcal{P}_{\mathit{fin}}Y$, $\mathcal{P}_{\mathit{fin}}f(S) := f[S]$. This functor preserves weak pullbacks and the evaluation function $\max\colon \mathcal{P}_{\mathit{fin}}([0,\infty]) \to [0,\infty]$ with $\max\emptyset = 0$ is well-behaved. The Kantorovich-Rubinstein duality holds and the resulting distance is the Hausdorff pseudometric which, for any pseudometric space (X,d) and any $X_1, X_2 \in \mathcal{P}_{\mathit{fin}} X$, is defined as

$$d_H(X_1,X_2) = \max\left\{\max_{x_1\in X_1}\min_{x_2\in X_2} d(x_1,x_2),\ \max_{x_2\in X_2}\min_{x_1\in X_1} d(x_1,x_2)\right\}.$$

Our second example is the following finite variant of the distribution functor.

▶ **Example 2.9** (Finitely Supported Distributions). The probability distribution functor $\mathcal{D}$ assigns to each set X the set $\mathcal{D}X = \{P\colon X \to [0,1] \mid |\mathrm{supp}(P)| < \infty, P(X) = 1\}$ and to each function $f\colon X \to Y$ the function $\mathcal{D}f\colon \mathcal{D}X \to \mathcal{D}Y$, $\mathcal{D}f(P)(y) = \sum_{x\in f^{-1}[\{y\}]} P(x) = P(f^{-1}[\{y\}])$. It preserves weak pullbacks and the evaluation function $ev_{\mathcal{D}}\colon \mathcal{D}[0,1] \to [0,1]$, $ev_{\mathcal{D}}(P) = \sum_{r\in[0,1]} r \cdot P(r)$ is well-behaved. For any pseudometric space (X,d) we obtain the Wasserstein pseudometric which, for any $P_1, P_2 \in \mathcal{D}X$, is defined as

$$d^{\downarrow\mathcal{D}}(P_1,P_2) = \min\left\{\sum_{x_1,x_2\in X} d(x_1,x_2)\cdot P(x_1,x_2) \,\middle|\, P \in \Gamma_{\mathcal{D}}(P_1,P_2)\right\}.$$

The Kantorovich-Rubinstein duality [25] holds from classical results in transportation theory.

While these two functors can be nicely lifted using the theory developed so far, there are other functors that require a more general treatment. For instance, consider the endofunctor $F = B \times \_$ (left product with B) for some fixed B. Notice that for $t_1, t_2 \in FX = B \times X$ with $t_i = (b_i, x_i)$ a coupling exists iff $b_1 = b_2$. As a consequence, when $b_1 \neq b_2$, irrespectively of the evaluation function we choose and of the distance between x_1 and x_2 in (X,d), the lifted Wasserstein pseudometric will always result in $d^{\downarrow F}(t_1,t_2) = \top$. This can be counterintuitive, e.g., taking $B = [0,1]$, $X \neq \emptyset$ and $t_1 = (0,x)$ and $t_2 = (\varepsilon, x)$ for a small $\varepsilon > 0$ and an $x \in X$. The reason is that we think of $B = [0,1]$ as endowed with a non-discrete pseudometric, like e.g. the Euclidean metric d_e, plugged into the product after the lifting. This intuition can be indeed formalized by considering the lifting of the product seen as a functor from **Set** × **Set** into **Set**. More generally, it can be seen that the definitions and results introduced so far for endofunctors on **Set** straightforwardly extend to multifunctors on **Set**, namely functors $F\colon \mathbf{Set}^n \to \mathbf{Set}$ on the product category $\mathbf{Set}^n$ for a natural number $n \in \mathbb{N}$. For ease of presentation we will not spell out the details here (they can be found in [2]), but just provide an important example of a bifunctor (i.e. $n = 2$).

▶ **Example 2.10** (Product Bifunctor). The weak pullback preserving product bifunctor $F\colon \mathbf{Set}^2 \to \mathbf{Set}$ maps two sets X_1, X_2 to $F(X_1,X_2) = X_1 \times X_2$ and two functions $f_i\colon X_i \to Y_i$ to the function $F(f_1,f_2) = f_1 \times f_2$. In this paper we will use the well-behaved evaluation functions $ev_F\colon [0,1]^2 \to [0,1]$ presented in the table below. Therein we also list the pseudometric $(d_1,d_2)^F\colon (X_1 \times X_2)^2 \to [0,\top]$ we obtain for pseudometric spaces (X_1,d_1), (X_2,d_2).

Parameters	$ev_F(r_1,r_2)$	$(d_1,d_2)^F((x_1,x_2),(y_1,y_2))$
$c_1, c_2 \in (0,1]$	$\max\{c_1r_1, c_2r_2\}$	$\max\{c_1d_1(x_1,y_1), c_2d_2(x_2,y_2)\}$
$c_1, c_2 \in (0,1], c_1 + c_2 \leq 1$	$c_1x_1 + c_2x_2$	$c_1d_1(x_1,y_1) + c_2d_2(x_2,y_2)$

For $c_1 = c_2 = 1$, the first evaluation map yields exactly the categorical product in **PMet**. In both cases the Kantorovich-Rubinstein duality holds and the supremum [infimum] of the Kantorovich [Wasserstein] pseudometric is always a maximum [minimum].

3 Compositionality for the Wasserstein Lifting

Our first step is to study compositionality of functor liftings, i.e., we identify some sufficient conditions ensuring $\overline{F}\,\overline{G} = \overline{FG}$. This technical result will be often very useful since it allows us to reason modularly and, consequently, to simplify the proofs needed in the treatment of our examples. We will explicitly only consider the Wasserstein approach which is the one employed in all the examples of this paper.

Given evaluation functions ev_F and ev_G, we can easily construct an evaluation function for the composition FG by defining $ev_{FG} := \widetilde{F}ev_G = ev_F \circ Fev_G$. Our first observation is that, whenever F and G preserve weak pullbacks, well-behavedness is inherited.

▶ **Proposition 3.1** (Well-Behavedness of Composed Evaluation Function). *Let F, G be endofunctors on* **Set** *with evaluation functions ev_F, ev_G. If both functors preserve weak pullbacks and both evaluation functions are well-behaved then also $ev_{FG} = ev_F \circ Fev_G$ is well-behaved.*

In the light of this result and the fact that FG certainly preserves weak pullbacks if F and G do, we can safely use the Wasserstein lifting for FG. A sufficient criterion for compositionality is the existence of optimal couplings for G.

▶ **Proposition 3.2** (Compositionality). *Let F, G be weak pullback preserving endofunctors on* **Set** *with well-behaved evaluation functions ev_F, ev_G and let (X, d) be a pseudometric space. Then $d^{\downarrow FG} \geq (d^{\downarrow G})^{\downarrow F}$. Moreover, if for all $t_1, t_2 \in GX$ there is an optimal G-coupling, i.e. $\gamma(t_1, t_2) \in \Gamma_G(t_1, t_2)$ such that $d^{\downarrow G}(t_1, t_2) = \widetilde{G}d(\gamma(t_1, t_2))$, then $d^{\downarrow FG} = (d^{\downarrow G})^{\downarrow F}$.*

This criterion will turn out to be very useful for our later results. Nevertheless it provides just a sufficient condition for compositionality as the next example shows.

▶ **Example 3.3.** We consider the finite powerset functor $\mathcal{P}_{\mathit{fin}}$ of Theorem 2.8 and the distribution functor $\mathcal{D}$ of Theorem 2.9 with their evaluation functions. Let (X, d) be a pseudometric space.

1. We have $d^{\downarrow \mathcal{D}\mathcal{D}} = \left(d^{\downarrow \mathcal{D}}\right)^{\downarrow \mathcal{D}}$, by Theorem 3.2, because optimal couplings always exist.
2. We have $d^{\downarrow \mathcal{P}_{\mathit{fin}}\mathcal{P}_{\mathit{fin}}} = \left(d^{\downarrow \mathcal{P}_{\mathit{fin}}}\right)^{\downarrow \mathcal{P}_{\mathit{fin}}}$ although $\mathcal{P}_{\mathit{fin}}$-couplings do not always exist.

Note that when we lift the functor $\mathcal{P}_{\mathit{fin}}$ we do not have couplings in the case when we determine the distance between an empty set $\emptyset$ and a non-empty set $Y \subseteq X$, since there exists no subset of $X \times X$ that projects to both.

Compositionality can be defined analogously for multifunctors. Again, we will not spell this out completely but we will use it to obtain the machine bifunctor. Before we can do that, we first need to define another endofunctor.

▶ **Example 3.4** (Input Functor). Let A be a fixed finite set of inputs. The input functor $F = \_^A\colon \mathbf{Set} \to \mathbf{Set}$ maps a set X to the exponential X^A and a function $f\colon X \to Y$ to $f^A\colon X^A \to Y^A$, $f^A(g) = f \circ g$. This functor preserves weak pullbacks. The two evaluation functions listed below are well-behaved and yield the given Wasserstein pseudometric on X^A for any pseudometric space (X, d).

$ev_F(s)$	$d^{\downarrow F}(s_1, s_2)$
$\max_{a \in A} s(a)$	$\max_{a \in A} d\big(s_1(a), s_2(a)\big)$
$\sum_{a \in A} s(a)$	$\sum_{a \in A} d\big(s_1(a), s_2(a)\big)$

By composing this functor with the product bifunctor we obtain the machine bifunctor which we will use to obtain trace semantics.

▶ **Example 3.5** (Machine Bifunctor). Let A be a finite set of inputs, $I = \_^A$ the input functor of Theorem 3.4, Id the identity endofunctor on **Set** and P be the product bifunctor of Theorem 2.10. The *machine bifunctor* is the composition $M := P \circ (\mathrm{Id} \times I)$ i.e. the bifunctor $M\colon \mathbf{Set}^2 \to \mathbf{Set}$ with $M(B, X) := B \times X^A$. Since for Id and I there are unique (thus optimal) couplings we have compositionality. Depending on the choices of evaluation function for P and I (for Id we always take $\mathrm{id}_{[0,1]}$) we obtain the following well-behaved evaluation functions $ev_M\colon [0,1]\times[0,1]^A \to [0,1]$.

Parameters	$ev_P(r_1,r_2)$	$ev_I(s)$	$ev_M(o,s)$
$c_1,c_2\in(0,1]$	$\max\{c_1r_1,c_2r_2\}$	$\max_{a\in A} s(a)$	$\max\left\{c_1o, c_2\max_{a\in A} s(a)\right\}$
$c_1,c_2\in(0,1], c_1+c_2\le 1$	$c_1x_1+c_2x_2$	$\lvert A\rvert^{-1}\sum_{a\in A}s(a)$	$c_1o+c_2\lvert A\rvert^{-1}\sum_{a\in A}s(a)$.

Let (B,d_B), (X,d) be pseudometric spaces. For any $t_1,t_2\in M(B,X)$ with $t_i=(b_i,s_i)\in B\times X^A$ there is a unique and therefore necessarily optimal coupling $t:=(b_1,b_2,\langle s_1,s_2\rangle)$. Depending on the evaluation function, we obtain for the first case $(d_B,d)^{\downarrow M}(t_1,t_2)=\max\{c_1d_B(b_1,b_2),c_2\cdot\max_{a\in A}d(s_1(a),s_2(a))\}$ and for the second case $(d_B,d)^{\downarrow M}(t_1,t_2)=c_1d_B(b_1,b_2)+c_2\lvert A\rvert^{-1}\sum_{a\in A}d(s_1(a),s_2(a))$.

Usually we will fix the first argument (the set of outputs) of the machine bifunctor and consider the obtained machine endofunctor $M_B:=M(B,\_)$. However, for the same reasons as explained above for the product bifunctor, we need to consider it as bifunctor. One notable exception is the case where $B=2$, endowed with the discrete metric. Then we have the following result.

▶ **Example 3.6.** Consider the machine endofunctor $M_2:=M(2,\_)=2\times\_^A$ with evaluation function $ev_{M_2}\colon 2\times[0,1]^A, (o,s)\mapsto c\cdot ev_I(s)$ where $c\in(0,1]$ and ev_I is one of the evaluation functions for the input functor from Theorem 3.4. If d_2 is the discrete metric on 2 and $c=c_2$ (where c_2 is the parameter for the evaluation function of the machine bifunctor as in Theorem 3.5) then the pseudometric obtained via the bifunctor lifting coincides with the one obtained by endofunctor lifting i.e. for all pseudometric spaces (X,d) we have $(d_2,d)^{\downarrow M}=d^{\downarrow M_2}$. Moreover, although couplings for M_2 do not always exist we have $d^{\downarrow \mathcal{P}_{\mathit{fin}} M_2}=\left(d^{\downarrow M_2}\right)^{\downarrow\mathcal{P}_{\mathit{fin}}}$.

4 Lifting of Natural Transformations and Monads

Recall that a monad on an arbitrary category $\mathbf{C}$ is a triple (T,η,μ) where $T\colon\mathbf{C}\to\mathbf{C}$ is an endofunctor and $\eta\colon \mathrm{Id}\Rightarrow T$, $\mu\colon T^2\Rightarrow T$ are natural transformations called *unit* (η) and *multiplication* (μ) such that the two diagrams below commute.

$$\begin{tikzcd}
T \arrow[r, "\eta T"] \arrow[dr, equal] & T^2 \arrow[d, "\mu"] & T \arrow[l, "T\eta"'] \arrow[dl, equal] \\
& T &
\end{tikzcd}
\qquad
\begin{tikzcd}
T^3 \arrow[r, "\mu T"] \arrow[d, "T\mu"'] & T^2 \arrow[d, "\mu"] \\
T^2 \arrow[r, "\mu"] & T
\end{tikzcd}$$

If we have a monad on **Set**, we can of course use our framework to lift the endofunctor T to a functor $\overline{T}$ on pseudometric spaces. A natural question that arises is, whether we also obtain a monad on pseudometric spaces, i.e., if the components of the unit and the multiplication are nonexpansive with respect to the lifted pseudometrics. In order to answer this question, we first take a closer look at sufficient conditions for lifting natural transformations.

▶ **Proposition 4.1** (Lifting of a Natural Transformation). *Let F, G be endofunctors on* **Set** *with evaluation functions ev_F, ev_G and $\lambda\colon F\Rightarrow G$ be a natural transformation. Then the following holds for all pseudometric spaces (X,d). For the Kantorovich lifting:*

1. *If $ev_G \circ \lambda_{[0,\top]} \leq ev_F$ then $d^{\uparrow G} \circ (\lambda_X \times \lambda_X) \leq d^{\uparrow F}$, i.e. λ_X is nonexpansive.*
2. *If $ev_G \circ \lambda_{[0,\top]} = ev_F$ then $d^{\uparrow G} \circ (\lambda_X \times \lambda_X) = d^{\uparrow F}$, i.e. λ_X is an isometry.*

while for the Wasserstein lifting:

3. *If $ev_G \circ \lambda_{[0,\top]} \leq ev_F$ then $d^{\downarrow G} \circ (\lambda_X \times \lambda_X) \leq d^{\downarrow F}$, i.e. λ_X is nonexpansive.*
4. *If $ev_G \circ \lambda_{[0,\top]} = ev_F$ and the Kantorovich Rubinstein duality holds for F, i.e. $d^{\uparrow F} = d^{\downarrow F}$, then $d^{\downarrow G} \circ (\lambda_X \times \lambda_X) = d^{\downarrow F}$, i.e. λ_X is an isometry.*

In the rest of the paper we will call a natural transformation λ nonexpansive [an isometry] if (and only if) each of its components are nonexpansive [isometries] and write $\overline{\lambda}$ for the resulting natural transformation from $\overline{F}$ to $\overline{G}$. Instead of checking nonexpansiveness separately for each component of a natural transformation, we can just check the above (in-)equalities involving the two evaluation functions.

By applying these conditions on the unit and multiplication of a given monad, we can now provide sufficient criteria for a monad lifting.

▶ **Corollary 4.2** (Lifting of a Monad). *Let (T, η, μ) be a* **Set***-monad and ev_T an evaluation function for T. Then the following holds.*

1. *If $ev_T \circ \eta_{[0,\top]} \leq \mathrm{id}_{[0,\top]}$ then η is nonexpansive for both liftings. Hence we obtain the unit $\overline{\eta}\colon \overline{\mathrm{Id}} \Rightarrow \overline{T}$ in* **PMet***.*
2. *If $ev_T \circ \eta_{[0,\top]} = \mathrm{id}_{[0,\top]}$ then η is an isometry for both liftings.*
3. *Let $d^T \in \{d^{\uparrow T}, d^{\downarrow T}\}$. If $ev_T \circ \mu_{[0,\top]} \leq ev_T \circ T ev_T$ and compositionality holds for TT, i.e. $(d^T)^T = d^{TT}$, then μ is nonexpansive, i.e. $d^T \circ (\mu_X \times \mu_X) \leq (d^T)^T$. This yields the multiplication $\overline{\mu}\colon \overline{T}\,\overline{T} \Rightarrow \overline{T}$ in* **PMet***.*

We conclude this section with two examples of liftable monads.

▶ **Example 4.3** (Finite Powerset Monad). The finite powerset functor $\mathcal{P}_{\mathit{fin}}$ of Theorem 2.8 can be seen as a monad, with unit η consisting of the functions $\eta_X\colon X \to \mathcal{P}_{\mathit{fin}}X$, $\eta_X(x) = \{x\}$ and multiplication given by $\mu_X\colon \mathcal{P}_{\mathit{fin}}\mathcal{P}_{\mathit{fin}}X \to \mathcal{P}_{\mathit{fin}}X$, $\mu_X(S) = \cup S$. We show that our conditions for the Wasserstein lifting are satisfied. Given $r \in [0,\infty]$ we have $ev_T \circ \eta_{[0,\infty]}(r) = \max\{r\} = r$ and for $\mathcal{S} \in \mathcal{P}_{\mathit{fin}}(\mathcal{P}_{\mathit{fin}}[0,\top])$ we have $ev_T \circ \mu_{[0,1]}(\mathcal{S}) = \max \cup\mathcal{S} = \max \cup_{S\in\mathcal{S}} S$ and $ev_T \circ T ev_T(\mathcal{S}) = \max(ev_T[\mathcal{S}]) = \max\{\max S \mid S \in \mathcal{S}\}$ and thus both values coincide. Moreover, we recall from Theorem 3.3.2 that we have compositionality for $\mathcal{P}_{\mathit{fin}}\mathcal{P}_{\mathit{fin}}$. Therefore, by Theorem 4.2 η is an isometry and μ nonexpansive.

▶ **Example 4.4** (Distribution Monad). The probability distribution functor $\mathcal{D}$ of Theorem 2.9 can be seen as a monad: the unit η consists of the functions $\eta_X\colon X \to \mathcal{D}X$, $\eta_X(x) = \delta_x^X$ where δ_x^X is the Dirac distribution and the multiplication is given by $\mu_X\colon \mathcal{D}\mathcal{D}X \to \mathcal{D}X$, $\mu_X(P) = \lambda x. \sum_{q\in\mathcal{D}X} P(q)\cdot q(x)$. We consider its Wasserstein lifting. Since $[0,1] = \mathcal{D}2$ we can see that $ev_{\mathcal{D}} = \mu_2$. Using this fact and the monad laws we have $ev_{\mathcal{D}} \circ \eta_{[0,1]} = \mu_2 \circ \eta_{\mathcal{D}2} = \mathrm{id}_{\mathcal{D}X} = \mathrm{id}_{[0,1]}$ and also $ev_{\mathcal{D}} \circ \mu_{[0,1]} = \mu_2 \circ \mu_{\mathcal{D}2} = \mu_2 \circ \mathcal{D}\mu_2 = ev_{\mathcal{D}} \circ \mathcal{D}ev_{\mathcal{D}}$. Moreover, since we always have optimal couplings, we have compositionality for $\mathcal{D}\mathcal{D}$ by Theorem 3.2. Thus by Theorem 4.2 η is an isometry and μ nonexpansive.

5 Trace Metrics in Eilenberg-Moore

As mentioned in the introduction, trace semantics can be characterized by means of coalgebras either over Kleisli [17, 11] or over Eilenberg-Moore [20, 14] categories. We focus on the latter approach. We first recall the basic notions of Eilenberg-Moore algebras and distributive laws, and discuss how the results in the paper can be used to "lift" the associated determinization construction. This is then applied to derive trace metrics for nondeterministic automata and probabilistic automata, by relying on suitable liftings of the machine functor.

5.1 Generalized Powerset Construction

An *Eilenberg-Moore algebra* for a monad (T, η, μ) is a $\mathbf{C}$-arrow $a\colon TA \to A$ making the left and middle diagram below commute. Given two such algebras $a\colon TA \to A$ and $b\colon TB \to B$, a morphism from a to b is a $\mathbf{C}$ arrow $f\colon A \to B$ making the right diagram below commute.

$$\begin{tikzcd}
A \arrow[r, "\eta_A"] \arrow[dr, equal] & TA \arrow[d, "a"] \\
& A
\end{tikzcd}
\qquad
\begin{tikzcd}
T^2A \arrow[r, "\mu_A"] \arrow[d, "Tc"'] & TA \arrow[d, "a"] \\
TA \arrow[r, "a"'] & A
\end{tikzcd}
\qquad
\begin{tikzcd}
TA \arrow[r, "Tf"] \arrow[d, "a"'] & TB \arrow[d, "b"] \\
A \arrow[r, "f"'] & B
\end{tikzcd}$$

Eilenberg-Moore algebras and their morphisms form a category denoted by $\mathcal{EM}(T)$. A functor $\widehat{F}\colon \mathcal{EM}(T) \to \mathcal{EM}(T)$ is called a *lifting* of $F\colon \mathbf{C} \to \mathbf{C}$ to $\mathcal{EM}(T)$ if $U^T\widehat{F} = FU^T$, with $U^T\colon \mathcal{EM}(T) \to \mathbf{C}$ the forgetful functor. A natural transformation $\lambda\colon TF \Rightarrow FT$ is an *$\mathcal{EM}$-law* (also called *distributive law*) if it satisfies:

$$\begin{tikzcd}
F \arrow[r, equal] \arrow[d, "\eta F"'] & F \arrow[d, "F\eta"] \\
TF \arrow[r, "\lambda"'] & FT
\end{tikzcd}
\qquad
\begin{tikzcd}
T^2F \arrow[r, "T\lambda"] \arrow[d, "\mu F"'] & TFT \arrow[r, "\lambda T"] & FT^2 \arrow[d, "F\mu"] \\
TF \arrow[rr, "\lambda"'] & & FT
\end{tikzcd}$$

Liftings and $\mathcal{EM}$-laws are related by the following folklore result (see e.g. [13]).

▶ **Proposition 5.1.** *There is a bijective correspondence between $\mathcal{EM}$-laws and liftings to $\mathcal{EM}$-categories.*

$\mathcal{EM}$-laws and liftings are crucial to characterize trace semantics via coalgebras. Given a coalgebra $c\colon X \to FTX$, for a functor F and a monad (T, η, μ) such that there is a distributive law $\lambda\colon TF \Rightarrow FT$, one can build an F-coalgebra as

$$c^\sharp := \left(TX \xrightarrow{Tc} TFTX \xrightarrow{\lambda_{TX}} FTTX \xrightarrow{F\mu_X} FTX\right)$$

If there exists a final F-coalgebra $\omega\colon \Omega \to F\Omega$, one can define a semantic map for the FT-coalgebra c into Ω. First let $[\![-]\!]\colon TX \to \Omega$ be the unique coalgebra morphism from $c^\sharp$. Then take the map $[\![-]\!] \circ \eta\colon X \to \Omega$.

$$\begin{tikzcd}
X \arrow[r, "\eta"] \arrow[d, "c"'] & TX \arrow[r, "{[\![-]\!]}"] \arrow[dl, "c^\sharp"] & \Omega \arrow[d, "\omega"] \\
FTX \arrow[rr, "F{[\![-]\!]}"'] & & T\Omega
\end{tikzcd}$$

One can readily check that $c^\sharp$ is an algebra map from the T-algebra μ_X to $\widehat{F}\mu_X$, namely it is an $\widehat{F}$-coalgebra or, equivalently, a λ-*bialgebra* [21, 15]. Similarly for ω, Ω carries a T-algebra structure obtained by finality and hence the final F-coalgebra ω can be lifted in order to obtain the final $\widehat{F}$-coalgebra (see [13, Prop. 4]).

This result holds for arbitrary categories and, in particular, we can reuse it for our setting: we only need an $\mathcal{EM}$-law on **PMet**. Note that Theorem 4.1 not only provides sufficient conditions for monad liftings but also can be exploited to lift $\mathcal{EM}$-laws. Indeed the additional commutativity requirements for $\mathcal{EM}$-laws trivially hold when all components are nonexpansive.

▶ **Corollary 5.2** (Lifting of an $\mathcal{EM}$-law). *Let F, G be weak pullback preserving endofunctors on* **Set** *with well-behaved evaluation functions ev_F, ev_G and $\lambda\colon FG \Rightarrow GF$ be an $\mathcal{EM}$-law. If the evaluation functions satisfy $ev_G \circ Gev_F \circ \lambda_{[0,\top]} \le ev_F \circ Fev_G$ and compositionality holds for FG, then λ is nonexpansive and hence $\overline{\lambda}\colon \overline{F}\,\overline{G} \Rightarrow \overline{G}\,\overline{F}$ is also an $\mathcal{EM}$-law.*

We will now consider $\mathcal{EM}$-laws for nondeterministic and probabilistic automata. In the first case, T is the powerset monad $\mathcal{P}_{\mathit{fin}}$ and F is the machine functor $M_2 = 2 \times \_^A$, while in the second case T is the distribution monad $\mathcal{D}$ and F is the machine functor $M_{[0,1]} = [0,1] \times \_^A$. Note however that while in the first case Theorem 5.2 is directly applicable, this is not true in the second case, since we need to deal with multifunctors.

▶ **Example 5.3** ($\mathcal{EM}$-law for Nondeterministic Automata). Let $(\mathcal{P}_{\mathit{fin}}, \eta, \mu)$ be the finite powerset monad from Theorem 4.3. The $\mathcal{EM}$-law $\lambda\colon \mathcal{P}_{\mathit{fin}}(2 \times \_^A) \Rightarrow 2 \times \mathcal{P}_{\mathit{fin}}(\_)^A$ is defined, for any set X, as

$$\lambda_X(S) = \big(o, \lambda a \in A.\, \{s'(a) \mid (o', s') \in S\}\big), \quad \text{where} \quad o = \begin{cases} 1 & \exists s' \in X^A.(1, s') \in S \\ 0 & \text{else} \end{cases} .$$

This is exactly the one exploited for the standard powerset construction from automata theory [20]. Indeed, for a nondeterministic automaton $c\colon X \to 2 \times \mathcal{P}_{\mathit{fin}}(X)^A$, the map $[\![-]\!] \circ \eta_X$ assigns to each state its accepted language. Theorem 5.2 ensures that it is nonexpansive (see the extended version [`arXiv:1505.08105`] for a detailed proof).

▶ **Example 5.4** ($\mathcal{EM}$-law for Probabilistic Automata). Let $(\mathcal{D}, \eta, \mu)$ be the distribution monad from Theorem 4.4 and M be the machine bifunctor from Theorem 3.5. There is a known [20] $\mathcal{EM}$-law $\lambda\colon \mathcal{D}([0,1] \times \_^A) \Rightarrow [0,1] \times \mathcal{D}^A$ given by the assignment

$$\lambda_X(P) = \left(\sum_{r \in [0,1]} r \cdot P(r, X^A), \lambda a \in A. \lambda x \in X. \sum_{s \in X^A,\, s(a) = x} P([0,1], s) \right)$$

Also this $\mathcal{EM}$-law is nonexpansive, as shown in the extended version [`arXiv:1505.08105`].

Any FT-coalgebra $c\colon X \to FTX$ can always be regarded as an $\overline{F}\,\overline{T}$-coalgebra by equipping X with the discrete metric assigning $\top$ to non equal states (in this way, c is trivially nonexpansive). The consequence of the nonexpansiveness of the $\mathcal{EM}$-law λ is the following: the "generalized determinization" procedure for nondeterministic and probabilistic automata can now be lifted to pass from $\overline{F}\,\overline{T}$-coalgebras to $\widehat{\overline{F}}$-coalgebras in $\mathcal{EM}(\overline{T})$ by using the upper adjunction in the diagram below (analogously to [13, 14]).

$$\begin{tikzcd}
\mathbf{PMet} \arrow[loop left, "\overline{F}"] \arrow[r, bend left, "L^{\overline{T}}"] \arrow[d, "U"'] & \mathcal{EM}(\overline{T}) \arrow[l, bend left, "U^{\overline{T}}"] \arrow[loop right, "\widehat{\overline{F}}"] \arrow[d, "V"] \\
\mathbf{Set} \arrow[loop left, "F"] \arrow[r, bend left, "L^{T}"] & \mathcal{EM}(T) \arrow[l, bend left, "U^{T}"] \arrow[loop right, "\widehat{F}"]
\end{tikzcd}$$

Since we can also lift the final $\overline{F}$-coalgebra to $\mathcal{EM}(\overline{T})$, we can use it to define trace distance. This procedure is detailed in the next section.

5.2 Final Coalgebra for the Lifted Machine Functor

If we fix the first component of the machine bifunctor M on **Set** we obtain an endofunctor $M_B\colon \mathbf{Set} \to \mathbf{Set}$, $M_B(X) = B \times \_^A$. It is known [16] that the final coalgebra for this functor is $\kappa\colon B^{A^*} \to B \times (B^{A^*})^A$ with $\kappa(t) = (t(\varepsilon), \lambda a \in A.\lambda w \in A^*.t(aw))$. We employ an analogous construction with our lifted machine bifunctor $\overline{M}$ on **PMet**, i.e. we fix a pseudometric space (B, d_B) of outputs and consider coalgebras of the functor $\overline{M}_{(B,d_B)} := \overline{M}((B, d_B), \_)$. To obtain the final coalgebra for this functor in **PMet**, we use the following result from [2].

▶ **Proposition 5.5** ([2, Thm. 6.1]). *Let $\overline{F}\colon \mathbf{PMet} \to \mathbf{PMet}$ be a lifting of a functor $F\colon \mathbf{Set} \to \mathbf{Set}$ which has a final coalgebra $\kappa\colon \Omega \to F\Omega$. For every ordinal i we construct a pseudometric $d_i\colon \Omega \times \Omega \to [0, \top]$ as follows: $d_0 := 0$ is the zero pseudometric, $d_{i+1} := d_i^F \circ (\kappa \times \kappa)$ for all ordinals i and $d_j = \sup_{i<j} d_i$ for all limit ordinals j. This sequence converges for some ordinal θ, i.e., $d_\theta = d_\theta^F \circ (\kappa \times \kappa)$. Moreover $\kappa\colon (\Omega, d_\theta) \xrightarrow{1} (F\Omega, d_\theta^F)$ is the final $\overline{F}$-coalgebra.*

It is hence enough to do fixed-point iteration for the functor F on the determinized state set TX in order to obtain trace distance. The lifted monad is ignored at this stage, but its lifting is of course necessary to establish the Eilenberg-Moore category and its adjunction.

We now consider our two examples, where in both cases F is the machine functor M_B (for two different choices of B):

▶ **Example 5.6** (Final Coalgebra Pseudometric). Let M be the machine bifunctor.

1. We start with nondeterministic automata where the output set is $B = 2$ and we use the discrete metric d_2 as distance on 2 as in Theorem 3.6. As maximal distance we take $\top = 1$ and as evaluation function we use $ev_M(o, s) = c \cdot \max_{a\in A} s(a)$ for $0 < c < 1$.
   For any pseudometric d on 2^{A^*} – the carrier of the final M_2-coalgebra – we know that for elements $(o_1, s_1), (o_2, s_2) \in 2 \times (2^{A^*})^A$ we have the Wasserstein pseudometric $d^{\downarrow F}\big((o_1, s_1), (o_2, s_2)\big) = \max\big\{d_2(o_1, o_2), c \cdot \max_{a\in A} d\big(s_1(a), s_2(a)\big)\big\}$. Thus the fixed-point equation from Theorem 5.5 is, for $L_1, L_2 \in 2^{A^*}$,

   $$d(L_1, L_2) = \max\left\{d_2\big(L_1(\varepsilon), L_2(\varepsilon)\big), c \cdot \max_{a\in A} d\big(\lambda w.L_1(aw), \lambda w.L_2(aw)\big)\right\}$$

   Now because d_2 is the discrete metric with $d_2(0, 1) = 1$ we see that $d_{2^{A^*}}$ as defined below is indeed the least fixed-point of this equation and thus $(2^{A^*}, d_{2^{A^*}})$ is the carrier of the final $\overline{M_2}$-coalgebra.

   $$d_{2^{A^*}}\colon 2^{A^*} \times 2^{A^*} \to [0, 1], \quad d_{2^{A^*}}(L_1, L_2) = c^{\inf\{n\in\mathbb{N} \mid \exists w\in A^n.L_1(w)\neq L_2(w)\}}\,.$$

   A determinized coalgebra has as carrier set sets of states $\mathcal{P}(X)$. Each of these sets is mapped to the language that it accepts and the distance between two languages $L_1, L_2\colon A^* \to 2$ can be determined by looking for a word w of minimal length which is contained in one and not in the other. Then, the distance is computed as $c^{|w|}$. This corresponds to the standard ultrametric on words.
2. Next we consider probabilistic automata where $B = [0, 1]$ equipped with the standard Euclidean metric d_e.
   Furthermore the remaining parameters are set as follows: let $\top = 1$ and the evaluation function is $ev_M(o, s) = c_1 o + c_2|A|^{-1} \sum_{a\in A} s(a)$ for $c_1, c_2 \in (0, 1)$ such that $c_1 + c_2 \leq 1$ as in Theorem 3.5. This time, the machine functor must be lifted as a bifunctor in order to obtain the appropriate distance (cf. the discussion before Theorem 2.10).

For any pseudometric d on $[0,1]^{A^*}$ we know that for $(r_1,s_1),(r_2,s_2) \in [0,1] \times ([0,1]^{A^*})^A$ we have $d^{\downarrow F}((r_1,s_1),(r_2,s_2)) = c_1|r_1 - r_2| + \frac{c_2}{|A|} \cdot \sum_{a \in A} d(s_1(a), s_2(a))$. Thus the fixed-point equation from Theorem 5.5 is, for $p_1, p_2 \in [0,1]^{A^*}$:

$$d(p_1,p_2) = c_1|p_1(\varepsilon) - p_2(\varepsilon)| + \frac{c_2}{|A|} \cdot \sum_{a \in A} d\Big(\lambda w.p_1(aw), \lambda w.p_2(aw)\Big)$$

It is again easy to see that $d_{[0,1]^{A^*}} \colon [0,1]^{A^*} \times [0,1]^{A^*} \to [0,1]$ as presented below is the least fixed-point of this equation and therefore $([0,1]^{A^*}, d_{[0,1]^{A^*}})$ the carrier of the final $\overline{M}_{([0,1],d_e)}$-coalgebra.

$$d_{[0,1]^{A^*}}(p_1,p_2) = c_1 \cdot \sum_{w \in A^*} \left(\frac{c_2}{|A|}\right)^{|w|} |p_1(w) - p_2(w)| \ .$$

Here, a determinized coalgebra has as carrier distributions on states $\mathcal{D}(X)$. Each such distribution is mapped to a function $p \colon A^* \to [0,1]$ assigning numerical values to words. Then the distance, which can be thought of as a form of total variation distance with discount, is computed by the above formula.

If instead of working in the interval $[0,1]$ we use $[0,\infty]$ with $\top = \infty$, we can drop the conditions $c_1, c_2 < 1$ and $c_1 + c_2 \leq 1$. In this case we may set $c_2 := |A|$ and $c_1 := 1/2$ and then the above distance is equal to the total variation distance, i.e.,

$$d_{[0,\infty]^{A^*}}(p_1,p_2) = \frac{1}{2} \cdot \sum_{w \in A^*} |p_1(w) - p_2(w)| \ .$$

6 Conclusion, Related and Future Work

In the last years, an impressive amount of papers has studied behavioral distances for both probabilistic and nondeterministic systems (see, e.g., [10, 7, 23, 1, 5, 6, 8]). The necessity of a general understanding of such metrics is not a mere intellectual whim but it is perceived also by researchers exploiting distances for differential privacy and quantitative information flow (see for instance [4]). As far as we know, the first use of coalgebras for this purpose dates back to [23], where the authors consider systems and distance for a fixed endofunctor on **PMet**. In [2], we introduced the Kantorovich and Wasserstein approaches as a general way to define "canonical liftings" to **PMet** and behavioral distances by finality. These are usually branching-time, while many properties of interest for applications (see again [4]) are usually expressed by means of distances on set of traces. In this paper, we have shown that the work developed in [2] can be fruitfully combined with [14] to obtain various trace distances.

Among the several trace distances introduced in literature, it is worth to mention [1, 5, 6, 8]. Similar to the trace distance we obtain in Theorem 5.6 for probabilistic automata is the one introduced in [1] for Semi-Markov chains with residence time. In [5, 6], both branching-time and linear-time distances are introduced for *metric transition systems*, namely Kripke structures where states are associated with elements of a fixed (pseudo-)metric space M, that would correspond to coalgebras of the form $X \to M \times \mathcal{P}(X)$. In [2], we have shown an example capturing branching-time distance for metric transition systems, but for linear distances we require a distributive law of the form $\mathcal{P}(M \times \_) \Rightarrow M \times \mathcal{P}(\_)$, for which we would need at least M carrying an algebra for the monad $\mathcal{P}$. We also plan to investigate trace metrics in a Kleisli setting [11], where it might be easier to incorporate such examples.

There are two other direct consequences of our work that we did not explain in the main text, but that are important properties of the distances that we obtain (and, indeed, are mentioned in [4] amongst the desiderata for "good" metrics). First, the behavioral branching-distance for $\overline{F}\,\overline{T}$ provides an upper bound to the linear-distance $\overline{F}$, analogously to the well-known fact that bisimilarity implies trace equivalence. To see this, it is enough to observe that there is a functor from the category of $\overline{F}\,\overline{T}$-coalgebras to the one of $\overline{F}$-coalgebras mapping $c\colon X \to \overline{F}\,\overline{T}X$ into $c^\sharp\colon \overline{T}X \to \overline{F}\,\overline{T}X$.

Second, since the final map $[\![\,-\,]\!]$ is a morphism in $\mathcal{EM}(\overline{T})$, the behavioral distance for $\overline{F}$ is nonexpansive w.r.t. the operators of the monad $\overline{T}$. Nonexpansiveness with respect to some operators is a desirable property which has been studied, for instance in [7], as a generalization of the notion of being a congruence for behavioral equivalence. Several researchers are now studying syntactic rule formats ensuring this and other sorts of compositionality (see e.g. [9] and the references therein) and we believe that our Theorem 5.2 may provide some helpful insights.

In this perspective, however, our results are still unsatisfactory if compared to what happens in the case of behavioral equivalences. From a fibrational point of view, one has a *canonical lifting* to **Rel** (the category of relations and relation preserving morphisms) such that compositionality holds on the nose and distributive laws always lift [12, Exercise 4.4.6]. The forgetful functor $U\colon \mathbf{PMet} \to \mathbf{Set}$ is also a fibration [2], but Kantorovich and Wasserstein liftings are not always so well-behaved. Fibrations might be useful also to guarantee soundness of up-to techniques [3] for behavioral distances that, hopefully, will lead to more efficient proofs and algorithms.

Another interesting future work would be to show that Kantorovich and Wasserstein liftings arise from some universal properties, i.e., that they are the smallest and largest metric in some continuum of metrics with certain properties. Here we would like to draw inspiration from [22] which characterizes the Giry monad via a universal property on monad morphisms.

Finally, we would like to have an abstract understanding of the Kantorovich-Rubinstein duality. Preliminary attempts suggest that this is very difficult: indeed the proof for the probabilistic case relies on specific properties of distributions.

References

1 Giorgio Bacci, Giovanni Bacci, Kim G. Larsen, and Radu Mardare. On the total variation distance of semi-markov chains. In *Foundations of Software Science and Computation Structures (FOSSACS 2015)*, volume 9034 of *Lecture Notes in Computer Science*. Springer, 2015.

2 Paolo Baldan, Filippo Bonchi, Henning Kerstan, and Barbara König. Behavioral Metrics via Functor Lifting. In Venkatesh Raman and S. P. Suresh, editors, *34th International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2014)*, volume 29 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 403–415. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2014.

3 Filippo Bonchi, Daniela Petrisan, Damien Pous, and Jurriaan Rot. Coinduction up to in a fibrational setting. In *Proc. of CSL-LICS'14*, 2014.

4 Konstantinos Chatzikokolakis, Daniel Gebler, Catuscia Palamidessi, and Lili Xu. Generalized bisimulation metrics. In *CONCUR 2014 – Concurrency Theory*, Lecture Notes in Computer Science, pages 32–46, 2014.

5 Luca de Alfaro, Marco Faella, and Mariëlle Stoelinga. Linear and branching metrics for quantitative transition systems. In *Proc. of ICALP'04*, volume 3142 of *Lecture Notes in Computer Science*, pages 97–109. Springer, 2004.

6 Luca de Alfaro, Marco Faella, and Mariëlle Stoelinga. Linear and branching system metrics. *IEEE Transactions on Software Engineering*, 25(2), 2009.

7 Josée Desharnais, Vineet Gupta, Radha Jagadeesan, and Prakash Panangaden. Metrics for labelled Markov processes. *Theoretical Computer Science*, 318(3):323–354, 2004.

8 Uli Fahrenberg, Axel Legay, and Claus Thrane. The quantitative linear-time–branching-time spectrum. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2011)*, volume 13 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 103–114. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2011.

9 Daniel Gebler and Simone Tini. Compositionality of approximate bisimulation for probabilistic systems. In *Proc. of EXPRESS/SOS'13*, pages 32–46, 2013.

10 Alessandro Giacalone, Chi-Chang Jou, and Scott A. Smolka. Algebraic reasoning for probabilistic concurrent systems. In *Proc. IFIP TC2 Working Conference on Programming Concepts and Methods*, pages 443–458. North-Holland, 1990.

11 Ichiro Hasuo, Bart Jacobs, and Ana Sokolova. Generic trace semantics via coinduction. *Logical Methods in Computer Science*, 3 (4:11):1–36, November 2007.

12 Bart Jacobs. Introduction to coalgebra. Towards mathematics of states and observations, 2012. Draft.

13 Bart Jacobs, Alexandra Silva, and Ana Sokolova. Trace semantics via determinization. In Dirk Pattinson and Lutz Schröder, editors, *Coalgebraic Methods in Computer Science*, volume 7399 of *Lecture Notes in Computer Science*, pages 109–129. Springer Berlin Heidelberg, 2012.

14 Bart Jacobs, Alexandra Silva, and Ana Sokolova. Trace semantics via determinization. *Journal of Computer and System Sciences*, 81(5):859–879, 2015. 11th International Workshop on Coalgebraic Methods in Computer Science, CMCS 2012 (Selected Papers).

15 Bartek Klin. Bialgebras for structural operational semantics: An introduction. *Theoretical Computer Science*, 412(38):5043–5069, 2011.

16 E.G. Manes and M.A. Arbib. Algebraic approaches to program semantics. *Texts and Monographs in Computer Science*, 1986.

17 John Power and Daniele Turi. A coalgebraic foundation for linear time semantics. *Electronic Notes in Theoretical Computer Science*, 29:259–274, 1999.

18 J.J.M.M. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249:3–80, 2000.

19 Lutz Schröder. Expressivity of coalgebraic modal logic: The limits and beyond. *Theoretical Computer Science*, 390(2–3):230–247, 2008.

20 Alexandra Silva, Filippo Bonchi, Marcello M. Bonsangue, and Jan J. M. M. Rutten. Generalizing determinization from automata to coalgebras. *Logical Methods in Computer Science*, 9(1), 2013.

21 Daniele Turi and Gordon D. Plotkin. Towards a mathematical operational semantics. In *Proc. of LICS'97*, pages 280–291, 1997.

22 Franck van Breugel. The metric monad for probabilistic nondeterminism, April 2005.

23 Franck van Breugel and James Worrell. A behavioural pseudometric for probabilistic transition systems. *Theoretical Computer Science*, 331:115–142, 2005.

24 Franck van Breugel and James Worrell. Approximating and computing behavioural distances in probabilistic transition systems. *Theoretical Computer Science*, 360(1):373–385, 2006.

25 Cédric Villani. *Optimal Transport – Old and New*, volume 338 of *A Series of Comprehensive Studies in Mathematics*. Springer, 2009.

A Fibrational Approach to Automata Theory

Liang-Ting Chen and Henning Urbat

Institut für Theoretische Informatik
Technische Universität Braunschweig, Germany
`{l.chen,h.urbat}@iti.cs.tu-bs.de`

Abstract

For predual categories $\mathcal{C}$ and $\mathcal{D}$ we establish isomorphisms between opfibrations representing local varieties of languages in $\mathcal{C}$, local pseudovarieties of $\mathcal{D}$-monoids, and finitely generated profinite $\mathcal{D}$-monoids. The global sections of these opfibrations are shown to correspond to varieties of languages in $\mathcal{C}$, pseudovarieties of $\mathcal{D}$-monoids, and profinite equational theories of $\mathcal{D}$-monoids, respectively. As an application, a new proof of Eilenberg's variety theorem along with several related results is obtained, covering uniformly varieties of languages and their coalgebraic modifications, Straubing's C-varieties, and fully invariant local varieties.

1998 ACM Subject Classification F.4.3 Formal Languages

Keywords and phrases Eilenberg's variety theorem, duality, coalgebra, Grothendieck fibration

Digital Object Identifier 10.4230/LIPIcs.CALCO.2015.50

1 Introduction

In algebraic automata theory, regular languages are studied in connection with associated algebraic structures, using Eilenberg's celebrated variety theorem [7]. This theorem establishes a one-to-one correspondence between varieties of languages and pseudovarieties of monoids. A *variety of languages* associates to each finite alphabet Σ a set V_Σ of regular languages over Σ which is closed under (a) the boolean operations (union, intersection and complement), (b) left derivatives $a^{-1}L = \{\, w \in \Sigma^* \mid aw \in L \,\}$ and right derivatives $La^{-1} = \{\, w \in \Sigma^* \mid wa \in L \,\}$, for $a \in \Sigma$, and (c) preimages of free monoid morphisms, i.e. for every free monoid homomorphism $f\colon \Sigma^* \to \Delta^*$ and every $L \in V_\Delta$, the preimage $f^{-1}[L]$ lies in V_Σ.

Not every interesting class of languages falls within this scope, though. For this reason several authors weakened the closure properties in the definition of a variety of languages, and proved Eilenberg-type theorems for these modified varieties. For example, Pin's *positive varieties* [14], omitting closure under complement, correspond to pseudovarieties of *ordered* monoids. Polák's *disjunctive varieties* [16], further dropping closure under intersection, correspond to pseudovarieties of idempotent semirings. Reutenauer's *xor varieties* [18], closed under symmetric difference in lieu of the boolean operations, correspond to pseudovarieties of associative algebras over the field $\mathbb{Z}_2$. Straubing [20] introduced C-*varieties of languages*, where one restricts to closure under preimages of a chosen class C of free monoid morphisms in lieu of all free monoid morphisms. They are in bijection with C-*pseudovarieties of monoid morphisms*, these being classes of monoid morphisms with suitable closure properties.

The above notions of a variety of languages treat the alphabet as a variable. A closely related line of work concerns "local" versions of Eilenberg's variety theorem where a *fixed* alphabet Σ is considered. Using the well-known duality between boolean algebras and Stone spaces, Pippenger [15] demonstrated that the boolean algebra $\mathsf{Reg}(\Sigma)$ of all regular languages over Σ dualises to the underlying Stone space of the free profinite monoid on Σ. Later, Gehrke, Grigorieff, and Pin [8] considered *local varieties of languages over* Σ, i.e. boolean

6th International Conference on Algebra and Coalgebra in Computer Science (CALCO'15).
Editors: Lawrence S. Moss and Paweł Sobociński; pp. 50–65
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

subalgebras of $\mathsf{Reg}(\Sigma)$ closed under left and right derivatives, and characterised them as sets of regular languages over Σ definable by profinite equations.

In the recent work of Adámek, Milius, Myers, and Urbat [1, 2] a categorical approach to Eilenberg-type theorems was presented, covering many of the aforementioned results uniformly. The leading idea is to take two varieties of (possibly ordered) algebras $\mathcal{C}$ and $\mathcal{D}$ whose full subcategories on finite algebras are dually equivalent. Local varieties of languages are then modelled as coalgebras in $\mathcal{C}$, and monoids as monoid objects in $\mathcal{D}$. The main result of [1], the **General Local Variety Theorem**, states that *local varieties of languages over* Σ *in* $\mathcal{C}$ (= sets of regular languages over Σ closed under $\mathcal{C}$-algebraic operations and left and right derivatives) correspond to *local pseudovarieties of* Σ*-generated* $\mathcal{D}$*-monoids* (= sets of Σ-generated finite $\mathcal{D}$-monoids closed under quotients and subdirect products). The **General Variety Theorem** of [2] establishes a correspondence between (non-local) *varieties of languages in* $\mathcal{C}$ and *pseudovarieties of* $\mathcal{D}$*-monoids.* Then the classical Eilenberg theorem is recovered by taking $\mathcal{C}$ = boolean algebras and $\mathcal{D}$ = sets, and other choices of $\mathcal{C}$ and $\mathcal{D}$ give its modifications due to Pin, Polák and Reutenauer along with new concrete Eilenberg-type correspondences.

The present paper investigates from a categorical perspective various important concepts of algebraic automata theory whose precise connection was left open in [1, 2], notably

(a) the connection between local pseudovarieties of $\mathcal{D}$-monoids and profinite $\mathcal{D}$-monoids;

(b) the connection between the local and non-local versions of the General Variety Theorem.

Our strategy is to organise all local varieties of languages into a category **LAN** whose objects are pairs (Σ, V) of a finite alphabet Σ and a local variety of languages over Σ in $\mathcal{C}$. With a suitable choice of morphisms in **LAN** (see Definition 3.7) the projection functor $p\colon \mathbf{LAN} \to \mathbf{Free}(\mathbf{Mon}\mathcal{D})$ into the category of finitely generated free $\mathcal{D}$-monoids, mapping (Σ, V) to the free $\mathcal{D}$-monoid over Σ, is an opfibration. Similarly one can form the category **LPV** of local pseudovarieties of $\mathcal{D}$-monoids and the category **PFMon** of finitely generated profinite $\mathcal{D}$-monoids, which again yield opfibrations over $\mathbf{Free}(\mathbf{Mon}\mathcal{D})$ as shown below:

$$\begin{tikzcd}
\mathbf{LAN} \arrow[r, "\cong"] \arrow[dr, "p"'] & \mathbf{LPV} \arrow[r, "\cong"] \arrow[d, "q"] & \mathbf{PFMon} \arrow[dl, "q'"] \\
& \mathbf{Free}(\mathbf{Mon}\mathcal{D}). &
\end{tikzcd}$$

Next we make two crucial observations. Firstly, we observe that the global sections, i.e. right inverse functors, of the above opfibrations p and q' correspond by definition to varieties of languages in $\mathcal{C}$ and profinite equational theories of $\mathcal{D}$-monoids, respectively. Secondly, we prove that the three opfibrations are isomorphic. The isomorphism $\mathbf{LAN} \cong \mathbf{LPV}$ is essentially the General Local Variety Theorem of [1], and the isomorphism $\mathbf{LPV} \cong \mathbf{PFMon}$ is based on a limit construction. It follows that the global sections of p and q' are in bijection, from which we derive our main result:

> *There is a bijective correspondence between* (i) *varieties of languages in* $\mathcal{C}$*,* (ii) *pseudovarieties of* $\mathcal{D}$*-monoids, and* (iii) *profinite equational theories of* $\mathcal{D}$*-monoids.*

The bijection (ii)↔(iii) amounts to a categorical presentation of the well-known Reiterman-Banaschewski theorem [17, 5]. And (i)↔(ii) gives an independent proof of the General Variety Theorem of [2] based on its local version; see also Remark 5.6. Furthermore, the flexibility of our fibrational setting leads to a number of additional new results without extra effort. For example, by replacing the category $\mathbf{Free}(\mathbf{Mon}\mathcal{D})$ with an arbitrary subcategory

$\mathsf{C} \hookrightarrow \mathbf{Free}(\mathbf{Mon}\mathcal{D})$ we obtain a generalised version of Straubing's variety theorem for C-varieties of languages, as well as a new local variety theorem for *fully invariant local varieties of languages*, i.e. local varieties closed under preimages of endomorphisms of free monoids.

Beyond these concrete results, we believe that the main contribution of the present paper is a further illumination of the intrinsic duality deeply hidden in algebraic language theory, most notably of the subtle interweavings of local and non-local structures, and the role of profinite theories.

2 Preliminaries

In this section we review the categorical approach to algebraic automata theory developed in [1, 2]. The idea is to interpret local varieties of languages inside a variety of algebras $\mathcal{C}$, and to relate them to finite monoids in another variety of (possibly ordered) algebras $\mathcal{D}$ which is **predual** to $\mathcal{C}$. The latter means that the full subcategories $\mathcal{C}_f$ and $\mathcal{D}_f$ on finite algebras are dually equivalent. Note that by an **ordered algebra** we mean an algebra (over a finitary signature Γ) with a poset structure on its underlying set making all operations monotone. Morphisms of ordered algebras are order-preserving Γ-homomorphisms. A *variety of ordered algebras* is a class of ordered algebras specified by inequalities $t_1 \leq t_2$ between Γ-terms.

▶ **Assumptions 2.1.** In the following $\mathcal{C}$ and $\mathcal{D}$ are predual varieties of algebras, where $\mathcal{D}$-algebras may be ordered, subject to the following conditions:

1. $\mathcal{C}$ and $\mathcal{D}$ are **locally finite**, i.e. every free algebra on a finite set is finite;
2. epimorphisms in $\mathcal{D}$ are surjective;
3. $\mathcal{D}$ is **entropic**[1], i.e. given an m-ary operation σ and an n-ary operation τ in the signature of $\mathcal{D}$ and variables x_{ij} $(i = 1, \dots, m,\ j = 1, \dots, n)$, the following equation holds in $\mathcal{D}$:

$$\sigma(\tau(x_{11}, \dots, x_{1n}), \dots, \tau(x_{m1}, \dots, x_{mn})) = \tau(\sigma(x_{11}, \dots, x_{m1}), \dots, \sigma(x_{1n}, \dots, x_{mn})).$$

▶ **Notation 2.2.** We write $\Phi\colon \mathbf{Set} \to \mathcal{C}$ and $\Psi\colon \mathbf{Set} \to \mathcal{D}$ for the left adjoints to the forgetful functors $|{-}|\colon \mathcal{C} \to \mathbf{Set}$ and $|{-}|\colon \mathcal{D} \to \mathbf{Set}$, respectively. By $\mathbf{1}_{\mathcal{C}} = \Phi\mathbf{1}$ and $\mathbf{1}_{\mathcal{D}} = \Psi\mathbf{1}$ denote the free algebras over the singleton set.

▶ **Example 2.3.** The following pairs of varieties $\mathcal{C}/\mathcal{D}$ satisfy our assumptions. The details of the first three examples can be found in [11].

1. $\mathbf{BA}/\mathbf{Set}$: The Stone Representation Theorem exhibits a dual equivalence between the categories of finite boolean algebras and finite sets. It assigns to any finite boolean algebra B the set $\mathbf{BA}(B, \mathbf{2})$ of all homomorphisms into the two-chain $\mathbf{2}$. The dual of $h\colon A \to B$ is given by precomposition with h, i.e. $f \in \mathbf{BA}(B, \mathbf{2})$ is mapped to $f \circ h \in \mathbf{BA}(A, \mathbf{2})$.
2. $\mathbf{DLat}/\mathbf{Pos}$: Similarly, the Birkhoff Representation Theorem exhibits a dual equivalence between the categories of finite distributive lattices with 0 and 1 and finite posets. It assigns to a finite distributive lattice L the poset $\mathbf{DLat}(L, \mathbf{2})$, ordered pointwise, where $\mathbf{2}$ is the two-chain. On morphisms the dual equivalence again acts by precomposition.
3. $\mathbf{SLat}/\mathbf{SLat}$: The category of finite semilattices with 0 is self-dual: the dual equivalence maps a finite semilattice S to the semilattice $\mathbf{SLat}_f(S, \mathbf{2})$ whose join is taken pointwise.

[1] In the unordered case, entropic varieties are precisely the categories of Eilenberg-Moore algebras for a commutative finitary monad on **Set**, see [12]

4. $\mathbb{Z}_2$-**Vec**/$\mathbb{Z}_2$-**Vec**: The category of finite-dimensional vector spaces over any field F is self-dual, by mapping a vector space V to its dual space $F\text{-}\mathbf{Vec}(V, F)$. By restricting F to the binary field $\mathbb{Z}_2$, the category is also locally finite.

▶ **Remark 2.4.** Given a small finitely complete and cocomplete category $\mathcal{A}$ we denote by $\mathcal{Y}\colon \mathcal{A} \to \mathsf{Ind}\mathcal{A}$ and $\mathcal{Y}^{op}\colon \mathcal{A} \to \mathsf{Pro}\mathcal{A}$ the ind- and pro-completion of $\mathcal{A}$, i.e. the free completion under filtered colimits and cofiltered limits, respectively. There is an adjunction $F \dashv U\colon \mathsf{Pro}\mathcal{A} \to \mathsf{Ind}\mathcal{A}$ such that $\mathcal{Y}^{op} = F \circ \mathcal{Y}$ and $\mathcal{Y} = U \circ \mathcal{Y}^{op}$.

$$\begin{tikzcd}
& \mathcal{A} \arrow[dl, "\mathcal{Y}"'] \arrow[dr, "\mathcal{Y}^{op}"] & \\
\mathsf{Ind}\mathcal{A} \arrow[rr, bend left, "F"] \arrow[rr, phantom, "\perp"] & & \mathsf{Pro}\mathcal{A} \arrow[ll, bend left, "U"]
\end{tikzcd}$$

Applying this to $\mathcal{A} = \mathcal{C}_f$ with $\mathsf{Ind}(\mathcal{C}_f) = \mathcal{C}$ and $\mathsf{Pro}(\mathcal{C}_f) = \mathsf{Ind}(\mathcal{C}_f^{op})^{op} \cong \mathsf{Ind}(\mathcal{D}_f)^{op} = \mathcal{D}^{op}$, we see that the equivalence $\mathcal{C}_f \cong \mathcal{D}_f^{op}$ extends to an adjunction between $\mathcal{C}$ and $\mathcal{D}^{op}$. We denote both the equivalence $\mathcal{C}_f \cong \mathcal{D}_f^{op}$ and the induced adjunction between $\mathcal{C}$ and $\mathcal{D}^{op}$ by

$$S \dashv P\colon \mathcal{D}_f^{op} \xrightarrow{\cong} \mathcal{C}_f \quad \text{and} \quad S \dashv P\colon \mathcal{D}^{op} \to \mathcal{C}.$$

2.1 Local varieties of languages in $\mathcal{C}$

The coalgebraic treatment of automata roots in the observation that a deterministic automaton without an initial state is a coalgebra $\gamma = \langle \gamma^{1\text{st}}, \gamma^{2\text{nd}} \rangle\colon Q \to \mathbf{2} \times Q^{\Sigma}$ for the set functor $T_{\Sigma}^{0} = \mathbf{2} \times (-)^{\Sigma}$. Here Σ is the finite input alphabet, $\mathbf{2} := \{\mathtt{yes}, \mathtt{no}\}$, $\gamma^{1\text{st}}\colon Q \to \mathbf{2}$ is the characteristic function of the final states, and $\gamma^{2\text{nd}}\colon Q \to Q^{\Sigma}$ is the transition map. In the following we consider automata in the category $\mathcal{C}$, which requires to replace the set $\mathbf{2}$ by a suitable "output" object in $\mathcal{C}$. Observe that the dual adjunction $S \dashv P\colon \mathcal{D}^{op} \to \mathcal{C}$ has *dualising objects* $O_{\mathcal{C}} := P\mathbf{1}_{\mathcal{D}}$ and $O_{\mathcal{D}} := S\mathbf{1}_{\mathcal{C}}$, that is, for all $M \in \mathcal{D}$ and $Q \in \mathcal{C}$ we have

$$|PM| \cong \mathcal{C}(\mathbf{1}_{\mathcal{C}}, PM) \cong \mathcal{D}(M, O_{\mathcal{D}}) \quad \text{and} \quad |SQ| \cong \mathcal{D}(\mathbf{1}_{\mathcal{D}}, SQ) \cong \mathcal{C}(Q, O_{\mathcal{C}}).$$

Taking $M = \mathbf{1}_{\mathcal{D}}$ we see that the set $|O_{\mathcal{C}}|$ is isomorphic to $|O_{\mathcal{D}}|$. Note that in each of the categories $\mathcal{C}/\mathcal{D}$ in Example 2.3 the objects $O_{\mathcal{C}}$ and $O_{\mathcal{D}}$ have a two-element carrier. Motivated by this observation, we replace the set $\mathbf{2}$ by the object $O_{\mathcal{C}}$ to define automata in $\mathcal{C}$.

▶ **Definition 2.5.** A **Σ-automaton in $\mathcal{C}$** is a coalgebra $\gamma = \langle \gamma^{1\text{st}}, \gamma^{2\text{nd}} \rangle\colon Q \to O_{\mathcal{C}} \times Q^{\Sigma}$ for the endofunctor $T_{\Sigma} := O_{\mathcal{C}} \times (-)^{\Sigma}$ on $\mathcal{C}$, where $(-)^{\Sigma}$ is the Σ-fold product. A **subautomaton** of (Q, γ) is a subcoalgebra of (Q, γ), represented by an injective coalgebra homomorphism into Q. An automaton is called **finite** if the object Q of states is finite, and **locally finite** if it is a filtered colimit of finite Σ-automata. The **rational fixpoint** ρT_{Σ} is the filtered colimit of all finite Σ-automata. The categories of Σ-automata, finite Σ-automata and locally finite Σ-automata in $\mathcal{C}$ are denoted by $\mathbf{Aut}\Sigma$, $\mathbf{Aut}_f\Sigma$ and $\mathbf{Aut}_{lf}\Sigma$, respectively. Their morphisms are coalgebra homomorphisms.

In [13, 3] it is shown that the rational fixpoint ρT_{Σ} is the terminal locally finite coalgebra (i.e. the terminal object of $\mathbf{Aut}_{lf}\Sigma$), with the structure map $\rho T_{\Sigma} \xrightarrow{\zeta} T_{\Sigma}(\rho T_{\Sigma})$ an isomorphism. The rational fixpoint of the set functor $T_{\Sigma}^{0} = \mathbf{2} \times (-)^{\Sigma}$ is the automaton of regular languages: the states of ρT_{Σ}^{0} form the set $\mathsf{Reg}(\Sigma)$ of regular languages over Σ, the final states are those languages containing the empty word ε, and the transitions are given by left derivatives, that is, $L \xrightarrow{a} a^{-1}L = \{\, w \in \Sigma^* \mid aw \in L \,\}$ for $L \in \mathsf{Reg}(\Sigma)$ and $a \in \Sigma$.

▶ **Remark 2.6.** To simplify the presentation, we assume in the following that $|O_{\mathcal{C}}| = |O_{\mathcal{D}}| = \mathbf{2}$. The main reason is that in this case the rational fixpoint ρT_Σ is a lifting of the above automaton of regular languages to $\mathcal{C}$, see the next proposition. Without this assumption one needs to replace regular languages by regular behaviors, i.e. functions $\Sigma^* \to |O_{\mathcal{C}}|$ realised by finite Moore automata with output set $|O_{\mathcal{C}}|$. See also the discussion in [2, Section V].

▶ **Proposition 2.7** (see [1]). *The rational fixpoint ρT_Σ is carried by the set* $\mathsf{Reg}(\Sigma)$*. Its coalgebra structure $\rho T_\Sigma \xrightarrow{\zeta} O_{\mathcal{C}} \times (\rho T_\Sigma)^\Sigma$ is given by the $\mathcal{C}$-morphisms*

$$\zeta^{1\text{st}}(L) = \begin{cases} \texttt{yes} & \textit{if } \varepsilon \in L; \\ \texttt{no} & \textit{otherwise,} \end{cases} \qquad \textit{and} \quad \zeta^{2\text{nd}}(L)(a) = a^{-1}L.$$

In the light of this proposition we also write $\mathsf{Reg}(\Sigma)$ for the rational fixpoint ρT_Σ.

▶ **Example 2.8.** For $\mathcal{C} = \mathbf{BA}$, the rational fixpoint of T_Σ is the boolean algebra $\mathsf{Reg}(\Sigma)$ (w.r.t. $\cup$, $\cap$, $(-)^{\complement}$, $\emptyset$ and Σ^*), endowed with the automata structure given by the boolean homomorphisms $\zeta^{1\text{st}}$ and $\zeta^{2\text{nd}}$. Similarly, for the other categories $\mathcal{C}$ of Example 2.3 the algebraic structure of $\rho T_\Sigma = \mathsf{Reg}(\Sigma)$ is (i) $\cup$, $\cap$, $\emptyset$, and Σ^* for $\mathcal{C} = \mathbf{DLat}$; (ii) $\cup$ and $\emptyset$ for $\mathcal{C} = \mathbf{SLat}$; (iii) symmetric difference $L \oplus L' = (L \setminus L') \cup (L' \setminus L)$ and $\emptyset$ for $\mathcal{C} = \mathbb{Z}_2\text{-}\mathbf{Vec}$.

▶ **Definition 2.9.** A **local variety of languages over $\mathbf{\Sigma}$ in $\mathcal{C}$** is a subautomaton V of ρT_Σ closed under right derivatives, i.e. $L \in |V|$ implies $La^{-1} = \{\, w \in \Sigma^* \mid wa \in L \,\} \in |V|$ for all $a \in \Sigma$. The $\bigcap$-semilattices of all (finite) local varieties of languages over Σ in $\mathcal{C}$ are denoted by $\mathbf{LAN}^f_\Sigma$ and $\mathbf{LAN}_\Sigma$, respectively.

Observe that a local variety of languages is closed under (i) the $\mathcal{C}$-algebraic operations of ρT_Σ, being a sub*algebra* of ρT_Σ in $\mathcal{C}$, and (ii) left derivatives, being a sub*coalgebra* of ρT_Σ. For $\mathcal{C} = \mathbf{DLat}$ ($\mathcal{C} = \mathbf{BA}$) a local variety of languages is precisely a *(boolean) quotienting algebra of languages* in the sense of Gehrke et al. [8]: a set of regular languages over Σ closed under finite union, finite intersection (and complement) as well as left and right derivatives.

2.2 $\mathcal{D}$-monoids

Every entropic variety $\mathcal{D}$ of (ordered) algebras can be equipped with a symmetric monoidal closed structure $(\mathcal{D}, \otimes, \mathbf{1}_{\mathcal{D}})$, see e.g. [4] and [6, Theorem 3.10.1]. The unit $\mathbf{1}_{\mathcal{D}}$ is the free one-generated algebra and $\otimes$ is the usual tensor product of algebras, giving rise to a natural bijection between morphisms and bimorphisms in $\mathcal{D}$:

$$\mathsf{Hom}(A \otimes B, C) \cong \mathsf{Bihom}(A \times B, C).$$

Recall that a **bimorphism** $f\colon A \times B \to C$ in $\mathcal{D}$ is a set-theoretic function from $A \times B$ to C such that $f(a,-)\colon B \to C$ and $f(-,b)\colon A \to C$ are $\mathcal{D}$-morphisms for any $a \in A$ and $b \in B$.

Since the tensor product represents bimorphisms, the monoid objects of the monoidal category $(\mathcal{D}, \otimes, \mathbf{1}_{\mathcal{D}})$ correspond to the following algebraic concept:

▶ **Definition 2.10.** A **$\mathcal{D}$-monoid** $(M, \bullet, e)$ is an object M of $\mathcal{D}$ equipped with a monoid structure $(|M|, \bullet, e)$ in $\mathbf{Set}$ whose multiplication $\bullet\colon M \times M \to M$ is a $\mathcal{D}$-bimorphism. By a **morphism** $f\colon (M, \bullet, e) \to (M', \bullet', e')$ of $\mathcal{D}$-monoids is meant a morphism $f\colon M \to M'$ of $\mathcal{D}$ that is also a monoid morphism between the underlying monoids in $\mathbf{Set}$. By $\mathbf{Mon}_f\mathcal{D}$ and $\mathbf{Mon}\mathcal{D}$ we denote the categories of (finite) $\mathcal{D}$-monoids and all $\mathcal{D}$-monoid morphisms.

▶ **Example 2.11.** For the categories $\mathcal{D} = \mathbf{Set}$, $\mathbf{Pos}$, $\mathbf{SLat}$ and $\mathbb{Z}_2$-$\mathbf{Vec}$ of Example 2.3, the $\mathcal{D}$-monoids are precisely ordinary monoids, ordered monoids, idempotent semirings (with 0 and 1) and associative algebras over the field $\mathbb{Z}_2$, respectively.

▶ **Remark 2.12.**

1. In $\mathcal{D}$ we choose the factorisation system (epi, strong mono). Recall that epimorphisms in $\mathcal{D}$ are precisely the surjective morphisms by Assumption 2.1.2. Strong monomorphisms are precisely the injective morphisms if $\mathcal{D}$ is a variety of algebras, and embeddings (i.e. injective order-reflecting morphisms) if $\mathcal{D}$ is a variety of ordered algebras. Hence every $\mathcal{D}$-morphism $f\colon A \to B$ factorises as $A \xrightarrow{\mathrm{Im}(f)} f[A] \xrightarrow{i} B$ where $\mathrm{Im}(f)$ is the restriction of f to the image $f[A]$ and i is injective (and order-reflecting). Further, the factorisation system has the **fill-in property**: given a surjective morphism e, an injective (and order-reflecting) morphism m and two morphisms u, v with $ue = mv$, there is a unique morphism d such that $u = md$ and $v = de$.
2. The factorisation system of $\mathcal{D}$ lifts to $\mathbf{Mon}\mathcal{D}$. Hence **submonoids** are represented by injective (order-reflecting) $\mathcal{D}$-monoid morphisms, and **quotient monoids** by surjective $\mathcal{D}$-monoid morphisms.

Since $\mathbf{Mon}\mathcal{D}$ is a variety of (ordered) algebras, the forgetful functor $\mathbf{Mon}\mathcal{D} \to \mathbf{Set}$ has a left adjoint constructing free $\mathcal{D}$-monoids. Here is a concrete construction:

▶ **Proposition 2.13** (see [1]). *The free $\mathcal{D}$-monoid on a set Σ is carried by the $\mathcal{D}$-object $\Psi\Sigma^*$. The monoid multiplication $\bullet$ extends the concatenation of words in Σ^*, and the unit is ε.*

A **finite Σ-generated $\mathcal{D}$-monoid** is a finite quotient $e_M\colon \Psi\Sigma^* \twoheadrightarrow M$ of the free $\mathcal{D}$-monoid on Σ. Given another finite Σ-generated $\mathcal{D}$-monoid $e_N\colon \Psi\Sigma^* \twoheadrightarrow N$ we write $M \leq N$ if there is a $\mathcal{D}$-monoid morphism $f\colon N \to M$ satisfying $e_M = fe_N$. With respect to this order all (isomorphism classes of) finite Σ-generated $\mathcal{D}$-monoids form a poset $\mathsf{Quo}_f(\Psi\Sigma^*)$. Observe that $\mathsf{Quo}_f(\Psi\Sigma^*)$ is a join-semilattice: the join of M and N is the **subdirect product**, viz. the image of the morphism $\langle e_M, e_N\rangle\colon \Psi\Sigma^* \to M \times N$ given by

$$M \vee N := \{\, (e_M(x), e_N(x)) \in M \times N \mid x \in \Psi\Sigma^* \,\}.$$

▶ **Definition 2.14.** A **local pseudovariety of $\mathcal{D}$-monoids over Σ** is an *ideal* of $\mathsf{Quo}_f(\Psi\Sigma^*)$, i.e. a set of finite Σ-generated $\mathcal{D}$-monoids closed under quotients and subdirect products. By $\mathbf{LPV}_\Sigma$ we denote the $\bigcap$-semilattice of local pseudovarieties of $\mathcal{D}$-monoids over Σ.

▶ **Theorem 2.15** (General Local Variety Theorem [1]). *For each finite alphabet Σ,*

$$\mathbf{LAN}^f_\Sigma \cong \mathsf{Quo}_f(\Psi\Sigma^*) \quad \textit{and} \quad \mathbf{LAN}_\Sigma \cong \mathbf{LPV}_\Sigma.$$

▶ **Remark 2.16.** 1. The first isomorphism takes a finite local variety $O_{\mathcal{C}} \xleftarrow{\gamma^{1\text{st}}} V \xrightarrow{\gamma^{2\text{nd}}} V^\Sigma$ in $\mathcal{C}$ and applies the equivalence functor $S\colon \mathcal{C}_f \xrightarrow{\cong} \mathcal{D}_f^{op}$ to its coalgebra structure. This yields an algebra $\mathbf{1}_{\mathcal{D}} \cong S(O_{\mathcal{C}}) \xrightarrow{S\gamma^{1\text{st}}} SV \xleftarrow{S\gamma^{2\text{nd}}} S(V^\Sigma) \cong \coprod_\Sigma SV$ for the functor $F_\Sigma = 1_{\mathcal{D}} + \coprod_\Sigma(-)$ on $\mathcal{D}$. Since the free $\mathcal{D}$-monoid $\Psi\Sigma^*$ also carries the initial algebra for F_Σ, there is a unique F_Σ-algebra homomorphism $e_{SV}\colon \Psi\Sigma^* \to SV$ into the algebra constructed above. One then shows that e_{SV} is surjective and there is a unique $\mathcal{D}$-monoid structure on SV making e_{SV} a $\mathcal{D}$-monoid morphism. We call $e_{SV}\colon \Psi\Sigma^* \twoheadrightarrow SV$ the **(finite Σ-generated) $\mathcal{D}$-monoid corresponding to V**.

2. The second isomorphism follows from the observation that (a) $\mathbf{LPV}_\Sigma$ is by definition the ideal completion of $\mathsf{Quo}_f(\Psi\Sigma^*)$, and (b) $\mathbf{LAN}_\Sigma$ is isomorphic to the ideal completion of $\mathbf{LAN}^f_\Sigma$. Indeed every finite local variety of languages is a compact element of $\mathbf{LAN}_\Sigma$, and every local variety is the directed union of its finite local subvarieties. Hence the isomorphism $\mathbf{LAN}_\Sigma \cong \mathbf{LPV}_\Sigma$ maps a local variety of languages $V \hookrightarrow \rho T_\Sigma$ to the local pseudovariety of all finite Σ-generated $\mathcal{D}$-monoids that correspond to some finite local subvariety of V. The inverse isomorphism maps a local pseudovariety P of $\mathcal{D}$-monoids over Σ to the directed union of all finite local varieties of languages in $\mathcal{C}$ that correspond to some element of P.

2.3 Preimages under $\mathcal{D}$-monoid morphisms

Recall from Remark 2.6 that we assume $|O_\mathcal{C}| = |O_\mathcal{D}| = \mathbf{2}$. Hence a language $L \subseteq \Delta^*$ may be identified with a morphism $L\colon \Psi\Delta^* \to O_\mathcal{D}$ of $\mathcal{D}$, viz. the adjoint transpose of the characteristic function $\Delta^* \to |O_\mathcal{D}|$. Given this identification, the **preimage** of L under a $\mathcal{D}$-monoid morphism $f\colon \Psi\Sigma^* \to \Psi\Delta^*$ is the composite $Lf\colon \Psi\Sigma^* \to \Psi\Delta^* \to O_\mathcal{D}$. By the adjunction $S \dashv P\colon \mathcal{D}^{op} \to \mathcal{C}$, the morphism Pf is essentially the preimage function, because

$$|Pf| \cong \mathcal{D}(f, O_\mathcal{D})\colon \mathcal{D}(\Psi\Delta^*, O_\mathcal{D}) \to \mathcal{D}(\Psi\Sigma^*, O_\mathcal{D}).$$

In [2] it was shown that $|Pf|$ restricts to a $\mathcal{C}$-morphism $f^{-1}\colon \mathsf{Reg}(\Delta) \to \mathsf{Reg}(\Sigma)$, taking any language $L\colon \Psi\Delta^* \to O_\mathcal{D}$ in $\mathsf{Reg}(\Delta)$ to its f-preimage. This observation makes the following definition evident:

▶ **Definition 2.17.** Let $f\colon \Psi\Sigma^* \to \Psi\Delta^*$ be a $\mathcal{D}$-monoid morphism and V and W local varieties of languages over Σ and Δ, respectively. Then V is said to be **closed under *f*-preimages of languages in *W*** if Diagram 1 below commutes for some $\mathcal{C}$-morphism h.

$$\begin{tikzcd} W \arrow[r, "h"] \arrow[d, tail] & V \arrow[d, tail] \\ \mathsf{Reg}(\Delta) \arrow[r, "f^{-1}"'] & \mathsf{Reg}(\Sigma) \end{tikzcd}$$

Diagram 1

$$\begin{tikzcd} \Psi\Sigma^* \arrow[r, "f"] \arrow[d, two heads, "e_M"'] & \Psi\Delta^* \arrow[d, two heads, "e_N"] \\ M \arrow[r, "g"'] & N \end{tikzcd}$$

Diagram 2

Here is a dual characterisation of preimage closure:

▶ **Lemma 2.18** (see [2]). *In Definition 2.17 let V and W be finite, and let $e_M\colon \Psi\Sigma^* \twoheadrightarrow M$ and $e_N\colon \Psi\Delta^* \twoheadrightarrow N$ be the finite $\mathcal{D}$-monoids corresponding to V and W, respectively. Then Diagram 1 commutes iff Diagram 2 with $g = Sh$ commutes.*

3 Fibrations for Languages and Monoids

We are ready to present our fibrational setting for (local) varieties of languages in $\mathcal{C}$ and (local) pseudovarieties of $\mathcal{D}$-monoids. For general information on fibred categories the reader is referred to [10]. Let us briefly recall some basic vocabulary:

▶ **Definition 3.1.** Let $p\colon \mathcal{E} \to \mathcal{B}$ be a functor.

1. An object $X \in \mathcal{E}$ is **above** $I \in \mathcal{B}$ if $pX = I$, and similarly a morphism f in $\mathcal{E}$ is above a morphism u in $\mathcal{B}$ if $pf = u$. A morphism f above id_I is called **vertical** (over I).

2. The **fibre** over $I \in \mathcal{B}$ is the subcategory $\mathcal{E}_I = p^{-1}(I)$ of $\mathcal{E}$ whose objects are the objects above I and whose morphisms are the vertical morphisms over I.
3. A morphism $f\colon X \to Y$ of $\mathcal{E}$ is **opcartesian over** $u\colon I \to J$ in $\mathcal{B}$ if $pf = u$ and for every morphism $g\colon X \to Z$ in $\mathcal{E}$ above wu for $w\colon J \to pZ$, there is a unique morphism $h\colon Y \to Z$ above w with $g = hf$.
4. $p\colon \mathcal{E} \to \mathcal{B}$ is an **opfibration** over $\mathcal{B}$ if for every $X \in \mathcal{E}$ and $u\colon pX \to J$ in $\mathcal{B}$ there is an opcartesian morphism $f\colon X \to Y$ above u, called an **opcartesian lifting** of u.
5. Two opfibrations $p\colon \mathcal{E} \to \mathcal{B}$ and $p'\colon \mathcal{E}' \to \mathcal{B}$ are **isomorphic** if there is an isomorphism $i\colon \mathcal{E} \cong \mathcal{E}'$ preserving indices, that is, $p'i = p$.
6. A **global section** of an opfibration $p\colon \mathcal{E} \to \mathcal{B}$ is a functor $s\colon \mathcal{B} \to \mathcal{E}$ with $es = id$.
7. A **poset opfibration** is an opfibration such that for each $I \in \mathcal{B}$ the fibre $\mathcal{E}_I$ is a poset.
8. A **$\mathcal{B}$-indexed poset** is a functor $\mathcal{H}\colon \mathcal{B} \to \mathbf{Pos}$.

All opfibrations we consider below are poset opfibrations. They are effectively interchangeable with indexed posets via the **Grothendieck construction**:

1. Given a poset opfibration $p\colon \mathcal{E} \to \mathcal{B}$ one defines an indexed poset $\mathcal{H}_p\colon \mathcal{B} \to \mathbf{Pos}$ as follows. Note first that every $\mathcal{B}$-morphism $I \xrightarrow{u} J$ with an object X above I has a *unique* opcartesian lifting $X \xrightarrow{f} u^*X$ because $\mathcal{E}_J$ is a poset. Then $\mathcal{H}_p$ is defined by
$$I \mapsto \mathcal{E}_I \quad \text{and} \quad \left(I \xrightarrow{u} J\right) \mapsto \left(\mathcal{E}_I \xrightarrow{u^*} \mathcal{E}_J\right)$$
where u^* maps X to u^*X.
2. Conversely, given an indexed poset $\mathcal{H}\colon \mathcal{B} \to \mathbf{Pos}$, define the **Grothendieck completion** of $\mathcal{H}$ to be the category $\int \mathcal{H}$ with

   objects (I, x) where $I \in \mathcal{B}$ and $x \in \mathcal{H}I$;
   morphisms $(I, x) \xrightarrow{u} (J, y)$ where $I \xrightarrow{u} J$ is a morphism in $\mathcal{B}$ with $\mathcal{H}u(x) \leq_{\mathcal{H}J} y$.

   Then the projection functor $p_{\mathcal{H}}\colon \int \mathcal{H} \to \mathcal{B}$ mapping (I, x) to I and $(I, x) \xrightarrow{u} (J, y)$ to $I \xrightarrow{u} J$ is an opfibration.

The Grothendieck construction gives rise to an equivalence between suitable 2-categories of indexed posets and opfibrations. We only need the following weaker statement:

▶ **Theorem 3.2** (Grothendieck). *Every poset opfibration $p\colon \mathcal{E} \to \mathcal{B}$ is isomorphic to $p_{\mathcal{H}_p}\colon \mathcal{E} \to \mathcal{B}$, and every indexed poset $\mathcal{H}\colon \mathcal{B} \to$ **Pos** is naturally isomorphic to $\mathcal{H}_{p_\mathcal{H}}\colon \mathcal{B} \to$ **Pos**. Furthermore, if $\mathcal{H}, \mathcal{H}'\colon \mathcal{B} \to$ **Pos** are two naturally isomorphic indexed posets then $p_\mathcal{H}, p_{\mathcal{H}'}$ are isomorphic opfibrations.*

3.1 Local pseudovarieties of $\mathfrak{D}$-monoids as an opfibration

In this section we organise the local pseudovarieties of $\mathcal{D}$-monoids into an opfibration $\mathbf{LPV} \to \mathbf{Free}(\mathbf{Mon}\mathcal{D})$, or equivalently into an indexed poset $\mathbf{Free}(\mathbf{Mon}\mathcal{D}) \to \mathbf{Pos}$. The base category $\mathbf{Free}(\mathbf{Mon}\mathcal{D})$ is the category of finitely generated free $\mathcal{D}$-monoids: its objects are finite sets Σ, and its morphisms $\Sigma \xrightarrow{f} \Delta$ are all $\mathcal{D}$-monoid morphisms $\Psi\Sigma^* \xrightarrow{f} \Psi\Delta^*$ between the free $\mathcal{D}$-monoids on Σ and Δ, respectively. Hence $\mathbf{Free}(\mathbf{Mon}\mathcal{D})$ is dual to the Lawvere theory of the variety $\mathbf{Mon}\mathcal{D}$.

▶ **Definition 3.3.** The indexed poset $(-)_\sharp\colon \mathbf{Free}(\mathbf{Mon}\mathcal{D}) \to \mathbf{Pos}$ is defined as follows:

1. To each finite set Σ it assigns the poset $\Sigma_\sharp = \mathbf{LPV}_\Sigma$ of all local pseudovarieties of $\mathcal{D}$-monoids over Σ, ordered by *reverse* inclusion $\supseteq$.

2. To each $\mathcal{D}$-monoid morphism $f\colon \Psi\Sigma^* \to \Psi\Delta^*$ it assigns the monotone map $f_\sharp\colon \mathbf{LPV}_\Sigma \to \mathbf{LPV}_\Delta$, where for $P \in \mathbf{LPV}_\Sigma$ the local pseudovariety $f_\sharp(P) \in \mathbf{LPV}_\Delta$ consists of all finite Δ-generated $\mathcal{D}$-monoids N with $e_N f = g e_M$ for some $M \in P$ and some morphism g; see Diagram 2.

▶ **Lemma 3.4.** $(-)_\sharp$ *is a well-defined functor.*

The Grothendieck construction applied to the indexed poset $(-)_\sharp\colon \mathbf{Free}(\mathbf{Mon}\mathcal{D}) \to \mathbf{Pos}$ yields the following equivalent opfibration:

▶ **Definition 3.5.** The category **LPV** of local pseudovarieties of $\mathcal{D}$-monoids has

objects (Σ, P) where P is a local pseudovariety of $\mathcal{D}$-monoids over Σ;

morphisms $(\Sigma, P) \xrightarrow{f} (\Delta, Q)$ where $f\colon \Psi\Sigma^* \to \Psi\Delta^*$ is a $\mathcal{D}$-monoid morphism such that for every $N \in Q$ there exists $M \in P$ and $g\colon M \to N$ subject to Diagram 2.

The projection $\mathbf{LPV} \xrightarrow{q} \mathbf{Free}(\mathbf{Mon}\mathcal{D})$ mapping (Σ, P) to Σ and $(\Sigma, P) \xrightarrow{f} (\Delta, Q)$ to f is called the **opfibration of local pseudovarieties of $\mathcal{D}$-monoids**.

3.2 Local varieties of languages in $\mathcal{C}$ as an opfibration

In complete analogy to Definition 3.3 and 3.5 we can define an indexed poset and its corresponding opfibration representing local varieties of languages in $\mathcal{C}$.

▶ **Definition 3.6.** The indexed poset $(-)_*\colon \mathbf{Free}(\mathbf{Mon}\mathcal{D}) \to \mathbf{Pos}$ is defined as follows:

1. To each finite set Σ it assigns the poset $\Sigma_* = \mathbf{LAN}_\Sigma$ of all local varieties of languages over Σ in $\mathcal{C}$, ordered by *reverse* inclusion $\supseteq$.
2. To each $\mathcal{D}$-monoid morphism $f\colon \Psi\Sigma^* \to \Psi\Delta^*$ it assigns the monotone map $f_*\colon \mathbf{LAN}_\Sigma \to \mathbf{LAN}_\Delta$, where for $V \in \mathbf{LAN}_\Sigma$ the local variety $f_*(V) \in \mathbf{LAN}_\Delta$ is the directed union of all local varieties W satisfying Diagram 1 for some h. In other words, $f_*(V)$ is the *largest* local variety of languages over Δ such that V is closed under f-preimages of languages in $f_*(V)$.

The Grothendieck construction gives the following opfibration:

▶ **Definition 3.7.** The category **LAN** of local varieties of languages in $\mathcal{C}$ has

objects (Σ, V) where V is a local variety of languages over Σ in $\mathcal{C}$;

morphisms $(\Sigma, V) \xrightarrow{f} (\Delta, W)$ where $f\colon \Psi\Sigma^* \to \Psi\Delta^*$ is a $\mathcal{D}$-monoid morphism such that V is closed under f-preimages of languages in W.

The projection $\mathbf{LAN} \xrightarrow{p} \mathbf{Free}(\mathbf{Mon}\mathcal{D})$ mapping (Σ, V) to Σ and $(\Sigma, V) \xrightarrow{f} (\Delta, W)$ to f is called the **opfibration of local varieties of languages in $\mathcal{C}$**.

The General Local Variety Theorem (see Theorem 2.15) implies that the two indexed posets $(-)_\sharp, (-)_*\colon \mathbf{Free}(\mathbf{Mon}\mathcal{D}) \to \mathbf{Pos}$ of Definition 3.3 and 3.6 are naturally isomorphic. Indeed, recall from Remark 2.16 that the isomorphism $\mathbf{LPV}_\Sigma \cong \mathbf{LAN}_\Sigma$ sends a local pseudovariety $P \in \mathbf{LPV}_\Sigma$ to the directed union of all finite local varieties of languages over Σ in $\mathcal{C}$ corresponding to the finite Σ-generated $\mathcal{D}$-monoids in P. From this and Lemma 2.18 we conclude that the diagram below commutes for all $\mathcal{D}$-monoid morphisms $f\colon \Psi\Sigma^* \to \Psi\Delta^*$.

$$\begin{tikzcd}
\mathbf{LPV}_\Sigma \arrow[r, "\cong"] \arrow[d, "f_\sharp"'] & \mathbf{LAN}_\Sigma \arrow[d, "f_*"] \\
\mathbf{LPV}_\Delta \arrow[r, "\cong"'] & \mathbf{LAN}_\Delta
\end{tikzcd}$$

Hence, by Theorem 3.2, we get an isomorphism between the corresponding opfibrations:

▶ **Theorem 3.8.** *The opfibrations* $p\colon \mathbf{LAN} \to \mathbf{Free}(\mathbf{Mon}\mathcal{D})$ *and* $q\colon \mathbf{LPV} \to \mathbf{Free}(\mathbf{Mon}\mathcal{D})$ *are isomorphic.*

▶ **Definition 3.9.** By a **variety of languages in** $\mathcal{C}$ is meant a global section of p, i.e. a functor $\mathcal{V}\colon \mathbf{Free}(\mathbf{Mon}\mathcal{D}) \to \mathbf{LAN}$ with $p\mathcal{V} = id$.

In more concrete terms, a variety of languages in $\mathcal{C}$ is given by a collection of local varieties $V_\Sigma \in \mathbf{LAN}_\Sigma$ (where Σ ranges over all finite alphabets) such that for every $f\colon \Psi\Sigma^* \to \Psi\Delta^*$ the local variety V_Σ is closed under f-preimages of languages in V_Δ. For $\mathcal{C} = \mathbf{BA}$ the above definition corresponds to Eilenberg's original concept, see Introduction. Similarly, varieties of languages in **DLat**, **SLat** and $\mathbb{Z}_2$-**Vec** are precisely the positive varieties of Pin [14], the disjunctive varieties of Polák [16] and the xor varieties of Reutenauer [18], respectively.

Theorem 3.8 implies that every global section of $p\colon \mathbf{LAN} \to \mathbf{Free}(\mathbf{Mon}\mathcal{D})$ corresponds uniquely to a global section of $q\colon \mathbf{LPV} \to \mathbf{Free}(\mathbf{Mon}\mathcal{D})$. In the next section we will see that also the global sections of q admit a concrete interpretation.

4 Profinite $\mathcal{D}$-Monoids

A **profinite** $\mathcal{D}$**-monoid** is a cofiltered limit of finite $\mathcal{D}$-monoids, and the **profinite completion** $\widehat{M}$ of a $\mathcal{D}$-monoid M is the cofiltered limit of the diagram of all its finite quotients. Since limits in $\mathbf{Mon}\mathcal{D}$ are formed on the level of **Set**, every profinite $\mathcal{D}$-monoid is equipped with a profinite topology, i.e. it can be viewed as a Stone space if $\mathcal{D}$ is a variety of algebras, resp. an ordered Stone space if $\mathcal{D}$ is a variety of ordered algebras.[2] By $\mathsf{Pro}\mathbf{Mon}_f\mathcal{D}$ denote the category of profinite $\mathcal{D}$-monoids with continuous (order-preserving) $\mathcal{D}$-monoid morphisms.

▶ **Theorem 4.1.** *1.* $\mathsf{Pro}\mathbf{Mon}_f\mathcal{D}$ *is the pro-completion of the category* $\mathbf{Mon}_f\mathcal{D}$ *of finite* $\mathcal{D}$*-monoids, cf. Remark 2.4.*

2. The profinite completion $M \mapsto \widehat{M}$ *gives a left adjoint to the forgetful functor* $\mathsf{Pro}\mathbf{Mon}_f\mathcal{D} \to \mathbf{Mon}\mathcal{D}$.

The first item follows from [11, Proposition VI.2.4 and Remark VI.2.5]. The argument given there for varieties of algebras also applies to ordered algebras. The second item follows from a standard argument for ordinary monoids, see e.g. [19, Theorem 3.2.7].

▶ **Example 4.2.** For our predual categories $\mathcal{C}/\mathcal{D}$ of Example 2.3 we obtain the following descriptions of the categories $\mathsf{Pro}\mathcal{D}_f$, $\mathbf{Mon}\mathcal{D}$ and $\mathsf{Pro}\mathbf{Mon}_f\mathcal{D}$, cf. [11, Theorem VI.2.9].

$\mathcal{C}$	$\mathcal{D}$	$\mathsf{Pro}\mathcal{D}_f$	$\mathbf{Mon}\mathcal{D}$	$\mathsf{Pro}\mathbf{Mon}_f\mathcal{D}$
BA	**Set**	**Stone**	**Mon**	**Stone**(**Mon**)
DLat	**Pos**	**OStone**	**OMon**	(to be characterised)
SLat	**SLat**	**Stone**(**SLat**)	**ISRing**	**Stone**(**ISRing**)
$\mathbb{Z}_2$-**Vec**	$\mathbb{Z}_2$-**Vec**	**Stone**($\mathbb{Z}_2$-**Vec**)	$\mathbb{Z}_2$-**Alg**	**Stone**($\mathbb{Z}_2$-**Alg**)

Stone and **OStone** are the categories of (ordered) Stone spaces and continuous (order-preserving) maps. The categories in the fourth column are the categories of monoids, ordered monoids, idempotent semirings and $\mathbb{Z}_2$-algebras, respectively; see Example 2.11. By

[2] An **(ordered) Stone space** is a compact space such that for every $x \neq y$ (resp. $x \not\leq y$) there exists a clopen (upper) set containing x but not y.

$\mathbf{Stone}(\mathcal{A})$ for a variety of algebras $\mathcal{A}$ we mean the category of $\mathcal{A}$-algebras in **Stone**. For example, $\mathbf{Stone}(\mathbf{Mon})$ is the category of monoids equipped with a Stone topology (making the monoid multiplication continuous) and continuous monoid morphisms.

4.1 Local pseudovarieties of $\mathcal{D}$-monoids vs. profinite $\mathcal{D}$-monoids

In this section we show how to identify local pseudovarieties of $\mathcal{D}$-monoids over Σ with Σ-generated profinite $\mathcal{D}$-monoids. In the following **quotients** of profinite $\mathcal{D}$-monoids are meant to be represented by surjective continuous $\mathcal{D}$-monoid morphisms. A **Σ-generated profinite $\mathcal{D}$-monoid** is a quotient of $\widehat{\Psi\Sigma^*}$, the profinite completion of the free $\mathcal{D}$-monoid $\Psi\Sigma^*$. Note that, by Theorem 4.1, $\widehat{\Psi\Sigma^*}$ is the free profinite $\mathcal{D}$-monoid on the free $\mathcal{D}$-monoid $\Psi\Sigma^*$ w.r.t. the forgetful functor $\mathsf{Pro}\mathbf{Mon}_f\mathcal{D} \to \mathbf{Mon}\mathcal{D}$, and hence also the free profinite $\mathcal{D}$-monoid on the set Σ w.r.t. the composite forgetful functor $\mathsf{Pro}\mathbf{Mon}_f\mathcal{D} \to \mathbf{Mon}\mathcal{D} \to \mathbf{Set}$. The following standard facts will be useful.

▶ **Lemma 4.3** (see e.g. [19, Chapter 3]). *Let $F\colon \mathcal{J} \to \mathbf{KHaus}$ be a cofiltered diagram in the category of compact Hausdorff spaces and continuous functions.*

1. *If every $F_i \xrightarrow{Ff} F_j$ for $i \xrightarrow{f} j$ is surjective, then the limit projections $\operatorname{Lim} F \xrightarrow{\pi_i} F_i$ are also surjective.*
2. *If $\varphi\colon \Delta X \Rightarrow F$ is a cone over F such that every projection $\varphi_i\colon X \to F_i$ is surjective, then the mediating morphism $X \to \operatorname{Lim} F$ is also surjective.*

▶ **Remark 4.4.** 1. To each local pseudovariety $P \in \mathbf{LPV}_\Sigma$ we associate a Σ-generated profinite $\mathcal{D}$-monoid as follows. Note first that P defines a cofiltered diagram in $\mathsf{Pro}\mathbf{Mon}\mathcal{D}$ via the projection $(e\colon \Psi\Sigma^* \twoheadrightarrow M) \mapsto M$. Since the connecting morphisms are surjective, the above lemma implies that every limit projection $\operatorname{Lim} P \to M$ for $M \in P$ is surjective. Moreover, given $P \subseteq P'$ in $\mathbf{LPV}_\Sigma$, there is a surjective mediating morphism $h\colon \operatorname{Lim} P' \to \operatorname{Lim} P$. In particular, taking P' to be the local pseudovariety of *all* finite quotients of $\Psi\Sigma^*$ with $\operatorname{Lim} P' = \widehat{\Psi\Sigma^*}$ we get a surjective morphism $\widehat{\Psi\Sigma^*} \twoheadrightarrow \operatorname{Lim} P$, i.e. a Σ-generated profinite $\mathcal{D}$-monoid.

2. Conversely, to each Σ-generated profinite $\mathcal{D}$-monoid $e_\Sigma\colon \Psi\Sigma^* \twoheadrightarrow F\Sigma$ we associate a local pseudovariety $\mathcal{V}_{F\Sigma} \in \mathbf{LPV}_\Sigma$ as follows: $\mathcal{V}_{F\Sigma}$ consists of all finite Σ-generated $\mathcal{D}$-monoids of the form $\Psi\Sigma^* \xrightarrow{\eta} \widehat{\Psi\Sigma^*} \xrightarrow{e_\Sigma} F\Sigma \xrightarrow{e_M} M$, where η is the universal arrow of the adjunction between $\mathsf{Pro}\mathbf{Mon}_f\mathcal{D}$ and $\mathbf{Mon}\mathcal{D}$ (see Theorem 4.1) and M is any finite quotient of $F\Sigma$. Observe that such a composite $e_M e_\Sigma \eta$ is always surjective: since $\widehat{\Psi\Sigma^*}$ is the limit of all finite quotients of $\Psi\Sigma^*$, and M is finite (hence a finitely copresentable object of $\mathsf{Pro}\mathbf{Mon}\mathcal{D}$), the morphism $e_M e_\Sigma$ factorises through some limit projection π_N, where N is a finite quotient of $\Psi\Sigma^*$:

$$\begin{tikzcd}
\Psi\Sigma^* \arrow[r, "\eta"] \arrow[dr] & \widehat{\Psi\Sigma^*} \arrow[r, two heads, "e_\Sigma"] \arrow[d, two heads, "\pi_N"'] & F\Sigma \arrow[d, two heads, "e_M"] \\
& N \arrow[r, two heads, "f"'] & M
\end{tikzcd}$$

It is not difficult to see that the two constructions of Remark 4.4 are mutually inverse. More precisely:

▶ **Theorem 4.5.** *Let Σ be a finite set.*

1. *Every Σ-generated profinite $\mathcal{D}$-monoid $F\Sigma$ corresponds uniquely to a local pseudovariety $\mathcal{V}_{F\Sigma}$ of $\mathcal{D}$-monoids over Σ. That is,*

$$\mathsf{Quo}(\widehat{\Psi\Sigma^*}) \cong \mathbf{LPV}_\Sigma,$$

where $\mathsf{Quo}(\widehat{\Psi\Sigma^*})$ denotes the poset of Σ-generated profinite $\mathcal{D}$-monoids.

2. *Let $f\colon \Psi\Sigma^* \to \Psi\Delta^*$ be a $\mathcal{D}$-monoid morphism, $F\Sigma$ a Σ-generated profinite $\mathcal{D}$-monoid and $F\Delta$ a Δ-generated profinite $\mathcal{D}$-monoid. Then the right-hand diagram below commutes for some h iff for every $N \in \mathcal{V}_{F\Delta}$ there is some $M \in \mathcal{V}_{F\Sigma}$ and a morphism h_N making the left-hand diagram commute:*

$$\begin{tikzcd}
\Psi\Sigma^* \arrow[r, "f"] \arrow[d, two heads] & \Psi\Delta^* \arrow[d, two heads] \\
M \arrow[r, "h_N"'] & N
\end{tikzcd}
\qquad
\begin{tikzcd}
\widehat{\Psi\Sigma^*} \arrow[r, "\widehat{f}"] \arrow[d, two heads] & \widehat{\Psi\Delta^*} \arrow[d, two heads] \\
F\Sigma \arrow[r, "h"'] & F\Delta
\end{tikzcd}$$

From the opfibration $q\colon \mathbf{LPV} \to \mathbf{Free}(\mathbf{Mon}\mathcal{D})$ we thus get the following isomorphic opfibration:

▶ **Definition 4.6.** The category **PFMon** has

objects $(\Sigma, F\Sigma)$ where $F\Sigma$ is a Σ-generated profinite $\mathcal{D}$-monoid;

morphisms $(\Sigma, F\Sigma) \xrightarrow{f} (\Delta, F\Delta)$ where $f\colon \Psi\Sigma^* \to \Psi\Delta^*$ is a $\mathcal{D}$-monoid morphism making the following diagram commute for some h:

$$\begin{tikzcd}
\widehat{\Psi\Sigma^*} \arrow[r, "\widehat{f}"] \arrow[d, two heads] & \widehat{\Psi\Delta^*} \arrow[d, two heads] \\
F\Sigma \arrow[r, "h"'] & F\Delta
\end{tikzcd} \tag{1}$$

The projection $\mathbf{PFMon} \xrightarrow{q'} \mathbf{Free}(\mathbf{Mon}\mathcal{D})$ sending $(\Sigma, F\Sigma)$ to Σ and $(\Sigma, F\Sigma) \xrightarrow{f} (\Delta, F\Delta)$ to f is called the **opfibration of finitely generated profinite $\mathcal{D}$-monoids**.

▶ **Corollary 4.7.** *The opfibrations $q\colon \mathbf{LPV} \to \mathbf{Free}(\mathbf{Mon}\mathcal{D})$ and $q'\colon \mathbf{PFMon} \to \mathbf{Free}(\mathbf{Mon}\mathcal{D})$ are isomorphic.*

4.2 Pseudovarieties of $\mathcal{D}$-monoids vs. profinite equational theories

By a **pseudovariety of $\mathcal{D}$-monoids** is meant a class of finite $\mathcal{D}$-monoids closed under submonoids, quotients and finite products. In this section we relate pseudovarieties of $\mathcal{D}$-monoids to profinite equational theories of $\mathcal{D}$-monoids.

▶ **Definition 4.8.** A **profinite equational theory** of $\mathcal{D}$-monoids is a global section $\mathcal{T}\colon \mathbf{Free}(\mathbf{Mon}\mathcal{D}) \to \mathbf{PFMon}$ of the opfibration $q'\colon \mathbf{PFMon} \to \mathbf{Free}(\mathbf{Mon}\mathcal{D})$.

More explicitly, a profinite equational theory associates to each finite set Σ a Σ-generated profinite monoid $e_\Sigma\colon \widehat{\Psi\Sigma^*} \twoheadrightarrow F\Sigma$ such that, for all $f\colon \Psi\Sigma^* \to \Psi\Delta^*$, diagram (1) commutes for some h.

▶ **Remark 4.9.**

1. To each profinite equational theory $\mathcal{T}$ with $\mathcal{T}\Sigma = (\Sigma, F\Sigma)$ we associate a pseudovariety $\mathcal{V}$ of $\mathcal{D}$-monoids as follows: $\mathcal{V}$ consists of all finite $\mathcal{D}$-monoids M such that for all $\mathcal{D}$-monoid morphisms $f\colon \widehat{\Psi\Sigma^*} \to M$ there exists a (necessarily unique) $\mathcal{D}$-monoid morphism $\overline{f}\colon F\Sigma \to M$ with $\overline{f}e_\Sigma = f$.

$$\begin{tikzcd}
\widehat{\Psi\Sigma^*} \arrow[r, two heads, "e_\Sigma"] \arrow[dr, "f"'] & F\Sigma \arrow[d, dashed, "\overline{f}"] \\
 & M
\end{tikzcd}$$

2. Conversely, to each pseudovariety $\mathcal{V}$ of $\mathcal{D}$-monoids we associate a profinite equational theory $\mathcal{T}$ with $\mathcal{T}\Sigma = (\Sigma, F\Sigma)$ as follows: given Σ, form the local pseudovariety P_Σ of all Σ-generated finite $\mathcal{D}$-monoids $e\colon \Psi\Sigma^* \twoheadrightarrow M$ with $M \in \mathcal{V}$. Then $F\Sigma$ is the Σ-generated profinite $\mathcal{D}$-monoid defined by P_Σ, see Remark 4.4 and Theorem 4.5.

Again, these constructions are mutually inverse:

▶ **Theorem 4.10.** *The maps $\mathcal{T} \mapsto \mathcal{V}$ and $\mathcal{V} \mapsto \mathcal{T}$ define a bijective correspondence between profinite equational theories and pseudovarieties of $\mathcal{D}$-monoids.*

▶ **Remark 4.11.** This theorem can be viewed as a categorical presentation of the well-known Reiterman-Banaschewski correspondence [17, 5]. The difference lies in the definition of a profinite theory: Reiterman and Banaschewski work with profinite equations (i.e. pairs of elements of free profinite monoids) while we work with quotients of free profinite monoids.

5 Eilenberg-type Correspondences

Putting the results of our paper together we will now derive a number of Eilenberg-type theorems. Each of these theorems is an immediate consequence of the isomorphisms we established between our opfibrations p, q and q' (see the diagram in the Introduction) and the characterisation of their global sections. First, by Theorem 4.5 we get another version of the General Local Variety Theorem, i.e. Theorem 2.15.

▶ **Theorem 5.1** (General Local Variety Theorem II)**.** *There is a one-to-one correspondence between local varieties of languages over Σ in $\mathcal{C}$ and Σ-generated profinite $\mathcal{D}$-monoids:*

$$\mathbf{LAN}_\Sigma \cong \mathsf{Quo}(\widehat{\Psi\Sigma^*}).$$

By Theorem 3.8, Corollary 4.7, and Theorem 4.10, we recover the main result of [2]:

▶ **Theorem 5.2** (General Variety Theorem)**.** *There is a one-to-one correspondence between varieties of languages in $\mathcal{C}$ and pseudovarieties of $\mathcal{D}$-monoids.*

An interesting generalisation of this theorem emerges by restricting $\mathbf{Free}(\mathbf{Mon}\mathcal{D})$ to a subcategory. Recall that the pullback in $\mathbf{Cat}$ of an opfibration $p\colon \mathcal{E} \to \mathcal{B}$ along any functor $F\colon \mathcal{B}' \to \mathcal{B}$ is again an opfibration, see e.g. [10, Lemma 1.5.1].

▶ **Definition 5.3.** For a subcategory $\mathsf{C} \hookrightarrow \mathbf{Free}(\mathbf{Mon}\mathcal{D})$, a **$\mathsf{C}$-variety of languages in $\mathcal{C}$** is a global section of the opfibration $p_{\mathsf{C}}\colon \mathbf{LAN}_{\mathsf{C}} \to \mathsf{C}$ obtained as the pullback of the opfibration p along the inclusion. Similarly, a **profinite equational C-theory of $\mathcal{D}$-monoids** is a global section of the opfibration $q'_{\mathsf{C}} \colon \mathbf{PFMon}_{\mathsf{C}} \to \mathsf{C}$ obtained as the pullback of $q' \colon \mathbf{PFMon} \to \mathbf{Free}(\mathbf{Mon}\mathcal{D})$ along the inclusion.

$$\begin{tikzcd}
\mathbf{LAN}_{\mathsf{C}} \arrow[r, hook] \arrow[d, "p_{\mathsf{C}}"'] \arrow[dr, phantom, "\lrcorner", very near start] & \mathbf{LAN} \arrow[d, "p"] \\
\mathsf{C} \arrow[r, hook] & \mathbf{Free}(\mathbf{Mon}\mathcal{D})
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathbf{PFMon}_{\mathsf{C}} \arrow[r, hook] \arrow[d, "q'_{\mathsf{C}}"'] \arrow[dr, phantom, "\lrcorner", very near start] & \mathbf{PFMon} \arrow[d, "q'"] \\
\mathsf{C} \arrow[r, hook] & \mathbf{Free}(\mathbf{Mon}\mathcal{D})
\end{tikzcd}$$

More explicitly, a profinite equational C-theory associates to each $\Sigma \in \mathsf{C}$ a Σ-generated profinite monoid $e_\Sigma \colon \widehat{\Psi\Sigma^*} \twoheadrightarrow F\Sigma$ such that, for all $f \colon \Psi\Sigma^* \to \Psi\Delta^*$ in C, diagram (1) commutes for some h. Similarly, a C-variety of languages determines a family $(V_\Sigma)_{\Sigma\in\mathsf{C}}$, where V_Σ is a local variety of languages over Σ in $\mathcal{C}$ and, for each $f \colon \Psi\Sigma^* \to \Psi\Delta^*$ in C, the local variety V_Σ is closed under f-preimages of languages in V_Δ. For the case where $\mathcal{C} = \mathbf{BA}$, $\mathcal{D} = \mathbf{Set}$ and the subcategory C contains all objects of $\mathbf{Free}(\mathbf{Mon})$, this definition coincides with the concept of a C-variety of languages introduced by Straubing [20]. He also proved a special case of Theorem 5.4 below. Observe that since the opfibrations p and q' are isomorphic, so are their pullbacks p_{C} and q'_{C}. Therefore:

▶ **Theorem 5.4** (General Variety Theorem for C-varieties of languages). *There is a one-to-one correspondence between C-varieties of languages in $\mathcal{C}$ and profinite equational C-theories of $\mathcal{D}$-monoids.*

As an application of this theorem, let us choose C to be the full subcategory of $\mathbf{Free}(\mathbf{Mon}\mathcal{D})$ on a single object Σ. Then a C-variety of languages in $\mathcal{C}$ is precisely a local variety of languages over Σ in $\mathcal{C}$ closed under preimages of $\mathcal{D}$-monoid endomorphisms $f \colon \Psi\Sigma^* \to \Psi\Sigma^*$. We call such a local variety **fully invariant**. A profinite equational C-theory consists of a single Σ-generated profinite $\mathcal{D}$-monoid $e \colon \widehat{\Psi\Sigma^*} \twoheadrightarrow F\Sigma$ such that, for all $\mathcal{D}$-monoid endomorphisms $f \colon \Psi\Sigma^* \to \Psi\Sigma^*$, $e\widehat{f}$ factors through e.

$$\begin{tikzcd}
\widehat{\Psi\Sigma^*} \arrow[r, "\widehat{f}"] \arrow[d, two heads, "e"'] & \widehat{\Psi\Sigma^*} \arrow[d, two heads, "e"] \\
F\Sigma \arrow[r, dashed] & F\Sigma
\end{tikzcd}$$

Again, such a Σ-generated profinite $\mathcal{D}$-monoid is called **fully invariant**. Hence full invariance means precisely that (in-)equalities are stable under translations, i.e. for every $x, y \in \widehat{\Psi\Sigma^*}$ and $f \colon \Psi\Sigma^* \to \Psi\Sigma^*$ we have that $e(x) = e(y)$ implies $e(\widehat{f}x) = e(\widehat{f}y)$ if $\mathcal{D}$-algebras are unordered; in the case that $\mathcal{D}$-algebras are ordered, $e(x) \le e(y)$ implies $e(\widehat{f}x) \le e(\widehat{f}y)$. Therefore Theorem 5.4 gives the following:

▶ **Theorem 5.5** (Local Variety Theorem for Fully Invariant Varieties). *There is a one-to-one correspondence between fully invariant local varieties over Σ in $\mathcal{C}$ and fully invariant Σ-generated profinite $\mathcal{D}$-monoids.*

▶ **Remark 5.6.** One may compare the approaches in the present paper and [2] as follows. In [2] the authors form a functor $\rho T \colon \mathbf{Set}_f^{op} \to \mathcal{C}$ assigning to each Σ the rational fixpoint $\rho T_\Sigma = \mathsf{Reg}(\Sigma)$, see Proposition 2.7, and define a variety of languages in $\mathcal{C}$ to be subfunctor V of ρT such that each $V\Sigma$ is a local variety of languages and closed under preimages of $\mathcal{D}$-monoid morphisms. The General Variety Theorem is then derived in a purely *order-theoretic* way: one proves that the complete lattice of all varieties of languages is algebraic, establishes an Eilenberg-type correspondence for its compact elements, and proceeds by ideal completion. In comparison to our present fibrational setting, neither the General Local Variety Theorem nor profinite algebras and the Reiterman-Banaschweski correspondence are used in [2]. Also the generalisation to C-varieties is not immediate in the functorial setting of [2].

6 Conclusions and Future Work

In this paper we studied varieties of languages, pseudovarieties of monoids and profinite equational theories from an abstract fibrational viewpoint. This led us to conceptually new proofs and generalisations for a number of Eilenberg-type results.

Our notion of profinite equational theory is introduced on a rather abstract level, and it would be helpful to characterise theories syntactically and compare them with classical developments [17, 5]. If $\mathcal{D}$-algebras are non-ordered, every Σ-generated profinite $\mathcal{D}$-monoid $e\colon \widehat{\Psi\Sigma^*} \twoheadrightarrow M$ is the coequaliser of its kernel pair $\pi_1, \pi_2\colon E \rightrightarrows \widehat{\Psi\Sigma^*}$, where E is the kernel congruence $E = \{\,(u,v) \in \widehat{\Psi\Sigma^*} \times \widehat{\Psi\Sigma^*} \mid e(u) = e(v)\,\}$. Hence a profinite equational theory corresponds to a family of *profinite equations*, i.e. pairs of elements of a free profinite monoid. From this observation it should be possible to obtain syntactic counterparts of our results, e.g. a generalisation of the main result of Gehrke et al. [8] that local varieties of languages in **BA** and **DLat** are definable by profinite identities.

In addition, it would be useful to develop a notion of *morphism* between profinite equational theories, and correspondingly between varieties of languages, hence lifting our generalised Eilenberg-Reiterman correspondences from an isomorphism of posets to an equivalence of categories. Such a result may further justify the importance of a categorical treatment of algebraic automata theory.

References

1 J. Adámek, S. Milius, R. S. Myers, and H. Urbat. Generalized Eilenberg Theorem I: Local Varieties of Languages. In A. Muscholl (ed.) *Found. Softw. Sci. Comput. Struct.* LNCS, vol. 8412, pp. 366–380. Springer Berlin Heidelberg, 2014. arXiv:1501.02834 [cs.FL]

2 J. Adámek, R. S. Myers, S. Milius, and H. Urbat. Varieties of Languages in a Category. Accepted for LICS 2015. arXiv:1501.05180 [cs.FL]

3 J. Adámek, S. Milius and J. Velebil. Iterative Algebras at Work. *Math. Structures Comput. Sci.*, 16 (6), pp. 1085–1131, 2006.

4 B. Banaschewski and E. Nelson. Tensor products and bimorphisms. Canad. Math. Bull. 19, pp. 385–402, 1976.

5 B. Banaschewski. The Birkhoff Theorem for varieties of finite algebras. *Algebr. universalis*, 17(1):360–368, 1983.

6 F. Borçeux. Handbook of Categorical Algebra: Volume 2, Categories and Structures. Cambridge University Press, 1994.

7 S. Eilenberg. *Automata, Languages, and Machines Vol. B*, Academic Press, New York, 1976.

8 M. Gehrke, S. Grigorieff, and J.-É. Pin. Duality and equational theory of regular languages. In *Autom. Lang. Program.* LNCS, vol. 5126, pp. 246–257. Springer Berlin Heidelberg, 2008.

9 C. Hermida and B. Jacobs. Structural induction and coinduction in a fibrational setting. *Inf. Comput.*, 145(2):107–152, 1998.

10 B. Jacobs. *Categorical Logic and Type Theory.* North Holland, Amsterdam, 1999.

11 P. T. Johnstone. *Stone spaces.* Cambridge University Press, 1982.

12 A. Kock. Monads on symmetric monoidal closed categories *Arch. Math.* 21:1–10, 1970

13 S. Milius. A sound and complete calculus for finite stream circuits. *25th Annu. IEEE Symp. Log. Comput. Sci.*, pp. 421–430, 2010.

14 J.-É. Pin. A variety theorem without complementation. *Russ. Math. (Iz. VUZ)*, 39:80–90, 1995.

15 N. Pippenger. Regular languages and stone duality. *Theory Comput. Syst.*, 30(2):121–134, 1997.

16 L. Polák. Syntactic semiring of a language. In J. Sgall, A. Pultr, and P. Kolman (eds.) *Math. Found. Comput. Sci.* LNCS, vol. 2136, pp. 611–620. Springer Berlin Heidelberg, 2001.

17 J. Reiterman. The Birkhoff theorem for finite algebras. *Algebr. Universalis*, 14(1):1–10, 1982.

18 C. Reutenauer. Séries formelles et algèbres syntactiques. *J. Algebr.*, 66(2):448–483, 1980.

19 J. Rhodes and B. Steinberg. *The q-theory of Finite Semigroups.* Springer US, 2009.

20 H. Straubing. On logical descriptions of regular languages. In S. Rajsbaum (ed.) *LATIN 2002 Theor. Informatics.* LNCS, vol. 2286, pp. 528–538. Springer Berlin Heidelberg, 2002.

Canonical Coalgebraic Linear Time Logics

Corina Cîrstea

University of Southampton, UK
cc2@ecs.soton.ac.uk

Abstract

We extend earlier work on linear time fixpoint logics for coalgebras with branching, by showing how propositional operators arising from the choice of branching monad can be canonically added to these logics. We then consider two semantics for the uniform modal fragments of such logics: the previously-proposed, step-wise semantics and a new semantics akin to those of path-based logics. We prove that the two semantics are equivalent, and show that the canonical choice made for resolving branching in these logics is crucial for this property. We also state conditions under which similar, non-canonical logics enjoy the same property – this applies both to the choice of a branching modality and to the choice of linear time modalities. Our logics allow reasoning about linear time behaviour in systems with non-deterministic, probabilistic or weighted branching. In all these cases, the logics enhanced with propositional operators gain in expressiveness. Another contribution of our work is a reformulation of fixpoint semantics, which applies to any coalgebraic modal logic whose semantics arises from a one-step semantics.

1998 ACM Subject Classification F.1.1 Models of Computation, F.4.1 Modal Logic

Keywords and phrases coalgebra, linear time logic, fixpoint logic

Digital Object Identifier 10.4230/LIPIcs.CALCO.2015.66

1 Introduction

Several recent works focus on the study of trace semantics for coalgebras with branching, and of associated trace logics. The majority of these works concerns *finite* traces, for which a coalgebraic account exists that uses using finality in either the Kleisli [7] or the Eilenberg-Moore category [10] of the monad defining the branching type. Logics for finite traces were studied in [11], with the approach involving a dual adjunction between the category of Eilenberg-Moore algebras of the branching monad and itself.

A canonical, modular approach to defining *maximal* (including infinite) traces for coalgebras with branching was proposed in [1], and linear time coalgebraic fixpoint logics that match this notion of linear time behaviour were studied in [2]. The logics in loc. cit. are interpreted over coalgebras of type $\mathsf{T}F$, with $\mathsf{T} : \mathsf{Set} \to \mathsf{Set}$ a branching monad and $F : \mathsf{Set} \to \mathsf{Set}$ a (typically polynomial) endofunctor. They use modalities arising from the endofunctor F to formalise properties of linear time behaviours, and a hidden modality arising canonically from the branching monad T to abstract away branching at each step. The semantics of these logics is quantitative, and uses $\mathsf{T}1$, with 1 a one-element set, as the domain of truth values. The logics have no propositional operators, and attempting to incorporate them in the usual way at the level of linear time formulas fails, as such operators do not always interact as expected with the branching modality. For example, taking T to be the powerset monad and $F = 1 + A \times \mathsf{Id}$, the associated (two-valued) linear time logics use the standard diamond modality implicitly in their semantics to abstract away branching at each step, and unary modalities

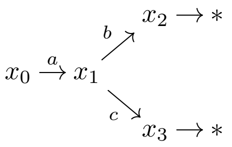


Figure 1

6th International Conference on Algebra and Coalgebra in Computer Science (CALCO'15).
Editors: Lawrence S. Moss and Paweł Sobociński; pp. 66–85
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

$[a]$ (with the formula $[a]\varphi$ stating that an a-transition is observed next, and subsequently φ holds) together with a nullary modality $*$ (expressing successful termination) to formalise properties of F-behaviours. The intention is that a linear time formula (containing modalities $*$ and $[a]$ with $a \in A$) should hold in a state of a $\mathcal{P}F$-coalgebra if there exists a maximal trace from that state satisfying the given formula. Unfortunately, the step-wise semantics makes the addition of a conjunction operator to the logic tricky: the obvious semantics (stipulating that $\varphi \wedge \psi$ holds in a state if both φ and ψ hold in that state) wrongly results in the pointed $\mathcal{P}(1 + A \times \mathsf{Id})$-coalgebra in Figure 1 satisfying the linear time formula $[a][b]* \wedge [a][c]*$, as there is no maximal trace starting in x_0 and satisfying both $[a][b]*$ and $[a][c]*$. The underlying problem is that, by abstracting away branching in a step-wise manner, information about which traces from a given state (x_1 in this case) satisfy a given formula (e.g. $[b]*$ or $[c]*$) is lost. In [2], this issue is dealt with by incorporating restricted versions of propositional operators into (additional) modalities, thereby enhancing the logic for F-behaviours.

Here we take a different approach, by showing how to incorporate propositional operators that arise canonically from the branching monad T into these logics. Our approach resembles that of [11] and involves lifting the logics to the Eilenberg-Moore category of T. This guarantees a smooth interaction between propositional operators and the branching modality, thereby avoiding the previously-mentioned problem. For concrete Ts, the resulting propositional operators add expressiveness to the logics: arbitrary disjunctions for non-deterministic branching, sub-convex combinations for probabilistic branching and linear combinations for weighted branching.

To justify the canonical choice for the branching modality employed by our logics, we provide an alternative, equivalent path-based semantics for their *uniform modal fragments*. Roughly speaking, these fragments only contain formulas whose modal depth is uniformly n for some $n \in \omega$; however, for formulas without variables, the uniformity condition is slightly less restrictive (see Section 5 for details). The definition of the path-based semantics involves the use of a canonical distributive law of T over F to flatten finite-step $\mathsf{T}F$-behaviours into T-branches of finite-step F-behaviours. For example, if $\mathsf{T} = \mathcal{P}$, the alternative semantics exactly captures the idea that a linear time formula holds in a state of a $\mathcal{P}F$-coalgebra if there exists a maximal trace from that state satisfying the given formula. The equivalence result crucially depends on the choice of branching modality. Its proof assumes canonical choices also for the linear time modalities, but a generalisation to logics where these modalities are not canonical is subsequently stated. In particular, this generalisation applies to modalities incorporating restricted versions of propositional operators, as used in [2].

Our technical approach relies on rephrasing the logics of [2] in the, by now standard, dual adjunction framework (originating with [16, 15] and later generalised by several other authors). In addition, we show how fixpoint logics can be accounted for in this framework, by exploiting the existence of a *coalgebraic structure* on their modal fragments.

Our results hold for coalgebras of arbitrary compositions of polynomial endofunctors with (possibly several occurrences of) a single branching monad on the category Set. However, for simplicity of presentation, this paper restricts to $\mathsf{T}F$-coalgebras.

A key assumption of the paper is that the branching monad T is commutative and *partially additive* (see Section 2). If, in addition to being partially additive, T is also assumed to be finitary, then one can show (see Remark 2.4) that T is isomorphic to a *weighted monad*, that is $\mathsf{T}X \simeq S^X$ with $(S, +, 0, \bullet, 1)$ a partial commutative semiring. Our examples of branching monads include finitary ones (modelling weighted branching that arises from partial commutative semirings) as well as infinitary ones (in particular, the full powerset monad and the sub-probability distribution monad). While the latter (infinitary) examples have a similar flavour to weighted branching, they do not strictly fall in this category.

Our contributions are:

1. We rephrase the logics of [2] in the dual adjunction framework (Section 3).
2. We extend these logics with propositional operators arising canonically from the monad T, by moving to the Eilenberg-Moore category of T (Section 4). The enhanced logics gain in expressiveness (see Example 4.4).
3. We show how fixpoint semantics fits into the dual adjunction framework (Sections 3 and 4). Our approach applies to any coalgebraic modal logic defined in this framework.
4. We show that the uniform modal fragments of the logics in [2] and of their canonical extensions with propositional operators admit an equivalent, path-based semantics (Theorems 5.13 and 5.24).
5. We show that this equivalence result depends crucially on the canonical choice for the branching modality. We also state conditions under which other choices for this modality, as used e.g. in [6, 12], enjoy the same property (Theorem 5.16). As an example, this allows the standard box modality to be used in linear time logics for coalgebras with non-deterministic, *non-empty* branching (see Example 5.15).
6. We generalise the equivalence result to other choices of linear time modalities, as used e.g. in [2] (Theorem 5.17).

Given the last two points, this paper is not only about canonical linear time logics, but also about equally well-behaved non-canonical ones.

Related Work. Several quantitative logics for probabilistic systems have been studied in the literature. Among these, the closer in spirit to our logics are perhaps those of [8], which have a linear time flavour with a semantics for modal operators that involves weighted averages. However, unlike the logics considered here, the logics of [8] employ conjunction and disjunction operators with several possible fuzzy interpretations (e.g. minimum or multiplication for conjunctions), none of which is canonical.

Following [2], a similar approach to defining *finite* trace logics has been taken in [12]. The logics in loc. cit. are parametric in the choice of both branching and linear time modalities, but contain neither propositional operators nor fixpoints. Moreover, the consequences of the generality resulting from parametricity are not sufficiently explored. As we show later (Example 5.14), the equivalence of the step-wise semantics with a more standard path-based semantics does not hold in general.

2 Preliminaries

2.1 Partially Additive Monads

We use commutative monads (T, η, μ) on Set (where $\eta : \mathsf{Id} \Rightarrow \mathsf{T}$ and $\mu : \mathsf{T} \circ \mathsf{T} \Rightarrow \mathsf{T}$ are the *unit* and *multiplication* of T) to capture branching in coalgebraic types. We write $\mathsf{st}_{X,Y} : X \times \mathsf{T}Y \to \mathsf{T}(X \times Y)$ and $\mathsf{dst}_{X,Y} : \mathsf{T}X \times \mathsf{T}Y \to \mathsf{T}(X \times Y)$ for the *strength* and respectively *double strength* maps of such a monad. The *swapped strength* map $\mathsf{st}'_{X,Y} : \mathsf{T}X \times Y \to \mathsf{T}(X \times Y)$ is defined using the *twist map* $\mathsf{tw}_{X,Y} : X \times Y \to Y \times X$ (taking $(x, y) \in X \times Y$ to (y, x)) by

$$\mathsf{T}X \times Y \xrightarrow{\mathsf{tw}_{\mathsf{T}X,Y}} Y \times \mathsf{T}X \xrightarrow{\mathsf{st}_{Y,X}} \mathsf{T}(Y \times X) \xrightarrow{\mathsf{T}\mathsf{tw}_{Y,X}} \mathsf{T}(X \times Y)$$

▶ **Example 2.1.** As examples of commutative monads, we consider:

1. The *powerset monad* $\mathcal{P} : \mathsf{Set} \to \mathsf{Set}$, modelling nondeterministic computations: $\mathcal{P}(X) = \{U \mid U \subseteq X\}$, with unit given by singletons, multiplication given by unions, strength given by $\mathsf{st}_{X,Y}(x, V) = \{x\} \times V$ and double strength given by $\mathsf{dst}_{X,Y}(U, V) = U \times V$.

2. The *semiring monad* $\mathsf{T}_S : \mathsf{Set} \to \mathsf{Set}$, with $(S, +, 0, \bullet, 1)$ a commutative semiring, modelling weighted computations: $\mathsf{T}_S(X) = \{\varphi : X \to S \mid \mathsf{supp}(\varphi) \text{ is finite}\}$, where $\mathsf{supp}(\varphi) = \{x \in X \mid \varphi(x) \neq 0\}$ is the *support* of φ. Its unit and multiplication are given by $\eta_X(x)(y) = \begin{cases} 1 & \text{if } y = x \\ 0 & \text{otherwise} \end{cases}$ and $\mu_X(\Phi) = \sum\limits_{\varphi \in \mathsf{supp}(\Phi)} \sum\limits_{x \in \mathsf{supp}(\varphi)} \Phi(\varphi) \bullet \varphi(x)$, while its strength and double strength are given by $\mathsf{st}_{X,Y}(x, \psi)(z, y) = \begin{cases} \psi(y) & \text{if } z = x \\ 0 & \text{otherwise} \end{cases}$ and $\mathsf{dst}_{X,Y}(\varphi, \psi)(z, y) = \varphi(z) \bullet \psi(y)$. As an example we consider the *tropical semiring* $W = (\mathbb{N}^\infty, \min, \infty, +, 0)$, with the weights being thought of as costs.
3. The *sub-probability distribution monad* $\mathcal{S} : \mathsf{Set} \to \mathsf{Set}$, modelling probabilistic computations: $\mathcal{S}(X) = \{\varphi : X \to [0, 1] \mid \sum\limits_{x \in \mathsf{supp}(\varphi)} \varphi(x) \leq 1\}$[1]. Its unit, multiplication, strength and double strength are defined similarly to those of the semiring monad.

It was shown in [13, 4] that any commutative monad $\mathsf{T} : \mathsf{Set} \to \mathsf{Set}$ induces a commutative monoid structure on the set $\mathsf{T}1$, with $1 = \{*\}$ a one-element set. The monoid multiplication $\bullet : \mathsf{T}1 \times \mathsf{T}1 \to \mathsf{T}1$ is given by the composition

$$\mathsf{T}1 \times \mathsf{T}1 \xrightarrow{\mathsf{dst}_{1,1}} \mathsf{T}(1 \times 1) \xrightarrow{\mathsf{T}\pi_2} \mathsf{T}1$$

whereas the unit is given by $\eta_1(*) \in \mathsf{T}1$.

In addition to being commutative, all the monads in Example 2.1 are *partially additive* [1]. Commutative, partially additive monads $\mathsf{T} : \mathsf{Set} \to \mathsf{Set}$ were shown in loc. cit. to induce a partial commutative semiring structure on the set $\mathsf{T}1$. The resulting partial semirings serve as the domains of truth values for the logics in [2]. To interpret fixpoint formulas, these logics make use of a partial order on the set $\mathsf{T}1$, canonically induced by the partial addition operation on $\mathsf{T}1$.

In order to recall the definition of partially additive monads, we note that any monad $\mathsf{T} : \mathsf{Set} \to \mathsf{Set}$ with $\mathsf{T}\emptyset = 1$ is such that, for any X, $\mathsf{T}X$ has a *zero element* $0 \in \mathsf{T}X$, obtained as $(\mathsf{T}!_X)(*)$. This yields a *zero map* $0 : Y \to \mathsf{T}X$ for any X, Y, given by

$$Y \xrightarrow{!_Y} \mathsf{T}\emptyset \xrightarrow{T!_X} \mathsf{T}X$$

with the maps $!_Y : Y \to \mathsf{T}\emptyset$ and $!_X : \emptyset \to X$ arising by finality and initiality, respectively. Partial additivity is then defined using the following map:

$$T(X + Y) \xrightarrow{\langle \mu_X \circ \mathsf{T}p_1, \mu_Y \circ \mathsf{T}p_2 \rangle} \mathsf{T}X \times \mathsf{T}Y \tag{1}$$

where $p_1 = [\eta_X, 0] : X + Y \to \mathsf{T}X$ and $p_2 = [0, \eta_Y] : X + Y \to \mathsf{T}Y$.

▶ **Definition 2.2** ([1]). A monad $\mathsf{T} : \mathsf{Set} \to \mathsf{Set}$ is called *partially additive* if $\mathsf{T}\emptyset = 1$ and the map in (1) is a monomorphism.

If the map in (1) is an isomorphism, then T is called *additive*. Additive monads were studied in [13, 4].

[1] This definition allows for sub-probability distributions with *countable* support.

A (partially) additive monad T induces a (partial) addition operation $+$ on the set $\mathsf{T}X$, given by $\mathsf{T}[1_X, 1_X] \circ q_{X,X}$:

$$\begin{tikzcd}
\mathsf{T}X & \mathsf{T}(X+X) \arrow[l, "{\mathsf{T}[1_X,1_X]}"'] \arrow[r, shift left, "{\langle \mu_X \circ \mathsf{T}p_1, \mu_Y \circ \mathsf{T}p_2 \rangle}"] & \mathsf{T}X \times \mathsf{T}X \arrow[l, dashed, shift left, "q_{X,X}"] \arrow[ll, bend left, "+"]
\end{tikzcd}$$

where $q_{X,X} : \mathsf{T}X \times \mathsf{T}X \to \mathsf{T}(X+X)$ is the (partial) left inverse of the map $\langle \mu_X \circ \mathsf{T}p_1, \mu_Y \circ \mathsf{T}p_2 \rangle$. That is, $a + b$ is defined if and only if $(a, b) \in \mathsf{Im}(\langle \mu_X \circ \mathsf{T}p_1, \mu_Y \circ \mathsf{T}p_2 \rangle)$. Hence, when T is additive, $+$ is a total operation.

The next result relates commutative, partially additive monads to *partial commutative semirings*. The latter are given by a set S carrying a partial commutative monoid structure $(S, +, 0)$ as well as a commutative monoid structure $(S, \bullet, 1)$, with $\bullet$ distributing over $+$. Specifically, for all $s, t, u \in S$, $s \bullet 0 = 0$, and whenever $t + u$ is defined, so is $s \bullet t + s \bullet u$ and moreover $s \bullet (t + u) = s \bullet t + s \bullet u$. A similar result in [4] relates additive monads and commutative semirings.

▶ **Proposition 2.3** ([1]). *Let* T *be a commutative, (partially) additive monad. Then* $(\mathsf{T}1, 0, +, \bullet, \eta_1(*))$ *is a (partial) commutative semiring.*

▶ Remark 2.4. If, in addition to being partially additive, T is also finitary, then one can show that T is isomorphic to the *partial semiring monad* $\mathsf{T}_S : \mathsf{Set} \to \mathsf{Set}$ induced by the partial commutative semiring $S = (\mathsf{T}1, 0, +, \bullet, \eta_1(*))$. This monad is defined similarly to the semiring monad T_S of Example 2.1, except that this time only those finitely-supported $\varphi : X \to S$ for which the sum $\sum\limits_{x \in \mathsf{supp}(\varphi)} \varphi(x)$ is defined are considered in $\mathsf{T}_S X$. That this yields a monad follows as for the sub-probability distribution monad. The previous observation then follows from $\mathsf{T}\emptyset = 1 \simeq (\mathsf{T}1)^{\emptyset}$, together with the existence of (natural) isomorphisms $\mathsf{T}X \simeq \mathsf{T}(\coprod_{x \in X} 1) \simeq \mathsf{T}_S X$ for X finite and non-empty, with the latter isomorphism being a consequence of the definition of $+$ on $\mathsf{T}1$ for T partially additive – $\mathsf{T}_S X$ is the subset of $\prod_{x \in X}(\mathsf{T}1) \simeq (\mathsf{T}1)^X$ reached by a map $\mathsf{T}(\coprod_{x \in X} 1) \to \prod_{x \in X}(\mathsf{T}1)$ defined similarly to the map in (1). While not all the monads in Example 2.1 are finitary ($\mathcal{P}$ and $\mathcal{S}$ are not), their finitary versions can be phrased as partial semiring monads.

For a partially additive monad T, the partial monoid $(\mathsf{T}1, +, 0)$ can be used to define a preorder relation on $\mathsf{T}1$:

$$x \sqsubseteq y \quad \text{if and only if} \quad \text{there exists } z \in S \text{ such that } x + z = y$$

It is shown in [1] that $\sqsubseteq$ has $0 \in S$ as bottom element and is preserved by $\bullet$ in each argument.

▶ **Example 2.5.** For the partially additive monads in Example 2.1, one obtains the commutative semirings $(\{\bot, \top\}, \vee, \bot, \wedge, \top)$ when $\mathsf{T} = \mathcal{P}$, $W = (\mathbb{N}^\infty, \min, \infty, +, 0)$ when $\mathsf{T} = \mathsf{T}_W$ and the *partial* commutative semiring $([0, 1], +, 0, *, 1)$ when $\mathsf{T} = \mathcal{S}$ (with $a + b$ defined if and only if $a + b \leq 1$). The preorders associated to these (partial) semirings are all partial orders: $\leq$ on $\{\bot, \top\}$ for $\mathsf{T} = \mathcal{P}$, $\geq$ on $\mathbb{N}^\infty$ for $\mathsf{T} = \mathsf{T}_W$, and $\leq$ on $[0, 1]$ for $\mathsf{T} = \mathcal{S}$.

From this point onwards, T denotes a commutative, partially additive monad with associated partial commutative semiring $(\mathsf{T}1, 0, +, \bullet, \eta_1(*))$ and associated preorder $\sqsubseteq$. We further assume that the unit of $\bullet$ is a top element for $\sqsubseteq$, and that $\sqsubseteq$ is both an *ω-chain complete* and an *ω^{op}-chain complete* partial order, that is, any increasing (decreasing) chain has a least upper bound (greatest lower bound). These assumptions hold for all the preorders in Example 2.5.

2.2 Coalgebraic Linear Time Logics

We now recall briefly a variant of the logics proposed in [2]. The difference w.r.t. loc. cit. is the lack of the propositional constant $\top$. The presence of $\top$ in the syntax of the logics would allow one to also express properties of *partial* traces. The logics below allow the formulation of properties of *completed*, i.e. maximal traces only, as defined in [1].

The syntax of the logics is given by

$$\varphi ::= x \mid [\lambda](\varphi_1, \ldots, \varphi_{\mathsf{ar}(\lambda)}) \mid \mu x.\varphi \mid \nu x.\varphi, \qquad x \in \mathcal{V},\ \lambda \in \Lambda$$

with $\mathcal{V}$ a set of variables and Λ a set of modal operators with associated *generalised predicate liftings* $[\![\lambda]\!] : (\mathsf{T}1)^{-} \times \ldots \times (\mathsf{T}1)^{-} \Rightarrow (\mathsf{T}1)^{F-}$. Then, for a $\mathsf{T}F$-coalgebra (X, γ) and a valuation $V : \mathcal{V} \to (\mathsf{T}1)^X$ (interpreting the variables in $\mathcal{V}$ as generalised predicates over X), a formula φ is itself interpreted as a generalised predicate $[\![\varphi]\!]_\gamma^V \in (\mathsf{T}1)^X$, defined inductively on the structure of φ by

- $[\![x]\!]_\gamma^V = V(x)$,
- $[\![[\lambda](\varphi_1, \ldots, \varphi_{\mathsf{ar}(\lambda)})]\!]_\gamma^V = \gamma^*(\mathsf{ext}_{FX}([\![\lambda]\!]_X([\![\varphi_1]\!]_\gamma^V, \ldots, [\![\varphi_{\mathsf{ar}(\lambda)}]\!]_\gamma^V)))$, where the generalised predicate lifting $\mathsf{ext} : (\mathsf{T}1)^{-} \Rightarrow (\mathsf{T}1)^{\mathsf{T}-}$, called *extension lifting* in [2], takes a generalised predicate $p : X \to \mathsf{T}1$ to the generalised predicate $\mu_1 \circ \mathsf{T}p : \mathsf{T}X \to \mathsf{T}1$ (with $\mu : \mathsf{T}^2 \Rightarrow \mathsf{T}$ the monad multiplication), and where $\gamma^* : (\mathsf{T}1)^{\mathsf{T}FX} \to (\mathsf{T}1)^X$ is given by pre-composition with $\gamma : X \to \mathsf{T}FX$.
- $[\![\mu x.\varphi]\!]_\gamma^{V\setminus\{x\}}$ ($[\![\nu x.\varphi]\!]_\gamma^{V\setminus\{x\}}$) is the least (respectively greatest) fixpoint of the operator on $(\mathsf{T}1)^X$ defined by $p \longmapsto [\![\varphi]\!]_\gamma^{V[p/x]}$, where the valuation $V[p/x] : \mathcal{V} \to (\mathsf{T}1)^X$ is given by $V[p/x](x) = p$ and $V[p/x](y) = V(y)$ for $y \in \mathcal{V} \setminus \{x\}$.

The use of the extension lifting in the definition of the semantics allows the branching present in the coalgebra γ to be abstracted away in a step-wise manner. For the operator in the last clause to be order-preserving, monotonicity of both ext and the generalised predicate liftings $[\![\lambda]\!]$, with $\lambda \in \Lambda$, is required. The fact that ext is monotone follows by an argument similar to that of [1, Proposition 5.3], with the proof making use of the definition of the order $\sqsubseteq$ on $\mathsf{T}1$ in terms of the partial addition operation on $\mathsf{T}1$. Monotonicity in each argument of the generalised predicate liftings $[\![\lambda]\!]$, with $\lambda \in \Lambda$, was shown in [2] under the assumptions that F is a polynomial functor and that the $[\![\lambda]\!]$s are canonically derived from a presentation of F as a coproduct of finite products of identity functors. (Given such a presentation, each coproduct component Id^n yields a modality of arity n. The existence of least, respectively greatest fixpoints as required by the last clause then follows by [5, Theorem 8.22]. We note that this result only requires an order-preserving operator on a complete partial order. If, in addition, $\mathsf{T}1$ is a complete lattice (which is the case in all our examples), then the Knaster-Tarski fixpoint theorem (see e.g. [5, Theorem 2.35]) also applies, and provides a characterisation of least (greatest) fixpoints as least pre-fixpoints (respectively greatest post-fixpoints).

▶ **Example 2.6.** For $F = 1 + A \times \mathsf{Id} \simeq 1 + \coprod_{a \in A} \mathsf{Id}$, the modal operators arising from the structure of F are a nullary modality $*$ together with unary modalities $[a]$ with $a \in A$. The associated predicate liftings, canonically derived from the structure of F, are given by $[\![*]\!]_X \in (\mathsf{T}1)^{FX}$, $[\![*]\!]_X(\iota_1(*)) = 1$ and $[\![*]\!]_X(\iota_a(x)) = 0$ for $x \in X$, and respectively $[\![a]\!]_X : (\mathsf{T}1)^X \to (\mathsf{T}1)^{FX}$, $[\![a]\!]_X(p)(\iota_1(*)) = 0$ and $[\![a]\!]_X(p)(\iota_{a'}(x)) = \begin{cases} p(x), & \text{if } a' = a \\ 0, & \text{otherwise} \end{cases}$, for $p \in (\mathsf{T}1)^X$ and $x \in X$. Similarly, for $F = A \times \mathsf{Id} \times \mathsf{Id} \simeq \coprod_{a \in A}(\mathsf{Id} \times \mathsf{Id})$, the induced modal operators are binary modalities $[a]$ with $a \in A$, with associated predicate liftings given by $[\![a]\!]_X(p_1, p_2)(\iota_{a'}(x, y)) = \begin{cases} p_1(x) \bullet p_2(y), & \text{if } a' = a \\ 0, & \text{otherwise} \end{cases}$, for $p_1, p_2 \in (\mathsf{T}1)^X$ and $x, y \in X$.

(Similar use of the monoid multiplication $\bullet$ is made for any generalised predicate lifting of arity ≥ 2 derived canonically from a polynomial endofunctor F.) Irrespective of the choice of F, when $\mathsf{T} = \mathcal{P}$, a formula φ of the resulting logic holds in a state of a $\mathsf{T}F$-coalgebra if that state admits a maximal trace (element of the final F-coalgebra) satisfying φ. Also, when $\mathsf{T} = \mathcal{S}$ ($T = \mathsf{T}_W$), $[\![\varphi]\!]_\gamma : X \to \mathsf{T}1$ measures the likelihood (respectively minimal cost) with which a maximal trace satisfying φ is exhibited by states of a $\mathsf{T}F$-coalgebra (X, γ).

3 Coalgebraic Linear Time Logics via Dual Adjunctions

This section rephrases the generalised predicate lifting approach to defining the semantics of coalgebraic linear time logics in terms of dual adjunctions.

For an endofunctor $F : \mathsf{Set} \to \mathsf{Set}$, the dual adjunction approach to defining a logic for F-coalgebras involves a contravariant adjunction $\begin{tikzcd} \mathcal{A} \arrow[r, bend left, "S"] \arrow[r, phantom, "\perp"] & \mathsf{Set}^{\mathsf{op}} \arrow[l, bend left, "P"] \end{tikzcd}$, a functor $L : \mathcal{A} \to \mathcal{A}$ and a natural transformation $\delta : LP \Rightarrow PF$. These yield a logic for F-coalgebras with syntax given by the initial L-algebra $(\mathcal{L}, \alpha)$ and with semantics $[\![\_]\!]_\gamma : \mathcal{L} \to PX$, for an F-coalgebra (X, γ), defined as the unique L-algebra homomorphism from α to $P\gamma \circ \delta_X$:

$$\begin{tikzcd}
L(\mathcal{L}) \arrow[r, "L[\![\_]\!]_\gamma"] \arrow[dd, "\alpha"'] & LPX \arrow[d, "\delta_X"] \\
 & PFX \arrow[d, "P\gamma"] \\
\mathcal{L} \arrow[r, "[\![\_]\!]_\gamma"] & PX
\end{tikzcd}$$

To match the syntax and semantics of the logics in [2], we consider the dual adjunction $\begin{tikzcd} \mathsf{Set} \arrow[r, bend left, "S"] \arrow[r, phantom, "\perp"] & \mathsf{Set}^{\mathsf{op}} \arrow[l, bend left, "P"] \end{tikzcd}$ with $S = P = (\mathsf{T}1)^-$. Following previous work on the modular construction of coalgebraic logics [3] (see also [12] for a similar approach to defining *forgetful logics*), we take a modular approach to defining a natural transformation $\delta : LP \Rightarrow P\mathsf{T}F$ that captures the above use of the extension lifting ext and of the generalised predicate liftings $[\![\lambda]\!]$ derived from the structure of F. The ingredients required for this are:

- an endofunctor $L : \mathsf{Set} \to \mathsf{Set}$ specifying the syntax of a logic for F-coalgebras, together with a natural transformation $\delta : LP \Rightarrow PF$, providing a one-step semantics for this logic,
- a natural transformation $\sigma : \mathsf{Id}P \Rightarrow P\mathsf{T}$, providing a one-step semantics for a logic for T-coalgebras.

The use of the identity functor to define a syntax for T reflects the fact that, in the logics of [2], the branching modality is hidden from the syntax. Then, to capture the use of the extension predicate lifting ext in the definition of the semantics, the components of $\sigma : P \Rightarrow P\mathsf{T}$ must be given by

$$X \xrightarrow{p} \mathsf{T}1 \quad \overset{\sigma}{\longmapsto} \quad \mathsf{T}X \xrightarrow{\mathsf{T}p} \mathsf{T}^2 1 \xrightarrow{\mu_1} \mathsf{T}1 \tag{2}$$

Following [2], other choices for a modality that abstracts away branching have been considered: both [6] and [12] propose using an arbitrary T-algebra structure $\tau : \mathsf{T}^2 1 \to \mathsf{T}1$ instead of μ_1 in the definition of σ. While most of the results in this paper concern the canonical choice of σ, we also explore the more general σs arising from a choice of τ as above. For this, we need the following lemma, where we write $\mathsf{Alg}(\mathsf{T})$ for the category of Eilenberg-Moore algebras of the monad T.

▶ **Lemma 3.1.** *For any* $(\mathsf{T}1, \tau)$ *in* $\mathsf{Alg}(\mathsf{T})$*, with induced* $\sigma : P \Rightarrow P\mathsf{T}$*, we have* $\sigma_{\mathsf{T}} \circ \sigma = P\mu \circ \sigma$*.*

Proof. $\sigma_{\mathsf{T}X} \circ \sigma_X$ maps a predicate $p : X \to P1$ to the predicate $\tau \circ \mathsf{T}\tau \circ \mathsf{T}^2 p$, whereas $P\mu_X \circ \sigma_X$ maps p to $\tau \circ \mathsf{T}p \circ \mu_X$. The conclusion now follows from the commutativity of

$$\begin{tikzcd}
\mathsf{T}^2 X \arrow[r, "\mathsf{T}^2 p"] \arrow[d, "\mu_X"'] & T^3 1 \arrow[r, "\mathsf{T}\tau"] \arrow[d, "\mu_{\mathsf{T}1}"'] & \mathsf{T}^2 1 \arrow[d, "\tau"] \\
\mathsf{T}X \arrow[r, "\mathsf{T}p"] & \mathsf{T}^2 1 \arrow[r, "\tau"] & \mathsf{T}1
\end{tikzcd}$$

where the left and right squares follow by naturality of μ and from $\tau \in \mathsf{Alg}(\mathsf{T})$, respectively. ◀

We now return to the endofunctor F and discuss the canonical choice for the corresponding L and $\delta : LP \Rightarrow PF$. As in [2], we assume that F is a polynomial endofunctor, and hence naturally isomorphic to a coproduct of finite (including empty) products of identity functors. Presenting F in this way canonically determines a set of modal operators (as already sketched in Example 2.6).

▶ **Definition 3.2.** Let $L ::= F = \coprod_{\lambda \in \Lambda} X^{\mathsf{ar}(\lambda)}$, with Λ a set of modal operators with specified arities. Also, let $\delta : LP \Rightarrow PF$ be given by

$$\begin{tikzcd}
(PX)^{\mathsf{ar}(\lambda)} \arrow[r, "\bullet_X \circ (P\pi_1 \times \ldots \times P\pi_{\mathsf{ar}(\lambda)})"] \arrow[d, "\iota_\lambda"'] & P(X^{\mathsf{ar}(\lambda)}) \arrow[r, "e_\lambda"] & P(\coprod_{\lambda \in \Lambda} X^{\mathsf{ar}(\lambda)}) \\
\coprod_{\lambda \in \Lambda} (PX)^{\mathsf{ar}(\lambda)} \arrow[urr, dashed, "\delta_X"'] & &
\end{tikzcd}$$

where in the above $\bullet_Y : (PY)^n \to PY$ is given by the transpose of the map

$$((\mathsf{T}1)^Y \times \ldots \times (\mathsf{T}1)^Y) \times Y \longrightarrow \mathsf{T}1\ , \qquad (p_1, \ldots, p_n, y) \mapsto p_1(y) \bullet \ldots \bullet p_n(y)$$

with $\bullet : \mathsf{T}1 \times \mathsf{T}1 \to \mathsf{T}1$ the multiplication operation on $\mathsf{T}1$ (extended to an n-ary operation), and with $e_\lambda : P(X^{\mathsf{ar}(\lambda)}) \to P(\coprod_{\lambda \in \Lambda} X^{\mathsf{ar}(\lambda)})$ being given by

$$X^{\mathsf{ar}(\lambda)} \xrightarrow{p} \mathsf{T}1 \qquad \overset{e_\lambda}{\longmapsto} \qquad \coprod_{\lambda \in \Lambda} X^{\mathsf{ar}(\lambda)} \xrightarrow{[0, \ldots, p, \ldots, 0]} \mathsf{T}1$$

The particular choice of L and δ in Definition 3.2 corresponds to a syntax with modal operators $\lambda \in \Lambda$, with associated generalised predicate liftings given by $e_\lambda \circ \bullet_X \circ (P\pi_1 \times \ldots \times P\pi_{\mathsf{ar}(\lambda)})$. That is, for $\lambda \in \Lambda$, the associated predicate lifting takes $(p_1, \ldots, p_{\mathsf{ar}(\lambda)})$ with $p_i : X \to \mathsf{T}1$ to the generalised predicate taking $x \in X$ to $e_\lambda(p_1(x) \bullet \ldots \bullet p_{\mathsf{ar}(\lambda)}(x)) \in \mathsf{T}1$. In particular, the generalised predicate liftings described in Example 2.6 are of this form. Moreover, as explained in Section 2.2, these generalised predicate liftings are monotone.

Having fixed $\delta : LP \Rightarrow PF$ and $\sigma : P \Rightarrow P\mathsf{T}$, a logic $\mathcal{L}$ for $\mathsf{T}F$-coalgebras arises from the one-step semantics specified by the natural transformation $\sigma_F \circ \delta$:

$$\begin{tikzcd}
LP \arrow[r, Rightarrow, "\delta"] & PF \arrow[r, Rightarrow, "\sigma_F"] & P\mathsf{T}F
\end{tikzcd}$$

That is, for a $\mathsf{T}F$-coalgebra (X, γ), the map $[\![\_]\!]_\gamma : \mathcal{L} \to PX$ arises as the unique L-algebra homomorphism from the initial L-algebra $(\mathcal{L}, \alpha)$ to $(PX, P\gamma \circ \sigma_{FX} \circ \delta_X)$. More generally, for a valuation $V : \mathcal{V} \to PX$, $[\![\_]\!]^V_\gamma : \mathcal{L}^{\mathcal{V}} \to PX$ is defined as the unique L-algebra homomorphism from the free L-algebra $(\mathcal{L}^{\mathcal{V}}, \alpha^{\mathcal{V}})$ over $\mathcal{V}$ to $(PX, P\gamma \circ \sigma_{FX} \circ \delta_X)$ which extends V:

$$\begin{tikzcd}
L(\mathcal{L}^{\mathcal{V}}) \arrow[r, "L({[\![\_]\!]}^V_\gamma)"] \arrow[dd, "\alpha^{\mathcal{V}}"'] & LPX \arrow[d, "\sigma_{FX} \circ \delta_X"] \\
& P\mathsf{T}FX \arrow[d, "P\gamma"] \\
\mathcal{L}^{\mathcal{V}} \arrow[r, "{[\![\_]\!]}^V_\gamma"] & PX
\end{tikzcd}$$

Extending the logic $\mathcal{L}$ with fixpoint formulas can now be done as before. We write $\mathcal{L}_\mu$ for the resulting logic, and conclude the section by providing an alternative definition of the semantics of fixpoint formulas. This exploits the existence of a *coalgebraic structure* on the modal fragment of the logic, and will later smoothly generalise to the case where the logics carry $\mathsf{Alg}(\mathsf{T})$-structure. To this end, we let $\varphi \in \mathcal{L}^{\{x\}+\mathcal{V}}$ and consider the $\mathcal{V} + L(\_)$-coalgebra $(\mathcal{L}^{\{x\}+\mathcal{V}}, \beta_\varphi)$, with $\beta_\varphi : \mathcal{L}^{\{x\}+\mathcal{V}} \to \mathcal{V} + L\mathcal{L}^{\{x\}+\mathcal{V}}$ the unique L-algebra homomorphism satisfying $\beta_\varphi(v) = v$ for $v \in \mathcal{V}$ and $\beta_\varphi(x) = \varphi$. (Note that the set $\mathcal{V} + L\mathcal{L}^{\{x\}+\mathcal{V}}$ inherits L-algebra structure from $(\mathcal{L}^{\{x\}+\mathcal{V}}, \alpha^{\{x\}+\mathcal{V}})$, as $\mathcal{L}^{\{x\}+\mathcal{V}} \simeq \{x\} + \mathcal{V} + L\mathcal{L}^{\{x\}+\mathcal{V}}$.)

▶ **Lemma 3.3.** *Let (X, γ) be a $\mathsf{T}F$-coalgebra, let $V : \mathcal{V} \to PX$ be a valuation, and let $\varphi \in \mathcal{L}^{\{x\}+\mathcal{V}}$. Consider the operator $\mathsf{O} : [\mathcal{L}^{\{x\}+\mathcal{V}}, PX] \to [\mathcal{L}^{\{x\}+\mathcal{V}}, PX]$ defined by $f \mapsto [V, P\gamma] \circ (\mathsf{id} + (\sigma_{FX} \circ \delta_X)) \circ (\mathsf{id} + Lf) \circ \beta_\varphi$:*

$$\begin{tikzcd}
\mathcal{V} + L\mathcal{L}^{\{x\}+\mathcal{V}} \arrow[r, "\mathsf{id}+Lf"] & \mathcal{V} + LPX \arrow[d, "\mathsf{id}+(\sigma_{FX}\circ\delta_X)"] \\
 & \mathcal{V} + P\mathsf{T}FX \arrow[d, "{[V,P\gamma]}"] \\
\mathcal{L}^{\{x\}+\mathcal{V}} \arrow[uu, "\beta_\varphi"] \arrow[r, "f"] & PX
\end{tikzcd}$$

Then $[\![\mu x.\varphi]\!]^V_\gamma$ ($[\![\nu x.\varphi]\!]^V_\gamma$) is given by $f_0(x)$, with $f_0 : \mathcal{L}^{\{x\}+\mathcal{V}} \to PX$ the least (resp. greatest) fixpoint of O.

Each application of the operator O above computes a new approximation of the semantics of formulas in $\mathcal{L}^{\{x\}+\mathcal{V}}$, obtained by replacing occurrences of the variable x by φ, and using the previous approximation for the semantics of φ. We note that, by definition, $\mathsf{O}(f)$ extends $V : \mathcal{V} \to PX$, and therefore so does f_0.

▶ Remark 3.4. In practice, one only needs the set of subformulas of $\varphi[\varphi/x]$, not the entire $\mathcal{L}^{\{x\}+\mathcal{V}}$, to define $[\![\mu x.\varphi]\!]^V_\gamma$ and $[\![\nu x.\varphi]\!]^V_\gamma$. This set inherits a $\mathcal{V} + L(\_)$-coalgebra structure from β_φ.

▶ Remark 3.5. Transporting the previous diagram via the dual adjunction to the category of spaces, we obtain an operator on $[X, S\mathcal{L}^{\{x\}+\mathcal{V}}]$:

$$\begin{tikzcd}
S\mathcal{V} \times \mathsf{T}FS\mathcal{L}^{\{x\}+\mathcal{V}} \arrow[d, swap, "\mathsf{id}+(\sigma^\flat_{L\mathcal{L}\{x\}+\mathcal{V}}\circ\mathsf{T}\delta^\flat_{\mathcal{L}\{x\}+\mathcal{V}})"] & S\mathcal{V} \times \mathsf{T}FX \arrow[l, swap, "\mathsf{id}\times\mathsf{T}Ff^\flat"] \\
S\mathcal{V} \times SL\mathcal{L}^{\{x\}+\mathcal{V}} \arrow[d, swap, "(\beta_\varphi)^\flat"] & \\
S\mathcal{L}^{\{x\}+\mathcal{V}} & X \arrow[l, "f^\flat"] \arrow[uu, swap, "{[V^\flat,\gamma]}"]
\end{tikzcd}$$

where $\delta^\flat : FS \Rightarrow SL$ and $\sigma^\flat : \mathsf{T}S \Rightarrow S$ are the *mates* of δ and σ, respectively (see e.g. [12] for a definition). Since taking least/greatest fixpoints in both $[\mathcal{L}^{\{x\}+\mathcal{V}}, PX]$ and $[X, S\mathcal{L}^{\{x\}+\mathcal{V}}]$ amounts to taking least/greatest fixpoints of operators on *generalised relations* on $X \times \mathcal{L}^{\{x\}+\mathcal{V}}$ (see [1, 2] for a treatment of generalised relations induced by T), the semantics of $\mathcal{L}_\mu$ can alternatively be defined in the category of spaces.

4 Enhanced Coalgebraic Linear Time Logics

$\mathcal{L}^{\mathcal{V}}$ and $\mathcal{L}_\mu$ only contain modal operators, not also propositional ones. We now show how to canonically add propositional operators to $\mathcal{L}^{\mathcal{V}}$ and $\mathcal{L}_\mu$, by lifting these logics to $\mathsf{Alg}(\mathsf{T})$.

It follows e.g. from [9, Exercise 5.4.11] that for T a strong monad and $(A,\alpha)\in\mathsf{Alg}(\mathsf{T})$, the dual adjunction $\begin{tikzcd}\mathsf{Set} \arrow[r, bend left, "S"] \arrow[r, phantom, "\perp"] & \mathsf{Set}^{\mathsf{op}} \arrow[l, bend left, "P"]\end{tikzcd}$ with $S=P=A^{-}$ lifts to $\begin{tikzcd}\mathsf{Alg}(\mathsf{T}) \arrow[r, bend left, "\tilde{S}"] \arrow[r, phantom, "\perp"] & \mathsf{Set}^{\mathsf{op}} \arrow[l, bend left, "\tilde{P}"]\end{tikzcd}$, with $\tilde{S}=(A,\alpha)^{-}$ and $\tilde{P}=A^{-}$, where the T-algebra required in the definition of $\tilde{P}X$ is the transpose of $\mathsf{T}(A^X)\times X \xrightarrow{\mathsf{st}'_{A^X,X}} \mathsf{T}(A^X\times X) \xrightarrow{\mathsf{Teval}} \mathsf{T}A \xrightarrow{\alpha} A$. We then have $S=\tilde{S}\mathsf{Free}$ and $P=U\tilde{P}$, where $\mathsf{Free}:\mathsf{Set}\to\mathsf{Alg}(\mathsf{T})$ takes X to $(\mathsf{T}X,\mu_X)$ and $U:\mathsf{Alg}(\mathsf{T})\to\mathsf{Set}$ takes (B,β) to B:

$$\begin{tikzcd}
\mathsf{Alg}(\mathsf{T}) \arrow[d, shift left, "U"] \arrow[rd, bend left, "\tilde{S}"] & \\
\mathsf{Set} \arrow[u, shift left, "\mathsf{Free}"] \arrow[r, bend left, "S"] & \mathsf{Set}^{\mathsf{op}} \arrow[lu, bend left, "\tilde{P}"] \arrow[l, bend left, "P"]
\end{tikzcd}$$

As before, our choice of (A,α) will be either $(\mathsf{T}1,\mu_1)$ or an arbitrary $(\mathsf{T}1,\tau)\in\mathsf{Alg}(\mathsf{T})$. Irrespective of this, we can lift the functor $L:\mathsf{Set}\to\mathsf{Set}$ from Section 3 to $\tilde{L}:\mathsf{Alg}(\mathsf{T})\to\mathsf{Alg}(\mathsf{T})$ by taking $\tilde{L}=\mathsf{Free}LU$. Then, the one-step semantics $\delta: LU\tilde{P}=LP\Rightarrow PF=U\tilde{P}F$ lifts to $\tilde{\delta}::=\delta^{\sharp}:\tilde{L}\tilde{P}=\mathsf{Free}LU\tilde{P}\Rightarrow\tilde{P}F$.

There is no need for a similar lifting of the identity functor on Set with associated one-step semantics σ to $\mathsf{Alg}(\mathsf{T})$, since the components of σ are already $\mathsf{Alg}(\mathsf{T})$-homomorphisms – this follows from an equivalent definition of $\sigma_X:(\mathsf{T}1)^X\to(\mathsf{T}1)^{\mathsf{T}X}$ as the transpose of the unique extension of $\mathsf{eval}:(\mathsf{T}1)^X\times X\to\mathsf{T}1$ to a 2-*linear map*[2] $(\mathsf{T}1)^X\times\mathsf{T}X\to\mathsf{T}1$, as shown in [14, Proposition 4.1]. We therefore simply write $\tilde{\sigma}:\tilde{P}\Rightarrow\tilde{P}\mathsf{T}$ for the natural transformation whose components are given by those of $\sigma:P\Rightarrow PT$.

This yields new logics $\tilde{\mathcal{L}}$ and $\tilde{\mathcal{L}}^{\mathsf{Free}(\mathcal{V})}$ carrying T-algebra structure, and associated semantics $[\![\_]\!]_\gamma:\tilde{\mathcal{L}}\to\tilde{P}X$ and $[\![\_]\!]_\gamma^{V^\sharp}:\tilde{\mathcal{L}}^{\mathsf{Free}(\mathcal{V})}\to\tilde{P}X$, for each $\mathsf{T}F$-coalgebra (X,γ) and valuation $V:\mathcal{V}\to PX$ (extending to a T-algebra homomorphism $V^\sharp:\mathsf{Free}(\mathcal{V})\to\tilde{P}X$). The syntax of these logics contains propositional operators arising from the T-algebra structure (see Example 4.4 at the end of this section for operators induced by specific monads) and modal operators $\lambda\in\Lambda$. To add fixpoints to these logics, we can now proceed as in Lemma 3.3.

▶ **Definition 4.1.** Let (X,γ) be a $\mathsf{T}F$-coalgebra, let $V:\mathcal{V}\to PX$ be a valuation, and let $\varphi\in\tilde{\mathcal{L}}^{\mathsf{Free}(\{x\}+\mathcal{V})}$. Consider the operator $\tilde{\mathsf{O}}:[\tilde{\mathcal{L}}^{\mathsf{Free}(\{x\}+\mathcal{V})},\tilde{P}X]\to[\tilde{\mathcal{L}}^{\mathsf{Free}(\{x\}+\mathcal{V})},\tilde{P}X]$ defined by $\tilde{f}\ \mapsto\ [V^\sharp,P\gamma]\circ(\mathsf{id}+(\tilde{\sigma}_{FX}\circ\tilde{\delta}_X))\circ(\mathsf{id}+\tilde{L}\tilde{f})\circ\tilde{\beta}_\varphi$:

$$\begin{tikzcd}
\mathsf{Free}(\mathcal{V})+\tilde{L}\tilde{\mathcal{L}}^{\mathsf{Free}(\{x\}+\mathcal{V}} \arrow[r, "\mathsf{id}+\tilde{L}\tilde{f}"] & \mathsf{Free}(\mathcal{V})+\tilde{L}\tilde{P}X \arrow[d, "\mathsf{id}+(\tilde{\sigma}_{FX}\circ\tilde{\delta}_X)"] \\
 & \mathsf{Free}(\mathcal{V})+\tilde{P}\mathsf{T}FX \arrow[d, "{[V^\sharp,P\gamma]}"] \\
\tilde{\mathcal{L}}^{\mathsf{Free}(\{x\}+\mathcal{V})} \arrow[uu, "\tilde{\beta}_\varphi"] \arrow[r, "\tilde{f}"] & \tilde{P}X
\end{tikzcd}$$

Then $[\![\mu x.\varphi]\!]_\gamma^{V^\sharp}$ (respectively $[\![\nu x.\varphi]\!]_\gamma^{V^\sharp}$) is defined as $\tilde{f}_0(x)$, where $\tilde{f}_0:\tilde{\mathcal{L}}^{\mathsf{Free}(\{x\}+\mathcal{V})}\to\tilde{P}X$ is the least (respectively greatest) fixpoint of $\tilde{\mathsf{O}}$. We write $\tilde{\mathcal{L}}_\mu$ for the resulting fixpoint logic.

[2] A 2-linear map is required to preserve the T-algebra structure in the second argument, where the assumed T-algebra structures on $\mathsf{T}X$ and $\mathsf{T}1$ are the free ones (μ_X and μ_1 respectively).

Now observe that for a formula $\varphi \in \mathcal{L}^{\mathcal{V}}$, one can consider the semantics of its translation to $\tilde{\mathcal{L}}^{\mathsf{Free}(\mathcal{V})}$, in addition to the semantics $[\![\varphi]\!]^{\mathcal{V}}_{\gamma}$. As expected, the two agree:

▶ **Proposition 4.2.** *Let* $!_{\mathcal{V}} : (\mathcal{L}^{\mathcal{V}}, \alpha^{\mathcal{V}}) \to (U\tilde{\mathcal{L}}^{\mathsf{Free}(\mathcal{V})}, U\tilde{\alpha}^{\mathsf{Free}(\mathcal{V})} \circ \eta_{LU\tilde{\mathcal{L}}^{\mathsf{Free}(\mathcal{V})}})$ *be the unique* L*-algebra morphism arising by freeness of* $(\mathcal{L}^{\mathcal{V}}, \alpha^{\mathcal{V}})$*. Then* $[\![\varphi]\!]^{V}_{\gamma} = U[\![!_{\mathcal{V}}(\varphi)]\!]^{V^\sharp}_{\gamma}$ *for* $\varphi \in \mathcal{L}^{\mathcal{V}}$*.*

Proof (sketch). The conclusion follows by freeness of $(\mathcal{L}^{\mathcal{V}}, \alpha^{\mathcal{V}})$ from the commutativity of

$$\begin{tikzcd}
L\mathcal{L}^{\mathcal{V}} \arrow[r, "L!_{\mathcal{V}}"] \arrow[dddd, "\alpha^{\mathcal{V}}"'] & LU\tilde{\mathcal{L}}^{\mathsf{Free}(\mathcal{V})} \arrow[r, "LU[\![\_]\!]^{V^\sharp}_{\gamma}"] \arrow[d, "\eta_{LU\tilde{\mathcal{L}}^{\mathsf{Free}(\mathcal{V})}}"'] & LU\tilde{P}X \arrow[r, equal] \arrow[d, "\eta_{LU\tilde{P}X}"] & LPX \arrow[dd, "\delta_X"] \\
 & U\mathsf{Free}LU\tilde{\mathcal{L}}^{\mathsf{Free}(\mathcal{V})} \arrow[r, "U\mathsf{Free}LU[\![\_]\!]^{V^\sharp}_{\gamma}"] \arrow[ddd, "U\tilde{\alpha}^{\mathsf{Free}(\mathcal{V})}"'] & U\mathsf{Free}LU\tilde{P}X \arrow[d, "U\delta^\sharp_X"] & \\
 & & U\tilde{P}FX \arrow[r, equal] \arrow[d, "U\tilde{\sigma}_{FX}"] & PFX \arrow[d, "\sigma_{FX}"] \\
 & & U\tilde{P}\mathsf{T}FX \arrow[r, equal] \arrow[d, "U\tilde{P}\gamma"] & P\mathsf{T}FX \arrow[d, "P\gamma"] \\
\mathcal{L}^{\mathcal{V}} \arrow[r, "!_{\mathcal{V}}"] & U\tilde{\mathcal{L}}^{\mathsf{Free}(\mathcal{V})} \arrow[r, "U[\![\_]\!]^{V^\sharp}_{\gamma}"] & U\tilde{P}X \arrow[r, equal] & PX
\end{tikzcd}$$

$$\begin{tikzcd}
\mathcal{V} \arrow[r, "\eta_{\mathcal{V}}"] \arrow[rr, bend right, "V"'] & U\mathsf{Free}(\mathcal{V}) \arrow[r, "UV^\sharp"] & PX
\end{tikzcd}$$

◀

Finally, Proposition 4.2 extends to formulas in $\mathcal{L}_\mu$.

▶ **Proposition 4.3.** *Let* $V : \mathcal{V} \to PX$*, let* $f_0 : \mathcal{L}^{\{x\}+\mathcal{V}} \to PX$ *be the least (greatest) fixpoint of the operator* O *in Lemma 3.3, and let* $\tilde{f}_0 : \tilde{\mathcal{L}}^{\mathsf{Free}(\{x\}+\mathcal{V})} \to \tilde{P}X$ *be the least (resp. greatest) fixpoint of the operator* $\tilde{\mathsf{O}}$ *in Definition 4.1. Then,* $\tilde{f}_0 \circ !_{\{x\}+\mathcal{V}} = f_0$*.*

Proof (sketch). The conclusion follows from the fact that if $\tilde{f}$ is a least (greatest) fixpoint of $\tilde{\mathsf{O}}$, then $U\tilde{f} \circ !_{\{x\}+\mathcal{V}}$ is a least (respectively greatest) fixpoint of O. This, in turn, follows from the commutativity of the left, top and right trapezoids in the following diagram

$$\begin{tikzcd}
\mathcal{V} + L\mathcal{L}^{\{x\}+\mathcal{V}} \arrow[rrr, "\mathsf{id}+Lf"] \arrow[dr, "{[U\iota_1, U\iota_2] \circ (\iota + L!_{\{x\}+\mathcal{V}})}"] & & & \mathcal{V} + LPX \arrow[dl, "{[U\iota_1, U\iota_2] \circ (\iota + \eta_{LPX})}"'] \arrow[dd, "\mathsf{id}+(\sigma_{FX} \circ \delta_X)"] \\
 & U(\mathsf{Free}(\mathcal{V}) + \tilde{L}\tilde{\mathcal{L}}^{\mathsf{Free}(\{x\}+\mathcal{V})}) \arrow[r, "U(\mathsf{id}+\tilde{L}\tilde{f})"] & U(\mathsf{Free}(\mathcal{V}) + \tilde{L}\tilde{P}X) \arrow[d, "U(\mathsf{id}+(\tilde{\sigma}_{FX} \circ \tilde{\delta}_X))"] & \\
 & & U(\mathsf{Free}(\mathcal{V}) + \tilde{P}\mathsf{T}FX) \arrow[d, "{U[V^\sharp, P\gamma]}"] & \mathcal{V} + P\mathsf{T}FX \arrow[dd, "{[V, P\gamma]}"] \\
 & U\tilde{\mathcal{L}}^{\mathsf{Free}(\{x\}+\mathcal{V})} \arrow[r, "U\tilde{f}"] \arrow[uu, "U\tilde{\beta}_\varphi"] & U\tilde{P}X \arrow[dr, equal] & \\
\mathcal{L}^{\{x\}+\mathcal{V}} \arrow[uuuu, "\beta_\varphi"] \arrow[ur, "!_{\{x\}+\mathcal{V}}"] \arrow[rrr, "f"] & & & PX
\end{tikzcd}$$

which is equivalent to the statement that for $\varphi \in \mathcal{L}^{\{x\}+\mathcal{V}}$, the additional structure in $\tilde{\mathcal{L}}^{\mathsf{Free}(\{x\}+\mathcal{V})}$ is not needed when defining $[\![\mu X.!_{\{x\}+\mathcal{V}}(\varphi)]\!]^{V^\sharp}_{\gamma}$ and $[\![\nu X.!_{\{x\}+\mathcal{V}}(\varphi)]\!]^{V^\sharp}_{\gamma}$. ◀

▶ **Example 4.4.**

1. For $\mathsf{T} = \mathcal{P}$, $\mathsf{Alg}(\mathsf{T})$ is isomorphic to the category of join semi-lattices, and the enhanced logic contains arbitrary disjunctions. With this, one can encode a "next" modality by letting $\bigcirc\varphi ::= \bigvee_{\lambda\in\Lambda}[\lambda](\varphi,\ldots,\varphi)$. This modality turns out to be very useful, for example, the formula $\nu x.\bigcirc x$ is true in a state of a $\mathcal{P}\circ F$-coalgebra if there exists a maximal trace from that state. For $F = A \times \mathsf{Id}$ and $\bigcirc$ as above, the formula $[a](\nu x.\bigcirc x)$ is true in a state if there exists a maximal (hence infinite) trace from that state that starts with an a. (Recall that our logics do not contain a propositional constant $\top$, and therefore partial traces cannot be formalised without using fixpoint operators.) For $F = A \times \mathsf{Id}$ and $[\overline{a}]\varphi ::= \bigvee_{b\in A\setminus\{a\}}[b]\varphi$ for $a \in A$, the formula $\nu x.\mu y.([a]x \vee [\overline{a}]y)$ is true in a state if there exists a maximal trace from that state containing an infinite number of as. Finally, for $F = 1 + A \times \mathsf{Id}$ and $[A]\varphi ::= \bigvee_{a\in A}[a]\varphi$, the formula $\mu x.(* \vee [A]x)$ holds in a state if there exists a *finite* maximal trace from that state.
2. For $\mathsf{T} = \mathcal{S}$, $\mathsf{Alg}(\mathsf{T})$ is isomorphic to the category of positive convex algebras, and the enhanced logic contains sub-convex combinations of formulas. With this, one can encode properties where preference is given to one observable linear time behaviour over another. For $F = 1 + A \times \mathsf{Id}$ and $[A]$ as above, the formula $\mu x.(\frac{1}{2} * + \frac{1}{4}[A]x)$ measures the likelihood of termination, in such a way that the smaller the number of steps required for termination, the higher the value associated to the formula by the semantics.
3. For $\mathsf{T} = \mathsf{T}_S$ with $S = (S, +, 0, \bullet, 1)$ a commutative semiring, $\mathsf{Alg}(\mathsf{T})$ is isomorphic to the category of modules over S, and the enhanced logic contains finite linear combinations of formulas. As in the previous case, the resulting logic supports weighted choices.

5 Path-based Semantics for Coalgebraic Linear Time Logics

This section provides alternative path-based semantics for what we call the *uniform fragments* of the logics $\mathcal{L}^{\mathcal{V}}$ and $\tilde{\mathcal{L}}^{\mathsf{Free}(\mathcal{V})}$, and proves their equivalence to the already-defined step-wise semantics. The main results (Theorems 5.13 and 5.24) assume canonical choices for both the branching and the linear time modalities, but generalisations to non-canonical choices (subject to additional requirements) are also discussed.

▶ **Definition 5.1.** The *uniform fragment* $u\mathcal{L}^{\mathcal{V}}$ of the logic $\mathcal{L}^{\mathcal{V}}$ is given by $\bigcup\limits_{n\in\omega} \mathcal{L}^{\mathcal{V}}_n$, with $\mathcal{L}^{\mathcal{V}}_n = L^n\mathcal{V}$ consisting of formulas of rank n, for $n \in \omega$.

The uniform fragment $u\tilde{\mathcal{L}}^{\mathsf{Free}(\mathcal{V})}$ of $\tilde{\mathcal{L}}^{\mathsf{Free}(\mathcal{V})}$ is defined similarly, namely by $u\tilde{\mathcal{L}}^{\mathsf{Free}(\mathcal{V})} ::= \bigcup\limits_{n\in\omega} \tilde{L}^n\mathsf{Free}(\mathcal{V})$.

A more concrete description of the set $\mathcal{L}^{\mathcal{V}}_n$ is as the set of formulas with nesting depth of modal operators at most n, and with each occurrence of a variable being in the scope of exactly n modal operators.

▶ **Example 5.2.** For $L : \mathsf{Set} \to \mathsf{Set}$ of the form $LX = \coprod_{\lambda\in\Lambda} X^{\mathsf{ar}(\lambda)}$, and for $\lambda_i \in \Lambda$ a modality of arity i, with $i \in \{0, 1, 2\}$, $[\lambda_2]([\lambda_1]X, [\lambda_0])$, $[\lambda_1]X \vee [\lambda_1]Y \vee [\lambda_0]$ and $[\lambda_1][\lambda_1]X \vee [\lambda_0]$ are uniform modal formulas, whereas $[\lambda_2]([\lambda_1]X, [\lambda_1][\lambda_0])$ and $[\lambda_1]X \vee [\lambda_1][\lambda_1]Y$ are not (where we assume $\mathsf{T} = \mathcal{P}$, and therefore $\vee$ is a propositional operator of the enhanced logic).

▶ Remark 5.3. When $\mathcal{V} = \emptyset$, $u\mathcal{L}^{\mathcal{V}}$ coincides with the full logic $\mathcal{L}^{\mathcal{V}}$. However, when $\mathcal{V} \neq \emptyset$, the inclusion $u\mathcal{L}^{\mathcal{V}} \subseteq \mathcal{L}^{\mathcal{V}}$ is strict. All the example formulas in this paper (e.g. all the modal formulas used to define the fixpoint formulas in Example 4.4) are uniform ones. Moreover, most modal formulas used in practice to define fixpoint formulas appear to belong to the uniform fragment.

5.1 Path-based Semantics for $u\mathcal{L}^{\mathcal{V}}$

For each polynomial endofunctor F and commutative monad T, one can define a canonical distributive law of T over F as shown below. This can be used to give a path-based semantics for the uniform fragment of the logic $\mathcal{L}_{\mathcal{V}}$, by delaying the use of the natural transformation σ when defining the interpretation of formulas in $u\mathcal{L}^{\mathcal{V}}$ for as long as possible.

▶ **Definition 5.4.** For $F = \coprod_{\lambda \in \Lambda} X^{\mathsf{ar}(\lambda)}$ the *canonical distributive law* $\lambda : F\mathsf{T} \Rightarrow \mathsf{T}F$ is given by

$$\begin{tikzcd}
(\mathsf{T}X)^{\mathsf{ar}(\lambda)} \arrow[r, "\mathsf{dst}_{\mathsf{ar}(\lambda)}"] \arrow[d, "\iota_\lambda"'] & \mathsf{T}(X^{\mathsf{ar}(\lambda)}) \arrow[r, "\mathsf{T}\iota_\lambda"] & \mathsf{T}(\coprod_{\lambda \in \Lambda} X^{\mathsf{ar}(\lambda)}) = \mathsf{T}FX \\
F\mathsf{T}X = \coprod_{\lambda \in \Lambda} (\mathsf{T}X)^{\mathsf{ar}(\lambda)} \arrow[urr, dashed, "\lambda_X"'] & &
\end{tikzcd}$$

where $\mathsf{dst}_n : (\mathsf{T}X)^n \to \mathsf{T}(X^n)$ is either $\eta_1 : 1 \to \mathsf{T}1$ (if $n = 0$), the identity map (if $n = 1$), or defined in the obvious way from the double strength of the monad T (if $n \geq 2$).

Given a $\mathsf{T}F$-coalgebra (X, γ), unfolding the coalgebra map $n \geq 1$ times yields a map $(\mathsf{T}F)^{n-1}\gamma \circ \ldots \circ \gamma : X \to (\mathsf{T}F)^n X$. Alternatively, one can use the distributive law λ to flatten, at each unfolding step, the branching arising from the presence of T in the coalgebra type:

▶ **Definition 5.5.** For a $\mathsf{T}F$-coalgebra (X, γ) and $n \geq 1$, let $\gamma_n : X \to \mathsf{T}F^n X$ be given by

- $\gamma_1 = \gamma$
- $\gamma_{n+1} = \mu_{F^{n+1}X} \circ \mathsf{T}\lambda_{F^n X} \circ \mathsf{T}F\gamma_n \circ \gamma$:

$$X \xrightarrow{\gamma} \mathsf{T}FX \xrightarrow{\mathsf{T}F\gamma_n} \mathsf{T}F\mathsf{T}F^n X \xrightarrow{\mathsf{T}\lambda_{F^n X}} \mathsf{T}^2 F^{n+1} X \xrightarrow{\mu_{F^{n+1}X}} \mathsf{T}F^{n+1}X$$

In order to relate the maps $(\mathsf{T}F)^{n-1}\gamma \circ \ldots \circ \gamma$ and γ_n, with $n \geq 1$, we note that any distributive law $\lambda : F\mathsf{T} \Rightarrow \mathsf{T}F$ yields natural transformations $(\mathsf{T}F)^n \Rightarrow \mathsf{T}F^n$:

▶ **Definition 5.6.** Let $\lambda_n : (\mathsf{T}F)^n \Rightarrow \mathsf{T}F^n$ for $n \geq 1$ be defined inductively by:

- $\lambda_1 = \mathsf{id}$
- $\lambda_{n+1} = \mu_{F^{n+1}} \circ \mathsf{T}\lambda_{F^n} \circ \mathsf{T}F\lambda_n$ for $n \geq 1$:

$$\mathsf{T}F(\mathsf{T}F)^n \xRightarrow{\mathsf{T}F\lambda_n} \mathsf{T}F\mathsf{T}F^n \xRightarrow{\mathsf{T}\lambda_{F^n}} \mathsf{T}^2F^{n+1} \xRightarrow{\mu_{F^{n+1}}} \mathsf{T}F^{n+1}$$

We can now state the following:

▶ **Lemma 5.7.** *For $n \geq 1$, we have $\gamma_n = \lambda_n \circ (\mathsf{T}F)^{n-1}\gamma \circ \ldots \circ \gamma$.*

Proof. Induction on n. The base case is trivial. The inductive step follows from the commutativity of

$$\begin{tikzcd}
X \arrow[r, "\gamma"] \arrow[d, "\gamma"] & \mathsf{T}FX \arrow[r, "\mathsf{T}F\gamma_n"] & \mathsf{T}F\mathsf{T}F^n X \arrow[r, "\mathsf{T}\lambda_{F^n X}"] & \mathsf{T}^2F^{n+1}X \arrow[r, "\mu_{F^{n+1}}"] & \mathsf{T}F^{n+1}X \\
\mathsf{T}FX \arrow[rr, "\mathsf{T}F((\mathsf{T}F)^{n-1}\gamma\circ\ldots\circ\gamma)"] & & (\mathsf{T}F)^{n+1}X \arrow[u, "\mathsf{T}F\lambda_n"'] \arrow[urr, "\lambda_{n+1}"'] & &
\end{tikzcd}$$

which, in turn, follows by the induction hypothesis and the definition of λ_{n+1}. ◀

We are finally ready to define an alternative semantics for $u\mathcal{L}^{\mathcal{V}}$. For this, recall that $u\mathcal{L}^{\mathcal{V}} = \bigcup_{n \in \omega} \mathcal{L}^{\mathcal{V}}_n$ with $\mathcal{L}^{\mathcal{V}}_n = L^n\mathcal{V}$ for $n \in \omega$.

▶ **Definition 5.8** (Path-based Semantics for $u\mathcal{L}^{\mathcal{V}}$). Let $\varphi \in \mathcal{L}_n^{\mathcal{V}}$, let (X, γ) be a $\mathsf{T}F$-coalgebra, and let $V : \mathcal{V} \to PX$ be a valuation. Define $(\!|\varphi|\!)_\gamma^V$ as the image of φ under the composition

$$L^n\mathcal{V} \xrightarrow{L^nV} L^nPX \xrightarrow{(\delta_n)_X} PF^nX \xrightarrow{\sigma_{F^nX}} P\mathsf{T}F^nX \xrightarrow{P\gamma_n} PX$$

where $\delta_n : L^nP \Rightarrow PF^n$ performs n successive applications of δ:

- $\delta_1 = \delta$
- $\delta_{n+1} = \delta_{F^n} \circ L\delta_n$ for $n \geq 1$:
$$L^{n+1}P \xRightarrow{L\delta_n} LPF^n \xRightarrow{\delta_{F^n}} PF^{n+1}$$

Thus, in the path-based semantics, in order to interpret a formula of rank n, the n-step behaviour of a state in a $\mathsf{T}F$-coalgebra is flattened into branches of n-step F-behaviours (using γ_n), and this results in the natural transformation σ (or equivalently, the extension lifting ext) only being used once, rather than at each unfolding of the coalgebra map.

Next, we show that the step-wise and path-based semantics for $u\mathcal{L}^{\mathcal{V}}$ are equivalent. For this, we need the following inductive formulation of the step-wise semantics for the uniform fragment $u\mathcal{L}^{\mathcal{V}}$:

▶ **Definition 5.9.** Let $V : \mathcal{V} \to PX$ be a valuation. For $n \geq 1$, let $\xi^n : L^n\mathcal{V} \to P(\mathsf{T}F)^nX$ be defined by:

- $\xi_1 = \sigma_{FX} \circ \delta_X \circ LV$:
$$L\mathcal{V} \xrightarrow{LV} LPX \xrightarrow{\delta_X} PFX \xrightarrow{\sigma_{FX}} P\mathsf{T}FX$$
- $\xi_{n+1} = \sigma_{F(\mathsf{T}F)^nX} \circ \delta_{(\mathsf{T}F)^nX} \circ L\xi_n$ for $n \geq 1$:
$$L^{n+1}\mathcal{V} \xrightarrow{L\xi_n} LP(\mathsf{T}F)^nX \xrightarrow{\delta_{(\mathsf{T}F)^nX}} PF(\mathsf{T}F)^nX \xrightarrow{\sigma_{F(\mathsf{T}F)^nX}} P(\mathsf{T}F)^{n+1}X$$

▶ **Lemma 5.10.** *For formulas in $\mathcal{L}_n^{\mathcal{V}}$, the step-wise semantics is obtained by post-composing ξ_n with $P\gamma \circ \ldots \circ P(\mathsf{T}F)^{n-1}\gamma : P(\mathsf{T}F)^nX \to PX$.*

Proof. Immediate from $\mathcal{L}_n^{\mathcal{V}} = L^n\mathcal{V}$. ◀

The last ingredient required for the proof of equivalence of the two semantics is the following key lemma, which allows us to move from alternating the use of the natural transformations σ and δ (as is done in the step-wise semantics) to only using the natural transformation σ once (as is done in the path-based semantics).

Since later in the paper we discuss other choices for σ, obtained by replacing $(\mathsf{T}1, \mu_1)$ with an arbitrary T-algebra $(\mathsf{T}1, \tau)$, most of the proof of the lemma uses τ instead of μ_1.

▶ **Lemma 5.11.** *Let $\delta : LP \Rightarrow PF$ and $\lambda : F\mathsf{T} \Rightarrow \mathsf{T}F$ be as in Definitions 3.2 and 5.4, respectively, and let $\sigma : P \Rightarrow P\mathsf{T}$ be the natural transformation induced by $\tau := \mu_1 : \mathsf{T}^21 \to \mathsf{T}1$, given by (2). Then the following diagram commutes:*

$$\begin{tikzcd} LP \arrow[r, Rightarrow, "L\sigma"] \arrow[d, Rightarrow, "\delta"'] & LP\mathsf{T} \arrow[r, Rightarrow, "\delta_{\mathsf{T}}"] & PF\mathsf{T} \\ PF \arrow[rr, Rightarrow, "\sigma_F"] & & P\mathsf{T}F \arrow[u, Rightarrow, "P\lambda"'] \end{tikzcd} \qquad (3)$$

Proof. The statement follows by expanding the corresponding definitions of δ and σ:

Given $(p_i) \in \iota_\lambda(PX)^n$ (with $n = \mathsf{ar}(\lambda) \geq 2$), we have:

$$\left(\begin{tikzcd} X \arrow[d, "p_i"] \\ \mathsf{T}1 \end{tikzcd}\right) \xmapsto{L\sigma_X} \left(\begin{tikzcd} \mathsf{T}X \arrow[d, "\mathsf{T}p_i"] \\ \mathsf{T}^21 \arrow[d, "\tau"] \\ \mathsf{T}1 \end{tikzcd}\right) \xmapsto{\delta_{\mathsf{T}X}} \begin{tikzcd} \coprod_{\lambda\in\Lambda}(\mathsf{T}X)^{\mathsf{ar}(\lambda)} \arrow[d, "{[0,\ldots,\bullet(\tau\circ\mathsf{T}p_i),\ldots,0]}"] \\ \mathsf{T}1 \end{tikzcd}$$

and

$$\begin{tikzcd}
 & & & \coprod_{\lambda\in\Lambda}(\mathsf{T}X^{\mathsf{ar}(\lambda)}) \arrow[d, "+_{\lambda\in\Lambda}\mathsf{dst}_{\mathsf{ar}(\lambda)}"] \\
 & & \mathsf{T}\coprod_{\lambda\in\Lambda} X^{\mathsf{ar}(\lambda)} \arrow[d, "\mathsf{T}{[0,\ldots,\bullet p_i,\ldots,0]}"] & \coprod_{\lambda\in\Lambda} \mathsf{T}(X^{\mathsf{ar}(\lambda)}) \arrow[d, "{[\mathsf{T}\iota_1,\ldots,\mathsf{T}\iota_n]}"] \\
\left(\begin{array}{c} X \\ \big\downarrow{\scriptstyle p_i} \\ \mathsf{T}1 \end{array}\right) \arrow[r, mapsto, "\delta_X"] & \coprod_{\lambda\in\Lambda} X^{\mathsf{ar}(\lambda)} \arrow[d, "{[0,\ldots,\bullet p_i,\ldots,0]}"] \arrow[r, mapsto, "\sigma_{FX}"] & \mathsf{T}^2 1 \arrow[d, "\tau"] \arrow[r, mapsto, "P\lambda"] & \mathsf{T}\coprod_{\lambda\in\Lambda} X^{\mathsf{ar}(\lambda)} \arrow[d, "\mathsf{T}{[0,\ldots,\bullet p_i,\ldots,0]}"] \\
 & \mathsf{T}1 & \mathsf{T}1 & \mathsf{T}^2 1 \arrow[d, "\tau"] \\
 & & & \mathsf{T}1
\end{tikzcd}$$

where for $q_i : X \to \mathsf{T}1$ with $i \in \{1, \ldots, n\}$, $\bullet(p_i) : X^n \to \mathsf{T}1$ takes $(x_1, \ldots, x_n)$ to $p_1(x_1) \bullet \ldots \bullet p_n(x_n)$. Thus, the commutativity of (3) amounts to the commutativity of

$$\begin{tikzcd}
\mathsf{T}X \times \mathsf{T}X \arrow[r, "\mathsf{T}p_1\times\mathsf{T}p_2"] \arrow[d, "\mathsf{dst}_{X,X}"'] & \mathsf{T}^2 1 \times \mathsf{T}^2 1 \arrow[rr, "\tau\times\tau"] \arrow[d, "\mathsf{dst}_{\mathsf{T}1,\mathsf{T}1}"'] & & \mathsf{T}1 \times \mathsf{T}1 \arrow[d, "\bullet"] \\
\mathsf{T}(X \times X) \arrow[r, "\mathsf{T}(p_1\times p_2)"] & \mathsf{T}(\mathsf{T}1 \times \mathsf{T}1) \arrow[r, "\mathsf{T}\bullet"] & \mathsf{T}^2 1 \arrow[r, "\tau"] & \mathsf{T}1
\end{tikzcd} \tag{4}$$

where for simplicity we assume $n = 2$. For $\tau = \mu_1$, the latter follows easily by naturality of dst (left rectangle) and exploiting the equivalent definition of $\bullet$ as $\mathsf{T}\pi_1 \circ \mathsf{dst}_{\mathsf{T}1,\mathsf{T}1}$, as given in [1] (right rectangle). The proof in the case when $\mathsf{ar}(\lambda) = 1$ is trivial, whereas the proof in the case when $\mathsf{ar}(\lambda) = 0$ uses the fact that $(\mathsf{T}1, \tau)$ is a T-algebra (and hence $\tau \circ \mathsf{T}\eta_1 = \mathsf{id}$). ◀

▶ Remark 5.12. The commutativity of (4) in the proof of Lemma 5.11 relies on the well-behavedness of μ_1 w.r.t. the double strength map. Replacing $\mu_1 : \mathsf{T}^2 1 \to \mathsf{T}1$ by an arbitrary T-algebra structure $\tau : \mathsf{T}^2 1 \to \mathsf{T}1$ will not, in general, make this diagram commute. For $\mathsf{T} = \mathcal{P}$, an example is the $\Box$-modality $\tau_\Box : \mathsf{T}^2 1 \to \mathsf{T}1$, defined from the $\Diamond$-modality μ_1 via the swap map $\mathsf{swap} : \mathcal{P}1 \to \mathcal{P}1$: $\tau = \mathsf{swap} \circ \mu_1 \circ \mathsf{T}\mathsf{swap}$; an easy calculation shows that commutativity of the previously mentioned diagram fails in this case.

Using Lemma 5.11, we can now state and prove the announced equivalence result.

▶ **Theorem 5.13.** *Let $\delta : LP \Rightarrow PF$, $\lambda : F\mathsf{T} \Rightarrow \mathsf{T}F$ and $\sigma : P \Rightarrow P\mathsf{T}$ be as in Lemma 5.11. Also, let (X, γ) be a $\mathsf{T}F$-coalgebra and let $V : \mathcal{V} \to PX$ be a valuation. For $\varphi \in u\mathcal{L}^{\mathcal{V}}$, $[\![\varphi]\!]^V_\gamma = (\!|\varphi|\!)^V_\gamma$.*

Proof. Since $u\mathcal{L}^{\mathcal{V}} = \bigcup_{n\in\omega} L^n\mathcal{V}$, the claim will follow from the commutativity of:

$$\begin{tikzcd}
L^n\mathcal{V} \arrow[rr, "\xi_n"] \arrow[d, "L^nV"'] & & P(\mathsf{T}F)^nX \arrow[r, "P(\mathsf{T}F)^{n-1}\gamma\circ\ldots\circ\gamma"] & PX \\
L^nPX \arrow[r, "(\delta_n)_X"'] & PF^nX \arrow[r, "\sigma_{F^nX}"'] & P\mathsf{T}F^nX \arrow[u, "P(\lambda_n)_X"'] \arrow[ur, "P\gamma_n"'] &
\end{tikzcd}$$

where the right triangle commutes by Lemma 5.7, and the commutativity of the left rectangle is proved below by induction on n.

The case $n = 1$ is trivial. The inductive step follows from the commutativity of

$$\begin{tikzcd}
L^{n+1}\mathcal{V} \arrow[rr, "L\xi_n"] \arrow[d, "L^{n+1}V"'] & & LP(\mathsf{T}F)^n X \arrow[r, "\delta_{(\mathsf{T}F)^n X}"] & PF(\mathsf{T}F)^n X \arrow[r, "\sigma_{F(\mathsf{T}F)^n X}"] & P(\mathsf{T}F)^{n+1} X \\
L^{n+1}PX \arrow[r, "L(\delta_n)_X"] \arrow[dr, "(\delta_{n+1})_X"'] & LPF^n X \arrow[r, "L\sigma_{F^n X}"] \arrow[d, "\delta_{F^n X}"] & LP\mathsf{T}F^n X \arrow[r, "\delta_{\mathsf{T}F^n X}"] \arrow[u, "LP(\lambda_n)_X"'] & PF\mathsf{T}F^n X \arrow[r, "\sigma_{F\mathsf{T}F^n}"] \arrow[u, "PF(\lambda_n)_X"'] & P\mathsf{T}F\mathsf{T}F^n X \arrow[u, "P\mathsf{T}F(\lambda_n)_X"'] \\
& PF^{n+1}X \arrow[rr, "\sigma_{F^{n+1}X}"'] & & P\mathsf{T}F^{n+1}X \arrow[r, "\sigma_{\mathsf{T}F^{n+1}X}"'] \arrow[u, "P\lambda_{F^n X}"'] & P\mathsf{T}^2 F^{n+1} X \arrow[u, "P\mathsf{T}\lambda_{F^n X}"']
\end{tikzcd}$$

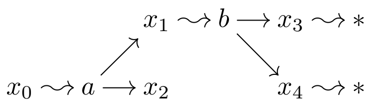
where the top arrow is ξ_{n+1}, the top-left rectangle commutes by the induction hypothesis, the top-middle rectangle commutes by naturality of δ, the bottom-left triangle is the definition of δ_{n+1}, the bottom-middle rectangle commutes by Lemma 5.11, the top-right and bottom-right rectangles commute by naturality of σ, and finally the long arrow from $L^{n+1}\mathcal{V}$ to $P(\mathsf{T}F)^{n+1}X$ is $P\lambda_{n+1} \circ \sigma_{F^{n+1}X} \circ (\delta_{n+1})_X \circ L^{n+1}V$ as required – the latter follows from:

$$\begin{aligned}
P\mathsf{T}F\lambda_n \circ P\mathsf{T}\lambda_{F^n} \circ \sigma_{\mathsf{T}F^{n+1}} \circ \sigma_{F^{n+1}} \quad &= \quad \text{(Lemma 3.1)} \\
P\mathsf{T}F\lambda_n \circ P\mathsf{T}\lambda_{F^n} \circ P\mu_{F^{n+1}} \circ \sigma_{F^{n+1}} \quad &= \quad \text{(Definition 5.6)} \\
P\lambda_{n+1} \circ \sigma_{F^{n+1}} &
\end{aligned}$$

This concludes the proof. ◀

The next example confirms that by using $\tau_\Box$ instead of μ_1 to resolve branching for $\mathsf{T} = \mathcal{P}$, Theorem 5.13 does not hold for functors F with associated linear time modalities of arity 2 or greater.

▶ **Example 5.14.** Assume that $\tau_\Box$ is used to resolve branching, and consider the following $\mathcal{P}(1 + A \times \mathsf{Id} \times \mathsf{Id})$-coalgebra (X, γ):



where $\rightsquigarrow$ is used for nondeterministic transitions (and thus x_2 is a deadlock state). Under the step-wise semantics, the formula $[a](*, *)$ does not hold in x_0, as although x_2 satisfies $*$ (since it has no outgoing transitions), x_1 does not: according to the definition, for a state to satisfy $*$, all transitions from that state (if any) must be terminating ones. However, the map $\gamma_2 : X \to \mathcal{P}(1 + A \times (1 + A \times X) \times (1 + A \times X))$ maps x_0 to the empty set: again, this is because x_2 has no transitions and therefore the flattening performed by γ_2 results in an empty set of linear time behaviours of depth 2; as a result, under the path-based semantics, the formula holds.

In spite of the above, a generalisation of Theorem 5.13 to the case when $\sigma : P \Rightarrow P\mathsf{T}$ arises from an arbitrary T-algebra $(\mathsf{T}1, \tau)$ can be formulated, as suggested by the next example.

▶ **Example 5.15.** The case $\mathsf{T}' = \mathcal{P}^+$ with $\mathcal{P}_+ : \mathsf{Set} \to \mathsf{Set}$ cannot be directly covered by our approach, since in this case $\mathsf{T}'\emptyset \neq 1$. However, any $\mathsf{T}'F$-coalgebra can be viewed as a $\mathsf{T}F$-coalgebra with $\mathsf{T} = \mathcal{P}$, and for $\mathsf{T}F$-coalgebras arising in this way, the proof of Theorem 5.13 does generalise, as it only requires the following to commute (instead of (3)):

$$\begin{tikzcd}
LP \arrow[r, Rightarrow, "L\sigma"] \arrow[d, Rightarrow, "\delta"'] & LP\mathsf{T}' \arrow[r, Rightarrow, "\delta_{\mathsf{T}'}"] & PF\mathsf{T}' \\
PF \arrow[rr, Rightarrow, "\sigma_F"] & & P\mathsf{T}'F \arrow[u, Rightarrow, "P\lambda"']
\end{tikzcd}$$

Using the same reasoning as in Lemma 5.11, the above follows from the commutativity of

$$\begin{tikzcd}
\mathsf{T}'X \times \mathsf{T}'X \arrow[r, "\iota_X \times \iota_X"] \arrow[d, "\mathsf{dst}'_{X,X}"'] & \mathsf{T}X \times \mathsf{T}X \arrow[r, "\mathsf{T}p_1 \times \mathsf{T}p_2"] \arrow[d, "\mathsf{dst}_{X,X}"'] & \mathsf{T}^2 1 \times \mathsf{T}^2 1 \arrow[rr, "\tau \times \tau"] \arrow[d, "\mathsf{dst}_{\mathsf{T}1,\mathsf{T}1}"'] & & \mathsf{T}1 \times \mathsf{T}1 \arrow[d, "\bullet"] \\
\mathsf{T}'(X \times X) \arrow[r, "\iota_{X \times X}"] & \mathsf{T}(X \times X) \arrow[r, "\mathsf{T}(p_1 \times p_2)"] & \mathsf{T}(\mathsf{T}1 \times \mathsf{T}1) \arrow[r, "\mathsf{T}\bullet"] & \mathsf{T}^2 1 \arrow[r, "\tau"] & \mathsf{T}1
\end{tikzcd}$$

where $\iota : \mathsf{T}' \Rightarrow \mathsf{T}$ is the inclusion. Thus, commutativity of (4) in the proof of Lemma 5.11 can be replaced by the commutativity of the outer diagram below:

$$\begin{tikzcd}
\mathsf{T}'\mathsf{T}1 \times \mathsf{T}'\mathsf{T}1 \arrow[r, "\iota_{\mathsf{T}1} \times \iota_{\mathsf{T}1}"] \arrow[d, "\mathsf{dst}'_{\mathsf{T}1,\mathsf{T}1}"'] & \mathsf{T}^2 1 \times \mathsf{T}^2 1 \arrow[rr, "\tau \times \tau"] \arrow[d, "\mathsf{dst}_{\mathsf{T}1,\mathsf{T}1}"'] & & \mathsf{T}1 \times \mathsf{T}1 \arrow[d, "\bullet"] \\
\mathsf{T}'(\mathsf{T}1 \times \mathsf{T}1) \arrow[r, "\iota_{\mathsf{T}1 \times \mathsf{T}1}"'] & \mathsf{T}(\mathsf{T}1 \times \mathsf{T}1) \arrow[r, "\mathsf{T}\bullet"] & \mathsf{T}^2 1 \arrow[r, "\tau"] & \mathsf{T}1
\end{tikzcd} \tag{5}$$

which states that the right rectangle in (4) commutes on the sub-domain $\mathsf{T}'\mathsf{T}1 \times \mathsf{T}'\mathsf{T}1$ of $\mathsf{T}^2 1 \times \mathsf{T}^2 1$. An easy calculation shows that this holds for $\tau_{\Box}$.

The argument in the previous example can be captured in a more general result on the equivalence between the path-based and the step-wise semantics for $u\mathcal{L}^{\mathcal{V}}$.

▶ **Theorem 5.16.** *Let T' be a commutative sub-monad of the monad T, let $\delta : LP \Rightarrow PF$ and $\lambda : F\mathsf{T} \Rightarrow \mathsf{T}F$ be as in Definitions 3.2 and 5.4, respectively, and let $\sigma : P \Rightarrow P\mathsf{T}$ be induced by a choice of $\tau : \mathsf{T}^2 1 \to \mathsf{T}1$ that makes the outer diagram in (5) commute. Then for a $\mathsf{T}'F$-coalgebra (X, γ) (viewed as a $\mathsf{T}F$-coalgebra), a valuation $V : \mathcal{V} \to PX$ and a formula $\varphi \in u\mathcal{L}^{\mathcal{V}}$, $\llbracket \varphi \rrbracket_\gamma^V = (\!|\varphi|\!)_\gamma^V$.*

In particular, Theorem 5.16 applies when $\mathsf{T}' = \mathsf{T}$ and $\tau : \mathsf{T}^2 1 \to \mathsf{T}1$ is such that the right rectangle in (5) commutes.

Finally, we note that the proof of Theorem 5.13 only makes use of the specific (canonical) choice of linear time modalities when it comes to applying Lemma 5.11. As a result, a generalisation of Theorem 5.13 to an arbitrary choice of linear time modalities can also be stated.

▶ **Theorem 5.17.** *Let $\lambda : F\mathsf{T} \Rightarrow \mathsf{T}F$ be as in Definition 5.4, and let $L : \mathsf{Set} \to \mathsf{Set}$, $\delta : LP \Rightarrow PF$ and $\sigma : P \to P\mathsf{T}$ (induced by $\tau : \mathsf{T}^2 1 \to \mathsf{T}1$) be such that Lemma 5.11 holds. Then the path-based and the step-wise semantics of $u\mathcal{L}^{\mathcal{V}}$ coincide.*

▶ **Example 5.18.** Modalities incorporating restricted disjunctions, as used e.g. in [2], can easily be added. For example, when $F = A \times \mathsf{Id} \simeq \coprod_{a \in A} \mathsf{Id}$, one can consider additional (binary) modalities of the form $[a]\_ \sqcup [b]\_$ with $a \neq b \in A$, with the obvious one-step interpretation:

$$\delta_X([a]p \sqcup [b]q)(\iota_c(x)) = \begin{cases} p(x), & \text{if } c = a \\ q(x), & \text{if } c = b \\ 0, & \text{otherwise} \end{cases}$$

This generalises to any polynomial functor F and similar disjunction-like modalities.

5.2 Path-based Semantics for $u\tilde{\mathcal{L}}^{\mathsf{Free}(\mathcal{V})}$

Giving a path-based semantics for $u\tilde{\mathcal{L}}^{\mathsf{Free}(\mathcal{V})}$ can be done in much the same way as for $u\mathcal{L}^{\mathcal{V}}$, since the logic functor used to deal with branching is still the identity functor (now on $\mathsf{Alg}(\mathsf{T})$). For completeness, this section sketches the main definitions and results, all very similar to their counterparts in Section 5.1.

▶ **Definition 5.19.** Let $\tilde{\delta}_n : \tilde{L}^n\tilde{P} \Rightarrow \tilde{P}F^n$ be given by:

- $\tilde{\delta}_1 = \tilde{\delta}$
- $\tilde{\delta}_{n+1} = \tilde{\delta}_{F^n} \circ \tilde{L}\tilde{\delta}_n$ for $n \geq 1$.

▶ **Definition 5.20** (Path-based Semantics for $u\tilde{\mathcal{L}}^{\mathsf{Free}(\mathcal{V})}$)**.** Let $\varphi \in \tilde{\mathcal{L}}_n^{\mathsf{Free}(\mathcal{V})}$, let (X, γ) be a $\mathsf{T}F$-coalgebra, and let $V : \mathcal{V} \to PX$ be a valuation. Define $\llparenthesis \varphi \rrparenthesis_\gamma^{V^\sharp}$ as the image of φ under the composition

$$\tilde{L}^n\mathsf{Free}(\mathcal{V}) \xrightarrow{\tilde{L}^n V^\sharp} \tilde{L}^n\tilde{P}X \xrightarrow{(\tilde{\delta}_n)_X} \tilde{P}F^nX \xrightarrow{\tilde{\sigma}_{F^nX}} \tilde{P}\mathsf{T}F^nX \xrightarrow{\tilde{P}\gamma_n} \tilde{P}X$$

▶ **Definition 5.21.** Let $V : \mathcal{V} \to PX$ be a valuation. For $n \geq 1$, let $\tilde{\xi}^n : \tilde{L}^n\mathsf{Free}(\mathcal{V}) \to \tilde{P}(\mathsf{T}F)^nX$ be defined by:

- $\tilde{\xi}_1 = \tilde{\sigma}_{FX} \circ \tilde{\delta}_X \circ \tilde{L}V^\sharp$,
- $\tilde{\xi}_{n+1} = \tilde{\sigma}_{F(\mathsf{T}F)^nX} \circ \tilde{\delta}_{(\mathsf{T}F)^nX} \circ \tilde{L}\tilde{\xi}_n$ for $n \geq 1$.

▶ **Lemma 5.22.** *For formulas in $\tilde{\mathcal{L}}_n^{\mathsf{Free}(\mathcal{V})}$, the step-wise semantics is obtained by post-composing $\tilde{\xi}_n$ with $\tilde{P}\gamma \circ \ldots \circ \tilde{P}(\mathsf{T}F)^{n-1}\gamma : \tilde{P}(\mathsf{T}F)^nX \to PX$.*

The next lemma allows Theorem 5.13 to be lifted to the logics $\tilde{\mathcal{L}}^{\mathsf{Free}(\mathcal{V})}$.

▶ **Lemma 5.23.** *Let $\delta : LP \Rightarrow PF$, $\lambda : F\mathsf{T} \Rightarrow \mathsf{T}F$ and $\sigma : P \Rightarrow P\mathsf{T}$ be as in Lemma 5.11, and let $\tilde{\delta} : \tilde{L}\tilde{P} \Rightarrow \tilde{P}F$ and $\tilde{\sigma} : \tilde{P} \Rightarrow \tilde{P}\mathsf{T}$ arise from δ and σ as before. Then the following diagram commutes:*

$$\begin{tikzcd}
\tilde{L}\tilde{P} \arrow[r, Rightarrow, "\tilde{L}\tilde{\sigma}"] \arrow[d, Rightarrow, "\tilde{\delta}"'] & \tilde{L}\tilde{P}\mathsf{T} \arrow[r, Rightarrow, "\tilde{\delta}_{\mathsf{T}}"] & \tilde{P}F\mathsf{T} \\
\tilde{P}F \arrow[rr, Rightarrow, "\tilde{\sigma}_F"'] & & \tilde{P}\mathsf{T}F \arrow[u, Rightarrow, "\tilde{P}\lambda"']
\end{tikzcd} \tag{6}$$

Proof. By freeness of $\tilde{L}\tilde{P}$, it suffices to show that pre-composing the image under U of the above diagram with $\eta_{LU\tilde{P}}$ commutes in Set:

$$\begin{tikzcd}
LP \arrow[r, equal] \arrow[dd, Rightarrow, "\delta"'] & LU\tilde{P} \arrow[r, Rightarrow, "LU\tilde{\sigma}=L\sigma"] \arrow[d, Rightarrow, "\eta_{LU\tilde{P}}"'] & LU\tilde{P}\mathsf{T} \arrow[d, Rightarrow, "\eta_{LU\tilde{P}\mathsf{T}}"'] \arrow[dr, Rightarrow, "\delta_{\mathsf{T}}"] & \\
& U\mathsf{Free}LU\tilde{P} \arrow[r, Rightarrow, "U\mathsf{Free}LU\tilde{\sigma}"] \arrow[d, Rightarrow, "U\tilde{\delta}"'] & U\mathsf{Free}LU\tilde{P}\mathsf{T} \arrow[r, Rightarrow, "U\tilde{\delta}_{\mathsf{T}}"] & U\tilde{P}F\mathsf{T} \\
PF \arrow[r, equal] & U\tilde{P}F \arrow[rr, Rightarrow, "U\tilde{\sigma}_F=\sigma_F"'] & & U\tilde{P}\mathsf{T}F \arrow[u, Rightarrow, "U\tilde{P}\lambda=P\lambda"']
\end{tikzcd}$$

This, in turn, is a direct consequence of Lemma 5.11. ◀

▶ **Theorem 5.24.** *Let $\delta : LP \Rightarrow PF$, $\lambda : F\mathsf{T} \Rightarrow \mathsf{T}F$ and $\sigma : P \Rightarrow P\mathsf{T}$ be as in Lemma 5.23. Also, let (X, γ) be a $\mathsf{T}F$-coalgebra and let $V : \mathcal{V} \to PX$ be a valuation. For $\varphi \in u\tilde{\mathcal{L}}^{\mathsf{Free}(\mathcal{V})}$, $\llbracket \varphi \rrbracket_\gamma^{V^\sharp} = \llparenthesis \varphi \rrparenthesis_\gamma^{V^\sharp}$.*

Proof. Exactly the same as the proof of Theorem 5.13, except that Lemma 5.23 is used instead of Lemma 5.11. ◀

6 Concluding Remarks

This paper showed how to incorporate propositional operators arising canonically from the branching monad T into the linear time logics proposed in [2]. This involved moving to the Eilenberg-Moore category of T. The addition of arbitrary propositional operators to the logics appears to be incompatible with their step-wise semantics, and our results provide operators that *can be* safely added to the logics. The paper also provided an alternative, equivalent path-based semantics for the uniform modal fragments of the logics in loc. cit., as well as of their enhancements with canonical propositional operators (assuming canonical choices for both the branching and the linear time modalities), and explored conditions under which non-canonical choices for the modalities do not disrupt the equivalence result.

Future work will investigate extending the path-based semantics proposed here to the full $\mathcal{L}^{\mathcal{V}}$ and $\tilde{\mathcal{L}}^{\mathsf{Free}(\mathcal{V})}$ in the first instance, and subsequently also to $\mathcal{L}_{\mu}$ and $\tilde{\mathcal{L}}_{\mu}$. We also plan to investigate the relationship between our logics and recent work on graded monads and associated trace logics [17].

Acknowledgements. Thanks are due to Ichiro Hasuo and Toby Wilkinson for stimulating discussions, and to the anonymous referees for useful comments. The challenge of incorporating arbitrary propositional operators into the step-wise semantics was the subject of extensive discussions with Ichiro Hasuo. The encoding of the $\bigcirc$ modality in Example 4.4 was suggested by Toby Wilkinson. The observation in Remark 2.4 was made by one of the anonymous referees.

References

1 Corina Cîrstea. From branching to linear time, coalgebraically. In *Proceedings, FICS 2013*, pages 11–27, 2013.

2 Corina Cîrstea. A coalgebraic approach to linear-time logics. In *Proceedings, FOSSACS 2014*, pages 426–440, 2014.

3 Corina Cîrstea and Dirk Pattinson. Modular construction of complete coalgebraic logics. *Theor. Comput. Sci.*, 388(1-3):83–108, 2007.

4 Dion Coumans and Bart Jacobs. Scalars, monads, and categories. In *Quantum Physics and Linguistics. A Compositional, Diagrammatic Discourse*, pages 184–216. Oxford Univ. Press, 2013.

5 B.A. Davey and H.A. Priestley. *Introduction to Lattices and Order (2. ed.)*. Cambridge University Press, 2002.

6 Ichiro Hasuo. Generic weakest precondition semantics from monads enriched with order. In *Proceedings, CMCS 2014*, pages 10–32, 2014.

7 Ichiro Hasuo, Bart Jacobs, and Ana Sokolova. Generic trace semantics via coinduction. *Logical Methods in Computer Science*, 3(4), 2007.

8 Michael Huth and Marta Z. Kwiatkowska. Quantitative analysis and model checking. In *Proceedings, LICS 1997*, pages 111–122, 1997.

9 Bart Jacobs. Introduction to coalgebra. Towards mathematics of states and observations. Draft.

10 Bart Jacobs, Alexandra Silva, and Ana Sokolova. Trace semantics via determinization. In *Proceedings, CMCS 2012*, pages 109–129, 2012.

11 Christian Kissig and Alexander Kurz. Generic trace logics. *CoRR*, abs/1103.3239, 2011.

12 Bartek Klin and Jurriaan Rot. Coalgebraic trace semantics via forgetful logics. In *Proceedings, FOSSACS 2015*, pages 151–166, 2015.

13 Anders Kock. Monads and extensive quantities, 2011. arXiv:1103.6009.

14 Anders Kock. Commutative monads as a theory of distributions. *Theory and Applications of Categories*, 26(4):97–131, 2012.

15 Clemens Kupke, Alexander Kurz, and Dirk Pattinson. Algebraic semantics for coalgebraic logics. *Electr. Notes Theor. Comput. Sci.*, 106:219–241, 2004.

16 Clemens Kupke, Alexander Kurz, and Yde Venema. Stone coalgebras. *Theor. Comput. Sci.*, 327(1-2):109–134, 2004.

17 Stefan Milius, Dirk Pattinson, and Lutz Schröder. Generic trace semantics and graded monads. This volume.

An Intensionally Fully-abstract Sheaf Model for π*

Clovis Eberhart[1], **Tom Hirschowitz**[2], **and Thomas Seiller**[3]

1 **Université Savoie Mont-Blanc, France**
2 **CNRS, Université Savoie Mont-Blanc, France**
3 **Université Paris 7, France**

Abstract

Following previous work on CCS, we propose a compositional model for the pi-calculus in which processes are interpreted as sheaves on certain simple sites. We define an analogue of fair testing equivalence in the model and show that our interpretation is intensionally fully abstract for it. That is, the interpretation preserves and reflects fair testing equivalence; and furthermore, any strategy is fair testing equivalent to the interpretation of some process. The central part of our work is the construction of our sites, whose heart is a combinatorial presentation of pi-calculus traces in the spirit of string diagrams. As in previous work, the sheaf condition is analogous to innocence in Hyland-Ong/Nickau games.

1998 ACM Subject Classification F.3.2 Semantics of Programming Languages

Keywords and phrases concurrency, sheaves, causal models, games

Digital Object Identifier 10.4230/LIPIcs.CALCO.2015.86

1 Introduction

Operational semantics of programming languages standardly model the execution of programs as paths in a certain labelled transition system (LTS). Under this interpretation, different possible interleavings of parallel actions yield different paths. Verification on LTSs thus incurs a well-known state explosion problem. Similarly, causality between various actions, visible in the syntax, is lost in the LTS, thus making, e.g., error diagnostics difficult [17].

Causal models, originally designed for Petri nets [37] and Milner's CCS [42], intend to remedy both problems, but have yet to be applied to full-scale programming languages. They have recently been extended in two different directions: (1) by Crafa et al. [10] to Milner's π-calculus, and (2) by Melliès [32] to Girard's linear logic. The former extension accounts for the subtle interaction of channel creation with synchronisation in π, a significant technical achievement, 30 years after the first causal semantics for CCS. The latter is the first causal model for functional languages (inspired by Hyland-Ong's and Nickau's *games* models for PCF [36, 24]). An important challenge is now the search for a causal model of full-fledged languages with both concurrent and functional features. Winskel and collaborators are currently working in this direction, using extensions of Melliès's approach [39, 43, 8].

In previous work [23, 21, 22], we have proposed a causal model for CCS based on a different approach. We here push this approach further by applying it to the π-calculus.

* The authors acknowledge support from French ANR projets blancs PiCoq ANR 2010 BLAN 0305 01 and Récré ANR-11-BS02-0010.

6th International Conference on Algebra and Coalgebra in Computer Science (CALCO'15).
Editors: Lawrence S. Moss and Paweł Sobociński; pp. 86–100
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1.1 Traces and naive concurrent strategies

In standard causal models, execution traces essentially consist of partially ordered sets of atomic 'events'. Our approach relies on a new notion of trace, which we briefly sketch. There is first a (straightforward) notion of *configuration*, which is essentially a finite hypergraph whose nodes are thought of as *agents*, and whose hyperedges between nodes $x_1, \ldots, x_n$ are thought of as communication channels shared by $x_1, \ldots, x_n$. There is then a notion of *atomic action* from one configuration to another, thought of as a 'rule of the game'. Examples of atomic actions are: an agent creates a new, private communication channel; an agent forks into two new agents connected to the same channels; an agent sends some channel a over some channel b to some other agent. We finally have a notion of trace which allows several atomic actions to occur, in a way that only retains some minimal causality information between them. We here mean, e.g., information such as: 'such agent outputs on such channel only after having created such other channel'.

The main purpose of our notion of trace is to interpret π-calculus processes as some kind of strategies over them. Most naively, a strategy on some configuration X is a prefix-closed set of 'accepted' traces from X. But what should prefix mean in our setting? Well, we may view traces with initial configuration X and final configuration Y as morphisms $Y \twoheadrightarrow X$. Sequential composition of traces, denoted by $\bullet$, yields an analogue of prefix ordering, defined by $t \leq t \bullet w$. This however fails to suit our needs on three counts.

We start by examining the first two problems. The first, easy one is that there is an obvious notion of isomorphism between traces, under which strategies should be closed. The second problem is more serious: until now, these too naive strategies are not concurrent enough to adequately model CCS or the π-calculus.

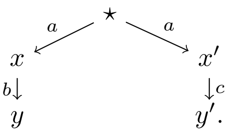
▶ **Example 1** (Milner's coffee machines)**.** Consider the CCS processes $P = (a.b + a.c)$ and $Q = a.(b + c)$. The process P has two ways of inputting on a and then, depending on the chosen way, inputs either on b or on c. The process Q inputs on a and then has both possibilities of inputting on b or c. Both processes, however, accept exactly the same traces (in the standard sense), namely $\{\epsilon, a, ab, ac\}$, where ϵ denotes the empty trace.

Thus, taking strategies to be prefix-closed sets of traces would prevent us from directly modelling any reasonably fine behavioural equivalence on processes. Inspired by *presheaf models* [26], we remedy both problems at once by passing from prefix-closed sets of traces to presheaves (of finite sets) on traces. Indeed, in the simple case where traces on X form a mere poset $\mathbb{T}(X)$ by prefix ordering, a prefix-closed set of traces is nothing but a contravariant functor from $\mathbb{T}(X)$ to the ordinal 2, viewed as a category. The latter has two objects 0 and 1 and just one non-trivial morphism $0 \to 1$. The idea is that a functor $S \colon \mathbb{T}(X)^{op} \to 2$ maps any trace to 1 when it is accepted, and to 0 otherwise. Furthermore, if $t \leq t'$, i.e., t is a prefix of t', then we have a morphism $t \to t'$ which should be mapped by S to some morphism $S(t') \to S(t)$. If t' is accepted then $S(t') = 1$, so this has to be a morphism $1 \to S(t)$. Because there are no morphisms $1 \to 0$, this entails $S(t) = 1$, hence prefix-closedness of the corresponding strategy.

Now our traces naturally form a proper category $\mathbb{T}(X)$, encompassing both prefix ordering and isomorphisms between traces, so we are led to considering functors $\mathbb{T}(X)^{op} \to 2$. This retains prefix-closedness and solves our first problem: for any $t \cong t'$, functoriality imposes $S(t) \cong S(t')$. Our second problem is then solved by replacing such functors with *presheaves*, i.e., functors $\mathbb{T}(X)^{op} \to \mathsf{Set}$.

▶ **Example 2.** In Example 1, the two ways that P has to accept inputting on a may be reflected by mapping the trace a to some two-element set. More precisely, P may be modelled by the presheaf S defined on the left and pictured on the right:

- $S(\epsilon) = \{\star\}$,
- $S(a) = \{x, x'\}$,
- $S(ab) = \{y\}$,
- $S(ac) = \{y'\}$,
- S empty otherwise,
- $S(\epsilon \hookrightarrow a) = \{x \mapsto \star, x' \mapsto \star\}$,
- $S(a \hookrightarrow ab) = \{y \mapsto x\}$,
- $S(a \hookrightarrow ac) = \{y' \mapsto x'\}$,



Presheaves thus may ‘accept a trace in several ways’: the trace a is here accepted in two ways, x and x'. The process Q is of course modelled by identifying x and x'.

As it turns out, we actually only need *finitely many* ways of accepting each trace. Thus, we arrive at a first sensible notion of strategy given by presheaves of finite sets, i.e., functors $\mathbb{T}(X)^{op} \to \mathsf{set}$, where set denotes the category with as objects all finite subsets of $\mathbb{N}$, with all maps between them. We call them *(naive) strategies* in the sequel.

▶ **Notation 3.** *For any $\mathcal{C}$, let $\widehat{\mathcal{C}}$ denote the category of presheaves of finite sets over $\mathcal{C}$.*

1.2 Innocence as a sheaf condition

The third problem evoked above is that functors $\mathbb{T}(X)^{op} \to \mathsf{set}$ allow some undesirable behaviours. Intuitively, in π just as in CCS, agents should not have any control over the routing of messages.

▶ **Example 4.** Consider a configuration X with three agents x, y, and z sharing a communication channel a, and a strategy S accepting (1) the trace where x outputs on a, (2) the trace where y inputs on a, and (3) the trace where z inputs on a. Then, both synchronisations should be accepted by S. However, one easily constructs a naive strategy in which one is refused (see Example 19).

In order to rectify this deficiency, we enrich strategies with ‘local’ information. The idea is that a strategy should not only accept or refuse traces on the whole configuration X, but also traces on all possible subconfigurations of X. Furthermore, this local information should fit together coherently.

▶ **Example 5.** Consider the configuration X of Example 4. Any strategy on X should now in particular include independent strategies for each of the three agents x, y, and z. Coherence means that in order for a trace to be accepted, it should be enough for it to be ‘locally accepted’, i.e., at every stage in the trace, each agent should approve what she sees of the next action. E.g., if the next action is a synchronisation $x \to y$ with x outputting and y inputting on some channel a, then all that’s required for the synchronisation to be accepted is that x accepts to output and y accepts to input. Consequently, if some other agent z also accepts to input on a at this stage, then the synchronisation $x \to z$ is also accepted.

We call this putative coherence condition *innocence* by analogy with Hyland and Ong’s notion [24]. In order to formalise it, we first extend our category of traces $\mathbb{T}(X)$ on X with new objects representing traces on subconfigurations of X. We also add new morphisms, which are about ‘locality’:

▶ **Example 6.** Consider the configuration X with two unary agents x_1 and x_2. There is a trace t on X in which both agents fork. Consider now the subconfiguration Y of X consisting solely of x_1 and the trace t' on Y in which x_1 merely forks. There is a morphism $t' \to t$ in our new category.

This extended category, $\mathbb{T}_X$, yields an intermediate notion of strategy, given by functors $\mathbb{T}_X^{op} \to \mathsf{set}$. Among the new objects, we have in particular traces on just one agent of X, which are obtained by sequentially composing atomic actions whose final configuration again consists of one agent. We call this particular kind of trace a *view*. Views are the most 'local' kind of objects in $\mathbb{T}_X$. They form a subcategory $\mathbb{V}_X$ of $\mathbb{T}_X$.

▶ **Example 7.** If X merely consists of an agent x linked to n communication channels, consider the atomic action given by x forking into two new agents, say x_1 and x_2. This action, viewed as an object of $\mathbb{T}_X$ has three subobjects which are views: (1) the 'identity' view, in which nothing happens, (2) π_n^l, which represents the left-hand branch (to x_1), (3) and π_n^r, which represents the right-hand branch (to x_2).

The inclusion $\mathbb{V}_X \hookrightarrow \mathbb{T}_X$ induces a simple Grothendieck topology [30] on $\mathbb{T}_X$, which amounts to decreeing that any trace is covered by its views. We finally call any $S \colon \mathbb{T}_X^{op} \to \mathsf{set}$ *innocent* precisely when it is a *sheaf* for this Grothendieck topology. In particular, giving an innocent presheaf on $\mathbb{T}_X$ is equivalent (up to isomorphism) to separately giving an innocent presheaf for each agent of X, which rules out the undesirable behaviour described in Example 4.

Sheaves on $\mathbb{T}_X$ form a category S_X, which is small thanks to our use of set instead of Set. They furthermore map back to naive strategies, i.e., presheaves on $\mathbb{T}(X)$, by forgetting the local information. (This forgetful functor has a left adjoint.) Finally, because the considered topology is particularly simple, sheaves are equivalent to presheaves on views, i.e., $\mathsf{S}_X \simeq \widehat{\mathbb{V}_X}$ (recalling Notation 3). In summary, we have three categories of strategies: *naive* strategies are presheaves on traces $\widehat{\mathbb{T}(X)}$, *innocent* strategies S_X are sheaves on the extended category of traces $\mathbb{T}_X$, and so-called *behaviours* B_X are presheaves on the category of views $\mathbb{V}_X$. The last two are equivalent, and we furthermore have an adjunction

$$\begin{tikzcd} \widehat{\mathbb{T}(X)} \arrow[r, bend left] \arrow[r, phantom, "\perp"] & \mathsf{S}_X \arrow[l, bend left] \end{tikzcd}.$$

We use both sides of the equivalence: behaviours directly lead to our compositional interpretation $[\![-]\!] \colon \mathit{Pi} \to \mathsf{S}$ of π-calculus processes, and innocent strategies are used below as the basis for our semantic definition of fair testing equivalence.

1.3 Main result

What should we do in order to demonstrate adequacy of our model? By definition, causal models expose some intensional information. Hence, equality is generally much finer than any reasonable behavioural equivalence, so we should not base our main result on it. On the other hand, causal models are supposed to be 'compositional', i.e., to come equipped with an interpretation of syntactic operations in the model. The natural thing to do is thus to choose some behavioural equivalence from the operational side, use compositionality to transpose it to the model, and prove that the two coincide. More precisely, the considered equivalence induces by quotienting two 'extensional collapses', one syntactic and the other semantic, and we want to prove that the translation $[\![-]\!]$ induces a bijection between both extensional collapses. Following [1], we call this *intensional full abstraction* for the considered equivalence.

We here focus on so-called *testing equivalences* [11, 35, 5, 38], which are defined in two stages. First, one chooses a 'mode of interaction'. That is, one defines what the relevant *tests* are for a given process and specifies how the two should interact. Typically, tests for P are other processes T with the same free communication channels as P, and interaction is just parallel composition $P \mid T$. The second stage amounts to choosing when $P \mid T$ is *successful*. E.g., in may testing equivalence $P \mid T$ is successful just when there exists a transition $(P \mid T) \overset{\heartsuit}{\Rightarrow} P'$ (that is, a $\heartsuit$ transition, possibly surrounded by silent transitions), where $\heartsuit$ is some action fixed in advance. In must testing equivalence, success is when all maximal (possibly infinite) transition sequences contain at least one $\heartsuit$ transition. In fair testing equivalence (see [7] for some motivation and an adaptation to π), one requires that all silent sequences $(P \mid T) \Rightarrow P'$ extend to some sequence $P' \Rightarrow P'' \xrightarrow{\heartsuit} P'''$ ending with a $\heartsuit$ transition. In this paper, we focus on the latter, i.e., we prove (Theorem 25) that our model is intensionally fully-abstract for fair testing equivalence. However, we show in the long version [12] that our proof applies to a wide range of testing equivalences.

1.4 Contributions

Since this paper follows the same approach as previous work on CCS [23, 21, 22], we should explain in which sense extending the approach to π is more than an easy application.

A first contribution comes from the fact that, in order to even define composition in our category of traces (see our online draft [12] for details), we need to show that traces form the total category of a *fibration* [25] over configurations. In previous work, this was done in an *ad hoc* way. We here introduce a more satisfactory approach based on *factorisation systems* [28, 15].

A second significant contribution is prompted by the interplay between synchronisation and private channels in π, which is notoriously subtle to handle. And indeed, our proof method for CCS fails miserably on π. One reason for this, we think, is that our notion of trace for π, though simple and natural, is not 'modular' enough, in the sense that a trace contains strictly more information than the collection of all 'local' information accessible to agents (i.e., of all of its views, in the above sense). Thus, adapting our proof technique from CCS would have required us to define a much more complex but more modular notion of trace. Instead, we here take a somewhat rougher route, as sketched in Section 4.

Finally, as mentioned above, our proof now applies not only to fair testing equivalence, but also to a whole class of testing equivalences.

1.5 Related work

Beyond the obviously closely related, already mentioned work of Winskel et al., we should mention other causal and interleaving models for π, e.g., [34, 13, 4, 9, 10, 6, 14, 40, 19]. All of these models are based on some LTS for π. Instead, ours is rather based on *reduction rules*. The subtleties usually showing up in LTSs, related to mixing synchronisation and private channels, do resurface in our proof of intensional full abstraction, but not in the definition of our model. Indeed, it merely goes by describing the 'rule of the game' in π, and applying the general framework of *playgrounds* [22].

Another general framework relating operational and denotational descriptions of programs is Kleene coalgebra [3], which is mainly designed for automata theory. Playgrounds may be viewed as adapting ideas from Kleene coalgebra to the process algebraic setting.

We should also mention Laird's games model of (a fragment of) π [27], which accounts for *trace* (a.k.a. *may testing*) equivalence. Standard game models view strategies as *sets* of traces (with well-formedness conditions), so, as we have seen, lend themselves better to modelling trace equivalence. In a non-deterministic, yet not concurrent setting, Harmer and McCusker [18] resort to an explicit action for divergence, which allows them to recover a finer behavioural equivalence. We feel that the presheaf-based approach is more general.

Furthermore, recent work by Tsukada and Ong [41] adapts and extends some ideas of [23, 21] to nondeterministic, simply-typed λ-calculus.

Let us moreover mention less closely related work: Girard's *ludics* [16], Melliès's game semantics in string diagrams [33], Harmer et al.'s categorical combinatorics of innocence [18], McCusker et al.'s graphical foundation for schedules [31]. Finally, Hildebrandt's work [20] also uses sheaves, though for a quite different purpose.

1.6 Plan

We describe our notion of trace at length in Section 2. We then sketch the model produced by the machinery of playgrounds, and state our main result in Section 3. We then conclude in Section 4, with a brief sketch of the proof and some future directions.

2 Traces

In this section, we introduce our notion of trace, which is based on certain combinatorial objects, close in spirit to string diagrams. We first define these string diagrams, and then use them to define traces. Configurations are special, hypergraph-like string diagrams whose vertices represent agents and whose hyperedges represent channels. A perhaps surprising point is that actions are not just a binary relation between configurations, because we not only want to say *when* there is an action from one configuration to another, but also *how* this action is performed. This will be implemented by viewing actions from X to Y as *cospans* $Y \to M \leftarrow X$ in a certain category $\widehat{\mathbb{C}}$, whose objects we call higher-dimensional string diagrams for lack of a better term. The idea is that X and Y respectively are the initial and final configurations, and that M describes how one goes from X to Y. By combining such actions (by pushout), we get a bicategory $\mathbb{D}_v$ of configurations and traces.

2.1 String diagrams

The category $\widehat{\mathbb{C}}$ will be a category of presheaves over a base category, $\mathbb{C}$. Let us motivate the definition of $\mathbb{C}$ by recalling that (directed, multi) graphs may be seen as presheaves over the category with two objects $\star$ and $[1]$, and two non-identity morphisms $s, t \colon \star \to [1]$. Any such presheaf G represents the graph with vertices in $G(\star)$ and edges in $G[1]$, the source and target of any $e \in G[1]$ being respectively $G(s)(e)$ and $G(t)(e)$, or $e \cdot s$ and $e \cdot t$ for short. A way to visualise how such presheaves represent graphs is to compute their *categories of elements* [30]. Recall that the category of elements $\int G$ for a presheaf G over $\mathbb{C}$ has as objects pairs (c, x) with $c \in \mathbb{C}$ and $x \in G(c)$, and as morphisms $(c, x) \to (d, y)$ all morphisms $f \colon c \to d$ in $\mathbb{C}$ such that $y \cdot f = x$. This category admits a canonical functor π_G to $\mathbb{C}$, and G is the colimit of the composite $\int G \xrightarrow{\pi_G} \mathbb{C} \xrightarrow{\mathsf{y}} \hat{\mathbb{C}}$ with the Yoneda embedding. E.g., the category of elements for $\mathsf{y}[1]$ is the poset $(\star, s) \xrightarrow{s} ([1], id_{[1]}) \xleftarrow{t} (\star, t)$, which could be pictured as •——▶——•, where dots represent vertices, the triangle represents the edge, and links materialise the graph of $G(s)$ and $G(t)$, the convention being that t connects to the apex of the triangle. We thus recover some graphical intuition.

Our string diagrams will also be defined as particular presheaves over some base category $\mathbb{C}$. However, since we'll only be interested in *finite* structures, we restrict ourselves to the category $\widehat{\mathbb{C}}$ of presheaves of finite sets. In the case of graphs, presheaves of finite sets are graphs whose nodes and edges are identified by natural numbers. Such graphs are thus finite. In our case, the base category $\mathbb{C}$ is infinite, so presheaves of finite sets may represent infinite structures. However, our notion of trace will only involve finite ones.

Let us give the formal definition of $\mathbb{C}$ for reference. We advise to skip it on a first reading, as we then attempt to provide some graphical intuition.

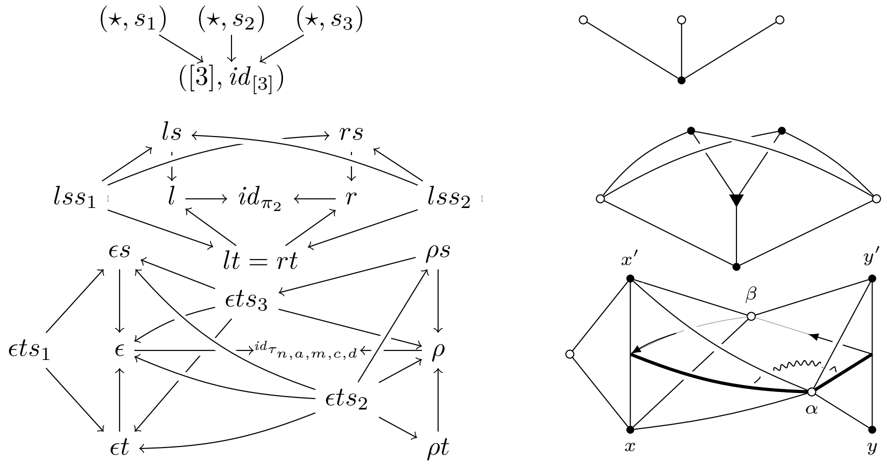


Figure 1 Categories of elements for $[3]$, π_2, and $\tau_{1,1,3,2,3}$, with graphical representation.

▶ **Definition 8.** Let $G_{\mathbb{C}}$ be the graph with, for all n, m, with $a, b \in n$ and $c, d \in m$:

- vertices $\star$, $[n]$, π^l_n, π^r_n, π_n, ν_n, $\heartsuit_n$, τ_n, $\iota_{n,a}$, $o_{n,a,b}$, and $\tau_{n,a,m,c,d}$;
- edges $s_1, ..., s_n : \star \to [n]$, plus, $\forall v \in \{\pi^l_n, \pi^r_n, \heartsuit_n, \tau_n, o_{n,a,b}\}$, edges $s, t : [n] \to v$;
- edges $[n] \xrightarrow{t} \nu_n \xleftarrow{s} [n+1]$ and $[n] \xrightarrow{t} \iota_{n,a} \xleftarrow{s} [n+1]$;
- edges $\pi^l_n \xrightarrow{l} \pi_n \xleftarrow{r} \pi^r_n$ and $\iota_{n,a} \xrightarrow{\rho} \tau_{n,a,m,c,d} \xleftarrow{\epsilon} o_{m,c,d}$.

Let $\mathbb{C}$ be the free category on $G_{\mathbb{C}}$, modulo the equations

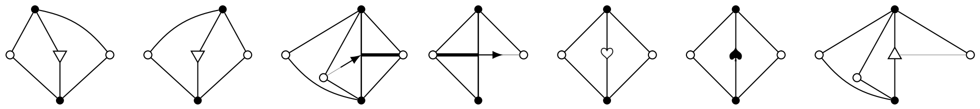
$$s \circ s_i = t \circ t_i \qquad l \circ t = r \circ t \qquad \rho \circ t \circ s_a = \epsilon \circ t \circ s_c \qquad \rho \circ s \circ s_{n+1} = \epsilon \circ s \circ s_d.$$

The first equation should be understood in $\mathbb{C}(\star, v)$ for all $n \in \mathbb{N}$, $i \in n$, and $v \in \cup_{a,b \in n}\{\pi^l_n, \pi^r_n, \heartsuit_n, \tau_n, \iota_{n,a}, o_{n,a,b}, \nu_n\}$. (This is rather elliptic: if v has the shape $\iota_{n,a}$ or ν_n, $s \circ s_i$ is really $\star \xrightarrow{s_i} [n+1] \xrightarrow{s} v$.) The second equation should be understood in $\mathbb{C}(\star, \pi_n)$ for all n, and the last two in $\mathbb{C}(\star, \tau_{n,a,m,c,d})$, for all n, m, $a \in n$, and $c, d \in m$.

Our category of string diagrams is the category of finite presheaves $\widehat{\mathbb{C}}$. To explain the design of $\mathbb{C}$, let us compute a few categories of elements. Let us start with an easy one, that of $[3] \in \mathbb{C}$ (we implicitly identify any $c \in \mathbb{C}$ with yc). An easy computation shows that it is the poset pictured in the top left part of Figure 1. We think of it as a configuration with one agent $([3], id_{[3]})$ connected to three channels, and draw it as in the top right part, where the bullet represents the agent, and circles represent channels. In the presheaf, elements over $[3]$ represent ternary agents, while elements over $\star$ represent channels. *Configurations* are finite presheaves empty except perhaps on $\star$ and $[n]$'s. Other objects will represent actions. A *morphism of configurations* is a morphism between presheaves which is injective except perhaps on channels. The intuition for a morphism $X \to Y$ between configurations is thus that X embeds into Y, possibly identifying some channels.

▶ **Definition 9.** Configurations and morphisms between them form a category $\mathbb{D}_h$.

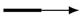
A more difficult category of elements is that of π_2. It is the poset generated by the left-hand graph in the second row of Figure 1 (omitting base objects for conciseness). We think of it as a binary agent (lt) forking into two agents (ls and rs), and draw it as on the right. The graphical convention is that a black triangle stands for the presence of id_{π_2}, l,



$$\begin{tikzcd}[p] \arrow[d] & {[p]} \arrow[d] & {[m]} \arrow[d] & {[n+1]} \arrow[d] & {[p]} \arrow[d] & {[p]} \arrow[d] & {[p+1]} \arrow[d] \\ \pi^l_p & \pi^r_p & o_{m,c,d} & \iota_{n,a} & \heartsuit_p & \tau_p & \nu_p \\ {[p]} \arrow[u] & {[p]} \arrow[u] & {[m]} \arrow[u] & {[n]} \arrow[u] & {[p]} \arrow[u] & {[p]} \arrow[u] & {[p]} \arrow[u]\end{tikzcd}$$

Figure 2 Pictures and corresponding cospans for π^l_p, π^r_p, $o_{m,c,d}$, $\iota_{n,a}$, $\heartsuit_p$, τ_p, and ν_p.

and r. Below, we represent just l as a white triangle with only a left-hand branch, and symmetrically for r. Furthermore, in all our pictures, time flows ‘upwards’.

Another category of elements, characteristic of the π-calculus, is the one for synchronisation $\tau_{n,a,m,c,d}$. The case $(n, a, m, c, d) = (1, 1, 3, 2, 3)$ is the poset generated by the graph on the bottom left of Figure 1, which we will draw as on the right. The left-hand ternary agent x outputs its 3rd channel, here β, on its 2nd channel, here α. The right-hand unary agent y receives the sent channel on its 1st channel, here α. Both agents have two occurrences, one before and one after the action, respectively marked as x/x' and y/y'. Both x and x' are ternary here, while y is unary and y', having gained knowledge of β, is binary. There are actually three actions here, in the sense that there are three higher-dimensional objects. The first is the output action ϵ from x to x', graphically represented as the middle point of (intended to evoke the point where β enters channel α). The second is the input action ρ from y to y', graphically represented as the middle point of  (where β exits channel α). The third action is the synchronisation itself, which ‘glues’ the other two together, as represented by the squiggly line.

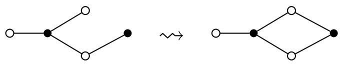
We leave the computation of other categories of elements as an exercise to the reader. The remaining string diagrams are depicted in the top row of Figure 2, for $p = 2$ and $(n, a, m, c, d) = (1, 1, 3, 2, 3)$.

The first two are *views*, in the game semantical sense, of the fork action π_2 explained above. The next two, $o_{m,c,d}$ (for ‘output’) and $\iota_{n,a}$ (for ‘input’), respectively are views for the sender and receiver in a synchronisation action. The τ_p action is a silent, dummy action as standard in the π-calculus. The $\heartsuit_n$ action is a special ‘tick’ action used for defining fair testing equivalence. The last one is a channel creation action.

2.2 From string diagrams to actions

In the previous section, we have defined our category of string diagrams as $\widehat{\mathbb{C}}$, and provided some graphical intuition on its objects. The next step is to construct a bicategory whose objects are configurations, and whose morphisms represent traces. We start in this section by defining in which sense higher-dimensional objects of $\mathbb{C}$ represent actions, and continue in the next one by explaining how to compose actions to form traces. Actions are defined in two stages: *seeds*, first, give the local forms of actions, which are then defined by embedding seeds into bigger configurations.

To start with, until now, our string diagrams contain no information about the ‘flow of time’, although we mentioned it informally in the previous section. To add this information, for each string diagram M representing an action, we define its initial and final configurations, say X and Y, and view the whole action as a cospan $Y \xrightarrow{s} M \xleftarrow{t} X$.



We have taken care, in drawing our pictures before, of placing initial configurations at the bottom, and final configurations at the top. So, e.g., the initial and final configurations for the synchronisation action are pictured above and they map into (the representable presheaf over) $\tau_{1,1,3,2,3}$ in the obvious ways, yielding the cospan $Y \xrightarrow{s} M \xleftarrow{t} X$.

$$\begin{tikzcd}
{[p] \mid [p]} \arrow[d] & {[m]\ {}_{c,d}|_{a,n+1}\ [n+1]} \arrow[d] \\
\pi_p & \tau_{n,a,m,c,d} \\
{[p]} \arrow[u] & {[m]\ {}_{c}|_{a}\ [n]} \arrow[u]
\end{tikzcd}$$

We leave it to the reader to define, based on the above pictures, the expected cospans for forking and synchronisation as on the right, plus the remaining ones specified in the bottom row of Figure 2 (where again $p = 2$ and $(n, a, m, c, d) = (1, 1, 3, 2, 3)$). Initial configurations are at the bottom, and we denote by $[m]\ {}_{a_1,\ldots,a_p}|_{c_1,\ldots,c_p}\ [n]$ the configuration consisting of an m-ary agent x and an n-ary agent y, quotiented by the equations $x \cdot s_{a_k} = y \cdot s_{c_k}$ for all $k \in p$. When both lists are empty, by convention, $m = n$ and the agents share all channels in order.

▶ **Definition 10.** These cospans are called *seeds.*

We now define actions from seeds by embedding the latter into bigger configurations. E.g., we allow a fork action to occur in a configuration with more than one agent.

▶ **Definition 11.** The *interface* I_X of a seed $Y \xrightarrow{s} M \xleftarrow{t} X$ denotes the configuration consisting only of the channels of the initial configuration X.

$$\begin{tikzcd}
 & Y \arrow[d] \arrow[rr] & & Y' \arrow[d, dashed] \\
 & M \arrow[rr] & & M' \\
I_X \arrow[ruu] \arrow[ru] \arrow[rr] \arrow[rd] & & Z \arrow[ruu] \arrow[ru] \arrow[rd] & \\
 & X \arrow[uu] \arrow[rr] & & X' \arrow[uu, dashed]
\end{tikzcd}$$

Since channels occurring in the initial configuration remain in the final one, we have for each seed a cone from I_X to the seed. For any morphism of positions $I_X \to Z$, pushing the cone along $I_X \to Z$ using the universal property of pushout as on the right yields a new cospan, say $Y' \to M' \leftarrow X'$.

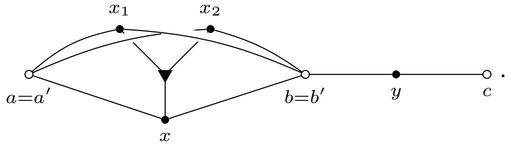
▶ **Definition 12.** Let *actions* be all such pushouts of seeds.

Intuitively, taking pushouts glues string diagrams together. Let us do a few examples.

▶ **Example 13.** The seed $[2] \mid [2] \xrightarrow{[ls,rs]} \pi_2 \xleftarrow{lt} [2]$ has as interface the presheaf $I_{[2]} = \star + \star$, consisting of two channels, say a and b. Consider the configuration $[2] + \star$ consisting of an agent y connected to two channels b' and c, plus an additional channel a'. Further consider the map $h\colon I_{[2]} \to [2] + \star$ defined by $a \mapsto a'$ and $b \mapsto b'$. The pushout

$$\begin{tikzcd}
I_{[2]} \arrow[r] \arrow[d] & {[2] + \star} \arrow[d] \\
\pi_2 \arrow[r] \arrow[ru, phantom, "\ulcorner" very near end] & M'
\end{tikzcd}$$

is



The meaning of such an action is that x forks while y is passive.

▶ **Example 14.** Because we push along *initial* channels, the interface of a seed may not contain all involved channels. E.g., in an input action (not part of any synchronisation), the received channel cannot be part of the initial configuration.

2.3 From actions to traces

Having defined actions, we now define their composition to yield our bicategory $\mathbb{D}_v$ of configurations and traces. Consider $\mathsf{Cospan}(\widehat{\mathbb{C}})$, the bicategory which has as objects all presheaves of finite sets on $\mathbb{C}$, as morphisms $X \to Y$ all cospans $X \to U \leftarrow Y$, and obvious 2-cells. Composition is given by pushout, and hence is not strictly associative.

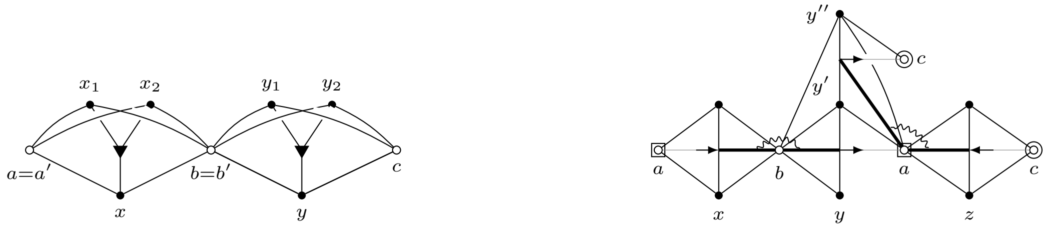


Figure 3 Example traces.

▶ **Notation 15.** *By convention, the initial configuration is the* target *of the morphism in* $\mathsf{Cospan}(\widehat{\mathbb{C}})$. *We denote morphisms in* $\mathsf{Cospan}(\widehat{\mathbb{C}})$ *with special arrows* $X \mathrel{-\!\!\bullet\!\!\!\rightarrow} Y$*; composition and identities are denoted with* $\bullet$ *and* $id^{\bullet}$.

▶ **Definition 16.** A *trace* is a finite, possibly empty composite of actions in $\mathsf{Cospan}(\widehat{\mathbb{C}})$. Let $\mathbb{D}_v$ denote the locally full subbicategory of $\mathsf{Cospan}(\widehat{\mathbb{C}})$ with configurations as objects and traces as morphisms.

Thus, arrows $X \to Y$ in $\mathbb{D}_h$ denote embeddings of X into Y (up to identification of channels), whereas arrows $Y \mathrel{-\!\!\bullet\!\!\!\rightarrow} X$ in $\mathbb{D}_v$ denote traces with X initial and Y final. Intuitively, composition in $\mathbb{D}_v$ glues string diagrams on top of each other, which yields a truly concurrent notion of trace: the only information retained in a trace about the order of occurrence of actions is their causal dependencies.

▶ **Example 17.** Composing the action of Example 13 with a forking action by y yields the first string diagram of Figure 3, which shows that the ordering between remote actions is irrelevant. To illustrate how composition retains causal dependencies between actions, consider the second string diagram. It is unfolded for readability: one should identify both framed nodes, resp. both circled ones. In the initial configuration, there are channels a, b, and c, and three agents $x(a, b)$, $y(b)$, and $z(a, c)$ (channels known to each agent are in parentheses). In a first action, x sends a on b, and y receives it. In a second action, z sends c on a, and the avatar y' of y receives it. The second action is enabled by the first, by which y gains knowledge of a.

3 Strategies, behaviours, and semantic fair testing

3.1 Strategies and behaviours

We now investigate notions of strategies. As announced in the introduction, we define a category $\mathbb{T}(X)$ combining prefix ordering and isomorphism of traces: $\mathbb{T}(X)$ has traces $u \colon Y \mathrel{-\!\!\bullet\!\!\!\rightarrow} X$ as objects, and as morphisms $u \to u'$ all pairs (w, α) with $w \colon Y' \mathrel{-\!\!\bullet\!\!\!\rightarrow} Y$ and α an isomorphism $u \bullet w \to u'$ in the hom-category $\mathbb{D}_v(Y', X)$, as on the right[1]. Thus, u' is an extension of u by w.

$$\begin{tikzcd} & Y' \arrow[dl, "w"'] \arrow[dd, bend left, "u'"] \\ Y \arrow[dr, "u"'] \arrow[r, phantom, "\overset{\cong}{\underset{\alpha}{\Rightarrow}}"] & {} \\ & X \end{tikzcd}$$

▶ **Definition 18.** Let the category of *(naive) strategies* on X be $\widehat{\mathbb{T}(X)}$.

Strategies do not yield a satisfactory model for π:

[1] There is a small problem, however: morphisms should only describe how u maps to u', not w. We actually quotient them out to rectify this.

▶ **Example 19.** Consider the configuration X with three agents x, y, z sharing a channel a, and the following traces on it: in $u_{x,y}$, x sends a on a, and y receives it; in $u_{x,z}$, x sends a on a, and z receives it; in i_z, z inputs on a. One may define a strategy S mapping $u_{x,y}$ and i_z to a singleton, and $u_{x,z}$ to $\emptyset$. Because $u_{x,y}$ is accepted, x accepts to send a on a; and because i_z is accepted, z accepts to input on a. The problem is that S rejecting $u_{x,z}$ roughly amounts to x refusing to synchronise with z, or conversely.

We want to rule out this kind of strategy from our model, by adapting the idea of innocence. We start by extending $\mathbb{T}(X)$ with objects representing traces on sub-configurations of X. For this, we consider the following category $\mathbb{T}_X$. It has as objects pairs (u, h) of a trace $u\colon Z \multimap Y$ and a morphism $h\colon Y \to X$ in $\mathbb{D}_h$. A morphism $(u, h) \to (u', h')$ consists of a trace $w\colon T \multimap Z$ and a triple (s, k, r) making the diagram on the right commute[1].

$$\begin{tikzcd}
T \arrow[r, "s"] \arrow[d] & Z' \arrow[d] \\
u \bullet w \arrow[r, "k"] & u' \\
Y \arrow[u] \arrow[r, "r"] \arrow[dr, "h"'] & Y' \arrow[u] \arrow[d, "h'"] \\
 & X
\end{tikzcd}$$

▶ **Example 20.** We adopt the convention of picturing the above diagram for morphisms as the left-hand diagram below: Now recalling the right-hand trace of Figure 3, say $u\colon Y \multimap X$, y's first action is an input on its unique channel b. This yields a trace $\iota_{1,1}\colon [2] \multimap [1]$. There is a morphism $(\iota_{1,1}, y) \to (u, id_X)$ in $\mathbb{T}_X$, pictured as the right-hand diagram, which we think of as an occurrence of the trace $\iota_{1,1}$ in u. Thus, morphisms in $\mathbb{T}_X$ account both for prefix inclusion and for 'spatial' inclusion, i.e., inclusion of a trace into some other trace on a larger configuration.

$$\begin{tikzcd}
T \arrow[r, "s"] \arrow[d, "w"'] & Z' \arrow[dd, "u'"] \\
Z \arrow[d, "u"'] \arrow[r, phantom, "\Longrightarrow"] & \\
Y \arrow[r, "r"] \arrow[dr, "h"'] & Y' \arrow[d, "h'"] \\
 & X
\end{tikzcd}
\qquad
\begin{tikzcd}
{[3]} \arrow[r, "y''"] \arrow[d, "\iota_{2,2}"'] & Y \arrow[dd, "u"] \\
{[2]} \arrow[d, "\iota_{1,1}"'] \arrow[r, phantom, "\Longrightarrow"] & \\
{[1]} \arrow[r, "y"] \arrow[dr, "y"'] & X \arrow[d, equal] \\
 & X.
\end{tikzcd}$$

We now define *views* within $\mathbb{T}_X$:

▶ **Definition 21.** Let *basic* seeds be all seeds of any shape among $\iota_{n,a}$, $o_{n,a,b}$, ν_n, $\heartsuit_n$, τ_n, π^l_n, and π^r_n, for $a, b \in n$. *Views* are (possibly empty) composites of basic seeds in $\mathbb{D}_v$. Let $\mathbb{V}_X$ denote the full subcategory of $\mathbb{T}_X$ spanning pairs (u, h) where u is a view.

Intuitively, basic seeds follow exactly one agent through an action. An object of $\mathbb{V}_X$ consists of a view, say $v\colon [n'] \multimap [n]$, plus a morphism $h\colon [n] \to X$ in $\mathbb{D}_h$, which by Yoneda is just an agent of X. So an object of $\mathbb{V}_X$ is just an agent of X and a view from it.

▶ **Definition 22.** The inclusion $\mathbb{V}_X \hookrightarrow \mathbb{T}_X$ induces a Grothendieck topology, for which a family $(u_i \xrightarrow{\alpha_i} u)_{i \in I}$ of morphisms to some trace u is *covering* iff it contains all morphisms from views into u. Let the category $\mathsf{S}_X \hookrightarrow \widehat{\mathbb{T}_X}$ of *innocent* strategies be the category of sheaves of finite sets for this topology. Let the category B_X of *behaviours* over X be $\widehat{\mathbb{V}_X}$.

As promised, S_X and B_X are equivalent. We obtain the innocent strategy S_B associated to a behaviour $B \in \mathsf{B}_X$, by taking its *right Kan extension* [29] along the inclusion $\mathsf{j}^{op}\colon \mathbb{V}^{op}_X \hookrightarrow \mathbb{T}^{op}_X$, as on the right. Explicitly, using standard results, we obtain the *end* $S_B(u, h) = \int_{(v,x) \in \mathbb{V}_X} B(v, x)^{\mathbb{T}_X((v,x),(u,h))}$, which is a kind of generalised product. In the boolean setting (functors to 2), this end reduces to a conjunction $\bigwedge_{\{(v,x) \in \mathbb{V}_X \mid \exists \alpha\colon (v,x) \to (u,h)\}} B(v, x)$, demanding precisely that all views of u are accepted by B. In the general case, the intuition is that a way of accepting u for S_B is a compatible family of ways of accepting the views of u for B. Existence of the right Kan extension is proved in the general case in [22, Lemma 4.34], and follows from the general fact that the considered limits are essentially finite. The forgetful functor $\mathcal{U}$ to naive strategies is then given by restricting along $\mathbb{T}(X)^{op} \hookrightarrow \mathbb{T}^{op}_X$ as above right. Some local information may be forgotten by $\mathcal{U}$, which is neither injective on objects, nor full, nor faithful. E.g., if two

$$\begin{tikzcd}
\mathbb{V}^{op}_X \arrow[r, hook, "\mathsf{j}^{op}"] \arrow[dr, "B"'] & \mathbb{T}^{op}_X \arrow[d, "S_B"] & \mathbb{T}(X)^{op} \arrow[l, hook'] \arrow[dl, "\mathcal{U}(S_B)"] \\
 & \mathsf{set} &
\end{tikzcd}$$

behaviours differ, but are both empty on the views of some agent, then both are mapped to the empty naive strategy.

▶ **Example 23.** Recalling X and S from Example 19, let us show that for any $B \in \mathsf{B}_X$, the associated strategy $\mathcal{U}(S_B) \in \widehat{\mathbb{T}(X)}$ cannot be S. Indeed, assuming it is, then because S accepts $u_{x,y}$ and i_z, B accepts the following views: (1) i_z, (2) o_x, in which x sends a on a (without any matching input), (3) i_y, in which y inputs on a, and (4) all identity views on x, y, and z. But then $\mathcal{U}(S_B)$ accepts both $u_{x,y}$ and $u_{x,z}$, because B accepts all views mapping into them.

3.2 Semantic fair testing

We now define our semantic analogue of fair testing equivalence, sketch our translation from π, and state our main result. Semantic fair testing rests on two main ingredients: a notion of *closed-world* trace, and an analogue of *parallel composition* in game semantics.

$$\begin{tikzcd}
\mathbb{V}^{op}_{X_1} \arrow[r, hook] \arrow[dr, "B_1"'] & \mathbb{V}^{op}_{X} \arrow[d, dashed, "{[B_1,B_2]}"] & \mathbb{V}^{op}_{X_2} \arrow[l, hook'] \arrow[dl, "B_2"] \\
 & \mathsf{set} &
\end{tikzcd}$$

The intuitive purpose of parallel composition is to let strategies interact. If we partition the agents of a configuration X into two teams, we obtain two subconfigurations $X_1 \hookrightarrow X \hookleftarrow X_2$, each agent of X belonging to X_1 or X_2 according to its team. The crucial fact is that the category $\mathbb{V}_X$ of views on X is isomorphic to the coproduct category $\mathbb{V}_{X_1} + \mathbb{V}_{X_2}$. Parallel composition of any $B_1 \in \mathsf{B}_{X_1}$ and $B_2 \in \mathsf{B}_{X_2}$ is then simply given by copairing $[B_1, B_2]$ as above.

We now describe closed-world actions and traces, which are then used as a criterion for success of tests. *Closed-world* actions are those which do not involve interaction with the environment, i.e., formally, pushouts of a seed of any shape among ν_n,τ_n,$\heartsuit_n$,π_n, and $\tau_{n,a,m,c,d}$. A trace is *closed-world* when it is a composite of closed-world actions. Let $\mathbb{W}(X) \xhookrightarrow{\mathfrak{i}} \mathbb{T}(X)$ denote the full subcategory of $\mathbb{T}(X)$ consisting of closed-world traces, and let the category of *closed-world strategies* be $\widehat{\mathbb{W}(X)}$. Further, denote by $B \mapsto \overline{B}$ the composite functor $\widehat{\mathbb{V}_X} \to \widehat{\mathbb{T}(X)} \xrightarrow{\Delta_{\mathfrak{i}^{op}}} \widehat{\mathbb{W}(X)}$, where $\Delta_{\mathfrak{i}^{op}}$ denotes restriction along $\mathfrak{i}^{op}$.

A closed-world trace is *successful* when it contains a $\heartsuit$ action, and *unsuccessful* otherwise. A state $\sigma \in S(u)$ of a strategy $S \in \widehat{\mathbb{W}(Z)}$ over a closed-world trace $u\colon Z' \mathrel{-\!\!\bullet\!\!\to} Z$ is successful iff u is. Define $\perp\!\!\!\perp_Z$ as the set of closed-world strategies $S \in \widehat{\mathbb{W}(Z)}$ such that any unsuccessful closed-world state admits a successful extension, i.e. $S \in \perp\!\!\!\perp_Z$ iff for all unsuccessful $u \in \mathbb{W}(Z)$ and $\sigma \in S(u)$, there exists a successful $u' \in \mathbb{W}(Z)$, a morphism $f\colon u \to u'$, and a state $\sigma' \in S(u')$ such that $\sigma' \cdot f = \sigma$. Finally, in order to compare behaviours for semantic fair testing equivalence, we specify what a test is for a given behaviour $B \in \mathsf{B}_X$. A *test* consists of a configuration Y and a behaviour $T \in \mathsf{B}_Y$. The behaviour B then *should pass* the test (Y, T) iff $I_X = I_Y$ and $\overline{[B,T]} \in \perp\!\!\!\perp_Z$, where I_X consists of all channels of X (recall Definition 11) and Z is the pushout $X +_{I_X} Y$ (X and Y thus form two teams on Z). At last, we define *semantic fair testing equivalence*, for any $B \in \mathsf{B}_X$ and $B' \in \mathsf{B}_{X'}$:

▶ **Definition 24.** Let $B \sim_f B'$ iff B and B' should pass the same tests.

3.3 Intensional full abstraction

We now sketch our translation from π-calculus processes to behaviours and state our main result. First, we consider processes to be infinite terms as generated by the grammar

$$P, Q ::= \textstyle\sum_{i \in n} G_i \quad | \quad (P \mid Q) \qquad\qquad G ::= \bar{a}\langle b \rangle.P \mid a(b).P \mid \nu a.P \mid \tau.P \mid \heartsuit.P,$$

up to renaming of bound variables as usual. Such a coinductive definition requires some care [12]: notably, processes come equipped with their finite set Γ of free channels, which we denote by $\Gamma \vdash P$. In order to translate processes to behaviours, we denote the coproduct in B_X by $\oplus$ (which is the pointwise coproduct of presheaves). Furthermore, let us denote by $\mathbb{B}_{[n]}$ the set of basic seeds $b \colon [n_b] \mathrel{\multimap\!\!\!\to} [n]$ from $[n]$. For any family $(B_b)_{b \in \mathbb{B}_{[n]}}$ of behaviours, where each B_b is a behaviour over $[n_b]$, we denote by $\langle (B_b)_{b \in \mathbb{B}_{[n]}} \rangle$ the behaviour B' over $[n]$ such that $B'(id^{\bullet}_{[n]}, id_{[n]}) = 1$, and for any view $b \bullet v$, $B'(b \bullet v, id_{[n]}) = B_b(v, id_{[n_b]})$. Armed with this notation, we coinductively map processes with free channels in $\{1, \ldots, \Gamma\}$ for some $\Gamma \in \mathbb{N}$ to behaviours on $[\Gamma]$ like so:

$$[\![\Gamma \vdash \textstyle\sum_i \alpha_i . P_i]\!] = \langle b \mapsto \oplus_{\{i \mid [\![\alpha_i]\!] = b\}} [\![\Gamma \cdot \alpha_i \vdash P_i]\!] \rangle \qquad [\![\Gamma \vdash P \mid Q]\!] = \left\langle \begin{array}{ccc} \pi^l_\Gamma & \mapsto & [\![\Gamma \vdash P]\!] \\ \pi^r_\Gamma & \mapsto & [\![\Gamma \vdash Q]\!] \end{array} \right\rangle,$$

where (1) $[\![\bar{a}\langle b \rangle]\!] = o_{\Gamma,a,b}$, $[\![a(b)]\!] = \iota_{\Gamma,a}$, $[\![\nu a]\!] = \nu_\Gamma$, $[\![\heartsuit]\!] = \heartsuit_\Gamma$, and $[\![\tau]\!] = \tau_\Gamma$, (2) all unmentioned basic seeds are mapped to the everywhere empty behaviour $\emptyset$, (3) $\Gamma \cdot \alpha_i$ denotes $\Gamma + 1$ when α_i is an input or a channel creation[2], and Γ otherwise. E.g., we have $[\![\Gamma \vdash a(b).P + a(b).Q]\!] = \langle \iota_{\Gamma,a} \mapsto [\![\Gamma + 1 \vdash P]\!] \oplus [\![\Gamma + 1 \vdash Q]\!] \rangle$. We then define fair testing equivalence $\sim^{Pi}_f$ for π-calculus processes as in [7]: let $\bot^{Pi}$ denote the set of processes P such that for all $P \Rightarrow P'$ there exists $P' \Rightarrow P'' \xrightarrow{\heartsuit} P'''$, and, given any two processes $\Gamma \vdash P$ and $\Gamma \vdash Q$, let $P \sim^{Pi}_f Q$ iff for all $\Gamma \vdash T$ we have $(P \mid T \in \bot^{Pi}) \Leftrightarrow (Q \mid T \in \bot^{Pi})$. Finally, our main result is:

▶ **Theorem 25.**

1. *For all* P, Q, $(\Gamma \vdash P) \sim^{Pi}_f (\Gamma \vdash Q)$ *iff* $[\![\Gamma \vdash P]\!] \sim_f [\![\Gamma \vdash Q]\!]$.
2. *For all* B *over* $[\Gamma]$, *there exists a process* $\Gamma \vdash P$ *such that* $[\![\Gamma \vdash P]\!] \sim_f B$.

4 Conclusion and future work

We have described our notion of trace and the induced model of π. We then have stated our main theorem. In our online long version [12], the interested reader may find the proof that our traces organise into a playground [22], and the proof of Theorem 25. For lack of space, we cannot give any detail. Still, we sketch the latter.

The idea is to reduce semantic fair testing equivalence to fair testing equivalence in the standard sense for some *ad hoc* LTS $\mathcal{S}$. We then single out a particular quotient $\mathcal{M}$ of $\mathcal{S}$, which admits a syntactic description very close to Berry and Boudol's *chemical abstract machine* [2], though with a kind of persistent explicit substitutions. Elements of $\mathcal{M}$ thus roughly consist of finite multisets of *molecules*. Multisets are here thought of as *chemical soups*, in which synchronisation is viewed as interaction between compatible molecules. In order to simplify matters, we also work with a chemical abstract machine presentation of the π-calculus. We then define a candidate 'pseudo-inverse' ζ to the translation map $[\![-]\!]$. These are maps between molecules for π and molecules for $\mathcal{M}$, which extend straightforwardly to chemical soups. We finally design a relation between π-calculus soups and $\mathcal{M}$ soups, which modularly allows π-calculus processes to correspond to their translation, and $\mathcal{M}$-molecules to correspond to their image under ζ. We are then able to show that this relation is a weak bisimulation which straightforwardly entails that $[\![-]\!]$ both preserves and reflects fair testing equivalence and is surjective up to fair testing equivalence, i.e., is intensionally fully-abstract.

Regarding future work, we of course plan to extend our approach to more complex calculi, e.g., calculi with passivation or functional calculi, and eventually consider some full-fledged

[2] W.l.o.g., we choose representatives of processes so that if $\Gamma \vdash a(b).P$ or $\Gamma \vdash \nu b.P$, then $b = (\Gamma + 1)$.

functional language with concurrency primitives. Furthermore, we consider generalising our technique for constructing playgrounds and applying them, e.g., to graph rewriting, error diagnostics, or efficient machine representation of reversible π-calculus processes.

References

1 Samson Abramsky, Radha Jagadeesan, and Pasquale Malacaria. Full abstraction for PCF. *Inf. Comput.*, 163(2):409–470, 2000.

2 Gérard Berry and Gérard Boudol. The chemical abstract machine. In *POPL*, pages 81–94, 1990.

3 Filippo Bonchi, Marcello M. Bonsangue, Jan J. M. M. Rutten, and Alexandra Silva. Deriving syntax and axioms for quantitative regular behaviours. In *CONCUR*, volume 5710 of *LNCS*, pages 146–162. Springer, 2009.

4 Michele Boreale and Davide Sangiorgi. A fully abstract semantics for causality in the π-calculus. *Acta Inf.*, 35(5):353–400, 1998.

5 Ed Brinksma, Arend Rensink, and Walter Vogler. Fair testing. In *CONCUR*, volume 962 of *LNCS*, pages 313–327. Springer, 1995.

6 Nadia Busi and Roberto Gorrieri. Distributed semantics for the π-calculus based on Petri nets with inhibitor arcs. *J. Log. Algebr. Program.*, 78(3):138–162, 2009.

7 Diletta Cacciagrano, Flavio Corradini, and Catuscia Palamidessi. Explicit fairness in testing semantics. *LMCS*, 5(2), 2009.

8 Simon Castellan, Pierre Clairambault, and Glynn Winskel. The parallel intensionally fully abstract games model of pcf. In *LICS*. IEEE, 2015.

9 Gian Luca Cattani and Peter Sewell. Models for name-passing processes: Interleaving and causal. In *LICS*, pages 322–333. IEEE, 2000.

10 Silvia Crafa, Daniele Varacca, and Nobuko Yoshida. Event structure semantics of parallel extrusion in the pi-calculus. In *FoSSaCS*, volume 7213 of *LNCS*, pages 225–239. Springer, 2012.

11 Rocco De Nicola and Matthew Hennessy. Testing equivalences for processes. *TCS*, 34:83–133, 1984.

12 Clovis Eberhart, Tom Hirschowitz, and Thomas Seiller. A πlayground. `http://lama.univ-smb.fr/~hirschowitz/papers/pilayground.pdf`, 2015.

13 Joost Engelfriet. A multiset semantics for the pi-calculus with replication. *TCS*, 153(1&2):65–94, 1996.

14 Marcelo P. Fiore, Eugenio Moggi, and Davide Sangiorgi. A fully-abstract model for the pi-calculus (extended abstract). In *LICS*, pages 43–54. IEEE, 1996.

15 Peter Freyd and G. M. Kelly. Categories of continuous functors, I. *JPAA*, 2:169–191, 1972.

16 Jean-Yves Girard. Locus solum: From the rules of logic to the logic of rules. *MSCS*, 11(3):301–506, 2001.

17 Gregor Gößler, Daniel Le Métayer, and Jean-Baptiste Raclet. Causality analysis in contract violation. In *Runtime Verification*, volume 6418 of *LNCS*, pages 270–284. Springer, 2010.

18 Russell Harmer, Martin Hyland, and Paul-André Melliès. Categorical combinatorics for innocent strategies. In *LICS*, pages 379–388. IEEE, 2007.

19 Matthew Hennessy. A fully abstract denotational semantics for the π-calculus. *TCS*, 278(1-2):53–89, 2002.

20 Thomas T. Hildebrandt. Towards categorical models for fairness: fully abstract presheaf semantics of SCCS with finite delay. *TCS*, 294(1/2):151–181, 2003.

21 Tom Hirschowitz. Full abstraction for fair testing in CCS. In *CALCO*, volume 8089 of *LNCS*, pages 175–190. Springer, 2013.

22 Tom Hirschowitz. Full abstraction for fair testing in CCS (expanded version). *LMCS*, 10(4), 2014.

23 Tom Hirschowitz and Damien Pous. Innocent strategies as presheaves and interactive equivalences for CCS. In *ICE*, pages 2–24, 2011.

24 J. M. E. Hyland and C.-H. Luke Ong. On full abstraction for PCF: I, II, and III. *Inf. Comput.*, 163(2):285–408, 2000.

25 Bart Jacobs. *Categorical Logic and Type Theory.* Number 141 in Studies in Logic and the Foundations of Mathematics. North Holland, Amsterdam, 1999.

26 André Joyal, Mogens Nielsen, and Glynn Winskel. Bisimulation and open maps. In *LICS*, pages 418–427. IEEE, 1993.

27 James Laird. Game semantics for higher-order concurrency. In *FSTTCS*, volume 4337 of *LNCS*, pages 417–428. Springer, 2006.

28 Saunders Mac Lane. Duality for groups. *Bull. AMS*, 56, 1950.

29 Saunders Mac Lane. *Categories for the Working Mathematician.* Number 5 in Graduate Texts in Mathematics. Springer, 2nd edition, 1998.

30 Saunders Mac Lane and Ieke Moerdijk. *Sheaves in Geometry and Logic: A First Introduction to Topos Theory.* Universitext. Springer, 1992.

31 Guy McCusker, John Power, and Cai Wingfield. A graphical foundation for schedules. *ENTCS*, 286:273–289, 2012.

32 Paul-André Melliès. Asynchronous games 4: A fully complete model of propositional linear logic. In *LICS*, pages 386–395. IEEE, 2005.

33 Paul-André Melliès. Game semantics in string diagrams. In *LICS*, pages 481–490. IEEE, 2012.

34 Ugo Montanari and Marco Pistore. Concurrent semantics for the π-calculus. *ENTCS*, 1:411–429, 1995.

35 V. Natarajan and Rance Cleaveland. Divergence and fair testing. In *ICALP*, volume 944 of *LNCS*, pages 648–659. Springer, 1995.

36 Hanno Nickau. Hereditarily sequential functionals. In *LFCS*, volume 813 of *LNCS*, pages 253–264. Springer, 1994.

37 M. Nielsen, G. Plotkin, and G. Winskel. Event structures and domains, part 1. *TCS*, 13:65–108, 1981.

38 Arend Rensink and Walter Vogler. Fair testing. *Inf. Comput.*, 205(2):125–198, 2007.

39 Silvain Rideau and Glynn Winskel. Concurrent strategies. In *LICS*, pages 409–418. IEEE, 2011.

40 Ian Stark. A fully abstract domain model for the π-calculus. In *LICS*, pages 36–42. IEEE, 1996.

41 Takeshi Tsukada and C.-H. Luke Ong. Nondeterminism in game semantics via sheaves. In *LICS*. IEEE, 2015.

42 Glynn Winskel. Event structure semantics for CCS and related languages. In Mogens Nielsen and Erik Meineche Schmidt, editors, *ICALP*, volume 140 of *LNCS*, pages 561–576. Springer, 1982.

43 Glynn Winskel. Strategies as profunctors. In *FoSSaCS*, volume 7794 of *LNCS*, pages 418–433. Springer, 2013.

Partial Higher-dimensional Automata

Uli Fahrenberg and Axel Legay

INRIA/IRISA, Campus de Beaulieu, 35042 Rennes CEDEX, France

Abstract

We propose a generalization of higher-dimensional automata, partial HDA. Unlike HDA, and also extending event structures and Petri nets, partial HDA can model phenomena such as priorities or the disabling of an event by another event. Using open maps and unfoldings, we introduce a natural notion of (higher-dimensional) bisimilarity for partial HDA and relate it to history-preserving bisimilarity and split bisimilarity. Higher-dimensional bisimilarity has a game characterization and is decidable in polynomial time.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases higher-dimensional automata, bisimulation

Digital Object Identifier 10.4230/LIPIcs.CALCO.2015.101

1 Introduction

Higher-dimensional automata (HDA) is a formalism for modeling and reasoning about behavior of concurrent systems. Like Petri nets [22], event structures [20], configuration structures [32], asynchronous transition systems [1, 27] and other similar formalisms, it is *non-interleaving* in the sense that it differentiates between concurrent and interleaving events; using CCS notation [19], $a|b \neq a.b + b.a$.

Introduced by Pratt [23] and van Glabbeek [29] in 1991 for the purpose of a *geometric* interpretation to the theory of concurrency, it has since been shown by van Glabbeek [30] that HDA provide a generalization (up to *history-preserving bisimilarity*) to "the main models of concurrency proposed in the literature" [30], including the ones mentioned above. Hence HDA are useful as a tool for comparing and relating different models, and also as a modeling formalism by themselves.

HDA are geometric in the sense that they are similar to the *simplicial complexes* used in algebraic topology, and research on HDA has drawn on tools and methods from geometry and topology such as homotopy [10, 8, 5, 9], homology [14], and model categories [12, 11], see also the surveys [13, 15].

Motivated by some examples of concurrent systems which *cannot* be modeled by HDA, we propose here an extension of the formalism, called *partial* or *incomplete* HDA. Intuitively, these are HDA in which some parts may be missing; transitions which do not have an end state, squares which miss parts of their boundary, etc. We will show that these can be used to model phenomena such as priorities and the disabling of events by other events.

We show that partial HDA admit a natural notion of bisimilarity, defined categorically through open maps in the spirit of Joyal, Nielsen and Winskel [35, 17]. (We have included a background section to introduce and motivate the categorical setting.) This opens up for using coinductive techniques for (partial) HDA. We also give a game characterization of this *hd-bisimilarity* and show that is decidable for finite partial HDA.

We then define unfoldings of partial HDA into higher-dimensional trees, which are given as the equivalence classes of computation paths under a certain notion of homotopy of computations, rather similarly to universal coverings in algebraic topology. These unfoldings

6th International Conference on Algebra and Coalgebra in Computer Science (CALCO'15).
Editors: Lawrence S. Moss and Pawel Sobocinski; pp. 101–115
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

are used to express hd-bisimilarity as an equivalence relation on homotopy classes of computations and, ultimately, directly on computations. This allows us to compare hd-bisimilarity to other common notions of equivalences for concurrent models, such as split bisimilarity [30], ST-bisimilarity [33] and history-preserving bisimilarity [25, 31]. We show that hd-bisimilarity is strictly weaker than history-preserving bisimilarity, but not weaker than split bisimilarity.

We start the paper by giving some categorical background for our developments in Section 2, with the purpose of introducing just enough category theory so that the rest of the paper, except perhaps for the last section, can be understood also by readers without a categorical inclination. Section 3 then introduces partial HDA and shows important examples of systems which can be modeled only as partial HDA. In Section 4 we then introduce our notion of hd-bisimilarity through open maps in the category of partial HDA. We give an elementary characterization of hd-bisimilarity in Theorem 9 and a characterization using games in Theorem 12.

Section 5, introducing homotopy of computations and unfoldings of partial HDA, is the technical core of the paper. Its central result is Corollary 16, that partial HDA are hd-bisimilar iff their unfoldings are so. This result is used for comparing hd-bisimilarity with other equivalences for concurrent models in Section 6, showing in Theorem 18 our main result that hd-bisimilarity is strictly weaker than history-preserving bisimilarity but not weaker than split bisimilarity.

For (total) HDA, the categorical setting on which our work is built was first introduced in [4, 5]. It has the advantage of a close analogy to the simplicial and cubical sets used in algebraic topology [26, 3, 16]. Later we have connected this work to history-preserving bisimilarity in [6], see also [7] for some corrections. Note that the version of hd-bisimilarity introduced in our earlier work [6, 7] for HDA is different from the one we define here; indeed the earlier variant is incomparable with history-preserving bisimilarity. This is essentially because HDA are required to have all boundaries and is avoided by passing to partial HDA.

2 Categorical Background

To warm up, we review some of the work in [35, 17] on the category of transition systems and open maps for bisimulations, modified slightly to suit our purposes. This categorical setting is useful for us, because it allows to state properties in an abstract generality which allows for immediate generalization to other settings. More specifically, the work of Joyal, Winskel and Nielsen in [35, 17] and other papers has been influential because through the categorical setting, properties can be stated and proven across formalisms and easily be transferred from one formalism to another. This has exposed some very useful similarities between formalisms which look very different, for example transition systems, Petri nets, and event structures. Hence category theory is useful here as an ordering principle.

2.1 Digraphs

A *digraph* $X = (X_1, X_0)$ consists of two sets X_1, X_0, of edges and vertices, together with *face maps* $\delta^0, \delta^1 : X_1 \to X_0$ assigning start and end vertices to every edge. Note that we allow loops and multiple edges in our digraphs.

A *morphism* of digraphs $f : X \to Y$ consists of two mappings $f_1 : X_1 \to Y_1$, $f_0 : X_0 \to Y_0$ which commute with the face maps, *i.e.* such that $f_0(\delta^0 a) = \delta^0 f_1(a)$ and $f_0(\delta^1 a) = \delta^1 f_1(a)$ for every edge $a \in X_1$. Hence morphisms are standard digraph homomorphisms.

Digraphs and their morphisms form a *category*, in that composition of morphisms is associative and every digraph X has an identity morphism id^X given by $\mathrm{id}_1^X(a) = a$ and $\mathrm{id}_0^X(x) = x$. We will denote this category by Dgr.

2.2 Transition Systems

A *transition system* (X, i_0) is a digraph X with a specified initial vertex (state) $i_0 \in X_0$. This is the same as specifying a mapping $i_0 : \{0\} \to X_0$ from a one-point set into the vertices, which can be extended (uniquely) to a morphism $i : * \to X$ from the *one-point digraph* (without edges). We have hence transferred an *internal* object, an element $i_0 \in X_0$, to an *external* setting, a morphism $i : * \to X$. This process of *externalization* is very important in applications of category theory, as it allows to transfer properties internal to objects or morphisms (here the very simple property of having a specified initial element) to an external setting which only uses objects and morphisms as-is.

Morphisms of transition systems are required to respect the initial states, *i.e.* if $f : (X, i) \to (Y, j)$ is such a morphism, then we must have $f(i) = j$. This is the same as saying that the category of transition systems is the *comma category* (or slice) of digraphs under the object $*$: objects are digraph morphisms $* \to X$ and morphisms are digraph morphisms $f : X \to Y$ for which the diagram

$$\begin{tikzcd}
 & * \arrow[dl, "i"'] \arrow[dr, "j"] & \\
X \arrow[rr, "f"'] & & Y
\end{tikzcd}$$

commutes. We denote this comma category by $* \downarrow \mathsf{Dgr}$.

2.3 Labeled Transition Systems

A *labeled transition system* (LTS) (X, i, λ_1), over some alphabet Σ, is a transition system $i : * \to X$ together with an edge labeling $\lambda_1 : X_1 \to \Sigma$. We externalize the edge labeling: Let $!\Sigma = (\{0\}, \Sigma)$ be the one-point digraph with edge set Σ (*i.e.* the digraph with one vertex 0 and $\delta^0\alpha = \delta^1\alpha = 0$ for every $\alpha \in \Sigma$), then any mapping $\lambda_1 : X_1 \to \Sigma$ can be extended (uniquely) to a digraph morphism $\lambda : X \to !\Sigma$. A LTS is, then, a system of digraph morphisms $* \xrightarrow{i} X \xrightarrow{\lambda} !\Sigma$: $i : * \to X$ specifies the initial state, and $\lambda : X \to !\Sigma$ associates labels to edges.

Morphisms of LTS $(X, i, \lambda_1) \to (Y, j, \mu_1)$ are digraph morphisms $f : X \to Y$ which respect the initial state and the labeling: for every $a \in X_1$, $\mu_1(f_1(a)) = \lambda_1(a)$. (For simplicity we only consider such label-preserving morphisms here; this is all we will need later.) This is the same as saying that the category of LTS is the double comma category $* \downarrow \mathsf{Dgr} \downarrow !\Sigma$ of digraphs between $*$ and $!\Sigma$: objects are structures $* \xrightarrow{i} X \xrightarrow{\lambda} !\Sigma$ and morphisms are commutative diagrams

$$\begin{tikzcd}
 & * \arrow[dl, "i"'] \arrow[dr, "j"] & \\
X \arrow[rr, "f"] \arrow[dr, "\lambda"'] & & Y \arrow[dl, "\mu"] \\
 & !\Sigma &
\end{tikzcd}$$

Posing initial states and labels as a double comma category has the advantage that many constructions can be simply defined on the base category (here: digraphs; below: partial precubical sets) and then lifted to the double comma category. We will exploit this below to do most of our work in the unlabeled category (of partial HDA) and only in the last section lift it to the labeled setting.

2.4 Open Maps and Bisimilarity

A digraph morphism $f : X \to Y$ is called an *open map* if it holds that for all $x \in X_0$ and $b \in Y_1$ with $\delta^0 b = f_0(x)$, there exists $a \in X_1$ with $\delta^0 a = x$ such that $b = f_1(a)$. Hence any edge b which starts in $f_0(x)$ can be lifted (not necessarily uniquely) to an edge a, emanating from x, for which $b = f_1(a)$.

One of the contributions of [17] is the lift of the above open maps to the usual *relational* setting of bisimulation [21, 19]: by a theorem of [17], two LTS X, Y are *bisimilar* iff there exists an LTS Z and a *span of open maps* $X \leftarrow Z \to Y$.

2.5 Path Objects

To externalize the property of being open, one defines a category of *path objects* (or *computations*). A path object is a transition system $(\{x_0, \dots, x_n\}, \{(x_0, x_1), \dots, (x_{n-1}, x_n)\}, x_0)$, for $n \geq 0$, *i.e.* a path in the graph-theoretical sense, with distinct states $x_0, \dots, x_n$ and transitions from x_i to x_{i+1} for all $i = 0, \dots, n-1$. Morphisms of path objects are inclusions of shorter paths into longer ones (hence path objects form a *full* subcategory of transition systems). It can then be shown that a transition system morphism $f : X \to Y$ is open iff there is a morphism (a *lift*) $r : Q \to X$ in any diagram of the form

$$\begin{tikzcd}
P \arrow[r] \arrow[d] & X \arrow[d, "f"] \\
Q \arrow[r] \arrow[ur, dotted, "r"] & Y ,
\end{tikzcd}$$

where P and Q are path objects.

We have established that bisimulation of LTS can be posed in an entirely external categorical setting, where two LTS are bisimilar iff here is a span of morphisms which have a special property of being open which is defined through a (right) lifting property with respect to a subcategory of paths. This will be a guidance for our developments in later sections.

3 Partial Higher-dimensional Automata

Higher-dimensional automata [23, 29] generalize transition systems in the sense that they allow for higher-dimensional transitions: they admit states and transitions, but also two-dimensional (squares) and three-dimensional (cubes) transitions, *etc.* *Partial* HDA, as presented in this paper, are a further generalization in which some transitions may not have start or end states, or squares may not have some of their start or end transitions, *etc.* As in the preceding section, we define partial HDA using a comma category construction.

3.1 HDA

We start by recalling HDA. A *precubical set* is a graded set $X = \{X_n\}_{n \in \mathbb{N}}$ together with mappings $\delta^\nu_{k(n)} : X_n \to X_{n-1}$, $k \in \{1, \dots, n\}$, $\nu \in \{0, 1\}$, satisfying the *precubical identity*

$$\delta^\nu_k \delta^\mu_\ell = \delta^\mu_{\ell-1} \delta^\nu_k \qquad (k < \ell)\,.$$

The mappings $\delta^\nu_{k(n)}$ are called *face maps* (note that we will omit the extra index (n)), and elements of X_n are called n*-cubes*. Faces $\delta^0_k x$ of an element $x \in X$ are to be thought of as *start faces*, $\delta^1_k x$ as *end faces*. The precubical identity expresses the fact that $(n-1)$-faces of an n-cube meet in common $(n-2)$-faces; see Fig. 1 for an example. Note how this generalizes

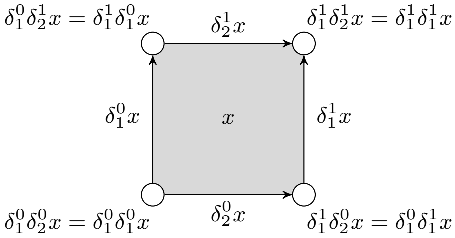

$\delta_1^0\delta_2^1x = \delta_1^1\delta_1^0x$
δ_2^1x
$\delta_1^1\delta_2^1x = \delta_1^1\delta_1^1x$
δ_1^0x
x
δ_1^1x
$\delta_1^0\delta_2^0x = \delta_1^0\delta_1^0x$
δ_2^0x
$\delta_1^1\delta_2^0x = \delta_1^0\delta_1^1x$

Figure 1 A 2-cube x with its four faces δ_1^0x, δ_1^1x, δ_2^0x, δ_2^1x and four corners.

digraphs to arbitrary dimensions: a precubical set includes vertices and edges, and some squares of edges may be filled in, some cubes of squares may be filled in, *etc.*

Similarly to digraph morphisms, morphisms $f : X \to Y$ of precubical sets are graded functions $f = \{f_n : X_n \to Y_n\}_{n\in\mathbb{N}}$ which commute with the face maps: $\delta_k^\nu \circ f_n = f_{n-1} \circ \delta_k^\nu$ for all $n \in \mathbb{N}$, $k \in \{1, \dots, n\}$, $\nu \in \{0, 1\}$. This defines a category of precubical sets.

The category of *HDA* is then the comma category of precubical sets under the one-point precubical set $*$ with one 0-cube and no other n-cubes. Hence a one-dimensional HDA is a transition system; indeed, the category of transition systems [35] is isomorphic to the full subcategory of one-dimensional HDA.

3.2 Partial HDA

The following example exposes a simple system which cannot be modeled as HDA; this motivates the introduction of partial HDA below.

▶ **Example 1.** Let a and b be two independent events (which hence may run concurrently) with the constraint that b cannot start before a and has to finish before a can finish. Hence b can only run "inside" a; by way of motivation, a could be a supervisor process which provides resources for b. (Hence this is an example of the disabling of an event by another event.)

Note that this system cannot be modeled as an event structure. We can represent it as an *ST-structure* as introduced in [24], which is comprised of configurations (S, T) of *started* (S) and *terminated* (T) events (hence always $T \subseteq S$):

$$(\emptyset, \emptyset) \xrightarrow[s]{a} (\{a\}, \emptyset) \xrightarrow[s]{b} (\{a, b\}, \emptyset) \xrightarrow[t]{b} (\{a, b\}, \{b\}) \xrightarrow[t]{a} (\{a, b\}, \{a, b\})$$

When trying to model this example as a HDA, *cf.* Fig. 2 below, we see that existence of the 2-cube corresponding to the configuration $(\{a, b\}, \emptyset)$ forces us to introduce all its boundaries into the model, *i.e.* not only the configurations $(\{a\}, \emptyset)$ and $(\{a, b\}, \{b\})$ as above, but also $(\{b\}, \emptyset)$ and $(\{a, b\}, \{a\})$. Thus we lose the property that b can only run inside a.

We hence define a *partial precubical set* (PPS) to be a graded set $X = \{X_n\}_{n\in\mathbb{N}}$ together with *partial* mappings $\delta_k^\nu : X_n \hookrightarrow X_{n-1}$, $k \in \{1, \dots, n\}$, $\nu \in \{0, 1\}$, satisfying the precubical identity

$$\delta_k^\nu \delta_\ell^\mu = \delta_{\ell-1}^\mu \delta_k^\nu \qquad (k < \ell) \tag{1}$$

whenever all the involved mappings are defined. We will always assume the sets X_n to be disjoint. For an n-cube $x \in X_n$, we denote by $\dim x = n$ its *dimension.*

Morphisms $f : X \to Y$ of PPS are graded *total* functions $f = \{f_n : X_n \to Y_n\}_{n\in\mathbb{N}}$ which commute with the face maps: $\delta_k^\nu \circ f_n = f_{n-1} \circ \delta_k^\nu$ for all $n \in \mathbb{N}$, $k \in \{1, \dots, n\}$,

$\nu \in \{0,1\}$ whenever the involved face maps are defined. This defines a category PPS of partial precubical sets and morphisms.

Products of PPS are given point-wise: for PPS X, Y, $X \times Y = Z$ with $Z_n = X_n \times Y_n$ and face maps defined by $\delta_k^\nu(x,y) = (\delta_k^\nu x, \delta_k^\nu y)$ iff both individual faces are defined. (This is the categorical product.) Also *subsets* are given point-wise: for $X, Y \in \mathsf{PPS}$, $Y \subseteq X$ iff $Y_n \subseteq X_n$ for all $n \in \mathbb{N}$.

A *pointed* PPS is a PPS X with a specified 0-cube $i \in X_0$, and a pointed morphism is one which respects the point. This defines a category which is isomorphic to the comma category $* \downarrow \mathsf{PPS}$, where $* \in \mathsf{PPS}$ is the precubical set with one 0-cube and no other n-cubes.

▶ **Definition 2.** The category of *partial higher-dimensional automata* (PHDA) is the comma category $\mathsf{PHDA} = * \downarrow \mathsf{PPS}$, with objects pointed PPS and morphisms commutative diagrams

$$\begin{tikzcd}
 & * \arrow[dl] \arrow[dr] & \\
X \arrow[rr, "f"] & & Y\,.
\end{tikzcd}$$

Intuitively, 0-cubes $x \in X_0$ are to be thought of as *states*, 1-cubes are *transitions*, and n-cubes for $n \geq 2$ model concurrent executions of n events. Note that a one-dimensional PHDA is a transition system in which transitions do not necessarily have start or end states. This may be useful for modeling deadlocks, even though we are not aware of any work in which this is done.

3.3 Labeled Partial HDA

For *labeling* PHDA, we let $\Sigma = \{a_1, a_2, \dots\}$ be a finite or infinite set of events. We construct a precubical set $!\Sigma = \{!\Sigma_n\}$ by letting $!\Sigma_n = \{(a_{i_1}, \dots, a_{i_n}) \mid i_k \leq i_{k+1} \text{ for all } k = 1, \dots, n-1\}$ with face maps defined by $\delta_k^\nu(a_{i_1}, \dots, a_{i_n}) = (a_{i_1}, \dots, a_{i_{k-1}}, a_{i_{k+1}}, \dots, a_{i_n})$. Note that $!\Sigma$ is a *torus*: start and end faces of any n-cube agree, hence all n-cubes are loops.

▶ **Definition 3.** The category of *labeled PHDA* over Σ is the double comma category $\mathsf{LHDA} = * \downarrow \mathsf{PPS} \downarrow !\Sigma$.

Note that this labels opposite transitions with the same event, *i.e.* $\lambda\delta_1^0 z = \lambda\delta_1^1 z$ and $\lambda\delta_2^0 z = \lambda\delta_2^1 z$, for every $z \in X_2$, whenever these boundaries exist in X_1. This conveys the intuition that opposite boundaries of a square execute the same event, connected by possible concurrent execution of another event. We develop most of the material in this paper for unlabeled PHDA and transport it to the labeled setting in Section 6.

▶ **Example 4.** We can now expose a labeled PHDA model for the system of Example 1. Let $X \in \mathsf{PHDA}$ be such that $X_n = \emptyset$ for $n \geq 3$, $X_2 = \{z\}$, $X_1 = \{y_1, y_2\}$ and $X_0 = \{x_0, x_2\}$, with face maps $\delta_2^0 z = y_1$, $\delta_2^1 z = y_2$, $\delta_1^0 y_1 = x_0$, $\delta_1^1 y_2 = x_2$ (and all others undefined), initial state x_0 and labeling $\lambda(y_1) = \lambda(y_2) = a$, $\lambda(z) = ab$, see Fig. 2. The computational interpretation of X is that b can only start while a is executing, and a can only finish once b is done.

▶ **Example 5.** For a slightly more involved example, let again a and b be independent events, but this time so that a is executed in a loop; once b has started, a cannot be started anymore; and b can only finish when a is not running (hence b has priority over a). By way of motivation, b could be a “shutdown” process which waits for other processes to terminate but does not allow new ones to start. As a labeled PHDA, this can be modeled as in Fig. 2. Note that this PHDA contains a cycle; the two copies of x_0 on the left indicate that they are to be identified, as can be seen on the right.

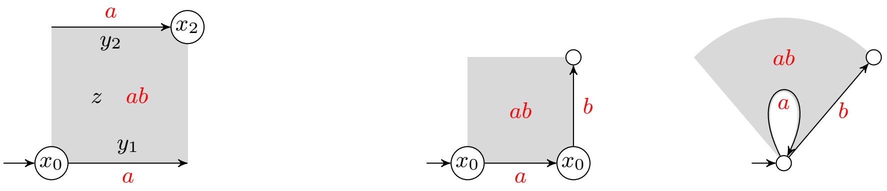


Figure 2 Labeled PHDA of Examples 4 and 5. The gray area signifies a 2-cube; labels are indicated in red.

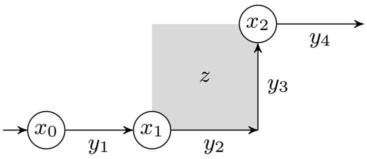


Figure 3 The two-dimensional path object $(x_0, y_1, x_1, y_2, z, y_3, x_2, y_4)$. Its computational interpretation is that y_1 is executed first; after it finishes, y_2 is started, and while y_2 is running, y_3 starts to execute. After this, y_2 finishes, then y_3 finishes, and then execution of y_4 is started. Note that the computation is partial, as y_4 does not finish.

4 Higher-dimensional Bisimilarity

Following the procedure outlined in Section 2, we now introduce path objects, define open maps as these morphisms which have the right-lifting property with respect to the path category, and use this to define bisimilarity. This is similar to what we did in [6], but because we are working with *partial* HDA, things are closer to the computational intuition.

4.1 Path Objects

We say that a PPS X is a *path object* if its n-cubes can be sorted into a (necessarily unique) sequence $(x_1, \dots, x_m)$ such that $x_i \neq x_j$ for $i \neq j$, for each $j = 1, \dots, m-1$, there is $k \in \mathbb{N}$ for which $x_j = \delta_k^0 x_{j+1}$ or $x_{j+1} = \delta_k^1 x_j$, and no other relations exist between the x_i. Hence a path object is a sequence of cubes which are connected so that either x_{j+1} is an extension of x_j, signifying the start of a new event, or x_{j+1} is an end face of x_j, signifying the end of an event, see Fig. 3 for an example.

A *pointed* path object $i : * \to X$ consists of a path object X and the mapping i which includes the point as x_1 (hence $x_1 \in X_0$). Intuitively, path objects are models of PHDA computations, just as paths are models of transition system computations (Section 2). Pointed path objects are computations from an initial state.

If X and Y are path objects with representations $(x_1, \dots, x_m)$, $(y_1, \dots, y_p)$, then a morphism $f : X \to Y$ is called a *cube path extension* if $x_j = y_j$ for all $j = 1, \dots, m$ (hence $m \leq p$). This models the extension of one computation by zero or more steps, in analogy to extensions of paths in Section 2.

▶ **Definition 6.** The category HDP of *higher-dimensional paths* is the subcategory of PHDA which as objects has pointed path objects and whose morphisms are generated by pointed cube path extensions and isomorphisms.

4.2 Open Maps and Hd-bisimilarity

▶ **Definition 7.** A pointed morphism $f : X \to Y$ in PHDA is an *open map* if it has the right lifting property with respect to HDP, *i.e.* if it is the case that there is a lift r in any commutative diagram as below, for morphisms $g : P \to Q \in \mathsf{HDP}$, $p : P \to X, q : Q \to Y \in \mathsf{PHDA}$:

$$\begin{tikzcd}
P \arrow[r, "p"] \arrow[d, "g"'] & X \arrow[d, "f"] \\
Q \arrow[r, "q"'] \arrow[ur, dotted, "r"] & Y
\end{tikzcd}$$

Note how this is entirely analogous to what we did in Section 2. Stating concepts in a categorical way has allowed us to transport them from transition systems to PHDA.

▶ **Definition 8.** PHDA X, Y are *hd-bisimilar* if there is $Z \in \mathsf{PHDA}$ and a span of open maps $X \leftarrow Z \to Y$ in PHDA.

A relational formulation of this is as follows:

▶ **Theorem 9.** *PHDA $i : * \to X$, $j : * \to Y$ are hd-bisimilar iff there exists a PPS $R \subseteq X \times Y$ for which $(i, j) \in R$, and such that for all $(x_1, y_1) \in R$,*

1. *for any $x_2 \in X$ for which $x_1 = \delta_k^0 x_2$ for some k, there exists $y_2 \in Y$ for which $y_1 = \delta_k^0 y_2$ and $(x_2, y_2) \in R$,*
2. *for any $x_2 \in X$ for which $x_2 = \delta_k^1 x_1$ for some k, there exists $y_2 \in Y$ for which $y_2 = \delta_k^1 y_1$ and $(x_2, y_2) \in R$,*
3. *for any $y_2 \in Y$ for which $y_1 = \delta_k^0 y_2$ for some k, there exists $x_2 \in X$ for which $x_1 = \delta_k^0 x_2$ and $(x_2, y_2) \in R$,*
4. *for any $y_2 \in Y$ for which $y_2 = \delta_k^1 y_1$ for some k, there exists $x_2 \in X$ for which $x_2 = \delta_k^1 x_1$ and $(x_2, y_2) \in R$.*

Proof. For the forward implication, let $X \xleftarrow{f} Z \xrightarrow{g} Y$ be a span of open maps and define $R = \{(x, y) \in X \times Y \mid \exists z \in Z : x = f(z), y = g(z)\}$. Then $(i, j) \in R$ because f and g are pointed morphisms, properties (1) and (3) hold because f and g are PPS morphisms, and properties (2) and (4) hold because f and g are open. For the backwards implication, let $\pi_X : R \to X$, $\pi_Y : R \to Y$ be the projections; these are easily shown to be open maps. ◀

▶ **Corollary 10.** *For finite PHDA, hd-bisimilarity is decidable in polynomial time.*

Proof. The condition in Theorem 9 immediately gives rise to a fixed-point algorithm similar to the one used to decide standard bisimilarity, *cf.* [19, 18]. ◀

▶ **Example 11.** The two (total) labeled HDA in Fig. 4 are hd-bisimilar, as witnessed by the following PPS $R \subseteq X \times X'$:

$$R_0 = \{(x_0, x_0'), (x_1, x_1'), (x_2, x_2'), (x_3, x_4'), (x_4, x_4')\}$$
$$R_1 = \{(y_1, y_1'), (y_2, y_2'), (y_3, y_4'), (y_4, y_4'), (y_5, y_5')\}$$
$$R_2 = \{(z, z')\}$$

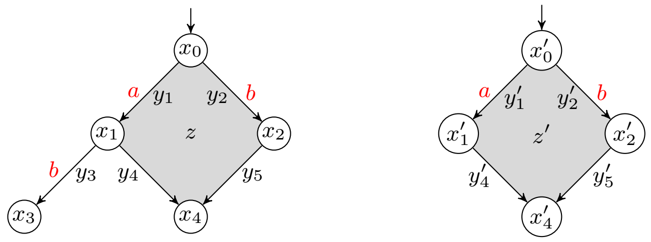


Figure 4 Two HDA pertaining to Example 11.

4.3 Hd-bisimulation Games

We can also expose a game characterization of hd-bisimilarity, similar to the notion of bisimulation game for interleaving bisimilarity [28]. The game is played by two players, *Spoiler* and *Duplicator*, and a configuration of the game is a pair (x, y) of n-cubes $x \in X$, $y \in Y$ of equal dimension. The initial configuration is (i, j).

At each round of the game, from a configuration (x_1, y_1), the spoiler chooses to play one of four moves: either

1. to choose $x_2 \in X$ with $x_1 = \delta^0_k x_2$ for some k,
2. to choose $x_2 \in X$ with $x_2 = \delta^1_k x_1$ for some k,
3. to choose $y_2 \in Y$ with $y_1 = \delta^0_k y_2$ for some k, or
4. to choose $y_2 \in Y$ with $y_2 = \delta^1_k y_1$ for some k.

Depending on the type of move of the spoiler, the duplicator now has to answer by, respectively,

1. choosing $y_2 \in Y$ with $y_1 = \delta^0_k y_2$,
2. choosing $y_2 \in Y$ with $y_2 = \delta^1_k y_1$,
3. choosing $x_2 \in X$ with $x_1 = \delta^0_k x_2$, or
4. choosing $x_2 \in X$ with $x_2 = \delta^1_k x_1$,

and the game continues from the configuration (x_2, y_2).

The spoiler wins the game if the duplicator gets stuck, *i.e.* if the game finishes because duplicator has no answer to a move of the spoiler. Otherwise (if the game is infinite, or if it finishes because the spoiler has no move) the duplicator has won. The proof of the following theorem is similar to the one for the game characterization of interleaving bisimilarity [28].

▶ **Theorem 12.** *PHDA X and Y are hd-bisimilar iff the duplicator has a winning strategy in the hd-bisimulation game between X and Y.*

5 Homotopy and Unfoldings

Most other common notions of equivalences for concurrent systems, such as (hereditary) history-preserving bisimilarity, ST-bisimilarity or split bisimilarity, are defined on computations rather than structurally (see [30]; we will define them formally below). Hence to compare our notion of hd-bisimilarity to these other equivalences, we need to lift it to a relation on computations. The vehicle for doing so is the *unfolding* of a PHDA, similar to the *universal covering space* in algebraic topology.

5.1 Computations

We have already introduced path objects above, which embody the intuition behind PHDA computations. Using these to define computations *within* a given PHDA, we say that a *cube*

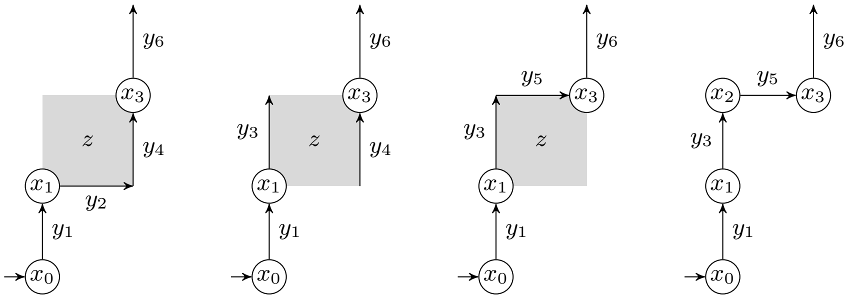


Figure 5 The cube path homotopy $(x_0, y_1, x_1, y_2, z, y_4, x_3, y_6) \sim (x_0, y_1, x_1, y_3, z, y_4, x_3, y_6) \sim (x_0, y_1, x_1, y_3, z, y_5, x_3, y_6) \sim (x_0, y_1, x_1, y_3, x_2, y_5, x_3, y_6)$.

path in a PPS X is a morphism $P \to X$ from a path object P. In elementary terms, this is a sequence $(x_1, \dots, x_m)$ of elements of X such that for each $j = 1, \dots, m-1$, there is $k \in \mathbb{N}$ for which $x_j = \delta^0_k x_{j+1}$ (start of a new part of a computation) or $x_{j+1} = \delta^1_k x_j$ (end of a computation part).

Note that cube paths, contrary to path objects, may have loops and self-intersections (conforming to the intuition that they be computations in a PHDA). As an example, the PHDA in Fig. 2 is not itself a path object, but any finite sequence of a-labeled transitions is a cube path within it, as is any finite sequence of a-labeled transitions followed by a b-labeled transition.

A *pointed* cube path in a PHDA $* \to X$ is a pointed morphism from a pointed path object. We will say that a cube path $(x_1, \dots, x_m)$ is *from* x_1 *to* x_m, and that an n-cube $x \in X$ in a PHDA X is *reachable* if there is a pointed cube path to x in X.

5.2 Homotopy of Computations

We define an equivalence relation on cube paths which formalizes the intuition of when two concurrent computations are the same. We say that cube paths $\rho = (x_1, \dots, x_m)$, $\sigma = (y_1, \dots, y_m)$ are *p-adjacent*, and write $\rho \overset{p}{\sim} \sigma$, for $p \in \{2, \dots, m-1\}$, if $x_p \neq y_p$ and $x_j = y_j$ for $j \neq p$, and one of the following conditions is satisfied:

- $x_{p-1} = \delta^0_k x_p$, $x_p = \delta^0_\ell x_{p+1}$, $y_{p-1} = \delta^0_{\ell-1} y_p$, and $y_p = \delta^0_k y_{p+1}$ for some $k < \ell$, or vice versa,
- $x_p = \delta^1_k x_{p-1}$, $x_{p+1} = \delta^1_\ell x_p$, $y_p = \delta^1_{\ell-1} y_{p-1}$, and $y_{p+1} = \delta^1_k y_p$ for some $k < \ell$, or vice versa,
- $x_p = \delta^0_k \delta^1_\ell y_p$, $y_{p-1} = \delta^0_k y_p$, and $y_{p+1} = \delta^1_\ell y_p$ for some $k < \ell$, or vice versa, or
- $x_p = \delta^1_k \delta^0_\ell y_p$, $y_{p-1} = \delta^0_\ell y_p$, and $y_{p+1} = \delta^1_k y_p$ for some $k < \ell$, or vice versa.

The intuition of adjacency is rather simple, even though the combinatorics may look complicated; see Fig. 5 for an example. Note that adjacencies come in two basic "flavors": the first two above in which the dimensions of x_p and y_p are the same, and the last two in which they differ by 2.

We say that two cube paths are adjacent if they are p-adjacent for some p, and *homotopy* of cube paths is defined to be the reflexive, transitive closure of the adjacency relation. We denote homotopy of cube paths using the symbol $\sim$, and the homotopy class of a cube path $(x_1, \dots, x_m)$ is denoted $[x_1, \dots, x_m]$.

5.3 Unfoldings

We will unfold PHDA into *higher-dimensional trees*, which are PHDA X for which it holds that there is precisely one homotopy class of cube paths to any n-cube $x \in X$. The full subcategory of PHDA spanned by the higher-dimensional trees is denoted HDT. Note that any path object is a higher-dimensional tree.

▶ **Definition 13.** The *unfolding* of a PHDA $i : * \to X$ consists of a PHDA $\tilde{i} : * \to \tilde{X}$ and a pointed *projection* morphism $\pi_X : \tilde{X} \to X$, which are defined as follows:

- $\tilde{X}_n = \big\{[x_1, \dots, x_m] \mid (x_1, \dots, x_m)$ pointed cube path in $X, x_m \in X_n\big\}$; $\tilde{i} = [i]$
- $\tilde{\delta}^0_k[x_1, \dots, x_m] = \big\{(y_1, \dots, y_p) \mid y_p = \delta^0_k x_m, (y_1, \dots, y_p, x_m) \sim (x_1, \dots, x_m)\big\}$ provided this set is non-empty; otherwise undefined
- $\tilde{\delta}^1_k[x_1, \dots, x_m] = [x_1, \dots, x_m, \delta^1_k x_m]$ if $\delta^1_k x_m$ exists; otherwise undefined
- $\pi_X[x_1, \dots, x_m] = x_m$

▶ **Theorem 14.** *The unfolding $(\tilde{X}, \pi_X)$ of a PHDA X is well-defined, and $\tilde{X}$ is a higher-dimensional tree. If X itself is a higher-dimensional tree, then the projection $\pi_X : \tilde{X} \to X$ is an isomorphism.*

Proof sketch. Note the complete analogy to the construction of universal covering spaces in algebraic topology: $\tilde{X}$ consists of homotopy classes of (cube) paths, and the projection maps a path to its end point (cube). The proof is similar to the one we gave for (total) HDA in [6], but with the important difference that a certain ("fan-shaped") normal form for cube paths, which we used in [6], is not available for partial HDA. We present a sketch of the proof here; the full proof is in appendix.

To see that $\tilde{X}$ is well-defined, we need to show that the face maps $\tilde{\delta}^0_k$ and $\tilde{\delta}^1_k$ are independent of the representative in the homotopy class. For $\tilde{\delta}^1_k$ this is trivial, but for $\tilde{\delta}^0_k$ it requires more work. We also need to prove that the precubical identity $\tilde{\delta}^\nu_k \tilde{\delta}^\mu_\ell = \tilde{\delta}^\mu_{\ell-1} \tilde{\delta}^\nu_k$ is satisfied whenever the faces exist; this is again trivial for $\nu = \mu = 1$ and more complicated for the other cases.

The projection $\pi : \tilde{X} \to X$ is clearly well-defined, as homotopic cube paths have identical end points. To see that it is a PHDA morphism, *i.e.* that $\pi_X \tilde{\delta}^\nu_k = \delta^\nu_k \pi_X$, is again trivial for $\nu = 1$ and more complicated for $\nu = 0$.

The proof that $\tilde{X}$ is a higher-dimensional tree is in appendix. If X itself is a higher-dimensional tree, then an inverse to π_X is given by mapping $x \in X$ to the unique homotopy class $[x_1, \dots, x_m] \in \tilde{X}$ of any pointed cube path $(x_1, \dots, x_m)$ in X with $x_m = x$. ◀

▶ **Theorem 15.** *Projections $\pi_X : \tilde{X} \to X$ are open, hence any PHDA is hd-bisimilar to its unfolding.*

Transitivity of hd-bisimilarity now implies the following, relating hd-bisimilarity of PHDA to hd-bisimilarity of homotopy classes of computations. This will be central in our comparison to other equivalences in Section 6.

▶ **Corollary 16.** *PHDA X, Y are hd-bisimilar iff their unfoldings $\tilde{X}$, $\tilde{Y}$ are hd-bisimilar.*

6 Relation to Other Equivalences

We now lift hd-bisimilarity to the labeled setting and relate it to other equivalences for concurrent models. We will show that hd-bisimilarity is implied by history-preserving bisimilarity, but not by split bisimilarity. As LHDA $= * \downarrow$ PPS $\downarrow !\Sigma$ is defined as a double comma category, our notions of open maps and hd-bisimilarity trivially carry over; in LHDA, these are now required to preserve labels.

We recall the notions of history-preserving bisimilarity, ST-bisimilarity and split bisimilarity from [30] (and extend them to partial HDA). For a labeled PHDA $* \to X \xrightarrow{\lambda} !\Sigma$, we extend λ to cube paths in X by $\lambda(x_1, \dots, x_m) = (\lambda(x_1), \dots, \lambda(x_m))$. Note that there is a one-to-one correspondence between label sequences $\lambda(\rho)$ and *split traces*, see [30, Sect. 7.5]. Below we use $\rightsquigarrow$ for cube path extensions, *i.e.* $\rho \rightsquigarrow \rho'$ iff ρ is a prefix of ρ'.

Labeled PHDA $* \xrightarrow{i} X \xrightarrow{\lambda} !\Sigma$, $* \xrightarrow{j} Y \xrightarrow{\mu} !\Sigma$ are *split bisimilar* iff there exists a relation R between pointed cube paths in X and pointed cube paths in Y for which $((i),(j)) \in R$, and such that for all $(\rho,\sigma) \in R$,

(1) $\lambda(\rho) = \mu(\sigma)$,
(2) for all $\rho \rightsquigarrow \rho'$ there exists $\sigma \rightsquigarrow \sigma'$ with $(\rho',\sigma') \in R$, and
(3) for all $\sigma \rightsquigarrow \sigma'$ there exists $\rho \rightsquigarrow \rho'$ with $(\rho',\sigma') \in R$.

X and Y are *ST-bisimilar* if, instead of condition 1 above, it holds that

(1') ST-*trace*$(\rho) = \mathit{ST}$-*trace*(σ).

Here ST-*trace*(ρ) is the ST-*trace* of ρ defined by annotating *split-trace*(ρ) with a mapping which gives the starting point of any terminating action, see [30] (this is important for auto-concurrency). X and Y are *history-preserving bisimilar* iff 6, 2 and 3 hold and, additionally, for all $(\rho,\sigma) \in R$ and all p,

(4) for all $\rho \overset{p}{\sim} \rho'$, there exists $\sigma \overset{p}{\sim} \sigma'$ with $(\rho',\sigma') \in R$, and
(5) for all $\sigma \overset{p}{\sim} \sigma'$, there exists $\rho \overset{p}{\sim} \rho'$ with $(\rho',\sigma') \in R$.

▶ **Example 11** (contd.). In the example in Fig. 4 above, there is an ST-bisimilarity relation which relates the cube path (x_0,y_1,x_1,y_3,x_3) to $(x_0',y_1',x_1',y_4',x_4')$, and in fact any ST-bisimilarity needs to do so. But then $(x_0',y_1',x_1',y_4',x_4')$ is 3-adjacent to (x_0',y_1',z',y_4',x_4'), whereas (x_0,y_1,x_1,y_3,x_3) admits no 3-adjacency. Hence these HDA are ST-bisimilar but not history-preserving bisimilar.

The following theorem expresses hd-bisimilarity in a way comparable to the above definitions.

▶ **Theorem 17.** *Labeled PHDA* $* \xrightarrow{i} X \xrightarrow{\lambda} !\Sigma$, $* \xrightarrow{j} Y \xrightarrow{\mu} !\Sigma$ *are hd-bisimilar iff there exists a relation R between pointed cube paths in X and pointed cube paths in Y for which* $((i),(j)) \in R$, *and such that for all* $(\rho,\sigma) \in R$,

1. $\lambda(\rho) \sim \mu(\sigma)$,
2. *for all* $\rho \rightsquigarrow \rho'$, *there exists* $\sigma \rightsquigarrow \sigma'$ *with* $(\rho',\sigma') \in R$,
3. *for all* $\sigma \rightsquigarrow \sigma'$, *there exists* $\rho \rightsquigarrow \rho'$ *with* $(\rho',\sigma') \in R$,
4. *for all* $\rho \sim \rho'$, *there exists* $\sigma \sim \sigma'$ *with* $(\rho',\sigma') \in R$, *and*
5. *for all* $\sigma \sim \sigma'$, *there exists* $\rho \sim \rho'$ *with* $(\rho',\sigma') \in R$.

▶ **Example 11** (contd.). Continuing the example in Fig. 4 above, a hd-bisimilarity relation as in Theorem 17 relates the cube path (x_0,y_1,x_1,y_3,x_3) to $(x_0',y_1',x_1',y_4',x_4')$, but also to (x_0',y_1',z',y_4',x_4') and to any other cube path in X' homotopic to $(x_0',y_1',x_1',y_4',x_4')$.

▶ **Theorem 18.** *Hd-bisimilarity is strictly weaker than history-preserving bisimilarity, but not weaker than split bisimilarity.*

Proof. When comparing the conditions in Theorem 17 with the ones for history-preserving bisimilarity above, we see that $\lambda(\rho) = \mu(\sigma)$ implies $\lambda(\rho) \sim \mu(\sigma)$ and adjacency implies homotopy. (For history-preserving bisimilarity, the adjacencies are required to happen *in the same place* in the cube paths.) Thus history-preserving bisimilarity implies hd-bisimilarity.

In Example 11 we have seen two labeled HDA which are hd-bisimilar but not history-preserving bisimilar, hence hd-bisimilarity is strictly weaker than history-preserving bisimilarity. Example 19 below will expose two labeled HDA which are split bisimilar but not hd-bisimilar, showing the last claim of the theorem. ◀

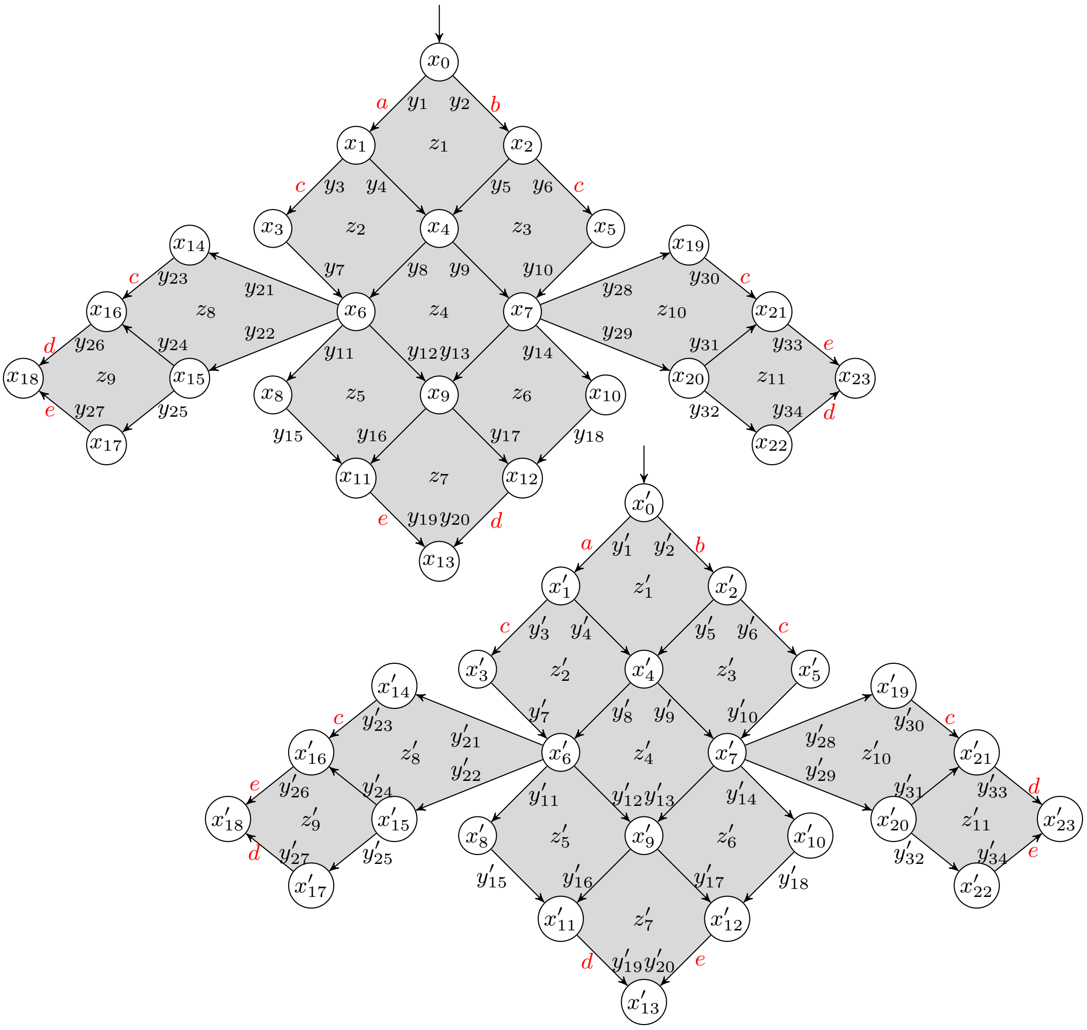

x_0
a y_1 y_2 b
x_1 z_1 x_2
c y_3 y_4 y_5 y_6 c
x_3 z_2 x_4 z_3 x_5
x_{14} x_{19}
c y_{23} y_{21} y_7 y_8 y_9 y_{10} y_{28} y_{30} c
x_{16} z_8 x_6 z_4 x_7 z_{10} x_{21}
d y_{26} y_{24} y_{22} y_{11} $y_{12}y_{13}$ y_{14} y_{29} y_{31} y_{33} e
x_{18} z_9 x_{15} x_8 z_5 x_9 z_6 x_{10} x_{20} z_{11} x_{23}
e y_{27} y_{25} y_{15} y_{16} y_{17} y_{18} y_{32} y_{34} d
x_{17} x_{11} z_7 x_{12} x_{22}
e $y_{19}y_{20}$ d
x_{13}
x'_0
a y'_1 y'_2 b
x'_1 z'_1 x'_2
c y'_3 y'_4 y'_5 y'_6 c
x'_3 z'_2 x'_4 z'_3 x'_5
x'_{14} x'_{19}
c y'_{23} y'_{21} y'_7 y'_8 y'_9 y'_{10} y'_{30} c
x'_{16} z'_8 y'_{22} x'_6 z'_4 x'_7 y'_{28} z'_{10} x'_{21}
e y'_{26} y'_{24} y'_{11} $y'_{12}y'_{13}$ y'_{14} y'_{29} y'_{31} y'_{33} d
x'_{18} z'_9 x'_{15} x'_8 z'_5 x'_9 z'_6 x'_{10} x'_{20} z'_{11} x'_{23}
d y'_{27} y'_{25} y'_{15} y'_{16} y'_{17} y'_{18} y'_{32} y'_{34} e
x'_{17} x'_{11} z'_7 x'_{12} x'_{22}
d $y'_{19}y'_{20}$ e
x'_{13}

Figure 6 Two HDA pertaining to Example 19.

▶ **Example 19.** Using a hd-bisimulation game, we show that the HDA in Fig. 6 are not hd-bisimilar. Note that according to [34], they are split bisimilar. This shows that split bisimilarity does not imply hd-bisimilarity. From the initial configuration (x_0, x'_0) of the game, the spoiler plays y_1, to which the duplicator can only answer y'_1. Then the spoiler plays z_1, with only possible answer z'_1, leading to the configuration (z_1, z'_1). Playing y_4 and then z_2, the spoiler forces the configuration (z_2, z'_2) and, playing y_8 and then z_4, leads the game to the cc-labeled configuration (z_4, z'_4). Here the spoiler plays y_{12}, which the duplicator has to answer by the z'_4-boundary *in the same direction*, hence y'_{12}. But then the spoiler can play the cd-labeled z_5, to which the duplicator has no answer.

7 Conclusion and Further Work

We have introduced a generalization of higher-dimensional automata, partial HDA, which alleviates some modeling shortcomings of HDA. We have seen that PHDA are useful for modeling priorities and the disabling of events by other events, but they should also be useful for example in the context of left-merge (*e.g.* in ACP [2]) and other asymmetric operators.

We have seen that PHDA have a natural notion of (higher-dimensional) bisimilarity, which is polynomial-time decidable for finite PHDA. We have lifted this notion to a relation on computations in PHDA and seen that it is strictly weaker than history-preserving bisimilarity but not weaker than split bisimilarity, but its precise placement in the concurrent hierarchy, especially its relation with ST-bisimilarity, remains open.

To the best of our knowledge, hd-bisimilarity is the first useful equivalence notion for concurrent systems which is defined directly on the structure, instead of on computations. This is important from a practical point of view: we have seen that it can be decided using a simple fixed-point algorithm, or alternatively using a sort of higher-dimensional bisimulation game. We plan to implement these algorithms in a tool for equivalence checking of PHDA; this would make equivalence checking of concurrent systems feasible in practice.

Acknowledgments. The authors wish to thank Cristian Prisacariu and Rob van Glabbeek for enlightening discussions on the subject of this paper, and the organizers of SMC 2014 in Lyon for providing a forum for these discussions.

References

1 Marek A. Bednarczyk. *Categories of asynchronous systems.* PhD thesis, University of Sussex, UK, 1987.

2 Jan A. Bergstra and Jan W. Klop. Process algebra for synchronous communication. *Inf. Cont.*, 60(1-3):109–137, 1984.

3 Ronald Brown and Philip J. Higgins. On the algebra of cubes. *J. Pure Appl. Alg.*, 21:233–260, 1981.

4 Uli Fahrenberg. A category of higher-dimensional automata. In *FOSSACS*, volume 3441 of *Lect. Notes Comput. Sci.*, pages 187–201. Springer, 2005.

5 Uli Fahrenberg. *Higher-Dimensional Automata from a Topological Viewpoint.* PhD thesis, Aalborg University, Denmark, 2005.

6 Uli Fahrenberg and Axel Legay. History-preserving bisimilarity for higher-dimensional automata via open maps. *Electr. Notes Theor. Comput. Sci.*, 298:165–178, 2013.

7 Uli Fahrenberg and Axel Legay. Homotopy bisimilarity for higher-dimensional automata. *CoRR*, abs/1409.5865, 2014.

8 Lisbeth Fajstrup. Dipaths and dihomotopies in a cubical complex. *Adv. Appl. Math.*, 35(2):188–206, 2005.

9 Lisbeth Fajstrup, Éric Goubault, Emmanuel Haucourt, Samuel Mimram, and Martin Raußen. Trace spaces: An efficient new technique for state-space reduction. In Helmut Seidl, editor, *ESOP*, volume 7211 of *Lecture Notes in Computer Science*, pages 274–294. Springer, 2012.

10 Lisbeth Fajstrup, Martin Raussen, and Éric Goubault. Algebraic topology and concurrency. *Theor. Comput. Sci.*, 357(1-3):241–278, 2006.

11 Philippe Gaucher. Homotopical interpretation of globular complex by multipointed d-space. *Theory Appl. Categories*, 22:588–621, 2009.

12 Philippe Gaucher. Towards a homotopy theory of higher dimensional transition systems. *Theory Appl. Categories*, 25:295–341, 2011.

13 Éric Goubault. Geometry and concurrency: A user's guide. *Math. Struct. Comput. Sci.*, 10(4):411–425, 2000.

14 Éric Goubault and Thomas P. Jensen. Homology of higher dimensional automata. In Rance Cleaveland, editor, *CONCUR*, volume 630 of *Lect. Notes Comput. Sci.*, pages 254–268. Springer, 1992.

15 Éric Goubault and Samuel Mimram. Formal relationships between geometrical and classical models for concurrency. *Electronic Notes in Theoretical Computer Science*, 283:77–109, 2012.

16 Marco Grandis. *Directed algebraic topology: models of non-reversible worlds.* New mathematical monographs. Cambridge Univ. Press, 2009.

17 André Joyal, Mogens Nielsen, and Glynn Winskel. Bisimulation from open maps. *Inf. Comput.*, 127(2):164–185, 1996.

18 Dexter Kozen. *Automata and Computability.* Undergraduate Texts in Computer Science. Springer, 1997.

19 Robin Milner. *Communication and Concurrency.* Prentice Hall, 1989.

20 Mogens Nielsen, Gordon D. Plotkin, and Glynn Winskel. Petri nets, event structures and domains, part I. *Theor. Comput. Sci.*, 13:85–108, 1981.

21 David M.R. Park. Concurrency and automata on infinite sequences. In Peter Deussen, editor, *IFIP TCS*, volume 104 of *Lect. Notes Comput. Sci.*, pages 167–183. Springer, 1981.

22 Carl A. Petri. *Kommunikation mit Automaten.* Bonn: Institut für Instrumentelle Mathematik, Schriften des IIM Nr. 2, 1962.

23 Vaughan Pratt. Modeling concurrency with geometry. In *POPL*, pages 311–322. ACM Press, 1991.

24 Cristian Prisacariu. The glory of the past and geometrical concurrency. In Andrei Voronkov, editor, *Turing-100*, volume 10 of *EPiC*, pages 252–267. EasyChair, 2012.

25 Alexander M. Rabinovich and Boris A. Trakhtenbrot. Behavior structures and nets. *Fund. Inf.*, 11(4):357–403, 1988.

26 Jean-Pierre Serre. *Homologie singulière des espaces fibrés.* PhD thesis, Ecole Normale Supérieure, 1951.

27 Mike W. Shields. Concurrent machines. *The Computer Journal*, 28(5):449–465, 1985.

28 Colin Stirling. Modal and temporal logics for processes. In *Proc. Banff Higher Order Workshop*, volume 1043 of *Lect. Notes Comput. Sci.*, pages 149–237. Springer, 1995.

29 Rob J. van Glabbeek. Bisimulations for higher dimensional automata. Email message, June 1991. `http://theory.stanford.edu/~rvg/hda`.

30 Rob J. van Glabbeek. On the expressiveness of higher dimensional automata. *Theor. Comput. Sci.*, 356(3):265–290, 2006.

31 Rob J. van Glabbeek and Ursula Goltz. Refinement of actions and equivalence notions for concurrent systems. *Acta Inf.*, 37(4/5):229–327, 2001.

32 Rob J. van Glabbeek and Gordon D. Plotkin. Configuration structures, event structures and petri nets. *Theor. Comput. Sci.*, 410(41):4111–4159, 2009.

33 Rob J. van Glabbeek and Frits W. Vaandrager. Petri net models for algebraic theories of concurrency. In J. W. de Bakker, A. J. Nijman, and Philip C. Treleaven, editors, *PARLE (2)*, volume 259 of *Lect. Notes Comput. Sci.*, pages 224–242. Springer, 1987.

34 Rob J. van Glabbeek and Frits W. Vaandrager. The difference between splitting in n and n+1. *Inf. Comput.*, 136(2):109–142, 1997.

35 Glynn Winskel and Mogens Nielsen. Models for concurrency. In Samson Abramsky, Dov M. Gabbay, and Thomas S.E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 4. Clarendon Press, Oxford, 1995.

A Recipe for State-and-Effect Triangles

Bart Jacobs

Radboud University, Nijmegen, The Netherlands
`bart@cs.ru.nl`

Abstract

In the semantics of programming languages one can view programs as state transformers, or as predicate transformers. Recently the author has introduced 'state-and-effect' triangles which captures this situation categorically, involving an adjunction between state- and predicate-transformers. The current paper exploits a classical result in category theory, part of Jon Beck's monadicity theorem, to systematically construct such a state-and-effect triangle from an adjunction. The power of this construction is illustrated in many examples, both for the Boolean and probabilistic (quantitative) case.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases Duality, predicate transformer, state transformer, state-and-effect triangle

Digital Object Identifier 10.4230/LIPIcs.CALCO.2015.116

1 Introduction

In program semantics three approaches can be distinguished.

- Interpreting programs themselves as morphisms in certain categories. Composition in the category then corresponds to sequential composition. Parallel composition may be modeled via tensors $\otimes$. Since [26] the categories involved are often Kleisli categories $\mathcal{K}\ell(T)$ of a monad T, where the monad T captures a specific form of computation: deterministic, non-deterministic, probabilistic, *etc.*
- Interpreting programs via their actions on states, as *state transformers*. For instance, in probabilistic programming the states may be probabilistic distributions over certain valuations (mapping variables to values). Execution of a program changes the state, by adapting the probabilities of valuations. The state spaces often have algebraic structure, and take the form of Eilenberg-Moore categories $\mathcal{EM}(T)$ of a monad T.
- Interpreting programs via their actions on predicates, as *predicate transformers*. The predicates involved describe what holds (is true) at a specific point. Execution of a program may then adapt the validity of predicates. A particular form of semantics of this sort is weakest precondition computation [6]. In the context of (coalgebraic) modal logic, these predicate transformers appear as modal operators.

A systematic picture of these three approaches has emerged in categorical language, using triangles of the form described below, see [15], and also [13, 14].

$$\begin{tikzcd}
\boxed{\textbf{Heisenberg}} & & \boxed{\textbf{Schrödinger}} \\
\mathbf{Log}^{\mathrm{op}} = \left(\begin{array}{c}\text{predicate}\\ \text{transformers}\end{array}\right) \arrow[rr, bend left] \arrow[rr, phantom, "\top"] & & \left(\begin{array}{c}\text{state}\\ \text{transformers}\end{array}\right) \arrow[ll, bend left] \\
& \left(\text{computations}\right) \arrow[ul, "\text{Pred}"] \arrow[ur, "\text{Stat}"'] &
\end{tikzcd} \qquad (1)$$

6th International Conference on Algebra and Coalgebra in Computer Science (CALCO'15).
Editors: Lawrence S. Moss and Pawel Sobocinski; pp. 116–129
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The three nodes in this diagram represent categories of which only the morphisms are described. The arrows between these nodes are functors, where the two arrows $\rightleftarrows$ at the top form an adjunction. The two triangles involved should commute. In the case where two up-going 'predicate' and 'state' functors Pred and Stat in (1) are full and faithful, we have three equivalent ways of describing computations. On morphisms, the predicate functor yields what is called substitution in categorical logic, but what amounts to a weakest precondition operation in program semantics, or a modal operator in programming logic. The upper category on the left is of the form $\mathbf{Log}^{\mathrm{op}}$, where $\mathbf{Log}$ is some category of logical structures. The opposite category $(-)^{\mathrm{op}}$ is needed because predicate transformers operate in the reverse direction, taking a postcondition to a precondition.

In a setting of quantum computation this translation back-and-forth $\rightleftarrows$ in (1) is associated with the different approaches of Heisenberg (logic-based, working backwards) and Schrödinger (state-based, working forwards), see *e.g.* [12]. In quantum foundations one speaks of the duality between states and effects (predicates). Since the above triangles first emerged in the context of semantics of quantum computation [14], they are sometimes referred to as 'state-and-effect' triangles.

In certain cases the adjunction $\rightleftarrows$ in (1) forms – or may be restricted to – an equivalence of categories, yielding a duality situation. It shows the importance of duality theory in program semantics and logic; this topic has a long history, going back to [1].

In [14] it is shown that in the presence of relatively weak structure in a category $\mathbf{B}$, a diagram of the form (1) can be formed, with $\mathbf{B}$ as base category of computations, with predicates forming effect modules (see below) and with states forming convex sets. A category with this relatively weak structure is now called an *effectus*, see [20].

The main contribution of this paper is a "new" way of generating state-and-effect triangles, namely from adjunctions. We write the word 'new' between quotes, because the underlying category theory uses a famous of result of Jon Beck, and is not new at all. What the paper contributes is mainly a new perspective: it reorganises the work of Beck in such a way that an appropriate triangle appears, see Section 2. The rest of the paper is devoted to illustrations of this recipe for triangles. These examples are either of a Boolean or a probabilistic nature, see Sections 3 and 4 respectively. The Boolean examples are all obtained from an adjunction using "homming into $\{0,1\}$", whereas the probabilistic (quantitative) examples all arise from "homming into $[0,1]$", where $[0,1]$ is the unit interval of probabilities.

The series of examples in this paper involves many mathematical structures, ranging from Boolean algebras to compact Hausdorff spaces and C^*-algebras. It is impossible to explain all these notions in detail here. Hence the reader is assumed to be reasonably familiar with these structures. It does not matter so much if some of the examples involve unfamiliar mathematical notions. The structure of these sections 3 and 4 is clear enough, and it does not matter if some of the examples are skipped.

An exception is made for the notions of effect algebra and effect module. They are explicitly explained (briefly) in the beginning of Section 4 because they play such a prominent role in quantitative logic.

The examples involve many adjunctions that are known in the literature. Here they are displayed in triangle form. In several cases monads arise that are familiar in coalgebraic research, like the neighbourhood monad $\mathcal{N}$ in Subsection 3.1, the monotone neighbourhood monad $\mathcal{M}$ in Subsection 3.2, the infinite distribution monad $\mathcal{D}_\infty$ in Subsection 4.4, and the Giry monad $\mathcal{G}$ in Subsection 4.5. Also we will see several examples where we have pushed the recipe to a limit, and where the monad involved is simply the identity.

2 A basic result about monads

We assume that the reader is familiar with the categorical concept of a monad T, and with its double role, describing a form of computation, via the associated Kleisli category $\mathcal{K}\ell(T)$, and describing algebraic structure, via the category $\mathcal{EM}(T)$ of Eilenberg-Moore algebras.

The following result is a basic part of the theory of monads, see *e.g.* [3, Prop. 3.15 and Exercise (KEM)] or [22, Prop. 6.5 and 6.7] or [2, Thm. 20.42], and describes the initiality and finality of the Kleisli category and Eilenberg-Moore category as 'adjunction resolutions' giving rise to a monad.

▶ **Theorem 1.** *Consider an adjunction $F \dashv G$ with induced monad $T = GF$. Then there are 'comparison' functors $\mathcal{K}\ell(T) \to \mathbf{A} \to \mathcal{EM}(T)$ in a diagram:*

$$\begin{tikzcd}
\mathcal{K}\ell(T) \arrow[r, "L"] \arrow[dr, bend right=10] \arrow[dr, phantom, "\top"] & \mathbf{A} \arrow[r, bend left, "K"] \arrow[r, phantom, "\top"] \arrow[d, bend left, "G"] \arrow[d, phantom, "\dashv"] & \mathcal{EM}(T) \arrow[l, dotted, bend left, "M"] \arrow[dl, bend left=10] \arrow[dl, phantom, "\perp"] \\
& \mathbf{B} \arrow[ul, bend right=10] \arrow[u, bend left, "F"] \arrow[ur, bend left=10] \arrow[loop below, "T=GF"] &
\end{tikzcd} \tag{2}$$

where the functor $L\colon \mathcal{K}\ell(T) \to \mathbf{A}$ is full and faithful.

In case the category $\mathbf{A}$ has coequalisers (of reflexive pairs), then K has a left adjoint M, as indicated via the dotted arrow, satisfying $MKL \cong L$.

The famous monadicity theorem of Jon Beck gives conditions that guarantee that the functor $K\colon \mathbf{A} \to \mathcal{EM}(T)$ is an equivalence of categories, so that objects of $\mathbf{A}$ are algebras. The existence of the left adjoint M is the part of this theorem that we use in the current setting. Other (unused) parts of Beck's theorem require that the functor G preserves and reflects coequalisers of reflexive pairs. For convenience we include a proof sketch.

Proof. Define $L(X) = F(X)$ and $L\big(X \xrightarrow{f} GF(Y)\big) = \varepsilon_{F(Y)} \circ F(f)\colon F(X) \to F(Y)$. This functor L is full and faithful because there is a bijective adjoint correspondence:

$$\begin{array}{c} F(X) \longrightarrow F(Y) \\ \hline\hline X \longrightarrow GF(Y) = T(Y) \end{array}$$

The functor $K\colon \mathbf{A} \to \mathcal{EM}(T)$ is defined as:

$$K(A) \quad = \quad \begin{pmatrix} GFG(A) \\ \big\downarrow {\scriptstyle G(\varepsilon_A)} \\ G(A) \end{pmatrix} \qquad\quad \text{and} \quad K\big(A \xrightarrow{f} B\big) \quad = \quad G(f).$$

We leave it to the reader to see that K is well-defined. For a Kleisli map $f\colon X \to T(Y)$ the map $KL(f)$ is Kleisli extension:

$$KL(f) \quad = \quad G(\varepsilon_{F(Y)} \circ F(f)) \quad = \quad \mu_Y \circ T(f) : T(X) \longrightarrow T(Y).$$

Assume that the category $\mathbf{A}$ has coequalisers. For an algebra $a\colon T(X) \to X$ let $M(X, a)$ be the (codomain of the) coequaliser in:

$$\begin{tikzcd}
FGF(X) \arrow[r, shift left, "F(a)"] \arrow[r, shift right, "\varepsilon_{F(X)}"'] & F(X) \arrow[r, two heads, "c"] & M(X,a)
\end{tikzcd}$$

It is not hard to see that there is a bijective correspondence:

$$\begin{array}{cc} M(X,a) \xrightarrow{\ f\ } A & \quad \text{in } \mathbf{A} \\ \hline\hline \begin{pmatrix} T(X) \\ \downarrow a \\ X \end{pmatrix} \xrightarrow[g]{} \begin{pmatrix} TG(A) \\ \downarrow G(\varepsilon_A) \\ G(A) \end{pmatrix} = K(A) & \quad \text{in } \mathcal{EM}(T) \end{array}$$

What remains is to show $MKL \cong L$. This follows because for each $X \in \mathbf{B}$, the following diagram is a coequaliser in $\mathbf{A}$.

$$\begin{tikzcd} FGFGF(X) \arrow[r, shift left, "F(\mu_X) = FG(\varepsilon_{F(X)})"] \arrow[r, shift right, "\varepsilon_{FGF(X)}"'] & FGF(X) \arrow[r, two heads, "\varepsilon_{F(X)}"] & F(X) \end{tikzcd}$$

Hence the codomain $MKL(X)$ of the coequaliser of $FKL(X) = FG(\varepsilon_{F(X)})$ and the counit map $\varepsilon_{FGF(X)}$ is isomorphic to $F(X) = L(X)$. Proving naturality of $MKL \cong L$ (wrt. Kleisli maps) is a bit of work, but is essentially straightforward. ◀

An essential 'aha moment' underlying this paper is that the above result can be massaged into triangle form. This is what happens in the next result, to which we will refer as the 'triangle corollary'. It is the 'recipe' that occurs in the title of this paper.

▶ **Corollary 2.** *Consider an adjunction $F \dashv G$, where F is a functor $\mathbf{B} \to \mathbf{A}$, the category $\mathbf{A}$ has coequalisers, and the induced monad on $\mathbf{B}$ is written as $T = GF$. Diagram* (2) *then gives rise to a triangle as below, where both up-going functors are full and faithful.*

$$\begin{tikzcd} \mathbf{A} \arrow[rr, bend left, "K"] \arrow[rr, phantom, "\top"] & & \mathcal{EM}(T) \arrow[ll, bend left, "M"] \\ & \mathcal{K}\ell(T) \arrow[ul, "\mathrm{Pred} = L"] \arrow[ur, "KL = \mathrm{Stat}"'] & \end{tikzcd} \tag{3}$$

This triangle commutes, trivially from left to right, and up-to-isomorphism from right to left, since $MKL \cong L$. In this context we refer to the functor L as the 'predicate' functor Pred, *and to the functor KL as the 'states' functor* Stat.

The remainder of the paper is devoted to instances of this triangle corollary. In each of these examples the category $\mathbf{A}$ will be of the form $\mathbf{P}^{\mathrm{op}}$, where $\mathbf{P}$ is a category of predicates (with equalisers). The full and faithfulness of the functors $\mathrm{Pred}\colon \mathcal{K}\ell(T) \to \mathbf{P}^{\mathrm{op}}$ and $\mathrm{Stat}\colon \mathcal{K}\ell(T) \to \mathcal{EM}(T)$ means that there are bijective correspondences between:

$$\begin{array}{c} X \xrightarrow{\text{computations}} T(Y) \\ \hline\hline \mathrm{Pred}(Y) \xrightarrow[\text{predicate transformers}]{} \mathrm{Pred}(X) \end{array} \qquad \begin{array}{c} X \xrightarrow{\text{computations}} T(Y) \\ \hline\hline \mathrm{Stat}(X) \xrightarrow[\text{state transformers}]{} \mathrm{Stat}(Y) \end{array} \tag{4}$$

Since $\mathrm{Stat}(X) = T(X)$, the correspondence on the right is given by Kleisli extension, sending a map $f\colon X \to T(Y)$ to $\mu \circ T(f)\colon T(X) \to T(Y)$. This bijective correspondence on the right is a categorical formality. But the correspondence on the left is much more interesting, since it precisely describes to which kind of predicate transformers (preserving which structure) computations correspond. This will be illustrated below.

Aside: as discussed in [14], the predicate functor $\mathrm{Pred}\colon \mathcal{K}\ell(T) \to \mathbf{A}$ is in some cases an *enriched* functor, preserving additional structure that is of semantical/logical relevance. For instance, operations on programs, like ∪ for non-deterministic sum, may be expressed as structure on Kleisli homsets. Preservation of this structure by the functor Pred gives the logical rules for dealing with such structure in weakest precondition computations. These enriched aspects will not be elaborated in the current context.

3 Boolean examples

We split our series of examples in two parts, namely into Boolean and probabilistic examples. The Boolean ones are obtained via adjunctions that involve 'homming into 2', where $2 = \{0, 1\}$ is the 2-element set of Booleans. The probabilistic (aka. quantitative) examples in the next section are obtained via 'homming into $[0, 1]$', where $[0, 1] \subseteq \mathbb{R}$ is the unit interval of probabilities.

3.1 Sets and sets

We will present examples in the following manner, in three stages.

$$\begin{tikzcd} \mathbf{Sets}^{\mathrm{op}} \arrow[d, bend left, "\mathcal{P}=\mathrm{Hom}(-{,}2)"] \arrow[d, phantom, "\dashv"] \\ \mathbf{Sets} \arrow[u, bend left, "\mathcal{P}=\mathrm{Hom}(-{,}2)"] \arrow[loop below, "\mathcal{N}=\mathcal{P}\mathcal{P}"] \end{tikzcd}$$

$$\begin{array}{c} \mathcal{P}(X) \xrightarrow{\mathbf{Sets}^{\mathrm{op}}} Y \\ \hline\hline Y \xrightarrow{\mathbf{Sets}} \mathcal{P}(X) \\ \hline\hline X \xrightarrow[\mathbf{Sets}]{} \mathcal{P}(Y) \end{array}$$

$$\begin{tikzcd} \mathbf{Sets}^{\mathrm{op}} \arrow[rr, bend left, "\top"] & & \mathcal{EM}(\mathcal{N}) = \mathbf{CABA} \arrow[ll, bend left] \\ & \mathcal{K}\ell(\mathcal{N}) \arrow[ul, "\mathrm{Pred}"] \arrow[ur, "\mathrm{Stat}"'] & \end{tikzcd}$$

On the left we describe the adjunction that forms the basis for the example at hand, together with the induced monad. In this case we have the familiar fact that the powerset functor $\mathcal{P}$ is adjoint to itself, as indicated. The induced double-powerset monad $\mathcal{P}\mathcal{P}$ is known in the coalgebra/modal logic community as the neighbourhood monad $\mathcal{N}$, because its coalgebras are related to neighbourhood frames in modal logic.

In the middle the bijective correspondence is described that forms the basis of the adjunction. In this case there is the obvious correspondence between functions $Y \to \mathcal{P}(X)$ and functions $X \to \mathcal{P}(Y)$ – which are all relations on $X \times Y$.

On the right the result is shown of applying the triangle corollary 2 to the adjunction on the left. The full and faithfulness of the predicate functor $\mathrm{Pred}\colon \mathcal{K}\ell(\mathcal{N}) \to \mathbf{Sets}^{\mathrm{op}}$ plays an important role in the approach to coalgebraic dynamic logic in [11], relating coalgebras $X \to \mathcal{N}(X)$ to predicate transformer functions $\mathcal{P}(X) \to \mathcal{P}(X)$, going in the opposite direction. The category $\mathcal{EM}(\mathcal{N})$ of Eilenberg-Moore algebras of the neighbourhood monad $\mathcal{N}$ is the category **CABA** of complete atomic Boolean algebras (see *e.g.* [28]). The adjunction $\mathbf{Sets}^{\mathrm{op}} \rightleftarrows \mathcal{EM}(\mathcal{N})$ is thus an equivalence.

3.2 Sets and posets

We now restrict the adjunction in the previous subsection to posets.

$$\begin{tikzcd} \mathbf{PoSets}^{\mathrm{op}} \arrow[d, bend left, "\mathrm{Up}=\mathrm{Hom}(-{,}2)"] \arrow[d, phantom, "\dashv"] \\ \mathbf{Sets} \arrow[u, bend left, "\mathcal{P}=\mathrm{Hom}(-{,}2)"] \arrow[loop below, "\mathcal{M}=\mathrm{Up}\mathcal{P}"] \end{tikzcd}$$

$$\begin{array}{c} Y \xrightarrow{\mathbf{PoSets}} \mathcal{P}(X) \\ \hline\hline X \xrightarrow[\mathbf{Sets}]{} \mathrm{Up}(Y) \end{array}$$

$$\begin{tikzcd} \mathbf{PoSets}^{\mathrm{op}} \arrow[rr, bend left, "\top"] & & \mathcal{EM}(\mathcal{M}) = \mathbf{CDL} \arrow[ll, bend left] \\ & \mathcal{K}\ell(\mathcal{M}) \arrow[ul, "\mathrm{Pred}"] \arrow[ur, "\mathrm{Stat}"'] & \end{tikzcd}$$

The functor $\mathrm{Up}\colon \mathbf{PoSets}^{\mathrm{op}} \to \mathbf{Sets}$ sends a poset Y to the collection of upsets $U \subseteq Y$, satisfying $y \geq x \in U$ implies $y \in U$. These upsets can be identified with monotone maps $p\colon Y \to 2$, namely as $p^{-1}(1)$.

Notice that this time there is a bijective correspondence between computations $X \to \mathcal{M}(Y) = \mathrm{Up}\mathcal{P}(Y)$ and *monotone* predicate transformers $\mathcal{P}(Y) \to \mathcal{P}(X)$. This fact is used in [11]. The algebras of the monad $\mathcal{M}$ are completely distributive lattices, see [24] and [21, I, Prop. 3.8].

3.3 Sets and meet-semilattices

We now restrict the adjunction further to meet semilattices, that is, to posets with finite meets $\wedge, \top$.

$$\begin{tikzcd}
\mathbf{MSL}^{\mathrm{op}} \arrow[d, bend left, "\mathrm{Hom}(-,2)"] \\
\mathbf{Sets} \arrow[u, bend left, "\mathcal{P}=\mathrm{Hom}(-,2)"] \arrow[u, phantom, "\dashv"] \arrow[loop below, "\mathcal{F}=\mathbf{MSL}(\mathcal{P}(-),2)"]
\end{tikzcd}
\qquad
\begin{array}{c}
Y \xrightarrow{\mathbf{MSL}} \mathcal{P}(X) \\ \hline\hline
X \xrightarrow[\mathbf{Sets}]{} \mathbf{MSL}(Y,2)
\end{array}
\qquad
\begin{tikzcd}
\mathbf{MSL}^{\mathrm{op}} \arrow[rr, bend left] \arrow[rr, phantom, "\top"] & & \mathcal{EM}(\mathcal{F}) = \mathbf{CCL} \arrow[ll, bend left] \\
& \mathcal{K}\ell(\mathcal{F}) \arrow[ul, "\mathrm{Pred}"] \arrow[ur, "\mathrm{Stat}"'] &
\end{tikzcd}$$

Morphisms in the category **MSL** of meet semilattices preserve the meet $\wedge$ and the top element $\top$ (and hence the order too). For $Y \in \mathbf{MSL}$ one can identify a map $Y \to 2$ with a *filter* of Y, that is, with an upset $U \subseteq Y$ closed under $\wedge, \top$.

The resulting monad $\mathcal{F}(X) = \mathbf{MSL}(\mathcal{P}(X), 2)$ gives the filters in $\mathcal{P}(X)$. This monad is thus called the *filter monad*. In [29] it is shown that its category of algebras $\mathcal{EM}(\mathcal{F})$ is the category **CCL** of continuous complete lattices, that is, of complete lattices in which each element x is the (directed) join $x = \bigvee\{y \mid y \ll x\}$ of the elements way below it.

3.4 Sets and Boolean algebras

We further restrict the adjunction to the category **BA** of Boolean algebras.

$$\begin{tikzcd}
\mathbf{BA}^{\mathrm{op}} \arrow[d, bend left, "\mathrm{Hom}(-,2)"] \\
\mathbf{Sets} \arrow[u, bend left, "\mathcal{P}=\mathrm{Hom}(-,2)"] \arrow[u, phantom, "\dashv"] \arrow[loop below, "\mathcal{U}=\mathbf{BA}(\mathcal{P}(-),2)"]
\end{tikzcd}
\qquad
\begin{array}{c}
Y \xrightarrow{\mathbf{BA}} \mathcal{P}(X) \\ \hline\hline
X \xrightarrow[\mathbf{Sets}]{} \mathbf{BA}(Y,2)
\end{array}
\qquad
\begin{tikzcd}
\mathbf{BA}^{\mathrm{op}} \arrow[rr, bend left] \arrow[rr, phantom, "\top"] & & \mathcal{EM}(\mathcal{U}) = \mathbf{CH} \arrow[ll, bend left] \\
& \mathcal{K}\ell(\mathcal{U}) \arrow[ul, "\mathrm{Pred}"] \arrow[ur, "\mathrm{Stat}"'] &
\end{tikzcd}$$

The functor $\mathrm{Hom}(-,2)\colon \mathbf{BA}^{\mathrm{op}} \to \mathbf{Sets}$ sends a Boolean algebra Y to the set $\mathbf{BA}(Y,2)$ of Boolean algebra maps $Y \to 2$. They can be identified with *ultrafilters* of Y. The resulting monad $\mathcal{U} = \mathbf{BA}(\mathcal{P}(-), 2)$ is the *ultrafilter monad*, sending a set X to the BA-maps $\mathcal{P}(X) \to 2$, or equivalently, the ultrafilters of $\mathcal{P}(X)$.

An important result of Manes (see [23], and also [21, III, 2.4]) says that the category of Eilenberg-Moore algebras of the ultrafilter monad $\mathcal{U}$ is the category **CH** of compact Hausdorff spaces. This adjunction $\mathbf{BA}^{\mathrm{op}} \rightleftarrows \mathbf{CH}$ restricts to an equivalence $\mathbf{BA}^{\mathrm{op}} \simeq \mathbf{Stone}$ called Stone duality, where $\mathbf{Stone} \hookrightarrow \mathbf{CH}$ is the full subcategory of Stone spaces – in which each open subset is the union of the clopens contained in it.

3.5 Sets and complete Boolean algebras

We can restrict the adjunction $\mathbf{BA}^{\mathrm{op}} \rightleftarrows \mathbf{Sets}$ from the previous subsection to an adjunction $\mathbf{CBA}^{\mathrm{op}} \rightleftarrows \mathbf{Sets}$ between *complete* Boolean algebras and sets. The resulting monad on **Sets** is of the form $X \mapsto \mathbf{CBA}(\mathcal{P}(X), 2)$. But here we hit a wall, since this monad is the identity.

▶ **Lemma 3.** *For each set X the unit map $\eta\colon X \to \mathbf{CBA}(\mathcal{P}(X), 2)$, given by $\eta(x)(U) = 1$ iff $x \in U$, is an isomorphism.*

Proof. Let $h\colon \mathcal{P}(X) \to 2$ be a map of complete Boolean algebras, preserving the BA-structure and all joins (unions). Since each subset $U \in \mathcal{P}(X)$ can be described as union of singletons, the function h is determined by its values $h(\{x\})$ for $x \in X$. We have $1 = h(X) = \bigcup_{x\in X} h(\{x\})$. Hence $h(\{x\}) = 1$ for some $x \in X$. But then $h(X - \{x\}) = h(\neg\{x\}) = \neg h(\{x\}) = \neg 1 = 0$. This implies $h(\{x'\}) = 0$ for each $x' \neq x$. But then $h = \eta(x)$. ◀

4 Probabilistic examples

The next series of examples starts from adjunctions that are obtained by homming into the unit interval $[0,1]$. The quantitative logic that belongs to these examples is given in terms of effect modules. These can be seen as "probabilistic vector spaces", involving scalar multiplication with scalars from the unit interval $[0,1]$, instead of from $\mathbb{R}$ or $\mathbb{C}$. We provide a crash course for these structures, and refer to [17, 15] or [7] for more information.

A partial commutative monoid (PCM) consists of a set M with a partial binary operation $\ovee$ and a zero element $0 \in M$. The operation $\ovee$ is commutative and associative, in an appropriate partial sense. One writes $x \perp y$ if $x \ovee y$ is defined.

An *effect algebra* is a PCM with an orthocomplement $(-)^\perp$, so that $x \ovee x^\perp = 1$, where $1 = 0^\perp$, and $x \perp 1$ implies $x = 0$. An effect algebra is automatically a poset, via the definition $x \leq y$ iff $x \ovee z = y$ for some z. The main example is the unit interval $[0,1]$, with $x \perp y$ iff $x + y \leq 1$, and in that case $x \ovee y = x + y$; the orthocomplement is $x^\perp = 1 - x$. A map of effect algebras $f\colon E \to D$ is a function that preserves 1 and $\ovee$, if defined. We write $\mathbf{EA}$ for the resulting category. Each Boolean algebra is an effect algebra, with $x \perp y$ iff $x \wedge y = 0$, and in that case $x \ovee y = x \vee y$. This yields a functor $\mathbf{BA} \to \mathbf{EA}$, which is full and faithful.

An *effect module* is an effect algebra E with an action $[0,1] \times E \to E$ that preserves $\ovee, 0$ in each argument separately. A map of effect modules f is a map of effect algebras that preserves scalar multiplication: $f(r \cdot x) = r \cdot f(x)$. We thus get a subcategory $\mathbf{EMod} \hookrightarrow \mathbf{EA}$. For each set X, the set $[0,1]^X$ of fuzzy predicates on X is an effect module, with $p \perp q$ iff $p(x)+q(x) \leq 1$ for all $x \in X$, and in that case $(p \ovee q)(x) = p(x) + q(x)$. Orthocomplement is given by $p^\perp(x) = 1 - p(x)$ and scalar multiplication by $r \cdot p \in [0,1]^X$, for $r \in [0,1]$ and $p \in [0,1]^X$, by $(r \cdot p)(x) = r \cdot p(x)$. This assignment $X \mapsto [0,1]^X$ yields a functor $\mathbf{Sets} \to \mathbf{EMod}^{\mathrm{op}}$ that will be used below. Important examples of effect modules arise in quantum logic. For instance, for each Hilbert space $\mathcal{H}$, the set $\mathcal{E}f(\mathcal{H}) = \{A\colon \mathcal{H} \to \mathcal{H} \mid 0 \leq A \leq \mathrm{id}\}$ of effects is an effect module. More generally, for a (unital) C^*-algebra A, the set of effects $[0,1]_A = \{a \in A \mid 0 \leq a \leq 1\}$ is an effect module. In [8] it is shown that taking effects yields a full and faithful functor:

$$\mathbf{Cstar}_{\mathrm{PU}} \xrightarrow{[0,1]_{(-)}} \mathbf{EMod} \tag{5}$$

Here we write $\mathbf{Cstar}_{\mathrm{PU}}$ for the category of C^*-algebras with positive unital maps.

An *MV-algebra* [5] can be understood as a 'commutative' effect algebra. It is an effect algebra with a join $\vee$, and thus also a meet $\wedge$, via De Morgan, in which the equation $(x \vee y)^\perp \ovee x = y^\perp \ovee (x \wedge y)$ holds. There is a subcategory $\mathbf{MVA} \hookrightarrow \mathbf{EA}$ with maps additionally preserving joins $\vee$ (and hence also $\wedge$). Within an MV-algebra one can define (total) addition and subtraction operations as $x + y = x \ovee (x^\perp \wedge y)$ and $x - y = (x^\perp + y)^\perp$. The unit interval $[0,1]$ is an MV-algebra, in which $+$ and $-$ are truncated (to 1 or 0), if needed.

There is a category $\mathbf{MVMod}$ of *MV-modules*, which are MV-algebras with $[0,1]$-scalar multiplication. Thus $\mathbf{MVMod}$ is twice a subcategory in: $\mathbf{MVA} \hookleftarrow \mathbf{MVMod} \hookrightarrow \mathbf{EMod}$.

The effect module $[0,1]^X$ of fuzzy predicates is an MV-module. For a commutative C^*-algebra A the set of effects $[0,1]_A$ is an MV-module. In fact there is a full and faithful functor:

$$\mathbf{CCstar}_{\mathrm{MIU}} \xrightarrow{[0,1]_{(-)}} \mathbf{MVMod} \tag{6}$$

where $\mathbf{CCstar}_{\mathrm{MIU}}$ is the category of commutative C^*-algebras, with MIU-maps, preserving multiplication, involution and unit (aka. $*$-homomorphisms).

Having seen this background information we continue our series of examples.

4.1 Sets and effect modules

As noted above, fuzzy predicates yield a functor $\mathbf{Sets} \to \mathbf{EMod}^{\mathrm{op}}$. This functor involves homming into $[0,1]$, and has an adjoint that is used as starting point for several variations.

$$\begin{tikzcd}
\mathbf{EMod}^{\mathrm{op}} \arrow[d, bend left, "\mathrm{Hom}(-,[0,1])"] \arrow[d, phantom, "\dashv"] \\
\mathbf{Sets} \arrow[u, bend left, "\mathrm{Hom}(-,[0,1])"] \arrow[loop below, "\mathcal{E}=\mathbf{EMod}([0,1]^{(-)},[0,1])"]
\end{tikzcd}
\qquad
\begin{array}{c}
Y \xrightarrow{\mathbf{EMod}} [0,1]^X \\
\hline\hline
X \xrightarrow[\mathbf{Sets}]{} \mathbf{EMod}(Y,[0,1])
\end{array}
\qquad
\begin{tikzcd}
\mathbf{EMod}^{\mathrm{op}} \arrow[rr, bend left] \arrow[rr, phantom, "\top"] & & \mathcal{EM}(\mathcal{E}) = \mathbf{CCH}_{\mathrm{sep}} \arrow[ll, bend left] \\
& \mathcal{K}\ell(\mathcal{E}) \arrow[ul, "\mathrm{Pred}"] \arrow[ur, "\mathrm{Stat}"'] &
\end{tikzcd}$$

The induced monad $\mathcal{E}$ is the *expectation* monad introduced in [16]. It can be understood as an extension of the (finite probability) distribution monad $\mathcal{D}$, since $\mathcal{E}(X) \cong \mathcal{D}(X)$ if X is a finite set. The triangle corollary on the right says in particular that Kleisli maps $X \to \mathcal{E}(Y)$ are in bijective correspondence with effect module maps $[0,1]^Y \to [0,1]^X$ acting as predicate transformers, on fuzzy predicates.

The category of algebras $\mathcal{EM}(\mathcal{E})$ of the expectation monad is the category $\mathbf{CCH}_{\mathrm{sep}}$ of convex compact Hausdorff spaces, with a separation condition (see [16, 18] for details). State spaces in quantum computing are typically such convex compact Hausdorff spaces.

Using the full and faithfulness of the functor $[0,1]_{(-)}\colon \mathbf{Cstar}_{\mathrm{PU}} \to \mathbf{EMod}$ from (5), the expectation monad can alternatively be described in terms of the states of the commutative C^*-algebra $\ell^\infty(X)$ of bounded functions $X \to \mathbb{C}$, via:

$$\begin{aligned}
\mathrm{Stat}(\ell^\infty(X)) \overset{\mathrm{def}}{=} \mathbf{Cstar}_{\mathrm{PU}}\big(\ell^\infty(X),\mathbb{C}\big) &\overset{(5)}{\cong} \mathbf{EMod}\big([0,1]_{\ell^\infty(X)},[0,1]_{\mathbb{C}}\big) \\
&= \mathbf{EMod}\big([0,1]^X,[0,1]\big) \;=\; \mathcal{E}(X).
\end{aligned} \tag{7}$$

In this way one obtains the result from [8] that there is a full & faithful functor:

$$\mathcal{K}\ell(\mathcal{E}) \longrightarrow \big(\mathbf{CCstar}_{\mathrm{PU}}\big)^{\mathrm{op}} \tag{8}$$

embedding the Kleisli category $\mathcal{K}\ell(\mathcal{E})$ of the expectation monad into commutative C^*-algebras with positive unital maps. On objects this functor (8) is given by $X \mapsto \ell^\infty(X)$.

4.2 Compact Hausdorff spaces and effect modules

In the previous example we have used the *set* $\mathbf{EMod}(E,[0,1])$ of effect module maps $E \to [0,1]$, for an effect module E. It turns out that this homset has much more structure: it is a compact Hausdorff space. The reason is that the unit interval $[0,1]$ is compact Hausdorff, and so the function space $[0,1]^E$ too, by Tychonoff. The homset $\mathbf{EMod}(E,[0,1]) \hookrightarrow [0,1]^E$

can be described via a closed subset of maps satisfying the effect module map requirements. Hence $\mathbf{EMod}(E, [0,1])$ is compact Hausdorff itself. We thus obtain the following situation.

$$\begin{tikzcd}
\mathbf{EMod}^{\mathrm{op}} \arrow[d, bend right, "\mathrm{Hom}(-{,}[0,1])"'] \arrow[d, phantom, "\dashv"] \\
\mathbf{CH} \arrow[u, bend right, "\mathrm{Hom}(-{,}[0,1])"'] \arrow[loop below, "\mathcal{R}=\mathbf{EMod}(\mathrm{C}(-{,}[0,1]){,}[0,1])"]
\end{tikzcd}
\qquad
\begin{array}{c}
Y \xrightarrow{\mathbf{EMod}} \mathrm{C}(X, [0,1]) \\
\hline\hline
X \xrightarrow[\mathbf{CH}]{} \mathbf{EMod}(Y, [0,1])
\end{array}
\qquad
\begin{tikzcd}
\mathbf{EMod}^{\mathrm{op}} \arrow[rr, bend left, ""] \arrow[rr, phantom, "\top"] & & \mathcal{EM}(\mathcal{R}) = \mathbf{CCH}_{\mathrm{sep}} \arrow[ll, bend left, ""] \\
& \mathcal{K}\ell(\mathcal{R}) \arrow[ul, "\mathrm{Pred}"] \arrow[ur, "\mathrm{Stat}"'] &
\end{tikzcd}$$

For a compact Hausdorff space X, the subset $\mathrm{C}(X, [0,1]) \hookrightarrow [0,1]^X$ of continuous maps $X \to [0,1]$ is a (sub) effect module. The induced monad $\mathcal{R}(X) = \mathbf{EMod}\big(\mathrm{C}(X, [0,1]), [0,1]\big)$ is the *Radon* monad. Using the full & faithful functor (5) the monad can equivalently be described as $X \mapsto \mathrm{Stat}(\mathrm{C}(X))$, where $\mathrm{C}(X)$ is the commutative C^*-algebra of functions $X \to \mathbb{C}$. The monad occurs in [25] as part of a topological and domain-theoretic approach to information theory. The main result of [8] is the equivalence of categories

$$\mathcal{K}\ell(\mathcal{R}) \quad \simeq \quad \big(\mathbf{CCstar}_{\mathrm{PU}}\big)^{\mathrm{op}}$$

between the Kleisli category of this Radon monad $\mathcal{R}$ and the category of commutative C^*-algebras and positive unital maps. This shows how (commutative) C^*-algebras appear in state-and-effect triangles (see also [15]).

The algebras of the Radon monad are convex compact Hausdorff spaces (with separation), like for the expectation monad $\mathcal{E}$, see [9] for details.

4.3 Compact Hausdorff spaces and MV-modules

The adjunction $\mathbf{EMod}^{\mathrm{op}} \rightleftarrows \mathbf{CH}$ can be restricted to an adjunction $\mathbf{MVMod}^{\mathrm{op}} \rightleftarrows \mathbf{CH}$, involving MV-modules instead of effect modules. This can be done since continuous functions $X \to [0,1]$ are appropriately closed under joins $\vee$, and thus form an MV-module. Additionally, for an MV-module E, the MV-module maps $E \to [0,1]$ form a compact Hausdorff space (using the same argument as in the previous subsection).

Via this restriction to an adjunction $\mathbf{MVMod}^{\mathrm{op}} \rightleftarrows \mathbf{CH}$ we hit a wall again.

▶ **Lemma 4.** *For a compact Hausdorff space X, the unit $\eta\colon X \to \mathbf{MVMod}\big(\mathrm{C}(X, [0,1]), [0,1]\big)$, given by $\eta(x)(p) = p(x)$, is an isomorphism in* $\mathbf{CH}$.

This result can be understood as part of the Yosida duality for Riesz spaces. It is well-known in the MV-algebra community, but possibly not precisely in this form. For convenience, we include a proof.

Proof. We only show that the unit η is an isomorphism, not that it is also a homeomorphism. Injectivity is immediate by Urysohn. For surjectivity, we first establish the following two auxiliary results.

1. For each $p \in \mathrm{C}(X, [0,1])$ and $\omega \in \mathbf{MVMod}\big(\mathrm{C}(X, [0,1]), [0,1]\big)$, if $\omega(p) = 0$, then there is an $x \in X$ with $p(x) = 0$.
   If not, then $p(x) > 0$ for all $x \in X$. Hence there is an inclusion $X \subseteq \bigcup_{r>0} p^{-1}\big((r, 1]\big)$. By compactness there are finitely many r_i with $X \subseteq \bigcup_i p^{-1}\big((r_i, 1]\big)$. Thus for $r = \bigwedge_i r_i > 0$ we have $p(x) > r$ for all $x \in X$. Find an $n \in \mathbb{N}$ with $n \cdot r \geq 1$. The n-fold sum $n \cdot p$ in the MV-module $\mathrm{C}(X, [0,1])$ then satisfies $p(x) = 1$ for all x, so that $n \cdot p = 1$ in $\mathrm{C}(X, [0,1])$. But now we get a contradiction: $1 = \omega(1) = \omega(n \cdot p) = n \cdot \omega(p) = 0$.

2. For each finite collection of maps $p_1, \ldots, p_n \in \mathrm{C}(X,[0,1])$ and for each function $\omega \in \mathbf{MVMod}\big(\mathrm{C}(X,[0,1]),[0,1]\big)$ there is an $x \in X$ with $\omega(p_i) = p_i(x)$ for all $1 \le i \le n$.
   For the proof, define $p \in \mathrm{C}(X,[0,1])$ using the MV-structure of $\mathrm{C}(X,[0,1])$ as:
   $$p \quad = \quad \bigvee_i \big(p_i - \omega(p_i)\cdot 1\big) \vee \big(\omega(p_i)\cdot 1 - p_i\big).$$
   Since the state $\omega\colon \mathrm{C}(X,[0,1]) \to [0,1]$ preserves the MV-structure we get in $[0,1]$:
   $$\omega(p) \quad = \quad \bigvee_i \big(\omega(p_i) - \omega(p_i)\cdot 1\big) \vee \big(\omega(p_i)\cdot 1 - \omega(p_i)\big) \quad = \quad 0.$$
   Hence by the previous point there is an $x \in X$ with $p(x) = 0$. But then $p_i(x) = \omega(p_i)$, as required.

Now we can prove surjectivity of the unit map $\eta\colon X \to \mathbf{MVMod}\big(\mathrm{C}(X,[0,1]),[0,1]\big)$. Let $\omega\colon \mathrm{C}(X,[0,1]) \to [0,1]$ be an MV-module map. Define for each $p \in \mathrm{C}(X,[0,1])$ the subset $U_p = \{x \in X \mid \omega(p) \neq p(x)\}$. This subset $U_p \subseteq X$ is open since it can be written as $f^{-1}(\mathbb{R} - \{0\})$, for the continuous function $f(x) = p(x) - \omega(p)$.

Suppose towards a contradiction that $\omega \neq \eta(x)$ for all $x \in X$. Thus, for each $x \in X$ there is a $p \in \mathrm{C}(X,[0,1])$ with $\omega(p) \neq \eta(x)(p) = p(x)$. This means $X \subseteq \bigcup_p U_p$. By compactness of X there are finitely many $p_i \in \mathrm{C}(X,[0,1])$ with $X \subseteq \bigcup_i U_{p_i}$. The above second point however gives an $x \in X$ with $\omega(p_i) = p_i(x)$ for all i. But then $x \notin \bigcup_i U_{p_i}$. ◀

4.4 Sets and directed complete effect modules

In the remainder of this paper we shall consider effect modules with additional completeness properties (wrt. its standard order). Specifically, we consider ω-complete, and directed-complete effect modules. In the first case each ascending ω-chain $x_0 \le x_1 \le \cdots$ has a least upperbound $\bigvee_n x_n$; and in the second case each directed subset D has a join $\bigvee D$. We write the resulting subcategories as:

$$\mathbf{DcEMod} \hookrightarrow \omega\text{-}\mathbf{EMod} \hookrightarrow \mathbf{EMod}$$

where maps are required to preserve the relevant joins $\bigvee$.

We start with the directed-complete case. The adjunction $\mathbf{EMod}^{\mathrm{op}} \rightleftarrows \mathbf{Sets}$ from Subsection 4.1 can be restricted to an adjunction as on the left below.

$$\begin{tikzcd} \mathbf{DcEMod}^{\mathrm{op}} \arrow[d, bend left, "\mathrm{Hom}(-,[0,1])"] \arrow[d, phantom, "\dashv"] \\ \mathbf{Sets} \arrow[u, bend left, "\mathrm{Hom}(-,[0,1])"] \arrow[loop below, "\mathcal{E}_\infty=\mathbf{DcEMod}([0,1]^{(-)},[0,1])"] \end{tikzcd} \qquad \begin{array}{c} Y \xrightarrow{\mathbf{DcEMod}} [0,1]^X \\ \hline\hline X \xrightarrow[\mathbf{Sets}]{} \mathbf{DcEMod}(Y,[0,1]) \end{array} \qquad \begin{tikzcd} \mathbf{DcEMod}^{\mathrm{op}} \arrow[rr, bend left=10] \arrow[rr, phantom, "\top"] & & \mathcal{EM}(\mathcal{E}_\infty)=\mathbf{Conv}_\infty \arrow[ll, bend left=10] \\ & \mathcal{K}\!\ell(\mathcal{E}_\infty) \arrow[ul, "\mathrm{Pred}"] \arrow[ur, "\mathrm{Stat}"'] & \end{tikzcd}$$

The resulting monad $\mathcal{E}_\infty = \mathbf{DcEMod}\big([0,1]^{(-)},[0,1]\big)$ on **Sets** is in fact isomorphic[1] to the infinite (discrete probability) distribution monad $\mathcal{D}_\infty$. We recall, for a set X,

$$\mathcal{D}_\infty(X) \quad = \quad \{\omega\colon X \to [0,1] \mid \mathrm{supp}(\omega) \text{ is countable, and } \textstyle\sum_x \omega(x) = 1\}.$$

[1] This isomorphism $\mathcal{E}_\infty \cong \mathcal{D}_\infty$ in Proposition 5 is inspired by work of Robert Furber (PhD Thesis, forthcoming): he noticed the isomorphism $\mathrm{NStat}(\ell^\infty(X)) \cong \mathcal{D}_\infty(X)$ in (11), which is obtained here as a corollary to Proposition 5.

The subset $\mathrm{supp}(\omega) \subseteq X$ contains the elements $x \in X$ with $\omega(x) \neq 0$. The requirement in the definition of $\mathcal{D}_\infty(X)$ that $\mathrm{supp}(\omega)$ be countable is superfluous, since it follows from the requirement $\sum_x \omega(x) = 1$. Briefly, $\mathrm{supp}(\omega) \subseteq \bigcup_{n>0} X_n$, where $X_n = \{x \in X \mid \omega(x) > \frac{1}{n}\}$ contains at most $n-1$ elements (see *e.g.* [27, Prop. 2.1.2]).

▶ **Proposition 5.** *There is an isomorphism of monads $\mathcal{D}_\infty \cong \mathcal{E}_\infty$, where $\mathcal{E}_\infty$ is the monad induced by the above adjunction $\mathbf{DcEMod}^{\mathrm{op}} \rightleftarrows \mathbf{Sets}$.*

Proof. For a subset $U \subseteq X$ we write $\mathbf{1}_U \colon X \to [0,1]$ for the 'indicator' function, defined by $\mathbf{1}_U(x) = 1$ if $x \in U$ and $\mathbf{1}_U(x) = 0$ if $x \notin U$. We write $\mathbf{1}_x$ for $\mathbf{1}_{\{x\}}$. This function $\mathbf{1}_{(-)} \colon \mathcal{P}(X) \to [0,1]^X$ is a map of effect algebras that preserves all joins.

Let $h \in \mathcal{E}_\infty(X)$, so h is a Scott-continuous map of effect modules $h \colon [0,1]^X \to [0,1]$. Define $\overline{h} \colon X \to [0,1]$ as $\overline{h}(x) = h(\mathbf{1}_x)$. Notice that if $U \subseteq X$ is a finite subset, then:

$$1 \;=\; h(1) \;=\; h(\mathbf{1}_X) \;\geq\; h(\mathbf{1}_U) \;=\; h(\textstyle\bigovee_{x\in U} \mathbf{1}_x) \;=\; \bigovee_{x\in U} h(\mathbf{1}_x) \;=\; \bigovee_{x\in U} \overline{h}(x).$$

We can write X as directed union of its finite subsets, and thus also $\mathbf{1}_X = \bigvee\{\mathbf{1}_U \mid U \subseteq X \text{ finite}\}$. But then $\overline{h} \in \mathcal{D}_\infty(X)$, because h preserves directed joins:

$$1 \;=\; h(\mathbf{1}_X) \;=\; \textstyle\bigvee\{h(\mathbf{1}_U) \mid U \subseteq X \text{ finite}\} \;=\; \bigvee\{\sum_{x\in U} \overline{h}(x) \mid U \subseteq X \text{ finite}\} \;=\; \sum_{x\in X} \overline{h}(x).$$

Conversely, given $\omega \in \mathcal{D}_\infty(X)$ we define $\overline{\omega} \colon [0,1]^X \to [0,1]$ as $\overline{\omega}(p) = \sum_{x\in X} p(x) \cdot \omega(x)$. It is easy to see that $\overline{\omega}$ is a map of effect modules. It is a bit more challenging to see that it preserves directed joins $\bigvee_i p_i$, for $p_i \in [0,1]^X$.

First we write the countable support of ω as $\mathrm{supp}(\omega) = \{x_0, x_1, x_2, \ldots\} \subseteq X$ in such a way that $\omega(x_0) \geq \omega(x_1) \geq \omega(x_2) \geq \cdots$. We have $1 = \sum_{x\in X} \omega(x) = \sum_{n\in\mathbb{N}} \omega(x_n)$. Hence, for each $N \in \mathbb{N}$ we get:

$$\textstyle\sum_{n>N} \omega(x_n) \;=\; 1 - \sum_{n\leq N} \omega(x_n).$$

By taking the limit $N \to \infty$ on both sides we get:

$$\lim_{N\to\infty} \textstyle\sum_{n>N} \omega(x_n) \;=\; 1 - \displaystyle\lim_{N\to\infty} \textstyle\sum_{n\leq N} \omega(x_n) \;=\; 1 - \sum_{n\in\mathbb{N}} \omega(x_n) \;=\; 1-1 \;=\; 0.$$

We have to prove $\overline{\omega}(\bigvee_i p_i) = \bigvee_i \overline{\omega}(p_i)$. The non-trivial part is $(\leq)$. For each $N \in \mathbb{N}$ we have:

$$\begin{array}{rcl}
\overline{\omega}(\bigvee_i p_i) & = & \sum_{n\in\mathbb{N}} (\bigvee_i p_i)(x_n) \cdot \omega(x_n) \\
& = & \sum_{n\in\mathbb{N}} (\bigvee_i p_i(x_n)) \cdot \omega(x_n) \\
& = & \sum_{n\in\mathbb{N}} \bigvee_i p_i(x_n) \cdot \omega(x_n) \\
& = & \Big(\sum_{n\leq N} \bigvee_i p_i(x_n) \cdot \omega(x_n)\Big) + \Big(\sum_{n>N} \bigvee_i p_i(x_n) \cdot \omega(x_n)\Big) \\
& = & \Big(\bigvee_i \sum_{n\leq N} p_i(x_n) \cdot \omega(x_n)\Big) + \Big(\sum_{n>N} \bigvee_i p_i(x_n) \cdot \omega(x_n)\Big) \\
& \leq & \Big(\bigvee_i \sum_{n\leq N} p_i(x_n) \cdot \omega(x_n)\Big) + \Big(\sum_{n>N} \omega(x_n)\Big) \qquad \text{since } p_i(x) \in [0,1].
\end{array}$$

Hence we are done by taking the limit $N \to \infty$. Notice that we use that the join $\bigvee$ can be moved outside a finite sum. This works precisely because the join is taken over a directed set.

What remains is to show that these mappings $h \mapsto \overline{h}$ and $\omega \mapsto \overline{\omega}$ yield an isomorphism $\mathcal{D}_\infty(X) \cong \mathcal{E}_\infty(X)$, which is natural in X, and forms an isomorphism of monads. This is left to the interested reader. ◀

As a result, the Eilenberg-Moore category $\mathcal{EM}(\mathcal{E}_\infty)$ is isomorphic to $\mathcal{EM}(\mathcal{D}_\infty) = \mathbf{Conv}_\infty$, where $\mathbf{Conv}_\infty$ is the category of countably-convex sets X, in which convex sums $\sum_{n\in\mathbb{N}} r_n x_n$ exist, where $x_n \in X$ and $r_n \in [0,1]$ with $\sum_n r_n = 1$.

We briefly look at the relation with C^*-algebras (actually W^*-algebras), like in Subsection 4.1. We write $\mathbf{Wstar}_{\mathrm{NPU}}$ for the category of W^*-algebras with normal positive unital maps. The term 'normal' is used in the operator algebra community for what is called 'Scott-continuity' (preservation of directed joins) in the domain theory community. This means that taking effects yields a full and faithful functor:

$$\mathbf{Wstar}_{\mathrm{NPU}} \xrightarrow{[0,1]_{(-)}} \mathbf{DcEMod} \tag{9}$$

This is similar to the situation in (5) and (6). One could also use AW^*-algebras here. Next, there is now a full and faithful functor to the category of commutative W^*-algebras:

$$\mathcal{K}\ell(\mathcal{D}_\infty) \cong \mathcal{K}\ell(\mathcal{E}_\infty) \longrightarrow \mathbf{CWstar}_{\mathrm{NPU}} \tag{10}$$

On objects it is given by $X \mapsto \ell^\infty(X)$. This functor is full and faithful since there is a bijective correspondence:

$$\begin{array}{cl} \ell^\infty(X) \longrightarrow \ell^\infty(Y) & \qquad \text{in } \mathbf{CWstar}_{\mathrm{NPU}} \\ \hline\hline Y \longrightarrow \mathrm{NStat}(\ell^\infty(X)) \cong \mathcal{E}_\infty(X) \cong \mathcal{D}_\infty(X) & \qquad \text{in } \mathbf{Sets} \end{array}$$

where the isomorphism $\cong$ describing normal states is given, like in (7), by:

$$\begin{array}{rcl} \mathrm{NStat}(\ell^\infty(X)) \stackrel{\text{def}}{=} \mathbf{Wstar}_{\mathrm{NPU}}\big(\ell^\infty(X), \mathbb{C}\big) & \overset{(9)}{\cong} & \mathbf{DcEMod}\big([0,1]_{\ell^\infty(X)}, [0,1]_{\mathbb{C}}\big) \\ & = & \mathbf{DcEMod}\big([0,1]^X, [0,1]\big) \\ & = & \mathcal{E}_\infty(X) \\ & \cong & \mathcal{D}_\infty(X). \end{array} \tag{11}$$

4.5 Measurable spaces and ω-complete effect modules

In our final example we use an adjunction between effect modules and measurable spaces (instead of sets or compact Hausdorff spaces). We write **Meas** for the category of measurable spaces (X, Σ_X), where $\Sigma_X \subseteq \mathcal{P}(X)$ is the σ-algebra of measurable subsets, with measurable functions between them (whose inverse image maps measurable subsets to measurable subsets). We use the unit interval $[0,1]$ with its standard Borel σ-algebra (the least one that contains all the usual opens). A basic fact in this situation is that for a measurable space X, the set $\mathbf{Meas}(X, [0,1])$ of measurable functions $X \to [0,1]$ is an ω-effect module. The effect module structure is inherited via the inclusion $\mathbf{Meas}(X, [0,1]) \hookrightarrow [0,1]^X$. Joins of ascending ω-chains $p_0 \leq p_1 \leq \cdots$ exists, because the (pointwise) join $\bigvee_n p_n$ is a measurable function again. In this way we obtain a functor $\mathbf{Meas}(-, [0,1]) \colon \mathbf{Meas} \to \omega\text{-}\mathbf{EMod}^{\mathrm{op}}$.

In the other direction there is also a hom-functor $\omega\text{-}\mathbf{EMod}(-, [0,1]) \colon \omega\text{-}\mathbf{EMod}^{\mathrm{op}} \to \mathbf{Meas}$. For an ω-effect module E we can provide the set of maps $\omega\text{-}\mathbf{EMod}(E, [0,1])$ with a σ-algebra, namely the least one that makes all the evaluation maps $\mathrm{ev}_x \colon \omega\text{-}\mathbf{EMod}(E, [0,1]) \to [0,1]$ measurable, for $x \in E$. This function ev_x is given by $\mathrm{ev}_x(p) = p(x)$. This gives the following situation.

$$\begin{tikzcd} \omega\text{-}\mathbf{EMod}^{\mathrm{op}} \arrow[d, bend left, "\mathrm{Hom}(-,[0,1])"] \arrow[d, phantom, "\dashv"] \\ \mathbf{Meas} \arrow[u, bend left, "\mathrm{Hom}(-,[0,1])"] \arrow[loop below, "\mathcal{G}=\omega\text{-}\mathbf{EMod}(\mathbf{Meas}(-,[0,1]),[0,1])"] \end{tikzcd} \qquad \begin{array}{c} Y \xrightarrow{\omega\text{-}\mathbf{EMod}} \mathbf{Meas}(X, [0,1]) \\ \hline\hline X \xrightarrow[\mathbf{Meas}]{} \omega\text{-}\mathbf{EMod}(Y, [0,1]) \end{array} \qquad \begin{tikzcd} \omega\text{-}\mathbf{EMod}^{\mathrm{op}} \arrow[rr, bend left=15] \arrow[rr, phantom, "\top"] & & \mathcal{EM}(\mathcal{G}) \arrow[ll, bend left=15] \\ & \mathcal{K}\ell(\mathcal{G}) \arrow[ul, "\mathrm{Pred}"] \arrow[ur, "\mathrm{Stat}"'] & \end{tikzcd}$$

We use the symbol $\mathcal{G}$ for the induced monad because of the following result.

▶ **Proposition 6.** *The monad* $\mathcal{G} = \omega\text{-}\mathbf{EMod}\big(\mathbf{Meas}(-,[0,1]),[0,1]\big)$ *on* $\mathbf{Meas}$ *in the above situation is (isomorphic to) the Giry monad [10], given by probability measures:*

$$\mathrm{Giry}(X) \;\overset{def}{=}\; \{\phi\colon \Sigma_X \to [0,1] \mid \phi \textit{ is a probability measure}\} \;=\; \omega\text{-}\mathbf{EA}(\Sigma_X,[0,1]).$$

Proof. The isomorphism involves Lebesgue integration:

$$\begin{tikzcd}
\mathcal{G}(X) = \omega\text{-}\mathbf{EMod}\big(\mathbf{Meas}(X,[0,1]),[0,1]\big) \arrow[r, bend left, "I\mapsto(M\mapsto I(\mathbf{1}_M))"] \arrow[r, phantom, "\cong"] & \omega\text{-}\mathbf{EA}(\Sigma_X,[0,1]) = \mathrm{Giry}(X) \arrow[l, bend left, "\phi\mapsto(p\mapsto\int p\,\mathrm{d}\phi)"]
\end{tikzcd}$$

See [13] or [19] for more details. ◀

The above triangle is further investigated in [13]. It resembles the situation described in [4] for Markov kernels (the ordinary, not the abstract, ones).

Acknowledgements. Several people have contributed to the ideas and examples presented here, including, in alphabetical order: Kenta Cho, Robert Furber, Helle Hansen, Bas and Bram Westerbaan. Thanks to all of them!

References

1 S. Abramsky. Domain theory in logical form. *Ann. Pure & Appl. Logic*, 51(1/2):1–77, 1991.

2 J. Adámek, H. Herrlich, and G.E. Stecker. *Abstract and Concrete Categories. The Joy of Cats.* John Wiley and Sons, New York, 1990. Republished in: *Reprints in Theory and Appl. of Categories* 17, see `http://www.tac.mta.ca/tac/reprints/articles/17/tr17.pdf`.

3 M. Barr and Ch. Wells. *Toposes, Triples and Theories.* Springer, Berlin, 1985. Revised and corrected version available from URL: `http://www.cwru.edu/artsci/math/wells/pub/ttt.html`.

4 P. Chaput, V. Danos, P. Panangaden, and G. Plotkin. Approximating Markov processes by averaging. *Journ. ACM*, 61(1), 2014.

5 R. Cignoli, I D'Ottaviano, and D. Mundici. *Algebraic Foundations of Many-Valued Reasoning*, volume 7 of *Trends in Logic.* Springer, 2000.

6 E. Dijkstra and C. Scholten. *Predicate Calculus and Program Semantics.* Springer, Berlin, 1990.

7 A. Dvurečenskij and S. Pulmannová. *New Trends in Quantum Structures.* Kluwer Acad. Publ., Dordrecht, 2000.

8 R. Furber and B. Jacobs. From Kleisli categories to commutative C^*-algebras: Probabilistic Gelfand duality. In R. Heckel and S. Milius, editors, *Conference on Algebra and Coalgebra in Computer Science (CALCO 2013)*, number 8089 in Lect. Notes Comp. Sci., pages 141–157. Springer, Berlin, 2013.

9 R. Furber and B. Jacobs. From Kleisli categories to commutative C^*-algebras: Probabilistic Gelfand duality, 2014. Extended journal version of [8], see arxiv.org/abs/1303.1115.

10 M. Giry. A categorical approach to probability theory. In B. Banaschewski, editor, *Categorical Aspects of Topology and Analysis*, number 915 in Lect. Notes Math., pages 68–85. Springer, Berlin, 1982.

11 H. H. Hansen, C. Kupke, and R. Leal. Strong completeness for iteration-free coalgebraic dynamic logics. In J. Diaz, I. Lanese, and D. Sangiorgi, editors, *Theoretical Computer Science*, number 8705 in Lect. Notes Comp. Sci., pages 281–295. Springer, Berlin, 2014.

12 T. Heinosaari and M. Ziman. *The Mathematical Language of Quantum Theory. From Uncertainty to Entanglement.* Cambridge Univ. Press, 2012.

13 B. Jacobs. Measurable spaces and their effect logic. In *Logic in Computer Science*. IEEE, Computer Science Press, 2013.

14 B. Jacobs. Dijkstra monads in monadic computation. In M. Bonsangue, editor, *Coalgebraic Methods in Computer Science (CMCS 2014)*, number 8446 in Lect. Notes Comp. Sci., pages 135–150. Springer, Berlin, 2014.

15 B. Jacobs. New directions in categorical logic, for classical, probabilistic and quantum logic. *LMCS, to appear*, 2015. See arxiv.org/abs/1205.3940.

16 B. Jacobs and J. Mandemaker. The expectation monad in quantum foundations. In B. Jacobs, P. Selinger, and B. Spitters, editors, *Quantum Physics and Logic (QPL) 2011*, volume 95 of *Elect. Proc. in Theor. Comp. Sci.*, pages 143–182, 2012.

17 B. Jacobs and J. Mandemaker. Relating operator spaces via adjunctions. In J. Chubb, A. Eskandarian, and V. Harizanov, editors, *Logic and Algebraic Structures in Quantum Computing*, volume 45 of *Lect. Notes in Logic*, pages 123–150. Cambridge Univ. Press, 2015. See arxiv.org/abs/1201.1272.

18 B. Jacobs, J. Mandemaker, and R. Furber. The expectation monad in quantum foundations, 2015.

19 B. Jacobs and A. Westerbaan. An effect-theoretic account of Lebesgue integration, 2015. Math. Found. of Programming Semantics XXXI.

20 B. Jacobs, B. Westerbaan, and A. Westerbaan. States of convex sets. In A. Pitts, editor, *Foundations of Software Science and Computation Structures*, number 9034 in Lect. Notes Comp. Sci., pages 87–101. Springer, Berlin, 2015.

21 P. Johnstone. *Stone Spaces.* Number 3 in Cambridge Studies in Advanced Mathematics. Cambridge Univ. Press, 1982.

22 J. Lambek and P. Scott. *Introduction to higher order Categorical Logic.* Number 7 in Cambridge Studies in Advanced Mathematics. Cambridge Univ. Press, 1986.

23 E. Manes. A triple-theoretic construction of compact algebras. In B. Eckman, editor, *Seminar on Triples and Categorical Homolgy Theory*, number 80 in Lect. Notes Math., pages 91–118. Springer, Berlin, 1969.

24 G. Markowsky. Free completely distributive complete lattices. *Proc. Amer. Math. Soc.*, 74(2):227–228, 1979.

25 M. Mislove. Probabilistic monads, domains and classical information. In E. Kashefi, J. Krivine, and F. van Raamsdonk, editors, *Developments of Computational Methods (DCM 2011)*, volume 88 of *Elect. Proc. in Theor. Comp. Sci.*, pages 87–100, 2012.

26 E. Moggi. Notions of computation and monads. *Inf. & Comp.*, 93(1):55–92, 1991.

27 A. Sokolova. *Coalgebraic Analysis of Probabilistic Systems.* PhD thesis, Techn. Univ. Eindhoven, 2005.

28 P. Taylor. Subspaces in abstract Stone duality. *Theory and Appl. of Categories*, 10(13):301–368, 2002.

29 O. Wyler. Algebraic theories of continuous lattices. In B. Banaschewski and R.-E. Hoffman, editors, *Continuous Lattices*, number 871 in Lect. Notes Math., pages 390–413. Springer, Berlin, 1981.

Towards Concept Analysis in Categories: Limit Inferior as Algebra, Limit Superior as Coalgebra*

Toshiki Kataoka[1,2] **and Dusko Pavlovic**[3]

1 **The University of Tokyo, Tokyo, Japan**
toshikik@is.s.u-tokyo.ac.jp

2 **JSPS Research Fellow**

3 **University of Hawaii at Manoa, Honolulu HI, US**
dusko@hawaii.edu

Abstract

While computer programs and logical theories begin by declaring the concepts of interest, be it as data types or as predicates, network computation does not allow such global declarations, and requires *concept mining* and *concept analysis* to extract shared semantics for different network nodes. Powerful semantic analysis systems have been the drivers of nearly all paradigm shifts on the web. In categorical terms, most of them can be described as bicompletions of enriched matrices, generalizing the Dedekind-MacNeille-style completions from posets to suitably enriched categories. Yet it has been well known for more than 40 years that ordinary categories themselves in general do not permit such completions. Armed with this new semantical view of Dedekind-MacNeille completions, and of matrix bicompletions, we take another look at this ancient mystery. It turns out that simple categorical versions of the *limit superior* and *limit inferior* operations characterize a general notion of Dedekind-MacNeille completion, that seems to be appropriate for ordinary categories, and boils down to the more familiar enriched versions when the limits inferior and superior coincide. This explains away the apparent gap among the completions of ordinary categories, and broadens the path towards categorical concept mining and analysis, opened in previous work.

1998 ACM Subject Classification I.2.6 Learning, H.3.3 Information Search and Retrieval

Keywords and phrases concept analysis, semantic indexing, category, completion, algebra

Digital Object Identifier 10.4230/LIPIcs.CALCO.2015.130

> *It is an open problem whether there exists a sup- and inf-complete category $\mathbb{A}''''$ with a sup- and inf-dense embedding $\mathbb{A} \to \mathbb{A}''''$ in analogy to the Dedekind completion of an ordered set.*
>
> Joachim Lambek [15, Introduction]

> *No Lambek extension of the one-object category $\mathbb{Z}_4$ has finite limits.*
>
> John Isbell [10, Thm. 3.1]

* T. K. is supported by Japan Society for the Promotion of Science. D. P. is partially supported by AFOSR and AFRL, and by UHM.

6th International Conference on Algebra and Coalgebra in Computer Science (CALCO'15).
Editors: Lawrence S. Moss and Pawel Sobocinski; pp. 130–155
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

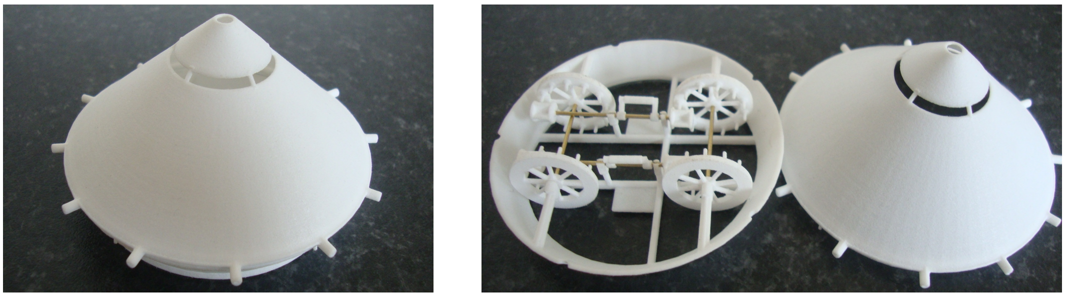


Figure 1 Unidentified object: The external and the internal view.

1 Introduction

1.1 Problem of concept mining and analysis

Suppose you come across upon the object depicted in Fig. 1. The conic top is easily removed to uncover the mechanism on the right. What is this thing?

You would surely approach the problem from both directions at once: on one hand, you would look how the parts fit together and try to discern the *structural components* of the device; on the other hand, you would twiddle with some parts and watch what moves together, trying to figure out the *functional modules.* The parts that move together may not be next to each other, but they probably belong to the same functional module. The parts that are related structurally are more likely to be related functionally. If you manage to discern some distinct components corresponding to distinct functionalities, then each such component-function pair will presumably correspond to a *concept* conceived by the designer of the device. By analyzing the device you will extract the designer's idea.

Similar analyses are formalized under different names in different research communities: some speak of *concept analysis*, some of *knowledge acquisition*, *semantic indexing*, or *data mining* [4, 8, 22]. The application domains and the formalisms vary very widely, from mathematical taxonomy [11], through text analysis [32] and pattern recognition [5], to web search and recommender systems [31]. The importance of formalizing and implementing concept analysis grew rapidly with the advent of the web, as almost anything found on the web requires some sort of concept mining and analysis, not only because there are no global semantical declarations, and the meaning has to be extracted from the network structure [23], but also to establish trust [25]. Diverse toy examples of such concept analysis tasks, motivating the modeling approach extended in this paper, can be found in [24, 25, 27, 28].

The analytic process that a formal concept analyst may initiate upon an encounter with the unidentified object from Fig. 1 is thus not all that different from what a curious child would do: they would both start by recording the observed components on one hand, and the observed functionalities on the other, and they would note which components are related to which functionalities. With the "yes-no" relations, the formal version of this process leads to the simple and influential method that goes under the name *Formal Concept Analysis (FCA)* [7, 6]. If the relations between the components and the functionalities are quantified by real numbers and stored in *pattern matrices*, then the analysis usually proceeds by the methods of statistics and linear algebra, and goes under the name *Principal Component Analysis (PCA)* [13], or *Latent Semantic Analysis (LSA)* [17], etc. It performs the singular value decomposition of the pattern matrix, and thus mines the concepts as the eigenspaces of the induced linear operators.

Interestingly, if you wanted to record that the unidentified device has 4 identical wheels, and that each wheel has 12 identical cogs, and that two of the wheels are related to two different functionalities, driving and steering, you would be led beyond the familiar concept mining approaches. While the experts in these approaches would surely figure out multiple tricks to record what is needed (e.g. by using multi-level pattern matrices), the straightforward approach leads beyond the FCA matrices of 0s and 1s, and beyond the LSA matrices of real numbers, to matrices of sets between components on one hand, matrices of sets between functionalities on the other hand, and matrices of sets between components and functionalities in-between. You would construct a category of components, a category of functionalities, and a profunctor/distributor between them. If the cog is recognized as a part, then a coproduct of 12 cogs would be embedded in each wheel. If the cogs are attached with rivets, then their morphisms may not be monic, since the distinctions of some of their parts may be obliterated through deformations. So why have such categorical models not been used in concept analysis?

Many of the concept mining approaches derived from LSA are instances of *spectral decomposition* [1]. Formalized in terms of *enriched category theory* [14], the problem of concept mining turns out to be an instance of a general spectral decomposition problem [25, 27], which can also be viewed as a problem of *minimal bicompletion* of a suitably enriched matrix [28]. Even the standard linear algebra of LSA seems to be an instance of such bicompletion, over a suitable category[1] of real numbers. The problem of minimal bicompletions of enriched categories, which subsume the Dedekind-MacNeille completions of posets, is the special case, arising when a category itself is viewed as a matrix. Instantiated to categories enriched over sets, also known as *"ordinary"* categories, this turned out to be a strange problem, as suggested by the quotations at the very beginning of the paper. Maybe this is the reason for the notable absence of ordinary categories in the extensive concept mining toolkits? We sketch the problem of bicompletions of ordinary categories in the next section.

1.2 Problem of minimal bicompletions of matrices and categories

Throughout the paper, we assume familiarity with the basic concepts of category theory, e.g. at the level of [20]. To understand the general approach to concept mining through minimal bicompletions, explained in this section, the reader may need some ideas about enriched categories as well, e.g. as presented in [14]. Beyond this section, the rest of the paper will be about ordinary categories.

Suppose that we have thus proceeded as in the preceding section, and built a category of components $\mathbb{A}$ and a category of functionalities $\mathbb{B}$. If we have recorded just the inclusion relations, then each of these categories is a poset, i.e. enriched over the ordered monoid $(\{0,1\}, \wedge, 1)$. If we have recorded the distances among the components on one hand, and among the functionalities on the other, then our categories are metric spaces [18], viewed as categories enriched over the monoidal poset $([0,\infty], +, 0)$. If we capture the components and the functionalities as ordinary categories, then $\mathbb{A}$ and $\mathbb{B}$ are enriched over the monoidal category $(\mathsf{Set}, \times, 1)$.

[1] not poset!

$$\begin{array}{c}
\Phi\colon \mathbb{A}^o\times\mathbb{B}\to\mathcal{V} \\
\hline
\Phi^{\#}\colon \mathbb{A}\to\left(\mathcal{V}^{\mathbb{B}}\right)^o \qquad\qquad \Phi_{\#}\colon \mathbb{B}\to\mathcal{V}^{\mathbb{A}^o} \\
\hline
\Phi^{*}\colon \mathcal{V}^{\mathbb{A}^o}\to\left(\mathcal{V}^{\mathbb{B}}\right)^o \qquad\qquad \Phi_{*}\colon \left(\mathcal{V}^{\mathbb{B}}\right)^o\to\mathcal{V}^{\mathbb{A}^o} \\
\hline
\overleftarrow{\Phi}=\Phi_*\Phi^*\colon \mathcal{V}^{\mathbb{A}^o}\to\mathcal{V}^{\mathbb{A}^o} \qquad\qquad \overrightarrow{\Phi}=\Phi^*\Phi_*\colon \left(\mathcal{V}^{\mathbb{B}}\right)^o\to\left(\mathcal{V}^{\mathbb{B}}\right)^o
\end{array}$$

Figure 2 Deriving the two extensions and the two kernels of a matrix Φ.

$$\begin{tikzcd}
\mathbb{A} \arrow[r, tail, "\nabla"] \arrow[d, "\Phi"', "\circ" marking] \arrow[dr, dashed, "\Phi^{\#}"' near start] \arrow[rr, bend left=60, "\nabla_\Phi"] & \Downarrow\mathbb{A} \arrow[loop above, "\overleftarrow{\Phi}"] \arrow[r, bend left=10, "\overline{\Phi}"] \arrow[d, bend left=15, "\Phi_*"] \arrow[d, phantom, "\dashv"] \arrow[r, phantom, "\perp"] & \left(\Downarrow\mathbb{A}\right)^{\overleftarrow{\Phi}} \arrow[l, bend left=10, "U"] \arrow[d, "\overleftrightarrow{\Phi}", "\circ" marking] \\
\mathbb{B} \arrow[r, tail, "\Delta"'] \arrow[ur, dashed, "\Phi_{\#}" near start] \arrow[rr, bend right=60, "\Delta_\Phi"'] & \Uparrow\mathbb{B} \arrow[loop below, "\overrightarrow{\Phi}"] \arrow[u, bend left=15, "\Phi^*"] \arrow[r, bend right=10, "\underline{\Phi}"'] \arrow[r, phantom, "\perp"] & \left(\Uparrow\mathbb{B}\right)^{\overrightarrow{\Phi}} \arrow[l, bend right=10, "V"']
\end{tikzcd}$$

Figure 3 Minimal bicompletion of a matrix Φ.

1.2.1 The setting of minimal bicompletion

The relationships between the components and the functionalities will be expressed as a $\mathcal{V}$-enriched functor $\Phi\colon \mathbb{A}^o\times\mathbb{B}\to\mathcal{V}$, where $\mathcal{V}$ is the enriching category, such as $\{0,1\}, [0,\infty]$ or Set above. We call such $\mathcal{V}$-enriched functor a *matrix*. In particular, given a $\mathcal{V}$-matrix $\Phi\colon \mathbb{A}^o\times\mathbb{B}\to\mathcal{V}$ we derive its extensions as in Fig. 2.

The functors $\Phi^{\#}$ and $\Phi_{\#}$ are the transpositions of Φ. The presheaves in the form $\Phi_{\#}b$ and the postsheaves in the form $\Phi^{\#}a$ are called Φ-*representable*. The functors Φ^* and Φ_* are the Kan extensions [14, Ch. 4] of $\Phi^{\#}$ and $\Phi_{\#}$. Since they form an adjunction, their composite $\overleftarrow{\Phi}$ is a monad and $\overrightarrow{\Phi}$ is a comonad.

When the enrichment is clear from the context, it is convenient to abbreviate the matrix $\mathbb{A}^o\times\mathbb{B}\to\mathcal{V}$ to $\mathbb{A}\looparrowright\mathbb{B}$ and the completions $\mathcal{V}^{\mathbb{A}^o}$ and $\left(\mathcal{V}^{\mathbb{B}}\right)^o$ to $\Downarrow\mathbb{A}$ and $\Uparrow\mathbb{B}$ respectively, so that the derivations in Fig. 2 give the diagram in Fig. 3 where ∇ and Δ are the Yoneda embeddings [14, Sec. 2.4]. The monad $\overleftarrow{\Phi}=\Phi_*\Phi^*$, induced by the Kan extensions $\Phi^*\dashv\Phi_*\colon\Uparrow\mathbb{B}\to\Downarrow\mathbb{A}$, induces the category of (Eilenberg-Moore) algebras $(\Downarrow\mathbb{A})^{\overleftarrow{\Phi}}$, whereas the comonad $\overrightarrow{\Phi}=\Phi^*\Phi_*$ induces $(\Uparrow\mathbb{B})^{\overrightarrow{\Phi}}$. The functor $\nabla_\Phi=\overline{\Phi}\circ\nabla$ maps $\mathbb{A}$ to free $\overleftarrow{\Phi}$-algebras generated by the representable presheaves, whereas the functor $\Delta_\Phi=\underline{\Phi}\circ\Delta$ maps $\mathbb{B}$ to cofree $\overrightarrow{\Phi}$-coalgebras cogenerated by the representable postsheaves.

1.2.2 Familiar cases

When $\mathcal{V}=\{0,1\}$, the $\mathcal{V}$-enriched categories $\mathbb{A}$ and $\mathbb{B}$ are posets. Then $\Downarrow\mathbb{A}$ consists of antitone maps $\overleftarrow{L}:\mathbb{A}^o\to\{0,1\}$, or equivalently of the lower-closed sets in $\mathbb{A}$, whereas $\Uparrow\mathbb{B}$ consists of the monotone maps $\overrightarrow{U} : \mathbb{B} \to \{0,1\}$, or equivalently of the upper-closed sets in $\mathbb{B}$. The Yoneda embedding $\nabla \colon \mathbb{A} \to \Downarrow\mathbb{A}$ is then the supremum (or join) completion, and the $\Delta \colon \mathbb{B} \to \Uparrow\mathbb{B}$ is the infimum (or meet) completion. A matrix $\Phi \colon \mathbb{A}^o \times \mathbb{B} \to \{0,1\}$ corresponds to a subset of the product poset which is lower closed in $\mathbb{A}$ and upper closed in $\mathbb{B}$. Its extensions are then

$$\Phi^* \overleftarrow{L} \quad = \quad \left\{ u \in \mathbb{B} \,\middle|\, \forall x.\, \overleftarrow{L}(x) \Rightarrow \Phi(x,u) \right\} \tag{1}$$

$$\Phi_* \overrightarrow{U} \quad = \quad \left\{ \ell \in \mathbb{A} \,\middle|\, \forall y.\, \overrightarrow{U}(y) \Rightarrow \Phi(\ell,y) \right\} \tag{2}$$

Intuitively, $\Phi^* \overleftarrow{L}$ can be construed as the set of upper bounds in $\mathbb{B}$ of the Φ-image of the lower set $\overleftarrow{L}$, whereas $\Phi_* \overrightarrow{U}$ can be construed as the set of Φ-lower bounds of the upper set $\overrightarrow{U}$. The operator $\overleftarrow{\Phi} = \Phi_* \Phi^*$ thus maps each lower set $\overleftarrow{L}$ to the set of the Φ-lower bounds of the set of its Φ-upper bounds; whereas the operator $\overrightarrow{\Phi} = \Phi^* \Phi_*$ maps each upper set $\overrightarrow{U}$ to the set of the Φ-upper bounds of its Φ-lower bounds. Both operators are thus *closure operators.* Their lattices of closed sets $(\Downarrow\mathbb{A})^{\overleftarrow{\Phi}}$ and $(\Uparrow\mathbb{B})^{\overrightarrow{\Phi}}$ turn out to be isomorphic, and form the *nucleus* of Φ [27]. A $\overleftarrow{\Phi}$-closed set in $\mathbb{A}$ and the corresponding $\overrightarrow{\Phi}$-closed set in $\mathbb{B}$, of course, completely determine each other, but the most informative presentation carries both, as Dedekind-style cuts. When $\mathbb{A} = \mathbb{B}$ and $\Phi \subseteq \mathbb{A}^o \times \mathbb{A}$ is the partial ordering

$$\Phi(x,y) \quad \iff \quad x \leq y \tag{3}$$

then the nucleus is just the Dedekind-MacNeille completion $\Updownarrow\mathbb{A}$ of the poset $\mathbb{A}$ [21]. This is the *minimal* bicompletion, in the sense that the embedding $\mathbb{A} \to \Updownarrow\mathbb{A}$ preserves any suprema and infima that $\mathbb{A}$ may already have, and only adds those that do not yet exist [21, 2, 12, III.3.11]. The consequence of this minimality is that every element of the completion $\Updownarrow\mathbb{A}$ is *both* a supremum and an infimum of the elements of $\mathbb{A}$. The nucleus of a $\{0,1\}$-matrix is a minimal bicompletion in a similar sense, as are the nuclei of $[0,1]$-matrices, and of $[0,\infty]$-matrices[2]: the nuclei give the *semantic bicompletions* of matrices, uncovering their concepts [27, 28].

1.2.3 The trouble with ordinary categories

Our main concern in the present paper are the minimal bicompletions of matrices and categories enriched over $(\mathsf{Set}, \times, 1)$. Categories enriched in Set are usually called *ordinary* categories. Set-matrices are variably called *profunctors* or *distributors.* We increase the wealth of terminology by calling them *matrices.* The functors $\overleftarrow{\alpha} \in \Downarrow\mathbb{A} = \mathsf{Set}^{\mathbb{A}^o}$ are called *presheaves.* The functors $\overrightarrow{\beta} \in \Uparrow\mathbb{B} = \left(\mathsf{Set}^{\mathbb{B}}\right)^o$ are usually called *covariant functors to* Set, but we call them *postsheaves.* We use without further explanation the well known fact [9, 19] that presheaves are equivalent to discrete fibrations, and that postsheaves are equivalent with discrete opfibrations.

We also call the categorical limits the *infima*, and the categorical colimits the *suprema*, following Lambek's 1966 Lectures on Completions of Categories [15], quoted at the beginning

[2] Since the monoidal posets $([0,\infty], +, 0)$ and $([0,1], \times, 1)$ are isomorphic as monoidal categories, all statements about categories enriched over them transfer trivially. However, isomorphisms are not always trivial phenomena. E.g., the Laplace transform is an isomorphism, which maps differential operations into algebraic operations, and thus allows solving differential equations as algebraic equations, and mapping back the solutions [30]. In a similar way, it often happens that a distance space presentation of a data pattern, enriched over $([0,\infty], +, 0)$, displays some geometric content, whereas an isomorphic proximity lattice presentation of the same data pattern, enriched over $([0,1], \times, 1)$, displays some generalized order structure, not apparent in the first interpretation.

of this paper. The Yoneda embeddings $\nabla\colon \mathbb{A} \to {\Downarrow}\mathbb{A}$ and $\Delta\colon \mathbb{B} \to {\Uparrow}\mathbb{B}$ are then again, respectively, the supremum and the infimum completion, this time of the categories $\mathbb{A}$ and $\mathbb{B}$. The transposes $\Phi^{\#}$ and $\Phi_{\#}$ now extend to the adjunction $\Phi^* \dashv \Phi_*\colon {\Uparrow}\mathbb{B} \to {\Downarrow}\mathbb{A}$, which are defined similarly to (1–2). More precisely, the mappings between the $\mathbb{A}$-presheaves and $\mathbb{B}$-postsheaves

$$\frac{\overleftarrow{\alpha} : \mathbb{A}^o \to \mathsf{Set}}{\Phi^*\overleftarrow{\alpha} : \mathbb{B} \to \mathsf{Set}} \qquad\qquad \frac{\overrightarrow{\beta} : \mathbb{B} \to \mathsf{Set}}{\Phi_*\overrightarrow{\beta} : \mathbb{A}^o \to \mathsf{Set}}$$

are defined as follows

$$\Phi^*\overleftarrow{\alpha}(u) \;=\; \underset{x\in\mathbb{A}}{\underleftarrow{\lim}}\Big(\overleftarrow{\alpha}(x) \Rightarrow \Phi(x,u)\Big) \;=\; {\Downarrow}\mathbb{A}\Big(\overleftarrow{\alpha}, \Phi_{\#}u\Big) \tag{4}$$

$$\Phi_*\overrightarrow{\beta}(\ell) \;=\; \underset{y\in\mathbb{B}}{\underleftarrow{\lim}}\Big(\overrightarrow{\beta}(y) \Rightarrow \Phi(\ell,y)\Big) \;=\; {\Uparrow}\mathbb{B}\Big(\Phi^{\#}\ell, \overrightarrow{\beta}\Big) \tag{5}$$

Here we write $X \Rightarrow Y$ for the set exponents Y^X not only because the multiple exponents tend to "fly away" in the latter notation, but also to emphasize the parallel with (1–2). When $\mathbb{A} = \mathbb{B}$ is the same category, and $\Phi = H\colon \mathbb{A}^o \times \mathbb{A} \to \mathsf{Set}$ is the hom-set matrix, then $H^*\overleftarrow{\alpha}(u)$ is the set of (right) cones from the presheaf $\overleftarrow{\alpha}$, viewed as a diagram, to the object u as the tip of the cone. Dually, $H_*\overrightarrow{\beta}(\ell)$ is the set of (left) cones from the tip ℓ to the diagram $\overrightarrow{\beta}$. For a general matrix $\Phi\colon \mathbb{A} \looparrowright \mathbb{B}$, thinking of the elements of each set $\Phi(a,b)$ as "arrows" from $a \in \mathbb{A}$ to $b \in \mathbb{B}$ also allows thinking of $\overrightarrow{\varrho} \in \Phi^*\overleftarrow{\alpha}(u)$ as a (right) "cone" from the diagram $\overleftarrow{\alpha}$ in $\mathbb{A}$ to the tip $u \in \mathbb{B}$, and of $\overleftarrow{\lambda} \in \Phi_*\overleftarrow{\beta}(\ell)$ as a (left) "cone" from the tip $\ell \in \mathbb{A}$ to a diagram $\overleftarrow{\beta}$ in $\mathbb{B}$. The presheaves and postsheaves of (4) and (5) thus generalize the lower and the upper sets of (1) and (2).

At the very beginning of his lectures, Lambek raised the question of the Dedekind-MacNeille completion of a category, and left it open. He did not raise the general question of semantic completions of matrices (profunctors, or distributors) only because the semantical impact was not clear at the time; but the general situation from Fig. 3 was well known. Lambek's open question of the Dedekind-MacNeille completion of a category was closed by Isbell a couple of years later, who showed in [10, Sec. 3] that already the group $\mathbb{Z}_4$, viewed as a category with a single object, cannot have a completion generated both by the suprema and by the infima.

However, taking a broader semantical view, and seeking semantic completions of matrices, shows that the story does not really end with Isbell's counterexample. A semantic completion of a matrix, relating, say, the parts and the moves observed within a device like the one on Fig. 1, should uncover the concepts underlying the design of the device. These concepts are expressed through the structural component of the device, and through its functional units. When the matrix is enriched over a monoidal poset, then there is a one-to-one correspondence between the structural components and the functional modules, and they form the nucleus of the matrix [27, 28]. In reality, though, a single structural component may play a role in several functional modules, and vice versa. While the posetal enrichment cannot capture this, the enrichment in sets, or in a proper category of real numbers, can record how many copies of a given a part are used for a certain function. Modeled in this way, the spaces of structural components and of functional modules will not be isomorphic. The concepts will not be uncovered as a single category of component-function pairs, like in the posetal case, but as a nontrivial matrix relating some component-concepts approximated by their functionalities with some function-concepts approximated by the components that perform them.

Contributions

To spell this out, we consider the following technical questions:

(a) What kind of completions of a given matrix $\Phi\colon \mathbb{A} \looparrowright \mathbb{B}$ are provided by the categories $(\Downarrow\mathbb{A})^{\overleftarrow{\Phi}}$ and $(\Uparrow\mathbb{B})^{\overrightarrow{\Phi}}$? (The idea is that the former captures the component-concepts, the latter the function-concepts.)

(b) What kind of matrix $\overleftrightarrow{\Phi} : (\Downarrow\mathbb{A})^{\overleftarrow{\Phi}} \looparrowright (\Uparrow\mathbb{B})^{\overrightarrow{\Phi}}$ is the minimal bicompletion of $\Phi\colon \mathbb{A} \looparrowright \mathbb{B}$? (Capturing the relations between the component-concepts and the function-concepts.)

Our approach to these questions is based on a new family of limits and colimits, introduced in the next section. It seems intuitive and appropriate to call them *limit inferior*, and *limit superior*. For consistency, we also revert, albeit just for the duration of this paper[3], from *limits* and *colimits* to *infima* and *suprema*, following Lambek [15]. The reader is reminded that in posets

- the limit inferior is the *supremum* of the lower bounds of a set, whereas
- the limit superior is the *infimum* of the upper bounds.

Mutatis mutandis, the categorical concepts will behave similarly.

Overview of the paper

In Sec. 2, we propose the answers to the above question. Sec. 2.1 spells out the preliminaries. Sec. 2.2 defines categorical limits inferior and superior and characterizes their completions. Sec. 2.3 proposes an answer to question (a) above. Sec. 2.4 proposes an answer to question (b) above. In Sec. 3 we study some simple examples, illustrating and validating the introduced concepts. Sec. 3.1 describes a monadicity workflow useful for analyzing the examples. Sec. 3.2 characterizes completions of constant matrices. Sections 3.3 and 3.4 characterize completions of the matrices representing groups or posets, respectively. Sec. 3.5 characterizes completions of a vector in the (cyclic) group $\mathbb{Z}_p$ of prime order p. Sec. 4 closes the paper, to some extent.

Due to the space constraints of this conference paper and the scope of the presented material, all proofs and many lemmas had to be moved into the appendices. Full details will require a significantly longer paper.

2 Categorical limit inferior and limit superior

2.1 Preliminaries

Although suprema and infima are very basic concepts, familiar to most readers, and easily found in [20, Sec. III.3–4], we spell them out here not only to introduce the notation and practice using the words *infimum* and *supremum* instead of limit and colimit, but also to align these familiar definitions with the variations needed to define the *limit superior* and the *limit inferior*.

Let $\mathbb{C}$ and $\mathbb{J}$ be categories and $\mathbb{C}^{\mathbb{J}}$ the category of functors between them, with natural transformations as morphisms. Let $\square\colon \mathbb{C} \to \mathbb{C}^{\mathbb{J}}$ be the functor taking each object of x of $\mathbb{C}$ to the constant functor $\square x\colon \mathbb{J} \to \mathbb{C}$, which maps all objects of $\mathbb{J}$ to $x \in \mathbb{C}$ and all morphisms of $\mathbb{J}$ to id_x.

[3] We hope that our terminological contributions, advancing from "profunctors" and "distributors" to "matrices" and from "covariant functors to Set" to "postsheaves", as well as retreating from "limits" to "infima" and from "colimits" to "suprema", will not end up being the central features of the paper.

The suprema and the infima in $\mathbb{C}$ can be defined as, respectively, the left and the right adjoint of the constant functor, i.e.

$$\varinjlim \dashv \Box \dashv \varprojlim \quad : \quad \mathbb{C}^{\mathbb{J}} \to \mathbb{C}$$

These adjunctions can be viewed as the natural bijections

$$\mathbb{C}^{\mathbb{J}}(F, \Box x) \quad \cong \quad \mathbb{C}\big(\varinjlim F, x\big) \tag{6}$$

$$\mathbb{C}^{\mathbb{J}}(\Box x, F) \quad \cong \quad \mathbb{C}\big(x, \varprojlim F\big) \tag{7}$$

It is well known that the Yoneda embeddings realize the $\varinjlim$ and $\varprojlim$-completions [20, Sec. X.6]:

- $\nabla\colon \mathbb{C} \to \Downarrow\mathbb{C}$ is the $\varinjlim$-completion of $\mathbb{C}$, whereas
- $\Delta\colon \mathbb{C} \to \Uparrow\mathbb{C}$ is the $\varprojlim$-completion of $\mathbb{C}$

where

- $\Downarrow\mathbb{C}$ denotes the category $\mathsf{Set}^{\mathbb{C}^o}$ of $\mathbb{C}$-*presheaves*, or equivalently[4] the category of discrete fibrations over $\mathbb{C}$,
- $\Uparrow\mathbb{C}$ denotes the category $\left(\mathsf{Set}^{\mathbb{C}}\right)^o$ of $\mathbb{C}$-*postsheaves*, or equivalently the opposite category of discrete opfibrations over $\mathbb{C}$.

For completeness, we note the following well known and routinely checkable fact.

▶ **Lemma 2.1.** *Given a functor $F\colon \mathbb{J} \to \mathbb{C}$, consider the presheaf and the postsheaf*

$$\left(\overleftarrow{F} : \mathbb{C}/\!\!/F \to \mathbb{C}\right) \in \Downarrow\mathbb{C} \qquad\qquad \left(\overrightarrow{F} : F/\!\!/\mathbb{C} \to \mathbb{C}\right) \in \Uparrow\mathbb{C} \tag{8}$$

where $\mathbb{C}/\!\!/F$ is the category of connected components[5] of the comma category $\mathbb{C}/F$ from $\mathrm{Id}_{\mathbb{C}}$ *to F, whereas $F/\!\!/\mathbb{C}$ is the category of connected components of the comma category $F/\mathbb{C}$ the other way around [20, Sections II.6 and IX.3]. Then*

$$\varinjlim F \;=\; \varinjlim \overleftarrow{F} \qquad\qquad \varprojlim F \;=\; \varprojlim \overrightarrow{F} \tag{9}$$

Notations have been introduced in Sec. 1.2, especially in Figures 2 and 3. The next section studies the special case $\Phi = H\colon \mathbb{C} \looparrowright \mathbb{C}$ of the matrix of hom-sets of a category.

2.2 Limit inferior and limit superior over a category

▶ **Definition 2.2.** For arbitrary categories $\mathbb{C}$ and $\mathbb{J}$ we define

- the category of *left saturated diagrams* $\mathbb{C}^{\mathbb{J}}_{\Downarrow}$ to consist of
  - objects $|\mathbb{C}^{\mathbb{J}}_{\Downarrow}| = |\mathbb{C}^{\mathbb{J}}|$
  - morphisms $\mathbb{C}^{\mathbb{J}}_{\Downarrow}(F, G) = \Downarrow\mathbb{C}\left(H_*\overrightarrow{F}, H_*\overrightarrow{G}\right)$
- the category of *right saturated diagrams* $\mathbb{C}^{\mathbb{J}}_{\Uparrow}$ to consist of
  - objects $|\mathbb{C}^{\mathbb{J}}_{\Uparrow}| = |\mathbb{C}^{\mathbb{J}}|$
  - morphisms $\mathbb{C}^{\mathbb{J}}_{\Uparrow}(F, G) = \Uparrow\mathbb{C}\left(H^*\overleftarrow{F}, H^*\overleftarrow{G}\right)$

[4] The equivalence between the “indexed” and “fibered” versions of sheaves lies at the heart of Grothendieck’s descent theory [9, VI], but also generalizes to substantially different purposes [29, 26].

[5] To be precise, each fiber category $(\mathbb{C}/\!\!/F)_x$ at $x \in \mathbb{C}$ is defined to be the category of connected components of the fiber category $(\mathbb{C}/F)_x = \mathbb{C}/Fx$.

▶ **Definition 2.3.** In a category $\mathbb{C}$ we define

- the *limit inferior* operation $\overrightarrow{\lim}$ over left diagrams from $\mathbb{J}$ by the adjunction

  $$\overrightarrow{\lim} \dashv \square \quad : \quad \mathbb{C} \to \mathbb{C}^{\mathbb{J}}_{\Downarrow}$$

  which can be viewed as the natural bijection

  $$\mathbb{C}^{\mathbb{J}}_{\Downarrow}(F, \square x) \quad \cong \quad \mathbb{C}\Big(\overrightarrow{\lim} F, x\Big)$$

- the *limit superior* operation $\overleftarrow{\lim}$ over right diagrams from $\mathbb{J}$ by the adjunction

  $$\square \dashv \overleftarrow{\lim} \quad : \quad \mathbb{C}^{\mathbb{J}}_{\Uparrow} \to \mathbb{C}$$

  which can be viewed as the natural bijection

  $$\mathbb{C}^{\mathbb{J}}_{\Uparrow}(\square x, F) \quad \cong \quad \mathbb{C}\Big(x, \overleftarrow{\lim} F\Big)$$

▶ Remark. Note that the operations $\overrightarrow{\lim}$ and $\overleftarrow{\lim}$ are defined over *arbitrary* diagrams. Indeed, the objects of the categories of *saturated* diagrams are arbitrary diagrams; the saturation is imposed on them in the definitions of the morphisms in these categories.

The operations $\varinjlim$ and $\varprojlim$ are also defined over arbitrary diagrams, but differently: the supremum of a diagram is equal to the supremum of the induced presheaf; and the infimum of a diagram is equal to the infimum of the induced postsheaf, as stated in Lemma 2.1. This is analogous to lattices, where a supremum of a set is equal to the supremum of its lower closure, whereas the infimum of a set is the infimum of the upper closure. However, the limit inferior of a diagram is the supremum of the presheaf induced by the postsheaf induced by the diagram; and the limit superior is the infimum of the postsheaf induced by the presheaf induced by the diagram. In a partially ordered set, the limit inferior of a set is the join of the lower bounds of all of its upper bounds; whereas the limit superior of a set is the meet of the upper bounds of all of its lower bounds.

▶ **Lemma 2.4.** *Every representable presheaf ∇x is a free algebra in $\Downarrow\mathbb{C}^{\overleftarrow{H}}$, with $\nabla x \overset{\eta}{\cong} \overleftarrow{H}\nabla x$. Every representable postsheaf Δx is a cofree coalgebra in $\Uparrow\mathbb{C}^{\overrightarrow{H}}$, with $\overrightarrow{H}\Delta x \overset{\varepsilon}{\cong} \Delta x$.*

▶ **Proposition 2.5.** *Every $\overleftarrow{H}$-algebra is a limit inferior in $\Downarrow\mathbb{C}^{\overleftarrow{H}}$ of representable presheaves, viewed as $\overleftarrow{H}$-algebras. Every $\overrightarrow{H}$-coalgebra is a limit superior in $\Uparrow\mathbb{C}^{\overrightarrow{H}}$ of representable postsheaves, viewed as $\overrightarrow{H}$-coalgebras.*

▶ **Corollary 2.6.** *$\Downarrow\mathbb{C}^{\overleftarrow{H}}$ is $\overrightarrow{\lim}$-complete. $\Uparrow\mathbb{C}^{\overrightarrow{H}}$ is $\overleftarrow{\lim}$-complete.*

▶ **Theorem 2.7.** *The extended Yoneda embeddings realize the limit inferior and limit superior completions:*

- $\nabla_H : \mathbb{C} \xrightarrow{\nabla} \Downarrow\mathbb{C} \xrightarrow{\overline{H}} \Downarrow\mathbb{C}^{\overleftarrow{H}}$ *is the $\overrightarrow{\lim}$-completion of $\mathbb{C}$, whereas*
- $\Delta_H : \mathbb{C} \xrightarrow{\Delta} \Uparrow\mathbb{C} \xrightarrow{\underline{H}} \Uparrow\mathbb{C}^{\overrightarrow{H}}$ *is the $\overleftarrow{\lim}$-completion of $\mathbb{C}$.*

2.3 Limit inferior and limit superior over a matrix

Given a category $\mathbb{C}$, Lem. 2.1 implies that the suprema and the infima, defined by (6) and (7) respectively, can be viewed as the left and the right adjoint of the corresponding Yoneda embeddings:

$$\begin{tikzcd} \mathbb{C} \arrow[r, bend right, "\nabla"'] \arrow[r, phantom, "\perp"] & \Downarrow\mathbb{C} \arrow[l, bend right, "\varinjlim"'] \end{tikzcd} \qquad\qquad \begin{tikzcd} \mathbb{C} \arrow[r, bend right, "\Delta"'] \arrow[r, phantom, "\top"] & \Uparrow\mathbb{C} \arrow[l, bend right, "\varprojlim"'] \end{tikzcd}$$

Given a matrix $\Phi\colon \mathbb{A}^o \times \mathbb{B} \to \mathsf{Set}$, the suprema and the infima *weighted* by its transposes $\Phi^\#\colon \mathbb{A} \to {\Uparrow}\mathbb{B}$ and $\Phi_\#\colon \mathbb{B} \to {\Downarrow}\mathbb{A}$ can similarly be viewed as adjoints:

$$\begin{tikzcd} \mathbb{B} \arrow[r, bend right=15, "\Phi_\#"'] \arrow[r, phantom, "\perp"] & {\Downarrow}\mathbb{A} \arrow[l, bend right=15, "\varinjlim_\Phi"'] \end{tikzcd} \qquad\qquad \begin{tikzcd} \mathbb{A} \arrow[r, bend right=15, "\Phi^\#"'] \arrow[r, phantom, "\top"] & {\Uparrow}\mathbb{B} \arrow[l, bend right=15, "\varprojlim_\Phi"'] \end{tikzcd}$$

It is, of course, well known and easy to see that the weighted limits can in ordinary categories be reduced to the ordinary limits. The situation is slightly more subtle with the weighted inferior and superior limits. To align the two situations, note that the adjunctions

$$\mathbb{B}\left(\varinjlim_\Phi \overleftarrow{\alpha}, b\right) \;\cong\; {\Downarrow}\mathbb{A}\left(\overleftarrow{\alpha}, \Phi_\# b\right) \qquad\qquad \mathbb{A}\left(a, \varprojlim_\Phi \overrightarrow{\beta}\right) \;\cong\; {\Uparrow}\mathbb{B}\left(\Phi^\# a, \overrightarrow{\beta}\right)$$

will now become

$$\mathbb{A}\left(\overrightarrow{\lim}_\Phi \overrightarrow{\beta}, a\right) \;\cong\; ({\Downarrow}\mathbb{A})^{\overleftarrow{\Phi}}\left(\Phi_* \overrightarrow{\beta}, \nabla_\Phi a\right) \qquad \mathbb{B}\left(b, \overleftarrow{\lim}_\Phi \overleftarrow{\alpha}\right) \;\cong\; ({\Uparrow}\mathbb{B})^{\overrightarrow{\Phi}}\left(\Delta_\Phi b, \Phi^* \overleftarrow{\alpha}\right)$$

▶ **Definition 2.8.** Given a matrix $\Phi\colon \mathbb{A}^o \times \mathbb{B} \to \mathsf{Set}$, with the induced extensions as in Fig. 3, we define the operations Φ-*limit inferior* $\overrightarrow{\lim}_\Phi$ and $\overleftarrow{\lim}_\Phi$ by the following adjunctions

$$\begin{tikzcd} \mathbb{A} \arrow[r, bend right=15, "\nabla_\Phi"'] \arrow[r, phantom, "\perp"] & ({\Downarrow}\mathbb{A})^{\overleftarrow{\Phi}} \arrow[l, bend right=15, "\overrightarrow{\lim}_\Phi"'] \end{tikzcd} \qquad\qquad \begin{tikzcd} \mathbb{B} \arrow[r, bend right=15, "\Delta_\Phi"'] \arrow[r, phantom, "\top"] & ({\Uparrow}\mathbb{B})^{\overrightarrow{\Phi}} \arrow[l, bend right=15, "\overleftarrow{\lim}_\Phi"'] \end{tikzcd}$$

where ∇_Φ and Δ_Φ are as defined in Fig. 3.

2.3.1 Two pairs of "Yoneda embeddings"

In this section we spell out the basic properties of the two kinds of "Yoneda embeddings" induced by a matrix $\Phi\colon \mathbb{A} \looparrowright \mathbb{B}$:

- $\overleftarrow{\Phi}$-algebra representables and $\overrightarrow{\Phi}$-coalgebra representables

  $$\nabla_\Phi\colon \mathbb{A} \to ({\Downarrow}\mathbb{A})^{\overleftarrow{\Phi}} \qquad\qquad \Delta_\Phi\colon \mathbb{B} \to ({\Uparrow}\mathbb{B})^{\overrightarrow{\Phi}}$$

- Φ-representable presheaves and postsheaves

  $$\Phi^\#\colon \mathbb{A} \to ({\Uparrow}\mathbb{B})^{\overrightarrow{\Phi}} \qquad\qquad \Phi_\#\colon \mathbb{B} \to ({\Downarrow}\mathbb{A})^{\overleftarrow{\Phi}}$$

The underlying functors are as in Fig. 3. The structures are as follows.

▶ **Lemma 2.9.** *Every presheaf $\overleftarrow{\alpha} \in {\Downarrow}\mathbb{A}$ induces the $\overrightarrow{\Phi}$-coalgebra $\Phi^*\overleftarrow{\alpha} \xrightarrow{\Phi^*\eta} \Phi^*\Phi_*\Phi^*\overleftarrow{\alpha}$. Every Φ-representable postsheaf $\Phi^\# a$ is thus canonically a $\overrightarrow{\Phi}$-coalgebra, since $\Phi^\# a = \Phi^*\nabla a$.*

Any postsheaf $\overrightarrow{\beta} \in {\Uparrow}\mathbb{B}$ induces the $\overleftarrow{\Phi}$-algebra $\Phi_*\overrightarrow{\beta} \xleftarrow{\Phi_*\varepsilon} \Phi_*\Phi^*\Phi_*\overrightarrow{\beta}$. Every Φ-representable presheaf $\Phi_\# b$ is thus canonically a $\overleftarrow{\Phi}$-algebra, since $\Phi_\# b = \Phi_*\Delta b$.

▶ **Lemma 2.10** (Matrix Yoneda Lemma). *For every $a \in \mathbb{A}$ and every $\overrightarrow{\beta} \in {\Uparrow}\mathbb{B}$ there is a natural bijection*

$$({\Downarrow}\mathbb{A})^{\overleftarrow{\Phi}}\left(\nabla_\Phi a, \Phi_*\overrightarrow{\beta}\right) \;\cong\; \Phi_*\overrightarrow{\beta}(a) \tag{10}$$

For every $b \in \mathbb{B}$ and every $\overleftarrow{\alpha} \in {\Downarrow}\mathbb{A}$ there is a natural bijection

$$({\Uparrow}\mathbb{B})^{\overrightarrow{\Phi}}\left(\Phi^*\overleftarrow{\alpha}, \Delta_\Phi b\right) \;\cong\; \Phi^*\overleftarrow{\alpha}(b) \tag{11}$$

▶ **Corollary 2.11** (Matrix Yoneda embedding).

$$({\Downarrow}\mathbb{A})^{\overleftarrow{\Phi}}\left(\nabla_\Phi a, \Phi_\# b\right) \;\cong\; \Phi(a,b) \;\cong\; ({\Uparrow}\mathbb{B})^{\overrightarrow{\Phi}}\left(\Phi^\# a, \Delta_\Phi b\right) \tag{12}$$

2.3.2 Completeness and generation

▶ **Corollary 2.12.** *$(\Downarrow\mathbb{A})^{\overleftarrow{\Phi}}$ is $\overrightarrow{\lim}_{\Phi}$-complete. $(\Uparrow\mathbb{B})^{\overrightarrow{\Phi}}$ is $\overleftarrow{\lim}_{\Phi}$-complete.*

▶ **Proposition 2.13.** *Every $\overleftarrow{\Phi}$-algebra is a limit inferior in $(\Downarrow\mathbb{A})^{\overleftarrow{\Phi}}$ of $\overleftarrow{\Phi}$-algebra representables. Every $\overrightarrow{\Phi}$-coalgebra is a limit superior in $(\Uparrow\mathbb{B})^{\overrightarrow{\Phi}}$ of $\overrightarrow{\Phi}$-coalgebra representables.*

▶ **Theorem 2.14.** *The Φ-extended Yoneda embeddings realize the $\overrightarrow{\lim}_{\Phi}$-completion and $\overleftarrow{\lim}_{\Phi}$-completion:*

- $\nabla_{\Phi} : \mathbb{A} \xrightarrow{\nabla} \Downarrow\mathbb{A} \xrightarrow{\overline{\Phi}} (\Downarrow\mathbb{A})^{\overleftarrow{\Phi}}$ *is the $\overrightarrow{\lim}$-completion of $\mathbb{A}$, whereas*
- $\Delta_{\Phi} : \mathbb{B} \xrightarrow{\Delta} \Uparrow\mathbb{B} \xrightarrow{\underline{\Phi}} (\Uparrow\mathbb{B})^{\overrightarrow{\Phi}}$ *is the $\overleftarrow{\lim}$-completion of $\mathbb{B}$.*

2.4 Minimal bicompletion of a matrix

2.4.1 Loose extensions

In general, a matrix $\Phi\colon \mathbb{A} \looparrowright \mathbb{B}$ always induces a *loose* extension $\Updownarrow\Phi\colon (\Downarrow\mathbb{A})^{\overleftarrow{\Phi}} \looparrowright (\Uparrow\mathbb{B})^{\overrightarrow{\Phi}}$, defined

$$\Updownarrow\Phi(a,b) \;=\; \left\{ f \in \Downarrow\mathbb{A}\Big(\overleftarrow{\alpha}, \Phi_*\overrightarrow{\beta}\Big) \;\middle|\; \begin{tikzcd} \Phi_*\Phi^*\overleftarrow{\alpha} \arrow[r, "\Phi_*\Phi^*f"] \arrow[d, "a"'] & \Phi_*\Phi^*\Phi_*\overrightarrow{\beta} \\ \overleftarrow{\alpha} \arrow[r, "f"'] & \Phi_*\overrightarrow{\beta} \arrow[u, "\Phi_*b"'] \end{tikzcd} \right\} \tag{13}$$

▶ **Proposition 2.15.** *Each of the following squares commutes if and only if the other one commutes.*

$$\begin{tikzcd} \Phi_*\Phi^*\overleftarrow{\alpha} \arrow[r, "\Phi_*\Phi^*f"] \arrow[d, "a"'] & \Phi_*\Phi^*\Phi_*\overrightarrow{\beta} \\ \overleftarrow{\alpha} \arrow[r, "f"'] & \Phi_*\overrightarrow{\beta} \arrow[u, "\Phi_*b"'] \end{tikzcd} \iff \begin{tikzcd} \Phi^*\Phi_*\Phi^*\overleftarrow{\alpha} \arrow[r, "\Phi^*\Phi_*f'"] \arrow[d, "\Phi^*a"'] & \Phi^*\Phi_*\overrightarrow{\beta} \\ \Phi^*\overleftarrow{\alpha} \arrow[r, "f'"'] & \beta \arrow[u, "b"'] \end{tikzcd}$$

The commutativity of the preceding squares implies the commutativity of the following squares, which are each other's transposes.

$$\begin{tikzcd} \overleftarrow{\alpha} \arrow[r, "f"] & \Phi_*\overrightarrow{\beta} \arrow[r, shift left, "\Phi_*b"] \arrow[r, shift right, "\eta"'] & \Phi_*\Phi^*\Phi_*\overrightarrow{\beta} \end{tikzcd} \iff \begin{tikzcd} \Phi^*\Phi_*\Phi^*\overleftarrow{\alpha} \arrow[r, "\Phi^*\Phi_*f'"] & \Phi^*\Phi_*\overrightarrow{\beta} \\ \Phi^*\overleftarrow{\alpha} \arrow[u, "\Phi_*\eta"] \arrow[r, "f'"'] & \overrightarrow{\beta} \arrow[u, "b"'] \end{tikzcd}$$

$$\begin{tikzcd} \Phi^*\Phi_*\Phi^*\overleftarrow{\alpha} \arrow[r, shift left, "\Phi^*a"] \arrow[r, shift right, "\varepsilon"'] & \Phi^*\overleftarrow{\alpha} \arrow[r, "f'"] & \overrightarrow{\beta} \end{tikzcd} \iff \begin{tikzcd} \Phi_*\Phi^*\overleftarrow{\alpha} \arrow[r, "\Phi_*\Phi^*f'"] \arrow[d, "a"'] & \Phi_*\Phi^*\Phi_*\overrightarrow{\beta} \arrow[d, "\Phi_*\varepsilon"] \\ \overleftarrow{\alpha} \arrow[r, "f'"'] & \Phi_*\overrightarrow{\beta} \end{tikzcd}$$

▶ **Conjecture 2.16.** $\Updownarrow\Phi$ *isomorphic with the matrix*

$$\Updownarrow\Phi(a,b) \quad = \quad \left(\Downarrow(\mathbb{A}\times\mathbb{B}^o)\right)^{\overleftarrow{\Phi}\times\overrightarrow{\Phi}}\left(\overleftarrow{\alpha}\times\overrightarrow{\beta},\Phi\right)$$

which is equivalent to the matrix of the adjunction $\Phi^{\circledast}\dashv\Phi_{\circledast}\colon(\Uparrow\mathbb{B})^{\overrightarrow{\Phi}}\to(\Downarrow\mathbb{A})^{\overleftarrow{\Phi}}$*, defined*

$$\Phi^{\circledast}\overleftarrow{\alpha}(u) \quad = \quad (\Downarrow\mathbb{A})^{\overleftarrow{\Phi}}\left(\overleftarrow{\alpha},\Phi^{\#}u\right) \tag{14}$$

$$\Phi_{\circledast}\overrightarrow{\beta}(\ell) \quad = \quad (\Uparrow\mathbb{B})^{\overrightarrow{\Phi}}\left(\Phi_{\#}\ell,\overrightarrow{\beta}\right) \tag{15}$$

with the structure maps induced by composition with the structure maps $a\colon\Phi_*\Phi^*\overleftarrow{\alpha}\to\overleftarrow{\alpha}$ *and* $b\colon\overrightarrow{\beta}\to\Phi^*\Phi_*\overrightarrow{\beta}$*.*

2.4.2 Tight extensions

But this loose extension is of little semantical value. E.g., when Φ is a partial ordering like in (3), $\Updownarrow\Phi$ picks all pairs of a saturated lower set and a saturated upper set which are contained in each other's sets of bounds, but do not necessarily contain *all* such bounds. So it does not capture the Dedekind cuts.

The tight extension $\overleftrightarrow{\Phi}$ brings us closer to the Dedekind cuts:

$$\overleftrightarrow{\Phi}(a,b) \quad = \quad \{f\in\Updownarrow\Phi(a,b) \mid f \text{ is mono, and } f' \text{ is epi}\} \tag{16}$$

Since $(\Downarrow\mathbb{A})^{\overleftarrow{\Phi}}$ and $(\Uparrow\mathbb{B})^{\overrightarrow{\Phi}}$ are regular categories, $\overleftrightarrow{\Phi}$ can be extracted from $\Updownarrow\Phi$ by two closure operators: first extracting the mono factors, and then the epis of their transposes, or equivalently the other way around. After the factorizations, in the first case the transpose of the resulting epi will be mono; in the second the transpose of the resulting mono will be epi. Either way, the process will stop.

The resulting matrix $\overleftrightarrow{\Phi}$ will be a reflective submatrix of $\Updownarrow\Phi$. The completeness and the generation will be inherited, but tight. We need to prove that the inferior limits that existed in $\mathbb{A}$ and the superior limits that existed in $\mathbb{B}$ are preserved.

▶ **Conjecture 2.17.** *For every matrix* $\Phi\colon\mathbb{A}\to\mathbb{B}$*, the tight extension* $\overleftrightarrow{\Phi}\colon(\Downarrow\mathbb{A})^{\overleftarrow{\Phi}}\looparrowright(\Uparrow\mathbb{B})^{\overrightarrow{\Phi}}$ *is the minimal bicompletion.*

3 When does limit inferior boil down to limit?

By the couniversal property of the (Eilenberg-Moore) categories of algebras for a monad [16, Part 0.6], there are always the comparison adjunctions between $\Downarrow\mathbb{A}$ and $(\Uparrow\mathbb{B})^{\overrightarrow{\Phi}}$, and between $\Uparrow\mathbb{B}$ and $(\Downarrow\mathbb{A})^{\overleftarrow{\Phi}}$, as displayed in the leftmost diagram of Fig. 4, since the monad $\overleftarrow{\Phi}$ and the comonad $\overrightarrow{\Phi}$ are induced by the adjunction $\Phi^*\dashv\Phi_*\colon\Uparrow\mathbb{B}\to\Downarrow\mathbb{A}$. When these comparisons are equivalences, then this adjunction transfers to the two Eilenberg-Moore categories, as indicated in the rightmost diagram of Fig. 4. Moreover, the inferior Φ-limits $\overrightarrow{\lim}_{\Phi}$ in $\mathbb{B}$ then boil down to the suprema $\underrightarrow{\lim}$ in $\mathbb{A}$, whereas the superior Φ-limits $\overleftarrow{\lim}_{\Phi}$ in $\mathbb{A}$ boil down to the infima $\underleftarrow{\lim}$ in $\mathbb{B}$. In terms of the concept mining example from the Introduction, the structural components represented in $(\Downarrow\mathbb{A})^{\overleftarrow{\Phi}}$ can be computed as infima functions in $\Uparrow\mathbb{B}$, whereas the functional modules represented in $(\Uparrow\mathbb{B})^{\overrightarrow{\Phi}}$ can be computed as suprema of parts in $\Downarrow\mathbb{A}$. Connecting the extensions $\Updownarrow\Phi$ and $\overleftrightarrow{\Phi}$ along the equivalences $\Downarrow\mathbb{A}\simeq(\Uparrow\mathbb{B})^{\overrightarrow{\Phi}}$ and $\Uparrow\mathbb{B}\simeq(\Downarrow\mathbb{A})^{\overleftarrow{\Phi}}$ shows that all loose extensions are already tight.

$$\begin{tikzcd}
\Downarrow\mathbb{A} \arrow[loop above, "\overleftarrow{\Phi}"] \arrow[r, bend left, "\overline{\Phi}"] \arrow[r, phantom, "\perp"] \arrow[d, bend right, "\Phi^*"'] \arrow[d, phantom, "\dashv"] \arrow[dr, dashed] & (\Downarrow\mathbb{A})^{\overleftarrow{\Phi}} \arrow[l, bend left, "U"] \\
\Uparrow\mathbb{B} \arrow[loop below, "\overrightarrow{\Phi}"] \arrow[u, bend right, "\Phi_*"'] \arrow[r, bend right, "\underline{\Phi}"'] \arrow[r, phantom, "\perp"] \arrow[ur, dashed] & (\Uparrow\mathbb{B})^{\overrightarrow{\Phi}} \arrow[l, bend right, "V"']
\end{tikzcd}$$

$$\begin{tikzcd}
\Downarrow\mathbb{A} \arrow[loop above, "\overleftarrow{\Phi}"] \arrow[rrr, bend left, "\overline{\Phi}"] \arrow[rrr, phantom, "\perp"] \arrow[ddd, bend right, "\Phi^*"'] \arrow[ddd, phantom, "\dashv"] & & & (\Downarrow\mathbb{A})^{\overleftarrow{\Phi}} \arrow[lll, bend left, "U"] \arrow[dl, equal] \\
& \mathcal{C} \arrow[ul] \arrow[loop above, "\overleftarrow{\Phi}|_{\mathcal{C}}"] \arrow[r, bend left] \arrow[r, phantom, "\perp"] \arrow[d, bend right, "\Phi^*|_{\mathcal{C}}"'] \arrow[d, phantom, "\dashv"] \arrow[dr, dashed] & \mathcal{C}^{\overleftarrow{\Phi}|_{\mathcal{C}}} \arrow[l, bend left] & \\
& \mathcal{D} \arrow[dl] \arrow[loop below, "\overrightarrow{\Phi}|_{\mathcal{D}}"] \arrow[u, bend right, "\Phi_*|_{\mathcal{D}}"'] \arrow[r, bend right] \arrow[r, phantom, "\perp"] \arrow[ur, dashed] & \mathcal{D}^{\overrightarrow{\Phi}|_{\mathcal{D}}} \arrow[l, bend right] \arrow[dr, equal] & \\
\Uparrow\mathbb{B} \arrow[loop below, "\overrightarrow{\Phi}"] \arrow[uuu, bend right, "\Phi_*"'] \arrow[rrr, bend right, "\underline{\Phi}"'] \arrow[rrr, phantom, "\perp"] & & & (\Uparrow\mathbb{B})^{\overrightarrow{\Phi}} \arrow[lll, bend right, "V"']
\end{tikzcd}$$

$$\begin{tikzcd}
\Downarrow\mathbb{A} \arrow[loop above, "\overleftarrow{\Phi}"] \arrow[d, bend right, "\Phi^*"'] \arrow[d, phantom, "\dashv"] & (\Downarrow\mathbb{A})^{\overleftarrow{\Phi}} \arrow[dl, hook] \arrow[d, bend left] \arrow[d, phantom, "\dashv"] \\
\Uparrow\mathbb{B} \arrow[loop below, "\overrightarrow{\Phi}"] \arrow[u, bend right, "\Phi_*"'] & (\Uparrow\mathbb{B})^{\overrightarrow{\Phi}} \arrow[ul, hook] \arrow[u, bend left]
\end{tikzcd}$$

Figure 4 Comparisons between the $\varprojlim$- and $\overrightarrow{\lim}_\Phi$-completions, and between the $\varinjlim$- and $\overleftarrow{\lim}_\Phi$-completions.

▶ **Proposition 3.1.** *For any matrix $\Phi\colon \mathbb{A} \nrightarrow \mathbb{B}$, the extensions $\Phi^* \dashv \Phi_*\colon \Uparrow\mathbb{B} \to \Downarrow\mathbb{A}$ are both monadic if and only if the loose and the tight extensions coincide, i.e. $\Updownarrow\Phi \simeq \overleftrightarrow{\Phi}$.*

The notion of *monadicity* [20, Sec. VI.7] here precisely captures the equivalences of interest, as $\Uparrow\mathbb{B} \simeq (\Downarrow\mathbb{A})^{\overleftarrow{\Phi}}$ means that Φ_* is monadic and $\Downarrow\mathbb{A} \simeq (\Uparrow\mathbb{B})^{\overrightarrow{\Phi}}$ means that Φ^* is (co)monadic. In this section, we study the monadicity of the extensions Φ_* and Φ^* in order to gain insight into the situations when the inferior and superior limits boil down to the ordinary limits, and the situations when they genuinely provide new information.

3.1 Monadicity workflow

As a reminder, we quote the Precise Monadicity Theorem in Appendix B. Intuitively, its impact on the concrete instances of our situation is that it allows constructing the inferior limits, which are in principle the suprema of lower bounds, as specific maximal cones into the infima.

We begin describing a convenient setting of subcategories, as displayed in the middle in Fig. 4. When $\Phi_*\colon \Uparrow\mathbb{B} \to \Downarrow\mathbb{A}$ restricts to a monadic functor $\mathcal{D} \to \mathcal{C}$, so that $\mathcal{D} \simeq \mathcal{C}^{\overleftarrow{\Phi}|_{\mathcal{C}}}$, then we have an embedding $(\Downarrow\mathbb{A})^{\overleftarrow{\Phi}} \rightarrowtail \Uparrow\mathbb{B}$ as indicated in the rightmost diagram in Fig. 4.

In the general framework of an adjunction as in Appendix B, items (a–b) of the Monadicity Theorem say that the induced Eilenberg-Moore category $\mathcal{C}^T$ is coreflective within the category $\mathcal{D}$ whenever $\mathcal{D}$ has and U preserves reflexive U-split coequalizers. However, its converse does not hold (see Example 3.6 and Prop. 3.8). The task is thus to spell out the full subcategories $\mathcal{C} \subseteq \Downarrow\mathbb{A}$, $\mathcal{D} \subseteq \Uparrow\mathbb{B}$ explicitly, even if we cannot apply the Monadicity Theorem to the setting $\mathcal{C} = \Downarrow\mathbb{A}$, $\mathcal{D} = \Uparrow\mathbb{B}$. Towards this goal, and to simplify calculations with the algebras, we propose the following lemma.

▶ **Definition 3.2.** An object B is said to be a retract of an object A if there exist morphisms $B \to A \to B$ whose composite is id_B. For a full subcategory $\mathcal{F} \subseteq \mathcal{E}$, we denote by $\mathrm{Retr}_{\mathcal{E}}(\mathcal{F}) \subseteq \mathcal{E}$ the full subcategory of all retracts in $\mathcal{E}$ of objects in $\mathcal{F}$.

Notational conventions. For a functor G, we denote its full image by $\operatorname{Im} G$. For a category $\mathcal{E}$ and its full subcategories $\mathcal{F}, \mathcal{F}'$, we loosely use $\mathcal{F} \subseteq \mathcal{F}'$ to denote that any object in $\mathcal{F}$ is isomorphic in $\mathcal{E}$ to some object in $\mathcal{F}'$.

▶ **Lemma 3.3.** *Let $F \dashv U \colon \mathcal{B} \to \mathcal{A}$ be an adjunction and T be its monad on $\mathcal{A}$.*

$$\begin{tikzcd}
\mathcal{A} \arrow[loop above, "T"] \arrow[rr, bend right=10] \arrow[dd, bend right, "F"'] \arrow[dd, phantom, "\dashv"] & & \mathcal{A}^T \arrow[ll, bend right=10] \arrow[ll, phantom, "\top"] \\
& \mathcal{C} \arrow[loop above, "T|_{\mathcal{C}}"] \arrow[ul, hook] \arrow[r, bend right] \arrow[d, bend right, "F|_{\mathcal{C}}"'] \arrow[d, phantom, "\dashv"] & \mathcal{C}^{T|_{\mathcal{C}}} \arrow[l, bend right] \arrow[l, phantom, "\top"] \arrow[u, tail] \\
\mathcal{B} \arrow[uu, bend right, "U"'] \arrow[r, equal] & \mathcal{B} \arrow[u, bend right, "U"'] &
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathcal{A} \arrow[loop above, "T"] \arrow[r, bend right=10] \arrow[dd, bend right, "F"'] \arrow[dd, phantom, "\dashv"] & \mathcal{A}^T \arrow[l, bend right=10] \arrow[l, phantom, "\top"] \arrow[ddl, bend left, "L"] \\
& \\
\mathcal{B} \arrow[uu, bend right, "U"'] \arrow[uur, "K"] \arrow[uur, phantom, "\top"] & \mathcal{D} \arrow[uu, tail, bend right] \arrow[l, tail, bend left]
\end{tikzcd}$$

1. *Let $\mathcal{C} \subseteq \mathcal{A}$ be a full subcategory such that $\operatorname{Im} U \subseteq \mathcal{C}$. If $\mathrm{Retr}_{\mathcal{A}}(\operatorname{Im} U) \subseteq \mathcal{C}$, then the canonical inclusion is an equivalence of categories $\mathcal{C}^{T|_{\mathcal{C}}} \simeq \mathcal{A}^T$.*
2. *Let $\mathcal{D}$ be the full subcategory that depicts the equivalence induced by the comparison functor $K \colon \mathcal{B} \to \mathcal{A}^T$ and its partial left adjoint $L \colon \mathcal{A}^T \rightharpoonup \mathcal{B}$ (see e.g. [14, Sec. 1.11]). Then, $\mathrm{Retr}_{\mathcal{B}}(\operatorname{Im} F) \subseteq \mathcal{D}$. In particular, $\mathrm{Retr}_{\mathcal{B}}(\operatorname{Im} F) \subseteq \mathcal{A}^T$ if the category $\mathcal{A}^T$ is a (coreflective) subcategory of $\mathcal{B}$ by $L \colon \mathcal{A}^T \rightarrowtail \mathcal{B}$.*

The above lemma intuitively means

- we need at most retracts of images under U in $\mathcal{A}$, and that
- we need at least retracts of images under F in $\mathcal{B}$,

in order to obtain a monadic functor of the form $\mathcal{A}^T \simeq \mathcal{D} \xrightarrow{U|_{\mathcal{D}}} \mathcal{C}$. In the later discussion, we restrict an adjunction as the diagram below and calculate the category of T-algebras by $\mathcal{A}^T = (\mathrm{Retr}(\operatorname{Im} U))^{T'} \supseteq (\mathrm{Retr}(\operatorname{Im} F))$.

$$\begin{tikzcd}
\mathcal{A} \arrow[loop above, "T"] \arrow[d, bend right, "F"'] \arrow[d, phantom, "\dashv"] & \mathrm{Retr}(\operatorname{Im} U) \arrow[loop above, "T'"] \arrow[l, tail] \arrow[r, bend right] \arrow[d, bend right, "F'"'] \arrow[d, phantom, "\dashv"] & \mathcal{A}^T \arrow[l, bend right] \arrow[l, phantom, "\top"] \\
\mathcal{B} \arrow[u, bend right, "U"'] & \mathrm{Retr}(\operatorname{Im} F) \arrow[l, tail] \arrow[u, bend right, "U'"'] \arrow[ur, tail, bend right, "K'"'] &
\end{tikzcd}
\quad \text{where} \quad
\begin{aligned}
T' &= U'F' = T|_{\mathrm{Retr}(\operatorname{Im} U)} \ , \\
F' &= F|_{\mathrm{Retr}(\operatorname{Im} U)} \ , \\
U' &= U|_{\mathrm{Retr}(\operatorname{Im} F)} \ , \\
K' &= K|_{\mathrm{Retr}(\operatorname{Im} F)}
\end{aligned}$$

3.2 Completing constant matrices

Any set R can be viewed as a constant matrix $\widetilde{R} \colon 1 \looparrowright 1$ by setting $\widetilde{R}(0,0) = R$, where $1 = \{0\}$. We abuse notation and write $\widetilde{R}$ as R. The extensions $R^* \dashv R_* \colon \mathsf{Set}^o \to \mathsf{Set}$ are thus $R^*X = R_*X = R^X$, and they induce the continuation monad $\overleftarrow{R}X = R^{R^X}$ on Set, and the same comonad $\overrightarrow{R}$ on Set^o.

Lem. B.2 in the Appendix B helps characterizing the monadicity of R^* and R_*.

▶ **Proposition 3.4.** *For a set R with at least 2 elements, the functor $R_* \colon \mathsf{Set}^o \to \mathsf{Set}$ is monadic. When R is a singleton, then the monad $\overleftarrow{R}$ on Set has a single algebra, and the comonad $\overrightarrow{R}$ on Set^o has a single coalgebra. When R is empty, then they have two algebras and coalgebras respectively.*

▶ **Corollary 3.5.** *The loose extension of the constant matrix R is always in the form $\Updownarrow R$: $\mathsf{Set} \looparrowright \mathsf{Set}$ with $\Updownarrow R(X,Y) = \mathsf{Set}(X,Y)$. The tight extension is*

- $\overleftrightarrow{R} = \Updownarrow R : \mathsf{Set} \looparrowright \mathsf{Set}$ *when R has at least 2 elements*
- $\overleftrightarrow{1} : 1 \looparrowright 1$ *with $\overleftrightarrow{1}(0,0) = 1$, where $1 = \{0\}$*
- $\overleftrightarrow{0} : 2 \looparrowright 2$ *with $\overleftrightarrow{0}(x,y) = 1$ if and only if $x \leq y$ within $2 = \{0 < 1\}$.*

3.3 Completing groups

Let $\mathbb{C}$ be a group G, viewed as a one-object category with invertible morphisms. The category $\Downarrow G$ of presheaves is the category of right G-sets, or the category $G^o\text{-}\mathsf{Set}$ of (left) G^o-sets. Indeed as a discrete fibration over the one-object category, the total category of the presheaf is a set X with an action $X \times G \to X$. The adjunction $H^* \dashv H_*$ is given explicitly as follows. We think of G as a (left G, right G)-set by the multiplication. For a right G-set X, the (left) G-set H^*X is the set $G^o\text{-}\mathsf{Set}(X, G)$ with the action $(g \cdot f)(i) = g\big(f(i)\big)$. Similarly, $H_*Y = G\text{-}\mathsf{Set}(Y, G)$ with the right action $(f \cdot g)(i) = \big(f(i)\big)g$ for a left G-set Y.

We assume later that the group G is nontrivial (i.e. G has at least two elements).

▶ **Example 3.6.** The diagram $0 \to 1 \rightrightarrows 1 + 1$ displays an equalizer of a reflexive H_*-split pair of left G-maps (i.e. a coequalizer in $\Uparrow G$). However, the image of this diagram under H_* is not a coequalizer: $1 \leftarrow 0 \leftleftarrows 0$.

▶ **Proposition 3.7.** *We have* $\operatorname{Im} H^* \simeq \{0\} \cup \{G^I \mid I \in \mathsf{Set}\}$ *and* $\operatorname{Retr}(\operatorname{Im} H^*) \simeq \{1\} \cup \{G \times I \mid I \in \mathsf{Set}\}$*, where* G^I *is the exponential in* Set *with the pointwise multiplication* $(g \cdot f)(i) = g\big(f(i)\big)$ *and* $G \times I$ *is the free* G*-set generated by the set* I *(i.e.* $g \cdot \langle h, i\rangle = \langle gh, i\rangle$*).*

We denote by $G\text{-}\mathsf{Set}_{1,\text{free}}$ the full subcategory $\{1\} \cup \{G \times I \mid I \in \mathsf{Set}\} \subseteq G\text{-}\mathsf{Set}$ of a singleton and free G-sets.

▶ **Proposition 3.8.** *The functor* $H_* \colon (G\text{-}\mathsf{Set}_{1,\text{free}})^o \to G^o\text{-}\mathsf{Set}_{1,\text{free}}$ *is monadic. In particular, the category* $(G^o\text{-}\mathsf{Set})^{\overleftarrow{H}}$ *of* $\overleftarrow{H}$*-algebras is equivalent to* $(G\text{-}\mathsf{Set}_{1,\text{free}})^o$*.*

▶ **Corollary 3.9.** *The loose extension of a nontrivial group is the canonical connection of its left and right actions. The tight extension is the canonical extension of its free actions.*

3.4 Completing posets

Let $\mathbb{C}$ be a poset $(P, \leq)$. We denote the poset of lower sets of P by $\downarrow P$, and the poset of upper sets of P by $\uparrow P$.[6] They are respectively the join and the meet completions. While P's categorical supremum completion $\Downarrow P = \mathsf{Set}^{P^o}$ and its infimum completion $\Uparrow P = \big(\mathsf{Set}^P\big)^o$ are proper categories, its limit inferior completion $\Downarrow P^{\overleftarrow{H}}$, and its limit superior completion $\Uparrow P^{\overrightarrow{H}}$, although still constructed over Set — turn out to be both equivalent to a lattice, and in particular to P's Dedekind-MacNeille completion $\updownarrow P$.

▶ **Lemma 3.10.** *The lattice of subobjects of the terminal object in* $\Downarrow P$ *is isomorphic to* $\downarrow P$*. The lattice of quotient objects of the initial object of* $\Uparrow P$ *is isomorphic to* $\uparrow P$*. The full subcategories* $\downarrow P \subseteq \Downarrow P$ *and* $\uparrow P \subseteq \Uparrow P$ *contain all the representables.*

▶ **Lemma 3.11.** *The adjunction* $H^* \dashv H_* \colon \Uparrow P \to \Downarrow P$ *restricts to an adjunction between posets* $\downarrow P, \uparrow P$ *(i.e. a Galois connection), which coincides with the* $\{0, 1\}$*-enriched construction. Moreover,* $\operatorname{Im} H^* = \operatorname{Im}(H^*|_{\downarrow P})$ *and* $\operatorname{Im} H_* = \operatorname{Im}(H_*|_{\uparrow P})$*.*

▶ **Corollary 3.12.** *It holds* $\operatorname{Retr}_{\Uparrow P}(\operatorname{Im} H^*) = \operatorname{Im}(H^*|_{\downarrow P})$*.*

Therefore, the category $(\Downarrow P)^{\overleftarrow{H}}$ is nothing more than the category of algebras for the adjunction $\uparrow P \leftrightarrows \downarrow P$.

▶ **Proposition 3.13.** *There exist equivalences of categories* $(\Downarrow P)^{\overleftarrow{H}} \simeq \updownarrow P \simeq (\Uparrow P)^{\overrightarrow{H}}$*.*

▶ **Corollary 3.14.** *The tight extension* $\overleftrightarrow{P}$ *of a poset* P *coincides with its Dedekind-MacNeille completion* $\updownarrow P$*.*

[6] The poset $\downarrow P$ is ordered by $L \leq L' \iff L \subseteq L'$, whereas $\uparrow P$ is ordered by $U \leq U' \iff U \supseteq U'$.

$\Phi = \underline{1}\Phi_1 + \underline{p}\Phi_p$	$\Phi_p = 0$	$\Phi_p \geq 1$
$\Phi_1 = 0$	$\{0,1\}$ $\downarrow\dashv\uparrow$ $\{0,\underline{1}\}^o$	Set $\downarrow\dashv\uparrow$ $(\{\underline{1}\} \cup \{\underline{p}U_p \mid U_p \in \text{Set}\})^o$
$\Phi_1 = 1$	$\{1\}$ $\downarrow\dashv\uparrow$ $\{\underline{1}\}^o$	Set $\downarrow\dashv\uparrow$ $\{\underline{1} + \underline{p}U_p \mid U_p \in \text{Set}\}^o$
$\Phi_1 \geq 2$	Set $\downarrow\dashv\uparrow$ $\{\underline{1}U_1 \mid U_1 \in \text{Set}\}^o$	Set $\downarrow\dashv\uparrow$ $(\mathbb{Z}_p\text{-Set})^o$

Figure 5 The $\overleftarrow{\lim}_\Phi$-completion and the $\overrightarrow{\lim}_\Phi$-completion of a $\mathbb{Z}_p$-vector $\Phi\colon 1 \looparrowright \mathbb{Z}_p$.

3.5 Completing a $\mathbb{Z}_p$-vector

A *vector* is a matrix in the form $\Phi\colon 1 \looparrowright \mathbb{B}$. We consider the vectors in $\mathbb{B} = \mathbb{Z}_p$, viewed as an additive cyclic group of prime order p. Every $\mathbb{Z}_p$-set X has an orbit-decomposition $X \cong 1 \times X_1 + \mathbb{Z}_p \times X_p$ where the action on 1 is trivial and the action on $\mathbb{Z}_p$ is defined by the addition.. We abbreviate the decomposition $1 \times X_1 + \mathbb{Z}_p \times X_p$ as $\underline{1}X_1 + \underline{p}X_p$.

▶ **Lemma 3.15.** $\mathbb{Z}_p\text{-Set}(\underline{1}X_1 + \underline{p}X_p, \underline{1}Y_1 + \underline{p}Y_p) \ \cong\ Y_1^{X_1}(Y_1 + pY_p)^{X_p}$.

Hence for a vector $\Phi = \underline{1}\Phi_1 + \underline{p}\Phi_p$, the adjunction $\Phi^* \dashv \Phi_*\colon (\mathbb{Z}_p\text{-Set})^o \to \text{Set}$ is explicitly

$$\begin{aligned} \Phi^*L \ &\cong\ (\underline{1}\Phi_1 + \underline{p}\Phi_p)^L && (L \in \text{Set}), \\ \Phi_*U \ &\cong\ \Phi_1^{U_1}(\Phi_1 + p\Phi_p)^{U_p} && (U = \underline{1}U_1 + \underline{p}U_p \in \mathbb{Z}_p\text{-Set}). \end{aligned}$$

▶ **Lemma 3.16.** *Let $f, g\colon U \rightrightarrows U'$ be a reflexive pair in $\mathbb{Z}_p$-Set. The $\mathbb{Z}_p$-sets U, U' have suitable isomorphisms to their orbit-decompositions such that the right square of the diagram*

$$\begin{tikzcd}
E \arrow[r, "e"] \arrow[d, dashed, "\cong"] & U \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] \arrow[d, "\cong"] & U' \arrow[d, "\cong"] \\
\underline{1}E_1 + \underline{p}E_p \arrow[r, "\underline{1}e_1 + \underline{p}e_p"] & \underline{1}U_1 + \underline{p}U_p \arrow[r, shift left, "\underline{1}f_1 + \underline{p}f_p"] \arrow[r, shift right, "\underline{1}g_1 + \underline{p}g_p"'] & \underline{1}U'_1 + \underline{p}U'_p
\end{tikzcd}$$

serially commutes for some maps $f_1, g_1\colon U_1 \rightrightarrows U'_1$, $f_p, g_p\colon U_p \rightrightarrows U'_p$. Moreover, equalizers $E_1 \xrightarrow{e_1} U_1$, $E_p \xrightarrow{e_p} U_p$ of the pairs (f_1, g_1), (f_p, g_p), which induce an equalizer $E \xrightarrow{e} U$ of the pair (f, g) in $\mathbb{Z}_p$-Set, satisfy the condition of Lem. B.2.2.

Let us find full subcategories $\operatorname{Retr}(\operatorname{Im}\Phi_*) \subseteq \mathcal{C} \subseteq \text{Set}$ and $\operatorname{Retr}(\operatorname{Im}\Phi^*) \subseteq \mathcal{D} \subseteq (\mathbb{Z}_p\text{-Set})^o$ to fit the scheme of Fig. 4.

▶ **Proposition 3.17.** *Fig. 5 depicts a restriction of the adjunction $\Phi^* \dashv \Phi_*\colon (\mathbb{Z}_p\text{-Set})^o \to \text{Set}$ that makes both Φ^* and Φ_* monadic, without changing the categories of algebras:*

$$\begin{tikzcd}
\text{Set} \arrow[d, bend right, "\Phi^*"'] \arrow[d, phantom, "\dashv"] & \mathcal{C} \arrow[l, tail] \arrow[d, bend right] \arrow[d, phantom, "\dashv"] \arrow[dr, "\simeq"] & \mathcal{C}^{\overleftarrow{\Phi}|_\mathcal{C}} \arrow[r, equal] & \text{Set}^{\overleftarrow{\Phi}} \\
(\mathbb{Z}_p\text{-Set})^o \arrow[u, bend right, "\Phi_*"'] & \mathcal{D} \arrow[l, tail] \arrow[u, bend right] \arrow[ur, "\simeq"'] & \mathcal{D}^{\overrightarrow{\Phi}|_\mathcal{D}} \arrow[r, equal] & (\mathbb{Z}_p\text{-Set}^o)^{\overrightarrow{\Phi}}
\end{tikzcd}.$$

Thus, the subcategories $\mathcal{C}$ and $\mathcal{D}$ are equivalent to the $\overleftarrow{\lim}_\Phi$- and $\overrightarrow{\lim}_\Phi$-completions, respectively.

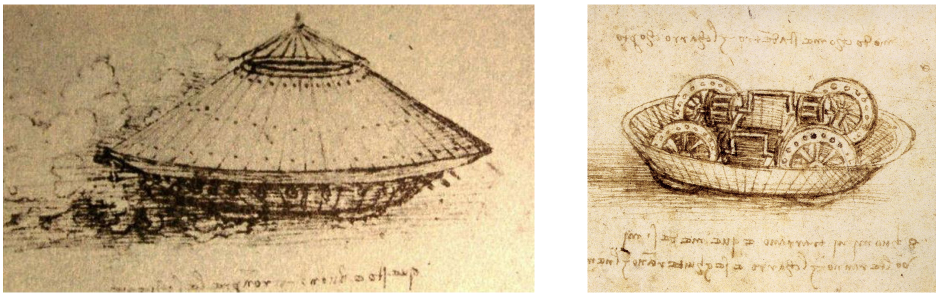


Figure 6 Identified object: The external and the internal view

4 Conclusion

Deploying the categorical concept analysis of the unidentified object from Fig. 1 according to the technical recipes proposed in this paper, our diligent reader has surely uncovered that the mysterious device consists of two main structural components: the internal mechanism of wheels and gears, and the external protection shell. On the other hand, the detailed categorical analysis has surely displayed three main functional modules: moving, defending from the outside attacks, and attacking from inside. As desired, the tight matrix then clearly shows that the object must be a model of a man-powered armored combat vehicle from XV century. It was conceived by Leonardo da Vinci, whose drawings are reproduced on Fig. 6. The advances of category theory will undoubtedly permit us to better understand Leonardo's conceptualizations of warfare.

References

1 Yossi Azar, Amos Fiat, Anna Karlin, Frank McSherry, and Jared Saia. Spectral analysis of data. In *Proceedings of the thirty-third annual ACM Symposium on Theory of Computing*, STOC'01, pages 619–626, New York, NY, USA, 2001. ACM.

2 Bernhard Banaschewski and Gunter Bruns. Categorical characterization of the MacNeille completion. *Archiv der Mathematik*, 18(4):369–377, September 1967.

3 Michael Barr and Charles Wells. *Toposes, Triples, and Theories.* Number 278 in Grundlehren der mathematischen Wissenschaften. Springer-Verlag, 1985.

4 Claudio Carpineto and Giovanni Romano. *Concept Data Analysis: Theory and Applications.* John Wiley & Sons, 2004.

5 Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification.* Wiley-Interscience, 2000.

6 Bernhard Ganter, Gerd Stumme, and Rudolf Wille, editors. *Formal Concept Analysis, Foundations and Applications*, volume 3626 of *Lecture Notes in Computer Science.* Springer, 2005.

7 Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations.* Springer, Berlin/Heidelberg, 1999.

8 Peter Gärdenfors. *The Geometry of Meaning: Semantics Based on Conceptual Spaces.* MIT Press, 2014.

9 Alexander Grothendieck. *Revêtements étales et groupe fondamental (SGA 1)*, volume 224 of *Lecture notes in mathematics.* Springer-Verlag, 1971.

10 John R. Isbell. Small subcategories and completeness. *Mathematical Systems Theory*, 2(1):27–50, 1968.

11 Nicolas Jardine and Robin Sibson. *Mathematical Taxonomy.* John Wiley & Sons, Ltd, 1971.

12 Peter Johnstone. *Stone Spaces*. Number 3 in Cambridge Studies in Advanced Mathematics. Cambridge University Press, 1982.

13 Ian T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer, 2002.

14 Gregory Max Kelly. *Basic Concepts of Enriched Category Theory*. Number 64 in London Mathematical Society Lecture Notes. Cambridge University Press, 1982.

15 Joachim. Lambek. *Completions of categories : seminar lectures given 1966 in Zurich*. Number 24 in Springer Lecture Notes in Mathematics. Springer-Verlag, 1966.

16 Joachim Lambek and Philip Scott. *Introduction to Higher Order Categorical Logic*. Number 7 in Cambridge Studies in Advanced Mathematics. Cambridge University Press, 1986.

17 Thomas K. Landauer, Danielle S. Mcnamara, Simon Dennis, and Walter Kintsch, editors. *Handbook of Latent Semantic Analysis*. Lawrence Erlbaum Associates, 2007.

18 F. William Lawvere. Metric spaces, generalised logic, and closed categories. *Rendiconti del Seminario Matematico e Fisico di Milano*, 43:135–166, 1973.

19 Saunders Mac Lane and Ieke Moerdijk. *Sheaves in Geometry and Logic: A First Introduction to Topos Theory*. Universitext. Springer-Verlag, New York, 1992.

20 Saunders MacLane. *Categories for the Working Mathematician*. Number 5 in Graduate Texts in Mathematics. Springer-Verlag, 1971.

21 Holbrook Mann MacNeille. Extensions of partially ordered sets. *Proc. Nat. Acad. Sci. USA*, 22(1):45–50, 1936.

22 Oded Maimon and Lior Rokach, editors. *Data Mining and Knowledge Discovery Handbook, 2nd ed.* Springer, 2010.

23 Dusko Pavlovic. Network as a computer: ranking paths to find flows. In Alexander Razborov and Anatol Slissenko, editors, *Proceedings of CSR 2008*, volume 5010 of *Lecture Notes in Computer Science*, pages 384–397. Springer Verlag, 2008. arxiv.org:0802.1306.

24 Dusko Pavlovic. On quantum statistics in data analysis. In Peter Bruza, editor, *Quantum Interaction 2008*. AAAI, 2008. arxiv.org:0802.1296.

25 Dusko Pavlovic. Quantifying and qualifying trust: Spectral decomposition of trust networks. In Pierpaolo Degano, Sandro Etalle, and Joshua Guttman, editors, *Proceedings of FAST 2010*, volume 6561 of *Lecture Notes in Computer Science*, pages 1–17. Springer Verlag, 2011. arxiv.org:1011.5696.

26 Dusko Pavlovic. Relating toy models of quantum computation: comprehension, complementarity and dagger autonomous categories. *E. Notes in Theor. Comp. Sci.*, 270(2):121–139, 2011. arxiv.org:1006.1011.

27 Dusko Pavlovic. Quantitative Concept Analysis. In Florent Domenach, Dmitry I. Ignatov, and Jonas Poelmans, editors, *Proceedings of ICFCA 2012*, volume 7278 of *Lecture Notes in Artificial Intelligence*, pages 260–277. Springer Verlag, 2012. arXiv:1204.5802.

28 Dusko Pavlovic. Bicompletions of distance matrices. In Bob Coecke, Luke Ong, and Prakash Panangaden, editors, *Computation, Logic, Games and Quantum Foundations. The Many Facets of Samson Abramsky*, volume 7860 of *Lecture Notes in Computer Science*, pages 291–310. Springer Verlag, 2013.

29 Dusko Pavlovic and Samson Abramsky. Specifying interaction categories. In E. Moggi and G. Rosolini, editors, *Category Theory and Computer Science 1997*, volume 1290 of *Lecture Notes in Computer Science*, pages 147–158. Springer Verlag, 1997.

30 Duško Pavlović and Martín Escardó. Calculus in coinductive form. In V. Pratt, editor, *Proceedings. Thirteenth Annual IEEE Symposium on Logic in Computer Science*, pages 408–417. IEEE Computer Society, 1998.

31 F. Ricci, L. Rokach, B. Shapira, and P.B. Kantor. *Recommender Systems Handbook*. Springer, 2010.

32 Ashok N. Srivastava and Mehran Sahami. *Text Mining: Classification, Clustering, and Applications*. Data Mining and Knowledge Discovery Series. CRC Press, 2009.

A Appendix: Proofs

Proof of Thm. 2.7. Suppose that $\mathbb{X}$ is a category with all limits inferior, and that $G\colon \mathbb{C} \to \mathbb{X}$ is an arbitrary functor. We show that G has a unique extension $G'\colon (\Downarrow\mathbb{C})^{\overleftarrow{H}} \to \mathbb{X}$, such that

$$G \;=\; G' \circ \overline{H} \circ \nabla \tag{17}$$

where $\overline{H}\colon \Downarrow\mathbb{C} \to (\Downarrow\mathbb{C})^{\overleftarrow{H}}$, as defined on Fig. 3 instantiated to $\Phi = H$, maps $\mathbb{C}$-presheaves to free $\overleftarrow{H}$-algebras, i.e. it is the left adjoint of the forgetful functor $U\colon (\Downarrow\mathbb{C})^{\overleftarrow{H}} \to \Downarrow\mathbb{C}$. The construction is illustrated on the following diagram.

$$\begin{tikzcd}
 & & (\Downarrow\mathbb{C})^{\overleftarrow{H}} \arrow[dd, dashed, "G'"] & & \overleftarrow{\gamma} & \overleftarrow{H}\overleftarrow{\gamma} \arrow[l, "a"'] \arrow[dd, mapsto, "G'"] \\
 & \Downarrow\mathbb{C} \arrow[ur, "\overline{H}"] & & & & \\
\mathbb{C} \arrow[ur, "\nabla"] \arrow[rr, "G"'] & & \mathbb{X} & & & \overrightarrow{\lim}\; G \circ \overrightarrow{\varphi}
\end{tikzcd}$$

Given an arbitrary $\overleftarrow{H}$-algebra $\overleftarrow{\gamma} \xleftarrow{\;a\;} \overleftarrow{H}\overleftarrow{\gamma}$ in $\Downarrow\mathbb{C}$, we construct the equalizer of postsheaves

$$\begin{tikzcd}
\overrightarrow{\varphi} \arrow[r, tail] & H_*\overleftarrow{\gamma} \arrow[r, shift left, "H_*a"] \arrow[r, shift right, "\eta_{H_*\overleftarrow{\gamma}}"'] & H_*\overleftarrow{H}\overleftarrow{\gamma}
\end{tikzcd}$$

which is a coequalizer in $\Uparrow\mathbb{C}$. Note that the $\overleftarrow{H}$-algebra a displays the presheaf $\overleftarrow{\gamma}$ as the coequalizer of the H^*-image of the pair $\langle H_*a, \eta_{H_*\overleftarrow{\gamma}}\rangle$. The $\overleftarrow{H}$-algebra a itself is the coequalizer of the free $\overleftarrow{H}$-algebras over this image, *and* as the limit inferior as decomposed in Proposition 2.5. We set the G'-image of the $\overleftarrow{H}$-algebra a to be the limit inferior of the functor $\mathbb{F} \xrightarrow{\varphi} \mathbb{C} \xrightarrow{G} \mathbb{X}$. Equation (17) follows from Lem. 2.4. The fact that G' preserves inferior limits follows from the fact that every inferior limit cone $H^*\overrightarrow{F} \xrightarrow{\lambda} \overleftarrow{\gamma}$ factors through any structure map $\overleftarrow{H}\overleftarrow{\gamma} \xrightarrow{a} \overleftarrow{\gamma}$: the factorization is the composite $H^*\overrightarrow{F} \xrightarrow{\lambda} \overleftarrow{\gamma} \xrightarrow{\eta} H^*\overleftarrow{\gamma}$, which obviously boils down to λ when further postcomposed with a. The uniqueness follows from Proposition 2.5. $\blacktriangleleft$

Proof of Lem. 2.10. Consider a natural transformation $\psi \in (\Downarrow\mathbb{A})^{\overleftarrow{\Phi}}\left(\Phi_*\Phi^\# a, \Phi_*\overrightarrow{\beta}\right)$ on the left-hand side of (10). By (5) and by the naturality of ψ, for $f \in \mathbb{A}(x,a)$ the left-hand square diagram in Fig. 7 must commute.

$$\begin{tikzcd}
\Phi_*\Phi^\# a(a) \arrow[r, phantom, "="] \arrow[d, "\Phi_*\Phi^\# a(f)"] & {\Uparrow\mathbb{B}\left(\Phi^\# a, \Phi^\# a\right)} \arrow[r, "\psi_a"] \arrow[d, "(-)\circ f"'] & {\Uparrow\mathbb{B}\left(\Phi^\# a, \overrightarrow{\beta}\right)} \arrow[r, phantom, "="] \arrow[d, "(-)\circ f"] & \Phi_*\overrightarrow{\beta}(a) \arrow[d, "\Phi_*\overrightarrow{\beta}(f)"'] \\
\Phi_*\Phi^\# a(x) \arrow[r, phantom, "="] & {\Uparrow\mathbb{B}\left(\Phi^\# x, \Phi^\# a\right)} \arrow[r, "\psi_x"] & {\Uparrow\mathbb{B}\left(\Phi^\# x, \overrightarrow{\beta}\right)} \arrow[r, phantom, "="] & \Phi_*\overrightarrow{\beta}(x)
\end{tikzcd}
\qquad
\begin{tikzcd}
\left[\nabla_\Phi a\right](\nabla_\Phi a) \arrow[r, "{[\psi]}"] \arrow[d, "\left[\nabla_\Phi a\right]\widehat{f}"] & \left[\Phi_*\overrightarrow{\beta}\right](\nabla_\Phi a) \arrow[d, "\left[\Phi_*\overrightarrow{\beta}\right]\widehat{f}"'] \\
\left[\nabla_\Phi a\right](\nabla_\Phi x) \arrow[r, "{[\psi]}"'] & \left[\Phi_*\overrightarrow{\beta}\right](\nabla_\Phi x)
\end{tikzcd}$$

Figure 7 Matrix Yoneda squares

Recall that $\overleftarrow{\Phi}$-algebras like $\nabla_\Phi a, \Phi_*\overrightarrow{\beta}\colon \mathbb{A}^o \to \mathsf{Set}$ always canonically extend to functors $\left[\nabla_\Phi a\right] \to \left[\Phi_*\overrightarrow{\beta}\right] : (\Downarrow\mathbb{A})^o_{\overleftarrow{\Phi}} \to \mathsf{Set}$, and that $\overleftarrow{\Phi}$-algebra homomorphism $\psi\colon \nabla_\Phi a \to \Phi_*\overrightarrow{\beta}$ extend to $[\psi]\colon \left[\nabla_\Phi a\right] \to \left[\Phi_*\overrightarrow{\beta}\right]$. It follows that a $\overleftarrow{\Phi}$-algebra homomorphism ψ must be natural with respect to all homomorphisms between free $\overleftarrow{\Phi}$-algebras, and not just with respect to those arising from $\mathbb{A}$.

In particular, consider the natural isomorphism

$$\Uparrow\mathbb{B}\left(\Phi^{\#}x, \Phi^{\#}a\right) \overset{(a)}{=} \Phi_*\Phi^{\#}a(x) \overset{(b)}{\cong} \Downarrow\mathbb{A}\left(\nabla x, \Phi_*\Phi^{\#}a\right) \overset{(c)}{=} (\Downarrow\mathbb{A})^{o}_{\overleftarrow{\Phi}}\left(\nabla_\Phi a, \nabla_\Phi x\right) \tag{18}$$

where (a) is based on (5), (b) on the usual Yoneda lemma, and (c) on the definition of the Kleisli category $(\Downarrow\mathbb{A})_{\overleftarrow{\Phi}}$. Every $f \in \Uparrow\mathbb{B}\left(\Phi^{\#}x, \Phi^{\#}a\right)$ thus induces a unique homomorphism $\widehat{f} \in (\Downarrow\mathbb{A})^{o}_{\overleftarrow{\Phi}}\left(\nabla_\Phi a, \nabla_\Phi x\right)$, and vice versa. The naturality condition on $\left[\psi\right]$ now implies that the right-hand square on Fig. 7 must commute, which implies

$$\left[\psi\right]_{\nabla_\Phi x}\left(\widehat{f}\right) = \left[\psi\right]_{\nabla_\Phi x}\circ\left[\nabla_\Phi a\right]\widehat{f}\left(\mathrm{id}_{\nabla_\Phi a}\right) = \left[\Phi_*\overrightarrow{\beta}\right]\widehat{f}\circ\left[\psi\right]_{\nabla_\Phi a}\left(\mathrm{id}_{\nabla_\Phi a}\right) = \left[\Phi_*\overrightarrow{\beta}\right]\widehat{f}\left(\left[\Psi\right]\right) \tag{19}$$

where $\left[\Psi\right] = \left[\psi\right]_{\nabla_\Phi a}\left(\mathrm{id}_{\nabla_\Phi a}\right)$. Hence the bijection between the natural transformations $\left[\psi\right] : \left[\nabla_\Phi a\right] \to \left[\Phi_*\overrightarrow{\beta}\right]$ and the elements $\left[\Psi\right]$ of $\left[\Phi_*\overrightarrow{\beta}\right]\left(\nabla_\Phi a\right)$. The restriction to $\psi\colon \nabla_\Phi a \to \Phi_*\overrightarrow{\beta}$ of $\left[\psi\right]$ must be coherent with respect to the natural bijection (18), which means that ψ must be natural with respect to $f \in \Uparrow\mathbb{B}\left(\Phi^{\#}x, \Phi^{\#}a\right)$ just like $\left[\psi\right]$ was with respect to $\widehat{f} \in (\Downarrow\mathbb{A})^{o}_{\overleftarrow{\Phi}}\left(\nabla_\Phi a, \nabla_\Phi x\right)$. The naturality of the left-hand square in Fig. 7 now gives

$$\psi_x\left(f\right) = \psi_x \circ \nabla_\Phi a f\left(\mathrm{id}_{\Phi^{\#} a}\right) = \Phi_*\overrightarrow{\beta}\left(f\right)\circ\psi_a\left(\mathrm{id}_{\Phi^{\#} a}\right) = \Phi_*\overrightarrow{\beta}\left(f\right)\Psi \tag{20}$$

where $\Psi = \psi_a\left(\mathrm{id}_{\Phi^{\#} a}\right)$. Hence the bijection between the $\overleftarrow{\Phi}$-algebra homomorphisms $\psi \in (\Downarrow\mathbb{A})^{\overleftarrow{\Phi}}\left(\nabla_\Phi a, \Phi_*\overrightarrow{\beta}\right)$ and the elements Ψ of $\Phi_*\overrightarrow{\beta}\left(a\right)$, as claimed in (10). Claim (11) is proven dually. ◀

Proof of Lem. 3.3.

1. Let $A \xleftarrow{h} UFA$ be a T-algebra. By the unit law of Eilenberg-Moore algebras, we have a retract $A \overset{\eta_A}{\rightarrowtail} UFA \overset{h}{\twoheadrightarrow} A$. In particular, the underlying object A of the algebra is a retract of an image under U.
2. Firstly, we shall show that

$$\mathcal{D} = \left\{B \in \mathcal{B} \;\middle|\; B \xleftarrow{\varepsilon_B} FUB \underset{\varepsilon_{FUB}}{\overset{FU\varepsilon_B}{\leftleftarrows}} FUFUB \text{ is a coequalizer}\right\} \tag{21}$$

as a full subcategory of $\mathcal{B}$. Recall that $KB = (UB \xleftarrow{U\varepsilon_B} UFUB)$. For a T-algebra $A \xleftarrow{h} UFA$, its image under $L\colon \mathcal{A}^T \rightharpoonup \mathcal{B}$ is defined by the representability:

$$\begin{aligned}\mathcal{B}\big(L(A \xleftarrow{h} UFA), B\big) &\cong \mathcal{A}^T\big((A \xleftarrow{h} UFA), (UB \xleftarrow{U\varepsilon_B} UFUB)\big)\\ &\cong \left\{f \in \mathcal{B}(FA, B) \;\middle|\; B \xleftarrow{f} FA \underset{\varepsilon_{FA}}{\overset{Fh}{\leftleftarrows}} FUFA \text{ commutes}\right\},\end{aligned}$$

where the latter isomorphism is essentially shown at the item (a) of the precise monadicity theorem. In particular, a counit $LKB \to B$ of the partial adjunction $L \dashv K$ (exists and) is an isomorphism if and only if the diagram in (21) is a equalizer in $\mathcal{B}$, because the above bijection maps $\mathrm{id}_{KB} \in \mathcal{A}^T(KB, KB)$ to $\varepsilon_B \in \mathcal{B}(FUB, B)$.

Secondly, we claim that $\operatorname{Im} F \subseteq \mathcal{D}$ as full subcategories of $\mathcal{B}$. It is obvious as the following is a split coequalizer diagram:

$$\begin{tikzcd}
FA & FUFA \arrow[l, "\varepsilon_{FA}"'] \arrow[r, bend right, "FUF\eta_A"'] & FUFUFA \arrow[l, shift right, "FU\varepsilon_{FA}"'] \arrow[l, shift left, "\varepsilon_{FUFA}"] \\
\arrow[from=1-1, to=1-2, bend right, "F\eta_A"']
\end{tikzcd}$$

.

Finally, we shall prove that the subcategory $\mathcal{D} \subseteq \mathcal{B}$ is closed under taking retracts. Let D be an object of $\mathcal{D}$, and $B \overset{s}{\rightarrowtail} D \overset{r}{\twoheadrightarrow} B$ be a retract of D in $\mathcal{B}$. We have shown that the upper row of the following diagram is a coequalizer in $\mathcal{B}$.

$$\begin{tikzcd}
D \arrow[d, shift left, "r"] & FUD \arrow[l, "\varepsilon_D"'] \arrow[d, shift left, "FUr"] & FUFUD \arrow[l, shift right, "FU\varepsilon_D"'] \arrow[l, shift left, "\varepsilon_{FUD}"] \arrow[d, shift left, "FUFUr"] \\
B \arrow[u, shift left, "s"] & FUB \arrow[l, "\varepsilon_B"'] \arrow[u, shift left, "FUs"] & FUFUB \arrow[l, shift right, "FU\varepsilon_B"'] \arrow[l, shift left, "\varepsilon_{FUB}"] \arrow[u, shift left, "FUFUs"]
\end{tikzcd}$$

The squares commutes serially, and all the columns are retracts. It is a straightforward consequence that the lower row of the diagram is also a coequalizer. ◀

▶ **Lemma A.1.** *Let X be a G-set and J be a set. Any G-map $f\colon X \to G \times J$ to the free G-set generated by J is a composite $X \cong G \times I \xrightarrow{\mathrm{id}_G \times k} G \times J$ for some $k\colon I \to J$ in* Set.

Proof of Lem. A.1. Let $I = f^{-1}(\{e\} \times J)$. The action of X induces an isomorphism $X \cong G \times I$ of G-sets. ◀

▶ **Lemma A.2.** *A retract of a singleton in G-Set is a singleton. A retract of a free G-set is free.*

Proof of Lem. A.2. The first claim is obvious. The latter claim is by Lem. A.1. ◀

Proof of Prop. 3.7. Let X be a right G-set. If there exists a right G-map $X \to G$, there exists an isomorphism $X \cong I \times G$ for some set I by Lem. A.1. Then, we have a bijection

$$H^*X = G^o\text{-}\mathsf{Set}(X, G) \cong G^o\text{-}\mathsf{Set}(I \times G, G) \cong \mathsf{Set}(I, G) = G^I \quad ,$$

which is moreover an isomorphism of left G-sets. If X does not have a right G-map $X \to G$, then we have $H^*X = 0$. For instance, letting $X = 1$ gives $H^*1 = 0$ since $|G| \geq 2$. ◀

Proof of Prop. 3.8. By the Monadicity Theorem, this proposition reduces to the following two lemmas. ◀

▶ **Lemma A.3.** *The following hold.*

1. *The category G-$\mathsf{Set}_{1,\mathrm{free}}$ has reflexive equalizers.*
2. *The functor $H_*\colon (G\text{-}\mathsf{Set}_{1,\mathrm{free}})^o \to G^o\text{-}\mathsf{Set}_{1,\mathrm{free}}$ preserves reflexive coequalizers.*

Proof. By Lem. A.2, a reflexive pair in G-$\mathsf{Set}_{1,\mathrm{free}}$ is either $1 \rightrightarrows 1$ or $G \times I \rightrightarrows G \times J$. The pair $1 \rightrightarrows 1$ trivially has an equalizer that is preserved by any functor.

Let $r\colon G{\times}J \to G{\times}I$ be a common retraction in G-$\mathsf{Set}_{1,\mathrm{free}}$ of the pair $(f, h)\colon G{\times}I \rightrightarrows G{\times}J$. We may assume $r = \mathrm{id}_G \times r'$ for some map $r'\colon J \to I$ by Lem. A.1. Define a map $f'\colon I \to J$ by $\langle g_i, f'(i)\rangle = f(\langle e, i\rangle)$ for each $i \in I$ where it turns out to hold $g_i = e$ since

$$\langle e, i\rangle = r\big(f(\langle e, i\rangle)\big) = r\big(\langle g_i, f'(i)\rangle\big) = \big\langle g_i, r'\big(f'(i)\big)\big\rangle \ .$$

Moreover for any $g \in G$, it holds

$$f(\langle g, i\rangle) = f(g \cdot \langle e, i\rangle) = g \cdot f(\langle e, i\rangle) = g \cdot \langle e, f'(i)\rangle = \langle g, f'(i)\rangle \ .$$

Therefore, there exist maps $f', h'\colon I \rightrightarrows J$ such that $f = \mathrm{id}_G \times f'$, $h = \mathrm{id}_G \times h'$, and r' is a common retraction of the pair (f', h') in Set.

Using an equalizer $E \to I \rightrightarrows J$ in Set, we have an equalizer $G \times E \to G \times I \rightrightarrows G \times J$ in G-$\mathsf{Set}_{1,\mathrm{free}}$. We shall show that this (co)equalizer is preserved by $H_* \colon (G\text{-}\mathsf{Set}_{1,\mathrm{free}})^o \to G^o\text{-}\mathsf{Set}_{1,\mathrm{free}}$, i.e. the diagram

$$G^E \leftarrow G^I \leftleftarrows G^J$$

is a coequalizer of right G-sets. Their underlying sets form a coequalizer diagram in Set because the functor $|G|^{(-)} \colon \mathsf{Set}^o \to \mathsf{Set}$ preserves reflexive coequalizers for $|G| \geq 2$ by Lem. B.2. Hence, the diagram is also a coequalizer in G^o-Set. ◀

▶ **Lemma A.4.** *The functor* $H_* \colon (G\text{-}\mathsf{Set}_{1,\mathrm{free}})^o \to G^o\text{-}\mathsf{Set}_{1,\mathrm{free}}$ *reflects isomorphisms.*

Proof. Let $f \colon X \to Y$ be a morphism in G-$\mathsf{Set}_{1,\mathrm{free}}$ such that $H_* f \colon H_* Y \to H_* X$ is an isomorphism. There are three cases of the G-map f:

$$1 \to 1 \ , \qquad\qquad G \times I \to 1 \ , \qquad\qquad G \times I \to G \times J \ .$$

The first two cases are trivial. For the last case, we may assume that $f = \mathrm{id}_G \times k$ for some map $k \colon I \to J$ by Lem. A.1. Then, the right G-bijection $H_* f \colon H_*(G \times J) \to H_*(G \times I)$ can be written as $G^k \colon G^J \to G^I$. By $|G| \geq 2$, the map k is a bijection, which shows that the G-map $f = \mathrm{id}_G \times k$ is an isomorphism. ◀

Proof of Lem. 3.10. The terminal object in $\Downarrow P$ is a constant presheaf 1. A presheaf $\overleftarrow{\alpha} \in \Downarrow P$ is a subobject of the presheaf 1 if and only if $\overleftarrow{\alpha} \rightarrowtail 1$ in Set for any $x \in P$.

A representable presheaf ∇x is a subobject of 1, since $(\nabla x)(y) = P(y, x) \rightarrowtail 1$. ◀

Proof of Lem. 3.11. Firstly, we shall show that $\operatorname{Im} H^* \subseteq \uparrow P$. Let $\overleftarrow{\alpha} \in \Downarrow P$ be a presheaf. We shall show $H^* \overleftarrow{\alpha} \in \uparrow P$. The set $(H^* \overleftarrow{\alpha})(x) = \Downarrow P(\overleftarrow{\alpha}, \nabla x)$ has at most one element for any $x \in P$, since ∇x is a subobject of a terminal object $1 \in \Downarrow P$. In particular, the postsheaf $H^* \overleftarrow{\alpha} \in \Uparrow P$ is an upper set of P.

Dually, we have $\operatorname{Im} H_* \subseteq \downarrow P$. Then, the adjunction $H^* \dashv H_*$ restricts as

$$\begin{tikzcd}
\downarrow P \arrow[r, tail] \arrow[d, bend right, "H^*|_{\downarrow P}"'] \arrow[d, phantom, "\dashv"] & \Downarrow P \arrow[d, bend right, "H^*"'] \arrow[d, phantom, "\dashv"] \\
\uparrow P \arrow[r, tail] \arrow[u, bend right, "H_*|_{\uparrow P}"'] & \Uparrow P \arrow[u, bend right, "H_*"'] \ .
\end{tikzcd}$$

It is obvious by definition that the restricted adjunction coincides with the $\{0, 1\}$-enriched construction.

The claim $\operatorname{Im} H^* = \operatorname{Im}(H^*|_{\downarrow P})$ follows from that the embedding $\downarrow P \rightarrowtail \Downarrow P$ has a left adjoint $\Downarrow P \to \downarrow P$, which maps a presheaf $\overleftarrow{\alpha}$ to the image of $\overleftarrow{\alpha} \to 1$. Just for reference, we describe the following elementary proof, which boils down to the above argument.

A presheaf $\overleftarrow{\alpha}$ can be written as a canonical colimit $\overleftarrow{\alpha} = \varinjlim_{i \in \mathbb{I}} \nabla a_i$. Let $L \subseteq P$ be the image of $\overleftarrow{\alpha} \to 1$, i.e.

$$L = \left\{ x \in P \;\middle|\; \overleftarrow{\alpha}(x) \text{ is nonempty} \right\} = \bigcup_{i \in |\mathbb{I}|} \{ x \in P \mid x \leq a_i \} = \bigcup_{i \in |\mathbb{I}|} \nabla a_i \ .$$

We shall show that this lower set $L \in \downarrow P$ satisfies $H^* L = H^* \overleftarrow{\alpha}$. We have

$$\begin{aligned}
H^* L &= H^* \bigcup_{i \in |\mathbb{I}|} \nabla a_i = \bigcap_{i \in |\mathbb{I}|} H^* \nabla a_i = \bigcap_{i \in |\mathbb{I}|} \Delta a_i && \in \uparrow P \ , \\
H^* \overleftarrow{\alpha} &= H^* \varinjlim_{i \in \mathbb{I}} \nabla a_i = \varinjlim_{i \in \mathbb{I}} H^* \nabla a_i = \varinjlim_{i \in \mathbb{I}} \Delta a_i && \in \Uparrow P
\end{aligned}$$

by the adjunctions $H^*|_{\downarrow P} \dashv H_*|_{\uparrow P}$ and $H^* \dashv H_*$, respectively. The colimit $\varinjlim_{i\in\mathbb{I}} \Delta a_i$ in $\Uparrow P$ is a limit in Set^P, and it is just a product in Set^P because the objects Δa_i are subobjects of 1 in Set^P. Therefore, $H^* L = H^* \overleftarrow{\alpha}$. ◀

Proof of Cor. 3.12. A retract of an upper set $U \subseteq P$ is also an upper set, because the retract is always U itself. Thus,

$$\begin{aligned} \mathrm{Retr}_{\Uparrow P}(\mathrm{Im}\, H^*) &= \mathrm{Retr}_{\Uparrow P}(\mathrm{Im}(H^*|_{\downarrow P})) && \text{by Lem. 3.11} \\ &= \mathrm{Im}(H^*|_{\downarrow P}) && \text{by } \mathrm{Im}(H^*|_{\downarrow P}) \subseteq {\uparrow} P. \end{aligned}$$ ◀

Proof of Prop. 3.13. By Lem. 3.11 and Cor. 3.12. ◀

Proof of Lem. 3.15. For a $\mathbb{Z}_p$-set $Y = \underline{1}Y_1 + \underline{p}Y_p$, an element $y \in Y$ forms a $\mathbb{Z}_p$-map $y\colon \underline{1} \to Y$ if and only if $y \in \underline{1}Y_1$. For the free $\mathbb{Z}_p$-set $\underline{p}$, an element $y \in Y$ bijectively corresponds to a $\mathbb{Z}_p$-map $\underline{p} \to Y$. Since an orbit-decomposition is a coproduct,

$$\begin{aligned} \mathbb{Z}_p\text{-}\mathsf{Set}(\underline{1}X_1 + \underline{p}X_p, \underline{1}Y_1 + \underline{p}Y_p) &\cong \big(\mathbb{Z}_p\text{-}\mathsf{Set}(\underline{1}, \underline{1}Y_1 + \underline{p}Y_p)\big)^{X_1} \big(\mathbb{Z}_p\text{-}\mathsf{Set}(\underline{p}, \underline{1}Y_1 + \underline{p}Y_p)\big)^{X_p} \\ &\cong Y_1^{X_1} (Y_1 + pY_p)^{X_p} \ . \end{aligned}$$ ◀

Proof of Lem. 3.16. Let r be a common retraction of the pair (f, g), and $I = \mathrm{Im}\, f \cup \mathrm{Im}\, g \cong \underline{1}I_1 + \underline{p}I_p$. By the existence of retraction, we have $f = f'_1 + f'_p\colon \underline{1}U_1 + \underline{p}U_p \to \underline{1}U'_1 + \underline{p}U'_p$ for some $\mathbb{Z}_p$-maps f'_1, f'_p, and similar for g. Hence, there exists $r|_I = r'_1 + r'_p\colon \underline{1}I_1 + \underline{p}I_p \to \underline{1}U_1 + \underline{p}U_p$. We may assume $r'_1 + r'_p = \underline{1}r_1 + \underline{p}r_p$ by modifying the coercing isomorphism $U' \cong \underline{1}U'_1 + \underline{p}U'_p$ on $I \subseteq U'$. Under the assumption, we obtain $f'_1 + f'_p = \underline{1}f_1 + \underline{p}f_p$ and $g'_1 + g'_p = \underline{1}g_1 + \underline{p}g_p$.

The reflexive equalizer in $\mathbb{Z}_p$-Set is also a reflexive equalizer in Set, which induces the following pullback of injections in Set by Lem. B.2.1.

$$\begin{tikzcd} E_1 + pE_p \arrow[r, tail, "e_1 + pe_p"] \arrow[d, tail, "e_1 + pe_p"'] & U_1 + pU_p \arrow[d, tail, "f_1 + pf_p"] \\ U_1 + pU_p \arrow[r, tail, "g_1 + pg_p"'] & U'_1 + pU'_p \ . \end{tikzcd}$$

Changing the base by maps $U'_1 \to U'_1 + pU'_p$, $U'_p \to U'_1 + pU'_p$ concludes the proof. ◀

Proof of Prop. 3.17. It is easy to check the full subcategories $\mathcal{C}, \mathcal{D}$ contain all retracts of images. Then, by Lem. 3.3.1, we have only to show that the restrictions $\Phi_*|_{\mathcal{D}}\colon \mathcal{D} \to \mathcal{C}$, $\Phi^*|_{\mathcal{C}}\colon \mathcal{C} \to \mathcal{D}$ are monadic functors. By the Monadicity Theorem, it follows from the following two lemmas that the functor $\Phi_*|_{\mathcal{D}}$ is monadic. The comonadicity of $\Phi^*|_{\mathcal{C}}$ is shown similarly to the comonadicity of the restriction of $R^*\colon \mathsf{Set} \to \mathsf{Set}^o$ (Prop. 3.4) for $R = \Phi_1 + p\Phi_p$. ◀

▶ **Lemma A.5.** *For the adjunction $\Phi^*|_{\mathcal{C}} \dashv \Phi_*|_{\mathcal{D}}\colon \mathcal{D} \to \mathcal{C}$ in Fig. 5, the category $\mathcal{D}$ has and the functor $\Phi_*|_{\mathcal{D}}$ preserves reflexive coequalizers.*

Proof. By Lem. 3.16, an equalizer in $\mathbb{Z}_p$-Set of a reflexive pair in $\mathcal{D}^o \subseteq \mathbb{Z}_p$-$\mathsf{Set}$ can be taken as

$$\begin{tikzcd} E \arrow[r, "e"] \arrow[d, equal] & U \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] \arrow[d, equal] & U' \arrow[d, equal] \\ \underline{1}E_1 + \underline{p}E_p \arrow[r, "\underline{1}e_1 + \underline{p}e_p"] & \underline{1}U_1 + \underline{p}U_p \arrow[r, shift left, "\underline{1}f_1 + \underline{p}f_p"] \arrow[r, shift right, "\underline{1}g_1 + \underline{p}g_p"'] & \underline{1}U'_1 + \underline{p}U'_p \end{tikzcd}$$

for some equalizers

$$\begin{tikzcd} E_1 \arrow[r, "e_1"] & U_1 \arrow[r, shift left, "f_1"] \arrow[r, shift right, "g_1"'] & U_1' \end{tikzcd} , \qquad \begin{tikzcd} E_p \arrow[r, "e_p"] & U_p \arrow[r, shift left, "f_p"] \arrow[r, shift right, "g_p"'] & U_p' \end{tikzcd} .$$

It is easy to show that $\underline{1}E_1 + \underline{p}E_p \in \mathcal{D}$. For example, if $\Phi = \underline{1} + \underline{p}\Phi_p$ then $U_1 = U_1' = 1$, which implies $E_1 = 1$.

By Lem. B.2.2, the diagrams

$$\begin{tikzcd} \Phi_1^{E_1} & \Phi_1^{U_1} \arrow[l, "\Phi_1^{e_1}"'] & \Phi_1^{U_1'} \arrow[l, shift right, "\Phi_1^{f_1}"'] \arrow[l, shift left, "\Phi_1^{g_1}"] \end{tikzcd} , \quad \begin{tikzcd} (\Phi_1 + p\Phi_p)^{E_p} & (\Phi_1 + p\Phi_p)^{U_p} \arrow[l, "(\Phi_1+p\Phi_p)^{e_p}"'] & (\Phi_1 + p\Phi_p)^{U_p'} \arrow[l, shift right, "(\Phi_1+p\Phi_p)^{f_p}"'] \arrow[l, shift left, "(\Phi_1+p\Phi_p)^{g_p}"] \end{tikzcd}$$

are split coequalizers. Hence, their pointwise product

$$\begin{tikzcd} \Phi_*(\underline{1}E_1 + \underline{p}E_p) & \Phi_*(\underline{1}U_1 + \underline{p}U_p) \arrow[l, "\Phi_*(\underline{1}e_1+\underline{p}e_p)"'] & \Phi_*(\underline{1}U_1' + \underline{p}U_p') \arrow[l, shift right, "\Phi_*(\underline{1}f_1+\underline{p}f_p)"'] \arrow[l, shift left, "\Phi_*(\underline{1}g_1+\underline{p}g_p)"] \end{tikzcd}$$

is a (split) coequalizer. ◀

▶ **Lemma A.6.** *The right adjoint functor $\Phi_*|_\mathcal{D} \colon \mathcal{D} \to \mathcal{C}$ in Fig. 5 reflects isomorphisms.*

Proof. Let $f \colon U \to U'$ be a $\mathbb{Z}_p$-map. In general, f is of the form

$$\underline{1}U_1 + \underline{p}(U_{p,1} + U_{p,p}) \ \to \ \underline{1}U_1' + \underline{p}U_p'$$

induced by maps $f_1 \colon U_1 \to U_1'$, $g \colon U_{p,1} \to U_1'$, $h \colon U_{p,p} \to pU_p'$ up to isomorphisms, by Lem. 3.15. Modify the embedding $\underline{p}U_{p,p} \rightarrowtail U$, and we may assume further that the map h is induced by a map $f_p \colon U_{p,p} \to U_p'$. Then, $f = [\mathrm{id}_{\underline{1}} \times f_1, ! \times g, \mathrm{id}_{\underline{p}} \times f_p]$ where $! \colon \underline{p} \to \underline{1}$ is a unique $\mathbb{Z}_p$-map.

Assume that $U, U' \in \mathcal{D}$ and that $\Phi_* f \colon \Phi_* U' \to \Phi_* U$ is a bijection. We have only to prove the bijectivity of the maps f_1, f_p and $U_{p,1} = 0$. The map $\Phi_* f$ factors through an injection as the following diagram:

$$\begin{tikzcd} & \Phi_1^{U_1}\Phi_1^{U_{p,1}}(\Phi_1 + p\Phi_p)^{U_{p,p}} \arrow[d, tail] \\ \Phi_1^{U_1'}(\Phi_1 + p\Phi_p)^{U_p'} \arrow[ur, "\langle \Phi_1^{f_1}, \Phi_1^{g} \rangle \times (\Phi_1 + p\Phi_p)^{f_p}"] \arrow[r, "\Phi_* f"'] & \Phi_1^{U_1}(\Phi_1 + p\Phi_p)^{U_{p,1}}(\Phi_1 + p\Phi_p)^{U_{p,p}} \end{tikzcd} .$$

Since the injection must be a bijection, we have

$$\Phi_1^{U_1}(\Phi_1 + p\Phi_p)^{U_{p,p}} = 0 \quad \text{or} \quad \Phi_p = 0 \quad \text{or} \quad U_{p,1} = 0 \ .$$

The rest is straightforward for each $\Phi \colon 1 \nrightarrow \mathbb{Z}_p$.

For example, let $\Phi = \underline{p}\Phi_p$. The full subcategory $\mathcal{D} \subseteq (\mathbb{Z}_p\text{-}\mathsf{Set})^o$ contains only $\mathbb{Z}_p$-sets with trivial actions, i.e. $\mathcal{D} \subseteq \mathsf{Set}^o \subseteq (\mathbb{Z}_p\text{-}\mathsf{Set})^o$. In particular, we have $U_{p,1} = U_{p,p} = 0$, and the claim reduces to the monadicity of a restriction of the functor $\Phi_1^{(-)} \colon \mathsf{Set}^o \to \mathsf{Set}$. ◀

B Appendix: General propositions

▶ **Proposition B.1** (Precise monadicity theorem). *Let $U\colon \mathcal{D} \to \mathcal{C}$ be a functor that has a left adjoint $F\colon \mathcal{C} \to \mathcal{D}$, and $T = U \circ F$ be the induced monad.*

$$\begin{tikzcd}
\mathcal{C} \arrow[loop above, "T"] \arrow[r, bend right, "F^T"'] \arrow[r, phantom, "\top"] \arrow[d, bend right, "F"'] \arrow[d, phantom, "\dashv"] & \mathcal{C}^T \arrow[l, bend right, "U^T"'] \arrow[ld, dashed, bend left, "L"] \\
\mathcal{D} \arrow[u, bend right, "U"'] \arrow[ru, "K"] \arrow[ru, phantom, "\top"] &
\end{tikzcd}$$

$$KD = (UD \xleftarrow{U\varepsilon_D} UFUD) \ ,$$

$$L(C \xleftarrow{h} UFC) \leftarrow FC \underset{\varepsilon_{FC}}{\overset{Fh}{\leftleftarrows}} FUFC \quad \textit{is a coequalizer.}$$

(a) *The comparison functor $K\colon \mathcal{D} \to \mathcal{C}^T$ has a left adjoint $L\colon \mathcal{C}^T \to \mathcal{D}$ if the category $\mathcal{D}$ has reflexive U-split coequalizers.*

(b) *The functor L is full and faithful if $\mathcal{D}$ has and U preserves reflexive U-split coequalizers.*

(c) *The comparison functor K is full and faithful if U reflects isomorphisms [3, Sec. 3.3].*

In particular, the right adjoint functor U is monadic if U creates reflexive U-split coequalizers.

Conversely, for a monad T, the forgetful functor $U^T : \mathcal{C}^T \to \mathcal{C}$ creates U^T-split coequalizers.

The statements and the proof of the following lemma is inspired by the proof of the monadicity of powerset functors $\Omega^{(-)}$ (see e.g. [3, Sec. 5.1]).

▶ **Lemma B.2.** *Let X, Y be sets, and $f, g\colon X \rightrightarrows Y$ be maps.*

1. *If the pair (f, g) is reflexive, then the maps f, g are injections and the diagram*

$$\begin{tikzcd}
Z \arrow[r, "e"] \arrow[d, "e"'] & X \arrow[d, "f"] \\
X \arrow[r, "g"'] & Y
\end{tikzcd} \tag{22}$$

is a pullback for an equalizer $Z \xrightarrow{e} X$ of the pair (f, g).

2. *Let R be a nonempty set and $e\colon Z \to X$ be a map such that the above diagram (22) is a pullback. If the map f is an injection, then $R^Z \xleftarrow{R^e} R^X \underset{R^g}{\overset{R^f}{\leftleftarrows}} R^Y$ is a split coequalizer. In particular, the functor $R^{(-)}\colon \mathsf{Set}^o \to \mathsf{Set}$ preserves such coequalizers.*

Proof of Lem. B.2.

1. Obvious.
2. Firstly, the map e is an injection, because f is. Fix an element $r \in R$. Since the maps e, f are injective, we may and shall define maps $R^Z \xrightarrow{e_r} R^X \xrightarrow{f_r} R^Y$ by

$$e_r(k)(x) = \begin{cases} k(z) & \text{if } x = e(z), \\ r & \text{otherwise} \end{cases} \qquad\qquad f_r(h)(y) = \begin{cases} h(x) & \text{if } y = f(x), \\ r & \text{otherwise} \end{cases}$$

where $h\colon X \to R$ and $k\colon Z \to R$. The maps give a splitting

$$\begin{tikzcd}
R^Z \arrow[r, bend right, "e_r"'] & R^X \arrow[l, "R^e"'] \arrow[r, bend right, "f_r"'] & R^Y \arrow[l, shift right, "R^f"'] \arrow[l, shift left, "R^g"]
\end{tikzcd} \ ,$$

i.e. the diagrams

$$\begin{tikzcd}
& R^Z \\
R^Z \arrow[ur, "\mathrm{id}"] \arrow[r, "e_r"'] & R^X \arrow[u, "R^e"']
\end{tikzcd}
\qquad
\begin{tikzcd}
& R^X \\
R^X \arrow[ur, "\mathrm{id}"] \arrow[r, "f_r"'] & R^Y \arrow[u, "R^f"']
\end{tikzcd}
\qquad
\begin{tikzcd}
R^Z \arrow[r, "e_r"] & R^X \\
R^X \arrow[u, "R^e"] \arrow[r, "f_r"'] & R^Y \arrow[u, "R^f"']
\end{tikzcd}$$

commute. ◀

Codensity Liftings of Monads

Shin-ya Katsumata and Tetsuya Sato

Research Institute for Mathematical Sciences, Kyoto University
Kitashirakawaoiwakecho, Sakyoku, Kyoto, 606-8502, Japan
{sinya,satoutet}@kurims.kyoto-u.ac.jp

Abstract

We introduce a method to lift monads on the base category of a fibration to its total category using *codensity monads*. This method, called *codensity lifting*, is applicable to various fibrations which were not supported by the categorical $\top\top$-lifting. After introducing the codensity lifting, we illustrate some examples of codensity liftings of monads along the fibrations from the category of preorders, topological spaces and extended psuedometric spaces to the category of sets, and also the fibration from the category of binary relations between measurable spaces. We next study the liftings of *algebraic operations* to the codensity-lifted monads. We also give a characterisation of the class of liftings (along posetal fibrations with fibred small limits) as a limit of a certain large diagram.

1998 ACM Subject Classification F.3.2. Semantics of Programming Languages

Keywords and phrases Monads, Lifting, Fibration, Giry Monad

Digital Object Identifier 10.4230/LIPIcs.CALCO.2015.156

1 Introduction

Inspired by Lindley and Stark's work on extending the concept of reducibility candidates to monadic types [9, 10], the first author previously introduced its semantic analogue called *categorical* $\top\top$*-lifting* in [6]. It constructs a lifting of a strong monad $\mathcal{T}$ on the base category of a closed-structure preserving fibration $p : \mathbb{E} \to \mathbb{B}$ to its total category. The construction takes the inverse image of the continuation monad on the total category along the canonical monad morphism $b : \mathcal{T} \to (- \Rightarrow TR) \Rightarrow TR$ in the base category, which exists for any strong monad $\mathcal{T}$:

$$\begin{tikzcd}
\mathcal{T}^{\top\top} \arrow[r, dotted] & (- \Rightarrow S) \Rightarrow S \\
\mathcal{T} \arrow[r, "b"'] & (- \Rightarrow TR) \Rightarrow TR
\end{tikzcd}$$

The objects R and S (such that $TR = pS$) are presupposed parameters of this $\top\top$-lifting, and by varying them we can derive various liftings of $\mathcal{T}$. The categorical $\top\top$-lifting has been used to construct logical relations for monads [7] and to analyse the concept of preorders on monads [8].

One key assumption for the $\top\top$-lifting to work is that the fibration p preserves the *closed structure*, so that the continuation monad $(- \Rightarrow S) \Rightarrow S$ on the total category becomes a lifting of the continuation monad $(- \Rightarrow TR) \Rightarrow TR$ on the base category. Although many such fibrations are seen in the categorical formulations of logical relations [12, 3, 7], requiring fibrations to preserve closed structures on their total categories imposes a technical limitation to the applicability of the categorical $\top\top$-lifting. Indeed, outside the categorical semantics of type theories, it is common to work with the categories that have no closed structure. In the

6th International Conference on Algebra and Coalgebra in Computer Science (CALCO'15).
Editors: Lawrence S. Moss and Pawel Sobocinski; pp. 156–170
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

study of coalgebras, *predicate / relational liftings* of functors and monads are fundamental structures to formulate modal operators and (bi)simulation relations, and the underlying categories of them are not necessarily closed. For instance, the category **Meas** of measurable spaces, which is unlikely to be cartesian closed, is used to host labelled Markov processes. The categorical $\top\top$-lifting does not work in such situations.

To overcome this technical limitation, in this paper we introduce an alternative lifting method called *codensity lifting*. The idea is to replace the continuation monad $(- \Rightarrow S) \Rightarrow S$ with the *codensity monad* $\mathbf{Ran}_S S$ given by a right Kan extension. We then ask fibrations to preserve the right Kan extension, which is often fulfilled by the preservation of limits. We demonstrate that the codensity lifting is applicable to lift monads on the base categories of the following fibrations:

$$\begin{tikzcd}
\mathbf{Pre} \arrow[d] & \mathbf{Top} \arrow[d] & \mathbf{ERel}(\mathbf{Meas}) \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathbf{BRel}(\mathbf{Meas}) \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathbf{Pred} \arrow[d] & U^*\mathbf{EPMet} \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathbf{EPMet} \arrow[d] \\
\mathbf{Set} & \mathbf{Set} & \mathbf{Meas} \arrow[r, "\Delta"'] & \mathbf{Meas}^2 \arrow[r, "(\times)\circ U^2"'] & \mathbf{Set} & \mathbf{Meas} \arrow[r, "U"'] & \mathbf{Set}
\end{tikzcd}$$

Another issue when we have a lifting $\dot{\mathcal{T}}$ of a monad $\mathcal{T}$ is the liftability of *algebraic operations* for $\mathcal{T}$ to the lifting $\dot{\mathcal{T}}$. For instance, let $\dot{\mathcal{T}}$ be a lifting of the powerset monad $\mathcal{T}_p$ on **Set** along the canonical forgetful functor $p : \mathbf{Top} \to \mathbf{Set}$, which is a fibration. A typical algebraic operation for $\mathcal{T}_p$ is the union of A-indexed families of sets: $\mathrm{union}_X^A(f) = \bigcup_{a\in A} f(a)$. Then the question is whether we can "lift" the ordinary function $\mathrm{union}_X^A : A \pitchfork T_p X \to T_p X$ to a continuous function of type $A \pitchfork \dot{T}(X, \mathcal{O}_X) \to \dot{T}(X, \mathcal{O}_X)$ for every topological space $(X, \mathcal{O}_X)$. We show that the liftability of algebraic operations to codensity liftings has a good characterisation in terms of the parameters supplied to the codensity liftings.

We are also interested in the categorical property of the *collection* of liftings of a monad $\mathcal{T}$ (along a limited class of fibrations). We show a characterisation of the class of liftings of $\mathcal{T}$ as a limit of a large diagram of partial orders. This is yet an abstract categorical result, we believe that this will be helpful to construct and enumerate the possible liftings of a given monad $\mathcal{T}$.

1.1 Preliminaries

We use white bold letters $\mathbb{B}, \mathbb{C}, \mathbb{E}, \cdots$ to range over locally small categories. We sometimes identify an object in a category $\mathbb{C}$ and a functor of type $1 \to \mathbb{C}$.

We do a lot of 2-categorical calculations in **CAT**. To reduce the notational burden, we omit writing the composition operator $\circ$ between functors, or a functor and a natural transformation. For instance, for functors G, F, P, Q and a natural transformation $\alpha : P \to Q$, by $G\alpha F$ we mean the natural transformation $G(\alpha_{FI}) : G \circ P \circ F(I) \to G \circ Q \circ F(I)$. We use $\bullet$ and $*$ for the vertical and horizontal compositions of natural transformations, respectively.

Let A be a set and $X \in \mathbb{C}$. An *A-fold cotensor of X* is a pair of an object $A \pitchfork X$ and an A-indexed family of projection morphisms $\{\pi_a : A \pitchfork X \to X\}_{a\in A}$. They satisfy the following universal property: for any A-indexed family of morphisms $\{f_a : B \to X\}_{a\in A}$, there exists a unique morphism $m : B \to A \pitchfork X$ such that $\pi_a \circ m = f_a$ holds for all $a \in A$. Here are some examples of cotensors:

1. When $\mathbb{C} = \mathbf{Set}$, the function space $A \Rightarrow X$ and the evaluation function $\pi_a(f) = f(a)$ give an A-fold cotensor of X.
2. When $\mathbb{C}$ has small products, the product of A-fold copies of X and the associated projections give an A-fold cotensor of X.

3. When $\mathbb{C}$ has A-fold cotensors, any functor category $[\mathbb{D}, \mathbb{C}]$ also has A-fold cotensors, which can be given pointwisely: $(A \pitchfork F)X = A \pitchfork (FX)$.

A *right Kan extension* of $F : \mathbb{A} \to \mathbb{C}$ along $G : \mathbb{A} \to \mathbb{D}$ is a pair of a functor $\mathbf{Ran}_G F : \mathbb{D} \to \mathbb{C}$ and a natural transformation $c : \mathbf{Ran}_G F \circ G \to F$ making the following mapping ϕ_H:

$$\phi_H(\alpha) = c \bullet (\alpha G) : [\mathbb{D}, \mathbb{C}](H, \mathbf{Ran}_G F) \to [\mathbb{A}, \mathbb{C}](H \circ G, F)$$

bijective and natural on $H \in [\mathbb{D}, \mathbb{C}]$. A functor $p : \mathbb{C} \to \mathbb{C}'$ *preserves* a right Kan extension $(\mathbf{Ran}_G F, c)$ if $(p(\mathbf{Ran}_G F), pc)$ is a right Kan extension of pF along G. Thus for any right Kan extension $(\mathbf{Ran}_G(pF), c')$ of pF along G, we have $p(\mathbf{Ran}_G F) \simeq \mathbf{Ran}_G(pF)$ by the universal property.

Let $\mathcal{T}$ be a monad on a category $\mathbb{C}$. Its components are denoted by (T, η, μ). The Kleisli lifting of a morphism $f : I \to TJ$ is $\mu_J \circ Tf : TI \to TJ$, denoted by $f^{\#}$. We write $J : \mathbb{C} \to \mathbb{C}_{\mathcal{T}}$ and $K : \mathbb{C}_{\mathcal{T}} \to \mathbb{C}$ for the Kleisli adjunction of $\mathcal{T}$, and $\epsilon : JK \to \mathrm{Id}_{\mathbb{C}}$ for the counit of this adjunction. When $\mathcal{T}$ is decorated with an extra symbol, like $\dot{\mathcal{T}}$, the same decoration is applied to the notation of adjunction, like $\dot{\eta}$, $\dot{J}$, $\dot{\epsilon}$, etc.

For the definition of fibrations and related concepts, see [4].

▶ **Proposition 1** ([4, Exercise 9.2.4]). *Let $p : \mathbb{E} \to \mathbb{B}$ be a fibration, and assume that $\mathbb{B}$ has small limits. If p has fibred small limits, then $\mathbb{E}$ has small limits and p preserves them.*

2 Codensity Lifting of Monads

Fix a fibration $p : \mathbb{E} \to \mathbb{B}$ and a monad $\mathcal{T}$ on $\mathbb{B}$. We first introduce the main subject of this study, *liftings* of $\mathcal{T}$.

▶ **Definition 2.** A *lifting* of $\mathcal{T}$ (along p) is a monad $\dot{\mathcal{T}}$ on $\mathbb{E}$ such that $p\dot{T} = Tp, p\dot{\eta} = \eta p$ and $p\dot{\mu} = \mu p$.

We do not require *fibredness* on $\dot{\mathcal{T}}$. The codensity lifting is a method to construct a lifting of $\mathcal{T}$ from the following data called *lifting parameter*.

▶ **Definition 3.** A *lifting parameter* (for $\mathcal{T}$) is a span $\mathbb{B}_{\mathcal{T}} \xleftarrow{R} \mathbb{A} \xrightarrow{S} \mathbb{E}$ of functors such that $KR = pS$. We say that it is *single* if $\mathbb{A} = 1$.

A single lifting parameter is thus a pair (R, S) of objects $R \in \mathbb{B}$ and $S \in \mathbb{E}_{TR}$. This is the same data used in the original (single-result) categorical $\top\top$-lifting in [6].

In this section we first introduce the *codensity lifting* under the situation where the fibration and the lifting parameter satisfy the following *codensity condition*.

▶ **Definition 4.** We say that a fibration $p : \mathbb{E} \to \mathbb{B}$ and a functor $S : \mathbb{A} \to \mathbb{E}$ satisfy the *codensity condition* if

1. a right Kan extension of S along S exists, and
2. $p : \mathbb{E} \to \mathbb{B}$ preserves this right Kan extension.

Later in Section 6, we give the codensity lifting without relying on the codensity condition. Although it is applicable to wider situations, the codensity lifting using the right Kan extension given below has a conceptually simpler description.

The codensity condition relates the size of $\mathbb{A}$ and the completeness of $\mathbb{E}$.

▶ **Proposition 5.** *Let p be a fibration and $\mathbb{A}$ be a category. If one of the following conditions holds:*

1. *$\mathbb{E}$ has, and p preserves cotensors, and $\mathbb{A} = 1$*
2. *$\mathbb{E}$ has, and p preserves small products, and $\mathbb{A}$ is small discrete*
3. *$\mathbb{E}$ has, and p preserves small limits, and $\mathbb{A}$ is small*

then for any functor $S : \mathbb{A} \to \mathbb{E}$ from a category $\mathbb{A}$ satisfying the condition, the pair p, S satisfies the codensity condition.

▶ **Proposition 6.** *For any fibration p and right adjoint functor $S : \mathbb{A} \to \mathbb{E}$, p, S satisfies the codensity condition.*

Proof. Let P be a left adjoint of S. Then the assignment $F \mapsto FP$ extends to a right Kan extension of F along S. This Kan extension is *absolute* [11, Proposition X.7.3]. ◀

Fix a lifting parameter $\mathbb{B}_{\mathcal{T}} \xleftarrow{R} \mathbb{A} \xrightarrow{S} \mathbb{E}$ and assume that the fixed p, S satisfies the codensity condition. We take a right Kan extension $(\mathbf{Ran}_S S, c_S : (\mathbf{Ran}_S S)S \to S)$. As p preserves this right Kan extension, $(p(\mathbf{Ran}_S S), pc_S)$ is a right Kan extension of pS along S. Thus the following mapping:

$$\underline{(-)} = pc_S \bullet -S : [\mathbb{E}, \mathbb{B}](H, p(\mathbf{Ran}_S S)) \to [\mathbb{A}, \mathbb{B}](HS, pS).$$

is bijective and natural on $H : \mathbb{E} \to \mathbb{B}$. We write $\overline{(-)}$ for its inverse.

The right Kan extension $\mathbf{Ran}_S S$ is the functor part of the *codensity monad* [11, Exercise X.7.3]. Its unit $u_S : \mathrm{Id} \to \mathbf{Ran}_S S$ and multiplication $m_S : (\mathbf{Ran}_S S)\mathbf{Ran}_S S \to \mathbf{Ran}_S S$ are respectively given by the unique natural transformations such that $c_S \bullet u_S S = \mathrm{id}_S$ and $c_S \bullet m_S S = c_S \bullet (\mathbf{Ran}_S S)c_S$.

The codensity lifting constructs a lifting $\mathcal{T}^{\top\top} = (T^{\top\top}, \eta^{\top\top}, \mu^{\top\top})$ of $\mathcal{T}$ along p as follows.

We first lift the endofunctor T. We send $K\epsilon R : KJpS = KJKR \to KR = pS$ to $\overline{K\epsilon R} : Tp \to p(\mathbf{Ran}_S S)$, then take its cartesian lifting with respect to $\mathbf{Ran}_S S$; This is possible because $[\mathbb{E}, p] : [\mathbb{E}, \mathbb{E}] \to [\mathbb{E}, \mathbb{B}]$ is a fibration. We name the cartesian lifting σ. We then define $T^{\top\top}$ to be the codomain of σ.

$$\begin{tikzcd}
T^{\top\top} \arrow[r, dotted, "\sigma"] & \mathbf{Ran}_S S & {[\mathbb{E}, \mathbb{E}]} \arrow[d, "{[\mathbb{E},p]}"] \\
Tp \arrow[r, "\overline{K\epsilon R}"'] & p(\mathbf{Ran}_S S) & {[\mathbb{E}, \mathbb{B}]}
\end{tikzcd}$$

We next lift the unit η. Consider the following diagram:

$$\begin{tikzcd}
\mathrm{Id}_{\mathbb{E}} \arrow[dr, dotted, "\eta^{\top\top}"'] \arrow[drr, bend left, "u_S"] & & & \\
& T^{\top\top} \arrow[r, "\sigma"'] & \mathbf{Ran}_S S & {[\mathbb{E}, \mathbb{E}]} \arrow[dd, "{[\mathbb{E},p]}"] \\
p \arrow[dr, "\eta p"'] \arrow[drr, bend left, "pu_S"] & & & \\
& Tp \arrow[r, "\overline{K\epsilon R}"'] & p(\mathbf{Ran}_S S) & {[\mathbb{E}, \mathbb{B}]}
\end{tikzcd}$$

The triangle in the base category commutes by:

$$\overline{K\epsilon R} \bullet \eta p = \overline{K\epsilon R \bullet \eta pS} = \overline{K\epsilon R \bullet \eta KR} = \overline{\mathrm{id}_{KR}} = \overline{\mathrm{id}_{pS}} = pu_S.$$

Therefore from the universal property of σ, we obtain the unique natural transformation $\eta^{\top\top}$ above ηp making the triangle in the total category commute.

We finally lift the multiplication μ. Consider the following diagram.

$$\begin{tikzcd}
T^{\top\top}T^{\top\top} \arrow[r, "T^{\top\top}\sigma"] \arrow[dr, dotted, "\mu^{\top\top}"'] & T^{\top\top}\mathbf{Ran}_S S \arrow[r, "\sigma\mathbf{Ran}_S S"] & (\mathbf{Ran}_S S)\mathbf{Ran}_S S \arrow[dr, bend left, "m_S"] & & \\
& T^{\top\top} \arrow[rr, "\sigma"] & & \mathbf{Ran}_S S & {[\mathbb{E},\mathbb{E}]} \arrow[dd, "{[\mathbb{E},p]}"] \\
TTp \arrow[r, "T\overline{K\epsilon R}"] \arrow[dr, "\mu p"'] & Tp(\mathbf{Ran}_S S) \arrow[r, "\overline{K\epsilon R}\mathbf{Ran}_S S"] & p(\mathbf{Ran}_S S)\mathbf{Ran}_S S \arrow[dr, bend left, "pm_S"] & & \\
& Tp \arrow[rr, "\overline{K\epsilon R}"'] & & p(\mathbf{Ran}_S S) & {[\mathbb{E},\mathbb{B}]}
\end{tikzcd}$$

The pentagon in the base category commutes by:

$$\begin{aligned}
\underline{pm_S \bullet \overline{K\epsilon R}\mathbf{Ran}_S S \bullet T\overline{K\epsilon R}} &= pc_S \bullet p(\mathbf{Ran}_S S)c_S \bullet \overline{K\epsilon R}(\mathbf{Ran}_S S)S \bullet T\overline{K\epsilon R}S \\
\text{(interchange law)} &= pc_S \bullet \overline{K\epsilon R}S \bullet Tpc_S \bullet T\overline{K\epsilon R}S = K\epsilon R \bullet KJK\epsilon R \\
&= K\epsilon R \bullet \mu KR = K\epsilon R \bullet \mu pS = \underline{\overline{K\epsilon R} \bullet \mu p}.
\end{aligned}$$

Therefore from the universal property of σ, we obtain the unique morphism $\mu^{\top\top}$ above μp making the pentagon in the total category commute. We take $\mu^{\top\top}$ as the lifting of μ.

▶ **Theorem 7.** *Let $p : \mathbb{E} \to \mathbb{B}$ be a fibration, $\mathcal{T}$ be a monad on $\mathbb{B}$, $\mathbb{B}_{\mathcal{T}} \xleftarrow{R} \mathbb{A} \xrightarrow{S} \mathbb{E}$ be a lifting parameter for $\mathcal{T}$, and assume that p, S satisfies the codensity condition. The tuple $\mathcal{T}^{\top\top} = (T^{\top\top}, \eta^{\top\top}, \mu^{\top\top})$ constructed as above is a lifting of $\mathcal{T}$ along p.*

▶ **Corollary 8.** *The cartesian morphism $\sigma : T^{\top\top} \to \mathbf{Ran}_S S$ is a monad morphism.*

Any lifting of $\mathcal{T}$ along p can be obtained by the codensity lifting, although the choice of the lifting parameter is rather canonical.

▶ **Theorem 9.** *Let $p : \mathbb{E} \to \mathbb{B}$ be a fibration, $\mathcal{T}$ be a monad on $\mathbb{B}$ and $\dot{\mathcal{T}}$ be a lifting of $\mathcal{T}$. Then there exists a lifting parameter R, S such that p, S satisfies the codensity condition and $\dot{\mathcal{T}} \simeq \mathcal{T}^{\top\top}$.*

Proof. We write $p_k : \mathbb{E}_{\dot{\mathcal{T}}} \to \mathbb{B}_{\mathcal{T}}$ for the canonical functor extending $p : \mathbb{E} \to \mathbb{B}$ to Kleisli categories. Then the span $\mathbb{B}_{\mathcal{T}} \xleftarrow{p_k} \mathbb{E}_{\dot{\mathcal{T}}} \xrightarrow{\dot{K}} \mathbb{E}$ is a lifting parameter that satisfies the codensity condition by Proposition 6. We can even choose $\mathbf{Ran}_{\dot{K}}\dot{K}$ so that it equals $\dot{\mathcal{T}}$. Then the morphism $\overline{K\epsilon p_k} : Tp \to p(\mathbf{Ran}_{\dot{K}}\dot{K}) = Tp$ becomes the identity morphism. Hence $\dot{\mathcal{T}}$ is isomorphic to $\mathcal{T}$. ◀

3 Examples of Codensity Liftings with Single Lifting Parameters

We illustrate some examples of the codensity liftings of monads. The fibration $p : \mathbb{E} \to \mathbb{B}$ appearing in each example has fibred small limits, and its base category $\mathbb{B}$ has small limits. Hence $\mathbb{E}$ also has small limits that are preserved by p (Proposition 1). We focus on the codensity liftings of monads with single lifting parameters. We give a general scheme to calculate them.

▶ **Proposition 10.** *Let $p : \mathbb{E} \to \mathbb{B}$ a fibration such that p has fibred small limits and $\mathbb{B}$ has small limits, $\mathcal{T}$ be a monad on $\mathbb{B}$, and $R \in \mathbb{B}, S \in \mathbb{E}_{TR}$ be a single lifting parameter. Then the functor part of $\mathcal{T}^{\top\top}$ satisfies*

$$T^{\top\top}X \simeq \bigwedge_{f \in \mathbb{E}(X,S)} ((pf)^{\#})^{-1}(S) \tag{1}$$

where $\bigwedge$ stands for the fibred product in $\mathbb{E}_{T(pX)}$.

3.1 Lifting Set-Monads to the Category of Preorders

The canonical forgetful functor $p : \mathbf{Pre} \to \mathbf{Set}$ from the category $\mathbf{Pre}$ of preorders and monotone functions is a fibration with fibred small limits: the inverse image of a preorder $(J, \leq_J)$ along a function $f : I \to J$ is the preorder $(I, \leq_I)$ given by $i \leq_I i' \iff f(i) \leq_J f(i')$. The fibred small limits are given by the set-theoretic intersections of preorders on the same set. We note that p does not preserve exponentials, hence the $\top\top$-lifting in [6] is not applicable to p.

We consider the codensity lifting of a monad $\mathcal{T}$ over $\mathbf{Set}$ along $p : \mathbf{Pre} \to \mathbf{Set}$ with a single lifting parameter: a pair of $R \in \mathbf{Set}$ and $S = (TR, \leq) \in \mathbf{Pre}$. By instantiating (1), for every $(X, \leq_X) \in \mathbf{Pre}$ (X for short), the preorder $T^{\top\top}X$ is of the form $(TX, \leq_X^{\top\top})$ where the preorder $\leq_X^{\top\top}$ is given by

$$x \leq_X^{\top\top} y \iff \forall f \in \mathbf{Pre}(X, S) \;.\; (pf)^{\#}(x) \leq (pf)^{\#}(y). \tag{2}$$

We further instantiate this by letting $\mathcal{T}$ be the powerset monad $\mathcal{T}_p$, $R = 1$ and $\leq$ be the following partial orders on $T_p 1 = \{\emptyset, 1\}$:

1. Case $\leq = \{(\emptyset,\emptyset),(\emptyset,1),(1,1)\}$. The homset $\mathbf{Pre}(X,S)$ is isomorphic to the set $\mathbf{Up}(X)$ of upward closed subsets of X, and (2) is rewritten to:
$$\begin{aligned} x \leq_X^{\top\top} y \quad &\iff \quad (\forall F \in \mathbf{Up}(X) \;.\; x \cap F \neq \emptyset \implies y \cap F \neq \emptyset) \\ &\iff \quad \forall i \in x \;.\; \exists j \in y \;.\; i \leq_X j, \end{aligned}$$
that is, $\leq_X^{\top\top}$ is the lower preorder.
2. Case $\leq = \{(\emptyset,\emptyset),(1,\emptyset),(1,1)\}$. By the similar argument, $\leq^{\top\top}$ is the upper preorder:
$$x \leq_X^{\top\top} y \quad \iff \quad \forall j \in y \;.\; \exists i \in x \;.\; i \leq_X j.$$

In order to make $\leq^{\top\top}$ the *convex preorder* on $\mathcal{T}_p$:

$$x \leq_X^{\top\top} y \iff (\forall i \in x \;.\; \exists j \in y \;.\; i \leq_X j) \wedge (\forall j \in y \;.\; \exists i \in x \;.\; i \leq_X j),$$

we supply the cotupling $\mathbf{Set}_{\mathcal{T}_p} \leftarrow 1+1 \to \mathbf{Pre}$ of the above lifting parameters to the codensity lifting.

3.2 Lifting Set-Monads to the Category of Topological Spaces

The canonical forgetful functor $p : \mathbf{Top} \to \mathbf{Set}$ from the category $\mathbf{Top}$ of topological spaces and continuous functions is a fibration with fibred small limits. For a topological space $(X, \mathcal{O}_X)$ and a function $f : Y \to X$, the inverse image topological space $f^*(X, \mathcal{O}_X)$ is given by $(Y, \{f^{-1}(U) \mid U \in \mathcal{O}_X\})$. We note that each fibre category $\mathbf{Top}_X$ is the poset of topological spaces on a set X ordered in the opposite direction, that is, $(X, \mathcal{O}_1) \leq (X, \mathcal{O}_2)$ holds if and only if $\mathcal{O}_2 \subseteq \mathcal{O}_1$.

We consider the codensity lifting of a monad $\mathcal{T}$ over $\mathbf{Set}$ along $p : \mathbf{Top} \to \mathbf{Set}$ with a single lifting parameter: a pair of $R \in \mathbf{Set}$ and $S = (TR, \mathcal{O}_S) \in \mathbf{Top}$. By instantiating (1), for every $(X, \mathcal{O}_X) \in \mathbf{Top}$ (X for short), $T^{\top\top}X$ is the topological space $(TX, T^{\top\top}\mathcal{O}_X)$ whose topology $T^{\top\top}\mathcal{O}_X$ is the coarsest one making every set $((pf)^{\#})^{-1}(U)$ open, where f and U range over $\mathbf{Top}(X, S)$ and $\mathcal{O}_S$, respectively.

We further instantiate this by letting $\mathcal{T} = \mathcal{T}_p, R = 1$, and $\mathcal{O}_S$ be the following topologies on T_p1. The topologies given to powersets by the following liftings are similar to *lower* and *upper Vietoris topology.*

1. Case $\mathcal{O}_S = \{\emptyset, \{1\}, \{\emptyset, 1\}\}$. The topology $T_p^{\top\top}\mathcal{O}_X$ is the coarsest one making every set $\{V \subseteq pX \mid V \cap U \neq \emptyset\}$ open, where U ranges over $\mathcal{O}_X$. We call this *lower Vietoris lifting.*
2. Case $\mathcal{O}_S = \{\emptyset, \{\emptyset\}, \{\emptyset, 1\}\}$. The topology $T_p^{\top\top}\mathcal{O}_X$ is the coarsest one making every set $\{V \subseteq pX \mid V \subseteq U\}$ open, where U ranges over $\mathcal{O}_X$. We call this *upper Vietoris lifting.*

3.3 Simulations on Labelled Markov Processes by Codensity Lifting

We next move on to the category **Meas** of measurable spaces and measurable functions between them. Recall that **Meas** has small limits (as the canonical forgetful functor $U : \mathbf{Meas} \to \mathbf{Set}$ is topological). We introduce some notations: For $X \in \mathbf{Meas}$, by $\mathcal{M}_X$ we mean the σ-algebra of X. For $X \in \mathbf{Top}$, by $\mathcal{B}X \in \mathbf{Meas}$ we mean the Borel (measurable) space of X.

We consider the following two fibrations q, r obtained by the change-of-base of the subobject fibration of **Set**:

$$\begin{tikzcd}
\mathbf{ERel}(\mathbf{Meas}) \arrow[r] \arrow[d, "r"'] \arrow[dr, phantom, very near start, "\lrcorner"] & \mathbf{BRel}(\mathbf{Meas}) \arrow[rr] \arrow[d, "q"'] \arrow[drr, phantom, very near start, "\lrcorner"] & & \mathbf{Pred} \arrow[d, "p"] \\
\mathbf{Meas} \arrow[r, "\Delta"'] & \mathbf{Meas}^2 \arrow[r, "U^2"'] & \mathbf{Set}^2 \arrow[r, "Prod"'] & \mathbf{Set}
\end{tikzcd}$$

Here, Δ is the diagonal functor and and $Prod$ is the product functor. The legs q and r of the change-of-base are fibrations with fibred small limits. [1] The explicit description of **BRel(Meas)** is:

- An object X is a triple, whose components are denoted by X_0, X_1, X_2, such that X_1, X_2 are measurable spaces and $X_0 \subseteq UX_0 \times UX_1$.
- A morphism $(f_1, f_2) : X \to Y$ is a pair of measurable functions $f_1 : X_1 \to Y_1$ and $f_2 : X_2 \to Y_2$ such that $(Uf_1 \times Uf_2)(X_0) \subseteq X_1$.

The explicit description of **ERel(Meas)** is:

- An object is a pair, whose components are denoted by X_0, X_1, such that X_1 is a measurable space and $X_0 \subseteq UX_1 \times UX_1$.
- A morphism $f : X \to Y$ is a measurable function $f : X_1 \to Y_1$ such that $(Uf \times Uf)(X_0) \subseteq Y_0$.

For a binary relation $R \subseteq X \times Y$ and $A \subseteq X$, the image of A by R is defined to be the set $\{y \in Y \mid \exists x \in A \,.\, (x, y) \in R\}$, and is denoted by $R[A]$.

For $X \in \mathbf{Meas}$, by $\mathbf{SPMsr}(X)$ we mean the set of sub-probability measures on X. We equip it with the σ-algebra generated from the sets of the following form:

$$\{\mu \in \mathbf{SPMsr}(X) \mid \mu(U) \in V\} \quad (U \in \mathcal{M}_X, V \in \mathcal{M}_{\mathcal{B}[0,1]}),$$

and denote this measurable space by GX. The assignment $X \mapsto GX$ can be extended to a monad $\mathcal{G}$ on **Meas**, called *Giry monad* [2]. Notice that $G1 = \mathcal{B}[0, 1]$.

[1] **BRel** and **ERel** stands for binary relations and endo-relations, respectively.

We consider the codensity lifting of $\mathcal{G}$ along $r : \mathbf{ERel}(\mathbf{Meas}) \to \mathbf{Meas}$ with a single lifting parameter $R = 1$ (the one-point measurable space) and $S = (\leq, G1)$; here $\leq$ is the usual order on $[0,1] = U(G1)$. By instantiating (1), we obtain

$$(v_1, v_2) \in (G^{\top\top}X)_0 \iff \forall f \in \mathbf{ERel}(\mathbf{Meas})(X, S) \; . \int_{X_1} f \; dv_1 \leq \int_{X_1} f \; dv_2.$$

▶ **Theorem 11.** *The relation part* $(G^{\top\top}X)_0$ *satisfies:*

$$(v_1, v_2) \in (G^{\top\top}X)_0 \iff (\forall U \in \mathcal{M}_{X_1} \; . \; X_0[U] \subseteq U \implies v_1(U) \leq v_2(U)).$$

Proof.

($\subseteq$) Suppose $(v_1, v_2) \in G^{\top\top}X_0$. Let $U \in \mathcal{M}_{X_1}$ be a measurable set satisfying $X_0[U] \subseteq U$. The indicator function χ_U is a morphism in $\mathbf{ERel}(\mathbf{Meas})$ from X to S. Hence,

$$v_1(U) = \int_{X_1} \chi_U \; dv_1 \leq \int_{X_1} \chi_U \; dv_2 = v_2(U).$$

($\supseteq$) Suppose that $X_0[U] \subseteq U \implies v_1(U) \leq v_2(U)$ holds for all $U \in \mathcal{M}_{X_1}$. Let $f \in \mathbf{ERel}(\mathbf{Meas})(X, S)$ be a morphism and $\sum_{i=0}^{n} \alpha_i \chi_{A_i} \leq f$ be a positive measurable simple function. Without loss of generality, we may assume $A_0 \supseteq A_1 \supseteq \cdots \supseteq A_n$ and $\sum_{i=0}^{n} \alpha_i \leq 1$. Let C_i be $f^{-1}\left([\sum_{k=0}^{i} \alpha_k, 1]\right)$, the inverse image of the closed interval $[\sum_{k=0}^{i} \alpha_k, 1]$ along f. We have $\sum_{i=0}^{n} \alpha_i \chi_{A_i} \leq \sum_{i=0}^{n} \alpha_i \chi_{C_i} \leq f$, and we obtain $C_i \in \mathcal{M}_{X_1}$ and $X_0[C_i] \subseteq C_i$ because $f \in \mathbf{ERel}(\mathbf{Meas})(X, S)$. Hence,

$$\sum_{i=0}^{n} \alpha_i v_1(A_i) \leq \sum_{i=0}^{n} \alpha_i v_1(C_i) \leq \sum_{i=0}^{n} \alpha_i v_2(C_i) \leq \int_{X_1} f \; dv_2.$$

This implies

$$\int_{X_1} f \; dv_1 = \sup \left\{ \sum_{i=0}^{n} \alpha_i v_1(A_i) \;\middle|\; \sum_{i=0}^{n} \alpha_i \chi_{A_i} \leq f \right\} \leq \int_{X_1} f \; dv_2. \qquad ◀$$

This lifting is related to the concept of *simulation relation* between two states on the same *labelled Markov process (LMP)* in [15]. Let Act be a set (of actions). An LMP over $X_1 \in \mathbf{Meas}$ is a measurable function $x : X_1 \to Act \pitchfork GX_1$. Then a reflexive relation $X_0 \subseteq UX_1 \times UX_1$ is a *simulation* in the sense of [15, Definition 3] if and only if x is a morphism of type $(X_0, X_1) \to Act \pitchfork G^{\top\top}(X_0, X_1)$ in $\mathbf{ERel}(\mathbf{Meas})$.

We next consider the codensity lifting of the product Giry monad $\mathcal{G}^2$ on $\mathbf{Meas}^2$ along $q : \mathbf{BRel}(\mathbf{Meas}) \to \mathbf{Meas}^2$ with a single lifting parameter $R = (1, 1)$ and $S = (\leq, G1, G1)$. By instantiating (1), we obtain

$$(v_1, v_2) \in (G^{\top\top}X)_0 \iff \forall (f_1, f_2) \in \mathbf{ERel}(\mathbf{Meas})(X, S) \; . \int_{X_1} f_1 \; dv_1 \leq \int_{X_2} f_2 \; dv_2.$$

▶ **Theorem 12.** *The relation part* $(G^{\top\top}X)_0$ *satisfies:*

$$(v_1, v_2) \in (G^{\top\top}X)_0 \iff (\forall U \in \mathcal{M}_{X_1}, V \in \mathcal{M}_{X_2} \; . \; X_0[U] \subseteq V \implies v_1(U) \leq v_2(V)).$$

Employing this lifting, we naturally obtain the concept of simulation relation between two states in different LMPs. Let $X \in \mathbf{BRel}(\mathbf{Meas})$ and $x_i : X_i \to Act \pitchfork GX_i$ be LMPs

$(i = 1, 2)$. We say that X is a simulation from x_1 to x_2 if (x_1, x_2) is a morphism of type $X \to \mathit{Act} \pitchfork G^{\top\top} X$ in $\mathbf{BRel}(\mathbf{Meas})$. This is equivalent to:

$$\forall (s_1, s_2) \in X_0 \;.\; \forall U \in \mathcal{M}_{X_1}, V \in \mathcal{M}_{X_2} \;.\; X_0[U] \subseteq V \implies x_1(s_1)(U) \le x_2(s_2)(V).$$

One natural property we expect on simulation relations between LMPs is the *composability*. However, $\mathcal{G}^{\top\top}$ fails to satisfy the *lax compositionality* $(G^{\top\top} X)_0; (G^{\top\top} Y)_0 \subseteq (G^{\top\top}(X; Y))_0$ for general X, Y; here ";" is the left-first relation composition. Therefore the above definition of simulation relation is not closed under the relation composition. One way to solve this problem is to require each simulation relation X to preserve measurability in the following sense: $\forall U \in \mathcal{M}_{X_1} \;.\; X_0[U] \in \mathcal{M}_{X_2}$.

3.4 Kantorovich Metric by Codensity Lifting

An *extended pseudometric space* (we drop "extended" hereafter) is a pair (X, d) of a set X and a pseudometric $d : X \times X \to [0, \infty]$ giving distances (including ∞) between elements in X. The axioms for pseudometrics are

$$d(x, x) = 0, \quad d(x, y) = d(y, x), \quad d(x, y) + d(y, z) \ge d(x, z).$$

For pseudometric spaces $(X, d), (Y, e)$, a function $f : X \to Y$ is *non-expansive* if for any $x, x' \in X$, $d(x, x') \ge e(f(x), f(x'))$ holds. We define $\mathbf{EPMet}$ to be the category of extended pseudometric spaces and non-expansive functions. The canonical forgetful functor $p : \mathbf{EPMet} \to \mathbf{Set}$ is a fibration with fibred small limits. The inverse image of a pseudometric (Y, d) along a function $f : X \to Y$ is given by $f^*(Y, d) = (X, d \circ (f \times f))$. The fibred small limit of pseudometric spaces $\{(X, d_i)\}_{i \in I}$ above the same set X is given by the pointwise sup of pseudometrics: $\bigwedge_{i \in I}(X, d_i) = (X, \sup_{i \in I} d_i)$.

We first consider the codensity lifting of a monad $\mathcal{T}$ on $\mathbf{Set}$ along $p : \mathbf{EPMet} \to \mathbf{Set}$ with a single lifting parameter: a pair of $R \in \mathbf{Set}$ and $S = (TR, s) \in \mathbf{EPMet}$. By instantiating (1), for every $(X, d) \in \mathbf{EPMet}$ (X for short), the pseudometric space $T^{\top\top} X$ is of the form $(TX, T^{\top\top} d)$ where the pseudometric $T^{\top\top} d$ is given by

$$T^{\top\top} d(c, c') = \sup_{f \in \mathbf{EPMet}(X, S)} s(f^{\#}(c), f^{\#}(c')).$$

The following example is inspired by Ogawa's work deriving Kantorovich metric on subprobability distributions [13]. We perform the following change-of-base of the fibration

$$\begin{tikzcd}
U^*(\mathbf{EPMet}) \arrow[r] \arrow[d, "q"'] \arrow[dr, phantom, very near start, "\lrcorner"] & \mathbf{EPMet} \arrow[d, "p"] \\
\mathbf{Meas} \arrow[r, "U"'] & \mathbf{Set}
\end{tikzcd}$$

We obtain a new fibration q with fibred small limits. An object in $U^*(\mathbf{EPMet})$ is a pair of a measurable space $(X, \mathcal{M}_X)$ and a pseudometric d on X. A morphism from $((X, \mathcal{M}_X), d)$ to $((Y, \mathcal{M}_Y), e)$ in $U^*(\mathbf{EPMet})$ is a measurable function $f : (X, \mathcal{M}_X) \to (Y, \mathcal{M}_Y)$ that is also non-expansive with respect to pseudometrics d and e.

We consider the codensity lifting of $\mathcal{G}$ along $q : U^*\mathbf{EPMet} \to \mathbf{Meas}$ with the following single lifting parameter: a pair of $R = 1$ and $S = (G1, s) = (\mathcal{B}[0, 1], s)$ where $s(x, y) = |x - y|$. For every $(X, d) \in \mathbf{EPMet}$ (X for short), $G^{\top\top} X$ is the pair of the measurable space GX and the following pseudometric $G^{\top\top} d$ on the set $\mathbf{SPMsr}(X)$ of subprobability measures on X:

$$G^{\top\top} d(v_1, v_2) = \sup_f \left| \int_X f dv_1 - \int_X f dv_2 \right| ;$$

in the above sup, f ranges over $U^*\mathbf{EPMet}(X,S)$, the set of measurable functions of type $X \to \mathcal{B}[0,1]$ that are also non-expansive, that is, $\forall x,y \in UX \ . \ d(x,y) \geq |f(x)-f(y)|$. The pseudometric $G^{\top\top}d$ between subprobability measures is called *Kantorovich metric* [5].

We briefly mention two works related to this lifting.

- In a recent work [1], Baldan et al. introduces *Kantorovich lifting* of **Set**-functors. Although they consider lifting of general **Set**-functors rather than **Set**-monads, their lifting scheme is very close to the codensity lifting of **Set**-monads along $p : \mathbf{EPMet} \to \mathbf{Set}$.
- Ogawa reported that the Kantorovich metric on finite subprobability distributions can be derived using the technique of *observational algebra* [13].

4 Lifting Algebraic Operations to Codensity-Lifted Monads

We adopt the concept of *algebraic operation* [14] for general monads, and discuss their liftings to codensity-lifted monads. The following definition is a modification of [14, Proposition 2] for non-strong monads, and coincides with the original one when $\mathbb{C} = \mathbf{Set}$.

▶ **Definition 13.** Let $\mathbb{C}$ be a category, A be a set and assume that $\mathbb{C}$ has A-fold cotensors. An *A-ary algebraic operation* for a monad $\mathcal{T}$ on $\mathbb{C}$ is a natural transformation $\alpha : A \pitchfork K \to K$ (see Section 1.1 for K). We write $\mathbf{Alg}(\mathcal{T},A)$ for the class of A-ary algebraic operations for $\mathcal{T}$.

▶ **Example 14.** For each set A, the powerset monad $\mathcal{T}_p$ has the algebraic operation of *A-ary set-union* $\mathrm{union}^A_X : A \pitchfork T_pX \to T_pX$ given by $\mathrm{union}^A_X(f) = \bigcup_{x\in A} f(x)$.

Fix a fibration $p : \mathbb{E} \to \mathbb{B}$, a monad $\mathcal{T}$ on $\mathbb{B}$, a set A and assume that $\mathbb{E}$ has and p preserves A-fold cotensors.

▶ **Definition 15.** Let $\dot{\mathcal{T}}$ be a lifting of $\mathcal{T}$ along p. A *lifting* of an algebraic operation $\alpha \in \mathbf{Alg}(\mathcal{T},A)$ to $\dot{\mathcal{T}}$ is an algebraic operation $\dot{\alpha} \in \mathbf{Alg}(\dot{\mathcal{T}},A)$ such that $p\dot{\alpha} = \alpha p_k$; here $p_k : \mathbb{E}_{\dot{\mathcal{T}}} \to \mathbb{B}_{\mathcal{T}}$ is the canonical extension of p to Kleisli categories. We write $\mathbf{Alg}_\alpha(\dot{\mathcal{T}},A)$ for the class $\{\dot{\alpha} \in \mathbf{Alg}(\dot{\mathcal{T}},A) \mid p\dot{\alpha} = \alpha p_k\}$ of liftings of α to $\dot{\mathcal{T}}$.

▶ **Example 16.** (Continued from Example 14) Let $\dot{\mathcal{T}}$ be a lifting of $\mathcal{T}_p$ along $p : \mathbf{Top} \to \mathbf{Set}$. Since p is faithful, there is at most one lifting of union^A to $\dot{\mathcal{T}}$. It exists if and only if for every $(X,\mathcal{O}_X) \in \mathbf{Top}$, union^A_X is a continuous function of type $A \pitchfork \dot{T}(X,\mathcal{O}_X) \to \dot{T}(X,\mathcal{O}_X)$.

We give a characterisation of the liftings of algebraic operations to codensity-lifted monads. Fix a lifting parameter $\mathbb{B}_{\mathcal{T}} \xleftarrow{R} \mathbb{A} \xrightarrow{S} \mathbb{E}$ and assume that p, S satisfies the codensity condition. Note that the canonical extension $p_k : \mathbb{E}_{\mathcal{T}^{\top\top}} \to \mathbb{B}_{\mathcal{T}}$ of p satisfies

$$p_k J^{\top\top} = Jp, \quad pK^{\top\top} = Kp_k, \quad p\eta^{\top\top} = \eta p, \quad p_k\epsilon^{\top\top} = \epsilon p_k.$$

Starting from a natural transformation $\alpha_0 : A \pitchfork S \to S$ such that $p\alpha_0 = \alpha R$, we construct a lifting $\phi(\alpha_0) \in \mathbf{Alg}_\alpha(\mathcal{T}^{\top\top},A)$ of α as follows.

From $A \pitchfork S = (A \pitchfork \mathrm{Id}_{\mathbb{E}})S$, the natural trasnformation α_0 induces the mate $\overline{\alpha_0} : A \pitchfork \mathrm{Id}_{\mathbb{E}} \to \mathbf{Ran}_S S$. We then obtain the following situation:

$$\begin{tikzcd}
A \pitchfork \mathrm{Id}_{\mathbb{E}} \arrow[dr, dotted, "\beta"'] \arrow[drr, bend left, "\overline{\alpha_0}"] & & & \\
& T^{\top\top} \arrow[r, "\sigma"'] & \mathbf{Ran}_S S & {[\mathbb{E}, \mathbb{E}]} \arrow[dd, "{[\mathbb{E}, p]}"] \\
A \pitchfork p \arrow[dr, "\alpha Jp \bullet A \pitchfork \eta p"'] \arrow[drr, bend left, "\overline{\alpha R}"] & & & \\
& Tp \arrow[r, "\overline{K\epsilon R}"'] & p(\mathbf{Ran}_S S) & {[\mathbb{E}, \mathbb{B}]}
\end{tikzcd}$$

The triangle in the base category commutes by:

$$\begin{aligned}
\overline{K\epsilon R} \bullet \alpha Jp \bullet A \pitchfork \eta p &= \overline{K\epsilon R \bullet \alpha JpS \bullet A \pitchfork \eta pS} = \overline{K\epsilon R \bullet \alpha JKR \bullet A \pitchfork \eta KR} \\
&= \overline{(K\epsilon \bullet \alpha JK \bullet A \pitchfork \eta K)R} = \overline{(\alpha \bullet A \pitchfork K\epsilon \bullet A \pitchfork \eta K)R} = p\overline{\alpha_0}.
\end{aligned}$$

We thus obtain the unique morphism β above $\alpha Jp \bullet A \pitchfork \eta p$ making the triangle in the total category commute. Using this β, we define $\phi(\alpha_0) : A \pitchfork K^{\top\top} \to K^{\top\top}$ by

$$\phi(\alpha_0) = K^{\top\top}\epsilon^{\top\top} \bullet \beta K^{\top\top} : A \pitchfork K^{\top\top} \to K^{\top\top}.$$

This algebraic operation is a lifting of α to $T^{\top\top}$:

$$p\phi(\alpha_0) = p(K^{\top\top}\epsilon^{\top\top} \bullet \beta K^{\top\top}) = (K\epsilon \bullet \alpha JK \bullet A \pitchfork \eta K)p_k = (\alpha \bullet A \pitchfork K\epsilon \bullet A \pitchfork \eta K)p_k = \alpha p_k.$$

The following theorem shows that ϕ characterises the class of liftings of α to the codensity-lifted monads. It is an analogue of Theorem 11 in [7], which is stated for the categorical $\top\top$-lifting.

▶ **Theorem 17.** *Let $p : \mathbb{E} \to \mathbb{B}$ be a fibration, $\mathcal{T}$ be a monad on $\mathbb{B}$, and $\mathbb{B}_{\mathcal{T}} \xleftarrow{R} \mathbb{A} \xrightarrow{S} \mathbb{E}$ be a lifting parameter, and A be a set. Suppose that $\mathbb{B}, \mathbb{E}$ has, and p preserves A-fold cotensor. Then for any $\alpha \in \mathbf{Alg}(\mathcal{T}, A)$, the mapping ϕ constructed as above has the following type and is bijective:*

$$\phi : [\mathbb{A}, \mathbb{E}]_{\alpha R}(A \pitchfork S, S) \to \mathbf{Alg}_\alpha(\mathcal{T}^{\top\top}, A).$$

▶ **Example 18.** (Continued from Example 16) We look at liftings of $\mathrm{union}^A \in \mathbf{Alg}(\mathcal{T}_p, A)$ to the codensity liftings of $\mathcal{T}_p$ along $p : \mathbf{Top} \to \mathbf{Set}$ with some single lifting parameters.

Let $R \in \mathbf{Set}$ and $S = (T_p R, \mathcal{O}_S) \in \mathbf{Top}$ be a single lifting parameter. Theorem 17 is instantiated to the following statement: a lifting of union^A to $\mathcal{T}_p^{\top\top}$ exists if and only if $\mathrm{union}^A_R : A \pitchfork T_p R \to T_p R$ is a continuous function of type $A \pitchfork S \to S$. Here, $A \pitchfork S$ is the product of A-fold copies of S, and its topology $\mathcal{O}_{A \pitchfork S}$ is generated from all the sets of the form $\pi_a^{-1}(U)$, where a and U range over A and $\mathcal{O}_S$, respectively. We further instantiate the single lifting parameter as follows (see Section 3.2):

1. Case $R = 1, \mathcal{O}_S = \{\emptyset, \{1\}, \{\emptyset, 1\}\}$. For any set A, union^A_1 is a continuous function of type $A \pitchfork S \to S$ because $(\mathrm{union}^A_1)^{-1}(\{1\}) = \bigcup_{a \in A} \pi_a^{-1}(\{1\}) \in \mathcal{O}_{A \pitchfork S}$. From Theorem 17, for any set A, union^A lifts to the lower Vietoris lifting $\mathcal{T}_p^{\top\top}$.
2. Case $R = 1, \mathcal{O}_S = \{\emptyset, \{\emptyset\}, \{\emptyset, 1\}\}$. For any finite set A, union^A_1 is a continuous function of type $A \pitchfork S \to S$ because $(\mathrm{union}^A_1)^{-1}(\{\emptyset\}) = \bigcap_{a \in A} \pi_a^{-1}(\{\emptyset\}) \overset{*}{\in} \mathcal{O}_{A \pitchfork S}$. On the other hand, the membership $\overset{*}{\in}$ does not hold when A is infinite. From Theorem 17, for any set A, union^A lifts to the upper Vietoris lifting $\mathcal{T}_p^{\top\top}$ if and only if A is finite.

5 Pointwise Codensity Lifting

Fix a fibration $p : \mathbb{E} \to \mathbb{B}$, a monad $\mathcal{T}$ on $\mathbb{B}$ and a lifting parameter $\mathbb{B}_{\mathcal{T}} \xleftarrow{R} \mathbb{A} \xrightarrow{S} \mathbb{E}$. When $\mathbb{A}$ is a large category, or $\mathbb{B}, \mathbb{E}$ are not very complete, the right Kan extension $\mathbf{Ran}_S S$ may not exist, hence the codensity lifting in Section 2 is not applicable to lift $\mathcal{T}$. In this section we introduce an alternative method (called *pointwise codensity lifting*) that relies on fibred limits of p. The point of this method is to swap the order of computation. Instead of taking the inverse image after computing $\mathbf{Ran}_S S$, we first take the inverse image of the components of $\mathbf{Ran}_S S$, bringing everything inside a fibre, then compute the right Kan extension as a fibred limit.

We assume that $\mathbb{A}$ is small (resp. large) and p has fibred small (resp. large) limits. The pointwise codensity lifting lifts $\mathcal{T}$ as follows.

We first lift T to an object mapping $\dot{T} : |\mathbb{E}| \to |\mathbb{E}|$. Let $X \in \mathbb{E}$. Consider the following diagram:

$$\begin{tikzcd}
X \downarrow S \arrow[r, "\pi_X"] \arrow[d, "!_{X\downarrow S}"'] \arrow[dr, phantom, "\Rightarrow \gamma_X"] & \mathbb{A} \arrow[r, "R"] \arrow[d, "S"] & \mathbb{B}_{\mathcal{T}} \arrow[r, equal] \arrow[d, "K"'] \arrow[dr, phantom, "\Rightarrow \epsilon", near start] & \mathbb{B}_{\mathcal{T}} \arrow[d, "K"] \\
1 \arrow[r, "X"'] & \mathbb{E} \arrow[r, "p"'] & \mathbb{B} \arrow[r, "T"'] \arrow[ur, "J"'] & \mathbb{B}
\end{tikzcd}$$

where $(X \downarrow S, \pi_X, !_{X\downarrow S}, \gamma_X)$ is the *comma category*. The middle square commutes as R, S is a lifting parameter. We let $\delta_X = K\epsilon R\pi_X \bullet Tp\gamma_X$ be the composite natural transformation, and take the inverse image of $S\pi_X$ along δ_X:

$$\begin{tikzcd}
\delta_X^{-1}(S\pi_X) \arrow[r, dotted, "\overline{\delta_X}(S\pi_X)"] & S\pi_X & {[X \downarrow S, \mathbb{E}]} \arrow[d, "{[X\downarrow S, p]}"] \\
TpX!_{X\downarrow S} \arrow[r, "\delta_X"'] & KR\pi_X & {[X \downarrow S, \mathbb{B}]}
\end{tikzcd}$$

We obtain a functor $\delta_X^{-1}(S\pi_X) : X \downarrow S \to \mathbb{E}$ such that $p\delta_X^{-1}(S\pi_X) = TpX!_{X\downarrow S}$. We then define $T^{\top\top}X$ by $T^{\top\top}X = \lim(\delta_X^{-1}(S\pi_X))$, where right hand side is the fibred limit. In the following calculations we will use the vertical projection and the tupling operation of this fibred limit, denoted by

$$P_X : (T^{\top\top}X)!_{X\downarrow S} \to \delta_X^{-1}(S\pi_X),$$
$$\langle - \rangle : [X \downarrow S, \mathbb{E}]_{f!_{X\downarrow S}}(Y!_{X\downarrow S}, \delta_X^{-1}(S\pi_X)) \to \mathbb{E}_f(Y, T^{\top\top}X) \quad (f \in \mathbb{E}(Y, TpX)).$$

We next lift η. Consider the following diagram:

$$\begin{tikzcd}
X!_{X\downarrow S} \arrow[dr, dotted, "\eta'_X"'] \arrow[drr, bend left, "\gamma_X"] & & & \\
& \delta_X^{-1}(S\pi_X) \arrow[r, "\overline{\delta_X}(S\pi_X)"'] & S\pi_X & {[X \downarrow S, \mathbb{E}]} \arrow[dd, "{[X\downarrow S, p]}"] \\
pX!_{X\downarrow S} \arrow[dr, "\eta pX!_{X\downarrow S}"'] \arrow[drr, bend left, "p\gamma_X"] & & & \\
& TpX!_{X\downarrow S} \arrow[r, "\delta_X"'] & KR\pi_X & {[X \downarrow S, \mathbb{B}]}
\end{tikzcd}$$

where the lower triangle commute by:

$$\delta_X \bullet \eta pX!_{X\downarrow S} = K\epsilon R\pi_X \bullet \eta pS\pi_X \bullet p\gamma_X = K\epsilon R\pi_X \bullet \eta KR\pi_X \bullet p\gamma_X = p\gamma_X.$$

Therefore there exists the unique natural transformation η'_X above $\eta pX!_{X\downarrow S}$ making the upper triangle commute. We define $\eta^{\top\top}_X = \langle \eta'_X \rangle$, which is above ηpX.

We finally lift the Kleisli lifting $(-)^{\#}$ of $\mathcal{T}$. Let $g : X \to T^{\top\top}Y$ be a morphism in $\mathbb{E}$, and $f = P_Y \bullet g!_{Y\downarrow S} : X!_{Y\downarrow S} \to \delta_Y^{-1}(S\pi_Y)$ be a morphism, which is above $pg!_{Y\downarrow S}$ and satisfies $g = \langle f \rangle$. We obtain the composite natural transformation $\overline{\delta_Y}(S\pi_Y) \bullet f : X!_{Y\downarrow S} \to \delta_Y^{-1}(S\pi_Y) \to S\pi_Y$. From the universal property of the comma category, we obtain the unique functor $M_f : Y \downarrow S \to X \downarrow S$ such that $\pi_X M_f = \pi_Y$ and $\gamma_X M_f = \overline{\delta_Y}(S\pi_Y) \bullet f$. We next consider the following diagram:

$$\begin{tikzcd}
\delta_X^{-1}(S\pi_X)M_f \arrow[dr, dotted, "f^\flat"'] \arrow[drr, bend left, "\overline{\delta_X}(S\pi_X)M_f"] & & & \\
& \delta_Y^{-1}(S\pi_Y) \arrow[r, "\overline{\delta_Y}(S\pi_Y)"'] & S\pi_Y & {[Y\downarrow S, \mathbb{E}]} \arrow[dd, "{[Y\downarrow S,p]}"] \\
TpX!_{Y\downarrow S} \arrow[dr, "\mu pY!_{Y\downarrow S} \bullet Tpf"'] \arrow[drr, bend left, "\delta_X M_f"] & & & \\
& TpY!_{Y\downarrow S} \arrow[r, "\delta_Y"'] & KR\pi_Y & {[Y\downarrow S, \mathbb{B}]}
\end{tikzcd}$$

where the lower triangle commutes. Therefore there exists the unique natural transformation $f^\flat$ above $\mu pY!_{Y\downarrow S} \bullet Tpf = \mu pY!_{Y\downarrow S} \bullet Tpg!_{Y\downarrow S} = (pg)^{\#}!_{Y\downarrow S}$ making the upper triangle commute. Then we define $g^{\#^{\top\top}} = \langle f^\flat \bullet P_X M_f \rangle$, which is above $(pg)^{\#}$.

▶ **Theorem 19.** *Let $p : \mathbb{E} \to \mathbb{B}$ be a fibration with fibred small (resp. large) limits, $\mathcal{T}$ be a monad on $\mathbb{B}$, $\mathbb{B}_\mathcal{T} \xleftarrow{R} \mathbb{A} \xrightarrow{S} \mathbb{E}$ be a lifting parameter for $\mathcal{T}$ and assume that $\mathbb{A}$ is small (resp. large). The tuple $(T^{\top\top}, \eta^{\top\top}, (-)^{\#^{\top\top}})$ constructed as above is a Kleisli triple on $\mathbb{E}$, and the corresponding monad is a lifting of $\mathcal{T}$.*

The pointwise codensity lifting coincides with the codensity lifting in Section 2, provided that $\mathbf{Ran}_S S$ and $p(\mathbf{Ran}_S S)$ are both pointwise.

▶ **Theorem 20.** *Let $p : \mathbb{E} \to \mathbb{B}$ be a fibration, $\mathcal{T}$ be a monad on $\mathbb{B}$ and $\mathbb{B}_\mathcal{T} \xleftarrow{R} \mathbb{A} \xrightarrow{S} \mathbb{E}$ be a lifting parameter. Assume that p, S satisfies the codensity condition, and moreover $\mathbf{Ran}_S S$ and $p(\mathbf{Ran}_S S)$ are both pointwise. Then $((\overline{K\epsilon R})^{-1}(\mathbf{Ran}_S S))X \simeq \lim(\delta_X^{-1}(S\pi_X))$.*

6 Characterising lift($\mathcal{T}$) as a Limit

We give a characterisation of the class of liftings of $\mathcal{T}$ as a limit of a large diagram. This is shown for *posetal* fibrations $p : \mathbb{E} \to \mathbb{B}$ with fibred small limits, which bijectively correspond to functors of type $\mathbb{B}^{op} \to \mathbf{Lat}_\wedge$; here $\mathbf{Lat}_\wedge$ is the category of complete lattices and meet-preserving functions. Notice that each fibre actually admits *large* limits computed by meets.

Fix such a fibration $p : \mathbb{E} \to \mathbb{B}$ and a monad $\mathcal{T}$ on $\mathbb{B}$. Since p is posetal, p is faithful. Thus we regard each homset $\mathbb{E}(X, Y)$ as a subset of $\mathbb{B}(pX, pY)$, and make p implicit.

▶ **Definition 21.** We define $\mathbf{lift}(\mathcal{T})$ to be the class of liftings of $\mathcal{T}$ along p. We introduce a partial order $\preceq$ on them by $\dot{T} \preceq \dot{T}' \iff \forall X \in \mathbb{E} \,.\, \dot{T}X \leqslant \dot{T}'X \quad (\text{in } \mathbb{E}_{T(pX)})$.

The partially ordered class $\mathbf{lift}(\mathcal{T})$ admits arbitrary large meets given by the pointwise meet.

We introduce a specific notation for the codensity liftings of $\mathcal{T}$ with a single lifting parameter R, S. By $[S]^R$ we mean the codensity lifting $\mathcal{T}^{\top\top}$ with R, S. Using Proposition 10, it is given as: $[S]^R X = \bigwedge_{f \in \mathbb{E}(X,S)} (f^\#)^{-1}(S)$.

▶ **Definition 22.** Let $X \in \mathbb{E}$. An object $S \in \mathbb{E}_{T(pX)}$ is *closed* with respect to X if 1) $\eta_{pX} \in \mathbb{E}(X, S)$ and 2) for all $f \in \mathbb{E}(X, S)$, we have $f^\# \in \mathbb{E}(S, S)$.

▶ **Proposition 23.** *Let $X \in \mathbb{E}$. Then $S \in \mathbb{E}_{T(pX)}$ is closed with respect to X if and only if $S = [S]^{pX} X$.*

▶ **Definition 24.** We define $\mathbf{Cls}(\mathcal{T}, X)$ to be the subposet $(\{S \mid S = [S]^{pX} X\}, \leq)$ of $\mathbb{E}_{T(pX)}$ consisting of closed objects with respect to X. We also define the following mappings:

$$\begin{tikzcd} \mathbf{Cls}(\mathcal{T}, X) \arrow[r, shift left, "{[-]^{pX}}"] & \mathbf{lift}(\mathcal{T}) \arrow[l, shift left, "q_X"] \end{tikzcd}, \quad [S]^{pX} = T^{\top\top(pX,S)}, \quad q_X(\dot{T}) = \dot{T}X.$$

The mapping q_X is monotone, while $[-]^{pX}$ is *not*, because its argument is used both in positive and negative way. Still, we have the following adjoint-like relationship:

▶ **Theorem 25.** *For each $X \in \mathbb{E}$, we have $q_X \circ [-]^{pX} = \mathrm{id}_{\mathbf{Cls}(\mathcal{T},X)}$ and $\mathrm{id}_{\mathbf{lift}(\mathcal{T})} \preceq [-]^{pX} \circ q_X$.*

We define a function $\phi_{X,Y} : \mathbf{Cls}(\mathcal{T}, X) \to \mathbf{Cls}(\mathcal{T}, Y)$ by $\phi_{X,Y}(S) = q_Y([S]^{pX}) = [S]^{pX} Y$. This is not monotone. Theorem 25 asserts that $\phi_{X,X} = \mathrm{id}_{\mathbf{Cls}(\mathcal{T},X)}$. Using the second inequality of Theorem 25, for each $X, Y \in \mathbb{E}$, we also have

$$[S]^{pX} \preceq [[S]^{pX} Y]^{pY} = [\phi_{X,Y}(S)]^{pY}. \tag{3}$$

From Theorem 25, $\dot{T}$ is a lower bound of the class $\{[q_X(\dot{T})]^{pX} \mid X \in \mathbb{E}\}$. In fact, $\dot{T}$ is the *greatest* lower bound:

▶ **Theorem 26.** *For any lifting $\dot{\mathcal{T}}$ of $\mathcal{T}$, we have $\dot{T} = \bigwedge_{X \in \mathbb{E}} [q_X(\dot{T})]^{pX}$.*

▶ **Definition 27.** We say that $X \in \mathbb{E}$ is a *split subobject* of $Y \in \mathbb{E}$, (denoted by $X \lhd Y$) if there is a split monomorphism $m : X \to Y$.

▶ **Lemma 28.** *Let $X \lhd Y$ in $\mathbb{E}$. The following holds: 1) $\phi_{Y,X} \circ q_Y = q_X$. 2) For any $Z \in \mathbb{E}$, $\phi_{Y,X} \circ \phi_{Z,Y} = \phi_{Z,X}$. 3) $[q_Y(\dot{T})]^{pY} \preceq [q_X(\dot{T})]^{pX}$.*

Let us write $\mathbf{Split}(\mathbb{E})$ for the large preorder of $\mathbb{E}$-objects ordered by $\lhd$. We extend $\mathbf{Cls}(\mathcal{T}, -)$ to a functor of type $\mathbf{Split}(\mathbb{E})^{op} \to \mathbf{Pre}$ by $\mathbf{Cls}(\mathcal{T}, X \lhd Y) = \phi_{Y,X} : \mathbf{Cls}(\mathcal{T}, Y) \to \mathbf{Cls}(\mathcal{T}, X)$. This is indeed a functor thanks to Theorem 25 (for identity) and Lemma 28-2 (for composition). Moreover, $q_X : \mathbf{lift}(\mathcal{T}) \to \mathbf{Cls}(\mathcal{T}, X)$ is a large cone over the diagram $\mathbf{Cls}(\mathcal{T}, -)$ by Lemma 28-1. When $\mathbf{Split}(\mathbb{E})$ is directed, q is a limiting cone.

▶ **Theorem 29.** *Suppose that $\mathbf{Split}(\mathbb{E})$ is directed. Then the cone $q_X : \mathbf{lift}(\mathcal{T}) \to \mathbf{Cls}(\mathcal{T}, X)$ over the large diagram $\mathbf{Cls}(\mathcal{T}, -)$ is limiting.*

7 Conclusion and Future Work

We introduced the codensity lifting of monads along the fibrations that preserve the right Kan extensions giving codensity monads (this codensity condition was relaxed later in Section 5). The codensity lifting allows us to lift various monads on non-closed base / total categories, which was not possible by the previous $\top\top$-lifting [6].

Theorem 29 is an analogue of the characterisation of the collection of preorders on a **Set**-monad as a limit of $\mathbf{Card}^{op}$-chain in [8]. There we exploited this characterisation to enumerate the collection of preorders on some monads. We are wondering whether Theorem 29 is also useful to identify all the liftings of a given monad $\mathcal{T}$.

Acknowledgement. The derivation of Kantorovich metric by the codensity lifting of Giry monad in Section 3.4 was constructed after the first author learned Ogawa's work on deriving Kantorovich metric for finitely-supported subdistributions using observational algebra [13]. The first author is grateful to him for the discussion about Kantorovich metric and pseudometric spaces in CSCAT 2015. This paper was supported by JSPS KAKENHI Grant Number 24700012.

References

1 P. Baldan, F. Bonchi, H. Kerstan, and B. König. Behavioral Metrics via Functor Lifting. In *Proc. FSTTCS 2014*, volume 29 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 403–415, Dagstuhl, Germany, 2014. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

2 M. Giry. A categorical approach to probability theory. In B. Banaschewski, editor, *Categorical Aspects of Topology and Analysis*, volume 915 of *LNM*, pages 68–85. Springer, 1982.

3 C. Hermida. *Fibrations, Logical Predicates and Indeterminants.* PhD thesis, University of Edinburgh, 1993.

4 B. Jacobs. *Categorical Logic and Type Theory.* Elsevier, 1999.

5 L. Kantorovich. On the transfer of masses (in russian). *Doklady Akademii Nauk*, 5(1), October 1942. Translated in Management Science, 5(1):1–4, 1958.

6 S. Katsumata. A semantic formulation of ⊤⊤-lifting and logical predicates for computational metalanguage. In *Proc. CSL'05*, volume 3634 of *LNCS*, pages 87–102. Springer, 2005.

7 S. Katsumata. Relating computational effects by ⊤⊤-lifting. *Inf. Comput.*, 222:228–246, 2013.

8 S. Katsumata and T. Sato. Preorders on monads and coalgebraic simulations. In *Proc. FOSSACS 2013*, volume 7794 of *LNCS*, pages 145–160, 2013.

9 S. Lindley. *Normalisation by Evaluation in the Compilation of Typed Functional Programming Languages.* PhD thesis, University of Edinburgh, 2005.

10 S. Lindley and I. Stark. Reducibility and ⊤⊤-lifting for computation types. In *Proc. TLCA 2005*, volume 3461 of *LNCS*, pages 262–277, 2005.

11 S. MacLane. *Categories for the Working Mathematician (Second Edition)*, volume 5 of *Graduate Texts in Mathematics.* Springer, 1998.

12 J. Mitchell and A. Scedrov. Notes on sconing and relators. In *Proc. CSL'92*, volume 702 of *LNCS*, pages 352–378. Springer, 1993.

13 H. Ogawa. Quotient and Kantorovich metric via observational-algebra in Lawvere theory. Oral Presentation in CSCAT 2015, Kagoshima University, Japan, Mar 14, 2015.

14 G. D. Plotkin and J. Power. Algebraic operations and generic effects. *Applied Categorical Structures*, 11(1):69–94, 2003.

15 F. van Breugel, M. W. Mislove, J. Ouaknine, and J. Worrell. Domain theory, testing and simulation for labelled markov processes. *Theor. Comput. Sci.*, 333(1-2):171–197, 2005.

A First-order Logic for String Diagrams

Aleks Kissinger and David Quick

Department of Computer Science
University of Oxford, UK
`{aleks.kissinger,david.quick}@cs.ox.ac.uk`

Abstract

Equational reasoning with string diagrams provides an intuitive means of proving equations between morphisms in a symmetric monoidal category. This can be extended to proofs of infinite families of equations using a simple graphical syntax called !-box notation. While this does greatly increase the proving power of string diagrams, previous attempts to go beyond equational reasoning have been largely ad hoc, owing to the lack of a suitable logical framework for diagrammatic proofs involving !-boxes. In this paper, we extend equational reasoning with !-boxes to a fully-fledged first order logic with conjunction, implication, and universal quantification over !-boxes. This logic, called !L, is then rich enough to properly formalise an induction principle for !-boxes. We then build a standard model for !L and give an example proof of a theorem for non-commutative bialgebras using !L, which is unobtainable by equational reasoning alone.

1998 ACM Subject Classification D.3.1 Formal Definitions and Theory, F.4.1 Mathematical Logic

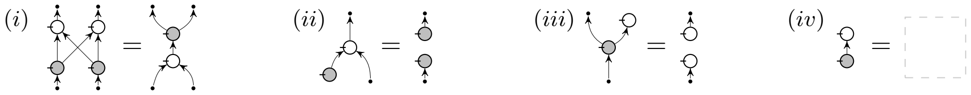
Keywords and phrases string diagrams, compact closed monoidal categories, abstract tensor systems, first-order logic

Digital Object Identifier 10.4230/LIPIcs.CALCO.2015.171

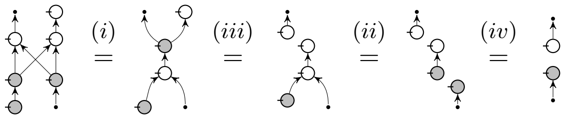
1 Introduction

Many processes come with natural notions of parallel and sequential composition. In such cases, it is advantageous to switch from traditional term-based (i.e. one-dimensional) syntax to the two-dimensional syntax of string diagrams. This diagrams, which consist of boxes (or various other shapes) connected by wires, form a sound and complete language for compositions of morphisms in a monoidal category [8]. Recently, the use of string diagrams has gained much interest in a wide variety of areas, including categorical quantum mechanics [4, 3, 5], computational linguistics [9] and control theory [2, 1].

What many of these applications have in common is they make extensive use of equational reasoning for string diagrams. That is, proofs are constructed by starting with a fixed set of diagram equations, e.g.

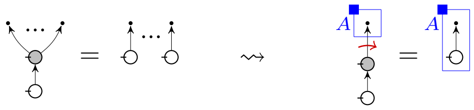


and using those to construct new equations by substitution of sub-diagrams. For example, the following is a derivation making use of the four rules above:



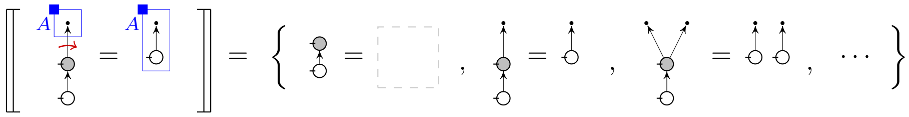
6th International Conference on Algebra and Coalgebra in Computer Science (CALCO'15).
Editors: Lawrence S. Moss and Pawel Sobocinski; pp. 171–189
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

However, to prove more powerful theorems, one often needs to pass from statements about single diagrams to entire families of diagrams and diagram equations. One way to do this, while staying within the realm of string diagrams is to use !-box notation (pronounced 'bang-box notation'), introduced in [6] and formalised in [11]. In this notation, certain sub-diagrams are wrapped in boxes, which mean 'repeat this sub-diagram any number of times'. For example, suppose we considered a family of 'copy' operations with 1 input and n outputs. Then, if we had some other map with just a single output, we might ask that connecting it to the n-fold 'copy' results in n copies. We can represent this family of rules using !-box notation as follows:

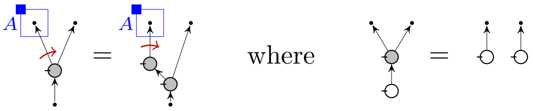


(1)

Whereas the equation on the left is informal, the expression on the right defines a family of equations without ambiguity. Formally, a !-box rule represents a set of string diagram rules obtained by *instantiating* the !-box, which essentially amounts fixing the number of times to copy each !-box. For example, the instances of the !-box rule above are precisely the ones we meant to capture with the informal expression:

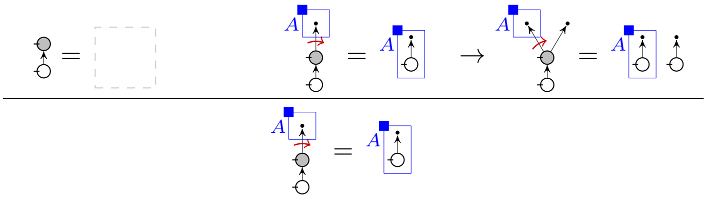


where the 'blank space' in the first equation represents the monoidal unit. We can even use this more expressive notation to make recursive definitions. For instance, we could recursively define the n-fold copy operation as a tree of binary copy operations:



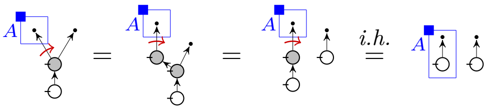
(2)

Using just equational reasoning, there is no way to get from the equations in (2) to the n-fold copy equation (1). However, if we introduce an induction principle:



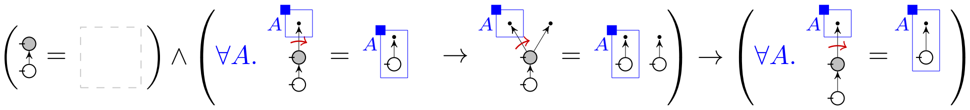
(Induct)

we can split into a base case (zero copies of the !-box) and a step case (n copies implies $n+1$ copies). Taking the base case as given, we can prove the step case using the induction hypothesis and the rules in (2):



Unfortunately, this doesn't quite work. If we interpret → to mean 'the rule on the left can be used in the proof of the rule on the right', the step case is vacuous. The rule on the right is *already* an instance of the rule on the left. This is a bit like saying: $(\forall n.Pn) \to (\forall n.P(n+1))$, which is of course true for any P.

The problem is, when we pass to !-box notation, where single diagram rules now represent whole families of rules, our existing reasoning tools do not provide enough control over instances of rules, and how those instances interact with each other. This problem was solved for the specific case of induction in [15] using an operation called *fixing*, which essentially freezes a !-box so it can't be instantiated. However, this was introduced more as a stopgap, until a proper logic could be developed, suitable for handling conjunction, implication, and crucially universal quantification over !-boxes. In this paper, we develop that logic. With this new *!-logic* in hand, we can correct our failed attempt at induction to:



In addition to giving a solid foundation for proofs constructed using !-boxes, a major motivating factor for the development of a formal logic of !-boxes is its implementation in the proof assistant Quantomatic [14]. Currently, Quantomatic supports pure equational reasoning on string diagrams with !-boxes. The implementation of !-logic will allow it to support diagrammatic versions of all the usual trappings of a fully-featured proof assistant, such as local assumptions, goal-driven (i.e. backward) reasoning, and of course inductive proofs.

There are two essentially equivalent ways to formalise string diagrams with !-boxes: one combinatoric (as in the original formulation) and one syntactic, building on the *tensor notation* for compact closed categories. Here we opt for the latter, as it more conveniently fits into the presentation of the logic and provides a means of elegantly representing commutative *and non-commutative* generators. Equational reasoning using *!-tensor* notation was presented in [12] allowing some basic rules which were shown to be sound in the extended version [13]. In this paper we begin by reviewing compact closed categories, tensor notation, and !-tensors in Section 2. Next, we define the concept of an instantiation, which will play a central role in the logic in Section 3. We introduce the syntax of our logic, namely *!-formulas*, in Section 4 and give the rules of the logic in Section 5. We provide a semantics for !-formulas based on sets of instantiations evaluated in a compact closed category $\mathcal{C}$ in Section 6. We conclude by exhibiting a non-trivial proof involving non-commutative bialgebras, which can be done entirely within !L and diagram rewriting.

2 Preliminaries

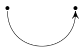
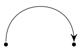
2.1 Compact closed categories and signatures

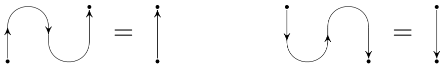
Throughout this paper, we will work with *compact closed categories*, i.e. symmetric monoidal categories where every object X has a *dual* object X^* and two morphisms $\eta_X : I \to X^* \otimes X$, $\epsilon_X : X \otimes X^* \to I$ satisfying the *yanking equations*:

$$(\epsilon_X \otimes 1_X) \circ (1_X \otimes \eta_X) = 1_X \qquad\qquad (1_{X^*} \otimes \epsilon_X) \circ (\eta_X \otimes 1_{X^*}) = 1_{X^*}$$

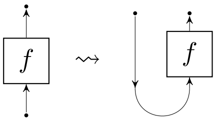
For simplicity, we will focus on *strict* compact closed categories, where associativity and unitality of $\otimes$ hold on-the-nose. However, all of the concepts we will use in this paper go through virtually unmodified by Mac Lane's coherence theorem.

As string diagrams, we will depict X as a wire directed upwards, and X^* as a wire directed downwards. Thus η_X and ϵ_X can be depicted as half-turns:

η_X =  ϵ_X =  which we typically call 'cups' and 'caps', respectively. Using this notation, the yanking equations resemble their namesake:



One consequence of the inclusion of cups and caps is that we can now introduce 'feedback loops', allowing us to make sense of arbitrary string diagrams, not just directed acyclic ones. A second consequence is that any map $f : X \to Y$ can be equivalently represented as a map of the form $\widetilde{f} : I \to X^* \otimes Y$ just by 'bending' the input up to be an output:



Thus, we will always assume that our generating morphisms can be written in the form $\phi : I \to X_1 \otimes X_2 \otimes \ldots \otimes X_n$ for objects $X_1, X_2, \ldots, X_n$. A morphism whose domain is the monoidal unit is called a *point*.

▶ **Definition 1.** A *compact closed signature* Σ consists of a set $\mathcal{O} := \{x, y, \ldots\}$ and a set $\mathcal{M}$ of pairs (ψ, w), where w is a word in $\{x, x^*, y, y^*, \ldots\}$. If ψ occurs precisely once in $\mathcal{M}$, it is said to have *fixed arity*, otherwise it has *variable arity*.

For simplicity, we will assume every generator in Σ is defined for every arity. This can be avoided if we add extra conditions to Definition 6 to ensure we never get 'undefined' generators, but for our purposes, this won't be necessary.

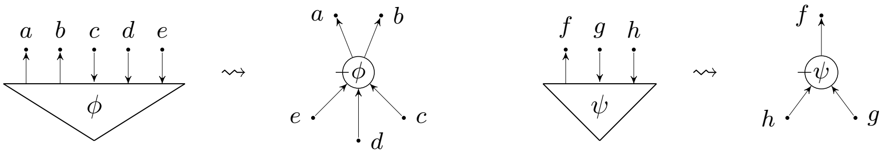
▶ **Definition 2.** For a compact closed category $\mathcal{C}$, a *valuation* $[\![-]\!] : \Sigma \to \mathcal{C}$ is a choice of object $X \in \text{ob}(\mathcal{C})$ for every $x \in \mathcal{O}$, and a choice of point $[\![\psi]\!] : I \to X_1 \otimes X_2^* \otimes \ldots \otimes X_n$ for every $(\psi, x_1 x_2^* \ldots x_n) \in \mathcal{M}$.

When there can be no confusion, we write pairs $(\psi, x_1 x_2^* \ldots x_n)$ also as $\psi : I \to X_1 \otimes X_2^* \otimes \ldots X_n$. As usual , the free compact closed category Free(Σ) is characterised by the universal property that any valuation lifts uniquely to functor $[\![-]\!] : \text{Free}(\Sigma) \to \mathcal{C}$ preserving all of the compact closed structure [12]. In the next section, we will give a convenient syntactic presentation of this category.

2.2 Tensor notation for compact closed categories

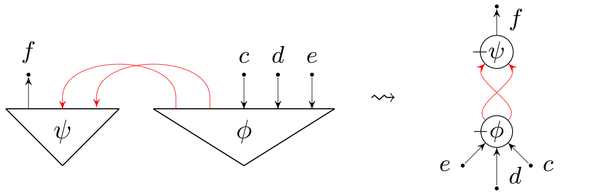
From now on, we will assume that Σ only has one object X, so morphisms will be maps from I to monoidal products of X and X^*.

Suppose that we have two generators in Σ, $\phi : I \to X \otimes X \otimes X^* \otimes X^* \otimes X^*$ and $\psi : I \to X \otimes X^* \otimes X^*$. Diagrammatically we will depict these generators as circular nodes with the edges ordered clockwise around the node. To avoid ambiguity we place a tick on the node between the last and first edge. We will name free edges so they can be referred to when manipulating diagrams. Hence the generators in our example (with arbitrarily named edges) are:



(3)

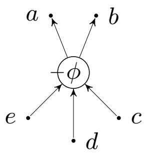
Now, wires connecting these dots indicate the presence of caps:



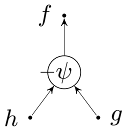
(4)

To succinctly express these kinds of string diagrams syntactically, we can use *tensor notation*. Here, we represent generators by writing their names, followed by a list of subscripts indicating their (named) inputs and outputs:

$\phi_{\hat a\hat b\check c\check d\check e} :=$

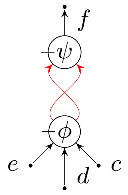


$\psi_{\hat f\check g\check h} :=$



Inputs (i.e. outputs of type X^*) are represented as names with 'checks' $\check a, \check b, \dots$, whereas outputs are represented as names with 'hats' $\hat a, \hat b, \dots$. We combine generators into a single diagram by concatenating them, and the process of connecting generators together by caps—which we call *contraction*—is indicated by repeating names:

$\psi_{\hat f\check a\check b}\phi_{\hat a\hat b\check c\check d\check e} :=$



(5)

If a name occurs once, it is called a *free edgename*. If it is repeated, it is called a *bound edgename*. As the name would suggest, bound edgenames have no meaning in their own right, and can be changed (a.k.a. α-converted) at will. Hence the expressions $\psi_{\hat f\check a\check b}\phi_{\hat a\hat b\check c\check d\check e}$ and $\phi_{\hat g\hat h\check c\check d\check e}\psi_{\hat f\check g\check h}$ both represent (5). Also, since it is the names that indicate inputs/outputs of a tensor expression, the order in which we write tensor symbols is irrelevant. So, for example, $\psi_{\hat f\check a\check b}\phi_{\hat a\hat b\check c\check d\check e} = \phi_{\hat a\hat b\check c\check d\check e}\psi_{\hat f\check a\check b}$.

This notation gives a simple presentation of string diagrams, and hence of morphisms in the free compact closed category over Σ. The only mismatch between tensors and morphisms in the free category is that tensors use *names* to identify inputs/outputs, whereas categories use *positions*. Thus, to relate the two concepts, we assume the set of edgenames contains two disjoint sets $\{a_1, a_2, \dots\}$ and $\{b_1, b_2, \dots\}$ that are totally ordered and (countably) infinite, and introduce the notion of *canonically named tensors*.

▶ **Definition 3.** A tensor is *canonically named* if its free names are $a_1, \dots a_m, b_1, \dots, b_n$ for some $m, n \geq 0$.

We can then express a morphism in Free(Σ) as a tensor whose i-th input is named a_i and whose j-th output is named b_j. It was shown in [10] (for the traced case) and [13] (for the compact closed case) that Free(Σ) is equivalent to the category whose morphisms are canonically-named tensors, with $\circ$ and $\otimes$ defined in the obvious way using renaming and contraction. This gives us an important consequence:

▶ **Theorem 4.** *For any compact closed signature Σ, a valuation $[\![-]\!] : \Sigma \to \mathcal{C}$ lifts uniquely to an operation which sends canonically named tensors G over Σ to morphisms $[\![G]\!]$ in $\mathcal{C}$.*

2.3 !-tensors

As mentioned in the intro, a string diagram with !-boxes represents a family of string diagrams, where the sub-diagram in the !-box has been copied an arbitrary number of times. To formalise this, we extend the tensor syntax to include !-boxes. These extended expressions are called !-tensors. Fix disjoint, infinite sets $\mathcal{E}$ and $\mathcal{B}$ of *edgenames* and *boxnames*, respectively.

▶ **Definition 5.** The set of *edgeterms* $\mathcal{T}_e$ is defined inductively as follows:

- $\epsilon \in \mathcal{T}_e$ (empty edgeterm)
- $\check{a}, \hat{a} \in \mathcal{T}_e$ $a \in \mathcal{E}$
- $\langle e]^A, [e\rangle^A \in \mathcal{T}_e$ $e \in \mathcal{T}_e,\ A \in \mathcal{B}$
- $ef \in \mathcal{T}_e$ $e, f \in \mathcal{T}_e$

Letting 1 represent the empty !-tensor and $1_{\hat{a}\check{b}}$ represent an identity edge with input named b and output named a, we can define !-tensor expressions as follows:

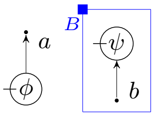
▶ **Definition 6.** The set of all !-tensor expressions $\mathcal{T}_\Sigma$ for a signature Σ is defined inductively as:

- $1, 1_{\hat{a}\check{b}} \in \mathcal{T}_\Sigma$ $a, b \in \mathcal{E}$
- $\phi_e \in \mathcal{T}_\Sigma$ $e \in \mathcal{T}_e, \phi \in \Sigma$
- $[G]^A \in \mathcal{T}_\Sigma$ $G \in \mathcal{T}_\Sigma,\ A \in \mathcal{B}$
- $GH \in \mathcal{T}_\Sigma$ $G, H \in \mathcal{T}_\Sigma$

Subject to the conditions that (F1) $\check{a}$ and $\hat{a}$ must occur at most once for each edgename a and (F2) $[\ldots]^A$ must occur at most once for each boxname A, as well as consistency conditons (C1)–(C3) for !-boxes given in [13].

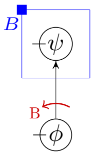
We omit the formal statement of (C1)-(C3) here, as they are easiest to understand in the graphical presentation of !-tensors. Sub-expressions of the form $[\ldots]^A$ are represented by wrapping a box around part of the string diagram:

$\phi_{\hat{a}}[\psi_{\check{b}}]^B \ :=$

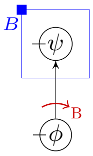


Edges connecting into or out of a !-box must be annotated with the !-box name and a direction, indicating whether the new edgenames should be produced to the left (anticlockwise) or to the right (clockwise) when a !-box is expanded. We indicate this direction by drawing an arc over the annotated edges:

$\phi_{\langle\hat{a}]^B}[\psi_{\check{a}}]^B \ :=$

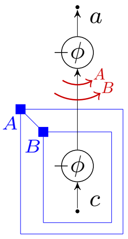


vs. $\phi_{[\hat{a}\rangle^B}[\psi_{\check{a}}]^B \ :=$

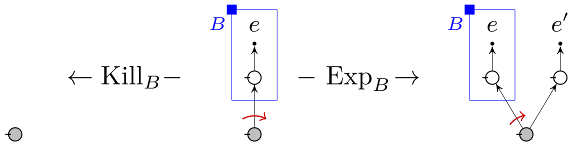


We drop the label on the arc when it can be inferred from context. The remaining consistency conditions say that any edge connecting into or out of a !-box must have an annotation, and those annotations should respect nesting of !-boxes, as in e.g.:

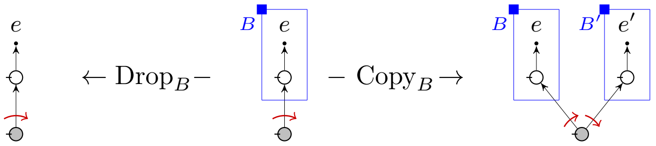
$$\phi_{\hat{a}\langle\langle\check{b}]^B]^A}[[\phi_{\hat{b}\check{c}}]^B]^A \quad :=$$



For a fully rigorous account of these conditions, see [13]. However, the above description should suffice for the purposes of this paper, so we'll proceed to how !-tensors are instantiated. The primary instantiation operations are *expand*, which produces a new copy of the contents of a !-box and *kill*, which removes the !-box from the diagram:



These two operations suffice to produce all *concrete instances*, that is all instances not involving any !-boxes, of a !-tensor. If we wish to get *all* instances of a !-tensor, including those with !-boxes, we factorise expand into two additional operations: *copy*, which makes a copy of the !-box and its contents, and *drop*, which removes a !-box and leaves its contents behind:



We can define all four of these operations recursively on !-tensor expressions. We first give the recursive cases where all four operations behave the same:

$$\begin{aligned}
\mathrm{Op}_B(GH) &:= \mathrm{Op}_B(G)\,\mathrm{Op}_B(H) & \mathrm{Op}_B(ef) &:= \mathrm{Op}_B(e)\,\mathrm{Op}_B(f)\\
\mathrm{Op}_B([G]^A) &:= [\mathrm{Op}_B(G)]^A & \mathrm{Op}_B([e\rangle^A) &:= [\mathrm{Op}_B(e)\rangle^A\\
\mathrm{Op}_B(\phi_e) &:= \phi_{\mathrm{Op}_B(e)} & \mathrm{Op}_B(\langle e]^A) &:= \langle\mathrm{Op}_B(e)]^A\\
\mathrm{Op}_B(x) &:= x
\end{aligned}$$

where $A \neq B$ and $x \in \{1, 1_{\hat{a}\check{b}}, \check{a}, \hat{a}, \epsilon\}$. The four operations are distinguished on the remaining three cases:

$$\begin{aligned}
\mathrm{Exp}_B([G]^B) &:= [G]^B\,\mathbf{fr}(G) & \mathrm{Kill}_B([G]^B) &:= 1\\
\mathrm{Exp}_B([e\rangle^B) &:= [e\rangle^B\,\mathbf{fr}(e) & \mathrm{Kill}_B([e\rangle^B) &:= \epsilon\\
\mathrm{Exp}_B(\langle e]^B) &:= \mathbf{fr}(e)\langle e]^B & \mathrm{Kill}_B(\langle e]^B) &:= \epsilon
\end{aligned}$$

$$\begin{aligned}
\mathrm{Copy}_B([G]^B) &:= [G]^B[\mathbf{fr}(G)]^{\mathbf{fr}(B)} & \mathrm{Drop}_B([G]^B) &:= G\\
\mathrm{Copy}_B([e\rangle^B) &:= [e\rangle^B[\mathbf{fr}(e)\rangle^{\mathbf{fr}(B)} & \mathrm{Drop}_B([e\rangle^B) &:= e\\
\mathrm{Copy}_B(\langle e]^B) &:= \langle\mathbf{fr}(e)]^{\mathbf{fr}(B)}\langle e]^B & \mathrm{Drop}_B(\langle e]^B) &:= e
\end{aligned}$$

Where $\mathbf{fr}$ is a function assigning fresh names to all edges and !-boxes in an expression. We occasionally write $\text{Exp}_{B,\mathbf{fr}}$ and $\text{Copy}_{B,\mathbf{fr}}$ to explicitly reference the freshness function of a !-box operation. Note that if B is not contained in G, each of these operations will leave G unchanged.

3 Compatibility and instantiations of !-boxes

In Section 4, we will define the formulas of !-logic. It only makes sense to combine !-tensors into single formulas if their !-boxes are compatible in some sense, so we first provide some basic notions relating to compatibility.

▶ **Definition 7.** If F is a set and $\prec$ is a binary relation on F then the pair $(F, \prec)$ is called a *forest* if it forms a cycle-free directed graph where each node A has at most one node B s.t $A \prec B$. A forest can also be seen as a graph made up of disconnected directed trees. We write $<$ for the transitive closure and $\leq$ for the reflexive and transitive closure of $\prec$.

Let $\downarrow X$ and $\uparrow X$ be the downward and upward closure of $X \subseteq F$, respectively. For a single element $A \in F$, we write $\downarrow A$ for $\downarrow\{A\}$.

▶ **Definition 8.** If a subset $X \subseteq F$ is both upward and downward closed (i.e. $X =\downarrow X =\uparrow X$) then we say X is a *component* of $(F, \prec)$. If it contains no proper sub-components, it is called a *connected component*.

We write $F^{\top} \subseteq F$ for the set of maximal elements with respect to $\leq$. Note that for $A \in F^{\top}$ the set $\downarrow A$ is always a connected component, and for F finite, all connected components are of this form.

▶ **Definition 9.** Two forests F, F' are said to be *compatible*, written $F \triangle F'$, if the intersection $F \cap F'$ is a (possibly empty) component of both F and F'.

Equivalently, F, F' are compatible if and only if there exist forests X, Y, Z such that $F = X \uplus Y$ and $G = Y \uplus Z$. As a consequence, the union of compatible forests is always well-defined ($F \cup F' := X \uplus Y \uplus Z$), and itself a forest. For any !-tensor, we can always associate a forest of !-boxes:

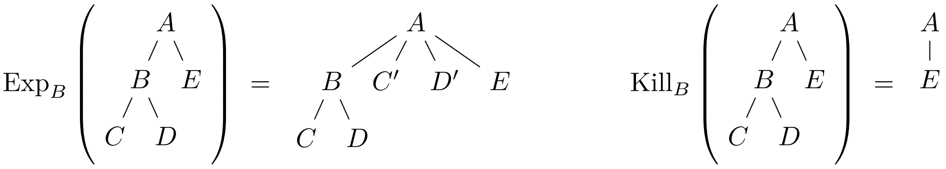
▶ **Definition 10.** For a !-tensor G, let $(\text{Boxes}(G), \prec_G)$ be the forest of !-boxes in G, where $A \prec_G B$ iff A is a direct descendent of B. That is, A is nested inside of B with no intervening !-boxes.

An important concept for !-tensors is that of *instantiations*. These capture precisely the sequence of operations by which a !-tensor is transformed into some instance of itself. For a !-tensor G, an *instantiation* i of G is a sequence of zero or more Exp and Kill operations such that $i(G)$ doesn't contain any !-boxes.

In fact, we can divorce the notion of instantiation from a particular !-tensor if we notice that instantiations make sense for any forest. For a forest F, define the Exp_B and Kill_B operations as identity maps if $B \notin F$ and else as follows:

$$\text{Exp}_B(F) := F \cup \mathbf{fr}(\downarrow B \setminus B) \qquad\qquad \text{Kill}_B(F) := F \setminus \downarrow B$$

where the top elements of $\mathbf{fr}(\downarrow B \setminus B)$ are added as descendants of the parent of B (if it has one). So, Kill_B removes B and all of its children, whereas Exp_B behaves just like expanding a !-box, in that it adds a fresh copy of all of the children as siblings:



We can now define instantiations in a way that only refers to forests:

▶ **Definition 11.** For a forest F, an *instantiation* of F is a composition i of zero or more operations Exp_B, Kill_B such that B is in the domain of each operation and $i(F) = \{\}$. Let $\mathrm{Inst}(F)$ be the set of all instantiations of F.

In particular, if F is empty, $\mathrm{Inst}(F)$ only contains the trivial instantiation 1. Note that for *any* instantiation $i \in \mathrm{Inst}(F)$ where $F \triangle \mathrm{Boxes}(G)$, $i(G)$ is a well defined !-tensor. This added flexibility will be important to the interpretation of !-logic formulas, where instantiations may act on many !-tensors simultaneously.

4 !-logic formulas

In this section, we will introduce the syntax of !-logic. The atomic !-logic formulas are well-formed equations between !-tensors and generic formulas are built up from the atomic formulas using conjunction, implication, and universal quantification. There does not appear to be any obstacle to adding negation (and hence existential quantification) to !-logic formulae but no application has currently been found by the authors.

Well-formed !-tensor equations are pairs of !-tensors with the property that any simultaneous instantiation of the LHS and RHS produces a valid equation between tensors. That is, the LHS and the RHS of any instance of the equation should have identical free edgenames for their inputs and outputs.

▶ **Definition 12.** A !-tensor equation $G = H$ is *well-formed* if G and H have identical inputs and outputs, $\mathrm{Boxes}(G) \triangle \mathrm{Boxes}(H)$, and an input $\check{a}$ (resp. output $\hat{a}$) occurs in a !-box A in G iff it occurs in the same !-box in H.

Note that by '$\check{a}$ occurs in A' we mean $\check{a}$ occurs as a sub-expression of $[\ldots]^A$, $\langle\ldots]^A$ or $[\ldots\rangle^A$. Also note that we only require the !-boxes on the LHS and RHS be compatible, rather than identical. This is unproblematic, since as mentioned at the end of Section 2.3, operations on !-boxes that are missing from the LHS or the RHS will simply be ignored.

The other formulas are built inductively, while maintaining the property that the subformulas have compatible !-boxes. To accomplish this, it is most convenient to define the set of !-formulas while simultaneously defining the operation $\mathrm{Boxes}(X)$ for any !-formula X.

▶ **Definition 13.** The set of *!-formulas*, $\mathcal{F}_\Sigma$, for a signature Σ is defined inductively as:

- $G = H \in \mathcal{F}_\Sigma$ $\qquad G, H \in \mathcal{T}_\Sigma,\ G = H$ well-formed
- $X \wedge Y \in \mathcal{F}_\Sigma$ $\qquad X, Y \in \mathcal{F}_\Sigma,\ \mathrm{Boxes}(X) \triangle \mathrm{Boxes}(Y)$
- $X \to Y \in \mathcal{F}_\Sigma$ $\qquad X, Y \in \mathcal{F}_\Sigma,\ \mathrm{Boxes}(X) \triangle \mathrm{Boxes}(Y)$
- $\forall A.\ X \in \mathcal{F}_\Sigma$ $\qquad X \in \mathcal{F}_\Sigma,\ A \in \mathrm{Boxes}(X)^\top$

where $\mathrm{Boxes}(-)$ is defined recursively on !-formulas by:

- $\text{Boxes}(G = H) := \text{Boxes}(G) \cup \text{Boxes}(H)$
- $\text{Boxes}(X \wedge Y) := \text{Boxes}(X) \cup \text{Boxes}(Y)$
- $\text{Boxes}(X \to Y) := \text{Boxes}(X) \cup \text{Boxes}(Y)$
- $\text{Boxes}(\forall A.\ X) := \text{Boxes}(X) \setminus \downarrow A$

Just like one can read formulas in predicate logic as mappings from values of the free variables to truth values, one should read !-formulas as mappings from *instantiations of !-boxes* to truth values. Thus, universal quantification over !-boxes states that a particular formula holds for *all instantiations* involving those !-boxes. We will make this interpretation precise in Section 6.

One important thing to note is that universal quantification over a top-level !-box A should be interpreted as quantifying over the entire *connected component* $\downarrow A$. In the absence of nesting, this is the same as quantifying over individual !-boxes. However, in the presence of nesting, this restriction to only quantifying over entire components seems to be necessary for giving a consistent interpretation to !-logic formulas. This boils down to the fact that !-box operations on separate components of $\text{Boxes}(X)$ commute, whereas arbitrary !-box operations do not.

▶ Remark. Note that the set $\mathcal{F}_\Sigma$ in Definition 13 is defined inductively by relying on a simultaneous recursive definition of Boxes. This is non-circular, since the inductive steps always rely on calls to Boxes on strictly smaller formulas. Unsurprisingly, this style of definition is called *induction-recursion* [7].

In order to talk about instances of !-formulas, we must extend !-box operations from !-tensors to arbitrary formulas.

▶ **Definition 14.** For Op_B one of the operations $\text{Kill}_B, \text{Exp}_{B,\mathbf{fr}}, \text{Copy}_{B,\mathbf{fr}}, \text{Drop}_B$:

- $\text{Op}_B(G = H) := \text{Op}_B(G) = \text{Op}_B(H)$
- $\text{Op}_B(X \wedge Y) := \text{Op}_B(X) \wedge \text{Op}_B(Y)$
- $\text{Op}_B(X \to Y) := \text{Op}_B(X) \to \text{Op}_B(Y)$
- $\text{Op}_B(\forall A.\ X) := \begin{cases} \forall A.\ X & B \in \downarrow A \\ \forall A.\ \text{Op}_B(X) & B \notin \downarrow A \end{cases}$

▶ **Theorem 15.** *!-box operations preserve the property of being a formula.*

Proof. We prove this using structural induction on !-formulas.

- If $G = H$ is a formula then G and H have the same free edges in the same !-boxes. Hence $\text{Op}_B(G)$ and $\text{Op}_B(H)$ have the same free edges (a or $\mathbf{fr}(a)$ for a free in $G = H$) and these are in the same !-boxes.
- For the next two cases we have $\text{Boxes}(X)$ and $\text{Boxes}(Y)$ compatible. Op_B takes the unique connected component, S, containing B and replaces it with $\text{Op}_B(S)$. This can only have gained fresh !-box names so $\text{Boxes}(\text{Op}_B(X))$ and $\text{Boxes}(\text{Op}_B(Y))$ are still compatible.
- If $B \in \downarrow A$ then the final case is trivial. If $B \notin \downarrow A$ then the component $\downarrow A$ is not affected by Op_B so is still a component of $\text{Op}_B(X)$. ◀

5 The rules of !L

We now define a simple logic over !-formulas, which we call !L. Our presentation is given in terms of sequents of the form: $\Gamma \vdash Y$, where $\Gamma := X_1, X_2, \ldots, X_n$ is a finite sequence of

!-formulas. We will always assume in writing a sequent that all of the formulas involved have compatible !-boxes. We take the core logical rules to be those from positive intuitionistic logic with cut:

$$\frac{}{X \vdash X}\,(\text{Ident}) \qquad \frac{\Gamma \vdash Y}{\Gamma, X \vdash Y}\,(\text{Weaken}) \qquad \frac{\Gamma, X, Y, \Delta \vdash Z}{\Gamma, Y, X, \Delta \vdash Z}\,(\text{Perm}) \qquad \frac{\Gamma, X, X \vdash Y}{\Gamma, X \vdash Y}\,(\text{Contr})$$

$$\frac{\Gamma \vdash X \qquad \Delta \vdash Y}{\Gamma, \Delta \vdash X \wedge Y}\,(\wedge I) \qquad \frac{\Gamma \vdash X \wedge Y}{\Gamma \vdash X}\,(\wedge E_1) \qquad \frac{\Gamma \vdash X \wedge Y}{\Gamma \vdash Y}\,(\wedge E_2)$$

$$\frac{\Gamma \vdash X \to Y}{\Gamma, X \vdash Y}\,(\to E) \qquad \frac{\Gamma, X \vdash Y}{\Gamma \vdash X \to Y}\,(\to I) \qquad \frac{\Gamma \vdash X \qquad \Delta, X \vdash Y}{\Gamma, \Delta \vdash Y}\,(\text{Cut})$$

The rules for introducing and eliminating $\forall$ are also analogous to the usual rules. Given any $\mathbf{rn} : \mathcal{B} \to \mathcal{B}$ a bijective renaming function for !-boxes that is identity except on $\downarrow A$, and let $\mathbf{rn}(X)$ be the application of that renaming to a formula. Then:

$$\frac{\Gamma \vdash \mathbf{rn}(X)}{\Gamma \vdash \forall A.\ X}\,(\forall I) \qquad \frac{\Gamma \vdash \forall A.\ X}{\Gamma \vdash \mathbf{rn}(X)}\,(\forall E)$$

where in the case of $\forall I$ we also require that $\mathbf{rn}(\downarrow A)$ is disjoint from $\text{Boxes}(\Gamma)$.

To these core logical rules, we add rules capturing the fact that $=$ is an equivalence relation and a congruence:

$$\frac{}{\Gamma \vdash G = G}\,(\text{Refl}) \qquad \frac{\Gamma \vdash G = H}{\Gamma \vdash H = G}\,(\text{Symm}) \qquad \frac{\Gamma \vdash G = H \qquad \Gamma \vdash H = K}{\Gamma \vdash G = K}\,(\text{Trans})$$

$$\frac{\Gamma \vdash G = H}{\Gamma \vdash [G]^A = [H]^A}\,(\text{Box}) \qquad \frac{\Gamma \vdash G = H}{\Gamma \vdash FG = FH}\,(\text{Prod}) \qquad \frac{\Gamma \vdash G = G'}{\Gamma \vdash \text{Ins}_{A \ni K}(G) = \text{Ins}_{A \ni K}(G')}\,(\text{Ins})$$

where $\text{Ins}_{A \ni K}$ inserts the expression K into the !-box $A \in \text{Boxes}(G)$. The last three rules allow an equation to be applied to a sub-expression. The first two rules allow us to build the context on to the outside of an equation, whereas the third one allows us to add some extra context within any !-box in an equation. These are precisely the equational reasoning rules introduced for !-tensors in [13]. The only difference is we call the 'weakening' operation from that paper 'insertion' to avoid clash with the logical notion.

The main utility of universal quantification is to control the application of !-box operations. In order to start instantiating a !-box (or one of its children), it must be under a universal quantifier:

$$\frac{\Gamma \vdash \forall A.\ X}{\Gamma \vdash \text{Kill}_B(X)}\,(\text{Kill}) \qquad \frac{\Gamma \vdash \forall A.\ X}{\Gamma \vdash \text{Exp}_B(X)}\,(\text{Exp})$$

$$\frac{\Gamma \vdash \forall A.\ X}{\Gamma \vdash \text{Drop}_B(X)}\,(\text{Drop}) \qquad \frac{\Gamma \vdash \forall A.\ X}{\Gamma \vdash \text{Copy}_B(X)}\,(\text{Copy})$$

where $B \leq A \in \text{Boxes}(X)$. These rules, along with $(\forall E)$ play an analogous role to the substitution of a universally-quantified variable for an arbitrary term.

The final rule of the logic is *!-box induction*, which allows us to introduce new !-boxes. For a top-level !-box A, we have:

$$\frac{\Gamma \vdash \text{Kill}_A(X) \qquad \Delta, X \vdash \forall B_1. \ \ldots \forall B_n.\ \text{Exp}_A(X)}{\Gamma, \Delta \vdash X}\,(\text{Induct})$$

where A does not occur free in Γ or Δ and B_1 to B_n are the fresh names of children of A coming from its expansion.

6 Semantics

In this section, we give a semantic interpretation for !-logic formulas using a compact closed category $\mathcal{C}$. For any compact closed category $\mathcal{C}$, a choice of valuation $[\![-]\!] : \Sigma \to \mathcal{C}$ of the generators in Σ will fix a unique morphism $[\![G]\!]$ for any *concrete* (i.e. !-box-free) tensor G. Thus $\mathcal{C}$ comes with an interpretation for equality between concrete tensors. From this, we can build up everything else. For concrete tensors G, H, there is an obvious way to assign a truth value to the formula $G = H$:

$$[\![G = H]\!] := \begin{cases} T & \text{if } [\![G]\!] = [\![H]\!] \\ F & \text{otherwise} \end{cases} \tag{6}$$

As we first mentioned in Section 4, !-logic formulas should be thought of as mappings from instantiations to truth values. Equivalently, they can be thought of as sets of instantiations: namely the set of all instantiations for which the formula holds. Applying this interpretation to atomic formulas yields the following definition:

▶ **Definition 16.** For an atomic !-formula $G = H$ and a valuation $[\![-]\!] : \Sigma \to \mathcal{C}$, we let:

$$[\![G = H]\!] = \left\{ i \in \mathrm{Inst}(\mathrm{Boxes}(G = H)) \;\middle|\; [\![i(G)]\!] = [\![i(H)]\!] \right\} \tag{7}$$

Concrete tensors are equal if and only if they are equal for the trivial instantiation 1. We can interpret truth values as a special case of sets of instantiations: $T = \{1\}$ and $F = \{\}$. Then, in the case of concrete tensors, (7) reduces to (6).

For a forest F and any $i \in \mathrm{Inst}(F)$, and a component $S \subseteq F$, we write $i|_S$ for the restriction of i to only operations involving elements of S (or fresh copies thereof). For a !-formula X, we write $i|_X$ for $i|_{\mathrm{Boxes}(X)}$. Using restrictions of instantiations, we can lift the above definition from atoms to all formulas.

▶ **Definition 17.** The interpretation $[\![-]\!]$ of a !-logic formula is defined recursively as:

$$\begin{aligned}
[\![X \wedge Y]\!] &:= \left\{ \; i \in \mathrm{Inst}(\mathrm{Boxes}(X \wedge Y)) \;\middle|\; i|_X \in [\![X]\!] \wedge i|_Y \in [\![Y]\!] \; \right\} \\
[\![X \to Y]\!] &:= \left\{ \; i \in \mathrm{Inst}(\mathrm{Boxes}(X \to Y)) \;\middle|\; i|_X \in [\![X]\!] \to i|_Y \in [\![Y]\!] \; \right\} \\
[\![\forall A.\ X]\!] &:= \left\{ \; i \in \mathrm{Inst}(\mathrm{Boxes}(\forall A.\ X)) \;\middle|\; \forall j \in \mathrm{Inst}(\downarrow A) \ . \ i \circ j \in [\![X]\!] \; \right\}
\end{aligned}$$

This style of interpretation is very much analogous to that of predicate logic. Whereas one can interpret a predicate with free variables as the set of all values of those variables for which the predicate is true, a !-logic formula is interpreted as the set of all instantiations at which the resulting concrete formula holds.

By contrast, we always interpret sequents as truth values. To do so, we push all of the assumptions to the right and universally quantify over any free !-boxes:

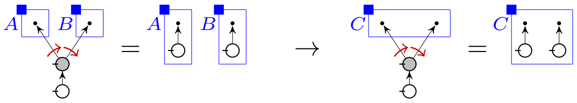
$$[\![X_1, \ldots, X_n \vdash Y]\!] := [\![\forall A_1 \ldots \forall A_m.((X_1 \wedge \ldots \wedge X_n) \to Y)]\!]$$

where $\downarrow A_1, \ldots, \downarrow A_m$ are the free !-boxes in $X_1, \ldots, X_n, Y$.

▶ **Theorem 18** (Soundness). *If $\Gamma \vdash X$ is derivable in !L, then $[\![\Gamma \vdash X]\!]$ is true for any compact closed category $\mathcal{C}$.*

Proof. See Appendix A. ◀

The question of completeness for !L is still open. For the case of atomic !-formulas, this seems to follow straightforwardly from the fact that string diagrams (or equivalently, tensors) are sound and complete for compact closed categories. So, concrete !-tensor equations are true in all models if and only if they are identical tensors. Thus, for the case of general !-tensor equations, the problem reduces to deciding whether two !-tensors with corresponding !-boxes always have identical instances. However, once implication enters the game, we get many non-trivial formulas that hold in all models. For example, an equation with two !-boxes without edges between them always implies another equation obtained by *merging* those !-boxes:

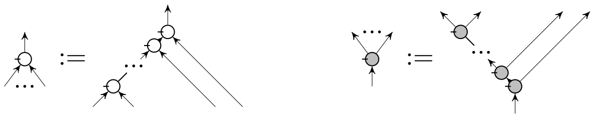


In this case, it is always possible to use !-box induction to prove such an implication (and many others). However, whether the rules in Section 5 suffice to get everything is a topic of continuing research.

7 Inductive proofs for non-commutative bialgebras

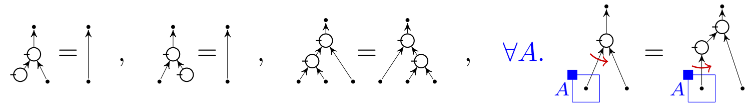
In this section, we will give a flavour for formal proofs in !L and show how they can be used to derive highly non-trivial !-box equations using a combination of !-box induction and rewriting. To avoid massive proof trees, we will abbreviate stacks of equational reasoning rules as sequences of rewrite steps (marked with (*)'s), suppress ∀-intro/elim, and write (Assm) to abbreviate using an assumption.

Recall that a *bialgebra* consists of a monoid, a comonoid, and four extra equations governing their interaction. We will extend the signature of (co)monoids to also allow for n-ary operations, standing for left-associated trees of multiplications and comultiplications:

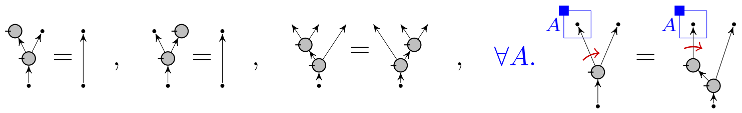


We then assume the usual (co)monoid laws, along with the definition of a higher-arity tree:

$\Gamma_M \ :=$

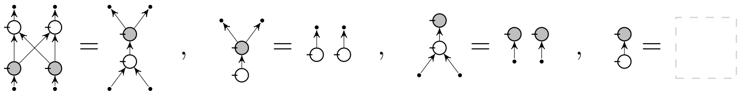


$\Gamma_C \ :=$



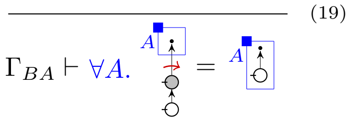
For bialgebras, we start with these equations and add four more:

$\Gamma_{BA} \ := \quad \Gamma_M, \ \Gamma_C,$

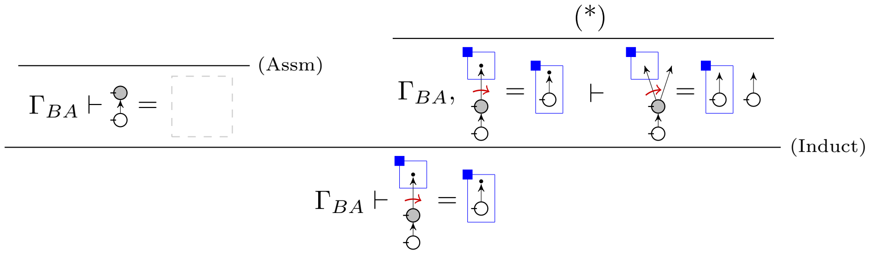


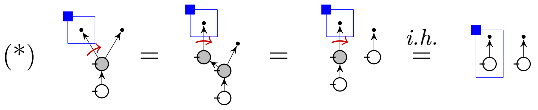
As can be seen from the second bialgebra equation, units are copied by comultiplications. We saw an n-ary generalisation of this in equation (1), which we can now prove formally (and succinctly!):

▶ **Theorem 19.**



Proof.

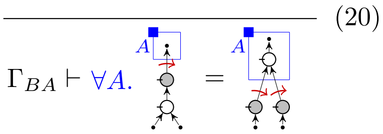




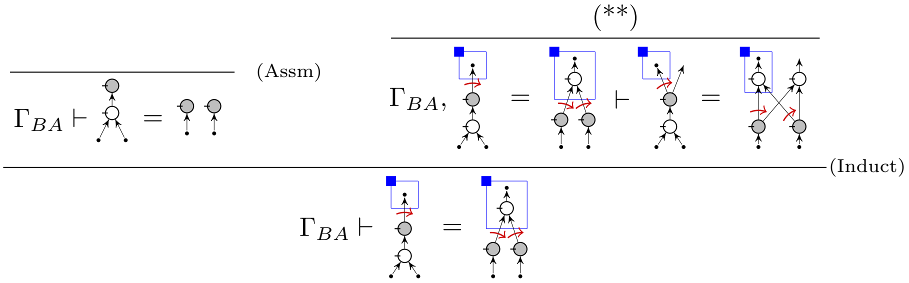
◀

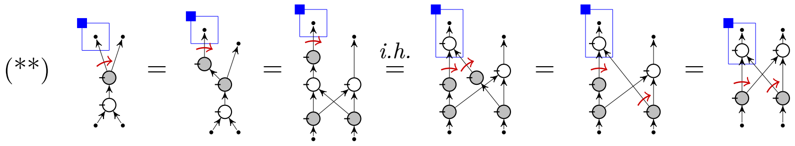
The fact that counits are copied by trees of multiplications could be proved similarly, but we can generalise even more. We now prove that a tree of multiplications, followed by a tree of comultiplications is equal to a complete bipartite graph of comultiplications before multiplications. This rule generalises (and hence can replace) all 4 of the existing bialgebra rules. First we need a little lemma:

▶ **Lemma 20.**



Proof.

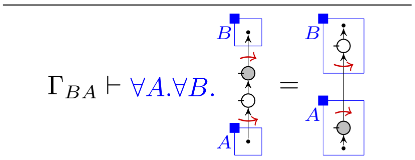




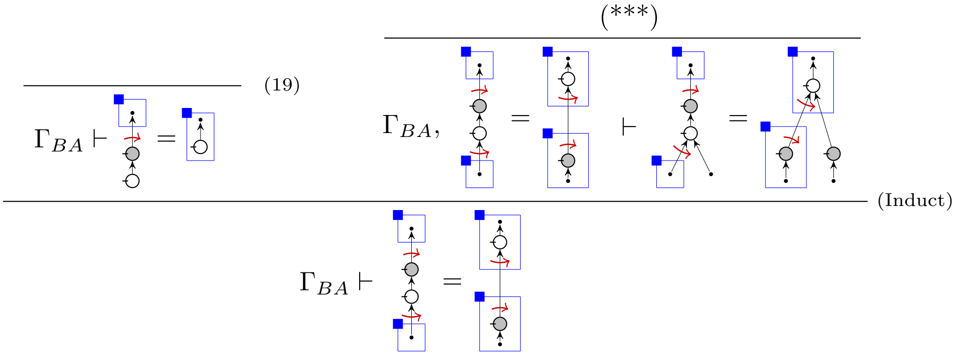
◀

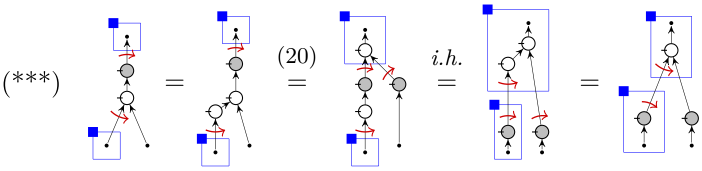
. . . from which we can prove the main theorem:

▶ **Theorem 21.**



Proof.





◀

References

1 John C. Baez and Jason Erbele. Categories in control. Technical report, arXiv:1405.6881, 2014.

2 F. Bonchi, P. Sobocinski, and F. Zanasi. A categorical semantics of signal flow graphs. In *CONCUR'14: Concurrency Theory.*, volume 8704 of *Lecture Notes in Computer Science*, pages 435–450. Springer, 2014.

3 B. Coecke. Quantum picturalism. *Contemporary Physics*, 51:59–83, 2009. arXiv:0908.1787.

4 B. Coecke and R. Duncan. Interacting quantum observables. In *Proceedings of the 37th International Colloquium on Automata, Languages and Programming (ICALP)*, Lecture Notes in Computer Science, 2008.

5 B. Coecke, R. Duncan, A. Kissinger, and Q. Wang. Strong complementarity and non-locality in categorical quantum mechanics. In *Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science (LICS)*. IEEE Computer Society, 2012. arXiv:1203.4988.

6 Lucas Dixon and Ross Duncan. Extending Graphical Representations for Compact Closed Categories with Applications to Symbolic Quantum Computation. *AISC/MKM/Calculemus*, pages 77–92, 2008.

7 Peter Dybjer and Anton Setzer. A finite axiomatization of inductive-recursive definitions. In Jean-Yves Girard, editor, *Typed Lambda Calculi and Applications*, volume 1581 of *Lecture Notes in Computer Science*, pages 129–146. Springer Berlin Heidelberg, 1999.

8 Andre Joyal and Ross Street. The geometry of tensor calculus I. *Advances in Mathematics*, 88:55–113, 1991.

9 D. Kartsaklis. *Compositional Distributional Semantics with Compact Closed Categories and Frobenius Algebras*. PhD thesis, University of Oxford, 2014.

10 Aleks Kissinger. Abstract tensor systems as monoidal categories. In C Casadio, B Coecke, M Moortgat, and P Scott, editors, *Categories and Types in Logic, Language, and Physics: Festschrift on the occasion of Jim Lambek's 90th birthday*, volume 8222 of *Lecture Notes in Computer Science*. Springer, 2014. arXiv:1308.3586 [math.CT].

11 Aleks Kissinger, Alex Merry, and Matvey Soloviev. Pattern graph rewrite systems. In *Proceedings of DCM 2012*, volume 143 of *EPTCS*, 2012. arXiv:1204.6695 [math.CT].

12 Aleks Kissinger and David Quick. Tensors, !-graphs, and non-commutative quantum structures. In *Proceedings of the 11th workshop on Quantum Physics and Logic, QPL 2014, Kyoto, Japan, 4-6th June 2014.*, pages 56–67, 2014. arXiv:1412.8552 [cs.LO].

13 Aleks Kissinger and David Quick. Tensors, !-graphs, and non-commutative quantum structures (extended version), 2015. arXiv:1503.01348.

14 Aleks Kissinger and Vladimir Zamdzhiev. Quantomatic: A proof assistant for diagrammatic reasoning, 2015. arXiv:1503.01034.

15 Alexander Merry. *Reasoning with !-Graphs*. PhD thesis, University of Oxford, 2014.

A Proof of soundness for !L

In this section, we prove Theorem 18, i.e. the soundness of $[\![-]\!]$ with respect to !L. To do so, it suffices to show that $[\![-]\!]$ respects each of the rules of the logic.

For $i \in \text{Inst}(F)$ and a formula X such that $\text{Boxes}(X)$ is a component of F, we will write $i \vDash X$ as shorthand for $i|_X \in [\![X]\!]$. Using this notation, we can rewrite the interpretation as follows:

$$\begin{aligned}
i \vDash G = H &\iff [\![i(G)]\!] = [\![i(H)]\!] && i \in \text{Inst}(\text{Boxes}(G = H)) \\
i \vDash X \wedge Y &\iff i \vDash X \wedge i \vDash Y && i \in \text{Inst}(\text{Boxes}(X \wedge Y)) \\
i \vDash X \to Y &\iff i \vDash X \to i \vDash Y && i \in \text{Inst}(\text{Boxes}(X \to Y)) \\
i \vDash \forall A.X &\iff \forall j \in \text{Inst}(\downarrow A).\ i \circ j \vDash X && i \in \text{Inst}(\text{Boxes}(\forall A.X))
\end{aligned}$$

Universal quantification over entire components of $\text{Boxes}(X)$ is well-behaved for the following reason:

▶ **Lemma 22.** *For a forest F, let A, B be elements in distinct connected components of F, and let $\text{Boxes}(X) \triangle F$. Then, $\text{Op}_A(\text{Op}'_B(X)) = \text{Op}'_B(\text{Op}_A(X))$ for any !-box operations $\text{Op}_A, \text{Op}'_B$.*

Proof. Since !-box operations recurse down to equations between !-tensors, it suffices to show that $\text{Op}_A(\text{Op}'_B(G = H)) = \text{Op}'_B(\text{Op}_A(G = H))$. Since neither A nor B is a child of the other, this is easy to check. The only complication is dealing with the freshness functions $\mathbf{fr}_A, \mathbf{fr}_B$ (possibly) associated with the two operations. These necessarily operate on disjoint sets of boxnames, so the only overlap might be on edgenames. However, since there is an infinite supply of fresh edgenames, it is always possible to choose new freshness functions such that $\mathbf{fr}_A \circ \mathbf{fr}_B = \mathbf{fr}'_B \circ \mathbf{fr}'_A$. Then, it is straightforward to check that $\text{Op}_{A,\mathbf{fr}_A}(\text{Op}'_{B,\mathbf{fr}_B}(G = H)) = \text{Op}'_{B,\mathbf{fr}'_B}(\text{Op}_{A,\mathbf{fr}'_A}(G = H))$. ◀

A related fact about re-ordering operations in an instantiation is that they can always be put in normal form:

▶ **Lemma 23.** *Given an instantiation $i \in \text{Inst}(X)$ and a top-level !-box $A \in X^\top$, i can be rewritten as $i' \circ \text{Kill}_A \circ \text{Exp}^n_A$ where $i' \in \text{Inst}(\text{Kill}_A \circ \text{Exp}^n_A(X))$.*

Proof. We need to check that operations on A can always be commuted to the right, past other operations. If B is not nested in A, this is true by Lemma 22. Otherwise, $B \leq A$ and:

- If $\text{Op}_A = \text{Kill}_A$ then killing A will erase any part of the !-formula resulting from Op_B, i.e. $\text{Kill}_A \circ \text{Op}_B = \text{Kill}_A$.
- If $\text{Op}_A = \text{Exp}_{A,\mathbf{fr}}$ then $\text{Exp}_{A,\mathbf{fr}} \circ \text{Op}_B = \text{Op}_{\mathbf{fr}(B)} \circ \text{Op}_B \circ \text{Exp}_{A,\mathbf{fr}}$. In the case that $\text{Op}_B = \text{Exp}_B$, freshness functions on the RHS need to be chosen to produce identical names to the LHS. ◀

▶ **Notation 24.** We will write KE^n_A as a shorthand for $\text{Kill}_A \circ \text{Exp}^n_A$.

▶ **Lemma 25.** *For any !-formula X and for $B_1, \ldots B_n$ the free, top-level !-boxes in X:*

$$\forall i \in \text{Inst}(\text{Boxes}(X)).\ i \vDash X \iff [\![\forall B_1 \ldots \forall B_n.X]\!] = \{1\} = T$$

Proof. First, assume the LHS, which is equivalent to $[\![X]\!] = \text{Inst}(\text{Boxes}\,X)$. For any !-formula Y, if $B_k \in \text{Boxes}(Y)^\top$ and $[\![Y]\!] = \text{Inst}(\text{Boxes}(Y))$, then $[\![Y]\!]$ contains all possible instantiations of $\text{Boxes}(Y)$. In particular, it contains $i \circ j$ for any $i \in \text{Inst}(\text{Boxes}(\forall B_k.Y))$

and $j \in \mathrm{Inst}(\downarrow B_k)$. Thus, $[\![\forall B_k.Y]\!] = \mathrm{Inst}(\mathrm{Boxes}(\forall B_k.Y))$. Iterating this implication, we have $[\![\forall B_1 \ldots \forall B_n.X]\!] = \mathrm{Inst}(\mathrm{Boxes}(\forall B_1 \ldots \forall B_n.X)) = \{1\} = T$.

Conversely, assume $[\![\forall B_1 \ldots \forall B_n.X]\!] = T$. Then every instantiation of the form $j = i_1 \circ i_2 \circ \ldots \circ i_n$, where the operations in i_k only involve !-boxes in $\downarrow B_k$ is in $[\![X]\!]$. But then, by Lemma 22, we can freely commute !-box operations in distinct components of $\mathrm{Boxes}(X)$. So, in fact, every instantiation $i \in \mathrm{Inst}(\mathrm{Boxes}(X))$ is equivalent to an instantiation of the form of j. Then, since $j \in [\![X]\!]$, so is i. ◀

▶ **Theorem 26.** *For any valuation $[\![-]\!] : \Sigma \to \mathcal{C}$, the rules (Ident), (Weaken), (Perm), (Contr), ($\wedge I$), ($\wedge E_1$), ($\wedge E_2$), ($\to E$), ($\to I$), (Cut), ($\forall I$), ($\forall E$), (Refl), (Symm), (Tran), (Box), (Prod), (Ins), (Kill), (Exp), (Drop), (Copy), and (Induct) are sound with respect to $[\![-]\!]$.*

Proof. The basic structural rules just reduce to the same rules concerning instantiations. Let K be the conjunction of Γ and K' the conjunction of Δ throughout. By Lemma 25, to check that $[\![\Gamma \vdash X]\!]$ is true, it suffices to check that, for all $i \in \mathrm{Inst}(\mathrm{Boxes}(K \to X))$, $i \vDash K \to X$.

- (Ident) Fix $i \in \mathrm{Inst}(\mathrm{Boxes}(X))$. We need to show $i \in X \to X$, but this is equivalent to $i \vDash X \to i \vDash X$, which is trivially true.
- (Weaken) Fix $i \in \mathrm{Inst}(\mathrm{Boxes}((K \wedge X) \to Y))$ and assume $i \vDash K \to Y$. Then, if $i \vDash K \wedge X$, then $i \vDash K$. So, by assumption, $i \vDash Y$. Thus $i \vdash (K \wedge X) \to Y$.
- (Perm) and (Contr) follow from associativity, commutativity and idempotence of $\wedge$.
- ($\wedge I$) Fix $i \in \mathrm{Inst}(\mathrm{Boxes}((K \wedge K') \to (X \wedge Y)))$ and assume $i \vDash K \to X$ and $i \vDash K' \to Y$. If $i \vDash K \wedge K'$, we have $i \vDash K$ and hence $i \vDash X$. We also have $i \vDash K'$ and hence $i \vDash Y$. Thus $i \vDash X \wedge Y$.
- ($\wedge E1$) Fix $i \in \mathrm{Inst}(\mathrm{Boxes}(K \to X))$. Then, there exists $i' \in \mathrm{Inst}(\mathrm{Boxes}(K \to (X \wedge Y)))$ that restricts to i. Assume $i' \vDash K \to (X \wedge Y)$. If $i \vDash K$ then $i' \vDash K$ and hence $i' \vDash X \wedge Y$, which implies that $i' \vDash X$. So, $i \vDash X$.
- ($\wedge E2$) is similar to ($\wedge E1$).
- ($\to E$) Fix $i \in \mathrm{Inst}(\mathrm{Boxes}((K \wedge X) \to Y))$ and assume $i \vDash K \to (X \to Y)$. Then, if $i \vDash K \wedge X$ then $i \vDash K$. So, $i \vDash X \to Y$. But, since it is also the case that $i \vDash X$, $i \vDash Y$. Thus $i \vDash (K \wedge X) \to Y$.
- ($\to I$) is the same as ($\to E$) in reverse.
- (Cut) Fix $i \in \mathrm{Inst}(\mathrm{Boxes}(K \wedge K' \to Y))$. Then, there exists $i' \in \mathrm{Inst}(\mathrm{Boxes}(K \to X) \cup \mathrm{Boxes}((K' \wedge X) \to Y))$ that restricts to i. Assume $i' \vDash K \to X$ and $i' \vDash (K' \wedge X) \to Y$. If $i \vDash K \wedge K'$, then $i' \vDash K \wedge K'$ so $i' \vDash K$ and $i' \vDash K'$. The former also implies that $i' \vDash X$. So, $i' \vDash K' \wedge X$ and hence $i' \vDash Y$. Finally, this implies $i \vDash Y$.

($\forall I$) Fix $i \in \mathrm{Inst}(\mathrm{Boxes}(K \to \forall A.X))$. We need to show that for any $j \in \mathrm{Inst}(\downarrow A)$, $i \circ j \vDash K \to X$. Assume without loss of generality that any !-box names on operations in i are disjoint from $\mathbf{rn}(\downarrow A)$. This is possible because $\mathbf{rn}(\downarrow A)$ must already be disjoint from $\mathrm{Boxes}(\Gamma)$ (by side-condition) and it must be disjoint from $\mathrm{Boxes}(\forall A.X) = \mathrm{Boxes}(X) \backslash \downarrow A$ by injectivity of $\mathbf{rn}$. The only other !-box names in i are those introduced during instantiation, which can be freely chosen. Let $\mathbf{rn}(j)$ be the instantiation of $\mathbf{rn}(\downarrow A)$ obtained by renaming operations according to $\mathbf{rn}$. Then, by assumption of the rule, we have $i \circ \mathbf{rn}(j) \vDash K \to \mathbf{rn}(X)$. Since $\mathbf{rn}$ is identity except on $\downarrow A$, we have $\mathbf{rn}(i \circ j) \vDash \mathbf{rn}(K \to X)$ and thus $i \circ j \vDash K \to X$.

($\forall E$) Fix $i \in \mathrm{Inst}(\mathrm{Boxes}(K \to \mathbf{rn}(X)))$. Suppose $i \vDash K$, then, by assumption $i \vDash \forall A.X$. Let $i' = i|_{\forall A.X}$, then $i' \vDash \forall A.X$, which implies that for all $j \in \mathrm{Inst}(\downarrow A)$, we have $i' \circ j \vDash X$. Renaming both sides yields $\mathbf{rn}(i' \circ j) \vDash \mathbf{rn}(X)$, and since $\mathbf{rn}$ is identity except on $\downarrow A$,

$i' \circ \mathbf{rn}(j) \vDash \mathbf{rn}(X)$. Now, since we are free to choose j, we choose it such that $(i' \circ \mathbf{rn}(j))|_{\mathbf{rn}(X)}$ is equivalent to $i|_{\mathbf{rn}(X)}$. Then $i \vDash \mathbf{rn}(X)$.

The rules (Refl), (Symm), and (Trans) reduce to the properties of equality in $\mathcal{C}$. The congruence rules (Box), (Prod), and (Ins) were proven sound in [13], where the only difference here is the additional (unused) context Γ.

(Kill) Fix $i \in \text{Inst}(\text{Boxes}(K \to \text{Kill}_B(X)))$. Then if $i \vDash K$, by assumption $i \vDash \forall A.X$. Since $B \leq A$ does not occur free in $\forall A.X$, $i \circ \text{Kill}_B \vDash \forall A.X$. For $i' = i|_{\forall A.X}$, choose $j \in \text{Inst}(\downarrow A)$ such that $(i' \circ j)|_X$ is equivalent to $(i \circ \text{Kill}_B)|_X$. Then, $i' \circ j \vDash X$, so $i \circ \text{Kill}_B \vDash X$, and $i \vDash \text{Kill}_B(X)$. (Exp) is similar.

(Copy) and (Drop) are also similar. However, when we choose $j \in \text{Inst}(\downarrow A)$ such that $(i' \circ j)|_X$ is equivalent to $(i \circ \text{Copy}_B)|_X$ or $(i \circ \text{Copy}_B)|_X$, we make use of the fact that instantiations involving Copy / Drop can always be reduced to a normal form which only includes Exp and Kill. This was proven in [13].

Finally, we prove the (Induct) rule. For any top-level !-box A, Lemma 23 says that we can write any instantiation i equivalently as $j \circ \text{KE}_A^n$, where j doesn't contain A. Thus, we will show that, for all n, and all instantiations of the form $i := j \circ \text{KE}_A^n$, $i \vDash (K \wedge K') \to X$. We proceed by induction on n.

For the base case, $i = j \circ \text{Kill}_A$. If $i \vDash K$, then since K doesn't contain A, $i \vDash K$ implies $j \vDash K$. So, by the first premise $j \vDash \text{Kill}_A(X)$. Thus $j \circ \text{Kill}_A \vDash X$, as required. For the step case, assume that for all instantiations of $(K \wedge K') \to X$ of the form $i := j \circ \text{KE}_A^n$, $i \vDash (K \wedge K') \to X$. We need to show for all $i' := j \circ \text{KE}_A^{n+1}$, $i' \vDash (K \wedge K') \to X$. If $i' \vDash K \wedge K'$, then $i' \vDash K'$. Then, since K doesn't contain A, $i \vDash K'$. Combining this with the induction hypothesis yields $i \vDash \forall B_1 \ldots \forall B_m. \text{Exp}_A(X)$. Thus, for any instantiation k of the $\downarrow B_1, \ldots, \downarrow B_m$, $i \circ k \vDash \text{Exp}_A(X)$. So, $i \circ k \circ \text{Exp}_A \vDash X$. i' is equivalent to $i \circ k \circ \text{Exp}_A$ for some i, k, so $i' \vDash X$. ◀

Soundness of !L with respect to $[\![-]\!]$ then follows from Theorem 26.

Presenting Morphisms of Distributive Laws*

Bartek Klin[1] and Beata Nachyła[2]

1 University of Warsaw, Poland
klin@mimuw.edu.pl

2 Institute of Computer Science, Polish Academy of Sciences, Poland
beatanachyla@gmail.com

Abstract

A format for well-behaved translations between structural operational specifications is derived from a notion of distributive law morphism, previously studied by Power and Watanabe.

1998 ACM Subject Classification F.3.2 Semantics of Programming Languages – Operational semantics

Keywords and phrases coalgebra, bialgebra, distributive law, structural operational semantics

Digital Object Identifier 10.4230/LIPIcs.CALCO.2015.190

1 Introduction

Since [13], distributive laws of functors (or pointed functors, or monads) over other functors (or copointed functors, or comonads), and bialgebras for them, have been a useful categorical tool to study various kinds of structural operational semantics (SOS, [10, 1]). Several formats of well-behaved operational specifications can be understood as kinds of distributive laws, and some desired properties (mostly compositionality of behavioural equivalences) can be proved in terms of bialgebras. See [7] for a recent introduction to this topic.

One advantage of abstraction is that sometimes notions or results readily available at the abstract level, can be instantiated in the concrete setting in previously unforeseen ways. One such example are morphisms of distributive laws, studied by Power and Watanabe [11, 14] as abstract notions of well-behaved translations between operational specifications.

The issue of translating specifications have attracted limited attention in the SOS community so far; apart from a general notion of conservative extension [1], which can be seen as a simple embedding of one specification into another, only isolated examples of well-behaved translations have been studied [5], with no attempt at a general theory. This is unfortunate, as translating operational semantics from one language to another, and from one type of behaviour to another, is very useful in modular SOS development. When different parts of a language, or different aspects of its semantics, are specified separately, they must be combined somehow, for example via a family of translations.

In [11], distributive law morphisms were studied in the abstract, with a few concrete examples provided in [14] (see also [4] for a slightly different application). In this paper we pick up that line of work and provide a characterization of morphisms between GSOS specifications [3] understood as distributive laws. A morphism between laws is presented as a syntactic translation, together with a behavioural translation presented by inference rules similar to SOS, subject to a compatibility condition. Importantly, for finite specifications the condition is decidable. Decidability is a key property of any reasonable format for

* This work was supported by the Polish National Science Centre (NCN) grant 2012/07/E/ST6/03026.

6th International Conference on Algebra and Coalgebra in Computer Science (CALCO'15).
Editors: Lawrence S. Moss and Pawel Sobocinski; pp. 190–204
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

specifications or their translations, and it is the reason why we choose to work with GSOS rather than with more general, and mathematically more pleasing, distributive laws of monads over comonads [13, 9]. The latter do not admit a decidable presentation (see [8]).

This paper is only a small step in what should become a long-term research programme. We only study morphisms between GSOS specifications of labelled transition systems (LTSs), whereas the abstract framework of distributive laws and their morphisms covers different types of behaviour and even different underlying categories. A question of translating operational descriptions up to various behavioural equivalences, using e.g. ideas from [6], is not touched upon. A framework for defining diagrams of distributive law morphisms and combining operational descriptions by means of (co)limits of such diagrams, and many other related tasks, are left for future work. However, we believe that even a simple characterization of GSOS specification morphisms sheds some light on the strengths and limitations of the bialgebraic approach to SOS translation.

We begin the paper by studying a special case of GSOS, so called simple SOS specifications, recalled from [7] in Section 3. We work out this special case in detail: in Section 4 we define morphisms of simple distributive laws, a special case of definitions studied in [11, 14]. In Section 5, syntactic and behavioural translations are defined, and in Section 6 a compatibility condition is shown that is equivalent to the translations forming a morphism of distributive laws. In Section 7, the case of full GSOS is sketched briefly: while technically slightly more complicated, it does not introduce any new conceptual difficulties. Section 8 contains some illustrative examples.

2 Preliminaries

A *signature* is a set Σ of symbols, where each symbol $\mathtt{f} \in \Sigma$ has an associated finite *arity* $\sharp\mathtt{f} \in \mathbb{N}$. With any signature we associate an endofunctor $\Sigma X = \coprod_{\mathtt{f}\in\Sigma} X^{\sharp\mathtt{f}}$ on the category **Set** of sets and functions. We shall only consider endofunctors Σ that arise from signatures in this way. Algebras for the functor (in the categorical sense) are then exactly algebras for the signature (in the universal-algebraic sense). For any set X, the set Σ^*X of Σ-terms with variables from X carries an obvious Σ-algebra structure; Σ^*0, for 0 the empty set, is an initial Σ-algebra. The construction Σ^* is a monad on **Set**; it is the free monad over Σ.

All coalgebras [12] considered in this paper are for **Set**-functors of the form $BX = (\mathcal{P}_\omega X)^L$, where $\mathcal{P}_\omega$ is the finite powerset functor and L is a finite set of *labels*. As is well known, such coalgebras are finitely branching L-labeled transition systems (LTSs). B-coalgebra morphisms are functional bisimulations, i.e., functions whose graphs are bisimulation relations.

3 Simple distributive laws and SOS

A *simple distributive law* of an endofunctor Σ over an endofunctor B is a natural transformation $\lambda : \Sigma B \Longrightarrow B\Sigma$. A λ-*bialgebra* is then a Σ-algebra $g : \Sigma X \to X$ and a B-coalgebra $k : X \to BX$ with the same carrier X, such that the following diagram commutes:

$$\begin{tikzcd}
\Sigma X \arrow[r, "g"] \arrow[d, "\Sigma k"'] & X \arrow[r, "k"] & BX \\
\Sigma BX \arrow[rr, "\lambda_X"'] & & B\Sigma X. \arrow[u, "Bg"']
\end{tikzcd}$$

Morphisms of bialgebras are functions between their carriers that are simultaneously algebra and coalgebra morphisms between the respective bialgebra components. Bialgebras for a fixed distributive law form a category. Under our assumptions an initial λ-bialgebra always exists, and is of the form:

$$\Sigma\Sigma^*0 \xrightarrow{\cong} \Sigma^*0 \xrightarrow{k_\lambda} B\Sigma^*0\ , \tag{1}$$

where the algebraic component is an initial Σ-algebra, hence (by Lambek's Lemma) an isomorphism. The coalgebraic component k_λ of an initial λ-bialgebra is called the B-coalgebra induced by λ. For a detailed introduction to these notions see e.g. [7].

If Σ is a polynomial functor arising from a signature, and if $BX = (\mathcal{P}_\omega X)^L$ for some finite set L of labels, simple distributive laws may be presented by sets of inference rules. This has been known since [13] for the more general case of GSOS laws (see Section 7.1), and studied in detail in [2, 7]. We now briefly recall the main idea.

For any set X, an *X-literal* (or simply a *literal*, when no risk of confusion arises) is an expression $x \xrightarrow{a} y$ where $x, y \in X$ and $a \in L$. In such a literal, x is called the *source*, a the *label* and y the *target*.

Fix a countably infinite set of variables $\mathcal{V} \ni \mathtt{x}, \mathtt{y}, \mathtt{z}, \ldots$.

▶ **Definition 1.** A *simple SOS specification* (over Σ and L) is a finite set of *simple SOS rules*, i.e., expressions of the form

$$\frac{\left\{\mathtt{x}_{i_j} \xrightarrow{a_j} \mathtt{y}_j\right\}_{j=1..m} \quad \left\{\mathtt{x}_{i_k} \not\xrightarrow{b_k}\right\}_{k=1..l}}{\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_{\sharp\mathtt{f}}) \xrightarrow{c} \mathtt{g}(\mathtt{z}_1, \ldots, \mathtt{z}_{\sharp\mathtt{g}})} \tag{2}$$

where $\mathtt{f}, \mathtt{g} \in \Sigma$ and $m, l \in \mathbb{N}$; all $i_j, i_k \in \{1, \ldots, \sharp\mathtt{f}\}$; all $\mathtt{x}_i, \mathtt{y}_j \in \mathcal{V}$ are pairwise distinct; $\mathtt{z}_1, \ldots, \mathtt{z}_{\sharp\mathtt{g}} \in \{\mathtt{y}_1, \ldots, \mathtt{y}_m\}$; and $a_j, b_k, c \in L$.

The $\mathcal{V}$-literals $\mathtt{x}_{i_j} \xrightarrow{a_j} \mathtt{y}_j$ above are called *positive premises*, and the expressions $\mathtt{x}_{i_k} \not\xrightarrow{b_k}$ *negative premises* of the rule. The $\Sigma\mathcal{V}$-literal below the inference line is called the *conclusion*. The *source* and *target* of the rule are, respectively, the source and the target of its conclusion. Note that variables from the source of a simple SOS rule do not appear in its target.

For any set X, we say that a rule R as in (2) is *triggered* by a set Φ of X-literals if there is a substitution $\sigma : \mathcal{V} \to X$ such that:

- for each positive premise $\mathtt{x}_i \xrightarrow{a} \mathtt{y}$ in R, the literal $\sigma(\mathtt{x}_i) \xrightarrow{a} \sigma(\mathtt{y})$ is in Φ,
- for each negative premise $\mathtt{x}_i \not\xrightarrow{b}$ in R, there is no literal of the form $\sigma(\mathtt{x}_i) \xrightarrow{b} y$ in Φ.

If that is the case, the rule R *infers* from Φ the ΣX-literal

$$\mathtt{f}(\sigma(\mathtt{x}_1), \ldots, \sigma(\mathtt{x}_{\sharp\mathtt{f}})) \xrightarrow{c} \mathtt{g}(\sigma(\mathtt{z}_1), \ldots, \sigma(\mathtt{z}_{\sharp\mathtt{g}})).$$

Note that a single rule may infer more than one ΣX-literal from the same set of X-literals, depending on the substitution σ used. Note also that the literal inferred from a rule, if any, depends only on how σ acts on the variables present in the rule. Moreover, the target of the literal and its transition label depend only on how σ acts on the variables $\mathtt{y}_j$.

By $\Lambda[\Phi]$ we will denote the set of literals inferred from Φ by rules from a specification Λ. Given a specification Λ, for any set X define a function $\lambda_X : \Sigma(\mathcal{P}_\omega X)^L \to (\mathcal{P}_\omega \Sigma X)^L$ by:

$$\begin{gathered}\lambda_X(\mathtt{f}(\gamma_1, \ldots, \gamma_{\sharp\mathtt{f}}))(c) = \left\{t \in \Sigma X \mid (\mathtt{f}(x_1, \ldots, x_{\sharp\mathtt{f}}) \xrightarrow{c} t) \in \Lambda[\Phi]\right\}, \text{ where} \\ \Phi = \{x_i \xrightarrow{a} y \mid 1 \leq i \leq \sharp\mathtt{f},\ a \in L,\ y \in \gamma_i(a)\}\end{gathered} \tag{3}$$

for any $\mathtt{f} \in \Sigma$, a family of functions $\gamma_1, \ldots, \gamma_{\sharp f} : L \to \mathcal{P}_\omega X$, a $c \in L$, and a family of distinct $x_1, \ldots, x_{\sharp\mathtt{f}}$. It is not difficult to see that for a given $\mathtt{f}(\gamma_1, \ldots, \gamma_{\sharp\mathtt{f}})$, the value of λ_X does not depend on the choice of $x_1, \ldots, x_{\sharp\mathtt{f}}$, as long as they are distinct; this is thanks to the fact that variables from the source of a simple SOS rule do not appear in its target. Note that one could even choose some $x_i = x_j$ as long as $\gamma_i = \gamma_j$, without affecting the value of λ_X.

▶ **Example 2.** Consider a signature $\Sigma = \{\mathtt{f}, \mathtt{g}\}$ with $\sharp\mathtt{f} = 2$, $\sharp\mathtt{g} = 1$, giving rise to a functor $\Sigma X = X^2 + X$. Consider also a set of labels $L = \{a, b\}$, and let a specification Λ consist of the following three rules:

$$\frac{\mathtt{x} \xrightarrow{a} \mathtt{x}' \quad \mathtt{y} \xrightarrow{b} \mathtt{y}'}{\mathtt{f}(\mathtt{x}, \mathtt{y}) \xrightarrow{a} \mathtt{f}(\mathtt{x}', \mathtt{y}')} \qquad \frac{\mathtt{x} \stackrel{b}{\not\longrightarrow} \quad \mathtt{x} \xrightarrow{a} \mathtt{x}' \quad \mathtt{y} \xrightarrow{a} \mathtt{y}'}{\mathtt{f}(\mathtt{x}, \mathtt{y}) \xrightarrow{a} \mathtt{g}(\mathtt{y}')} \qquad \frac{\mathtt{x} \xrightarrow{b} \mathtt{x}'}{\mathtt{g}(\mathtt{x}) \xrightarrow{b} \mathtt{f}(\mathtt{x}', \mathtt{x}')}$$

where $\mathtt{x}, \mathtt{y}, \mathtt{y}', \mathtt{y}' \in \mathcal{V}$. For $X = \{u, v, w\}$, consider the set of X-literals:

$$\Phi = \{u \xrightarrow{a} v,\ u \xrightarrow{a} w,\ v \xrightarrow{a} u,\ v \xrightarrow{b} w\}.$$

Then all three rules are triggered, and the following literals are inferred:

$$\begin{aligned}\Lambda[\Phi] = \{&\mathtt{f}(u, v) \xrightarrow{a} \mathtt{f}(v, w),\ \mathtt{f}(u, v) \xrightarrow{a} \mathtt{f}(w, w),\ \mathtt{f}(v, v) \xrightarrow{a} \mathtt{f}(u, w),\ \mathtt{f}(u, u) \xrightarrow{a} \mathtt{g}(v),\\ &\mathtt{f}(u, u) \xrightarrow{a} \mathtt{g}(w),\ \mathtt{f}(u, v) \xrightarrow{a} \mathtt{g}(u),\ \mathtt{g}(v) \xrightarrow{b} \mathtt{f}(w, w)\}.\end{aligned}$$

Keeping Λ and X as above, choose $\mathtt{f} \in \Sigma$ and $a \in L$, and consider $\gamma_1, \gamma_2 : L \to P_\omega X$ defined by: $\gamma_1(a) = \{v, w\}$, $\gamma_1(b) = \emptyset$, $\gamma_2(a) = \{u\}$, $\gamma_2(b) = \{w\}$. For any $x_1 \neq x_2$, according to (3) this gives rise to:

$$\begin{gathered}\Phi = \{x_1 \xrightarrow{a} v,\ x_1 \xrightarrow{a} w,\ x_2 \xrightarrow{a} u,\ x_2 \xrightarrow{b} w\}, \text{ and}\\ \lambda_X(\mathtt{f}(\gamma_1, \gamma_2))(a) = \{\mathtt{f}(v, w),\ \mathtt{f}(w, w),\ \mathtt{g}(u)\};\end{gathered}$$

note how the latter does not depend on the choice of x_1 and x_2.

The following theorem is a special case of a more general result concerning GSOS specifications, formulated first in [13] and proved in detail in [2]; we omit the proof here.

▶ **Theorem 3.** *For any specification Λ, the functions λ_X defined by* (3) *form a natural transformation $\lambda : \Sigma(\mathcal{P}_\omega -)^L \Longrightarrow (\mathcal{P}_\omega \Sigma -)^L$. Moreover, every natural transformation of this type arises this way from some simple SOS specification.*

Rule-based presentation of distributive laws suggests a notion of derivation; this can be defined in standard terms of SOS theory, see e.g. [1]. A labelled transition system derived from a simple SOS specification Λ coincides with the coalgebra induced by the corresponding distributive law λ, i.e., with the coalgebraic component of the initial λ-bialgebra (1).

4 Distributive law morphisms

▶ **Definition 4.** A *distributive law morphism* from $\lambda : \Sigma B \Longrightarrow B\Sigma$ to $\lambda' : \Sigma' B' \Longrightarrow B'\Sigma'$ consists of natural transformations $\alpha : \Sigma \Longrightarrow \Sigma'$ and $\theta : B' \Longrightarrow B$ such that the following diagram commutes:

$$\begin{tikzcd}
\Sigma B' \arrow[r, Rightarrow, "\Sigma\theta"] \arrow[dr, Rightarrow, "\alpha B'"'] & \Sigma B \arrow[r, Rightarrow, "\lambda"] & B\Sigma \arrow[dr, Rightarrow, "B\alpha"] & \\
& \Sigma' B' \arrow[r, Rightarrow, "\lambda'"'] & B'\Sigma' \arrow[r, Rightarrow, "\theta\Sigma'"'] & B\Sigma'.
\end{tikzcd} \tag{4}$$

In [11, 14] distributive law morphisms were defined in a more general framework where Σ, B and Σ', B' operate on different categories connected by an additional functor. Although important for SOS specifications of systems other than LTSs, here we work in a simplified setting to illustrate the basic issues of presenting morphisms on a classical example.

In [11, 14] another definition of distributive law morphism was also considered, with two natural transformations $\alpha : \Sigma \Longrightarrow \Sigma'$, $\theta : B \Longrightarrow B'$ going in the same direction. We defer the issue of presenting such morphisms to a full version of this paper; their presentations are technically quite similar to the ones presented here, although they carry slightly different intuitions.

Any distributive law morphism as in Definition 4 induces a functor from the category of λ'-bialgebras to the category of λ-bialgebras (see also [4]), mapping every

$$\Sigma' X \xrightarrow{g} X \xrightarrow{k} B'X \qquad \text{to} \qquad \Sigma X \xrightarrow{\alpha_X} \Sigma' X \xrightarrow{g} X \xrightarrow{k} B'X \xrightarrow{\theta_X} BX.$$

Consider this functor applied to the initial λ'-bialgebra, and the unique morphism α_0^* from the initial λ-bialgebra to the result of that application:

$$\begin{tikzcd}
\Sigma\Sigma^* 0 \arrow[rr, "\cong"] \arrow[d, "\Sigma\alpha_0^*"'] & & \Sigma^* 0 \arrow[rr, "k_\lambda"] \arrow[d, "\alpha_0^*"'] & & BX \arrow[d, "B\alpha_0^*"] \\
\Sigma\Sigma'^* 0 \arrow[r, "\alpha_{\Sigma'^* 0}"] & \Sigma'\Sigma'^* 0 \arrow[r, "\cong"] & \Sigma'^* 0 \arrow[r, "k_{\lambda'}"] & B'\Sigma'^* 0 \arrow[r, "\theta_{\Sigma'^* 0}"] & B\Sigma'^* 0
\end{tikzcd}$$

As an algebra morphism from an initial Σ-algebra, α_0^* is an inductively defined translation of Σ-terms to Σ'-terms according to α (hence the name). As a bialgebra morphism, it is also a B-coalgebra morphism.

For $BX = (\mathcal{P}_\omega X)^L$, coalgebra morphisms are functional bisimulations. As a result, if α and θ form a distributive law morphism from λ to λ' then the translation α_0^* of Σ-terms to Σ'-terms according to α, maps a term in the transition system k_λ induced by λ, to a bisimilar term in the system $k_{\lambda'}$ induced by λ', with the behaviour translated according to θ. A useful intuition to hold is that B' is somehow richer than B, and θ projects B'-behaviours to B-behaviours by ignoring some components. In the following sections we will provide examples of both α and θ that will illustrate these intuitions.

5 Syntactic and behavioural translations

To characterize morphisms of distributive laws in terms of rules, premises, literals etc., we first provide straightforward complete characterizations of natural transformations α and θ introduced in the previous section.

5.1 Syntactic translations

▶ **Definition 5.** For signatures Σ, Σ', a *syntactic translation* from Σ to Σ' consists of:

- a function $\alpha : \Sigma \to \Sigma'$ between the underlying sets of function symbols,
- for each $\mathtt{f} \in \Sigma$, a function $\alpha_{\mathtt{f}} : \{1, \ldots, \sharp\alpha(\mathtt{f})\} \to \{1, \ldots, \sharp\mathtt{f}\}$.

For any set X, a syntactic translation α determines a function $\alpha_X : \Sigma X \to \Sigma' X$ by:

$$\alpha_X(\mathtt{f}(x_1, \ldots, x_{\sharp f})) = \alpha(\mathtt{f})\big(x_{\alpha_{\mathtt{f}}(1)}, \ldots, x_{\alpha_{\mathtt{f}}(\sharp\alpha(\mathtt{f}))}\big). \tag{5}$$

We abuse the notation by denoting different entities by α, but this should not lead to any confusion.

▶ **Example 6.** For $\Sigma = \{\|\}$ with $\sharp(\|) = 2$, consider a syntactic translation from Σ to Σ that exchanges the arguments of $\|$, defined by $\alpha(\|) = \|$ and $\alpha_{\|}(1) = 2$, $\alpha_{\|}(2) = 1$. For any set X, this determines a function $\alpha_X : \Sigma X \to \Sigma X$ given by $\alpha_X(x \parallel y) = y \parallel x$ for $x, y \in X$.

Neither component of a syntactic translation is required to be an injective function. For example, consider $\Sigma = \{\mathtt{f}, \mathtt{g}\}$ and $\Sigma' = \{\mathtt{k}\}$ with $\sharp\mathtt{f} = \sharp\mathtt{g} = 1$ and $\sharp\mathtt{k} = 2$, and a syntactic translation from Σ to Σ' defined by: $\alpha(\mathtt{f}) = \alpha(\mathtt{g}) = \mathtt{k}$ and $\alpha_{\mathtt{f}}(1) = \alpha_{\mathtt{f}}(2) = \alpha_{\mathtt{g}}(1) = \alpha_{\mathtt{g}}(2) = 1$. This determines, for any X, a function $\alpha_X(\mathtt{f}(x)) = \alpha_X(\mathtt{g}(x)) = \mathtt{k}(x, x)$.

It is a standard exercise to prove that for any syntactic translation, the functions α_X defined by (5) form a natural transformation $\alpha : \Sigma \Longrightarrow \Sigma'$. Moreover, every natural transformation of this type arises this way from some syntactic translation.

5.2 Behavioural translations

Natural transformations $\theta : (\mathcal{P}_\omega -)^{L'} \Longrightarrow (\mathcal{P}_\omega -)^L$ are almost a special case of simple distributive laws $\lambda : \Sigma(\mathcal{P}_\omega -)^L \Longrightarrow (\mathcal{P}_\omega \Sigma -)^L$ considered in Section 3, for $\Sigma = \mathrm{Id}$; the only difference is the use of two sets of transition labels L, L'. Accordingly, specifications of such transformations look almost like degenerated cases of simple SOS specifications.

▶ **Definition 7.** A *behavioural translation* Θ (from L' to L) is a set of *behavioural rules*, i.e., expressions of the form

$$\frac{\{\mathtt{x} \xrightarrow{a_j} \mathtt{y}_j\}_{j=1..m} \qquad \{\mathtt{x} \overset{b_k}{\not\longrightarrow}\}_{k=1..l}}{\mathtt{x} \xrightarrow{c} \mathtt{y}} \tag{6}$$

where $m, l \in \mathbb{N}$; $\mathtt{x}$ and all $\mathtt{y}_j \in \mathcal{V}$ are all distinct; $\mathtt{y} \in \{\mathtt{y}_1, \ldots, \mathtt{y}_m\}$; $a_j, b_k \in L'$ and $c \in L$.

For any set X, rules are triggered by sets of X-literals just as in the case of SOS specifications; the only difference is that they infer X-literals rather than ΣX-literals, and that the triggering literals use labels from L' rather than L. For a behavioural translation Θ, $\Theta[\Phi]$ will denote the set of literals inferred from Φ by rules from Θ.

A behavioural translation Θ induces a function $\theta_X : (\mathcal{P}_\omega X)^{L'} \Longrightarrow (\mathcal{P}_\omega X)^L$ by:

$$\theta_X(\gamma)(c) = \{y \in X \mid (x \xrightarrow{c} y) \in \Theta[\Phi]\}, \text{ where } \Phi = \{x \xrightarrow{a} y \mid a \in L',\ y \in \gamma(a)\} \tag{7}$$

for a function $\gamma : L' \to \mathcal{P}_\omega X$, a $c \in L$, and any $x \in X$. As in (3), the value of θ_X does not depend on the choice of x.

▶ **Example 8.** Consider a totally ordered set of labels $L = \{a_1, a_2, \ldots, a_n\}$, with the intuition that a_i has a higher "priority" than a_j, for $i < j$. Consider the following behavioural translation from L to L:

$$\frac{\{\mathtt{x} \overset{a_j}{\not\longrightarrow}\}_{j<i} \qquad \mathtt{x} \xrightarrow{a_i} \mathtt{y}}{\mathtt{x} \xrightarrow{a_i} \mathtt{y}} \qquad \text{for } i = 1, \ldots, n.$$

Intuitively, this translation selects transitions with labels of the highest available priority. According to (7), this defines:

$$\theta_X(\gamma)(a_i) = \begin{cases} \gamma(a_i) & \text{if } \gamma(a_j) = \emptyset \text{ for all } j < i \\ \emptyset & \text{otherwise.} \end{cases}$$

▶ **Theorem 9.** *For any behavioural translation from L' to L, the functions θ_X defined in (7) form a natural transformation $\theta : (\mathcal{P}_\omega -)^{L'} \Longrightarrow (\mathcal{P}_\omega -)^L$. Moreover, every natural transformation of this type arises this way from some behavioural translation.*

Proof. This is essentially a special case of Theorem 3; the distinction between sets of labels L and L' does not change the proof in any essential way. ◀

Some behavioural translations arise from functions between transition labels. On one hand, a function $l : L \to L'$ gives rise to a natural transformation $\theta : (\mathcal{P}_\omega -)^{L'} \Longrightarrow (\mathcal{P}_\omega -)^{L}$:

$$\theta_X(\gamma)(c) = \gamma(l(c)) \text{ for } \gamma : L' \to \mathcal{P}_\omega X \text{ and } c \in L,$$

specified by rules $\dfrac{\mathtt{x} \xrightarrow{l(c)} \mathtt{y}}{\mathtt{x} \xrightarrow{c} \mathtt{y}}$ for $c \in L$. On the other hand, a function $k : L' \to L$ determines a transformation θ of the same type by:

$$\theta_X(\gamma)(c) = \bigcup\{\gamma(a) \mid k(a) = c\} \text{ for } \gamma : L' \to \mathcal{P}_\omega X \text{ and } c \in L,$$

specified by rules $\dfrac{\mathtt{x} \xrightarrow{a} \mathtt{y}}{\mathtt{x} \xrightarrow{k(a)} \mathtt{y}}$ for $a \in L'$.

6 Compatible translations

A morphism from an SOS specification Λ over syntax Σ and transition labels L, to a specification Λ' over syntax Σ' and transition labels L', should be presented by a syntactic translation from Σ to Σ' and a behavioural translation from L' to L, specified as in Sections 5.1-5.2, subject to a condition abstractly presented as the diagram (4). We shall now present that condition in elementary terms.

▶ **Definition 10.** A syntactic translation α together with a behavioural translation Θ are *compatible translations* from Λ to Λ' if for any set Φ of $\mathcal{V}$-literals with transition labels from L', a term $\mathtt{s} \in \Sigma\mathcal{V}$ and a label $c \in L$:

- for every $\mathtt{r} \in \Sigma\mathcal{V}$ such that $\mathtt{s} \xrightarrow{c} \mathtt{r}$ is in $\Lambda[\Theta[\Phi]]$, the literal $\alpha_\mathcal{V}(\mathtt{s}) \xrightarrow{c} \alpha_\mathcal{V}(\mathtt{r})$ is in $\Theta[\Lambda'[\Phi]]$, and
- for every $\mathtt{t} \in \Sigma'\mathcal{V}$ such that $\alpha_\mathcal{V}(\mathtt{s}) \xrightarrow{c} \mathtt{t}$ is in $\Theta[\Lambda'[\Phi]]$, there is some $\mathtt{r} \in \Sigma\mathcal{V}$ such that: $\alpha_\mathcal{V}(\mathtt{r}) = \mathtt{t}$ and the literal $\mathtt{s} \xrightarrow{c} \mathtt{r}$ is in $\Lambda[\Theta[\Phi]]$.

This bisimulation-like condition can be more succinctly written as:

$$\{\alpha_\mathcal{V}(\mathtt{r}) \in \Sigma'\mathcal{V} \mid (\mathtt{s} \xrightarrow{c} \mathtt{r}) \in \Lambda[\Theta[\Phi]]\} = \{\mathtt{t} \in \Sigma'\mathcal{V} \mid (\alpha_\mathcal{V}(\mathtt{s}) \xrightarrow{c} \mathtt{t}) \in \Theta[\Lambda'[\Phi]]\}. \tag{8}$$

Before we prove that compatible translations are equivalent to distributive law morphisms, let us elaborate the sets $\Lambda[\Theta[\Phi]]$ and $\Theta[\Lambda'[\Phi]]$, which arise from different ways of combining SOS rules with behavioural rules. In the following, a *$\mathcal{V}$-instance* of a rule (either an SOS one, or one of a behavioural translation) will mean a rule translated along some renaming function $\sigma : \mathcal{V} \to \mathcal{V}$, not necessarily bijective.

For a Λ and Θ as above, a *$\Lambda\Theta$-derivation* consists of:

- an $\mathcal{V}$-instance R of a rule in Λ,
- for each positive premise of R, a $\mathcal{V}$-instance of a rule in Θ with that premise as the conclusion.

A derivation is naturally presented as a simple tree-like structure built of rule instances. It has three types of premises: (a) positive premises of the Θ-rules, (b) negative premises of the Θ-rules, and (c) negative premises of the Λ-rule. Note that transition labels in premises of type (a) and (b) come from L', and of type (c) – from L.

We say that such a derivation is *triggered* by a set Φ of $\mathcal{V}$-literals with labels from L' if:

- every Θ-rule in the derivation is triggered by Φ, and
- for every premise $\mathtt{x} \not\xrightarrow{a}$ of type (c), there is no instance of a Θ-rule triggered by Φ and with conclusion of the form $\mathtt{x} \xrightarrow{a} \mathtt{y}$ for any $\mathtt{y} \in \mathcal{V}$.

In that case we say that the $\Sigma\mathcal{V}$-literal (with a transition label from L) that is the conclusion of the derivation, is $\Lambda\Theta$-*inferred* from Φ. It is straightforward to check that $\Lambda[\Theta[\Phi]]$ is the set of all literals $\Lambda\Theta$-inferred from Φ.

▶ **Example 11.** For $\Sigma = \{\otimes\}$ with $\sharp(\otimes) = 2$, and for $L = \{a_1, \ldots, a_n\}$, consider a specification Λ with rules:

$$\frac{\mathtt{x} \xrightarrow{a_i} \mathtt{x}' \quad \mathtt{y} \xrightarrow{a_i} \mathtt{y}'}{\mathtt{x} \otimes \mathtt{y} \xrightarrow{a_i} \mathtt{x}' \otimes \mathtt{y}'} \qquad \text{for } i = 1, \ldots, n.$$

For the behavioural translation Θ from Example 8, $\Lambda\Theta$-derivations are of the form:

$$\frac{\dfrac{\{\mathtt{w} \not\xrightarrow{a_j}\}_{j<i} \quad \mathtt{w} \xrightarrow{a_i} \mathtt{w}'}{\mathtt{w} \xrightarrow{a_i} \mathtt{w}'} \qquad \dfrac{\{\mathtt{z} \not\xrightarrow{a_j}\}_{j<i} \quad \mathtt{z} \xrightarrow{a_i} \mathtt{z}'}{\mathtt{z} \xrightarrow{a_i} \mathtt{z}'}}{\mathtt{w} \otimes \mathtt{z} \xrightarrow{a_i} \mathtt{w}' \otimes \mathtt{z}'}$$

for any (not necessarily distinct) $\mathtt{w}, \mathtt{w}', \mathtt{z}, \mathtt{z}' \in \mathcal{V}$ and $a_i \in L$. A set Φ of $\mathcal{V}$-literals triggers the derivation if and only if $\{\mathtt{w} \xrightarrow{a_i} \mathtt{w}',\ \mathtt{z} \xrightarrow{a_i} \mathtt{z}'\} \subseteq \Phi$ and Φ contains no literals with source $\mathtt{w}$ or $\mathtt{z}$ and label a_j for $j < i$.

On the other hand, a $\Theta\Lambda'$-*derivation* consists of:

- an $\Sigma'\mathcal{V}$-instance R of a rule in Θ,
- for each positive premise of R, a $\mathcal{V}$-instance of a rule in Λ' with that premise as the conclusion.

Such a derivation has three types of premises: (a) positive premises of the Λ'-rules, (b) negative premises of the Λ'-rules, and (c) negative premises of the Θ-rule. Note that transition labels in all these premises come from L'. However, while sources of premises of type (a) and (b) come from $\mathcal{V}$, sources of premises of type (c) come from $\Sigma'\mathcal{V}$. Such a derivation is *triggered* by a set Φ of $\mathcal{V}$-literals with transition labels from L', if:

- every Λ'-rule in the derivation is triggered by Φ, and
- for every premise $\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_{\sharp\mathtt{f}}) \not\xrightarrow{a}$ of type (c), there is no instance of a Λ'-rule triggered by Φ and with conclusion of the form $\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_{\sharp\mathtt{f}}) \xrightarrow{a} \mathtt{t}$ for any $\mathtt{t} \in \Sigma'\mathcal{V}$.

In that case we say that the $\Sigma'\mathcal{V}$-literal that is the conclusion of the derivation, is $\Theta\Lambda'$-*inferred* from Φ. As before, it is easy to check that the set of all such literals is equal to $\Theta[\Lambda'[\Phi]]$.

▶ **Example 12.** For Σ, L, $\Lambda' = \Lambda$ and Θ as in Example 11, $\Theta\Lambda$-derivations are of the form:

$$\frac{\{\mathtt{w} \otimes \mathtt{z} \not\xrightarrow{a_j}\}_{j<i} \qquad \dfrac{\mathtt{w} \xrightarrow{a_i} \mathtt{w}' \quad \mathtt{z} \xrightarrow{a_i} \mathtt{z}'}{\mathtt{w} \otimes \mathtt{z} \xrightarrow{a_i} \mathtt{w}' \otimes \mathtt{z}'}}{\mathtt{w} \otimes \mathtt{z} \xrightarrow{a_i} \mathtt{w}' \otimes \mathtt{z}'}$$

for any (not necessarily distinct) $\mathtt{w}, \mathtt{w}', \mathtt{z}, \mathtt{z}' \in \mathcal{V}$ and $a_i \in L$. A set Φ of $\mathcal{V}$-literals triggers the derivation if and only if $\{\mathtt{w} \xrightarrow{a_i} \mathtt{w}',\ \mathtt{z} \xrightarrow{a_i} \mathtt{z}'\} \subseteq \Phi$ and Λ infers from Φ no literals with source $\mathtt{w} \otimes \mathtt{z}$ and label a_j for $j < i$.

For example, $\Phi = \{\mathtt{w} \xrightarrow{a_2} \mathtt{w}',\ \mathtt{z} \xrightarrow{a_1} \mathtt{z}', \mathtt{z} \xrightarrow{a_2} \mathtt{z}'\}$ triggers the above derivation (for $i = 2$), but it does not trigger the $\Theta\Lambda$-derivation in Example 11. Indeed, it is easy to check that $\mathtt{w} \otimes \mathtt{z} \xrightarrow{a_2} \mathtt{w}' \otimes \mathtt{z}$ is $\Lambda\Theta$-inferred but not $\Theta\Lambda$-inferred from Φ so, according to Definition 10, the identity syntactic translation from Σ to itself, together with Θ from Example 8, do not form a compatible translation from Λ to itself.

We are now ready for our main characterization of distributive law morphisms:

▶ **Theorem 13.** *Translations α and Θ are compatible from Λ to Λ' if and only if α and the corresponding θ form a morphism of the corresponding distributive laws from λ to λ'.*

Proof. The diagram in Definition 4 states that two composite natural transformations from $\Sigma B'$ to $B\Sigma'$ are equal. It is not difficult to see that one can equivalently ask for their components at $\mathcal{V}$ to be equal. Indeed, in general, for any natural transformations $\phi, \psi : F \Longrightarrow G$ between functors on **Set**, if F is finitary and G preserves monomorphisms then for any infinite set $\mathcal{V}$, if $\phi_{\mathcal{V}} = \psi_{\mathcal{V}}$ then $\phi = \psi$. All functors considered in this paper are finitary and preserve monomorphisms, therefore we can replace the diagram in Definition 4 by its component at $\mathcal{V}$:

$$\begin{tikzcd}
\Sigma B'\mathcal{V} \arrow[r, "\Sigma\theta_{\mathcal{V}}"] \arrow[dr, "\alpha_{B'\mathcal{V}}"'] & \Sigma B\mathcal{V} \arrow[r, "\lambda_{\mathcal{V}}"] & B\Sigma\mathcal{V} \arrow[dr, "B\alpha_{\mathcal{V}}"] & \\
& \Sigma' B'\mathcal{V} \arrow[r, "\lambda'_{\mathcal{V}}"'] & B'\Sigma'\mathcal{V} \arrow[r, "\theta_{\Sigma'\mathcal{V}}"'] & B\Sigma'\mathcal{V}
\end{tikzcd} \tag{9}$$

and ask for it to commute. To this end, consider an arbitrary $\mathbf{A} = \mathtt{f}(\gamma_1, \ldots, \gamma_n) \in \Sigma B'\mathcal{V}$ where $n = \sharp\mathtt{f}$, and denote:

$$\mathbf{B} = \Sigma\theta_{\mathcal{V}}(\mathbf{A}) \in \Sigma B\mathcal{V}, \qquad \mathbf{C} = \lambda_{\mathcal{V}}(\mathbf{B}) \in B\Sigma\mathcal{V}, \qquad \mathbf{D} = B\alpha_{\mathcal{V}}(\mathbf{C}) \in B\Sigma'\mathcal{V},$$
$$\mathbf{E} = \alpha_{B'\mathcal{V}}(\mathbf{A}) \in \Sigma' B'\mathcal{V}, \qquad \mathbf{F} = \lambda'_{\mathcal{V}}(\mathbf{E}) \in B'\Sigma'\mathcal{V}, \qquad \mathbf{G} = \theta_{\Sigma'\mathcal{V}}(\mathbf{F}) \in B\Sigma'\mathcal{V}.$$

Our goal is to show that α and Θ are compatible if and only if $\mathbf{D} = \mathbf{G}$, for any $\mathbf{A}$. To this end, we unfold the definition of θ according to (7) to obtain:

$$\mathbf{B} = \mathtt{f}(\delta_1, \ldots, \delta_n) \qquad \text{where } \delta_i(b) = \{\mathtt{y} \in X \mid (\mathtt{x}_i \xrightarrow{b} \mathtt{y}) \in \Theta[\Phi_i]\} \text{ for any } b \in L,$$
$$\Phi_i = \{\mathtt{x}_i \xrightarrow{a} \mathtt{y} \mid a \in L',\ \mathtt{y} \in \gamma_i(a)\}$$

where for each $i = 1..n$ a distinct variable $\mathtt{x}_i \in \mathcal{V}$ is chosen. Further, we unfold the definition of λ using the same variables $\mathtt{x}_1, \ldots, \mathtt{x}_n$ according to (3), to get:

$$\mathbf{C}(c) = \{\mathtt{r} \in \Sigma\mathcal{V} \mid (\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_n) \xrightarrow{c} \mathtt{r}) \in \Lambda[\Upsilon]\}$$
$$\text{where } \Upsilon = \{\mathtt{x}_i \xrightarrow{b} \mathtt{y} \mid 1 \le i \le n,\ b \in L,\ \mathtt{y} \in \delta_i(b)\} = \bigcup_{i=1}^{n} \Theta[\Phi_i] = \Theta\left[\bigcup_{i=1}^{n} \Phi_i\right]$$

for any $c \in L$. The last equality holds by definition of Θ, since literals in distinct Φ_i have distinct sources, and all premises in any single Θ-rule have the same source. Denoting $\Phi = \bigcup_{i=1}^{n} \Phi_i$, we further obtain

$$\mathbf{D}(c) = \{\alpha_{\mathcal{V}}(\mathtt{r}) \in \Sigma'\mathcal{V} \mid (\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_n) \xrightarrow{c} \mathtt{r}) \in \Lambda[\Theta[\Phi]]\}$$

for any $c \in L$. For the other side of the diagram, put $\mathtt{g} = \alpha(\mathtt{f})$ and $m = \sharp\mathtt{g}$; then $\mathbf{E} = \mathtt{g}(\gamma_{\alpha_{\mathtt{f}}(1)}, \ldots, \gamma_{\alpha_{\mathtt{f}}(m)})$. Further, unfold according to (3) the definition of λ' using variables $\mathtt{x}_{\alpha_{\mathtt{f}}(1)}, \ldots, \mathtt{x}_{\alpha_{\mathtt{f}}(m)}$ where each $\mathtt{x}_i$ was chosen above, to obtain:

$$\mathbf{F}(b) = \{\mathtt{t} \in \Sigma'\mathcal{V} \mid (\mathtt{g}(\mathtt{x}_{\alpha_{\mathtt{f}}(1)}, \ldots, \mathtt{x}_{\alpha_{\mathtt{f}}(m)}) \xrightarrow{b} \mathtt{t}) \in \Lambda'[\Psi]\}$$
$$\text{where } \Psi = \{\mathtt{x}_{\alpha_{\mathtt{f}}(i)} \xrightarrow{a} \mathtt{y} \mid 1 \le i \le m,\ a \in L',\ \mathtt{y} \in \gamma_{\alpha_{\mathtt{f}}(i)}(a)\}$$

for any $b \in L'$. Observe that $\Psi \subseteq \Phi$, and Ψ coincides with Φ when restricted to literals whose sources are among $\mathtt{x}_{\alpha_{\mathtt{f}}(1)}, \ldots, \mathtt{x}_{\alpha_{\mathtt{f}}(m)}$ (indeed, $\Psi = \Phi$ if $\alpha_{\mathtt{f}} : m \to n$ is surjective). This implies that $\Lambda'[\Psi]$ coincides with $\Lambda'[\Phi]$ when restricted to literals with source $\mathtt{g}(\mathtt{x}_{\alpha_{\mathtt{f}}(1)}, \ldots, \mathtt{x}_{\alpha_{\mathtt{f}}(m)}))$. As a result, we may write:

$$\mathbf{F}(b) = \{\mathtt{t} \in \Sigma'\mathcal{V} \mid (\mathtt{g}(\mathtt{x}_{\alpha_{\mathtt{f}}(1)}, \ldots, \mathtt{x}_{\alpha_{\mathtt{f}}(m)}) \xrightarrow{b} \mathtt{t}) \in \Lambda'[\Phi]\}.$$

Finally, we unfold the definition of θ according to (7), using $\mathtt{g}(\mathtt{x}_{\alpha_\mathtt{f}(1)},\ldots,\mathtt{x}_{\alpha_\mathtt{f}(m)}) \in \Sigma'\mathcal{V}$ as the variable, to obtain:

$$\mathbf{G}(c) = \{\mathtt{t} \in \Sigma'\mathcal{V} \mid (\mathtt{g}(\mathtt{x}_{\alpha_\mathtt{f}(1)},\ldots,\mathtt{x}_{\alpha_\mathtt{f}(m)}) \xrightarrow{c} \mathtt{t}) \in \Theta[\Xi]\}$$

$$\text{where } \Xi = \{(\mathtt{g}(\mathtt{x}_{\alpha_\mathtt{f}(1)},\ldots,\mathtt{x}_{\alpha_\mathtt{f}(m)}) \xrightarrow{b} \mathtt{t}) \mid b \in L',\ \mathtt{t} \in \mathbf{F}(b)\} = \Lambda'[\Phi].$$

Putting it all together, the diagram (9) commutes if and only if, for every $\mathtt{f}(\gamma_1,\ldots,\gamma_n) \in \Sigma B\mathcal{V}$ and every $c \in L$:

$$\{\alpha_\mathcal{V}(\mathtt{r}) \in \Sigma'\mathcal{V} \mid (\mathtt{f}(\mathtt{x}_1,\ldots,\mathtt{x}_n) \xrightarrow{c} \mathtt{r}) \in \Lambda[\Theta[\Phi]]\} = \{\mathtt{t} \in \Sigma'\mathcal{V} \mid (\mathtt{g}(\mathtt{x}_{\alpha_\mathtt{f}(1)},\ldots,\mathtt{x}_{\alpha_\mathtt{f}(m)}) \xrightarrow{c} \mathtt{t}) \in \Theta[\Lambda'[\Phi]]\} \quad (10)$$

$$\text{where } \Phi = \{\mathtt{x}_i \xrightarrow{a} \mathtt{y} \mid 1 \le i \le n,\ a \in L',\ \mathtt{y} \in \gamma_i(a)\}. \quad (11)$$

It is easy to see that (10) is implied by the condition (8) of the definition of compatible translation, therefore if α and Θ form a compatible translation from Λ to Λ' then the diagram (9) commutes.

For the implication from (10) to (8), consider any $\mathtt{s} = \mathtt{f}(\mathtt{x}_1,\ldots,\mathtt{x}_n) \in \Sigma\mathcal{V}$, any $c \in L$ and any set Ψ of $\mathcal{V}$-literals. Define $\gamma_i(a) = \{y \in \mathcal{V} \mid (\mathtt{x}_i \xrightarrow{a} y) \in \Psi\}$ for $1 \le i \le n$ and $a \in L'$, and define Φ from the γ_i as in (11). Clearly $\Phi \subseteq \Psi$; moreover, Φ and Ψ coincide on literals whose sources are among the $\mathtt{x}_i$. As a result:

- $\Lambda[\Theta[\Phi]]$ and $\Lambda[\Theta[\Psi]]$ coincide on literals with source $\mathtt{f}(\mathtt{x}_1,\ldots,\mathtt{x}_n)$, and
- $\Theta[\Lambda'[\Phi]]$ and $\Theta[\Lambda'[\Psi]]$ coincide on literals with source $\mathtt{g}(\mathtt{x}_{\alpha_\mathtt{f}(1)},\ldots,\mathtt{x}_{\alpha_\mathtt{f}(m)})$.

The implication from (10) to (8) follows immediately. ◀

▶ Remark. It is important to note that the defining property of compatible translations, Definition 10, is decidable for given Λ, Λ', α and Θ. First, for a fixed finite set Φ of literals it is possible to compute $\Lambda[\Theta[\Phi]]$ and $\Theta[\Lambda'[\Phi]]$ by checking all combinations of Λ-, Λ'- and Θ-rules; each rule can have infinitely many $\mathcal{V}$-instances, but one only needs to consider those instances where all variables are present in Φ, which only leaves finitely many cases to check.

Moreover, one may restrict attention to finitely many sets Φ, all of them finite. Indeed, if a literal with source $\mathtt{s}$ is $\Theta\Lambda$- or $\Lambda\Theta$-inferred from Φ, then the derivation only depends on $\mathcal{V}$-literals with sources that are present in $\mathtt{s}$; this gives a bound on the number of different source variables in Φ's worth considering. The number of literals in Φ with a particular source variable can be bound by the number of premises with the same source in a $\Theta\Lambda$- or $\Lambda\Theta$-derivation; this gives a computable bound on the size of Φ worth checking. Finally, the condition of Definition 10 is invariant with respect to bijective renaming of variables. Altogether, this gives an effective procedure for checking whether given translations form a valid distributive law morphism.

7 Extensions

The framework of simple SOS specifications is very restrictive, and covers very few interesting examples of operational semantics. Right from the beginning [13], the distributive law approach to SOS was designed to cover far more general classes of specifications. So far in this paper we only treated simple SOS, to explain the general idea of presenting distributive law morphisms in a relatively basic setting. In this section we sketch two extensions of that basic setting: to GSOS specifications, and to extended syntactic translations. Fortunately, as we shall see, these extensions require little effort and everything works essentially as before.

7.1 GSOS specifications

A *GSOS law* of Σ over B is a natural transformation $\lambda : \Sigma(\mathrm{Id} \times B) \Longrightarrow B\Sigma^*$, where Σ^* is the (underlying functor of) the free monad over Σ. Refer to [13] to see a notion of bialgebra for such laws, or [9] to see how they are equivalent to distributive laws of the copointed functor $\mathrm{Id} \times B$ over the monad Σ^*. Those results are crucial for the abstract theory of GSOS laws, but not necessary for understanding of our elementary development.

The following, introduced in [3], is a generalization of Definition 1:

▶ **Definition 14.** A *GSOS specification* Λ is finite set of *GSOS rules*, i.e., expressions of the form

$$\frac{\left\{\mathtt{x}_{i_j} \xrightarrow{a_j} \mathtt{y}_j\right\}_{j=1..m} \quad \left\{\mathtt{x}_{i_k} \not\xrightarrow{b_k}\right\}_{k=1..l}}{\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_{\sharp\mathtt{f}}) \xrightarrow{c} \mathtt{t}} \tag{12}$$

where: $\mathtt{f} \in \Sigma$; $m, l \in \mathbb{N}$; all $i_j, i_k \in \{1, \ldots, \sharp\mathtt{f}\}$; all $\mathtt{x}_i, \mathtt{y}_j \in \mathcal{V}$ are pairwise distinct, $a_j, b_k, c \in L$; and $\mathtt{t}$ is a Σ-term whose all variables come from the $\mathtt{x}_i$ and $\mathtt{y}_j$.

GSOS rules generalize simple SOS rules in that their targets $\mathtt{t}$ are arbitrary terms rather than single operations from Σ, and in that their source variables $\mathtt{x}_i$ are allowed in the target.

The notions of triggering rules and inferred literals $\Lambda[\Phi]$ are as for simple SOS specifications, except that now targets of inferred literals are arbitrary Σ-terms. A GSOS specification Λ determines a function $\lambda_X : \Sigma(X \times (\mathcal{P}_\omega X)^L) \to (\mathcal{P}_\omega \Sigma^* X)^L$ as in (3):

$$\begin{aligned}\lambda_X(\mathtt{f}(x_1, \gamma_1, \ldots, x_{\sharp\mathtt{f}}, \gamma_{\sharp\mathtt{f}}))(c) = \big\{t \in \Sigma^* X \mid (\mathtt{f}(x_1, \ldots, x_{\sharp\mathtt{f}}) \xrightarrow{c} t) \in \Lambda[\Phi]\big\}, \text{ where}\\ \Phi = \{x_i \xrightarrow{a} y \mid 1 \le i \le \sharp\mathtt{f},\ a \in L,\ y \in \gamma_i(a)\},\end{aligned}$$

except this definition is actually slightly simpler than (3), since the values $x_1, \ldots, x_{\sharp\mathtt{f}}$ need not be chosen arbitrarily, as they are provided in the argument of λ_X.

A generalization of Theorem 3, which was actually the result stated in [13] and proved in [2], shows that GSOS specifications correspond to GSOS laws just as simple SOS specifications correspond to simple distributive laws.

By analogy to Definition 4, a morphism of GSOS laws is a pair of natural transformations $\alpha : \Sigma \Longrightarrow \Sigma'$ and $\theta : B' \Longrightarrow B$ such that

$$\begin{tikzcd}
\Sigma(\mathrm{Id} \times B') \arrow[r, Rightarrow, "\Sigma(\mathrm{id}\times\theta)"] \arrow[dr, Rightarrow, "\alpha(\mathrm{Id}\times B')"'] & \Sigma(\mathrm{Id} \times B) \arrow[r, Rightarrow, "\lambda"] & B\Sigma^* \arrow[dr, Rightarrow, "B\alpha^*"] & \\
& \Sigma'(\mathrm{Id} \times B') \arrow[r, Rightarrow, "\lambda'"'] & B'\Sigma'^* \arrow[r, Rightarrow, "\theta\Sigma'"'] & B\Sigma'^*
\end{tikzcd}$$

commutes, where $\alpha^* : \Sigma^* \Longrightarrow \Sigma'^*$ is the obvious inductive extension of α to all Σ-terms.

Definition 10 of compatible translations carries over to GSOS specifications almost verbatim, except that $\alpha_{\mathcal{V}}(\mathtt{r})$ needs to be replaced by $\alpha^*_{\mathcal{V}}(\mathtt{r})$, as now $\mathtt{r} \in \Sigma^*\mathcal{V}$ is an arbitrary term. Stated succinctly by analogy to (8), the compatibility condition becomes:

$$\{\alpha^*_{\mathcal{V}}(\mathtt{r}) \in \Sigma'^*\mathcal{V} \mid (\mathtt{s} \xrightarrow{c} \mathtt{r}) \in \Lambda[\Theta[\Phi]]\} = \{\mathtt{t} \in \Sigma'^*\mathcal{V} \mid (\alpha_{\mathcal{V}}(\mathtt{s}) \xrightarrow{c} \mathtt{t}) \in \Theta[\Lambda'[\Phi]]\}$$

for all $\mathtt{s} \in \Sigma\mathcal{V}$, $c \in L$ and Φ a set of $\mathcal{V}$-literals. Theorem 13 still holds with these changes, with a completely analogous proof.

7.2 Generalized syntactic translations

In practical examples of translations between operational specifications, one often wishes to interpret an operation from the source signature not as a single operation, but as a complex term over the target signature. A natural way to model such situations is to consider an extended definition of a syntactic translation as a natural transformation $\alpha : \Sigma \Longrightarrow \Sigma'^*$. Such a transformation can be presented much the same as in Section 5.1, by a function from Σ to Σ'-terms (over some fixed set of variables), together with a function $\alpha_{\mathtt{f}}$ for each $\mathtt{f} \in \Sigma$ as in Definition 5, where arity of complex terms is defined inductively in an obvious way.

▶ **Example 15.** Consider signatures $\Sigma = \{\|\}$ and $\Sigma' = \{\lfloor, +\}$, where all operators have arity 2 (so that $\Sigma X = X^2$ and $\Sigma' X = X^2 + X^2$). A translation that maps every term $\mathtt{x} \parallel y \in \Sigma X$ to $(x \lfloor y) + (y \lfloor x) \in \Sigma'^* X$ is formally defined by a function that maps the symbol $\|$ to a term $(1 \lfloor 2) + (3 \lfloor 4) \in \Sigma'^* \mathbb{N}$ together with a mapping $\alpha_{\|}(1) = \alpha_{\|}(4) = 1,\ \alpha_{\|}(2) = \alpha_{\|}(3) = 2$.

▶ **Example 16.** Consider signatures $\Sigma = \{\mathtt{p}\}$ with $\sharp\mathtt{p} = 3$ and $\Sigma' = \{\|\}$ with $\sharp(\|) = 2$. Two extended syntactic translations from Σ to Σ' come to mind, given by:

$$\alpha_X(\mathtt{p}(x, y, z)) = (x \parallel y) \parallel z \qquad \text{or} \qquad \alpha'_X(\mathtt{p}(x, y, z)) = x \parallel (y \parallel z), \qquad \text{for } x, y, z \in X.$$

Generalized syntactic translations make sense already in connection to morphisms of simple distributive laws, but they are more naturally considered in the context of GSOS laws. A GSOS law $\lambda : \Sigma(\mathrm{Id} \times B) \Longrightarrow B\Sigma^*$ extends, by induction on Σ-terms, to a natural transformation $\lambda^* : \Sigma^*(\mathrm{Id} \times B) \Longrightarrow B\Sigma^*$ (see [9] for a detailed study of this and related issues). It is natural to redefine a morphism between GSOS laws λ and λ' as an (extended) syntactic translation $\alpha : \Sigma \Longrightarrow \Sigma'^*$ and a natural transformation $\theta : B' \Longrightarrow B$ as before, such that

$$\begin{tikzcd}
\Sigma(\mathrm{Id} \times B') \arrow[r, Rightarrow, "\Sigma(\mathrm{id}\times\theta)"] \arrow[dr, Rightarrow, "\alpha(\mathrm{Id}\times B')"'] & \Sigma(\mathrm{Id} \times B) \arrow[r, Rightarrow, "\lambda"] & B\Sigma^* \arrow[dr, Rightarrow, "B\alpha^*"] & \\
 & \Sigma'^*(\mathrm{Id} \times B') \arrow[r, Rightarrow, "\lambda'^*"'] & B'\Sigma'^* \arrow[r, Rightarrow, "\theta\Sigma'^*"'] & B\Sigma'^*.
\end{tikzcd}$$

commutes, where $\alpha^* : \Sigma^* \Longrightarrow \Sigma'^*$ is the inductive extension of α to all Σ-terms.

A corresponding notion of compatible translations is straightforward to define, but a little less so than in Section 7.1. For a GSOS specification Λ, one defines Λ^*-*derivations* as well-formed trees built of instances of rules from Λ. For example, for Λ as in Example 11,

$$\dfrac{\dfrac{\mathtt{x} \xrightarrow{a_i} \mathtt{x}' \quad \mathtt{y} \xrightarrow{a_i} \mathtt{y}'}{\mathtt{x} \otimes \mathtt{y} \xrightarrow{a_i} \mathtt{x}' \otimes \mathtt{y}'} \qquad \mathtt{z} \xrightarrow{a_i} \mathtt{z}'}{(\mathtt{x} \otimes \mathtt{y}) \otimes \mathtt{z} \xrightarrow{a_i} (\mathtt{x}' \otimes \mathtt{y}') \otimes \mathtt{z}'}$$

is a valid derivation triggered by $\Phi = \{\mathtt{x} \xrightarrow{a_i} \mathtt{x}',\ \mathtt{y} \xrightarrow{a_i} \mathtt{y}',\ \mathtt{z} \xrightarrow{a_i} \mathtt{z}'\}$, and it infers the literal in the conclusion; the (usually infinite) set of all inferred literals is denoted by $\Lambda^*[\Phi]$.

Compatible translations are then defined similarly as in Definition 10, with Λ'-derivations replaced by Λ'^*-derivations. By analogy to (8), the compatibility condition becomes:

$$\{\alpha^*_{\mathcal{V}}(\mathtt{r}) \in \Sigma'^*\mathcal{V} \mid (\mathtt{s} \xrightarrow{c} \mathtt{r}) \in \Lambda[\Theta[\Phi]]\} = \{\mathtt{t} \in \Sigma'^*\mathcal{V} \mid (\alpha_{\mathcal{V}}(\mathtt{s}) \xrightarrow{c} \mathtt{t}) \in \Theta[\Lambda'^*[\Phi]]\} \qquad (13)$$

for all $\mathtt{s} \in \Sigma\mathcal{V}$, $c \in L$ and Φ a set of $\mathcal{V}$-literals. The corresponding version of Theorem 13 still holds; the proof is slightly more complex technically due to the presence of additional induction on Σ-terms, but no essentially new aspects arise in it.

▶ Remark. Although even for a finite Φ the set $\Theta[\Lambda'^*[\Phi]]$ will often be infinite, it will still be finite when restricted to literals with the source $\alpha_\mathcal{V}(\mathtt{s})$, for any fixed $\mathtt{s} \in \Sigma\mathcal{V}$. As a result, the above compatibility condition remains decidable.

8 Examples

▶ **Example 17** (Conservative extension)**.** Consider a signature Σ' and its subsignature Σ (i.e., $\Sigma \subseteq \Sigma'$ and the arities of Σ-symbols are matched in Σ'), and two GSOS specifications Λ and Λ' over Σ and Σ' respectively, with the same set L of transition labels. We say that Λ' is a *conservative extension* of Λ if $\Lambda \subseteq \Lambda'$ and if no rule from $\Lambda' \setminus \Lambda$ has its source from Σ.

Note that this notion is more restrictive than usual definitions of conservative extension considered in SOS theory [1]. There, global properties of specifications play a role; for example, a new rule with a source from Σ may be allowed in Λ' as long as it it has some positive premise whose label cannot possibly be matched by a conclusion of a Λ-rule.

Consider a trivial inclusion syntactic translation $\alpha : \Sigma \Longrightarrow \Sigma'$, and the identity behavioural translation Θ. It is easy to see that α and Θ form a compatible translation from Λ to Λ'. Indeed, for any $\mathtt{s} = \mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_n) \in \Sigma\mathcal{V}$ and $c \in L$, there is a correspondence between $\Lambda\Theta$- and $\Theta\Lambda'$-derivations:

$$\frac{\left\{\dfrac{\mathtt{x}_{i_j} \xrightarrow{a_j} \mathtt{y}_j}{\mathtt{x}_{i_j} \xrightarrow{a_j} \mathtt{y}_j}\right\}_{j=1..m} \quad \left\{\mathtt{x}_{i_k} \not\xrightarrow{b_k}\right\}_{k=1..l}}{\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_n) \xrightarrow{c} \mathtt{t}} \quad \text{vs.} \quad \frac{\dfrac{\left\{\mathtt{x}_{i_j} \xrightarrow{a_j} \mathtt{y}_j\right\}_{j=1..m} \quad \left\{\mathtt{x}_{i_k} \not\xrightarrow{b_k}\right\}_{k=1..l}}{\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_n) \xrightarrow{c} \mathtt{t}}}{\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_n) \xrightarrow{c} \mathtt{t}} \qquad (14)$$

that are triggered by the same sets Φ and infer the same conclusions, for any rule from Λ as in (12). No other $\Theta\Lambda'$-derivation for $\mathtt{f}$ is possible, by definition of conservative extension.

▶ **Example 18** (Nservative coextension)**.** Dually, consider a signature Σ, two sets of labels $L \subseteq L'$, and two GSOS specifications Λ and Λ' over Σ and over labels L and L', respectively. We say that Λ' is, for lack of a better name, an *nservative coextension* of Λ if $\Lambda \subseteq \Lambda'$ and if the conclusion of each rule from $\Lambda' \setminus \Lambda$ has a transition label from $L' \setminus L$.

Intuitively, just as a conservative extension (Example 17) defines behaviour for a new part of syntax while leaving the behaviour of the old syntax intact, an nservative coextension defines new aspects of behaviour (i.e., $L' \setminus L$-transitions) while leaving old aspects intact.

Consider the identity syntactic translation on Σ and a trivial behavioural translation Θ from L' to L specified by rules $\dfrac{\mathtt{x} \xrightarrow{c} \mathtt{y}}{\mathtt{x} \xrightarrow{c} \mathtt{y}}$ for $c \in L$. Again, it is easy to see that α and θ are a compatible translation from Λ to Λ'. Indeed, for any $\mathtt{s} = \mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_n) \in \Sigma\mathcal{V}$ and $c \in L$, a correspondence between $\Lambda\Theta$- and $\Theta\Lambda'$-derivations is exactly the same as in (14); again, no other $\Theta\Lambda'$-derivation is possible since $c \in L$ and there are no new rules in Λ' with c as the conclusion label.

In the next example, neither α nor θ is identity.

▶ **Example 19.** For $\Sigma = \{!\}$ with $\sharp(!) = 2$ and $L = \{a_1, \ldots, a_n\}$, consider Λ with rules:

$$\frac{\mathtt{x} \xrightarrow{a_i} \mathtt{x}' \quad (\mathtt{y} \not\xrightarrow{a_j})_{j<i}}{\mathtt{x}!\mathtt{y} \xrightarrow{a_i} \mathtt{x}'!\mathtt{y}} \qquad \frac{(\mathtt{x} \not\xrightarrow{a_j})_{j<i} \quad \mathtt{y} \xrightarrow{a_i} \mathtt{y}'}{\mathtt{x}!\mathtt{y} \xrightarrow{a_i} \mathtt{x}!\mathtt{y}'} \quad \text{for } i = 1, \ldots, n. \qquad (15)$$

Moreover, let $\Sigma' = \{\|\}$ with $\sharp(\|) = 2$ and, for the same L, let Λ' consist of rules:

$$\frac{\mathtt{x} \xrightarrow{a_i} \mathtt{x}'}{\mathtt{x} \parallel \mathtt{y} \xrightarrow{a_i} \mathtt{x}' \parallel \mathtt{y}} \qquad \frac{\mathtt{y} \xrightarrow{a_i} \mathtt{y}'}{\mathtt{x} \parallel \mathtt{y} \xrightarrow{a_i} \mathtt{x} \parallel \mathtt{y}'} \quad \text{for } i = 1, \ldots, n. \qquad (16)$$

Consider a syntactic translation from Σ to Σ' defined by $\alpha_X(x!y) = x \parallel y$, and the behavioural translation Θ from Example 8. Then $\Lambda\Theta$-derivations are of the form:

$$\frac{\dfrac{\left\{\mathtt{x}\not\xrightarrow{a_j}\right\}_{j<i} \qquad \mathtt{x}\xrightarrow{a_i}\mathtt{x}'}{\mathtt{x}\xrightarrow{a_i}\mathtt{x}'} \qquad \left\{\mathtt{y}\not\xrightarrow{a_j}\right\}_{j<i}}{\mathtt{x!y}\xrightarrow{a_i}\mathtt{x}'\mathtt{!y}} \qquad\qquad \frac{\left\{\mathtt{x}\not\xrightarrow{a_j}\right\}_{j<i} \qquad \dfrac{\left\{\mathtt{y}\not\xrightarrow{a_j}\right\}_{j<i} \qquad \mathtt{y}\xrightarrow{a_i}\mathtt{y}'}{\mathtt{y}\xrightarrow{a_i}\mathtt{y}'}}{\mathtt{x!y}\xrightarrow{a_i}\mathtt{x!y}'}$$

and $\Theta\Lambda'$-derivations are of the form:

$$\frac{\dfrac{\mathtt{x}\xrightarrow{a_i}\mathtt{x}'}{\mathtt{x}\parallel\mathtt{y}\xrightarrow{a_i}\mathtt{x}'\parallel\mathtt{y}} \qquad \left\{\mathtt{x}\parallel\mathtt{y}\not\xrightarrow{a_j}\right\}_{j<i}}{\mathtt{x}\parallel\mathtt{y}\xrightarrow{a_i}\mathtt{x}'\parallel\mathtt{y}} \qquad\qquad \frac{\left\{\mathtt{x}\parallel\mathtt{y}\not\xrightarrow{a_j}\right\}_{j<i} \qquad \dfrac{\mathtt{y}\xrightarrow{a_i}\mathtt{y}'}{\mathtt{x}\parallel\mathtt{y}\xrightarrow{a_i}\mathtt{x}\parallel\mathtt{y}'}}{\mathtt{x}\parallel\mathtt{y}\xrightarrow{a_i}\mathtt{x}\parallel\mathtt{y}'}$$

It is easy to see that for any Φ these derivations infer the same literals up to α, since a negative premise $\mathtt{x}\parallel\mathtt{y}\not\xrightarrow{a_j}$ holds for Φ if and only if both premises $\mathtt{x}\not\xrightarrow{a_j}$ and $\mathtt{y}\not\xrightarrow{a_j}$ hold. As a result, α with Θ form a morphism from Λ to Λ'.

Note that the identity syntactic translation together with Θ does *not* give a morphism from Λ' to itself, for reasons similar to Example 12. Indeed, $\Lambda'\Theta$-derivations are of the form:

$$\frac{\dfrac{\left\{\mathtt{x}\not\xrightarrow{a_j}\right\}_{j<i} \qquad \mathtt{x}\xrightarrow{a_i}\mathtt{x}'}{\mathtt{x}\xrightarrow{a_i}\mathtt{x}'}}{\mathtt{x}\parallel\mathtt{y}\xrightarrow{a_i}\mathtt{x}'\parallel\mathtt{y}} \qquad\qquad \frac{\dfrac{\left\{\mathtt{y}\not\xrightarrow{a_j}\right\}_{j<i} \qquad \mathtt{y}\xrightarrow{a_i}\mathtt{y}'}{\mathtt{y}\xrightarrow{a_i}\mathtt{y}'}}{\mathtt{x}\parallel\mathtt{y}\xrightarrow{a_i}\mathtt{x}\parallel\mathtt{y}'}$$

and for $\Phi = \{\mathtt{x}\xrightarrow{a_2}\mathtt{x}',\ \mathtt{y}\xrightarrow{a_1}\mathtt{y}'\}$ they infer the literal $\mathtt{x}\parallel\mathtt{y}\xrightarrow{a_2}\mathtt{x}'\parallel\mathtt{y}$, whereas $\Theta\Lambda'$-literals above do not.

The following example was considered also in [14].

▶ **Example 20.** Consider Σ and Σ' from Example 15, over the same set of labels L. Let Λ consist of rules:

$$\frac{\mathtt{x}\xrightarrow{a}\mathtt{x}'}{\mathtt{x}\parallel\mathtt{y}\xrightarrow{a}\mathtt{x}'\parallel\mathtt{y}} \qquad \frac{\mathtt{y}\xrightarrow{a}\mathtt{y}'}{\mathtt{x}\parallel\mathtt{y}\xrightarrow{a}\mathtt{y}'\parallel\mathtt{x}} \qquad \text{for } a\in L, \tag{17}$$

and let Λ' be the GSOS specification:

$$\frac{\mathtt{x}\xrightarrow{a}\mathtt{x}'}{\mathtt{x}+\mathtt{y}\xrightarrow{a}\mathtt{x}'} \qquad \frac{\mathtt{y}\xrightarrow{a}\mathtt{y}'}{\mathtt{x}+\mathtt{y}\xrightarrow{a}\mathtt{y}'} \qquad \frac{\mathtt{x}\xrightarrow{a}\mathtt{x}'}{\mathtt{x}\lfloor\mathtt{y}\xrightarrow{a}(\mathtt{x}'\lfloor\mathtt{y})+(\mathtt{y}\lfloor\mathtt{x}')} \qquad \text{for } a\in L.$$

Pick a syntactic translation is as in Example 15, and let Θ be the identity behavioural translation. $\Lambda\Theta$-derivations are very simple:

$$\frac{\dfrac{\mathtt{x}\xrightarrow{a}\mathtt{x}'}{\mathtt{x}\xrightarrow{a}\mathtt{x}'}}{\mathtt{x}\parallel\mathtt{y}\xrightarrow{a}\mathtt{x}'\parallel\mathtt{y}} \qquad\qquad \frac{\dfrac{\mathtt{y}\xrightarrow{a}\mathtt{y}'}{\mathtt{y}\xrightarrow{a}\mathtt{y}'}}{\mathtt{x}\parallel\mathtt{y}\xrightarrow{a}\mathtt{y}'\parallel\mathtt{x}}$$

$\Theta\Lambda'^*$-derivations are more interesting; the only ones for the term $\alpha_{\mathcal{V}}(\mathtt{x}\parallel\mathtt{y}) = (\mathtt{x}\lfloor\mathtt{y}) + (\mathtt{y}\lfloor\mathtt{x})$ are:

$$\frac{\dfrac{\dfrac{\mathtt{x}\xrightarrow{a}\mathtt{x}'}{\mathtt{x}\lfloor\mathtt{y}\xrightarrow{a}(\mathtt{x}'\lfloor\mathtt{y})+(\mathtt{y}\lfloor\mathtt{x}')}}{(\mathtt{x}\lfloor\mathtt{y})+(\mathtt{y}\lfloor\mathtt{x})\xrightarrow{a}(\mathtt{x}'\lfloor\mathtt{y})+(\mathtt{y}\lfloor\mathtt{x}')}}{(\mathtt{x}\lfloor\mathtt{y})+(\mathtt{y}\lfloor\mathtt{x})\xrightarrow{a}(\mathtt{x}'\lfloor\mathtt{y})+(\mathtt{y}\lfloor\mathtt{x}')} \qquad \frac{\dfrac{\dfrac{\mathtt{y}\xrightarrow{a}\mathtt{y}'}{\mathtt{y}\lfloor\mathtt{x}\xrightarrow{a}(\mathtt{y}'\lfloor\mathtt{x})+(\mathtt{x}\lfloor\mathtt{y}')}}{(\mathtt{x}\lfloor\mathtt{y})+(\mathtt{y}\lfloor\mathtt{x})\xrightarrow{a}(\mathtt{y}'\lfloor\mathtt{x})+(\mathtt{x}\lfloor\mathtt{y}')}}{(\mathtt{x}\lfloor\mathtt{y})+(\mathtt{y}\lfloor\mathtt{x})\xrightarrow{a}(\mathtt{y}'\lfloor\mathtt{x})+(\mathtt{x}\lfloor\mathtt{y}')}$$

It is easy to see that the condition (13) is satisfied, therefore α and Θ form a morphism from Λ to Λ'. Note the slight difference between the targets of rules (16) and (17). There seems to be no morphism from (16) to Λ', which suggests that the notion of distributive law morphism could perhaps be relaxed in a useful way. We leave this for future work.

▶ **Example 21.** Consider Σ, Σ' and α from Example 16. Let Λ over Σ consist of rules:

$$\frac{\mathtt{x} \xrightarrow{a} \mathtt{x'}}{\mathtt{p(x,y,z)} \xrightarrow{a} \mathtt{p(x',y,z)}} \qquad \frac{\mathtt{y} \xrightarrow{a} \mathtt{y'}}{\mathtt{p(x,y,z)} \xrightarrow{a} \mathtt{p(x,y',z)}} \qquad \frac{\mathtt{z} \xrightarrow{a} \mathtt{z'}}{\mathtt{p(x,y,z)} \xrightarrow{a} \mathtt{p(x,y,z')}} \qquad \text{for } a \in L,$$

and let Λ' over Σ' be defined by rules as in (16). For the identity Θ, there is an easy correspondence between $\Lambda\Theta$-derivations and $\Theta\Lambda'$-derivations, for example:

$$\frac{\dfrac{\mathtt{x} \xrightarrow{a} \mathtt{x'}}{\mathtt{x} \xrightarrow{a} \mathtt{x'}}}{\mathtt{p(x,y,z)} \xrightarrow{a} \mathtt{p(x',y,z)}} \qquad \text{vs.} \qquad \frac{\dfrac{\dfrac{\mathtt{x} \xrightarrow{a} \mathtt{x'}}{\mathtt{x} \parallel \mathtt{y} \xrightarrow{a} \mathtt{x'} \parallel \mathtt{y}}}{(\mathtt{x} \parallel \mathtt{y}) \parallel \mathtt{z} \xrightarrow{a} (\mathtt{x'} \parallel \mathtt{y}) \parallel \mathtt{z}}}{(\mathtt{x} \parallel \mathtt{y}) \parallel \mathtt{z} \xrightarrow{a} (\mathtt{x'} \parallel \mathtt{y}) \parallel \mathtt{z}}$$

which shows that α with Θ form a morphism from Λ and Λ'. By analogy, α' with Example 16 forms a similar morphism with Θ. This proves that the equation $(x \parallel y) \parallel z = x \parallel (y \parallel z)$ holds up to bisimilarity in the transition system induced by Λ'. This suggests a connection to quotients of distributive laws studied in [4]; we leave this for future work.

References

1 L. Aceto, W. J. Fokkink, and C. Verhoef. Structural operational semantics. In J. A. Bergstra, A. Ponse, and S. Smolka, editors, *Handbook of Process Algebra*, pages 197–292. Elsevier, 2002.

2 F. Bartels. *On Generalised Coinduction and Probabilistic Specification Formats.* PhD dissertation, CWI, Amsterdam, 2004.

3 B. Bloom, S. Istrail, and A. Meyer. Bisimulation can't be traced. *Journal of the ACM*, 42:232–268, 1995.

4 M. Bonsangue, H. H. Hansen, A. Kurz, and J. Rot. Presenting distributive laws. In *Procs. CALCO'13*, volume 8089 of *LNCS*, pages 95–109, 2013.

5 M. Hennessy, W. Li, and G. D. Plotkin. A first attempt at translating CSP into CCS. In *Proc. Second International Conference on Distributed Systems*, pages 105–115, 1981.

6 B. Klin. Bialgebraic methods and modal logic in structural operational semantics. *Information and Computation*, 207:237–257, 2009.

7 B. Klin. Bialgebras for structural operational semantics: An introduction. *Theoretical Computer Science*, 412(38):5043–5069, 2011. CMCS Tenth Anniversary Meeting.

8 B. Klin and B. Nachyła. Distributive laws and decidable properties of SOS specifications. In *Procs. EXPRESS/SOS'14*, volume 160 of *ENTCS*, pages 79–93, 2014.

9 M. Lenisa, J. Power, and H. Watanabe. Category theory for operational semantics. *Theoretical Computer Science*, 327(1-2):135–154, 2004.

10 G. D. Plotkin. A structural approach to operational semantics. *Journal of Logic and Algebraic Programming*, 60-61:17–139, 2004.

11 J. Power and H. Watanabe. Combining a monad and a comonad. *Theor. Comput. Sci.*, 280:137–162, 2002.

12 J. J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249:3–80, 2000.

13 D. Turi and G. D. Plotkin. Towards a mathematical operational semantics. In *Proc. LICS'97*, pages 280–291. IEEE Computer Society Press, 1997.

14 H. Watanabe. Well-behaved translations between structural operational semantics. *ENTCS*, 65, 2002.

Approximation of Nested Fixpoints – A Coalgebraic View of Parametric Datatypes

Alexander Kurz[1], Alberto Pardo[2], Daniela Petrişan[3], Paula Severi[1], and Fer-Jan de Vries[1]

1 **Department of Computer Science, University of Leicester, UK**
2 **Instituto de Computación, Universidad de la República, Uruguay**
3 **Radboud University, The Netherlands**

Abstract

The question addressed in this paper is how to correctly approximate infinite data given by systems of simultaneous corecursive definitions. We devise a categorical framework for reasoning about regular datatypes, that is, datatypes closed under products, coproducts and fixpoints. We argue that the right methodology is on one hand coalgebraic (to deal with possible non-termination and infinite data) and on the other hand 2-categorical (to deal with parameters in a disciplined manner). We prove a coalgebraic version of Bekič lemma that allows us to reduce simultaneous fixpoints to a single fix point. Thus a possibly infinite object of interest is regarded as a final coalgebra of a many-sorted polynomial functor and can be seen as a limit of finite approximants. As an application, we prove correctness of a generic function that calculates the approximants on a large class of data types.

1998 ACM Subject Classification F.3.2 Semantics of Programming Languages

Keywords and phrases coalgebra, Bekič lemma, infinite data, functional programming, type theory

Digital Object Identifier 10.4230/LIPIcs.CALCO.2015.205

1 Introduction

As forcefully argued in [4], the initial algebra semantics underpins much of the theory of functional programming languages, allowing for structured recursion over data structures. As long as we restrict our attention to the fragment of Haskell programs that actually terminate, this is indeed enough. But what happens if we want to take into account infinite computations? Recursively defined datatypes in Haskell are inherently *coinductive* and, as we shall see in this paper, further complications arise when several nested fixpoints are involved.

Let us consider the parametrised datatype of streams and two functions on integers `const` and `matrix` defined as follows.

```
data Stream a = Cons a (Stream a)

const :: Int -> Stream Int
const n = Cons n (const n)
matrix :: Int -> Stream (Stream Int)
matrix n = Cons (const n) (matrix (n+1))
```

The function `const` takes an integer `n` and evaluates to an *infinite* normal form, the constant stream `n:n:n...` where we abbreviate `Cons` by a colon to improve readability. The function

6th International Conference on Algebra and Coalgebra in Computer Science (CALCO'15).
Editors: Lawrence S. Moss and Pawel Sobocinski; pp. 205–220
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

matrix evaluated at 0 also yields an infinite computation whose normal form is the stream of streams

```
(0:0:0:...):(1:1:1:...):(2:2:2:...):...
```

This is obtained as the limit of the following reduction sequence:

```
matrix  0  →  Cons  (const 0)  (matrix  1)
           →  Cons  (Cons  0  (const  0))  (Cons  (const 1)  (matrix  2))
           →  ...
```

With finite resources printing an infinite stream like (0:0...) is impossible. One could try to see this stream as the limit of an ω long sequence of growing finite terms

```
0, 0:0, 0:0:0, ...
```

How would this work for the infinite normal form of matrix 0? Continuing in the same manner as before would give a converging sequence of length ω^2 of prefixes:

```
0             0:0             ...  (0:0:...)
(0:0:...):1   (0:0:...):1:1   ...  (0:0:...):(1:1:...)
...
```

Note that there is no clue that after 0:0:... the sequence will continue and indeed a naive Haskell implementation would evaluate only the head of matrix 0 and thus would render only a sequence of 0$'s$. In the above sequence of approximants we are missing any indication where the terms are incomplete. A much better sequence of approximants would be

```
•, (0:•):•, (0:0:•):(1:•):•, (0:0:0:•):(1:1:•):(2:•):•, ...
```

Each of these truncations (or approximating terms) is finite and can in principle be printed, as we show in Section 4. This raises interesting Haskell questions: given a nested datatype T, how to define a generic datatype B T for such truncations and how to define a generic function trunc of type Nat -> T -> B T that allows us to print these approximants. Moreover the above sequence of truncations is ω-long, suggesting that conceptually we transformed the two nested fixpoints involved in the datatype Stream(Stream(Int)) into a single fixpoint. Let us analyse this problem from a category theoretic perspective. Consider a category $\mathcal{C}$, assumed for simplicity complete and cocomplete. The datatype Stream(X) depending on the parameter X in $\mathcal{C}$ can be regarded as the final coalgebra, i.e. the greatest fixpoint νF_X of a functor $F_X : \mathcal{C} \to \mathcal{C}$ given by $F_X Z = X \times Z$. Therefore the datatype Stream(Stream(X)) is isomorphic to a final coalgebra of $F_{\mathtt{Stream}(X)}$, that is,

$$\nu Y.(\nu Z.X \times Z) \times Y. \qquad (1)$$

Performing ω reductions of matrix 0 following the leftmost strategy in the example above is roughly the same as approximating this final coalgebra by iterating only on Z. But the sequence obtained in this way does not converge to the infinite normal form. On the other hand, iterating on Y alone does not work either because $\nu Z.X \times Z$ contains infinite streams which are not printable. The solution is to iterate on Y and Z simultaneously. This can be done by using a version of the Bekič lemma which states that the final coalgebra (1) can be obtained as the first projection of the final coalgebra for the endofunctor on $\mathcal{C}^2$ given by

$$\begin{pmatrix} Y \\ Z \end{pmatrix} \mapsto \begin{pmatrix} Z \times Y \\ X \times Z \end{pmatrix}.$$

Since this is a polynomial functor, a final coalgebra can be obtained as a limit of the final ω-chain in $\mathcal{C}^2$. Thus, the problem of obtaining an ω-long sequence of truncations for an element of a datatype containing several nested fixpoints can be solved by approximating instead elements of a final coalgebra for a polynomial functor on $\mathcal{C}^n$ for a positive integer n.

Another question is how to deal with parameters in a coherent manner. We saw that the parametrised datatype $\mathtt{Stream}(X)$ can be obtained as the fixpoint νF_X of a $\mathcal{C}$-endofunctor. But `Stream` is itself a functor, and this can be *proved* using the universal property of the final coalgebra. One can define its action on arrows (the so called `map` function), using a mediating coalgebra morphism arising from finality (the `unfold` map).

However, a more systematic approach to defining the parametrised datatype is to consider families of fixpoints in one go. So instead of considering a $\mathcal{C}$-endofunctor F_X with a parameter we consider the bifunctor $F : \mathcal{C}^2 \to \mathcal{C}$ given by $F(X, Y) = X \times Y$. This induces the endofunctor $F \circ \langle 1, - \rangle$ on the category $[\mathcal{C}, \mathcal{C}]$ of endofunctors on $\mathcal{C}$, given by $G \mapsto F \circ \langle 1, G \rangle$, as in [20]. The functor `Stream` can be defined as the greatest fixpoint of the higher-order functor $F \circ \langle 1, - \rangle : [\mathcal{C}, \mathcal{C}] \to [\mathcal{C}, \mathcal{C}]$, where we write 1 for identity functors.

$$\begin{tikzcd}
\mathcal{C} \arrow[rr, "\mathtt{Stream}"] \arrow[dr, bend right, "{\langle 1,\mathtt{Stream}\rangle}"'] & {} \arrow[d, Rightarrow, "\mathsf{unfold}"] & \mathcal{C} \\
& \mathcal{C} \times \mathcal{C} \arrow[ur, bend right, "F"'] &
\end{tikzcd}$$

We have a canonical natural transformation unfold and for every X in $\mathcal{C}$ the morphism $\mathsf{unfold}_X : \mathtt{Stream}(X) \to X \times \mathtt{Stream}(X)$ is the structure map of the final F_X-coalgebra.

Higher-order functors have been used in [6, 21] to obtain a categorical semantics for nested datatypes, that is, parametrised datatypes defined inductively and in whose declarations the type parameter changes. This approach – inherently 2-categorical – is essential for formalising the relation between the different categories of coalgebras whose final objects are considered.

It remains to understand what is the precise formulation of the Bekič rule that we can apply. The result that allows to transform nested fixpoints into a single one is originally due to Bekič, see [16], and was formulated in terms of least fixpoints of continuous maps on cpo's. Categorical fixpoints rules have been established by Lehmann and Smyth [20] and, under the assumption of algebraic completeness, by Freyd [13] and Fiore [12]. Stronger versions of Freyd's results are given in [2]. We are also indebted to an unpublished note of Pitts [22] which develops a 2-categorical calculus for fixpoints and covers, albeit without proof, simultaneous least fixpoints.

The 2-categorical theory of nested initial algebra has been developed in [7, 11]. Of course these results can be recast in the dual coalgebraic setting. In particular the dual of the Bekič rule we consider in this paper is called the "pairing identity" in [7]. Nevertheless, our proof of this identity (see Theorem 3 and Corollary 4) is rather different and relies on exhibiting some adjunctions between categories of coalgebras (Lemma 5 and Lemma 6). The latter results are interesting in their own right and can also be used for reducing multi-sorted coalgebras to one-sorted ones. The paper is organised as follows. Section 2 establishes adjunctions between several categories of coalgebras and as a consequence we obtain a 2-categorical proof of the Bekič rule. Section 3 explains how to obtain truncations of elements of a final coalgebra. Section 4 gives a generic implementation of the truncations as a function that can operate on a large class of data types and applies the theory developed in the previous sections to prove its correctness.

2 A coalgebraic treatment of the Bekič rule

As argued in the introduction, we need higher-order functors to deal with parametricity, and thus our setting is 2-categorical. For the convenience of the reader we briefly recall basic definitions, but we emphasise that very little of this theory is required to understand our results. Moreover, the basic example of a 2-category is Cat – the category of small categories. The reader unfamiliar with 2-categories is asked to instantiate *0-cell* with *category* (or data type), *1-cell* with *functor* (or parametrised type) and *2-cell* with *natural transformation* (or polymorphic function). Formally, a 2-category $\mathbb{C}$ consists of the following data:

- a class of objects or *0-cells*, denoted $A, B, C, \ldots$
- for any 0-cells A, B a category $\mathbb{C}(A, B)$. Objects of $\mathbb{C}(A, B)$ are called *1-cells* and are denoted by $f : A \to B$, while morphisms in $\mathbb{C}(A, B)$, usually denoted by $\alpha : f \Rightarrow g : A \to B$, or simply $\alpha : f \Rightarrow g$, are called *2-cells*. Composition in $\mathbb{C}(A, B)$ is denoted by $\circ$ and is called *vertical composition*. The identity arrow on $f : A \to B$ will be denoted 1_f, or sometimes just f.
- for any 0-cells A, B, C a functor $* : \mathbb{C}(A, B) \times \mathbb{C}(B, C) \to \mathbb{C}(A, C)$, called *horizontal composition*. The horizontal composition of 2-cells $\alpha : f \Rightarrow f' : A \to B$ and $\beta : g \Rightarrow g' : B \to C$ is denoted by $\beta\alpha : gf \Rightarrow g'f' : A \to C$.
- for any 0-cell A an identity 1-cell $1_A : A \to A$, called the identity on A.

Furthermore, both horizontal and vertical composition are required to be associative and unitary. The graphical representation of 2-cells is very useful since one can compose or *paste* 2-cells in any order. This is sound because from the functoriality of the horizontal composition we obtain the so called interchange law: $(\beta'\alpha') \circ (\beta\alpha) = (\beta' \circ \beta)(\alpha' \circ \alpha)$ for any 2-cells as in the diagram.

$$\begin{tikzcd}[column sep=large]
A \arrow[r, bend left=60, "f"{name=U}] \arrow[r, "f'"{name=M, description}] \arrow[r, bend right=60, "f''"'{name=D}] & B \arrow[r, bend left=60, "g"{name=U2}] \arrow[r, "g'"{name=M2, description}] \arrow[r, bend right=60, "g''"'{name=D2}] & C
\arrow[Rightarrow, from=U, to=M, "\alpha"]
\arrow[Rightarrow, from=M, to=D, "\alpha'"]
\arrow[Rightarrow, from=U2, to=M2, "\beta"]
\arrow[Rightarrow, from=M2, to=D2, "\beta'"]
\end{tikzcd}$$

To improve readability of diagrams and for consistency with the 2-category theoretical literature, we use small letters for 1-cells. But, when we instantiate $\mathbb{C}$ with a category of categories, 1-cells correspond to functors, and will be denoted by capital letters.

In what follows we will consider a 2-category $\mathbb{C}$ that has 2-products. This implies the existence of products at the level of 0-cells satisfying the usual universal property; so a pair of 1-cells $p : A \to B$ and $q : A \to C$ yields a unique 1-cell $\langle p, q \rangle : A \to B \times C$. Moreover, any 2-cell of the form $f \Rightarrow g : A \to B \times C$ is essentially a pair $\langle \xi, \zeta \rangle$ with $\xi : \pi_1 f \Rightarrow \pi_1 g : A \to B$ and $\zeta : \pi_2 f \Rightarrow \pi_2 g : A \to C$. For the exact definition, see [17]. We need 2-products to incorporate parameters.

Given a 1-cell $f : A \times B \to B$ we will consider, as motivated by the example given in the introduction, the functor $f\langle 1, - \rangle : \mathbb{C}(A, B) \to \mathbb{C}(A, B)$, that maps a 1-cell $u : A \to B$ to the 1-cell $f\langle 1_A, u \rangle$. To simplify the notation, we will denote the categories of coalgebras for the $\mathbb{C}(A, B)$-functor $f\langle 1, - \rangle$ by $\mathsf{Coalg}^A_B(f)$ and, we will call an $f\langle 1, - \rangle$-coalgebra simply an f-coalgebra. Objects of this category are of the form (u, ξ) where $u : A \to B$ is a 1-cell in $\mathbb{C}$

and ξ is a 2-cell as in the next figure.

$$\begin{tikzcd}
A \arrow[rr, "u"] \arrow[dr, bend right, "\langle 1,u\rangle"'] & {} \arrow[d, Rightarrow, "\xi"] & B \\
& A \times B \arrow[ur, bend right, "f"'] &
\end{tikzcd}$$

A morphism between coalgebras (u, ξ) and (u', ξ') is a 2-cell $\alpha : u \Rightarrow u'$ such that $\xi' \circ \alpha = f\langle 1, \alpha\rangle \circ \xi$.

As an example, instantiate $\mathbb{C}$ to Cat, put $A = B = \mathsf{Set}$ and let the 1-cell f be the bifunctor $F : \mathsf{Set}^2 \to \mathsf{Set}$ mapping (X, Y) to $X \times Y$. The final coalgebra in $\mathsf{Coalg}^A_B(f)$ in this instance is the `Stream` functor described in the introduction.

A final object in the category $\mathsf{Coalg}^A_B(f)$, when it exists, will be denoted by $(\nu f, \mathsf{ufld})$. Notice that $\nu f : A \to B$ is a 1-cell.

Aside form having 2-products, we require the 2-categories we consider to have a *stability of fixpoints* property (2) that was also required in [22].

Stability of fixpoints. Consider 1-cells $f : A \times B \to B$ and $h : A' \to A$. Using h we obtain a 1-cell $f(h \times 1) : A' \times B \to B$. This in turn gives rise to the endofunctor $f\langle h, -\rangle$ on $\mathbb{C}(A', B)$, for which we consider the corresponding category of coalgebras $\mathsf{Coalg}^{A'}_B(f(h \times 1))$. Then we require that the final coalgebra $\nu(f(h \times 1))$ exists whenever the final coalgebra νf does, and moreover

$$\nu(f(h \times 1)) = (\nu f)h \tag{2}$$

▶ **Remark 1.** *If for a 1-cell $h : A' \to A$ the functor $\mathbb{C}(h, B) : \mathbb{C}(A, B) \to \mathbb{C}(A', B)$ mapping $u : A \to B$ to $uh : A' \to B$ has a left adjoint,*[1] *then (2) is satisfied. Indeed, since $\mathbb{C}(h, B) \circ (f\langle 1, -\rangle) \simeq (f\langle h, -\rangle) \circ \mathbb{C}(h, B)$, the adjunction between $\mathbb{C}(A, B)$ and $\mathbb{C}(A', B)$ lifts to an adjunction between the corresponding categories of coalgebras:*

$$\begin{tikzcd}
\mathsf{Coalg}^A_B(f) \arrow[r, phantom, "\perp"] \arrow[r, bend right] \arrow[d] & \mathsf{Coalg}^{A'}_B(f(h \times 1)) \arrow[l, bend right] \arrow[d] \\
\mathbb{C}(A, B) \arrow[loop left, "f\langle 1,-\rangle"] \arrow[r, phantom, "\perp"] \arrow[r, bend right, "\mathbb{C}(h,B)"'] & \mathbb{C}(A', B) \arrow[l, bend right] \arrow[loop right, "f\langle h,-\rangle"]
\end{tikzcd}$$

In particular, since right adjoints preserve all limits, and hence final objects, the final coalgebra $\nu(f(h \times 1))$ exists whenever νf does, and moreover (2) holds.

The situation we are interested in is the following. Assume we have two 1-cells

$$f : A \times B \times C \to B \qquad g : A \times B \times C \to C$$

such that there exists a final coalgebra in $\mathsf{Coalg}^{A \times B}_C(g)$. We can 'plug in' the 1-cell $\nu g : A \times B \to C$ into f to obtain a 1-cell that we will denote by $f \lhd \nu g : A \times B \to B$, which is

[1] As an example, let $\mathbb{C}$ be the 2-category of locally finitely presentable categories. In this case, left Kan extensions exist, so the functors $\mathbb{C}(h, B)$ have left adjoints. However, in general, the existence of left Kan extensions is a stronger requirement than (2) and is not required in the proof of the Bekič rule.

formally defined as the composition $f\langle \pi_A, \pi_B, \nu g\rangle$. The 1-cell of interest is the final coalgebra

$$\nu(f\langle \pi_A, \pi_B, \nu g\rangle) \tag{3}$$

In Cat when we instantiate f and g to functors $F : \mathcal{A} \times \mathcal{B} \times \mathcal{C} \to \mathcal{B}$ and $G : \mathcal{A} \times \mathcal{B} \times \mathcal{C} \to \mathcal{C}$, the 1-cell in (3) actually gives a functor from $\mathcal{A}$ to $\mathcal{B}$ that for a parameter X in $\mathcal{A}$ computes $\nu Y.F(X, Y, \nu Z.G(X, Y, Z))$. The aim of the generalised Bekič rule is to establish that this fixpoint is the first projection of the greatest fixpoint of the many-sorted functor given by

$$\begin{pmatrix} Y \\ Z \end{pmatrix} \mapsto \begin{pmatrix} F(X, Y, Z) \\ G(X, Y, Z) \end{pmatrix}.$$

Coming back to the 2-categorical setting, we want to find the connection between the final objects (when they exist) of the categories $\mathsf{Coalg}^A_B(f\langle \pi_A, \pi_B, \nu g\rangle)$, respectively $\mathsf{Coalg}^A_{B\times C}(\langle f, g\rangle)$.

▶ **Notation 2.** *Consider 1-cells $f : A \times B \times C \to B$ and $h : A \times B \to C$. Let $f \lhd h$ denote the composition $f\langle \pi_A, \pi_B, h\rangle : A \times B \to B$ and, we abbreviate by $\langle f, h\rangle$ the pair $\langle f, h\langle \pi_A, \pi_B\rangle\rangle : A \times B \times C \to B \times C$. The natural way to obtain from f and h a 1-cell with codomain $B \times C$ is to precompose h with projections and then use the standard product pairing. Denoting this 1-cell by $\langle f, h\rangle$ is only a mild abuse of notation.*

Using this notation, we study the diagram

$$\begin{tikzcd}
\mathsf{Coalg}^A_B(f \lhd \nu g) \arrow[r, bend right, "I"'] \arrow[r, phantom, "\perp"] \arrow[d] & \mathsf{Coalg}^A_{B\times C}(\langle f, \nu g\rangle) \arrow[l, bend right, "L"'] \arrow[d] & \mathsf{Coalg}^A_{B\times C}(\langle f, g\rangle) \arrow[l, "(-)^\dagger"'] \arrow[d] \\
\mathbb{C}(A, B) & \mathbb{C}(A, B \times C) \arrow[l, "\pi_1 -"'] & \mathbb{C}(A, B \times C) \arrow[l, "Id"']
\end{tikzcd}$$

where the vertical arrows are forgetful functors and π_1- denotes composition with the projection $\pi_1 : B \times C \to B$. The functors L and I are explained in Lemmas 5, while the functor $(-)^\dagger$ is introduced in Lemma 6. From these lemmas we obtain:

▶ **Theorem 3.** *The diagram above commutes. Further, $\mathsf{Coalg}^A_B(f \lhd \nu g)$ is a full reflective subcategory of $\mathsf{Coalg}^A_{B\times C}(\langle f, \nu g\rangle)$ and $(-)^\dagger$ creates final objects.*

Before stating the two lemmas let us show how the Bekič rule follows.

▶ **Corollary 4.** *Assume $\nu(f \lhd \nu g)$ exists. Then $\nu\langle f, g\rangle$ also exists and we have*

$$\nu(f \lhd \nu g) = \pi_1 \nu\langle f, g\rangle. \tag{4}$$

Proof. If $\mathsf{Coalg}^A_B(f \lhd \nu g)$ has a final object, then by Lemma 5, I preserves it, since as a right adjoint it preserves all limits, see [8, Prop 3.2.2]. Moreover, L also preserves the final coalgebra, since by Lemma 5 L is a reflector, and thus we can use [8, Prop 3.5.3]. By Lemma 6 the functor $(-)^\dagger$ creates final objects, so $\nu\langle f, g\rangle$ exists. We conclude that $L \circ (-)^\dagger$ maps the final coalgebra in $\mathsf{Coalg}^A_{B\times C}(\langle f, g\rangle)$ to the final coalgebra in $\mathsf{Coalg}^A_B(f \lhd \nu g)$. But the composite $L \circ (-)^\dagger$ acts on the carrier 1-cells of the final coalgebras by composing with the projection on the first component. Therefore we obtain (4). ◀

▶ **Lemma 5.** *Consider 1-cells $f : A \times B \times C \to B$ and $h : A \times B \to C$. Then $\mathsf{Coalg}^A_B(f \lhd h)$ is isomorphic to a full reflective subcategory of $\mathsf{Coalg}^A_{B\times C}(\langle f,h\rangle)$.*

Sketch. We exhibit an adjunction $L \dashv I : \mathsf{Coalg}^A_B(f \lhd h) \to \mathsf{Coalg}^A_{B\times C}(\langle f,h\rangle)$ and show that I is full and faithful. I acts on an $f \lhd h$-coalgebra ξ by

$$\begin{tikzcd}
A \arrow[rr, "u"] \arrow[dr, bend right, "{\langle 1,u\rangle}"'] & {} \arrow[d, phantom, "\Downarrow\xi"] & B & \mapsto & A \arrow[rr, "{\langle u,h\langle 1,u\rangle\rangle}"] \arrow[dr, bend right, "{\langle 1,u,h\langle 1,u\rangle\rangle}"'] & {} \arrow[d, phantom, "\Downarrow\langle\xi,1\rangle"] & B\times C \\
& A\times B \arrow[ur, bend right, "f\lhd h"'] & & & & A\times B\times C \arrow[ur, bend right, "{\langle f,h\rangle}"'] &
\end{tikzcd}$$

A coalgebra homomorphism $\alpha : u \Rightarrow u' : A \to B$ in $\mathsf{Coalg}^A_B(f \lhd h)$ is mapped by I to $\langle \alpha, h\langle 1,\alpha\rangle\rangle$ and it is immediate to verify that this is indeed a coalgebra morphism in $\mathsf{Coalg}^A_{B\times C}(\langle f,h\rangle)$.

The functor L acts on an $\langle f,h\rangle$-coalgebra $\langle \xi,\zeta\rangle$ as follows. We use that $f \lhd h = f\langle 1,h\rangle$.

$$\begin{tikzcd}
A \arrow[rr, "{\langle u,v\rangle}"] \arrow[dr, bend right, "{\langle 1,\langle u,v\rangle\rangle}"'] & {} \arrow[d, phantom, "\Downarrow\langle\xi,\zeta\rangle"] & B\times C & \mapsto & A \arrow[rrr, "u"] \arrow[drr, bend right, "{\langle 1,u,v\rangle}"] \arrow[ddr, bend right, "{\langle 1,u\rangle}"'] & & {} \arrow[d, phantom, "\Downarrow\xi"] & B \\
& A\times B\times C \arrow[ur, bend right, "{\langle f,h\rangle}"'] & & & & {} \arrow[d, phantom, "\Downarrow\langle 1,1,\zeta\rangle"] & A\times B\times C \arrow[ur, bend right, "f"] & \\
& & & & & A\times B \arrow[ur, bend right, "{\langle 1,h\rangle}"'] \arrow[uurr, bend right, "f\lhd h"'] & &
\end{tikzcd}$$

It is routine to check that when $\langle \alpha,\beta\rangle$ is a morphism in $\mathsf{Coalg}^A_{B\times C}(\langle f,h\rangle)$ then α is a morphism in $\mathsf{Coalg}^A_B(f \lhd h)$. Moreover, one readily verifies that L is left adjoint to I. ◀

▶ **Lemma 6.** *Consider 1-cells $\langle f,g\rangle : A \times B \times C \to B \times C$. Then there exists a faithful functor $(-)^\dagger : \mathsf{Coalg}^A_{B\times C}(\langle f,g\rangle) \to \mathsf{Coalg}^A_{B\times C}(\langle f,\nu g\rangle)$ that creates final objects.*

Sketch. Consider a coalgebra $\langle \xi,\zeta\rangle$ in $\mathsf{Coalg}^A_{B\times C}(\langle f,g\rangle)$ as in the left diagram in (5). Then $\zeta : v \Rightarrow g\langle 1,u,v\rangle$ is a coalgebra in $\mathsf{Coalg}^A_C(g(\langle 1,u\rangle \times 1))$. By the stability condition (2), the final object in the latter category is isomorphic to

$$\begin{tikzcd}
A \arrow[rr, "{\langle 1,u\rangle}"] \arrow[dr, bend right, "{\langle 1,\nu g\langle 1,u\rangle\rangle}"'] & {} \arrow[d, phantom, "="] & A\times B \arrow[rr, "\nu g"] \arrow[dr, bend right, "{\langle 1,\nu g\rangle}"] & {} \arrow[d, phantom, "\Downarrow \mathsf{ufld}"] & C \\
& A\times C \arrow[rr, "{\langle 1,u\rangle\times 1}"'] & & A\times B\times C \arrow[ur, bend right, "g"'] &
\end{tikzcd}$$

Thus there exists a unique coalgebra morphism $\zeta^\dagger : v \Rightarrow \nu g\langle 1,u\rangle$ such that

$$(\mathsf{ufld}\langle 1,u\rangle) \circ \zeta^\dagger = (g(\langle 1,u\rangle \times 1)\langle 1,\zeta^\dagger\rangle) \circ \zeta$$

We can now define the functor $(-)^\dagger$ on objects by

$$\begin{tikzcd}
A \arrow[rr, "{\langle u,v\rangle}"] \arrow[dr, bend right, "{\langle 1,\langle u,v\rangle\rangle}"'] & {} \arrow[d, phantom, "\Downarrow\langle\xi,\zeta\rangle"] & B\times C & \mapsto & A \arrow[rr, "{\langle u,v\rangle}"] \arrow[dr, bend right, "{\langle 1,\langle u,v\rangle\rangle}"'] & {} \arrow[d, phantom, "\Downarrow\langle\xi,\zeta^\dagger\rangle"] & B\times C \\
& A\times B\times C \arrow[ur, bend right, "{\langle f,g\rangle}"'] & & & & A\times B\times C \arrow[ur, bend right, "{\langle f,\nu g\rangle}"'] &
\end{tikzcd} \tag{5}$$

On arrows the functor $(-)^\dagger$ is defined as identity. One can show that $(-)^\dagger$ creates final objects. ◀

$$\begin{tikzcd}
1 \arrow[dr, "j_0"] \arrow[dd, bend right, "i_0"'] & F1 \arrow[l, "p_0"'] & \dots & F^n 1 \arrow[dr, "j_n"] \arrow[dd, bend right, "i_n"'] & & \dots & F^\omega 1 \arrow[lll, bend right, "p_n^\omega"'] \arrow[dd, bend left, "i"] \\
0 \arrow[r] \arrow[d, "e_0"] & (\bullet + F)0 \arrow[d] & \dots & (\bullet + F)^n 0 \arrow[r] \arrow[d] & (\bullet + F)^{n+1} 0 \arrow[rr, bend left=10] \arrow[d, "e_{n+1}"] & \dots & (\bullet + F)^\omega 0 \arrow[d, "e"'] \\
1 & (\bullet + F)1 \arrow[l, "q_0"] & \dots & (\bullet + F)^n 1 & (\bullet + F)^{n+1} 1 \arrow[l, "q_n"'] & \dots & (\bullet + F)^\omega 1 \arrow[ll, bend right=10, "q_{n+1}^\omega"'] \arrow[lll, bend left, "q_n^\omega"]
\end{tikzcd}$$

Figure 1 Approximating $F^\omega 1$.

3 Truncating elements of final coalgebras

Having eliminated fixpoints using the Bekič rule, the remaining problem is approximating elements of the final F-coalgebra, for a polynomial F on Set^n for some positive integer n. Even though it is essential for us to allow Set^n for $n > 1$ to account for nested fixpoints, the following simple example is illustrative.

▶ **Example 7.** Let $F : \mathsf{Set} \to \mathsf{Set}$ be given by $FX = \{a, b\} \times X \times X$. We want to approximate infinite binary trees with nodes a, b, that is, elements of the final F-coalgebra, the carrier of which we write as $F^\omega 1$. As a data type of approximants we choose the initial $(\{\bullet\}+F)$-algebra, the carrier of which we write as $(\bullet + F)^\omega 0$. We then have injections i and e

$$F^\omega 1 \xrightarrow{i} (\bullet + F)^\omega 1 \xleftarrow{e} (\bullet + F)^\omega 0$$

In the following we will need the final sequence of F consisting of projections of elements of $F^\omega 1$ and the initial sequence of $(\{\bullet\} + F)$ consisting of the possible truncations of elements of $F^\omega 1$. Both sequences embed into the final sequence of $(\{\bullet\} + F)$, which will be used to define the metric that allows us to capture the approximation of infinite elements of $F^\omega 1$ by truncations.

Writing $\bullet$ for $\{\bullet\}$, this data is made visible in Fig. 1. We consider the horizontal arrows in the middle row as inclusions and do not give them names. Similarly, we will often treat i and e as inclusions and drop them from our notation. i_0, p_0, e_0, q_0 are uniquely determined by their types and j_0 maps the element of 1 to $\bullet$. We put $p_{n+1} = Fp_n$ and for $f \in \{i, e, j, q\}$ we let $f_{n+1} = \mathsf{inr} \circ Ff_n$, where inr is a right coprojection map. The p_n^ω, q_n^ω, and i are determined by $F^\omega 1$ and $(\bullet + F)^\omega 1$ being limits.

▶ **Proposition 8.** *In Fig. 1, we have $q_n \circ e_{n+1} \circ j_n = i_n$.*

We call elements of $(\bullet + F)^n 0$ *truncations* (or approximations).

▶ Remark. Why do we use the $(\bullet + F)^n 0$ and not the $F^n 1$ as the range of truncations? As will become clear in Section 4, in our code we need a datatype for truncations in order to print them. To this end we need to use a constructor, which, in our code, is given by $\bullet$. As far as the implementation is concerned, it is indisputable that we need $\bullet$. The only question that remains is how to interpret the $\bullet$ in the code from a semantic perspective. In particular, why don't we interpret the $\bullet$ of the program as the element of 1 in $\coprod F^n 1$? There are two reasons for this.

1. It is true that the final sequence suggests considering truncations as elements of $\coprod F^n 1$. However, truncations are required to be finite terms, so from a conceptual point of view, it is natural to regard them as elements of an initial algebra or an inductively defined type. Since the initial algebra of F may be empty, as in the case of streams, we are using the initial algebra of $\bullet + F$. This section carries out the indispensable analysis of the relation between the initial sequence of $\bullet + F$ and the final sequence of F.
   A benefit of this analysis is that we solve a question raised by Barrâs theorem, namely how to approximate a final coalgebra by an initial algebra if the initial algebra sequence is empty (due to $F0 = 0$). To replace F by $\bullet + F$ is an obvious idea, but one needs to deal with the fact that bullets can appear now at all levels and that is what we do in this section.
2. Our methodology of proving the productivity of programs in Section 4 should be able to support correctness proofs of any implementation of printing. It is true that there always is an implementation that prints the $\bullet$ exactly where the final sequence has a $\star$ (if $\star \in 1$ is the element of 1 in the final sequence $F^n 1$). But there are many other implementations. This becomes particularly important in the case of nested fixpoints. The implementation corresponding to the final sequence corresponds to a quite particular strategy of when each fixpoint is unfolded.
   To reinforce this point, let us consider an as an example streams of streams of Int. After applying Bekič, we get the equations

   $$\begin{aligned} T &= S \times T \\ S &= \mathsf{Int} \times S \end{aligned}$$

   telling us that S is the type of streams over Int and T is the type of streams over S. Let us develop the final sequence for the functor

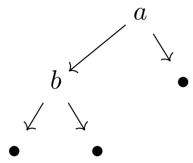
   $$F(T, S) = (S \times T, \mathsf{Int} \times S).$$

   We use "o" for the 1 of type T and \$ for the 1 of type S. And we write "," for product. The first four elements of the sequence $F^n 1$ are of the following types, with the first line referring to the first component (the T-component) and the second line referring to the second component (the S-component):

   $$\begin{array}{cccc} o & (\$, o) & (\mathsf{Int}, \$), (\$, o) & (\mathsf{Int}, \mathsf{Int}, \$), (\mathsf{Int}, \$), (\$, o) \\[1em] \$ & (\mathsf{Int}, \$) & (\mathsf{Int}, \mathsf{Int}, \$) & (\mathsf{Int}, \mathsf{Int}, \mathsf{Int}, \$) \end{array} \tag{6}$$

   This corresponds to an implementation that prints n elements of the first stream, $n - 1$ elements of the second stream and so on. But there are many other ways of printing streams of streams. For example, we may want to say that "we want to see more of early streams" and implement printing $2n$ elements of the first stream, $2(n - 1)$ elements of the second stream, etc. So we need truncations of type $((\mathsf{Int}, \mathsf{Int}, \mathsf{Int}, \mathsf{Int}, \$), (\mathsf{Int}, \mathsf{Int}, \$), (\$, o))$ which are not in any $F^n 1$, that is, they don't appear in the upper row of Diagram (6).

▶ **Example 9.** Let $FX = \{a, b\} \times X \times X$. Then (7) shows a truncation that cannot be obtained from any of the F^n1.



(7)

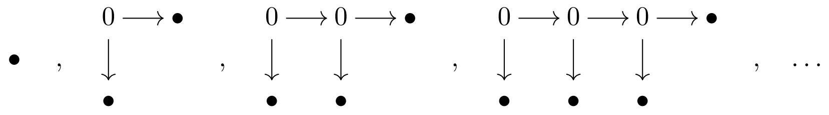
The set of truncations, that is, $(\bullet + F)^\omega 0$, carries a metric induced by the embedding $e : (\bullet + F)^\omega 0 \to (\bullet + F)^\omega 1$ and the final sequence $(\bullet + F)^n 1$ as in Barr [3, Proposition 3.1]. Explicitly, for $t, s \in (\bullet + F)^\omega 1$ let $d(t, s) = 0$ if $t = s$ and else $d(t, s) = 2^{-n}$ where n is the largest natural number such that $q^\omega_n(t) = q^\omega_n(s)$.

Consequently, we have a notion of convergence and Cauchy sequence we can apply to sequences of truncations. For an example consider some $t \in F^\omega 1$ and its *canonical sequence of truncations* $j_n(p^\omega_n(t))_{n<\omega}$:

▶ **Proposition 10.** *For all $t \in F^\omega 1$ the sequence $j_n(p^\omega_n(t))_{n<\omega}$ converges to t.*

Not all converging sequences of truncations converge to an element in $F^\omega 1$:

▶ **Example 11.** Let $FX = \{0, 1\} \times X \times X$. Then the following is a converging sequence of truncations that does not converge to an element of $F^\omega 1$:



The following definition captures the intuition that a productive sequence is a sequence in which all bullets get eventually eliminated (because $j_n(t_n)$ has no bullets below level n):

▶ **Definition 12.** A sequence $(a_k)_{k<\omega}$ is called *productive* if it is Cauchy in $(\bullet + F)^\omega 0$ and if for all n there is $t_n \in F^n 1$ and $k < \omega$ such that $q^\omega_n(e(a_k)) = i_n(t_n)$.

We read $q^\omega_n(e(a_k)) = i_n(t_n)$ as t_n 'is below' a_k. Observe that the t_n in the definition necessarily converge to the same limit as the a_k. For an example note that for any $t \in F^\omega 1$ the canonical sequence converging to t introduced Proposition 10 is productive (the proof uses Proposition 8).

▶ **Corollary 13.** *If $(a_k)_{k<\omega}$ is 'a productive sequence of truncations of t', that is, if for all n there is k such that $q^\omega_n(e(a_k)) = i_n(p^\omega_n(t))$, then $\lim a_k = t$.*

This shows that all elements of $F^\omega 1$ are approximated by a productive sequence. A stronger statement is:

▶ **Proposition 14.** *Let $(a_k)_{k<\omega}$ be a sequence in $(\bullet + F)^\omega 0$. Then, $\lim a_k \in F^\omega 1$ iff $(a_k)_{k<\omega}$ is productive.*

Proof. If $\lim a_n = t \in F^\omega 1$, then for all n there is k such that $q^\omega_n(e(a_k)) = i_n(t_n)$ as required by Def. 12. Conversely, if a_k is productive, then it converges against the same limit as the t_n from Def. 12. Now $q^\omega_n(e(a_k)) = i_n(t_n)$ implies that there is $t \in F^\omega 1$ such that $\lim t_n = t$, hence $\lim a_k = t$. ◀

Our analysis improves somewhat over Barr's original result as we do not have to assume that $F0 \neq 0$ (which excludes e.g. streams). This comes at the cost that $(\bullet + F)^\omega 1$ has 'spurious elements' that are not in $F^\omega 1$. The following summarises two satisfactory characterisations of $F^\omega 1$ as a subset of $(\bullet + F)^\omega 1$.

▶ **Theorem 15.** *Let $F : \mathsf{Set}^n \to \mathsf{Set}^n$ be a many-sorted polyomial functor. An element of $(\bullet + F)^\omega 1$ is in $F^\omega 1$ iff it is approximated by a productive sequence iff it is bullet-free.*[2]

Proof. The first 'iff' is Proposition 14. The second 'only if' is obvious. For the other direction, we show by induction that if an element in $(\bullet + F)^n 1$ is bullet-free then it is already in $F^n 1$, or, in other words, that the image of i_n contains all bullet-free elements of $(\bullet + F)^n 1$. ◀

4 Implementing truncations in Haskell

We apply the theory developed in the previous sections to prove correctness of an effective procedure for printing infinite objects in Haskell. A naive attempt to printing the infinite normal form of the stream of streams `(matrix 0)` only shows an infinite stream of `0`'s and we never see the other streams. Our solution is to print the sequence of truncations of a term. The truncation of a term at depth n is obtained from the tree representation of the term by replacing the subterms at depth n by $\bullet$. Printing the sequence of all truncations is a *faithful* way of printing infinite data if we prove the following two correctness properties:

- the truncations are always finite (and hence, printable in a finite amount of time) and
- the sequence of truncations has length ω and converges to the infinite normal form of the program at issue.

To this end, we assume that our programs are infinitary normalising (productive). For example, we exclude programs that do not have infinite normal form at all such as `(novalue 0)` defined as follows.

```
loop = loop     novalue n = Cons loop (matrix n)
```

The set of (potentially infinite) normal forms can be seen as the carrier of a final coalgebra of the form $F^\omega 1$. Section 4.2 gives the correctness proof of our implementation and relies on the results of Section 3. The full implementation is available at [18].

We give the type declaration of a class `Trunc` introducing a new data type

```
  B :: * -> Nat -> *
```

used for implementing the truncations. The truncation of a term of type `a` at level `n` will have type `B a n`. The dependency of `(B a n)` on `n :: Nat` ensures that inhabitants of this data type are finite, and thus amenable to printing.

```
data Nat = Zero | Succ Nat            class Trunc (a :: *) where
data SNat n where                       data B :: * -> Nat -> *
  SZ :: SNat Zero                       trunc :: SNat n -> a -> B a n
  SS :: SNat n -> SNat (Succ n)
```

[2] A tree is *bullet-free* if it contains no occurrence of $\bullet$.

Note that we fake dependent types in Haskell using data promotion and singletons [26, 10]. The data type Nat is promoted to kinds. The singleton type (SNat n) can be thought of as having one inhabitant, intuitively, the natural number n.

To illustrate how the function trunc works we first present non-generic implementations. We make the parametric datatype (Stream a) an instance of the class Trunc provided the parameter a is also an instance of the class Trunc.

```
data B (Stream a) n where
  Bullet :: B (Stream a) Zero
  ConsS :: B a n -> B (Stream a) n -> B (Stream a) (Succ n)

trunc SZ _ = Bullet
trunc (SS n) (Cons x xs) = ConsS (trunc n x) (trunc n xs)
```

We also consider the data type for rose trees which is a multi-way tree structure in which each node may have an arbitrary number of children [5].

```
data RoseTree a = RoseTree a [RoseTree a]
```

The definition of trunc needs more care, the following would be wrong:

```
trunc SZ   x          = BulletRose
trunc (SS n) (RoseTree x xs) = RoseTreeB (trunc n x) (map (trunc n) xs)
```

because (map (trunc n) xs) is infinite if xs is infinite. In order to truncate rosetrees correctly, we replace the second line in the above code by the following:

```
trunc (SS n) (RoseTree x xs) = RoseTreeB (trunc n x) (trunc n xs)
```

where the last truncation is applied to the list xs. The theoretical foundation behind this solution is the Bekič rule on datatypes as explained in Section 2, which allows us to rewrite the definition of rose trees $\nu X.A \times (\nu Y.1 + X \times Y)$ as $\Pi_1(\nu(X,Y).(A \times Y, 1 + X \times Y))$. In other words, (Rosetree a) is written as the solution of two mutually corecursive equations:

$$\begin{aligned} X &= A \times Y \\ Y &= 1 + X \times Y \end{aligned} \tag{8}$$

After applying the Bekič rule, the many-sorted functor associated to rosetrees is

$$\begin{pmatrix} X \\ Y \end{pmatrix} \xrightarrow{\mathtt{FRT}_A} \begin{pmatrix} A \times Y \\ 1 + X \times Y \end{pmatrix}.$$

In our implementation, since we are using overloading polymorphism we do not see that we actually have two different versions of the function trunc, one for each recursive equation (assuming A is a basic type).

4.1 A generic implementation of truncations

In this section, we give a uniform implementation of truncation for a wide class of data types. The data types should have the form $T_1(T_2(\ldots T_n(Int)\ldots))$ where $T_i(X) = \nu Y.F_i(X,Y)$ and F_i is a polynomial functor[3]. The view of data types as fixpoints of functors is implemented through a type class called Rep similar to the class Regular but for bifunctors instead of functors [23, 25]. Associated to this class there is a type family FunctorRep.

[3] Actually, *Int* could be replaced by any basic data type.

```
type family FunctorRep (t :: * -> *) :: * -> * -> *
```

This type family can be seen as a function that given a parametric datatype T, it gives a polynomial bifunctor F such that $T(X) = \nu Y.F(X,Y)$. Polynomial bifunctors are represented by the following type constructors which are all made instances of the class `BiFunctor`:

- `U` for constant,
- `P1` for first projection,
- `P2` for second projection,
- `:**:` for product and `:++:` for sum.

```
class Bifunctor f where
 bimap :: (a -> c) -> (b -> d) -> (f a b -> f c d)
```

For our applications, we restrict the type `a` in the constant functor `U a` to be a basic type whose elements are all finitely normalising (i.e. printable in a finite amount of time) such as `Int`. We can now associate `Stream` to the functor `FStream` by means of the type family `FunctorRep` as follows.

```
 type instance FunctorRep Stream = FStream
 type FStream = P1 :**: P2
```

The class `Rep` has two methods that witness the isomorphism in the fixpoint equation $T(A) \cong F(A, T(A))$.

```
class Rep t where
 getRep :: t a -> (FunctorRep t) a (t a)
 fromRep :: FunctorRep t a (t a) -> t a
```

We make `Stream` an instance of the class `Rep` as follows.

```
 getRep (Cons x xs) = P1 x :**: P2 xs
 fromRep (P1 x :**: P2 xs) = Cons x xs
```

Figure 2 illustrates the correspondence between our generic Haskell implementation of truncations and the semantic view given in the previous sections. In our code we have the following definition for the function `trunc`, which we explain below and in Figure 2 step by step.

```
 trunc SZ x = Bullet
 trunc (SS n) x = NextStep (bimap (trunc n) (trunc n) (getRep x))
```

In the second case we define the truncation at level $n+1$ of a term `x` of type `t a`. Semantically, this is a coinductive type $T(A) = \nu Y.F(A,Y)$ obtained as the greatest fixpoint of a functor F, which syntactically is given by `FunctorRep t`.

To compute the truncation of `x` at level $n+1$, first we "unfold" `x` via the function `getRep`. Then we truncate at level n the terms obtained from the unfolding, and we apply F. In our code the application of a bifunctor F to two morphisms is done using the function `bimap`. To be able to use `bimap` in this case, we should require that `FunctorRep t` belongs to the class `Bifunctor`.

The type `B (t a) n` of the truncation is defined as a data type with two constructors `Bullet` and `NextStep` as follows.

```
 data B (t a) n where
  Bullet :: B (t a) Zero
  NextStep :: FunctorRep t (B a n) (B (t a) n) -> B(t a)(Succ n)
```

$$\begin{tikzcd}
\texttt{t a} \arrow[d, "\texttt{getRep}"] & T(A) = \nu Y.F(A,Y) \arrow[d, "\alpha_F"] \\
\texttt{FunctorRep t a (t a)} \arrow[d, "\texttt{bimap (trunc n) (trunc n)}"] & F(A,T(A)) \arrow[d, "F((trunc\ n),(trunc\ n))"] \\
\texttt{FunctorRep t (B a n) (B (t a) n)} \arrow[d, "\texttt{NextStep}"] & F(B(A,n),B(T(A),n)) \arrow[d, equal] \\
\texttt{B (t a) (Succ n)} & B(T(A),n+1)
\end{tikzcd}$$

Figure 2 Generic definition of truncations in Haskell.

At a semantic level, $B(T(A),n)$ is defined inductively as follows. For the base type *Int* we have

$$B(Int,0) = \{\bullet\} \qquad B(Int,n+1) = Int$$

while for a coinductive type $T(A) = \nu Y.F(A,Y)$ we have

$$B(T(A),0) = \{\bullet\} \qquad B(T(A),n+1) = F(B(A,n),B(T(A),n))$$

This generic definition of `trunc` does not work if F itself contains a fixpoint. For example, in the case of rosetrees where $F(A,X) = A \times (\nu Y.1 + X \times Y)$ lists are not truncated but remain infinite. We explain briefly how to extend this generic implementation to include data types of the form $T_1(T_2(\ldots T_n(Int)))$ where $T_i(X) = \nu Y.F_i(X,Y)$ and F_i contains a fixed point. As opposed to polynomial F_i, if the functors F_i contain fixpoints, we need to make use of the Bekič rule in the implementation (as well as in the proof of its correctness). This makes it necessary to deal with mutually recursive equations, something that our implementation in terms of the Rep class is not able to manipulate at the moment. There exists a Haskell package that deals with mutually recursive equations generically [24]. We would need to extend this package to include parametric data types.

4.2 Correctness of the implementation

For every natural number n the truncation `(trunc n p)` has a finite normal form $\mathtt{v_n}$ independently of whether the normal form `v` to which `p` evaluates is finite or not. This is proved by induction on n. Assume that the program `p` has type $T_1(T_2(..T_m(Int)))$ where $T_i(X) = \nu Y_i.F_i(X,Y_i)$ and F_i is a polynomial functor for all $1 \leq i \leq m$. To complete the correctness proof we need to show:

$$\mathtt{v} = \lim_{\mathtt{n}\to\infty} \mathtt{v}_n \ .$$

Using Bekič rule (Corollary 4), we have that

$$T_1(T_2(..T_n(Int))) = \Pi_1 \circ \nu \begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_m \end{pmatrix} . \begin{pmatrix} F_1(Y_2,Y_1) \\ F_2(Y_3,Y_2) \\ \vdots \\ F_m(Int,Y_m) \end{pmatrix} .$$

Let $F(Y_1,\dots,Y_m) = (F_1(Y_2,Y_1), F_2(Y_3,Y_2),\dots F_m(Int,Y_m))$. We also consider the vector $\overrightarrow{\texttt{trunc n}} = (\texttt{trunc}\ \ \texttt{n},\dots,\texttt{trunc}\ \ \texttt{n})$ of length m. Then, it is not difficult to show by induction on n and using the definition of `trunc` given in Section 4.1 that `trunc` satisfies the following:

$$\begin{aligned}
\overrightarrow{\texttt{trunc}\ \ 0}\ t &= (\bullet,\dots,\bullet)\\
\overrightarrow{\texttt{trunc}\ \ \texttt{(n+1)}} &= F(\overrightarrow{\texttt{trunc}\ \ \texttt{n}})\circ\alpha_F
\end{aligned}$$

From the above, it is easy to prove by induction on n that $\overrightarrow{\texttt{trunc n}}\ t$ is equal to the canonical sequence $j_n \circ p^{\omega}_n(t)$ of truncations as defined in Section 3. By Proposition 10, the sequence $\overrightarrow{\texttt{trunc}\ \ \texttt{n}}\ t$ converges to t. It is enough to take a t whose first component is `v` where `v` is the infinite normal form of our original program p of type $T_1(T_2(\dots T_n(Int)))$.

5 Conclusion

Whereas some of the techniques we used are well known, there are original theoretical contributions (Theorems 3 and 15) as well as a novel solution to the problem of printing infinite datatypes in Haskell. Hutton and Gibbons generalised the approximation lemma from [5] to a certain class of datatypes that includes the polynomial ones [15]. Their definition of approximant does not cover the case of parametric datatypes. Danielsson *et al.* implemented their notion of approximant in the ChasingBottoms package [14], which is implemented using a style of generic programming called *Scrap Your Boilerplate* [19]. Our approach is different, as we use the class `Regular` which views datatypes as fixed points of functors [25].

It will be interesting to compare our work with [1, 9]. It is currently not clear to us whether their recursion schemes are strong enough to define truncations of rose trees.

Of course, printing can be seen as an illustrative example only and others, such as the incremental sending of infinite data over a channel, will be pursued in the future. Moreover, there are many topics we didn't touch upon, such as nested datatypes in the sense of [6], higher order functors, or dependent types, as well as other functional programming languages such as Agda or Coq. It will also be of interest to investigate type theories with an explicit Bekič rule.

Acknowledgements. The fourth author would like to acknowledge a Daphne Jackson fellowship sponsored by EPSRC and the University of Leicester.

References

1 R. Atkey and C. McBride. Productive coprogramming with guarded recursion. In *ACM SIGPLAN International Conference on Functional Programming, ICFP'13, Boston, MA, USA – September 25–27, 2013*, pages 197–208, 2013.

2 R. C. Backhouse, M. Bijsterveld, R. van Geldrop, and J. van der Woude. Categorical fixed point calculus. In *Category Theory and Computer Science*, 1995.

3 M. Barr. Terminal coalgebras for endofunctors on sets. *Theoretical Computer Science*, 114(2):299–315, 1999.

4 R. S. Bird and O. de Moor. *Algebra of programming*. Prentice Hall, 1997.

5 R. S. Bird. *Introduction to Functional Programming using Haskell (second edition)*. Prentice Hall, 1998.

6 R. S. Bird and R. Paterson. Generalised folds for nested datatypes. *Formal Asp. Comput.*, 11(2), 1999.

7 S.L. Bloom, Z. Ésik, A. Labella, and E. G. Manes. Iteration 2-theories. *Applied Categorical Structures*, 9(2):173–216, 2001.

8 F. Borceux. *Handbook of Categorical Algebra I.* Cambridge University Press, 1994.

9 A. Cave, F. Ferreira, P. Panangaden, and B. Pientka. Fair reactive programming. In *The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL'14, San Diego, CA, USA, January 20-21, 2014*, pages 361–372, 2014.

10 R. A. Eisenberg and S. Weirich. Dependently typed programming with singletons. In *Proceedings of the 5th ACM SIGPLAN Symposium on Haskell, Copenhagen*, pages 117–130, 2012.

11 Z. Ésik and A. Labella. Equational properties of iteration in algebraically complete categories. *Theoretical Computer Science*, 195(1):61–89, 1998.

12 M. Fiore. *Axiomatic Domain Theory in Categories of Partial Maps.* PhD thesis, University of Edinburgh, 1994.

13 P. Freyd. Remarks on algebraically compact categories. In *Applications of Categories in Computer Science*, volume 77 of *London Math. Soc. Lecture Notes Series*, pages 95—-106. Cambridge University Press, 1992.

14 Haskell Chasing Bottoms Package: For testing partial and infinite values. `http://hackage.haskell.org/package/ChasingBottoms`. Online: accessed March 2015.

15 G. Hutton and J. Gibbons. The generic approximation lemma. *Inf. Process. Lett.*, 79(4):197–201, 2001.

16 C. B. Jones, editor. *Programming Languages and Their Definition – Hans Bekic (1936–1982)*, LNCS 177. Springer, 1984.

17 G. M. Kelly and R. Street. Review of the elements of 2-categories. In *Category Seminar (Proc. Sem. Sydney 1972/73)*, LNM 420, pages 75–103. Springer, 1974.

18 A. Kurz, D. Petrişan, A. Pardo, P. Severi, and F.-J. de Vries. Haskell code for this paper. `http://www.cs.le.ac.uk/people/ps56/code.xml`, 2015.

19 R. Lämmel and S. L. Peyton Jones. Scrap your boilerplate: a practical design pattern for generic programming. In *TLDI'03*, 2003.

20 D. J. Lehmann and M. B. Smyth. Algebraic specification of data types: A synthetic approach. *Mathematical Systems Theory*, 14:97–139, 1981.

21 C. E. Martin, J. Gibbons, and I. Bayley. Disciplined, efficient, generalised folds for nested datatypes. *Formal Asp. Comput.*, 16(1):19–35, 2004.

22 A. Pitts. An elementary calculus of approximations. Unpublished note.

23 Haskell Regular: Generic programming library for regular datatypes. `http://hackage.haskell.org/package/regular`. Online accessed: March 2015.

24 A. Rodriguez, S. Holdermans, A. Löh, and J. Jeuring. Generic programming with fixed points for mutually recursive datatypes. In *ICFP*, 2009.

25 T. van Noort, A. Rodriguez Yakushev, S. Holdermans, J. Jeuring, B. Heeren, and J.P. Magalhães. A lightweight approach to datatype-generic rewriting. *J. Funct. Program.*, 20(3-4):375–413, 2010.

26 B. A. Yorgey, S. Weirich, J. Cretin, S. L. Peyton Jones, D. Vytiniotis, and J. P. Magalhães. Giving Haskell a promotion. In *TLDI 2012*, pages 53–66, 2012.

Final Coalgebras from Corecursive Algebras

Paul Blain Levy

School of Computer Science, University of Birmingham, UK
p.b.levy@cs.bham.ac.uk

Abstract

We give a technique to construct a final coalgebra in which each element is a set of formulas of modal logic. The technique works for both the finite and the countable powerset functors. Starting with an injectively structured, corecursive algebra, we coinductively obtain a suitable subalgebra called the "co-founded part". We see – first with an example, and then in the general setting of modal logic on a dual adjunction – that modal theories form an injectively structured, corecursive algebra, so that this construction may be applied. We also obtain an initial algebra in a similar way.

We generalize the framework beyond **Set** to categories equipped with a suitable factorization system, and look at the examples of **Poset** and $\mathbf{Set}^{\mathrm{op}}$.

1998 ACM Subject Classification F.1.1 Models of Computation, F.1.2 Modes of Computation, F.4.1 Mathematical Logic – Modal logic

Keywords and phrases coalgebra, modal logic, bisimulation, category theory, factorization system

Digital Object Identifier 10.4230/LIPIcs.CALCO.2015.221

1 Introduction

1.1 The Problem

Consider image-countable labelled transition systems, i.e. coalgebras for the Set functor $B : X \mapsto (\mathcal{P}_c X)^{\mathcal{A}}$. Here $\mathcal{A}$ is a fixed set (not necessarily countable) of labels and $\mathcal{P}_c X$ is the set of countable subsets of X. It is well-known [25] that, in order to distinguish all pairs of non-bisimilar states, Hennessy-Milner logic with finitary conjunction is not sufficiently expressive, and we instead require infinitary conjunction. For example, we may take all formulas

$$\phi ::= \bigwedge_{i \in I} \phi_i \mid \neg\phi \mid [a]\phi$$

where the indexing sets I are countable; and write $\bigvee$ and $\langle a \rangle$ for the de Morgan duals of $\bigwedge$ and $[a]$ respectively. Alternatively, it is sufficient to take the following $\Diamond$*-layered formulas.*

$$\phi ::= \langle a \rangle \, (\bigwedge_{i \in I} \phi_i \wedge \bigwedge_{j \in J} \neg\phi_j) \tag{1}$$

For a B-coalgebra (X, ζ), the semantics of these formulas is given by

$$u \models \langle a \rangle \, (\bigwedge_{i \in I} \phi_i \wedge \bigwedge_{j \in J} \neg\phi_j) \iff \exists x \in (\zeta(u))_a . \, (\forall i \in I . x \models \phi_i \wedge \forall j \in J . \, x \not\models \psi_j) \tag{2}$$

Following [15, 22], we obtain a final B-coalgebra in which states are sets of formulas, or, alternatively, sets of $\Diamond$-layered formulas. Specifically, if $(\!|x|\!)_{X,\zeta}$ is the set of $\Diamond$-layered

6th International Conference on Algebra and Coalgebra in Computer Science (CALCO'15).
Editors: Lawrence S. Moss and Pawel Sobocinski; pp. 221–237
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

formulas satisfied by a state x within the coalgebra (X, ζ), then the final coalgebra has carrier

$$M = \{(\!|x|\!)_{X,\zeta} \mid (X, \zeta) \text{ a } T\text{-coalgebra, } x \in X\}$$

and its structure sends $(\!|x|\!)_{X,\zeta}$ to the image of x along the function

$$X \xrightarrow{\zeta} FX \xrightarrow{F(\!|-|\!)_{X,\zeta}} FM$$

It may, however, be argued that this construction is not quite satisfactory, because it is couched in terms of all B-coalgebras. We might as well just form the sum of all B-coalgebras[1] and then take the strongly extensional quotient, i.e. the quotient by bisimilarity. The modal logic is not playing any real role.

We therefore ask: is it possible to construct the final coalgebra purely out of the logic, without referring to other coalgebras? In particular, we shall need to characterize when a set of formulas is of the form $(\!|x|\!)_{X,\zeta}$.

In the case of coalgebras of the *finite* powerset functor – for which finite conjunctions are expressive enough to distinguish non-bisimilar states – this question was answered in [4, Theorem 5.9] following [1, 23] and [29, Theorem 7.4]. The first step is to construct a transition system, called the "canonical model of modal logic K" [7], consisting of sets of modal formulas closed under certain inference rules. Then the subsystem consisting of hereditarily image-finite elements is a final coalgebra.

It is, however, not evident whether or how this construction could be adapted to logic with infinite conjunctions. We shall not consider that question in this paper. Instead we present a different solution, which is applicable quite generally.

Our solution treats sets of modal formulas as elements not of a transition system but of an *algebra*. We then cut down that algebra by a novel "co-founded part" construction, and this gives the final coalgebra.

1.2 Structure of Paper

The paper is in three sections.

In Section 2, we introduce our main construction: the co-founded part of an algebra. We see how this construction, applied to a suitable algebra, gives a final coalgebra.

In Section 3 we generalize our work to any modal logic on a dual adjunction. We see how such a logic, if it is expressive, will always give a suitable corecursive algebra so that our final coalgebra construction can be applied.

In Section 4 we further generalize our results, from **Set** to other categories equipped with a factorization system. We look at two examples of particular interest:

- **Poset**, giving a model of similarity;
- $\mathbf{Set}^{\mathrm{op}}$, giving a new construction of initial algebras.

1.3 Notation

Let X be a set.

- We write $\mathcal{P}X$ for the poset of subsets of X, ordered by inclusion.
- We write $\mathrm{EqRel}(X)$ for the poset of equivalence relations on X, ordered by inclusion.

[1] The sum is a proper class, but this may be avoided e.g. by including only coalgebras carried by a subset of $\mathbb{N}$.

- For $U \in \mathcal{P}X$, we write $^{\circ}U$ for U regarded as a set, and $i_U : {^{\circ}U} \to X$ for the inclusion.
- For $(\equiv) \in \mathrm{EqRel}(X)$ we write $X/\equiv$ for the quotient set, and $e_{\equiv} : X \to (X/\equiv)$ for $x \mapsto [x]_{\equiv}$.
- For $U \subseteq V \in \mathcal{P}X$ we write $i_{U,V} : {^{\circ}U} \to^{\circ} V$ for the inclusion.
- For $(\equiv) \subseteq (\equiv') \in \mathrm{EqRel}(X)$ we write $e_{\equiv,\equiv'} : (X/\equiv) \to (X/\equiv')$ for $[x]_{\equiv} \mapsto [x]_{\equiv'}$.

In diagrams, $\rightarrowtail$ indicates an injection and $\twoheadrightarrow$ a surjection.

A *partial function* from a set X to a set Y is a pair (U, f) of $U \in \mathcal{P}X$ and $f : U \to Y$. We write $(U, f) \sqsubseteq (V, g)$ is when $U \subseteq V$ and

$$\begin{tikzcd} U \arrow[r, "i_{U,V}"] \arrow[dr, "f"'] & V \arrow[d, "g"] \\ & Y \end{tikzcd}$$

. We write $X \rightharpoonup Y$ for the poset of partial functions ordered by $\sqsubseteq$.

2 Solving the Problem

This section solves the problem set out in Section 1.1. We construct an algebra of *theories*. Then we describe how every algebra has a special subalgebra called the *co-founded part*. The co-founded part of our algebra of theories provides a final coalgebra as required.

2.1 The B-Algebra of Theories

Our first step is to obtain a B-algebra from the modal logic, where B is our endofunctor $X \mapsto (\mathcal{P}_c\, X)^{\mathcal{A}}$.

Say that a *theory* is any set of $\Diamond$-layered formulas; this is a crude notion of theory, with no requirement of deductive closure. Let Form be the set of all theories. Our B-algebra is (Form, α) where $\alpha : B\,\mathsf{Form} \to \mathsf{Form}$ can be thought of as describing how the theory of a state x can be obtained from the theories of its successors. Explicitly, α sends $\mathcal{M} \in B\,\mathsf{Form}$ to the set of formulas $\langle a\rangle\ (\bigwedge_{i\in I}\phi_i \wedge \bigwedge_{j\in J}\neg\psi_j)$ for which there exists $M \in \mathcal{M}_a$ such that $\forall i \in I.\ \phi_i \in M$ and $\forall j \in J.\ \psi_j \notin M$.

This B-algebra has two key properties. Firstly it is *corecursive*, which we explain in the next section. Secondly it is *injectively structured* i.e. α is an injection; we defer the proof of this until Section 3.4.

2.2 Corecursive Algebras

We reprise here the basic concepts of recursive coalgebras and corecursive algebras.

Let B be an endofunctor on a category $\mathcal{C}$. We write $\mathrm{Alg}(B)$ and $\mathrm{Coalg}(B)$ for the categories of B-algebras and B-coalgebras respectively. The evident bijection between isomorphically structured B-algebras and isomorphically structured B-coalgebras will be written $(-)^{-1}$, in either direction.

As explained in [33], a common patten for recursively defining a function $f : X \to Y$ is to first parse $x \in X$ into constituent parts, then apply f to each part, then combine the results. This motivated the following definition.

▶ **Definition 1.** [10, 11, 14, 32] A *B-coalgebra-to-algebra map* from a B-coalgebra (X, ζ) to a B-algebra (Y, θ) is a morphism $f : X \to Y$ satisfying

$$\begin{tikzcd}
BX \arrow[r, "Bf"] & BY \arrow[d, "\theta"] \\
X \arrow[u, "\zeta"] \arrow[r, "f"'] & Y
\end{tikzcd}$$

Equivalently, it is a fixpoint of the endofunction

$$\begin{tikzcd}
\mathcal{C}(X,Y) \arrow[r, "B_{X,Y}"] & \mathcal{C}(BX,BY) \arrow[r, "\mathcal{C}(\zeta,\theta)"] & \mathcal{C}(X,Y)
\end{tikzcd}$$

Such a map may be composed with a B-algebra map $(X', \zeta') \to (X, \zeta)$ or a B-coalgebra map $(Y, \theta) \to (Y', \theta')$ in the evident way.

▶ **Definition 2.**

1. A B-coalgebra is *recursive* when there is a unique map from it to each B-algebra.
2. Dually, a B-algebra is *corecursive* when there is a unique map from each B-coalgebra to it.

▶ **Proposition 3.**

1. $(-)^{-1}$ *gives a bijection between initial B-algebras and isomorphically structured recursive coalgebras.*
2. *Dually,* $(-)^{-1}$ *gives a bijection between final B-coalgebras and isomorphically structured corecursive algebras.*

Proof. By Lambek's lemma. ◀

Recursive coalgebras are an easily grasped concept, thanks to Taylor's characterization of recursive coalgebras as *well-founded* coalgebras in the case where $\mathcal{C} = \mathbf{Set}$ and B preserves inverse images [33, 32].

Corecursive algebras (other than free ones [2]) appear not to have such a simple characterization [11]. Still, it is evident that our B-algebra of theories in Section 2.1 is corecursive. The unique map from a B-coalgebra (X, ζ) to our algebra is $(\!|-|\!)_{X,\zeta}$.

2.3 The Co-founded Part of an Algebra

Certain elements of a B-algebra are said to be *co-founded.* This is a coinductively defined predicate. To get some intuition, consider first the case where B is presented by operations. For an element of a B-algebra to be co-founded, it must be of the form $c(y_i \mid i \in I)$ where each y_i is co-founded.

Now for the general case. Let B be an endofunctor on **Set**, and (Y, θ) a B-algebra.

▶ **Definition 4.** We define an endofunction p on $\mathcal{P}Y$ as follows. For $U \in \mathcal{P}Y$ we define $p(U) \subseteq Y$ to be the range of the composite

$$\begin{tikzcd}
B^{\circ}U \arrow[r, "Bi_U"] & BY \arrow[r, "\theta"] & Y
\end{tikzcd}$$

This gives a square

$$\begin{tikzcd}
B^{\circ}U \arrow[r, "Bi_U"] \arrow[d, two heads, "r_U"'] & BY \arrow[d, "\theta"] \\
{}^{\circ}p(U) \arrow[r, tail, "i_{p(U)}"'] & Y
\end{tikzcd}$$

We next see that p is monotone and r is natural.

▶ **Proposition 5.** *If $U \subseteq V \in \mathcal{P}Y$, then $p(U) \subseteq p(V)$ and*

$$\begin{tikzcd}
B^{\circ}U \arrow[r, "Bi_{U,V}"] \arrow[d, two heads, "r_U"'] & B^{\circ}V \arrow[d, two heads, "r_V"] \\
{}^{\circ}p(U) \arrow[r, tail, "i_{p(U),p(V)}"'] & {}^{\circ}p(V)
\end{tikzcd}$$

writing $i_{U,V}$ for the inclusion of U in V.

Proof. The diagram

$$\begin{tikzcd}
{}^{\circ}U \arrow[rr, tail, "i_{U,V}"] \arrow[dr, tail, "i_U"'] & & {}^{\circ}V \arrow[dl, tail, "i_V"] \\
& Y &
\end{tikzcd}$$

commutes, so

$$\begin{tikzcd}
B^{\circ}U \arrow[rr, "Bi_{U,V}"] \arrow[d, two heads, "r_U"'] \arrow[dr, "Bi_U"] & & B^{\circ}V \arrow[d, two heads, "r_V"] \arrow[dl, "Bi_V"'] \\
{}^{\circ}p(U) \arrow[dr, tail, "i_{p(U)}"'] & BY \arrow[d, "\theta"] & {}^{\circ}p(V) \arrow[dl, tail, "i_{p(V)}"] \\
& Y &
\end{tikzcd}$$

commutes.

Diagonal fill-in gives

$$\begin{tikzcd}
B^{\circ}U \arrow[rr, "Bi_{U,V}"] \arrow[d, two heads, "r_U"'] & & B^{\circ}V \arrow[d, two heads, "r_V"] \\
{}^{\circ}p(U) \arrow[rr, dotted, "n"'] \arrow[dr, tail, "i_{p(U)}"'] & & {}^{\circ}p(V) \arrow[dl, tail, "i_{p(V)}"] \\
& Y &
\end{tikzcd}$$

So $p(U) \subseteq p(V)$ and $n = i_{p(U),p(V)}$. ◀

▶ **Definition 6.**

1. A *subalgebra* of (Y, θ) is $U \in \mathcal{P}Y$ for which there exists a (necessarily unique) function

$$\begin{tikzcd}
B^{\circ}U \arrow[r, "Bi_U"] \arrow[d, dotted] & BY \arrow[d, "\theta"] \\
{}^{\circ}U \arrow[r, "i_U"'] & Y
\end{tikzcd}$$

. Equivalently, it is a prefixpoint of p.

2. The least prefixpoint μp is called the *least subalgebra*.
3. The greatest postfixpoint μp is called the *co-founded part* of (Y, θ).

To summarize, we have B-algebra morphisms:

$$\begin{tikzcd}
B^{\circ}\mu p \arrow[r, "Bi_{\mu p,\nu p}"] \arrow[d, two heads, "r_{\mu p}"'] & B^{\circ}\nu p \arrow[r, "Bi_{\nu p}"] \arrow[d, two heads, "r_{\nu p}"'] & BY \arrow[dd, "\theta"] \\
{}^{\circ}p(\mu p) \arrow[r, tail, "i_{p(\mu p),p(\nu p)}"'] \arrow[d, equal] & {}^{\circ}p(\nu p) \arrow[d, equal] & \\
{}^{\circ}\mu p \arrow[r, tail, "i_{\mu p,\nu p}"'] & {}^{\circ}\nu p \arrow[r, tail, "i_{\nu p}"'] & Y
\end{tikzcd}$$

Clearly the least subalgebra and co-founded parts of (Y, θ) are both surjectively structured B-algebras. (More generally, a surjectively structured subalgebra is precisely a fixpoint of p.)

We next see that any map to (Y,θ) from either a surjectively structured algebra or a coalgebra has range contained in the co-founded part.

▶ **Lemma 7.**

1. *Any B-algebra homomorphism $f : (X,\phi) \to (Y,\theta)$ with ϕ surjective, factorizes uniquely as* $\begin{tikzcd}(X,\phi) \arrow[r, "g"] & ({}^\circ\nu p, r_{\nu p}) \arrow[r, tail, "i_{\nu p}"] & (Y,\theta)\end{tikzcd}$
2. *Any B-coalgebra-to-algebra-map $f : (X,\zeta) \to (Y,\theta)$ factorizes uniquely as*

$$\begin{tikzcd}(X,\zeta) \arrow[r, "g"] & ({}^\circ\nu p, r_{\nu p}) \arrow[r, tail, "i_{\nu p}"] & (Y,\theta)\end{tikzcd}$$

Proof. We encompass both cases by supposing a commutative diagram

$$\begin{tikzcd}
Z \arrow[r, "\zeta"] \arrow[dr, two heads, "\phi"'] & BX \arrow[r, "Bf"] & BY \arrow[d, "\theta"] \\
& X \arrow[r, "f"'] & Y
\end{tikzcd}$$

Writing U for the range of f gives
$$\begin{tikzcd}
Z \arrow[r, "\zeta"] \arrow[d, two heads, "\phi"'] & BX \arrow[r, "Be"] \arrow[dr, "Bf"'] & B^\circ U \arrow[r, two heads, "r_U"] \arrow[d, "Bi_{U,Y}"] & {}^\circ p(U) \arrow[dd, tail, "i_{p(U)}"] \\
X \arrow[drrr, "f"] \arrow[d, two heads, "e"'] & & BY \arrow[dr, "\theta"] & \\
{}^\circ U \arrow[rrr, tail, "i_U"'] & & & Y
\end{tikzcd}$$

Diagonal fill-in then gives
$$\begin{tikzcd}
Z \arrow[r, "\zeta"] \arrow[d, two heads, "\phi"'] & BX \arrow[r, "Be"] & B^\circ U \arrow[r, two heads, "r_U"] & {}^\circ p(U) \arrow[dd, tail, "i_{p(U)}"] \\
X \arrow[d, two heads, "e"'] & & & \\
{}^\circ U \arrow[rrr, tail, "i_U"'] \arrow[uurrr, dotted] & & & Y
\end{tikzcd}$$

so U is a postfixpoint of p, so $U \subseteq \nu p$. There is a morphism
$$\begin{tikzcd}
& {}^\circ\nu p \arrow[dr, "i_{\nu p}"] & \\
X \arrow[ur, dotted, "g"] \arrow[rr, "f"'] & & Y
\end{tikzcd}$$

viz. the composite $\begin{tikzcd}X \arrow[r, two heads, "e"] & {}^\circ U \arrow[r, tail, "i_{U,\nu p}"] & \nu p\end{tikzcd}$ because
$$\begin{tikzcd}
& {}^\circ\nu p \arrow[dr, bend left, "i_{\nu p}"] & \\
X \arrow[ur, "g"] \arrow[r, two heads, "e"] \arrow[rr, bend right, "f"'] & {}^\circ U \arrow[u, tail, "i_{U,\nu p}"'] \arrow[r, tail, "i_U"] & Y
\end{tikzcd}$$

Since $i_{\nu p}$ is monic, g is unique and
$$\begin{tikzcd}
Z \arrow[r, "\zeta"] \arrow[dr, two heads, "\phi"'] & BX \arrow[r, "Bg"] & B^\circ\nu p \arrow[d, "r_{\nu p}"] \\
& X \arrow[r, "g"'] & {}^\circ\nu p
\end{tikzcd}$$
commutes.

◀

▶ **Corollary 8.**

1. *The co-founded part of* (Y, θ) *is its coreflection into the full subcategory of* $\mathrm{Alg}(B)$ *on surjectively structured algebras.*
2. *If* (Y, θ) *is corecursive then so is its co-founded part.*

Proof. Each part follows from the corresponding part of Lemma 7. ◀

2.4 Injectively Structured Algebras

Let B be an endofunctor on **Set** preserving injections.

▶ **Lemma 9.** *Let* (Y, θ) *be an injectively structured* B*-algebra. For any* $U \in \mathcal{P}Y$*, the map* $r_U : B^\circ U \to^\circ p(U)$ *is an isomorphism.*

Proof. Def. 4 is factorizing an injection.. ◀

▶ **Theorem 10.** *The (co-founded part)*$^{-1}$ *of an injectively structured, corecursive* B*-algebra is a final* B*-coalgebra.*

Proof. The co-founded part is a corecursive B-algebra by Corollary 8(2) and isomorphically structured by Lemma 9. So we apply Proposition 3(2). ◀

To obtain an initial algebra, we may apply an old result [34, Theorem II.4]

▶ **Theorem 11.** *The least subalgebra of an injectively structured* B*-algebra is an initial* B*-algebra.*

Proof. Consider the endofunction q on $Y \rightharpoonup Z$ that sends a partial function (U, f) to the partial function $({}^\circ p(U),\ {}^\circ p(U) \xrightarrow{r_U^{-1}} B^\circ U \xrightarrow{Bf} BZ \xrightarrow{\phi} Z\ )$.

To show q monotone, if $(U, f) \sqsubseteq (V, g)$ i.e.

$$\begin{tikzcd}
& {}^\circ U \arrow[ld, hook', "i_U"'] \arrow[rd, "f"] \arrow[dd, "i_{U,V}"] & \\
Y & & Z \\
& {}^\circ V \arrow[lu, hook, "i_V"] \arrow[ru, "g"'] &
\end{tikzcd}$$

then Proposition 5 gives

$$\begin{tikzcd}
& {}^\circ p(U) \arrow[ld, hook', "i_{p(U)}"'] \arrow[r, "r_U^{-1}"'] \arrow[dd, "i_{p(U),p(V)}"] \arrow[rrr, bend left, "q(U,f)"] & B^\circ U \arrow[rd, "Bf"] \arrow[dd, "Bi_{U,V}"] & & \\
Y & & & BZ \arrow[r, "\phi"] & Z \\
& {}^\circ p(V) \arrow[lu, hook, "i_{p(V)}"] \arrow[r, "r_V^{-1}"] \arrow[rrru, bend right, "q(V,g)"'] & B^\circ V \arrow[ru, "Bg"'] & &
\end{tikzcd}$$

Now we have

$$\begin{tikzcd}
(Y \rightharpoonup Z) \arrow[r, "q"] \arrow[d, "\mathrm{dom}"'] & (Y \rightharpoonup Z) \arrow[d, "\mathrm{dom}"] \\
\mathcal{P}Y \arrow[r, "p"'] & \mathcal{P}Y
\end{tikzcd}$$

Since $Y \rightharpoonup Z$ has and dom preserves suprema of ordinal chains, we obtain $\mathrm{dom}(\mu q) = \mu p$. Therefore μq is the unique fixpoint of q whose domain is μp, i.e. the unique B-algebra homomorphism from $(\mu p, r_{\mu p})$ to (Z, ϕ). ◀

Returning to our example: we began in Section 2.1 with a corecursive B-algebra of theories that is injectively structured (though we have still to prove that). By Theorem 10, its (co-founded part)$^{-1}$ is a final coalgebra; and by Theorem 11, its least subalgebra is an initial algebra. Both are constructed purely from the logic, as stipulated in Section 1.1.

3 Final Coalgebras From Modal Logic on a Dual Adjunction

In the previous section, we saw an example where modal formulas give rise to an injectively structured corecursive algebra, as required for Theorem 10. We shall now see how this arises in the general setting of modal logic over a dual adjunction [9, 12, 19, 20, 27]. We begin with an explanation of this formulation of modal logic, based on [20].

3.1 Dual Adjunctions

An adjunction $F \dashv G : \mathcal{D} \to \mathcal{C}$ may be described by an isomorphism

$$\mathcal{C}(X, G\Phi) \cong \mathcal{D}(FX, \Phi) \qquad \text{natural in } X \in \mathcal{C}^{\mathrm{op}}, \Phi \in \mathcal{D}.$$

This gives a functor $\mathcal{O} : \mathcal{C}^{\mathrm{op}} \times \mathcal{D} \to \mathbf{Set}$ (also known as a "bimodule" or "profunctor"), sending (X, Φ) to either $\mathcal{C}(X, G\Phi)$ or $\mathcal{D}(FX, \Phi)$; it does not matter which, since they are isomorphic. This suggests an alternative (equivalent) definition of adjunction: as a functor $\mathcal{O}$ together with two isomorphisms

$$\mathcal{C}(X, U\Phi) \cong \mathcal{O}(X, \Phi) \cong \mathcal{D}(FX, \Phi) \qquad \text{natural in } X \in \mathcal{C}^{\mathrm{op}}, \Phi \in \mathcal{D}.$$

For example, we can describe the adjunction $\mathcal{P} \dashv \mathcal{P} : \mathbf{Set}^{\mathrm{op}} \to \mathbf{Set}$ by the isomorphisms

$$\mathbf{Set}(X, \mathcal{P}\Phi) \cong \mathbf{Rel}(X, \Phi) \cong \mathbf{Set}(\Phi, \mathcal{P}X) \qquad \text{natural in } X \in \mathbf{Set}^{\mathrm{op}}, \Phi \in \mathbf{Set}^{\mathrm{op}}.$$

where $\mathbf{Rel}(X, \Phi)$ is the set of relations between X and Φ. In particular, if X is the set of states of a transition system (X, ζ) and Φ is the set of formulas of our logic, the satisfaction relation $\models$ is an element of $\mathbf{Rel}(X, \Phi)$. It corresponds to a map $X \to \mathcal{P}\Phi$ viz. $(\!|-|\!)_{X,\zeta}$ and also to a map $\Phi \to \mathcal{P}X$ sending each formula to its satisfying states. This example is a dual adjunction between $\mathbf{Set}$ and itself; more generally we want a dual adjunction between a category $\mathcal{C}$, whose objects we think of as sets of states, and a category $\mathcal{D}$, whose objects we think of as sets of formulas. We summarize as follows.

▶ **Definition 12.** A *dual adjunction* for a category $\mathcal{C}$ consists of

- a category $\mathcal{D}$
- a functor $\mathcal{O} : \mathcal{C}^{\mathrm{op}} \times \mathcal{D}^{\mathrm{op}} \to \mathbf{Set}$
- functors $\mathcal{O}_* : \mathcal{C}^{\mathrm{op}} \to \mathcal{D}$ and $\mathcal{O}^* : \mathcal{D}^{\mathrm{op}} \to \mathcal{C}$, and isomorphisms

$$\mathcal{C}(X, \mathcal{O}^*\Phi) \;\cong\; \mathcal{O}(X, \Phi) \;\cong\; \mathcal{D}(\Phi, \mathcal{O}_*X) \qquad \text{natural in } X \in \mathcal{C}^{\mathrm{op}}, \Phi \in \mathcal{D}^{\mathrm{op}}.$$

natural in $X \in \mathcal{C}^{\mathrm{op}}$ and $\Phi \in \mathcal{D}^{\mathrm{op}}$.

3.2 Modal Logic on a Dual Adjunction

Recall that, for a set X of states, BX is the set of *single-step behaviours* ending in a state in X. In our example $BX = \mathcal{P}_c(\mathcal{A} \times X)$.

As explained in [20], there are two essential ingredients required to build a modal logic.

Firstly the syntax. For a set Φ of atoms, let $L\Phi$ be the set of *single-layer formulas* with atoms drawn from Φ. In our example, following (1), $L\Phi$ is the set of formulas

$$\langle a \rangle \; (\bigwedge_{i \in I} \phi_i \wedge \bigwedge_{j \in J} \neg \psi_j) \qquad\qquad (\phi_i, \psi_j \in \Phi)$$

More succinctly $L\Phi = \mathcal{A} \times \mathcal{P}_c\Phi \times \mathcal{P}_c\Phi$. General formulas form an initial L-algebra.

Secondly the semantics. Given a relation $\models$ between X and Φ saying which states satisfy which atoms, let $\rho_{X,\Phi}(\models)$ be the induced relation between single-step behaviours (BX) and single-layer formulas ($L\Phi$). In our example, following (2), we have for $s \in BX$

$$s \; (\rho_{X,\Phi}(\models)) \; \langle a \rangle \; (\bigwedge_{i \in I} \phi_i \wedge \bigwedge_{j \in J} \neg \psi_j) \iff \exists x \in s_a. \; (\forall i \in I. x \models \phi_i \wedge \forall j \in J. \; x \not\models \psi_j)$$

The general situation is summarized as follows.

▶ **Definition 13.** For an endofunctor B on $\mathcal{C}$, a *modal logic on a dual adjunction*, or just *modal logic*, consists of

- a dual adjunction $(\mathcal{D}, \mathcal{O})$ for $\mathcal{C}$
- an endofunctor L on $\mathcal{D}$, the *syntax functor*
- a map $\rho_{X,\Phi} : \mathcal{O}(X, \Phi) \to \mathcal{O}(BX, L\Phi)$ natural in $X \in \mathcal{C}^{\mathsf{op}}, \Phi \in \mathcal{D}^{\mathsf{op}}$, called the *semantics*.

We have expressed the semantics ρ in terms of $\mathcal{O}$, but could alternatively express it in terms of $\mathcal{O}_*$ or $\mathcal{O}^*$.

▶ **Proposition 14.** *Let $(\mathcal{D}, \mathcal{O}, L, \rho)$ be a modal logic for an endofunctor B on $\mathcal{C}$.*

1. *There is a unique natural transformation*

$$\begin{tikzcd}
\mathcal{C}^{\mathsf{op}} \arrow[r, "\mathcal{O}_*"] \arrow[d, "B"'] \arrow[dr, phantom, "\Downarrow \rho_*"] & \mathcal{D} \arrow[d, "L"] \\
\mathcal{C}^{\mathsf{op}} \arrow[r, "\mathcal{O}_*"'] & \mathcal{D}
\end{tikzcd}$$

from which $\rho_{X,\Phi}$ may be recovered via

$$\begin{tikzcd}
\mathcal{O}(X, \Phi) \arrow[rr, "\rho_{X,\Phi}"] \arrow[d, dash, "\cong"'] & & \mathcal{O}(BX, L\Phi) \arrow[d, dash, "\cong"] \\
\mathcal{D}(\Phi, \mathcal{O}_* X) \arrow[r, "L_{\Phi, \mathcal{O}_* X}"'] & \mathcal{D}(L\Phi, L\mathcal{O}_* X) \arrow[r, "\mathcal{D}(L\Phi, \rho_*^X)"'] & \mathcal{D}(L\Phi, \mathcal{O}_* BX)
\end{tikzcd}$$

2. *There is a unique natural transformation*

$$\begin{tikzcd}
\mathcal{D}^{\mathsf{op}} \arrow[r, "\mathcal{O}^*"] \arrow[d, "L"'] \arrow[dr, phantom, "\Downarrow \rho^*"] & \mathcal{C} \arrow[d, "B"] \\
\mathcal{D}^{\mathsf{op}} \arrow[r, "\mathcal{O}^*"'] & \mathcal{C}
\end{tikzcd}$$

from which $\rho_{X,\Phi}$ may be recovered via

$$\begin{tikzcd}
\mathcal{O}(X, \Phi) \arrow[rr, "\rho_{X,\Phi}"] \arrow[d, dash, "\cong"'] & & \mathcal{O}(BX, L\Phi) \arrow[d, dash, "\cong"] \\
\mathcal{C}(X, \mathcal{O}^* \Phi) \arrow[r, "B_{X, \mathcal{O}^* \Phi}"'] & \mathcal{C}(BX, B\mathcal{O}^* \Phi) \arrow[r, "\mathcal{C}(BX, \rho^*_\Phi)"'] & \mathcal{C}(BX, \mathcal{O}^* L\Phi)
\end{tikzcd}$$

Before proving this, let us explain these maps in logical terms.

- ρ_*^X associates, to each single-layer formula ϕ built from properties on X, the property on single-step behaviours ending in X that ϕ describes. In our example logic, it sends $\langle a \rangle\ (\bigwedge_{i \in I} \phi_i \wedge \bigwedge_{j \in J} \neg\psi_j)$, where ϕ_i and ψ_j are subsets of X, to the set of $s \in BX$ such that $\exists x \in s_a.\ (\forall i \in I.\ x \in \phi_i \wedge \forall j \in J.\ x \notin \psi_j)$
- ρ_Φ^* associates, to each single-step behaviour ending in theories on Φ, the theory consisting of single-layer formulas constructed from Φ satisfied by that behaviour. In our example, it sends $s \in B\mathcal{P}\Phi$ to the set of formulas $\langle a \rangle\ (\bigwedge_{i \in I} \phi_i \wedge \bigwedge_{j \in J} \neg\psi_j)$, with $\phi_i, \psi_j \in \Phi$, such that $\exists M \in s_a.\ (\forall i \in I.\ \phi_i \in M \wedge \forall j \in J.\ \psi_j \notin M)$

Proof. (of Proposition 14) For part (1), we use calculus of ends.

$$\begin{aligned}
& \text{Maps} \quad \mathcal{O}(X, \Phi) \to \mathcal{O}(BX, L\Phi) \quad \text{natural in } X,\ \Phi \\
\cong\ & \text{Maps} \quad \mathcal{D}(\Phi, \mathcal{O}_* X) \to \mathcal{D}(L\Phi, \mathcal{O}_* BX) \quad \text{natural in } X,\ \Phi \\
\cong\ & \int_{X,\Phi} \mathbf{Set}(\mathcal{D}(\Phi, \mathcal{O}_* X), \mathcal{D}(L\Phi, \mathcal{O}_* BX)) \\
\cong\ & \int_X \mathcal{D}(L\mathcal{O}_* X, \mathcal{O}_* BX) \qquad \text{(by the Yoneda Lemma)} \\
\cong\ & \text{Maps} \quad L\mathcal{O}_* X \to \mathcal{O}_* BX \quad \text{natural in } X
\end{aligned}$$

Tracing through the bijection backwards gives the constructions described. Part (2) is proved similarly. ◀

Explicitly, ρ_*^X is the image of $\mathsf{id}_{\mathcal{O}_* X}$ in the composite

$$\mathcal{D}(\mathcal{O}_* X, \mathcal{O}_* X) \ \cong\ \mathcal{O}(X, \mathcal{O}_* X) \xrightarrow{\rho_{X,\mathcal{O}_* X}} \mathcal{O}(BX, L\mathcal{O}_* X) \ \cong\ \mathcal{D}(L\mathcal{O}_* X, \mathcal{O}_* BX)$$

and ρ_Φ^* is the image of $\mathsf{id}_{\mathcal{O}^* \Phi}$ in the composite

$$\mathcal{C}(\mathcal{O}^* \Phi, \mathcal{O}^* \Phi) \ \cong\ \mathcal{O}(\mathcal{O}^* \Phi, \Phi) \xrightarrow{\rho_{\mathcal{O}^* \Phi, \Phi}} \mathcal{O}(B\mathcal{O}^* \Phi, L\Phi) \ \cong\ \mathcal{C}(B\mathcal{O}^* \Phi, \mathcal{O}^* L\Phi)$$

3.3 Relating States to Modal Formulas

In this section, let $(\mathcal{D}, \mathcal{O}, L, \rho)$ be a modal logic for an endofunctor B on $\mathcal{C}$.

We want to relate B-coalgebras (transition systems) to the initial L-algebra (the set of formulas).

▶ **Definition 15.**

1. An $(\mathcal{O}|\rho)$-*connection* between a B-coalgebra (X, ζ) and an L-coalgebra (Φ, γ) is a fixpoint of the endofunction $\mathcal{O}(X, \Phi) \xrightarrow{\rho_{X,\Phi}} \mathcal{O}(BX, L\Phi) \xrightarrow{\mathcal{O}(\zeta, \gamma)} \mathcal{O}(X, \Phi)$.
2. Let $(\mathcal{O}|\rho)_*\ :\ \mathrm{Coalg}(B) \to \mathrm{Alg}(L)$ be the functor sending (X, ζ) to
$$(\mathcal{O}_* X,\ \ L\mathcal{O}_* X \xrightarrow{\rho_*^X} \mathcal{O}_* BX \xrightarrow{\mathcal{O}_* \zeta} \mathcal{O}_* X\ )$$
3. Let $(\mathcal{O}|\rho)^*\ :\ \mathrm{Coalg}(L) \to \mathrm{Alg}(B)$ be the functor sending (Φ, γ) to
$$(\mathcal{O}^* \Phi,\ \ B\mathcal{O}^* \Phi \xrightarrow{\rho_\Phi^*} \mathcal{O}^* L\Phi \xrightarrow{\mathcal{O}^* \gamma} \mathcal{O}^* \Phi\ )$$

▶ **Proposition 16.** *Let (X, ζ) be a B-coalgebra and (Φ, γ) an L-coalgebra. For $f \in \mathcal{O}(X, \Phi)$ corresponding to $f_*\ :\ \Phi \to \mathcal{O}_* X$ and $f^*\ :\ X \to \mathcal{O}^* \Phi$, the following are equivalent.*

- *f is an $(\mathcal{O}|\rho)$-connection between (X, ζ) and (Φ, γ).*
- *f_* is an L-coalgebra-to-algebra map $(\Phi, \gamma) \to (\mathcal{O}|\rho)_*(X, \zeta)$.*
- *f^* is a B-coalgebra-to-algebra map $(X, \zeta) \to (\mathcal{O}|\rho)^*(\Phi, \gamma)$.*

Proof. The following diagram commutes:

$$\begin{tikzcd}
\mathcal{C}(X, \mathcal{O}^*\Phi) \arrow[r, "B_{X,\mathcal{O}^*\Phi}"] \arrow[d, dash, "\cong"'] & \mathcal{C}(BX, B\mathcal{O}^*\Phi) \arrow[r, "\mathcal{C}(BX,\rho^*_\Phi)"'] \arrow[rr, bend left, "\mathcal{C}(\zeta,(\rho^*_\Phi;\mathcal{O}^*\gamma))"] & \mathcal{C}(BX, \mathcal{O}^*L\Phi) \arrow[r, "\mathcal{C}(\zeta,\mathcal{O}^*\gamma)"'] \arrow[d, dash, "\cong"'] & \mathcal{C}(X, \mathcal{O}^*\Phi) \arrow[d, dash, "\cong"] \\
\mathcal{O}(X, \Phi) \arrow[rr, "\rho_{X,\Phi}"] \arrow[d, dash, "\cong"'] & & \mathcal{O}(BX, L\Phi) \arrow[r, "\mathcal{O}(\zeta,\gamma)"] \arrow[d, dash, "\cong"'] & \mathcal{O}(X, \Phi) \arrow[d, dash, "\cong"] \\
\mathcal{D}(\Phi, \mathcal{O}_*X) \arrow[r, "L_{\Phi,\mathcal{O}_*X}"'] & \mathcal{D}(L\Phi, L\mathcal{O}_*X) \arrow[r, "\mathcal{D}(L\Phi,\rho^X_*)"] \arrow[rr, bend right, "\mathcal{D}(\gamma,(\rho^X_*;\mathcal{O}_*\zeta))"'] & \mathcal{D}(L\Phi, \mathcal{O}_*BX) \arrow[r, "\mathcal{D}(\gamma,\mathcal{O}_*\zeta)"] & \mathcal{D}(\Phi, \mathcal{O}_*X)
\end{tikzcd}$$

An $(\mathcal{O}|\rho)$-connection between (X, ζ) and (Φ, γ) is a fixpoint of the central line.

An L-coalgebra-to-algebra map $(\Phi, \gamma) \to (\mathcal{O}|\rho)_*(X, \zeta)$ is a fixpoint of the bottom line.

A B-coalgebra-to-algebra map $(X, \zeta) \to (\mathcal{O}|\rho)^*(\Phi, \gamma)$ is a fixpoint of the top line. ◀

▶ **Corollary 17.**

1. *The functor $(\mathcal{O}|\rho)_*$ sends recursive B-coalgebras to corecursive L-algebras.*
2. *The functor $(\mathcal{O}|\rho)^*$ sends recursive L-coalgebras to corecursive B-algebras.*

Now suppose we have an initial L-algebra μL, and regard this as the set of all L-formulas. Let (X, ζ) be a B-coalgebra.

- The unique $(\mathcal{O}|\rho)$-connection between (X, ζ) and $(\mu L)^{-1}$ is regarded as the satisfaction relation $\models$ between states and formulas.
- The unique L-coalgebra-to-algebra map $(\mu L)^{-1} \to (\mathcal{O}|\rho)_*(X, \zeta)$ can be described more simply as the unique L-algebra homomorphism $\mu L \to (\mathcal{O}|\rho)_*(X, \zeta)$. We regard this as the function sending each L-formula to the set of states that satisfy it.
- The unique B-coalgebra-to-algebra map $(X, \zeta) \to (\mathcal{O}|\rho)^*((\mu L)^{-1})$ is regarded as the function $(\!|-|\!)_{X,\zeta}$ sending each state to the set of formulas it satisfies.

We have now seen that $(\mathcal{O}|\rho)^*((\mu L)^{-1})$ is a corecursive B-algebra. The other requirement of Theorem. 10, the injective structure, will be addressed in the next section.

▶ Remark. Proposition 16 and Corollary 17, as well as corresponding results for primitive recursion and corecursion, have recently appeared (for covariant adjunctions) as part of a general account of recursion schemes [17, Theorems 3.4 and 5.6].

3.4 Expressive Modal Logics

The key notion of [20] is the following abstract definition of an expressive modal logic.

▶ **Definition 18.** A modal logic $(\mathcal{D}, \mathcal{O}, L, \rho)$ for an injection-preserving endofunctor B on **Set** is said to be *expressive* when ρ^*_Φ is injective for every $\Phi \in \mathcal{D}$.

For such a logic, we can state our main theorem.

▶ **Theorem 19.** *Let $(\mathcal{D}, \mathcal{O}, L, \rho)$ be an expressive modal logic for an injection-preserving endofunctor B on* **Set** *. Let μL be an initial algebra for L.*

1. *The (co-founded part)$^{-1}$ of $(\mathcal{O}|\rho)^*((\mu L)^{-1})$ is a final B-coalgebra.*
2. *The least subalgebra of $(\mathcal{O}|\rho)^*((\mu L)^{-1})$ is an initial B-algebra.*

Proof. Prop. 2(1) tells us that $(\mu L)^{-1}$ is an isomorphically structured recursive coalgebra. So $(\mathcal{O}|\rho)^*((\mu L)^{-1})$ is a corecursive B-algebra by Corollary 17(2), and injectively structured by the definition of $(\mathcal{O}|\rho)^*$. So part (1) follows from Theorem 10, and part (2) from Theorem 11. ◀

It remains to establish that our example of a modal logic from described in Sect. 3.2 is expressive.

Proof. (essentially the same as [20, Section 6.1])

Let $s, t \in B\mathcal{O}^*\Phi$ with $\rho^*_\Phi s = \rho^*_\Phi t$. For $a \in \mathcal{A}$, we want to show $s_a = t_a$.

Let $M \in s_a$. Define the sets $I = \{N \in t_a \mid M \not\subseteq N\}$ and $J = \{N \in t_a \mid N \not\subseteq M\}$, which are countable since t_a is. For $N \in I$ choose $\phi_N \in M \setminus N$, and for $N \in J$ choose $\psi_N \in N \setminus M$. The formula $\langle a \rangle\ (\bigwedge_{N\in I} \phi_N \wedge \bigwedge_{N \in J} \neg\psi_N)$ is in $\rho^*_\Phi s = \rho^*_\Phi t$, so there is $P \in t_a$ such that

1. for all $N \in I$, $\phi_N \in P$ (implying $P \neq N$);
2. for all $N \in J$, $\psi_N \notin P$ (implying $P \neq N$).

(1) gives $P \notin I$, so $M \subseteq P$. (2) gives $P \notin J$, so $P \subseteq M$. Thus $P = M$, giving $M \in t_a$.

Likewise $M \in t_a$ implies $M \in s_a$. ◀

4 Beyond Set

In this section we generalize our results to categories other than **Set**. We give our general results in Section 4.1, and examine the special cases of **Poset** in Section 4.2 and $\mathbf{Set}^{\mathrm{op}}$ in Section 4.3.

4.1 General Results

We work with a category $\mathcal{C}$ equipped with an *orthogonal factorization system* $(\mathcal{E}, \mathcal{M})$. This consists of two lluf subcategories $\mathcal{E}$ and $\mathcal{M}$ of $\mathcal{C}$, containing all isomorphisms, with every $\mathcal{C}$-morphism $X \xrightarrow{f} Y$ having an essentially unique factorization into an $\mathcal{E}$-morphism $X \overset{e}{\twoheadrightarrow} U$ and a $\mathcal{M}$-morphism $U \overset{m}{\rightarrowtail} Y$ See e.g. [3] for an account of these systems. Here are some examples:

- on **Set**, let $\mathcal{E}$ consist of surjections, and $\mathcal{M}$ of injections;
- on **Poset**, let $\mathcal{E}$ consist of surjective maps, and $\mathcal{M}$ of order-reflecting (hence injective) maps;
- on $\mathbf{Set}^{\mathrm{op}}$, let $\mathcal{E}$ consists of injections, and $\mathcal{M}$ of surjections.

We further require that all $\mathcal{M}$-morphisms are monic, and the $\mathcal{M}$-subobjects of any object form a small complete lattice. (A stronger assumption, which apparently includes all examples of interest, is that $\mathcal{C}$ is equipped with a well-powered sink factorization system. See [3] for an account of source and sink factorization.)

Let B be an $\mathcal{M}$-preserving endofunctor on $\mathcal{C}$. We adapt our results as follows; the proofs are essentially unchanged.

▶ **Theorem 20** (generalizing Theorem 10)**.** *Let (Y, θ) be an $\mathcal{M}$-structured, corecursive B-algebra. Then its (co-founded part)$^{-1}$ is a final B-coalgebra.*

▶ **Theorem 21** ([34, Theorem II.4], generalizing Theorem 11)**.** *Suppose that $\mathcal{M}$ has, and the inclusion $\mathcal{M} \subseteq \mathcal{C}$ preserves, colimits of ordinal chains. Then the least subalgebra of an injectively structured B-algebra is an initial B-algebra.*

Note the extra condition imposed here, needed to ensure the poset $Y \rightharpoonup Z$ has suprema of ordinal chains. The condition is true for **Poset**, but false for $\mathbf{Set}^{\mathrm{op}}$, where $\mathcal{M}$ lacks an initial object because it is the opposite of the category of surjections.

▶ **Theorem 22** (generalizing Theorem 19). *Let $(\mathcal{D}, \mathcal{O}, L, \rho)$ be a modal logic for B that is* $\mathcal{M}$-expressive, *i.e. $\rho^*_\Phi \in \mathcal{M}$ for every $\Phi \in \mathcal{D}$ [19]. Let μL be an initial algebra for L.*

1. *The (co-founded part)$^{-1}$ of $(\mathcal{O}|\rho)^*((\mu L)^{-1})$ is a final B-coalgebra.*
2. *Suppose that $\mathcal{M}$ has, and the inclusion $\mathcal{M} \subseteq \mathcal{C}$ preserves, colimits of ordinal chains. Then the least subalgebra of $(\mathcal{O}|\rho)^*((\mu L)^{-1})$ is an initial B-algebra.*

All the proofs of the above theorems are essentially the same as the ones we gave for **Set**.

We now look at two examples of this more general theory.

4.2 Poset Example

Notation. For a poset X we write

- $\mathsf{Up}\, X$ for the set of upsets
- $\mathsf{Down}\, X$ for the set of downsets
- $\mathsf{Up}_c\, X$ for the set of countably generated upsets
- $\mathsf{Down}_c\, X$ for the set of countably generated downsets.

all ordered by inclusion.

It was shown in [24] following [16, 18, 36] that the collection of states of image-countable transition systems can be characterized modulo *similarity* as a final coalgebra for the endofunctor B on **Poset** sending X to $(\mathsf{Down}_c\, X)^{\mathcal{A}}$. Similarity is also characterized by modal formulas of the form

$$\phi ::= \quad \langle a \rangle \bigwedge_{i \in I} \phi_i$$

Coalgebraic accounts of logic for similarity have been given in [5, 13, 35].

Once again we ask how to construct a final coalgebra directly from the logic. We answer this with the following modal logic $(\mathcal{D}, \mathcal{O}, L, \rho)$ for B.

- $\mathcal{D} = \mathcal{C} = \mathbf{Poset}$.
- $\mathcal{O}(X, \Phi) = \mathsf{Up}\,(X \times \Phi)$, because if $x \models \phi$ and $x \lesssim y$ and $\phi \Rightarrow \psi$ then $y \models \psi$.
- $\mathcal{O}_*$ and $\mathcal{O}^*$ are Up, with evident natural isomorphisms $\mathbf{Poset}(X, \mathsf{Up}\,\Phi) \cong \mathsf{Up}\,(X \times \Phi) \cong \mathbf{Poset}(\Phi, \mathsf{Up}\, X)$.
- L maps Φ to the set of formulas $\langle a \rangle \bigwedge_{i \in I} \phi_i$ modulo the following preorder: we have $\langle a \rangle \bigwedge_{i \in I} \phi_i \leqslant \langle b \rangle \bigwedge_{j \in J} \psi_j$ when $a = b$ and for all $j \in J$ there is $i \in I$ with $\phi_i \Rightarrow \psi_j$. More briefly L maps Φ to the poset $\mathcal{A} \times \mathsf{Up}_c \Phi$.
- $\rho_{X,\Phi}(\models)$ is the relation from BX to $L\Phi$ that relates s to $\langle a \rangle \bigwedge_{i \in I} \phi_i$ when $\exists x \in s_a.\ \forall i \in I. x \models \phi_i$.

We deduce the form of ρ_* and ρ^*.

- The function ρ^X_* maps $\langle a \rangle \bigwedge_{i \in I} \phi_i$, where ϕ_i and ψ_j are upsets of X, to the upset of $s \in BX$ such that $\exists x \in s_a.\ \forall i \in I.\ x \in \phi_i$
- The function ρ^*_Φ maps $s \in B\,\mathsf{Up}\,\Phi$ to the upset of formulas $\langle a \rangle \bigwedge_{i \in I} \phi_i$, with $\phi_i, \psi_j \in \Phi$, such that $\exists M \in s_a.\ \forall i \in I.\ \phi_i \in M$

To apply Theorem 22, we show that ρ^*_Φ is order-injective.

Proof. Suppose $s, t \in B\,\mathsf{Up}\,\Phi$ and $\rho^*_\Phi s \subseteq \rho^*_\Phi t$. For $a \in \mathcal{A}$, we want to show $s_a \subseteq t_a$.

Let $M \in s_a$. It is a downset in $\mathsf{Up}\,\Phi$ generated by $\{\phi_i \mid i \in I\}$, where I is countable. The formula $\langle a \rangle \bigwedge_{i \in I} \phi_i$ is in $\rho^*_\Phi(s)$, and hence is in $\rho^*_\Phi(t)$, so there is $N \in t_a$ such that $\forall i \in I.\, \phi_i \in N$. Hence $M \subseteq N$. Since t_a is a downset in $\mathsf{Up}\,\Phi$, we have $M \in t_a$. ◀

We therefore obtain both a final coalgebra and an initial algebra from Theorem 22.

4.3 The Dual Construction

We briefly consider the dual of Theorem 10, i.e. the case of Theorem 20 where $\mathcal{C} = \mathbf{Set}^{\mathsf{op}}$. Here the complete lattice $\mathcal{P}Y$ is replaced by the complete lattice $\mathrm{EqRel}(Y)$ of equivalence relations on Y.

Let B be an endofunctor on **Set**. We now need B to preserve surjections, not injections, but that is automatic since surjections are split epis. An injectively structured coalgebra is sometimes called an *extensional* coalgebra, after ZF set theory's Axiom of Extensionality.

Given a B-coalgebra (Y, ζ), and $(\equiv) \in \mathrm{EqRel}(Y)$, we define $p(\equiv) \in \mathrm{EqRel}(Y)$ to be the kernel of the composite

$$\begin{tikzcd} Y \arrow[r, "\zeta"] & BY \arrow[r, two heads, "Be_{\equiv}"] & B(Y/\equiv) \end{tikzcd}$$

where $e_\equiv : Y \to (Y/\equiv)$ sends $x \mapsto [x]_\equiv$. This gives a square

$$\begin{tikzcd} Y \arrow[r, two heads, "e_{p(\equiv)}"] \arrow[d, "\zeta"'] & Y/p(\equiv) \arrow[d, tail, "r_{\equiv}"] \\ BY \arrow[r, two heads, "Be_{\equiv}"'] & B(Y/\equiv) \end{tikzcd}$$

Then p is a monotone endofunction on $\mathrm{EqRel}(Y)$. Its least prefixpoint μp is called *extensional equivalence*, and the B-coalgebra $(Y/\mu p, r_{\mu p})$ is called the *extensional quotient* of (X, ζ). This is dual to the co-founded part construction. Therefore, dually to Corollary 8(1), the extensional quotient is a reflection of (Y, ζ) into the full subcategory of $\mathrm{Coalg}(B)$ on extensional coalgebras.

The dual of Theorem 10 is as follows.

▶ **Theorem 23.** *Let (X, ζ) be a surjectively structured, recursive B-coalgebra. Then its (extensional quotient)$^{-1}$ is an initial B-algebra.*

We illustrate this with the endofunctor $B : X \mapsto \mathcal{P}_c X$. Let X be the set of well-founded terms built from an ω-ary operation c and a constant d. Let ζ be the function

$$\begin{array}{rcl} c(t_i \mid i \in \mathbb{N}) & \mapsto & \{t_i \mid i \in \mathbb{N}\} \\ d & \mapsto & \{\,\} \end{array}$$

The $\mathcal{P}_c$-coalgebra (X, ζ) is surjectively structured, and it is recursive because it is well-founded [33]. Therefore, by Theorem 23 its (extensional quotient)$^{-1}$ is an initial $\mathcal{P}_c$-algebra.

5 Conclusions and Further Work

We now have a general machinery for building final coalgebras from modal formulas. Many interesting questions remain.

Having considered several least prefixpoints and greatest postfixpoints, we may ask how long it takes to reach these fixpoints.

- If the functor B preserves arbitrary intersections of subsets, then p will preserve nonempty intersections of subsets. Therefore νp will be reached at ω, cf. [37].
- If the functor B preserves κ-filtered colimits then p will do so too. Therefore μp will be reached at κ.

Our example functor $X \mapsto (\mathcal{P}_c X)^{\mathcal{A}}$ preserves intersections and ω_1-filtered colimits, so νp is reached at ω and μp at ω_1, at the latest.

But this leaves the question of functors on **Set** that do not preserve intersection, cf. [37], and also the examples in Section 4. We leave these for future work.

Another task remaining is to consider canonical models for infinitary modal logics, and the relationship with logical completeness results [26, 28, 30, 31].

Finally, there are intriguing connections to explore with the use of algebras, coalgebras and duality in [6, 8, 21], and with the recent general account of recursion schemes in [17].

References

1 Samson Abramsky. A cook's tour of the finitary non-well-founded sets. In Sergei N. Artëmov, Howard Barringer, Artur S. d'Avila Garcez, Luís C. Lamb, and John Woods, editors, *We Will Show Them! Essays in Honour of Dov Gabbay, Volume One*, pages 1–18. College Publications, 2005.

2 Jiří Adámek, Mahdieh Haddadi, and Stefan Milius. Corecursive algebras, corecursive monads and bloom monads. *Logical Methods in Computer Science*, 10(3), 2014.

3 Jiří Adamek, Horst Herrlich, and George Strecker. *Abstract and Concrete Categories – The Joy of Cats*. Wiley, 1990.

4 Jiří Adámek, Paul B. Levy, Stefan Milius, Lawrence S. Moss, and Lurdes Sousa. On final coalgebras of power-set functors and saturated trees. *Applied Categorical Structures*, June 2014.

5 Alexandru Baltag. A logic for coalgebraic simulation. *Electronic Nptes in Theoretical Computer Science*, 33, 2000.

6 Nick Bezhanishvili, Clemens Kupke, and Prakash Panangaden. *Minimization via duality*, volume 7456 LNCS of *Lecture notes in computer science / Theoretical Computer Science and General Issues*, pages 191–205. Springer, 2012.

7 Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*. Cambridge University Press, 2002.

8 Filippo Bonchi, Marcello M. Bonsangue, Helle H. Hansen, Prakash Panangaden, Jan J. M. M. Rutten, and Alexandra Silva. Algebra-coalgebra duality in Brzozowski's minimization algorithm. *ACM Transactions on Computational Logic*, 15(1):3:1–3:29, March 2014.

9 Marcello Bonsangue and Alexander Kurz. Duality for logics of transition systems. In *FOSSACS: International Conference on Foundations of Software Science and Computation Structures*. Lecture Notes in Computer Science, 2005.

10 Venanzio Capretta, Tarmo Uustalu, and Varmo Vene. Recursive coalgebras from comonads. *INFCTRL: Information and Computation (formerly Information and Control)*, 204, 2006.

11 Venanzio Capretta, Tarmo Uustalu, and Varmo Vene. Corecursive algebras: A study of general structured corecursion. In M. Oliveira and J. Woodcock, editors, *Formal Methods: Foundations and Applications, 12th Brazilian Symposium on Formal Methods, SBMF 2009, Gramado, Brazil, Revised Selected Papers*, volume 5902 of *LNCS*, pages 84–100. Springer, 2009.

12 Liang-Ting Chen and Achim Jung. On a categorical framework for coalgebraic modal logic. *Electronic Notes in Theoretical Computer Science*, 308:109–128, 2014.

13 Corina Cîrstea. A modular approach to defining and characterising notions of simulation. *Information and Computation*, 204(4):469–502, 2006.

14 Adam Eppendahl. Coalgebra-to-algebra morphisms. *Electronic Notes in Theoretical Computer Science*, 29:42–49, 1999.

15 Robert Goldblatt. Final coalgebras and the hennessy-milner property. *Annals of Pure and Applied Logic*, 138(1-3):77–93, 2006.

16 Wim H. Hesselink and Albert Thijs. Fixpoint semantics and simulation. *Theoretical Computer Science*, 238(1-2):275–311, 2000.

17 Ralf Hinze, Nicolas Wu, and Jeremy Gibbons. Conjugate hylomorphisms – or: The mother of all structured recursion schemes. In Sriram K. Rajamani and David Walker, editors, *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015, Mumbai, India, January 15-17, 2015*, pages 527–538. ACM, 2015.

18 Jesse Hughes and Bart Jacobs. Simulations in coalgebra. *Theoretical Computer Science*, 327(1-2):71–108, 2004.

19 Bart Jacobs and Ana Sokolova. Exemplaric expressivity of modal logics. *Journal of Logic and Computation*, 20(5):1041–1068, 2010.

20 Bartek Klin. Coalgebraic modal logic beyond sets. *Electronic Notes in Theoretical Computer Science*, 173:177–201, 2007.

21 Bartek Klin and Jurriaan Rot. Coalgebraic trace semantics via forgetful logics. In Andrew M. Pitts, editor, *Foundations of Software Science and Computation Structures – 18th International Conference, FoSSaCS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings*, volume 9034 of *Lecture Notes in Computer Science*, pages 151–166. Springer, 2015.

22 Clemens Kupke and Raul Andres Leal. Characterising behavioural equivalence: Three sides of one coin. In Alexander Kurz, Marina Lenisa, and Andrzej Tarlecki, editors, *Algebra and Coalgebra in Computer Science, Third International Conference, CALCO 2009, Udine, Italy, September 7-10, 2009. Proceedings*, volume 5728 of *Lecture Notes in Computer Science*, pages 97–112. Springer, 2009.

23 Alexander Kurz and Dirk Pattinson. Coalgebraic modal logic of finite rank. *Mathematical Structures in Computer Science*, 15(3):453–473, 2005.

24 Paul B. Levy. Similarity quotients as final coalgebras. In Martin Hofmann, editor, *Proceedings, 14th International Conference on Foundations of Software Science and Computational Structures, Saarbrücken, Germany*, volume 6604 of *Lecture Notes in Computer Science*, pages 27–41. Springer, 2011.

25 Robin Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

26 Pierluigi Minari. Infinitary modal logic and generalized Kripke semantics. *Annali del Dipartimento di Filosofia*, 17(1), 2012.

27 Dusko Pavlovic, Michael W. Mislove, and James Worrell. Testing semantics: Connecting processes and process logics. In Michael Johnson and Varmo Vene, editors, *Algebraic Methodology and Software Technology, 11th International Conference, AMAST 2006, Kuressaare, Estonia, July 5-8, 2006, Proceedings*, volume 4019 of *Lecture Notes in Computer Science*, pages 308–322. Springer, 2006.

28 Slavian Radev. Infinitary propositional normal modal logic. *Studia Logica*, 46(4):291–309, 1987.

29 Jan J. M. M. Rutten. A calculus of transition systems (towards universal coalgebra). In *97*, page 25. Centrum voor Wiskunde en Informatica (CWI), ISSN 0169-118X, January 31 1995. CS-R9503.

30 Lutz Schröder and Dirk Pattinson. Strong completeness of coalgebraic modal logics. In S. Albers and J.-Y. Marion, editors, *Proc. STACS 2009*, volume 09001 of *Dagstuhl Seminar Proceedings*, pages 673–684. Schloss Dagstuhl, 2009.

31 Yoshihito Tanaka and Hiroakira Ono. Rasiowa-Sikorski lemma, Kripke completeness of predicate and infinitary modal logics. In Michael Zakharyaschev, Krister Segerberg, Maarten de Rijke, and Heinrich Wansing, editors, *Advances in Modal Logic*, pages 401–420. CSLI Publications, 1998.

32 Paul Taylor. Towards a unified treatment of induction i: the general recursion theorem. preprint, 1996.

33 Paul Taylor. *Practical Foundations of Mathematics*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, Cambridge, 1999.

34 Vera Trnková, Jiří Adámek, Václav Koubek, and Jan Reiterman. Free algebras, input processes and free monads. *Commentationes Mathematicae Universitatis Carolinae*, 016(2):339–351, 1975.

35 Toby Wilkinson. A characterisation of expressivity for coalgebraic bisimulation and simulation. *Electronic Notes in Theoretical Computer Science*, 286:323–336, 2012.

36 James Worrell. Coinduction for recursive data types: partial orders, metric spaces and ω-categories. *Electronic Notes in Theoretical Computer Science*, 33, 2000.

37 James Worrell. On the final sequence of a finitary set functor. *Theoretical Computer Science*, 338(1–3):184–199, June 2005.

Uniform Interpolation for Coalgebraic Fixpoint Logic

Johannes Marti, Fatemeh Seifan, and Yde Venema

ILLC, Universiteit van Amsterdam, The Netherlands
johannes.marti@gmail.com, F.Seifan@uva.nl, Y.Venema@uva.nl

Abstract

We use the connection between automata and logic to prove that a wide class of coalgebraic fixpoint logics enjoys uniform interpolation. To this aim, first we generalize one of the central results in coalgebraic automata theory, namely closure under projection, which is known to hold for weak-pullback preserving functors, to a more general class of functors, i.e., functors with quasi-functorial lax extensions. Then we will show that closure under projection implies definability of the bisimulation quantifier in the language of coalgebraic fixpoint logic, and finally we prove the uniform interpolation theorem.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases μ-calculus, uniform interpolation, coalgebra, automata

Digital Object Identifier 10.4230/LIPIcs.CALCO.2015.238

1 Introduction

The connection between automata and logic goes back to the early seventies by the works of Büchi [3] and Elgot [6], who showed that finite automata and monadic second-order logic have the same expressive power over finite words, and that the transformations from formulas to automata and vice versa are effective. This connection has found important applications and landmark results, such as Rabin's decidability theorem [18]. During the last twenty years study of the link between automata and logic has been continued and many interesting results have been obtained, such as results in [10], where Janin and Walukiewicz established the connection between the modal μ-calculus and parity automata operating on labeled transition systems.

The coalgebraic perspective on the link between automata and logic has been uniformly studied in [22], where the author introduces the notion of a coalgebra automaton and establishes the connection between these automata and coalgebraic fixpoint logic based on Moss' modality ∇ [15]. Coalgebraic fixpoint logic is a powerful extension of coalgebraic modal logic [15] with fixpoint operators. The main contribution of this paper will be to add *uniform interpolation* to the list of properties of coalgebraic fixpoint logic.

A logic has *interpolation* if, whenever we have formulas a and b such that $\models a \to b$ (meaning that the formula $a \to b$ holds in every model), then there is an *interpolant* formula c in the *common language* of a and b (i.e., c may use only propositional letters that appear both in a and b), such that $\models a \to c$ and $\models c \to b$. This notion is familiar from first-order logic, and is known there as Craig interpolation [4] . Some logics enjoy a much stronger version of interpolation, namely uniform interpolation, which has been introduced by Pitts in [17]. A logic has uniform interpolation if the interpolant c does not really depend on b itself, but only on the language that b shares with a. Although it is easy to show that classical propositional logic has uniform interpolation, not many logics have this property, for instance first-order logic has interpolation, but it does not enjoy the uniform version [9].

6th International Conference on Algebra and Coalgebra in Computer Science (CALCO'15).
Editors: Lawrence S. Moss and Pawel Sobocinski; pp. 238–252
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

As a motivation for studying uniform interpolation, let us mention some recent works on this property. Starting with the seminal work of Pitts [17] who introduced this version of interpolation and proved that intuitionistic logic has uniform interpolation, the study of this property for different logics has been actively pursued by various authors. In modal logic, Shavrukov [21] proved that the Gödel-Löb logic **GL** has uniform interpolation. Subsequently, Ghilardi [7] and Visser [23] independently established the property for modal logic **K**, while [8] contains negative results for modal logic **S4**. In the theory of modal fixpoint logic D'Agostino and Hollenberg proved that the modal μ-calculus has uniform interpolation [5].

In this paper we study uniform interpolation in the context of coalgebraic fixpoint logic. More specifically, we restrict attention to set functors T that preserve finite sets and that admit a so-called quasi-functorial lax extension L, that is, a certain kind of relation lifting satisfying somewhat weaker conditions than the standard Barr extension (see Definition 2.11 for the details). This class of functors includes the ones that preserve weak pullbacks, but also the monotone neighborhood functor, and it is closed under various natural operations on functors, see Fact 2.15. For each of these functors we consider a coalgebraic modal fixpoint logic $\mu\mathcal{L}^{\mathsf{T}}_{L}$, in the style of [22], where the semantics of the Moss-style modality is given by the relation lifting L (a definition is given in section 3). Our main result, Theorem 5.3 states that the resulting logic enjoys the property of uniform interpolation; our proof also applies to the fixpoint-free fragment of the logic.

As usual in the setting of modal logic, our proof is based on the link between uniform interpolation and the definability of a certain nonstandard second-order quantifier, the so-called *bisimulation quantifier*. More specifically, our aim will be to define, for each proposition letter p, a map $\exists p$ on $\mu\mathcal{L}^{\mathsf{T}}_{L}$, and prove that this map satisfies

$$\mathbb{S}, s \Vdash \exists p.b \quad \text{iff} \quad \mathbb{S}', s' \Vdash b, \text{ for some } \mathbb{S}', s' \text{ with } \mathbb{S}, s \leftrightarroweq_p \mathbb{S}', s', \tag{1}$$

for all pointed coalgebras $(\mathbb{S}, s)$. Here $\leftrightarroweq_p$ denotes bisimilarity, with respect to the relation lifting L, up to the proposition letter p.

Our proof follows the automata-theoretic approach by D'Agostino and Hollenberg. That is, in section 3.2 we define a class of nondeterministic parity automata that closely correspond to our language, in the sense that there are effective translation transforming a $\mu\mathcal{L}^{\mathsf{T}}_{L}$-formula into an equivalent automaton, and vice versa. Our main technical result, generalizing earlier work by Kupke and the third author [12], is in section 4; it provides a construction revealing that the class of coalgebra automata associated with our logic, is closed under projection. From this we may easily derive the definability of bisimulation quantifiers in our logic.

In order to finish the introduction, let us mention some related work. First of all, our paper should be considered as the publication of results from the first author's MSc thesis [13] on uniform interpolation for the fixpoint-free fragment of our language. Pattinson [16] introduced a variant of Moss-style coalgebraic modal logic which nicely works for all set functors: the so-called logic of exact covers, and he showed that this logic enjoys uniform interpolation. Since the modality of this language seems to be inherently non-monotonic, it is not so clear how to extend his result to the setting with fixpoint operators.

Overview. We first fix notation and terminology and equip the reader with the necessary background material. In section 3 we introduce coalgebraic fixpoint logic and give a brief introduction to automata theory. After that, we prove in section 4 our main technical result. We show that if functor $\mathsf{T} : \mathsf{Set} \to \mathsf{Set}$ has a quasi-functorial lax extension L which preserves diagonals, then T-automata are closed under projection. Finally in section 5 we combine the results from section 3 and section 4 in order to prove uniform interpolation for coalgebraic fixpoint logic $\mu\mathcal{L}^{\mathsf{T}}_{L}$. We finish the paper in section 6 with an outlook on future results.

2 Preliminaries

This section contains some of the preliminaries and fixes the notation. We presuppose that reader has made contact with basic concepts from category theory before. For example we assume familiarity with basic notions such as categories, functors, natural transformations and isomorphic categories.

We also presupposes knowledge of the theory of coalgebras. An extensive introduction is given for example in [19].

2.1 Set Functors

We will work in the category Set, that has sets as objects and functions as arrows. For sets $X' \subseteq X$, the inclusion map from X' to X is denoted by $i_{X',X} : X' \hookrightarrow X$, $x \mapsto x$. For a function $f : X \to Y$ we define the set $Rng(f) = \{y \in Y \mid \exists x \in X, f(x) = y\} \subseteq Y$. In the following we assume, if not explicitly stated otherwise, that functors are covariant endofunctors in the category Set.

We first introduce some of the functors that concern us in this paper. The *powerset functor* is the functor $\mathcal{P} : \mathsf{Set} \to \mathsf{Set}$, which maps a set S to the set of all its subsets $\mathcal{P}S = \{V \mid V \subseteq S\}$. A function $f : S \to T$ is mapped to $\mathcal{P}f : \mathcal{P}S \to \mathcal{P}T$, which is defined for any $V \subseteq S$ by $\mathcal{P}f(V) = f[V] = \{f(v) \mid v \in V\}$. The *contravariant powerset* functor $\breve{\mathcal{P}}$ also maps a set S to $\breve{\mathcal{P}}S = \mathcal{P}S$. On functions $\breve{\mathcal{P}}$ is the inverse image map, that is for an $f : S \to T$ we have $\breve{\mathcal{P}}f : \breve{\mathcal{P}}S \to \breve{\mathcal{P}}T$, $V \mapsto f^{-1}[V]$. The *neighborhood* functor $\mathcal{N} = \breve{\mathcal{P}}\breve{\mathcal{P}}$ is the double contravariant powerset functor. Given a set S and an element $\alpha \in \mathcal{N}S$, we define

$$\alpha^{\uparrow} := \{X \in \mathcal{P}S \mid Y \subseteq X \text{ for some } Y \in \alpha\}$$

and we say that α is *upward closed* if $\alpha = \alpha^{\uparrow}$. The *monotone neighborhood* functor $\mathcal{M}$ is the restriction of the neighborhood functor to upward closed sets. More concretely the functor $\mathcal{M}$ is given by $\mathcal{M}S := \{\beta \in \mathcal{N}S \mid \beta \text{ is upward closed}\}$, while for $f : S \to T$, we define $\mathcal{M}f : \mathcal{M}S \to \mathcal{M}T$ by $\mathcal{M}f(\beta) := (\mathcal{N}f(\beta))^{\uparrow}$.

▶ **Definition 2.1.** $\mathsf{T} : \mathsf{Set} \to \mathsf{Set}$ *preserves inclusions* if $\mathsf{T}i_{A,B} = i_{\mathsf{T}A,\mathsf{T}B}$ for all sets $A \subseteq B$.

▶ **Proposition 2.2.** *If* $\mathsf{T} : Set \to Set$ *preserves inclusions, then* $\mathsf{T}(Rng(f)) = Rng\mathsf{T}(f)$ *for any function* f *in* Set.

In coalgebraic logic one pays special attention to finitary, finite set preserving functors.

▶ **Definition 2.3.** A functor T *preserves finite sets* if $\mathsf{T}X$ is finite whenever X is.

An inclusion preserving functor T is called *finitary* if it satisfies

$$\mathsf{T}X = \bigcup\{\mathsf{T}X' \subseteq \mathsf{T}X \mid X' \subseteq X,\ X' \text{ is finite }\}$$

for all sets X. The finitary version T_{ω} of an inclusion preserving functor T is defined such that it maps a set X to $\mathsf{T}_{\omega}X = \bigcup\{\mathsf{T}X' \mid X' \subseteq X,\ X' \text{ is finite }\}$, and a function f to itself.

These definitions can be simply generalized to the class of all set functors.

An example of a finitary version of a functor is $\mathcal{P}_{\omega}$, that maps a set X to the set of all its finite subsets. An other important class of set functors in the context of coalgebraic modal logic is the class of intersection preserving functors.

▶ **Definition 2.4.** A set functor T *preserves finite intersections* if for all sets A and B, $\mathsf{T}(A \cap B) = \mathsf{T}A \cap \mathsf{T}B$.

2.2 Coalgebras

In the following part of this section, we will briefly recall the basic notions from the theory of coalgebras that we will use later. For a detailed introduction into coalgebras see e.g. [19].

▶ **Definition 2.5.** Given a set functor T, a T-*coalgebra* is a pair $\mathbb{S} = (S, \sigma)$ with $\sigma : S \to \mathsf{T}S$. A pointed T-coalgebra is a pair consisting of a T-coalgebra together with an element of (the carrier set of) that coalgebra. A T-*coalgebra morphism* from T-coalgebra $\mathbb{S} = (S, \sigma)$ to $\mathbb{S}' = (S', \sigma')$, written $f : \mathbb{S} \to \mathbb{S}'$, is a function $f : S \to S'$ such that $\mathsf{T}(f) \circ \sigma = \sigma' \circ f$. The collection of T-coalgebras with their morphisms form a category denoted by $Coalg(\mathsf{T})$.

▶ **Definition 2.6.** Let T be an endofunctor on the category Set, and C an arbitrary set of objects that we shall call *colors*. We let T_C denote the functor $\mathsf{T}_C S = \mathsf{T}S \times C$; that is, T_C maps a set S to the set $\mathsf{T}S \times C$ (and a function $f : S \to S'$ to the function $\mathsf{T}f \times id_C : \mathsf{T}S \times C \to \mathsf{T}S' \times C$). T_C-coalgebras will also be called C-*colored* T-*coalgebras*. We will usually denote T_C-coalgebras as triples $\mathbb{S} = (S, \sigma, \gamma)$, with $\sigma : S \to \mathsf{T}S$ the coalgebra map and $\gamma : S \to C$ the *coloring (marking)*.

▶ **Convention 2.7.** From now on in all our investigations, without lose of generality, we can assume set functor T preserves inclusions and finite intersections. Indeed given any T we can find a naturally isomorphic T' that preserves inclusions and finite intersections. The details can be found in [1]. The important point for us is that the category of $Coalg(\mathsf{T})$ and $Coalg(\mathsf{T}')$ are isomorphic.

2.3 Relation Lifting and Bisimulation

In the remaining part of this section we introduce the notion of relation lifting to define a very general notion of bisimulation for coalgebras. First we recall some central definitions and fix mathematical notation and terminology. Given sets X and Y, we denote a relation R between X and Y by $R : X \nrightarrow Y$ to specify its domain X and codomain Y. We write $R; S : X \nrightarrow Z$ for the composition of two relations $R : X \nrightarrow Y$ and $S : Y \nrightarrow Z$ and $R^\circ : Y \nrightarrow X$ for the converse of $R : X \nrightarrow Y$ with $(y, x) \in R^\circ$ iff $(x, y) \in R$. The graph of any function $f : X \to Y$ is a relation $f : X \nrightarrow Y$ between X and Y for which we also use the symbol f. It will be clear from the context in which a symbol f occurs whether it is meant as a function or a relation. Note that the composition of functions is denoted the other way round than the composition of relations, so we have $g \circ f = f; g$ for functions $f : X \to Y$ and $g : Y \to Z$. For a relation $R : X \nrightarrow Y$ we define the sets

$$\begin{aligned} Dom(R) &= \{x \in X \mid \exists y \in Y, (x, y) \in R\} \subseteq X, \\ Rng(R) &= \{y \in Y \mid \exists x \in X, (x, y) \in R\} \subseteq Y. \end{aligned}$$

The relation $R : X \nrightarrow Y$ is *full* on X if $Dom(R) = X$ and is full on Y if $Rng(R) = Y$. Given sets $X' \subseteq X$ and $Y' \subseteq Y$, we define the restriction $R\mid_{X' \times Y'}: X' \nrightarrow Y'$ of the relation $R : X \nrightarrow Y$ as $R\mid_{X' \times Y'} = R \cap (X \times Y)$. For any set X let $\in_X : X \to \mathcal{P}X$ be the *membership relation* between elements of X and subsets of X. Given a set X we define the *diagonal relation* $\Delta_X : X \nrightarrow X$ with $(x, x') \in \Delta_X$ iff $x = x'$. Note that $\Delta_X = id_X$, where id_X is the graph of the identity function.

▶ **Definition 2.8.** A relation lifting L for a set functor T is a collection of relations LR for every relation R, such that $LR : \mathsf{T}X \nrightarrow \mathsf{T}Y$ if $R : X \nrightarrow Y$. We require relation liftings to preserve converse, this means that $L(R^{\circ}) = (LR)^{\circ}$ for all relations R.

▶ **Example 2.9.**

(i) The *Egli-Milner lifting* $\overline{\mathcal{P}}$ is a relation lifting for covariant power set functor $\mathcal{P}$ that is defined for any $R : X \nrightarrow Y$ such that $\overline{\mathcal{P}}R = \overrightarrow{\mathcal{P}}R \cap \overleftarrow{\mathcal{P}}R$, where:

$$\overrightarrow{\mathcal{P}}R := \{(U,V) \in \mathcal{P}X \times \mathcal{P}Y \mid \forall u \in U \; \exists v \in V \text{ s.t. } (u,v) \in R\},$$

$$\overleftarrow{\mathcal{P}}R := \{(U,V) \in \mathcal{P}X \times \mathcal{P}Y \mid \forall v \in V \; \exists u \in U \text{ s.t. } (u,v) \in R\}.$$

(ii) For the constant functor D of a fixed set D define a relation lifting $\overline{D}$ for any $R : X \nrightarrow Y$ such that $\overline{D}R = \Delta_D$.

(iii) Recall the notion of $\overrightarrow{\mathcal{P}}R$ from (i) we can define a relation lifting $\widetilde{\mathcal{M}}$ for the monotone neighborhood functor $\mathcal{M}$ on a relation $R : X \nrightarrow Y$ as follows:

$$\widetilde{\mathcal{M}}R := \overrightarrow{\mathcal{P}}\overleftarrow{\mathcal{P}}R \cap \overleftarrow{\mathcal{P}}\overrightarrow{\mathcal{P}}R.$$

An important use of relation liftings is to yield a notion of bisimulation.

▶ **Definition 2.10.** Let L be a relation lifting for T and $\mathbb{S} = (S, \sigma)$ and $\mathbb{S}' = (S', \sigma')$ be two T-coalgebras. An *L-bisimulation* between $\mathbb{S}$ and $\mathbb{S}'$ is a relation $R : S \nrightarrow S'$ such that $(\sigma(s), \sigma'(s')) \in LR$, for all $(s,s') \in R$. Two states $s \in \mathbb{S}$ and $s' \in \mathbb{S}'$ are L-bisimilar if there is an L-bisimulation R between $\mathbb{S}$ and $\mathbb{S}'$ with $(s,s') \in R$. We write $\underline{\leftrightarrow}^L$ for the notion of L-bisimilarity between fixed coalgebras. Given two C-colored T-coalgebras $\mathbb{S} = (S, \sigma, \gamma)$ and $\mathbb{S}' = (S', \sigma', \gamma')$ and a relation lifting L for T, a relation $R : S \nrightarrow S'$ is an *L_C-bisimulation* between $\mathbb{S}$ and $\mathbb{S}'$, whenever $(\sigma(s), \sigma'(s')) \in LR$ and $\gamma(s) = \gamma'(s')$ for all $(s,s') \in R$.

Now we will give the definition of lax extensions, which are relation liftings satisfying certain conditions that make them well-behaved in the context of coalgebra.

▶ **Definition 2.11.** A relation lifting L for a functor T is called a *lax extension* of T if it satisfies, for all relations $R, R' : X \nrightarrow Z$ and $S : Z \nrightarrow Y$ and all functions $f : X \to Z$:

(L1) $R' \subseteq R$ implies $LR' \subseteq LR$,

(L2) $LR; LS \subseteq L(R;S)$,

(L3) $\mathsf{T}f \subseteq Lf$.

We say that a lax extension L *preserves diagonals* if it additionally satisfies:

(L4) $L\Delta_X \subseteq \Delta_{\mathsf{T}X}$.

We call a lax extension L of T *functorial*, if it distributes over composition, i.e., if $LR; LS = L(R;S)$, and *quasi-functorial*, if

$$LR; LS = L(R;S) \cap (Dom(LR) \times Rng(LS))$$

for all relations $R : X \nrightarrow Z$ and $S : Z \nrightarrow Y$.

▶ **Example 2.12.** The relation lifting $\widetilde{\mathcal{M}}$ which has been defined in Example 2.9, is quasi-functorial.

▶ **Proposition 2.13.** *Let T be a set functor and let L be a quasi-functorial lax extension for T. Then we have:*

(1) *L preserves fullness: If $R : X \nrightarrow Z$ is full on both sides, then so is $LR : \mathsf{T}X \nrightarrow \mathsf{T}Z$;*

(2) *If* $R : X \nrightarrow Z$ *is full on* X *and* $i : Z \hookrightarrow Z'$ *is the inclusion map between* Z *and* Z' *then* $L(R; i)$ *is full on* $\mathsf{T}X$*;*

(3) *If* L *preserves diagonals then for any function* f*,* $\mathsf{T}f = Lf$*.*

Let us now summarize some facts that we will use about L-bisimulations in the sequel.

▶ **Proposition 2.14.** *For a lax extension* L *of* T *and* T*-coalgebras* $\mathbb{S}$*,* $\mathbb{S}'$ *and* $\mathbb{Q}$ *the following hold:*

(1) *The graph of a coalgebra morphism* f *from* $\mathbb{S}$ *to* $\mathbb{S}'$ *is an* L*-bisimulation between* $\mathbb{S}$ *and* $\mathbb{S}'$*;*

(2) *if* $R : S \nrightarrow Q$ *respectively* $R' : Q \nrightarrow S'$ *are* L*-bisimulations between* $\mathbb{S}$ *and* $\mathbb{Q}$ *respectively* $\mathbb{Q}$ *and* $\mathbb{S}'$*, then* $R; R' : S \nrightarrow S'$ *is an* L*-bisimulation between* $\mathbb{S}$ *and* $\mathbb{S}'$*.*

For the proof we refer to [14, Proposition 3].

We will finish this section with a remark on some of the closure properties of the class of functors with a quasi-functorial lax extension:

▶ **Fact 2.15.** *The collection of functors with a quasi-functorial lax extension (FQL) has the following properties:*

1. *the identity functor* $I : \mathsf{Set} \to \mathsf{Set}$ *is in FQL;*
2. *for each set* D*, the constant functor* $D : \mathsf{Set} \to \mathsf{Set}$ *is in FQL;*
3. *the product* $X \mapsto \mathsf{T}_1(X) \times \mathsf{T}_2(X)$ *of two FQLs* T_1 *and* T_2 *is in FQL;*
4. *the coproduct* $X \mapsto \mathsf{T}_1(X) + \mathsf{T}_2(X)$ *of two FQLs* T_1 *and* T_2 *is in FQL;*
5. *the composition* $X \mapsto (\mathsf{T}_1 \circ \mathsf{T}_2)(X)$ *of a FQL functor* T_1 *and a functor* T_2 *which has a functorial lax extension, is in FQL.*

3 Coalgebraic Fixpoint Logic and Automata

3.1 Coalgebraic Fixpoint Logic

In this section we show how to define the syntax and semantics of a coalgebraic fixpoint logic, using a quasi-functorial lax extension L of T. For this purpose from now on we fix a functor T with a quasi-functorial lax extension L. Recall that by our convention 2.7, T preserves all inclusions and finite intersections. We also fix a set P of propositional letters and assume that L preserves diagonals.

▶ **Definition 3.1.** Given a functor T, we define for every set X the function

$$\mathit{Base} : \mathsf{T}_\omega X \to \mathcal{P}_\omega X,\ \alpha \mapsto \bigcap \{X' \subseteq X \mid \alpha \in \mathsf{T}X'\}.$$

The point of this notion is that $\mathit{Base}(\alpha) \in \mathcal{P}_\omega X$ is the least set $U \in \mathcal{P}_\omega X$ such that $\alpha \in \mathsf{T}U$.

The language of the coalgebraic fixpoint logic is defined as follows:

▶ **Definition 3.2.** For P as the set of propositional letters, define the language $\mu\mathcal{L}^{\mathsf{T}}_L(\mathsf{P})$ by the following grammar:

$$a ::= p \mid \neg a \mid \bigvee A \mid \nabla \alpha \mid \mu p.a,$$

where $p \in \mathsf{P}$, $A \in \mathcal{P}_\omega(\mu\mathcal{L}^{\mathsf{T}}_L)$ and $\alpha \in \mathsf{T}_\omega(\mu\mathcal{L}^{\mathsf{T}}_L(\mathsf{P}))$. There is a restriction on the formulation of the formulas $\mu p.a$, namely, no occurrence of p in a may be in the scope of an odd number of negations.[1]

[1] For a precise definition of the notions *scope* and *occurrence*, we can inductively define a construction tree of a formula, where the children of a node labeled $\nabla\alpha$ are given by the formulas in $\mathit{Base}(\alpha)$.

▶ **Remark 3.3.** For a given formula $a \in \mu\mathcal{L}_L^{\mathsf{T}}(\mathsf{P})$, $\mathsf{P}_a \subseteq \mathsf{P}$ denotes the set of all propositional letters occurring in a. Observe that for $\mathsf{Q}' \subseteq \mathsf{Q} \subseteq \mathsf{P}$, we have that $\mu\mathcal{L}_L^{\mathsf{T}}(\mathsf{Q}') \subseteq \mu\mathcal{L}_L^{\mathsf{T}}(\mathsf{Q})$. This can be proved by induction on the complexity of formulas in $\mu\mathcal{L}_L^{\mathsf{T}}(\mathsf{Q}')$.

Before we turn to the coalgebraic semantics of this language, there are a number of syntactic definitions to be fixed.

▶ **Definition 3.4.** We will write $b \trianglelefteq a$ if b is a subformula of a. Inductively we define the set $\mathit{Sfor}(a)$ of subformulas of a as follows:

$$\begin{aligned}
\mathit{Sfor}(p) &:= \{p\},\\
\mathit{Sfor}(\neg a) &:= \{\neg a\} \cup \mathit{Sfor}(a),\\
\mathit{Sfor}\left(\bigvee A\right) &:= \{\bigvee A\} \cup \bigcup_{a \in A} \mathit{Sfor}(a),\\
\mathit{Sfor}(\mu p.a) &:= \{\mu p.a\} \cup \mathit{Sfor}(a),\\
\mathit{Sfor}(\nabla\alpha) &:= \{\nabla\alpha\} \cup \bigcup_{a \in Base(\alpha)} \mathit{Sfor}(a)
\end{aligned}$$

The elements of $Base(\alpha)$ will be called the immediate subformulas of $\nabla\alpha$.

▶ **Definition 3.5.** A formula $a \in \mu\mathcal{L}_L^{\mathsf{T}}(\mathsf{P})$ is *guarded* if every subformula $\mu p.b$ of a has the property that all occurrences of p inside b are within the scope of a ∇.

We now introduce the semantics of coalgebraic fixpoint logic. For this purpose we define the notion of a T-model over a set P of propositional letters.

▶ **Definition 3.6.** A T-model $\mathbb{S} = (S, \sigma, V)$ is a T-coalgebra (S, σ) together with a valuation V that is a function $V : \mathsf{P} \to \mathcal{P}S$.

Using the fixed quasi-functorial lax extension L for the functor T we can define the semantics for the language $\mu\mathcal{L}_L^{\mathsf{T}}(\mathsf{P})$ on T-models, by giving the definition of the satisfaction relation $\Vdash_{\mathbb{S}}: S \nrightarrow \mu\mathcal{L}_L^{\mathsf{T}}(\mathsf{P})$ for a T-model $\mathbb{S} = (S, \sigma, V)$.

▶ **Definition 3.7.** Before going to the definition of the satisfaction relation, we need to fix some notation: For $X \subseteq S$, $V[p \mapsto X]$ denotes the valuation that is exactly like V apart from mapping p to X. We also use $[\![a]\!]_{\mathbb{S}}$ for the extension of formula a in a T-model $\mathbb{S}$: $[\![a]\!]_{\mathbb{S}} := \{s \in S \mid s \Vdash_{\mathbb{S}} a\}$.Then $[\![a]\!]_{\mathbb{S}[p \mapsto X]}$ denotes the extension of a considering the valuation $V[p \mapsto X]$, instead of V.
Now we are ready to define the satisfaction relation as follows:

$$\begin{aligned}
s \Vdash_{\mathbb{S}} p &\quad \text{iff} \quad s \in V(p)\\
s \Vdash_{\mathbb{S}} \neg a &\quad \text{iff} \quad \text{not } s \Vdash_{\mathbb{S}} a\\
s \Vdash_{\mathbb{S}} \bigvee A &\quad \text{iff} \quad s \Vdash_{\mathbb{S}} a \text{ for some } a \in A\\
s \Vdash_{\mathbb{S}} \nabla\alpha &\quad \text{iff} \quad (\sigma(s), \alpha) \in L \Vdash_{\mathbb{S}}\\
s \Vdash_{\mathbb{S}} \mu p.a &\quad \text{iff} \quad s \in \bigcap\{X \subseteq S \mid [\![a]\!]_{\mathbb{S}[p \mapsto X]} \subseteq X\}.
\end{aligned}$$

▶ **Remark 3.8.** The clauses in Definition 3.7 are not stated in a correct recursive way. In the recursive clause for the ∇ modality we make use of the unrestricted satisfaction relation $\Vdash_{\mathbb{S}}$ that has yet to be defined. We can only suppose that $\Vdash_{\mathbb{S}}|_{S \times Base(\alpha)}$ is already defined. The actual recursive definition is that $s \Vdash_{\mathbb{S}} \nabla\alpha$ iff $(\sigma(s), \alpha) \in L(\Vdash_{\mathbb{S}}|_{S \times Base(\alpha)})$. To see why this is equal to the clause given above, see [14, Proposition 6].

▶ **Remark 3.9.** Given a valuation $V : \mathsf{P} \to \mathcal{P}S$, one can think of it as a coloring $\gamma_V : S \to \mathcal{P}\mathsf{P}$ given by: $\gamma_V(s) := \{p \in \mathsf{P} \mid s \in V(p)\}$. So following Definition 2.6, a T-model $\mathbb{S} = (S, \sigma, V)$ can also be seen as a $\mathcal{P}(\mathsf{P})$-colored T-coalgebra denoted as $\hat{\mathbb{S}} = (S, \sigma, \gamma_V)$.

▶ **Definition 3.10.** The *projection* of a $\mathcal{P}(\mathsf{P})$-colored T-coalgebra $\mathbb{S} = (S, \sigma, \gamma)$ to a set $\mathsf{Q} \subseteq \mathsf{P}$ is the $\mathcal{P}(\mathsf{Q})$-colored T-coalgebra $\mathbb{S}^{\mathsf{Q}} = (S, \sigma, \gamma^{\mathsf{Q}})$ where $\gamma^{\mathsf{Q}} : S \to \mathcal{P}\mathsf{Q}$, $s \mapsto \gamma(s) \cap \mathsf{Q}$.

▶ **Definition 3.11.** Given a set $\mathsf{Q} \subseteq \mathsf{P}$, an L_{Q}-*bisimulation* between two T-models $\mathbb{S}$ and $\mathbb{Y}$ is defined to be an $L_{\mathcal{P}(\mathsf{Q})}$-bisimulation between $\mathcal{P}(\mathsf{Q})$-colored T-colagebras $\hat{\mathbb{S}}^{\mathsf{Q}}$ and $\hat{\mathbb{Y}}^{\mathsf{Q}}$, which are given by Remark 3.9 and Definition 3.10. More precisely, a relation $R : S \nrightarrow Y$ is an L_{Q}-bisimulation between T-models $\mathbb{S} = (S, \sigma, V_S)$ and $\mathbb{Y} = (Y, \lambda, V_Y)$ if and only if R is an L-bisimulation between T-coalgebras $\mathbb{S} = (S, \sigma)$ and $\mathbb{Y} = (Y, \lambda)$ and R preserves the truth of all propositional letters in Q, that is, for all $(s, y) \in R$ and $p \in \mathsf{Q}$, $s \in V_S(p)$ iff $y \in V_Y(p)$.

From this definition, it is easy to see that for any $\mathsf{Q}' \subseteq \mathsf{Q}$, if a relation R is an L_{Q}-bisimulation between T-models $\mathbb{S}$ and $\mathbb{Y}$, then it is also an $L_{\mathsf{Q}'}$-bisimulation between them.

▶ **Definition 3.12.** Given a propositional letter $p \in \mathsf{P}$, a relation $R : S \nrightarrow S'$ is an *up-to-p* L_{P}-*bisimulation* between two T-models $\mathbb{S} = (S, \sigma, V)$ and $\mathbb{S}' = (S', \sigma', V')$, if it is an $L_{\mathsf{P}\setminus\{p\}}$-bisimulation between T-models $\mathbb{S}$ and $\mathbb{S}'$. We write $s \leftrightarrows_p^L s'$ if s and s' are up-to-p L_{P}-bisimilar, that is where we disregard the proposition letter p.

Now we are going to look at the expressive power of $\mu\mathcal{L}_L^{\mathsf{T}}(\mathsf{P})$ with respect to states in T-models. For this, we start with a definition.

▶ **Definition 3.13.** Two states s in T-model $\mathbb{S} = (S, \sigma, V)$ and s' in T-model $\mathbb{S}' = (S', \sigma', V')$ are called *equivalent* for formulas in $\mu\mathcal{L}_L^{\mathsf{T}}(\mathsf{P})$ if $s \Vdash_{\mathbb{S}} a$ iff $s' \Vdash_{\mathbb{S}'} a$, for all $a \in \mu\mathcal{L}_L^{\mathsf{T}}(\mathsf{P})$.

An important property of our coalgebraic fixpoint logic is that truth is bisimulation invariant. This fact is given by the following proposition.

▶ **Proposition 3.14.** *Given a state s in a T-model $\mathbb{S} = (S, \sigma, V)$ and a state s' in a T-model $\mathbb{S}' = (S', \sigma', V')$, if s and s' are L_{P}-bisimilar then s and s' are equivalent for formulas in $\mu\mathcal{L}_L^T(P)$.*

For the proof of this proposition we refer to [22, Proposition 5.14], [13, Proposition 4.11] and the fact that lax extensions are monotone.

Now we are ready to state the last semantic result we will need throughout this paper.

▶ **Proposition 3.15.** *Each formula in $\mu\mathcal{L}_L^{\mathsf{T}}(\mathsf{P})$ can be transformed into an equivalent guarded formula in $\mu\mathcal{L}_L^{\mathsf{T}}(\mathsf{P})$.*

It can be proved by induction on the complexity of formulas, see [22, Proposition 5.15]

▶ **Convention 3.16.** Throughout this paper we always assume $\mu\mathcal{L}_L^{\mathsf{T}}(\mathsf{P})$-formulas to be guarded.

3.2 Coalgebraic Automata

Coalgebraic automata are supposed to operate on pointed coalgebras. Basically, the idea is that an initialized T-automaton will either *accept* or *reject* a given pointed T-coalgebra. In the following section, we will recall the basic definitions from coalgebraic automata theory.

▶ **Definition 3.17.** Given a functor $\mathsf{T} : \mathsf{Set} \to \mathsf{Set}$. A (non-deterministic) T-automaton over a color set C is a triple $\mathbb{A} = (A, \Delta, \Omega)$, with A some finite set (of states), $\Delta : A \times C \to \mathcal{P}\mathsf{T}A$ the *transition map* and $\Omega : A \to \omega$ a *parity map*. An *initialized* version of $\mathbb{A}$ is a pair $(\mathbb{A}, a)$ consisting of an automaton $\mathbb{A}$ together with an element $a \in A$, which is the *initial* state.

The acceptance condition for T-automaton is formulated in terms of a parity game[12]. The acceptance game $\mathcal{G}(\mathbb{S}, \mathbb{A})$ between initialized automaton $(\mathbb{A}, a_I)$ and a pointed coalgebra $(\mathbb{S}, s_I)$ is given by the Table 1. The game is played by two players: Éloise ($\exists$) and Abélard ($\forall$). A *match* of the game is a (finite or infinite) sequence of positions which is given by the two players moving from one position to another according to the rules of Table 1. Let us now give the formal definition of acceptance game.

▶ **Definition 3.18.** Let $(\mathbb{A}, a_I)$ be an initialized T-automaton over the color set C. Furthermore let $(\mathbb{S}, s_I) = (S, \sigma, \gamma, s_I)$ be a pointed C-colored T-coalgebra. Then the *acceptance game* $\mathcal{G}(\mathbb{S}, \mathbb{A})$ is given by the following table:

Table 1 Acceptance game for T-automaton.

Position	Player	Admissible moves	Parity
$(s, a) \in S \times A$	$\exists$	$(\sigma(s), \phi)$ *s.t.* $\phi \in \Delta(a, \gamma(s))$	$\Omega(a)$
$(\sigma(s), \phi) \in \mathsf{T}S \times \mathsf{T}A$	$\exists$	$\{Z : S \nrightarrow A \mid (\sigma(s), \phi) \in LZ$	0
$Z \subseteq S \times A$	$\forall$	Z	0

Positions of the form $(s, a) \in S \times A$ will be called *basic positions* of the game. A partial play of the game of the form $(s, a)(\sigma(s), \phi)Z(t, b)$ (with $(s, a) \in S \times A$, $(\sigma(s), \phi) \in \mathsf{T}S \times \mathsf{T}A$, $Z : S \nrightarrow A$ and $(t, b) \in Z$) will be called a *round* of the play. For the winning conditions, recall that finite matches are lost by the player who gets stuck. For infinite matches, consider an arbitrary such match:

$$\rho = (s_0, a_0)(\sigma(s_0), \phi_0)Z_0(s_1, a_1)(\sigma(s_1), \phi_1)Z_1(s_2, a_2)...$$

Clearly, ρ induces an infinite sequence of basic positions $(s_0, a_0)(s_1, a_1)(s_2, a_2)...$ and, thus, an infinite sequence of states in A: $\rho \upharpoonright_A := a_0 a_1 a_2 ...$ Now $\exists$ is the winner of the match ρ if the maximum priority occurring infinitely often on $\rho \mid_A$ is even. Otherwise $\forall$ wins ρ. A *positional* or *history free strategy* for $\exists$ is a pair of functions $(\Phi : S \times A \to \mathsf{T}A, Z : S \times A \to \mathcal{P}(S \times A))$. Such a strategy is *legitimate* if at any position, it maps the position to an admissible next position. A legitimate strategy is *winning* for $\exists$ from a position in the game, if it guarantees $\exists$ to win any match starting from that position, no matter how $\forall$ plays. A position starting from which $\exists$ has a winning strategy is called a *winning position* for $\exists$. The set of all winning positions for $\exists$ in $\mathcal{G}(\mathbb{S}, \mathbb{A})$ is denoted by $\mathrm{Win}_\exists(\mathbb{S}, \mathbb{A})$ or shortly by $\mathrm{Win}_\exists$. A history-free strategy (Φ, Z) initialized at $(s_I, b) \in S \times A$ is called *scattered* if the relation

$$\{(s_I, b)\} \cup \bigcup \{Z_{s,a} \subseteq S \times A \mid (s, a) \in \mathrm{Win}_\exists\},$$

with $Z_{s,a}$ the value of Z on (s, a), is functional. Finally we say that initialized T-automaton $(\mathbb{A}, a_I)$ *accepts* $(\mathbb{S}, s_I)$ if $\exists$ has a winning strategy in the game $\mathcal{G}(\mathbb{A}, \mathbb{S})$ initialized at position (s_I, a_I). If $\exists$ has a scattered winning strategy starting from (s_I, a_I), we will say $(\mathbb{A}, a_I)$ *strongly accepts* $(\mathbb{S}, s_I)$.

▶ **Definition 3.19.** For every initialized T-automaton $(\mathbb{A}, a_I)$ over some color set C, $L(\mathbb{A}, a_I)$, the *recognizable language* of $(\mathbb{A}, a_I)$, is the class of all pointed C-colored T-coalgebras that are accepted by $(\mathbb{A}, a_I)$. We call two initialized T-automata $(\mathbb{A}, a_I)$ and $(\mathbb{A}', a'_I)$ over set C *equivalent* iff $L(\mathbb{A}, a_I) = L(\mathbb{A}', a'_I)$.

3.3 Logic and Automata

There is a routine construction of an equivalent initialized T-automaton $(\mathbb{A}, a_I)$ from a $\mu\mathcal{L}_L^{\mathsf{T}}(\mathsf{P})$-formula, and vice versa. Given the finitary nature of our automata, this construction requires the functor T to preserve finite sets.

▶ **Proposition 3.20.** *Let T be a functor that preserves finite sets. There exists an effective procedure to transform a formula $b \in \mu\mathcal{L}_L^{T}(P)$ to an initialized T-automaton $(\mathbb{A}_b, a_b)$ over the set $C = \mathcal{P}(P)$ such that for every C-colored T-coalgebra $(\mathbb{S}, s)$: $(\mathbb{S}, s) \Vdash_{\mathbb{S}} b$ iff $(\mathbb{A}_b, a_b)$ accepts $(\mathbb{S}, s)$.*

Proof sketch. Our construction proceeds along the exact same lines as the construction of an initialized *alternating* T-automaton from a given formula [22, Theorem 2] and transforming it to a non-deterministic T-automaton [12, Theorem 1] in the case of a functor that preserves weak pullbacks, and uses some facts from [13] to the fact that also in our case, the nabla operator has certain desirable properties. The construction proceeds in the following stages:

(0) First of all we generalize our notion of a T-automaton (which is non-deterministic in nature) to that of an alternating T-automaton, which has a transition map of the type $\Delta : A \to \mathcal{P}\mathcal{P}\mathsf{T}A$. For these automata, acceptance is defined as for the alternating automata in [12] and [22].

(1) Using routine methods [22, Theorem 2] we can inductively show that every formula in our language can be effectively transformed into an equivalent alternating automaton. For the case of negation we use the method of [11] together with the fact that the dual of our nabla operator can be expressed using disjunctions and the nabla operator itself [13, Theorem 4.14].

(2) What is still missing is a *simulation theorem* stating that every alternating automaton can be replaced with an equivalent non-deterministic one. This result is in fact also a more or less routine result [2, section 9.6], since we can use the fact that our nabla also satisfies a certain modal distributive law, stating that the conjunction of nablas over some formulas is equivalent to a disjunction of nablas over some conjunctions of these formulas [13, Proposition 4.17].

(3) Combining (1) and (2) we see that every formula in our coalgebraic fixpoint language is equivalent to one of our automata indeed. ◀

▶ **Proposition 3.21.** *There exists an effective procedure transforming an initialized T-automaton $(\mathbb{A}, a_I)$ to an equivalent $\mu\mathcal{L}_L^{T}(P)$-formula $a_{\mathbb{A}}$.*

This result is rather standard, see for instance [22, Theorem 3].

4 Automata are Closed under Projection

This section is devoted to the proof of our main technical result i.e., closure under projection.

▶ **Definition 4.1.** Let $\mathbb{A} = (A, \Delta, \Omega)$ be a T-automaton over color set C. We call a state $a \in A$ a *true state* of $\mathbb{A}$ if $\Omega(a)$ is even and $\Delta(a, c) = \mathsf{T}(\{a\})$. We will standardly use the notation $a_\top$ to refer to a true state. Given $(a, c) \in A \times C$ we call $\phi \in \Delta(a, c)$ a *satisfiable element* of $\mathbb{A}$ if there is a witnessing T-coalgebra $(\mathbb{Q}_\phi, \rho, \gamma_Q)$, $\tau \in \mathsf{T}Q$ and a relation $Z_\phi : Q \nrightarrow A$ such that $(\tau, \phi) \in LZ_\phi$ and $Z_\phi \subseteq \mathrm{Win}_\exists(\mathbb{Q}, \mathbb{A})$. Finally we call a T-automaton $\mathbb{A}$ *totally satisfiable* whenever for all $(a, c) \in A \times C$ and $\phi \in \Delta(a, c)$, ϕ is satisfiable.

The following proposition states that without loss of generality we can always assume that an initialized T-automaton $(\mathbb{A}, a_I)$ is totally satisfiable and has a true state. Furthermore, we may always assume that there exists a witnessing T-coalgebra $\mathbb{Q}$ that works for all $(a, c) \in A \times C$ and $\phi \in \Delta(a, c)$.

▶ **Proposition 4.2.** *For any initialized* T*-automaton* $(\mathbb{A}, a_I)$ *over a color set* C *we have that:*

1. *There is an equivalent initialized* T*-automaton* $(\mathbb{A}', a_I)$ *such that* $\mathbb{A}'$ *has a true state.*
2. *There exists a totally satisfiable initialized* T*-automaton* $(\mathbb{A}', a_I')$ *which is equivalent to* $(\mathbb{A}, a_I)$*.*
3. *If* $(\mathbb{A}, a_I)$ *is totally satisfiable, then there is a* C*-colored witnessing coalgebra* $\mathbb{Q} = (Q, \rho, \gamma_Q)$ *and a relation* $Y : Q \nrightarrow A$ *with* $Y \subseteq Win_\exists(\mathbb{Q}, \mathbb{A})$ *such that for all* $(a, c) \in A \times C$ *and* $\phi \in \Delta(a, c)$*, there is a* $\tau \in \mathsf{T}Q$ *such that* $(\tau, \phi) \in LY$*.*

Now we will state the main technical result of this paper. Theorem 4.3 is a generalization of [12, Proposition 5.9], where the same result is proved for the weak-pullback preserving functors. In the following theorem we will generalize the proposition to the class of all functors with a quasi-functorial lax extension that preserves diagonals. The proof strategy is the same as in [12], but the construction here is more involved.

▶ **Theorem 4.3 (Closure under projection).** *Given an initialized* T*-automaton* $(\mathbb{A}, a_I)$ *over a color set* $\mathcal{P}(\mathsf{P})$ *and an element* $p \in \mathsf{P}$*, then there exists an initialized* T*-automaton* $(\exists_p.\mathbb{A}, a_I)$ *over the color set* $\mathcal{P}(\mathsf{P} \setminus \{p\})$ *such that:*

$$(\mathbb{S}, s_I) \in L(\exists_p.\mathbb{A}, a_I) \textit{ iff } (\overline{\mathbb{S}}, \overline{s}_I) \in L(\mathbb{A}, a_I) \textit{ for some } (\overline{\mathbb{S}}, \overline{s}_I) \textit{ with } \mathbb{S}, s_I \leftrightarrows_p^L \overline{\mathbb{S}}, \overline{s}_I. \qquad (2)$$

Proof. Given $(\mathbb{A}, a_I)$ over color a set $\mathcal{P}\mathsf{P}$, we define the initialized T-automaton $(\exists_p.\mathbb{A}, a_I)$ over the color set $\mathcal{P}(\mathsf{P} \setminus \{p\})$ as the automaton $(\exists_p.\mathbb{A}, a_I) := (A, \Delta_p, \Omega, a_I)$, where

$$\Delta_p : A \times \mathcal{P}(\mathsf{P} \setminus \{p\}) \to \mathcal{P}\mathsf{T}A, \; (a, c) \mapsto \Delta(a, c) \cup \Delta(a, c \cup \{p\}).$$

We need to show that $(\exists_p.\mathbb{A}, a_I)$ satisfies (2). The right-to-left direction of (2) is straightforward, since all legitimate moves of $\exists$ in the game $\mathcal{G}(\mathbb{A}, \mathbb{S}')$ are also legitimate in $\mathcal{G}(\exists_p.\mathbb{A}, \mathbb{S}'_p)$.

To show the left-to-right direction of (2) assume that $(\exists_p.\mathbb{A}, a_I)$ accepts the $\mathcal{P}(\mathsf{P} \setminus \{p\})$-colored T-coalgebra $(\mathbb{S}, s_I) = (S, \sigma, \gamma, s_I)$. We need to define a $\mathcal{P}\mathsf{P}$-colored coalgebra $(\overline{\mathbb{S}}, \overline{s}_I) = (\overline{S}, \overline{\sigma}, \overline{\gamma}, \overline{s}_I) \in L(\mathbb{A}, a)$ that is up-to-p bisimilar to $(\mathbb{S}, s_I)$.

By Proposition 4.2 we can assume that $(\exists_p.\mathbb{A}, a_I)$ has a true state and is totally satisfiable, which entails that there is a $\mathcal{P}\mathsf{P}$-colored coalgebra $\mathbb{Q} = (Q, \rho, \gamma_Q)$ and a relation $Y : Q \nrightarrow A$ with $Y \subseteq \mathrm{Win}_\exists(\mathbb{Q}, \exists_p.\mathbb{A})$ such that for all $(a, c) \in A \times C$ and $\phi \in \Delta(a, c)$ there is a $\tau \in \mathsf{T}Q$ with $(\tau, \phi) \in LY$.

The carrier of $(\overline{\mathbb{S}}, \overline{s}_I)$ is the set $\overline{S} := (S \times A) \uplus Q$. To define the coalgebra structure $\overline{\sigma} : \overline{S} \to \mathsf{T}\overline{S}$ we distinguish the following cases:

(1) If $q \in Q$, define $\overline{\sigma}(q) := \rho(q)$.

(2) If $(s, a) \in S \times A$ and $(s, a) \notin \mathrm{Win}_\exists(\mathbb{S}, \exists_p.\mathbb{A})$, define $\overline{\sigma}(s, a) := \mathsf{T}\kappa_a(\sigma(s))$, where $\kappa_a : S \to S \times A$, $s \mapsto (s, a)$.

(3) In the case where $(s, a) \in S \times A$ and $(s, a) \in \mathrm{Win}_\exists(\mathbb{S}, \exists_p.\mathbb{A})$ we define $\overline{\sigma}(s, a)$ as follows: From the winning strategy that witnesses $(s, a) \in \mathrm{Win}_\exists(\mathbb{S}, \exists_p.\mathbb{A})$ we obtain a $\phi_{s,a} \in \Delta(a, \gamma(s))$ and a relation $Z_{s,a} : S \nrightarrow A$ such that $Z_{s,a} \subseteq \mathrm{Win}_\exists(\mathbb{S}, \exists_p.\mathbb{A})$ and $(\sigma(s), \phi_{s,a}) \in LZ_{s,a}$. Because $(\mathbb{A}, a_I)$ contains a true state we can assume without loss of generality that $Z_{s,a}$ is full on S. We can write $Z_{s,a} = \pi_1^\circ ; \pi_2$ where $\pi_1 : Z_{s,a} \to S$ and $\pi_2 : Z_{s,a} \to A$ are the projections of $Z_{s,a}$. These projections can be seen as relations with domain $(S \times A) \uplus Q$ for which it then follows that $Z_{s,a} \subseteq \pi_1^\circ ; (\pi_s \uplus Y)$. Because L is a lax extension one obtains that $LZ_{s,a} \subseteq L(\pi_1^\circ ; (\pi_s \uplus Y))$ and hence $(\sigma(s), \phi_{s,a}) \in L(\pi_1^\circ ; (\pi_s \uplus Y))$. It also holds that $\sigma(s) \in Dom(L(\pi_1^\circ))$ because $Z_{s,a}$ is full on S, so π_1° is full on S, and hence by Proposition 2.13 (2) $L(\pi_1^\circ)$ is full on $\mathsf{T}S$.

Moreover $\phi_{s,a} \in Rng(L(\pi_2 \uplus Y))$ because $\phi_{s,a} \in Rng(LY)$ by the properties of Y and $LY \subseteq L(L(\pi_2 \uplus Y)$.
With the quasi-functoriality of L it now follows that $(\sigma(s), \phi_{s,a}) \in L(\pi_1^\circ); L(\pi_s \uplus Y)$. Hence it is possible to choose $\overline{\sigma}(s,a) \in \mathsf{T}((S \times A) \uplus Q)$ such that

$$(\sigma(s), \overline{\sigma}(s,a)) \in L(\pi_1^\circ) \text{ and } (\overline{\sigma}(s,a), \phi_{s,a}) \in L(\pi_2 \uplus Y).$$

To complete the definition of the $\mathcal{P}\mathsf{P}$-colored pointed coalgebra $(\overline{\mathbb{S}}, \overline{s_I})$ we set $\overline{s_I} := (s_I, a_I)$ and define the coloring $\overline{\gamma} : \overline{S} \to \mathcal{P}(\mathsf{P})$ by distinguishing the following cases:

(1) If $q \in Q$, define $\overline{\gamma}(q) := \gamma_Q(q)$.

(2) If $(s,a) \in S \times A$ and $(s,a) \notin \mathrm{Win}_\exists(\mathbb{S}, \exists_p.\mathbb{A})$, define $\overline{\gamma}(s,a) := \gamma(s)$.

(3) If $(s,a) \in S \times A$ and $(s,a) \in \mathrm{Win}_\exists(\mathbb{S}, \exists_p.\mathbb{A})$ we define $\overline{\gamma}(s,a)$ by considering the choice of $\exists$ at (s,a). Since (s,a) is a winning position for $\exists$, she picks an element $\phi_{s,a} \in \Delta_p(a, \gamma(s))$. The function Δ_p is defined such that $\Delta_p(a, \gamma(s)) = \Delta(a, \gamma(s)) \cup \Delta(a, \gamma(s) \cup \{p\})$. We set

$$\overline{\gamma}(s,a) := \begin{cases} \gamma(s) \cup \{p\} & \text{if } \phi_{s,a} \in \Delta(a, \gamma(s) \cup \{p\}), \\ \gamma(s) & \text{otherwise.} \end{cases}$$

We need to show that $\mathbb{S}, s_I \leftrightarrows_p^L \overline{\mathbb{S}}, (s_I, a_I)$ and that $((s_I, a_I), a_I) \in \mathrm{Win}_\exists(\overline{\mathbb{S}}, \mathbb{A})$.

▶ **Claim (1).** $\mathbb{S}, s_I \leftrightarrows_p^L \overline{\mathbb{S}}, (s_I, a_I)$.

Proof of claim (1). We show that graph of the projection $\pi_S : S \times A \to S$ seen as a relation between $\overline{S}$ and S is an up-to-p bisimulation between $\overline{\mathbb{S}}, \overline{s_I}$ and $\mathbb{S}, s_I$. We need to prove that

$$(\overline{\sigma}(s,a), \sigma(s)) \in L\pi_S \text{ and } \overline{\gamma}(s,a) \setminus \{p\} = \gamma(s) \text{ whenever } ((s,a),s) \in \pi_1.$$

That $\overline{\gamma}(s,a) \setminus \{p\} = \gamma(s)$ follows directly from the definition of $\overline{\gamma}$. For $(\overline{\sigma}(s,a), \sigma(s)) \in L\pi_S$ we distinguish two cases:

(i) If $(s,a) \in S \times A$ and $(s,a) \notin \mathrm{Win}_\exists(\mathbb{S}, \exists_p.\mathbb{A})$ then the statement holds because by definition $\overline{\sigma}(s,a) = \mathsf{T}\kappa_a(\sigma(s))$ and since L is a lax extensions and $\kappa_a \subseteq \pi_S$ we have that

$$(\mathsf{T}\kappa_a(\sigma(s)), \sigma(s)) \in \mathsf{T}\kappa_a = L\kappa_a \subseteq L\pi_S$$

(ii) If $(s,a) \in S \times A$ and $(s,a) \in \mathrm{Win}_\exists(\mathbb{S}, \exists_p.\mathbb{A})$ then we get by the definition of $\overline{\sigma}$ that $(\overline{\sigma}(s,a), \sigma(s)) \in L\pi_1$. It follows that $(\overline{\sigma}(s,a), \sigma(s)) \in L\pi_S$ because L is a lax extensions and $\pi_1 \subseteq \pi_S$ since $\pi_1 : Z_{s,a} \to S$ is the projection of the relation $Z_{s,a} \subseteq S \times A$. ◀

▶ **Claim (2).** $((s_I, a_I), a_I) \in \mathrm{Win}_\exists(\overline{\mathbb{S}}, \mathbb{A})$.

Proof of claim (2). Let (Ψ, Y') be a strategy for $\exists$ witnessing that $Y \subseteq \mathrm{Win}_\exists(\mathbb{Q}, \exists_p.\mathbb{A})$. Define $\exists$'s strategy in $\mathcal{G}(\overline{\mathbb{S}}, \mathbb{A})$ as follows:

$$\begin{aligned} \overline{\Phi} : \overline{S} \times A &\to \mathsf{T}A & \overline{Z} : \overline{S} \times A &\to \mathcal{P}(\overline{S} \times A) \\ ((s,b),a) &\mapsto \phi_{s,a} & ((s,b),a) &\mapsto \pi_2 \uplus Y \\ (q,a) &\mapsto \psi_{q,a} & (q,a) &\mapsto Y'_{q,a} \end{aligned}$$

where π_2 is the projection of $Z_{s,a}$, if $(s,a) \in \mathrm{Win}_\exists(\mathbb{S}, \exists_p.\mathbb{A})$, and arbitrary otherwise. ◀

▶ **Claim (2a).** For the following types of positions in $\mathcal{G}(\overline{\mathbb{S}}, \mathbb{A})$, the given strategy $(\overline{\Phi}, \overline{Z})$ provides legitimate moves for $\exists$:

(i) $(q,a) \in \overline{S} \times A$ and $(q,a) \in \mathrm{Win}_\exists(\mathbb{Q}, \exists p.\mathbb{A})$,

(ii) $((s,a),a) \in \overline{S} \times A$ and $(s,a) \in \mathrm{Win}_\exists(\mathbb{S}, \exists_p.\mathbb{A})$

Proof of Claim (2a).

(i) This is clear since $\overline{\sigma}(q) = \rho(q)$ and $\exists$ plays her winning strategy in $\mathcal{G}(\mathbb{Q}, \exists p.\mathbb{A})$.

(ii) By the definition of $\overline{\gamma}$ we have that $\overline{\phi}_{s,a} = \phi_{s,a} \in \Delta(a, \overline{\gamma}(s,a))$. Also $(\overline{\sigma}(s,a), \overline{\phi}_{s,a}) \in L\overline{Z}_{s,a}$ because $(\overline{\sigma}(s,a), \phi_{s,a}) \in L(\pi_2 \uplus Y)$.

◀

▶ **Claim (2b).** $(\overline{\Phi}, \overline{Z})$ guarantees $\exists$ to win any match of $\mathcal{G}(\overline{\mathbb{S}}, \mathbb{A})$ starting from $((s_I, a_I), a_I)$.

Proof of Claim (2b). To check that $(\overline{\Phi}, \overline{Z})$ is winning it suffices to distinguish the following two kinds of matches:

(i) At some stage $\forall$ chooses an element $(q, a) \in Y$. From this moment on, there is no way to go through the states of $\mathbb{S}$ and since $Y \subseteq \mathrm{Win}_{\exists}(\mathbb{Q}, \exists_p.\mathbb{A})$, $\exists$ plays her winning strategy in $\mathcal{G}(\mathbb{Q}, \mathbb{A})$ and wins the match.

(ii) $\forall$ never picks an element of the form (q, a). In this case any $(\overline{\Phi}, \overline{Z})$-conforming match is of the form

$$((s_I, a_I), a_I)((s_1, a_1), a_1)((s_2, a_2), a_2)\ldots$$

This match corresponds to the (Φ, Z)-conforming match

$$(s_I, a_I)(s_1, a_1)(s_2, a_2)\ldots$$

in the game $\mathcal{G}(\mathbb{S}, \mathbb{A})$. Since we assumed (Φ, Z) to be a winning strategy for $\exists$, $(\overline{\Phi}, \overline{Z})$ is also a winning strategy for her.

◀

This finishes the proof of Theorem 4.3. ◀

5 Uniform Interpolation for $\mu\mathcal{L}_L^{\mathsf{T}}$

In the following section we will prove the main theorem of this paper, viz., uniform interpolation for $\mu\mathcal{L}_L^{\mathsf{T}}$. Our proof follows and generalizes the proof in [20] which shows a similar result for monotone modal logic (without fixpoints). We first need some auxiliary definitions.

▶ **Definition 5.1.** Define the relation of logical consequence $\models$: $\mu\mathcal{L}_L^{\mathsf{T}}(\mathsf{P}) \nrightarrow \mu\mathcal{L}_L^{\mathsf{T}}(\mathsf{P})$ by $a \models a'$ if and only if $s \Vdash_{\mathbb{S}} a$ implies $s \Vdash_{\mathbb{S}} a'$ for all states s in any T-model $\mathbb{S}$.

▶ **Definition 5.2.** Given a formula $a \in \mu\mathcal{L}_L^{\mathsf{T}}(\mathsf{P})$, we let P_a denote the (obviously defined) set of proposition letters occurring in a.

Our main result can now be formulated as follows.

▶ **Theorem 5.3** (Uniform Interpolation)**.** *Let* T *be a set functor that preserves finite sets, and let* L *be a quasi-functorial lax extension for* T*. For any formula* $a \in \mu\mathcal{L}_L^{T}(\mathsf{P})$ *and any set* $\mathsf{Q} \subseteq \mathsf{P}_a$ *of propositional letters, there is a formula* $a_{\mathsf{Q}} \in \mu\mathcal{L}_L^{T}(\mathsf{Q})$*, effectively constructable from* a*, such that for every formula* $b \in \mu\mathcal{L}_L^{T}(\mathsf{P})$ *with* $\mathsf{P}_a \cap \mathsf{P}_b \subseteq \mathsf{Q}$*, we have that*

$$a \models b \quad \textit{iff} \quad a_{\mathsf{Q}} \models b.$$

If a *is fixpoint-free, then so is* a_{Q}*.*

As mentioned in the introduction, our proof is based on the definability of the bisimulation quantifier in our language.

▶ **Proposition 5.4.** *Given any proposition letter p, there is a map $\exists p : \mu\mathcal{L}^{\mathsf{T}}_{L}(\mathsf{P}) \longrightarrow \mu\mathcal{L}^{\mathsf{T}}_{L}(\mathsf{P})$ such that $\mathsf{P}_{\exists p.b} = \mathsf{P}_b \setminus \{p\}$ and*

$$\mathbb{S}, s \Vdash \exists p.b \quad \textit{iff} \quad \mathbb{S}', s' \Vdash b, \ \textit{for some } \mathbb{S}', s' \textit{ with } \mathbb{S}, s \mathrel{\underline{\leftrightarrow}}^{L}_{p} \mathbb{S}', s'. \tag{3}$$

for any formula $b \in \mu\mathcal{L}^{\mathsf{T}}_{L}(\mathsf{P})$.

Proof. Take a formula $b \in \mu\mathcal{L}^{\mathsf{T}}_{L}(\mathsf{P})$. By Proposition 3.20 we can transform it to an equivalent initialized T-automaton $(\mathbb{A}_b, a_b)$. From Theorem 4.3 we have an initialized T-automaton $(\exists_p.\mathbb{A}_b, a_b)$ such that:

$(\exists_p.\mathbb{A}_b, a_b)$ accepts $(\mathbb{S}, s)$ iff $(\mathbb{A}_b, a_b)$ accepts $(\mathbb{S}', s')$ for some $(\mathbb{S}', s')$ with $\mathbb{S}, s \mathrel{\underline{\leftrightarrow}}^{L}_{p} \mathbb{S}', s'$.

Now by Proposition 3.21 we can transform the initialized T-automaton $(\exists_p.\mathbb{A}_b, a_b)$ to an equivalent formula $a_{(\exists_p.\mathbb{A}_b)}$ and put $\exists p.b := a_{(\exists_p.\mathbb{A}_b)}$. It is easy to show that:

$$\mathbb{S}, s \Vdash a_{(\exists_p.\mathbb{A}_b)} \quad \text{iff} \quad \mathbb{S}', s' \Vdash b, \ \text{for some } \mathbb{S}', s' \text{ with } \mathbb{S}, s \mathrel{\underline{\leftrightarrow}}^{L}_{p} \mathbb{S}', s'.$$

We leave it for the reader to verify that $\mathsf{P}_{\exists p.b} = \mathsf{P}_b \setminus \{p\}$. ◀

Now we are ready to prove the uniform interpolation theorem:

Proof of Theorem 5.3. Let $p_0, p_1, ..., p_{n-1}$ enumerate the proposition letters in $\mathsf{P}_a \setminus \mathsf{Q}$, and set

$$a_{\mathsf{Q}} := \exists p_0 \exists p_1 \ldots \exists p_{n-1}.a.$$

It is not difficult to verify that a_{Q} is fixpoint-free if a is so.

In order to check that $a \models b$ iff $a_{\mathsf{Q}} \models b$, first assume that $a \models b$. To prove that $a_{\mathsf{Q}} \models b$ take a pointed T-model $(\mathbb{S}_0, s_0)$ with $s_0 \Vdash_{\mathbb{S}_0} a_{\mathsf{Q}}$. By the semantics of the bisimulation quantifiers we get states s_i in T-models $\mathbb{S}_i$ for $i = 1, 2, \ldots, n$ such that $s_i \mathrel{\underline{\leftrightarrow}}_{p_i} s_{i+1}$ for $i = 0, ..., n$ and $s_n \Vdash_{\mathbb{S}_n} a$. From the latter fact it follows that $s_n \Vdash_{\mathbb{S}_n} b$ since we have assumed $a \models b$. Because each of the witnessing up-to-p_i L_{P}-bisimulations for $i = 0, 1, \ldots, n-1$ is also an $L_{\mathsf{P}\setminus\{p_0,p_1,\ldots,p_{n-1}\}}$-bisimulation, we can compose them and obtain an $L_{\mathsf{P}\setminus\{p_0,p_1,\ldots,p_{n-1}\}}$-bisimulation between s_0 and s_n. Since $\mathsf{P}_b \subseteq \mathsf{P} \setminus \{p_0, p_1, \ldots, p_{n-1}\}$ we get $s_0 \Vdash_{\mathbb{S}_0} b$.

For the other direction, we show that $a \models a_{\mathsf{Q}}$, then $a \models b$ follows by transitivity from $a_{\mathsf{Q}} \models b$. Take any state s in a T-model $\mathbb{S} = (S, \sigma, V)$ with $s \Vdash_{\mathbb{S}} a$. Then $s \Vdash_{\mathbb{S}} a_{\mathsf{Q}}$ because s is up-to-p L_{P}-bisimilar to itself for any $p \in \mathsf{P}$, since the identity on S is an L_{P}-bisimulation. ◀

6 Conclusions and Future Work

In this paper we showed that the coalgebraic fixpoint logic for functors with a quasi-functorial lax extension that preserves diagonals, enjoys uniform interpolation. This suggests to further study the class of functors possessing such a relation lifting. For instance one might try to characterize this class of functors in categorical terms and investigate how the cover modality of such a relation lifting relates to modalities arising from predicate liftings [14].

References

1 Jiří Adámek and Věra Trnková. *Automata and algebras in categories*, volume 37 of *Mathematics and its Applications (East European Series)*. Kluwer Academic Publishers Group, Dordrecht, 1990.

2 A. Arnold and D. Niwiński. *Rudiments of μ-calculus*, volume 146 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Co., Amsterdam, 2001.

3 J. Richard Büchi. Weak second-order arithmetic and finite automata. *Z. Math. Logik Grundlagen Math.*, 6:66–92, 1960.

4 William Craig. Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. *J. Symb. Logic*, 22:269–285, 1957.

5 Giovanna D'Agostino and Marco Hollenberg. Logical questions concerning the μ-calculus: interpolation, Lyndon and Łoś-Tarski. *J. Symbolic Logic*, 65(1):310–332, 2000.

6 Calvin C. Elgot. Decision problems of finite automata design and related arithmetics. *Trans. Amer. Math. Soc.*, 98:21–51, 1961.

7 Silvio Ghilardi. An algebraic theory of normal forms. *Ann. Pure Appl. Logic*, 71(3):189–245, 1995.

8 Silvio Ghilardi and Marek Zawadowski. Undefinability of propositional quantifiers in the modal system S4. *Studia Logica*, 55(2):259–271, 1995.

9 Leon Henkin. An extension of the Craig-Lyndon interpolation theorem. *J. Symbolic Logic*, 28:201–216, 1963.

10 David Janin and Igor Walukiewicz. Automata for the modal μ-calculus and related results. In *Mathematical foundations of computer science 1995 (Prague)*, volume 969 of *Lecture Notes in Comput. Sci.*, pages 552–562. Springer, Berlin, 1995.

11 Christian Kissig and Yde Venema. Complementation of coalgebra automata. In *Algebra and coalgebra in computer science*, volume 5728 of *Lecture Notes in Comput. Sci.*, pages 81–96. Springer, Berlin, 2009.

12 Clemens Kupke and Yde Venema. Coalgebraic automata theory: basic results. *Log. Methods Comput. Sci.*, 4(4):4:10, 43, 2008.

13 Johannes Marti. Relation liftings in coalgebraic modal logic. Master's thesis, Universiteit van Amsterdam, 2011.

14 Johannes Marti and Yde Venema. Lax extensions of coalgebra functors and their logic. *J. Comput. System Sci.*, 81(5):880–900, 2015.

15 Lawrence S. Moss. Coalgebraic logic. *Ann. Pure Appl. Logic*, 96(1-3):277–317, 1999. Festschrift on the occasion of Professor Rohit Parikh's 60th birthday.

16 Dirk Pattinson. The logic of exact covers: Completeness and uniform interpolation. In *Logic in Computer Science (LICS), 2013 28th Annual IEEE/ACM Symposium on*, pages 418–427. IEEE, 2013.

17 Andrew M. Pitts. On an interpretation of second-order quantification in first-order intuitionistic propositional logic. *J. Symbolic Logic*, 57(1):33–52, 1992.

18 Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Trans. Amer. Math. Soc.*, 141:1–35, 1969.

19 J. J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theoret. Comput. Sci.*, 249(1):3–80, 2000. Modern algebra and its applications (Nashville, TN, 1996).

20 Luigi Santocanale and Yde Venema. Uniform interpolation for monotone modal logic. In *Advances in modal logic. Volume 8*, pages 350–370. Coll. Publ., London, 2010.

21 Vladimir Yurievich Shavrukov. *Adventures in diagonalizable algebras.* ILLC Publications, 1994.

22 Yde Venema. Automata and fixed point logics for coalgebras. In *Proceedings of the Workshop on Coalgebraic Methods in Computer Science*, volume 106 of *Electron. Notes Theor. Comput. Sci.*, pages 355–375 (electronic). Elsevier, Amsterdam, 2004.

23 Albert Visser. Uniform interpolation and layered bisimulation. In *Gödel'96 (Brno, 1996)*, volume 6 of *Lecture Notes Logic*, pages 139–164. Springer, Berlin, 1996.

Generic Trace Semantics and Graded Monads*

Stefan Milius[1], Dirk Pattinson[2], and Lutz Schröder[1]

1 **Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany**
2 **The Australian National University, Australia**

Abstract

Models of concurrent systems employ a wide variety of semantics inducing various notions of process equivalence, ranging from linear-time semantics such as trace equivalence to branching-time semantics such as strong bisimilarity. Many of these generalize to system types beyond standard transition systems, featuring, for example, weighted, probabilistic, or game-based transitions; this motivates the search for suitable coalgebraic abstractions of process equivalence that cover these orthogonal dimensions of generality, i.e. are generic both in the system type and in the notion of system equivalence. In recent joint work with Kurz, we have proposed a parametrization of system equivalence over an embedding of the coalgebraic type functor into a monad. In the present paper, we refine this abstraction to use *graded monads*, which come with a notion of depth that corresponds, e.g., to trace length or bisimulation depth. We introduce a notion of graded algebras and show how they play the role of formulas in trace logics.

1998 ACM Subject Classification F.3.2 Semantics of Programming Languages, F.4.1 Mathematical Logic, D.3.1 Formal Definitions and Theory

Keywords and phrases Traces, transition systems, monads, coalgebra, trace logics

Digital Object Identifier 10.4230/LIPIcs.CALCO.2015.253

1 Introduction

Concurrent systems are typically modelled as state-based systems of some form, with a notion of state transition. Often, transitions between states are given by a transition relation, i.e. the system records only whether or not a transition between two given states is possible. More generally, however, the transition system may implement a more fine-grained modelling that specifies also, for example, the probability or the weight of a given transition, games determining transitions depending on the choices of participating agents, or sets of jointly reachable states. The aim of universal coalgebra [25] is to provide a unified framework for the treatment of various system types such as these.

A core topic in concurrent systems are notions of observable equivalence, which range from linear-time equivalences to branching-time equivalences [33]. While branching-time equivalences, based on suitable notions of bisimilarity, fit in seamlessly with the coalgebraic paradigm [25, 32, 11], other, in particular linear-time, equivalences require more effort. Coalgebraic treatments of trace semantics have previously been based on splitting the functor into a monad, the so-called *branching type*, and a functor, the *transition type*; the transition type is then transferred, by means of suitable distributive laws, to the Kleisli category [12] or the Eilenberg-Moore category [17, 14, 30, 5] of the branching monad, and trace equivalence is cast as bisimilarity in the new category.

* The first and the last author acknowledge support by the German Research Council (DFG) in the project COAX (MI 717/5-1/SCHR 1118/12-1).

6th International Conference on Algebra and Coalgebra in Computer Science (CALCO'15).
Editors: Lawrence S. Moss and Pawel Sobocinski; pp. 253–269
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In recent joint work with Kurz [19], we have proposed a quite simple-minded approach where the coalgebraic type functor, for which no particular form of decomposition needs to be assumed, is embedded into a monad M; *monadic trace semantics* is then obtained by iterating the coalgebra structure in the Kleisli category of M. This approach covers the standard examples including traces without explicit termination (not handled in other approaches) and probabilistic traces; moreover, it subsumes Kleisli-style and Eilenberg-Moore style trace semantics in the sense that these coarser semantics are obtained by applying natural quotient maps to monadic traces for the expected monads.

In more detail, monadic traces for a monad M (e.g. $MX = \mathcal{P}(\Sigma^* \times X)$ in the basic example of labelled transition systems) are sequences of elements of $M1$; the n-th element of such a sequence represents the traces obtained after n transitions. These traces typically all have length n, so the type $M1$ given to the n-th element of the trace sequence is unnecessarily general. This observation motivates using *graded monads*: a graded monad [31] associates to each set X a family of sets M_nX intuitively understood as consisting of the terms of uniform depth n over X in a given algebraic theory. It turns out that the key examples for trace semantics have easier and more general descriptions using graded monads than ordinary monads. We introduce *graded Eilenberg-Moore algebras* for graded monads, and observe that unlike in the modelling via vanilla monads, taking graded Eilenberg-Moore algebras as formulas leads to reasonable *trace logics*, i.e. logics that are invariant under trace equivalence. As a particularly tractable class of graded monads, we identify *depth-*1 *graded monads*, corresponding to algebraic theories with only shallow equations; this class contains most of the leading examples. For depth-1 graded monads, we establish a compositionality result for graded algebras, which amounts to a compositional syntax for trace formulas.

Related Work. We study finite traces, orthogonally to work on coalgebraic infinite traces [7]. The previous coalgebraic treatments of finite traces mentioned above [12, 17, 14, 30, 5] generally restrict to traces ending in accepting states, i.e. focus on language semantics in the sense of automata theory rather than on trace semantics of reactive systems [1]. Especially in the approach via the Eilenberg-Moore category [14, 30, 5], trace semantics is defined via determinization, while in the present paper we opt for a direct definition. Recent work by Klin and Rot [18] is, like the present paper and [17], concerned with trace logics. It takes a principled choice of trace logic as the *definition* of trace equivalence, while we give a semantic definition of trace equivalence and then develop logics that are invariant under trace equivalence. The path-based semantics of linear-time logics considered in [8] implicitly uses Kleisli composition in the graded monad induced by a Kleisli law.

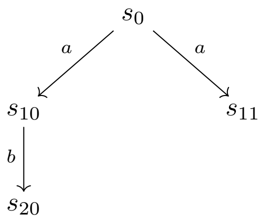
2 Preliminaries

We fix a base category $\mathbf{C}$ throughout. A *G-coalgebra* (X, γ) for a functor $G : \mathbf{C} \to \mathbf{C}$ consists of a $\mathbf{C}$-object X and a morphism $\gamma : X \to GX$. A *coalgebra morphism* between G-coalgebras (X, γ) and (Y, δ) is a morphism $f : X \to Y$ such that $\delta \circ f = Gf \circ \gamma$. When $\mathbf{C} = \mathsf{Set}$, we say that states $x \in X$ and $y \in Y$ in G-coalgebras (X, γ) and (Y, δ) are *behaviourally equivalent* if there exist coalgebra morphisms f, g with common codomain such that $f(x) = g(y)$. Behavioural equivalence can be approximated using the (initial ω-segment of the) *final coalgebra sequence* $(G^n1)_{n\in\mathbb{N}}$ where 1 is a final object in $\mathbf{C}$ and G^n is n-fold composition of G. The projections $p^{n+1}_n : G^{n+1}1 \to G^n1$ are defined by induction where $p^1_0 : G1 \to 1$ is the unique arrow to 1 and $p^{n+2}_{n+1} = G(p^{n+1}_n)$. For every G-coalgebra (X, γ), there is a *canonical cone* $\gamma_n : X \to G^n1$ defined inductively by $\gamma_0 : X \to 1$ and $\gamma_{n+1} = G(\gamma_n)\gamma$. We say that states $x \in X$, $y \in Y$ in G-coalgebras (X, γ) and (Y, δ) are *finite-depth behaviourally equivalent* if $\gamma_n(x) = \delta_n(y)$ for all $n \in \mathbb{N}$. If G is a finitary functor on Set, then finite-depth behavioural equivalence coincides with behavioural equivalence [34].

A *monad* is a triple (M, η, μ) where $M : \mathbf{C} \to \mathbf{C}$ is a functor, and $\eta : Id \to M$ and $\mu : MM \to M$ are natural transformations such that $\mu \circ M\eta = \mu \circ \eta M = id$ and $\mu \circ M\mu = \mu \circ \mu M$. Examples of monads are the powerset monad $\mathcal{P}$ and the distribution monad $\mathcal{D}$, which maps a set X to the set of functions $f : X \to [0,1]$ such that $\sum_{x \in X} f(x) = 1$. An *Eilenberg-Moore algebra* for a monad M is a morphism $a : MX \to X$ such that $a \circ \eta_X = id_X$ and $a \circ Ma = a \circ \mu_X$.

3 Monadic Trace Semantics, Informally

We recall monad-based trace semantics [19] using the basic example of labelled transition systems, and then motivate the transition to the more fine-grained modelling using graded monads. Consider the labelled transition system (LTS) over the alphabet $\Sigma = \{a, b\}$ depicted on the left below,



	Pretraces	Traces
Stage 0 :	$\{(\epsilon, s_0)\}$	$\{\epsilon\}$
Stage 1 :	$\{(a, s_{10}), (a, s_{11})\}$	$\{a\}$
Stage 2 :	$\{(ab, s_{20}))\}$	$\{ab\}$
Stage 3 :	$\emptyset$	$\emptyset$

whose traces at s_0 are the prefixes of ab. The idea of monadic trace semantics is to produce these traces from what we call *pretraces*, which in the case of LTS are pairs consisting of a trace and a poststate; pretraces are generated incrementally by iterating the coalgebra map representing the system in the Kleisli category of a suitable monad. LTS are standardly modelled as coalgebras for the functor G given on sets by $GX = \mathcal{P}(\Sigma \times X) \cong \mathcal{P}(X)^\Sigma$. We embed G into the monad M given by $MX = \mathcal{P}(\Sigma^* \times X)$ via an evident natural transformation α. Let γ be the G-coalgebra structure representing the above LTS. Then the pretraces of s_0 at stage n are the elements of $(\alpha\gamma)^n(s_0)$ where $(\alpha\gamma)^n$ denotes the n-fold Kleisli composite of $\alpha\gamma : X \to MX$, a morphism $X \to X$ in the Kleisli category of M. The traces are obtained as the first projections of the pretraces, i.e. at each stage the trace set is an element of $M1 \cong \mathcal{P}(\Sigma^*)$, as summarized in the table above right. We observe that the pretraces at stage n all have length n, so that viewing them just as elements of $\mathcal{P}(\Sigma^* \times X)$ loses information. One consequence of this loss of information is that a natural idea for developing a trace logic for a monadic trace semantics fails, as explained in Remark 3.2.

▶ **Remark 3.1.** A property of states is *trace-invariant* if it is closed under trace equivalence. In what follows, by a *trace logic* we mean a compositional syntax for trace-invariant properties. To set the stage for our considerations on trace logics, we remark that being trace-invariant alone is not a compositional property. E.g. in Hennessy-Milner logic, the formula $\Diamond_a \top \wedge \Diamond_b \top$ is trace-invariant – it states that a and b are both traces. Now any sufficiently expressive trace logic for LTS should presumably feature the operator $\Diamond_a$; however, $\Diamond_a(\Diamond_a \top \wedge \Diamond_b \top)$ fails to be trace invariant. Indeed, the (known) logics that characterise trace equivalence in labelled and probabilistic transition systems [3] (necessarily) do not come with the full set of standard boolean connectives.

Note also that in the case of probabilistic trace equivalence, the corresponding trace logic is not interpreted over the standard set $\{\bot, \top\}$ of truth values: for probabilistic systems, a formula $\Diamond_{a_1} \dots \Diamond_{a_n} \top$ evaluates to the probability of a system exhibiting a trace beginning with $a_1 \dots a_n$.

▶ **Remark 3.2.** In the above standard example, trace sets (at a given stage) are elements of $M1$, the carrier of the free Eilenberg-Moore algebra for M on one generator. A putative trace logic would have formulas whose evaluation at a state depends only on the traces of that state, i.e. the evaluation map will, at each stage, factor through $M1$. It is thus tempting to postulate that the semantics of trace formulas will arise via Eilenberg-Moore algebras for M on the set of intended truth values, say, on $2 = \{\bot, \top\}$ (see also a similar suggestion by Moggi [22]). Specifically, an M-algebra on 2 consists of a complete lattice structure, w.l.o.g. the usual one, and unary operations a, b; since the monad M arises via a distributive law between $\mathcal{P}$ and $\Sigma^* \times (-)$, a and b must moreover be join-continuous. There are only two join-continuous self-maps of 2, the identity and the constant map $\bot$. Thus, an M-algebra on 2 is determined by the subset $\Sigma_0 \subseteq \Sigma$ of letters that it interprets as the identity. Such an algebra yields an operator $\langle \Sigma_0 \rangle$ of our trace logic, and the formula $\langle \Sigma_0 \rangle \top$ holds for a state if each of its traces has a prefix mentioning only actions in Σ_0. However, this constraint is trivially satisfied for every trace by considering the empty prefix. Hence, we clearly want to impose a more precise condition: $\langle \Sigma_0 \rangle \top$ should be satisfied by states all of whose traces start with an action from Σ_0.

Let us make this point more formal. For any formula ϕ of a trace logic, its semantics $[\![\phi]\!]$ should be a sequence of predicates $[\![\phi]\!]_n : M1 \to 2$, $n \in \mathbb{N}$, determining at each stage n the sets of traces satisfying ϕ. Now for any operator L of the logic we expect to have a compositional definition of the semantics of L; that is, given a formula ϕ we wish to define the semantics of $L\phi$ knowing only the semantics $[\![\phi]\!]$ and using the interpretation of L as an M-algebra $[\![L]\!] : M2 \to 2$. With only the monad structure of M available, the only natural definition of the semantics of $L\phi$ that comes to mind requires that $[\![L]\!] M [\![\phi]\!]_n : MM1 \to 2$ factors through $\mu_1 : MM1 \to M1$, yielding $[\![L\phi]\!]_n : M1 \to 2$. But then $[\![L\phi]\!]_n = [\![L\phi]\!]_n \mu_1 M \eta_1 = [\![L]\!] M [\![\phi]\!]_n M \eta_1 = [\![L]\!] M ([\![\phi]\!]_n \eta_1)$ so that $[\![L\phi]\!]_n$ depends only on $[\![\phi]\!]_n \eta_1$, i.e. only on whether ϕ holds for the trace set $\{\epsilon\}$. Again, the problem here is that in the preimage of a trace set T under μ_1 we have the element $\{(\{\epsilon\}, T)\}$, i.e. we have no control over the length of prefixes that are split off T in the preimage under μ_1.

It is the core contribution of the current work to show that under a more fine-grained modelling via graded monads, discussed next, Eilenberg-Moore algebras do induce a reasonable notion of trace logic that is expressive enough in several important examples and, for so-called depth-1 graded monads, admits a compositional semantics, in fact following the ideas of the above remark.

4 Graded Monads

We discuss some of the basic theory of graded monads, originally introduced by Smirnov [31] (with grades in an arbitrary commutative monoid; here, we need only the case where grades are natural numbers). Recall that finitary monads on Set correspond to algebraic theories; under this correspondence, the functor part M of a monad may be thought of as mapping a set X to the set MX of terms over X modulo equality. A graded monad has sets $M_n X$ for all $n \in \mathbb{N}$, which may be thought of as sets of terms of uniform depth n.

▶ **Definition 4.1** (Graded Monad). A *graded monad* on $\mathbf{C}$ consists of a family of functors $M_n : \mathbf{C} \to \mathbf{C}$, $n \in \mathbb{N}$, a natural transformation $\eta : Id \to M_0$ (the *unit*), and a family of natural transformations

$$\mu^{nk} : M_n M_k \to M_{n+k} \qquad (n, k \in \mathbb{N}),$$

the *multiplication*. These data are subject to the *unit laws* for each $n \in \mathbb{N}$ on the left below,

$$\mu^{0n}\eta M_n = id_{M_n} = \mu^{n0} M_n \eta$$

$$\begin{tikzcd} M_n M_k M_m \arrow[r, "M_n \mu^{km}"] \arrow[d, "\mu^{nk} M_m"'] & M_n M_{k+m} \arrow[d, "\mu^{n,k+m}"] \\ M_{n+k} M_m \arrow[r, "\mu^{n+k,m}"'] & M_{n+k+m} \end{tikzcd}$$

and the *associative law* stating that for all $n, k, m \in \mathbb{N}$, the above right diagram commutes.

Notice that the above definition implies that (M_0, η, μ^{00}) is a monad. More abstractly, a graded monad can be defined either as a graded monoid in the endofunctor category $[\mathbf{C}, \mathbf{C}]$ [31] or as a lax monoidal functor $\mathbb{N} \to [\mathbf{C}, \mathbf{C}]$ (with $\mathbb{N}$ viewed as a discrete monoidal category), the latter making it an instance of the notion of parametric monad [21, 16].

Two standard equivalent presentations of monads carry over *mutatis mutandis* to the graded setting, namely Kleisli triples and, over Set, algebraic theories. Graded Kleisli triples have been introduced (in a more general setting) and proved equivalent to graded monads by Katsumata [16]. All we need here is the Kleisli star notation: For $f : X \to M_k Y$, we write

$$f^*_n = \mu^{nk}_Y M_n f : M_n X \to M_{n+k} Y.$$

The presentation of graded monads in terms of graded theories is a stepping stone for isolating depth-1 theories, which in turn are the key to obtaining compositional trace logics:

▶ **Definition 4.2** (Graded theory). A *graded theory* (Σ, E, d) consists of an algebraic theory, i.e. a (possibly class-sized) algebraic signature Σ and a class E of equations, and an assignment d of a *depth* $d(f) \in \mathbb{N}$ to every operation $f \in \Sigma$. This induces a notion of a term *having uniform depth* n: all variables have uniform depth 0, and $f(t_1, \dots, t_n)$ with $d(f) = k$ has uniform depth $n + k$ if all t_i have uniform depth n. (In particular, a constant c has uniform depth n for all $n \geq d(c)$). We then require that all equations $t = s$ in E have uniform depth, i.e. there exists n such that both t and s have uniform depth n. Moreover, we require that for every set X and every $k \in \mathbb{N}$, the class of terms of uniform depth k over variables from X modulo provable equality is small (i.e. in bijection with a set).

We defer the discussion of an example to the next section (Example 5.2.3).

Graded theories and graded monads on Set are essentially equivalent concepts; for the finitary case, this is implicit in [31]. In detail, a graded theory (Σ, E, d) induces a graded monad by taking $M_n X$ to be the set of Σ-terms over X that have uniform depth n, modulo equality derivable under E. Unit and multiplication are then defined as usual as conversion of variables into terms and collapsing of layered terms, respectively, noting that these operations behave as required w.r.t. uniform depth.

Conversely, a graded monad (M_n) over Set induces a graded theory (Σ, E, d) by taking Σ to be the disjoint union of all sets $M_n \kappa$ taken over all $n \in \mathbb{N}$ and all cardinals κ (so Σ is a proper class) and letting $f \in M_n \kappa$ have arity κ and depth n. Then every Σ-term t over X of uniform depth n has a canonical interpretation $[\![t]\!] \in M_n X$ defined recursively in the usual way, noting that this definition does produce an element of M_nX. We take E to consist of all equations $s = t$ of uniform depth n over X such that $[\![s]\!] = [\![t]\!]$ in M_nX.

Formally, these constructions establish an equivalence of categories between graded monads and graded monad morphisms in the obvious sense on one side, and graded theories and derived theory morphisms on the other side (i.e. maps that take signature symbols to terms, mapping axioms to derivable equations).

Examples. We proceed to discuss examples of graded monads; some of these are generic constructions that depend on grades being natural numbers.

▶ **Example 4.3.**

1. Every monad M with multiplication μ and unit η gives rise to a graded monad, by putting $M_n = M$ and $\mu^{nk} = \mu$.
2. If $F : \mathbf{C} \to \mathbf{C}$ is a functor, then $M_n = F^n$, the n-fold composition of F, defines a graded monad with unit $\eta = id$ and multiplication $\mu^{nk} = id_{F^{n+k}}$.
3. Let $\mathbf{C}$ have binary coproducts, and let $M_0 = Id$ and $M_{n+1} = FM_n + Id$. Define the natural transformations $\epsilon^{n,n+k} : M_n \to M_{n+k}$ by $\epsilon^{00} = id$, $\epsilon^{0,k+1} = \mathsf{inr}$ (right injection) and $\epsilon^{n+1,n+1+k} = F\epsilon^{nk} + Id$, and the multiplication $\mu^{nk} : M_nM_k \to M_{n+k}$ by $\mu^{0k} = id_{M_k}$ and $\mu^{n+1,k} = [\mathsf{inl} \circ F\mu^{nk}, \epsilon^{k,n+k+1}] : FM_nM_k + M_k \to FM_{n+k} + Id$. Then (M_n) is a graded monad with multiplication (μ^{nk}) and unit $\eta = id$. For $\mathbf{C} = \mathsf{Set}$, we may think of M_nX as the set of terms of depth *at most* n in an algebraic theory, i.e. (M_n) is the stratification of the free monad on F.

When $\mathbf{C}$ is monoidal, we have the following more specific example motivated fairly directly by trace semantics. For the sake of readability, we elide coherence isomorphisms.

▶ **Lemma 4.4.** *Let $(\mathbf{C}, \otimes, I)$ be a monoidal category, and let M be a strong monad on $\mathbf{C}$ with unit η, multiplication μ and strength t. Then for every object Σ of $\mathbf{C}$, the assignment*

$$M_nX = M(\Sigma^n \otimes X)$$

with the unit η and the multiplication with components

$$\mu_X^{nk} = (M(\Sigma^n \otimes M(\Sigma^k \otimes X)) \xrightarrow{Mt} M^2(\Sigma^{n+k} \otimes X) \xrightarrow{\mu} M(\Sigma^{n+k} \otimes X)),$$

where Σ^n denotes the n-th tensor power of the object Σ, defines a graded monad on $\mathbf{C}$.

($\Sigma = I$ yields Example 4.3.1.) We may think of Σ^n as an object of length-n traces; cf. Example 5.2.1.

Another example of graded monads is provided by monads that distribute over a functor by means of a so-called Kleisli law. Given a monad T with multiplication μ and unit η, and a functor F, a *Kleisli law* is a natural transformation $\lambda : FT \to TF$ such that $\lambda \circ F\eta = \eta F$ and $\lambda \circ F\mu = \mu_F \circ T\lambda \circ \lambda T$. It is well-known [23] that Kleisli laws are in 1-1 correspondence with liftings of F to the Kleisli category of M; they also induce graded monads:

▶ **Lemma 4.5.** *Let T be a monad with multiplication μ and unit η, F a functor, and $\lambda : FT \to TF$ a Kleisli-law. Define $\lambda^n : F^nT \to TF^n$ by*

$$\lambda^0 = id \qquad \textit{and} \qquad \lambda^n = \lambda^{n-1}F \circ F^{n-1}\lambda.$$

Then the data

$$M_n = TF^n \qquad \mu^{nk} = \mu F^{n+k} \circ T\lambda^n F^k$$

define a graded monad whose unit is the unit η of M.

A related example is obtained from a distributive law of a monad T over an endofunctor F (also called an *EM-law*), i.e., a natural transformation $\delta : TF \to FT$ such that $\lambda \circ \eta F = F\eta$ and $\lambda \circ \mu F = F\mu \circ \lambda T \circ T\lambda$. Such distributive laws are in 1-1 correspondence with liftings of F to the category of Eilenberg-Moore algebras for T [15], and like Kleisli laws they induce graded monads:

▶ **Lemma 4.6.** *Let M be a monad with multiplication μ and unit η, F a functor, and $\lambda : TF \to FT$ an EM-law. Define $\lambda^n : TF^n \to F^nT$ by*

$$\lambda^0 = id \qquad \text{and} \qquad \lambda^n = F\lambda^{n-1} \circ \lambda F^{n-1}.$$

Then the data

$$M_n = F^nT \qquad \mu^{nk} = F^{n+k}\mu \circ F^n\lambda^kT$$

define a graded monad whose unit is the unit η of M.

(Lemmas 4.5 and 4.6 both have Example 4.3.2 as a trivial special case.) This lemma is a 2-categorical dual of Lemma 4.5. Note that no accessibility assumptions on F or T are needed; contrastingly, to obtain a monad from F and T in the situation of the lemma as in [19], one needs to assume that F and T are finitary. Intuitively, a distributive law $TF \to FT$ allows shifting all F-operations to the top of a term only for terms of sufficiently uniform shape, i.e. those in TF^n.

5 Trace Semantics Via Graded Monads

We now give our generic definition of coalgebraic trace semantics, induced by a natural transformation from the coalgebraic type functor into a graded monad.

▶ **Definition 5.1** (Trace semantics). A *trace semantics* for G-coalgebras consists of a graded monad $(M_n)_{n\in\mathbb{N}}$ and a natural transformation

$$\alpha : G \to M_1.$$

The *α-pretrace sequence* $(\gamma^{(n)} : X \to M_nX)_{n\in\mathbb{N}}$ for a G-coalgebra $\gamma : X \to GX$ is defined by induction on n: $\gamma^{(0)} = \eta_X : X \to M_0X$ and

$$\gamma^{(n+1)} = (\gamma^{(n)})^*_1 \circ \alpha_X \circ \gamma = (X \xrightarrow{\alpha\gamma} M_1X \xrightarrow{M_1\gamma^{(n)}} M_1M_nX \xrightarrow{\mu^{1n}_X} M_{n+1}X).$$

The *α-trace sequence* T^α_γ is the sequence

$$(M_n! \circ \gamma^{(n)} : X \to M_n1)_{n\in\mathbb{N}}.$$

Over an unrestricted base category, we just view the α-trace sequence as the α-trace semantics, speaking informally of *α-trace equivalence* as identification under α-trace semantics, and of properties of states being *α-trace invariant* if they depend only on the α-trace sequence. If $\mathbf{C} = \mathsf{Set}$ then states $x \in X$, $y \in Y$ in G-coalgebras (X, γ), (Y, δ) are *α-trace equivalent* if $M_n!\circ\gamma^{(n)}(x) = M_n!\circ\delta^{(n)}(y)$ for all $n \in \mathbb{N}$. We think of M_nX as containing length-n pretraces over X and of M_n1 as containing length-n traces. The morphism $M_n! : M_nX \to M_n1$ forgets the poststate of a pretrace.

The graded monad (M_n) is a parameter of the framework, and typically arises by imposing additional equational laws such as distributivity on the graded monad $(G^n)_{n<\omega}$ of Example 5.2.4.

▶ **Example 5.2.** We proceed to elaborate some concrete instances of trace semantics via graded monads, beginning with our initial motivating example. In all these examples, α is just identity. (As a trivial example where α is not identity, take $M_nX = 1$, which makes all states α-trace equivalent.)

1. *Trace semantics of labelled transition systems:* Labelled transition systems are coalgebras for the functor G defined by $GX = \mathcal{P}(\Sigma \times X)$. To capture standard trace semantics, we define a graded monad $((M_n), \eta, (\mu^{nk}))$ by

$$M_nX = \mathcal{P}(\Sigma^n \times X),$$

with $\eta_X(x) = \{(\epsilon, x)\} \in M_0(X)$ for $x \in X$, and $\mu^{nk}(S) = \{(uv, x) \mid \exists (u, V) \in S. (v, x) \in V\}$. This is in fact just a special case of Lemma 4.4. We then have that given a state x in a G-coalgebra $\gamma : X \to \mathcal{P}(\Sigma \times X)$, i.e. in a labelled transition system, $\gamma^{(n)}(x) \in \mathcal{P}(\Sigma^n \times X)$ is the set of pairs (w, y) where w is a length-n trace of x and y is the corresponding poststate. Thus, the n-th component $M_n!\gamma^{(n)}$ of the α-trace sequence maps x to the the set of its length-n traces.

2. *Trace semantics of probabilistic labelled transition systems:* Recall that *generative probabilistic (transition) systems* (for simplicity without the possibility of deadlock, the latter not to be confused with explicit termination) are modelled as coalgebras for the functor $\mathcal{D}(\Sigma \times -)$ where $\mathcal{D}$ denotes the discrete distribution functor (i.e. $\mathcal{D}(X)$ is the set of discrete probability distributions on X, and $\mathcal{D}(f)$ takes image measures under f). That is, a coalgebra structure $\gamma : X \to \mathcal{D}(\Sigma \times X)$ assigns to each state $x \in X$ a probability distribution over pairs of actions and successor states. As in the previous example, we obtain a graded monad for probabilistic trace semantics as an instance of the construction from Lemma 4.4. In this case, we have $M_nX = \mathcal{D}(\Sigma^n \times -)$; $\eta(x)$ is the Dirac distribution at (ϵ, x); and for $\nu \in \mathcal{D}(\Sigma^n \times \mathcal{D}(\Sigma^k \times X))$,

$$\mu^{nk}(\nu)(u, y) = \sum_{u=vw, w \in \Sigma^k, \rho \in \mathcal{D}(\Sigma^k \times X)} \nu(v, \rho)\rho(w, y)$$

for $(u, y) \in \Sigma^{n+k} \times X$. We identify M_n1 with $\mathcal{D}(\Sigma^n)$. Thus, given a state x in a G-coalgebra γ, i.e. in a generative probabilistic system, $(M_n!\gamma^{(n)})(x)(u)$ is the probability of x to exhibit a trace $u \in \Sigma^n$ when run for n steps; this captures the standard notion of probabilistic trace [6].

3. *Mazurkiewicz traces:* The trace semantics proposed by Mazurkiewicz [20] takes concurrent actions into account. Given an action alphabet Σ and an *independence relation* I, i.e., a symmetric and irreflexive relation $I \subseteq \Sigma \times \Sigma$, let W_I denote the monoid obtained by quotienting Σ^* modulo commutation of independent letters, and put $W_I^n = \{[w] \in W_I \mid |w| = n\}$. Then $M_nX = \mathcal{P}(W_I^n \times X)$ models length-n Mazurkiewicz pretraces, consisting of a Mazurkiewicz trace and a poststate. Defining the unit and multiplication analogously as for standard traces, we obtain a graded monad $\mathbb{M}$ for Mazurkiewicz traces.

   By the considerations in Section 4, $\mathbb{M}$ corresponds to a graded theory. For the finitely branching case, i.e. replacing $\mathcal{P}$ with the finite powerset functor $\mathcal{P}_\mathsf{f}$, this theory is explicitly described as follows. It has the join semilattice operations and equations as operations and equations of depth 0, and one unary operation a of depth 1, called an *action*, for every $a \in \Sigma$. The theory expresses distribution of actions over the join semilattice structure, by the depth-1 equations

$$a(\bot) = \bot \qquad a(x \vee y) = a(x) \vee a(y), \tag{1}$$

and commutation of independent actions a, b by depth-2 equations $a(b(x)) = b(a(x))$.

4. *Finite-depth behavioural equivalence:* Recall that states in labelled transition systems are *finitely bisimilar* [10] if Duplicator wins all finite-length bisimulation games. More generally, two states in G-coalgebras are *finite-depth behaviourally equivalent* if their images coincide in all stages of the final sequence [27]; see Section 2. Finite-depth behavioural equivalence is an instance of α-trace equivalence: take the graded monad induced by G as in Example 4.3.2, i.e. $M_n = G^n$. Then for a state x in a G-coalgebra γ, $M! \circ \gamma^{(n)}(x) \in G^n 1$ is the image of x in the n-th stage of the final sequence under the canonical cone [19]; i.e. two states are α-trace equivalent iff they are finite-depth behaviourally equivalent.

5. *Kleisli liftings:* If G is of the form $G = TF$ for a functor F and a monad T with a Kleisli law $\lambda : FT \to TF$, then we obtain a graded monad with $M_n = TF^n$ as in Lemma 4.5. The arising α-trace equivalence is typically finer than the language equivalence targeted in previous work on Kleisli-lifting based semantics [12], which for the sake of distinction we shall refer to as *generic language semantics*. E.g. in the base example for language semantics, non-deterministic automata, the coalgebraic model is given by $G = \mathcal{P}(1 + \Sigma \times (-))$ for an alphabet Σ, with $1 = \{\checkmark\}$ denoting explicit termination, i.e. acceptance; here, $T = \mathcal{P}$, and $F = 1 + \Sigma \times -$. Language semantics effectively has TA as its semantic domain, where A is the initial F-algebra; in this case, $TA = \mathcal{P}(\Sigma^*)$, and indeed generic language semantics instantiates exactly to the standard language semantics of nondeterministic automata. In other words, generic language semantics focusses entirely on explicit termination; in cases where the latter is not present, e.g. labelled transition systems, generic language semantics becomes trivial.

   Contrastingly, α-trace equivalence in $TF^n 1 \cong \mathcal{P}(\sum_{i=0}^{n} \Sigma^i)$ records, at stage n, not only the accepted words of length at most $n-1$ (gathered in the first $n-1$ summands) but also those words of length n that are traces in the same sense as for labelled transition systems, i.e. can be run without blocking. This is similar in spirit to the denotational semantics of CSP [13], which distinguishes deadlock from successful termination $\checkmark$. Generic language semantics is obtained by just forgetting this last summand, and one can generalize this observation to show that generic language semantics is obtained as a natural quotient of α-trace semantics [19].

6. *Eilenberg-Moore liftings:* An alternative approach to generic language semantics defines trace equivalence as bisimilarity in a generic determinization that can be constructed under certain conditions [14]. We shall refer to this approach as *extension semantics*. It is based on assuming a coalgebraic type functor of the form $G = FT$ where F is a functor and T is a monad, together with a EM-law $\delta : TF \to FT$. The domain of extension semantics is the final coalgebra Z of F. The standard example of non-deterministic automata is subsumed under this approach by taking $FX = 2 \times X^\Sigma$ and $T = \mathcal{P}$. Here, $Z \cong \mathcal{P}(\Sigma^*)$, and extension semantics captures precisely the language semantics of non-deterministic automata. Like Kleisli-style generic language semantics, extension semantics becomes trivial in the absence of explicit termination; e.g. when we change F to be just $FX = X^\Sigma$, then the final F-coalgebra becomes trivial.

   In our framework, we define a graded monad with $M_n = F^n T$ as in Lemma 4.6. In the example of non-deterministic automata, we have $M_n 1 = F^n \mathcal{P} 1$ with $FX = 2 \times X^\Sigma$, i.e. $M_n 1$ consists of Σ-branching trees of uniform depth n, with inner nodes labelled in $2 = \{\bot, \top\}$ and leaves in $\mathcal{P}1$. Such a tree may be identified with a set A of words w of length $\leq n$ over Σ: if $|w| < n$ then $w \in A$ iff the inner node addressed by w is labelled by $\top$; w is then *accepted*. If $|w| = n$ then $w \in A$ iff the leaf node addressed by w is labelled by 1; w is then a *trace*, i.e. a state having w in its trace sequence at stage n can execute

w without deadlocking. Language equivalence is recovered from α-trace equivalence by canonically forgetting the information about traces; see [19] for details.

6 Graded Algebras

Fix for this section a graded monad $\mathbb{M} = ((M_n), \eta, (\mu^{nk}))$. As we think of M_n as constructing terms of uniform depth n, it is natural to take graded algebras as providing an interpretation of depth-n-terms to which additional layers can be added uniformly. The M_n-algebras introduced below allow interpreting terms up to uniform depth n, and M_ω-algebras terms of arbitrary depth.

▶ **Definition 6.1** (Graded algebras). For a given natural number n, an *M_n-algebra* $A = ((A_k)_{k \le n}, (a^{mk})_{m+k \le n})$ consists of a family of carrier objects A_i and structure morphisms

$$a^{mk} : M_m A_k \to A_{m+k}$$

such that $a^{0m} \eta_{A_m} = id_{A_m}$ for all $m \le n$, and whenever $m + r + k \le n$, the diagram on the left

$$\begin{tikzcd}
M_m M_r A_k \arrow[r, "M_m a^{rk}"] \arrow[d, "\mu^{mr}_{A_k}"'] & M_m A_{r+k} \arrow[d, "a^{m,r+k}"] & & M_m A_k \arrow[r, "M_m f_k"] \arrow[d, "a^{mk}"'] & M_m B_k \arrow[d, "b^{mk}"] \\
M_{m+r} A_k \arrow[r, "a^{m+r,k}"'] & A_{m+r+k} & & A_{m+k} \arrow[r, "f_{m+k}"'] & B_{m+k}
\end{tikzcd} \qquad (2)$$

commutes. A *morphism* of M_n-algebras from $A = ((A_k), (a^{mk}))$ to $B = ((B_k), (b^{mk}))$ is a family of morphisms $f_k : A_k \to B_k$, $k \le n$, such that the right hand diagram above commutes whenever $m + k \le n$. *M_ω-algebras* and their morphisms are defined similarly but with all indices ranging over $m, k, r \in \mathbb{N}$.

As expected, applying a graded monad to a given set yields a free algebra:

▶ **Proposition 6.2.** *For every $n \in \mathbb{N}$, $FX = ((M_m X)_{m \le n}, (\mu^{mk})_{m+k \le n})$ is an M_n-algebra, the free M_n-algebra over X w.r.t. the forgetful functor U_n that maps an M_n-algebra $((A_k), (a^{mk}))$ to A_0 and a morphism (f_k) to f_0; similarly for M_ω-algebras.*

In other words, M_n-algebras realize the monad M_0 by an adjunction; for $n = 0$, we just obtain the usual Eilenberg-Moore construction for M_0. For later use in the semantics of trace formulas, we note

▶ **Proposition 6.3.** *If $\mathbf{C}$ has products, then the category of M_n-algebras has products described as follows. The product of a family of M_n-algebras $A^i = ((A^i_k)_{k \le n}, (a^{mk}_i)_{m+k \le n})$ indexed over $i \in I$ has carriers $\prod_{i \in I} A^k_i$ for $k \le n$ and structure morphisms being composites*

$$M_m \prod_{i \in I} A^i_k \xrightarrow{\langle M_m \pi_i \rangle} \prod_{i \in I} M_m A^i_k \xrightarrow{\prod_{i \in I} a^{mk}_i} \prod_{i \in I} A^i_{m+k}.$$

7 Depth-1 Theories

Graded algebras in general need to be constructed monolithically – due to the entanglement between the structure morphisms imposed by Diagram (2), it is not in general possible to combine, say, an M_n-algebra and an M_k-algebra into an M_{n+k}-algebra. A combination mechanism becomes possible, however, if we restrict the depth of equations in the associated graded theory, as follows.

▶ **Definition 7.1** (Depth-1 generation and presentation)**.** We say that a graded theory is *depth-1-generated* if all its operations have depth 1, and *depth-1-presented* or just *depth-1* if additionally all its equations have depth 1. A graded monad on Set is said to have these properties if it can be generated by a corresponding graded theory.

▶ **Example 7.2.** All graded monads in Example 5.2 except the one in Example 5.2.3 are depth-1.

We proceed to develop a more abstract characterization of depth-1 monads usable over arbitrary base categories. Recall that an *epi-transformation* between set functors is a natural transformation with surjective components.

▶ **Proposition 7.3.** *A graded monad $\mathbb{M} = ((M_n), \eta, (\mu^{nk}))$ on* Set *is depth-*1 *generated iff all μ^{nk} are epi-transformations, equivalently if all μ^{1k} are epi-transformations. Moreover, $\mathbb{M}$ is depth-*1 *iff additionally the diagram below is a coequalizer diagram for every n:*

$$\begin{tikzcd} M_1M_0M_n \arrow[r, shift left, "M_1\mu^{0n}"] \arrow[r, shift right, "\mu^{10}M_n"'] & M_1M_n \arrow[r, "\mu^{1n}"] & M_{1+n}\,. \end{tikzcd} \tag{3}$$

▶ **Remark 7.4.** Notice that the coequalizer (3) is reflexive; indeed we have $M_1\mu^{0n} \circ M_1\eta M_n = id_{M_1M_n} = \mu^{10}M_n \circ M_1\eta M_n$ by the unit law of the graded monad.

This motivates the following definition (over unrestricted base categories).

▶ **Definition 7.5.** A graded monad $\mathbb{M} = ((M_n), \eta, (\mu^{nk}))$ is *depth-*1 *generated* if all μ^{nk} are epi-transformations. Moreover, $\mathbb{M}$ is *depth-*1 if it is depth-1 generated and for every n, the diagram (3) is a coequalizer diagram, and $M_0\mu^{1n}$ is an epi-transformation.

▶ **Remark 7.6.** [leftmargin=0pt,itemindent=3em]

1. Proposition 7.3 shows that Definition 7.1 and Definition 7.5 agree where both apply, i.e. for graded monads on Set. The condition that $M_0\mu^{1n}$ be an epi-transformation is automatic in this case, since each μ^{1n}_X is a coequalizer (hence a surjective map) and every functor on Set preserves surjective maps.
2. The condition that $M_0\mu^{1n}$ is an epi-transformation holds as soon as $\mathbf{C}$ is an algebraic category such that every finitely presentable object is regular projective and M_0 is finitary. Indeed, by [2, 6.30] M_0 preserves sifted colimits (and, in particular, reflexive coequalizers). Thus, $M_0\mu^{1n}$ is a (reflexive) coequalizer and therefore an epi-transformation.

The salient point about depth-1 monads is that they allow reducing M_n-algebras to families of M_1-algebras. We begin with morphisms:

▶ **Proposition 7.7.** *If $\mathbb{M}$ is depth-*1*-generated then given M_n-algebras $((A_k), (a^{kl}))$ and $((B_k), (b^{kl}))$, a family of maps $f_k : A_k \to B_k$ is a morphism of M_n-algebras iff for each $l < n$, (f_l, f_{l+1}) is a morphism of M_1-algebras; i.e. $f_{1+l}a^{1l} = b^{1l}M_1f_l$, and each f_l is a morphism $(A_l, a^{0l}) \to (B_l, b^{0l})$ of M_0-algebras.*

We now present our main technical result, which states essentially that M_n-algebras for depth-1 monads can be assembled from M_1-algebras:

▶ **Theorem 7.8.** *Let $\mathbb{M} = ((M_n), \eta, (\mu^{nk}))$ be a depth-*1 *graded monad, and let $n \in \mathbb{N}$. Then every family of morphisms*

$$a^{1k} : M_1A_k \to A_{k+1}, \qquad a^{0k} : M_0A_k \to A_k \qquad (k \le n)$$

such that for each $k < n$, $(a^{0k}, a^{0,k+1}, a^{1k})$ form an M_1-algebra extends uniquely to an M_n-algebra.

In other words, combining this with the previous proposition, we have that in the depth-1-case, an M_n-algebra is just a chain of M_1-algebras with compatible M_0-parts.

▶ **Remark 7.9.** In the corner case where $M_n = F^n$ for an endofunctor F (Example 4.3.2), M_0-algebras are trivial and M_1-algebras are just maps $FA_0 \to A_1$. Therefore, the *graded objects* studied by Ghilardi and Bezhanishvili [9, 4] can formally be seen as M_ω-algebras with additional structure.

8 Trace Logics

We now return to our original goal, to identify a generic notion of α-trace logic, understood as a compositional syntax for α-trace-invariant properties (see Remark 3.1). The key ingredient in our approach is the compositionality of graded algebras for depth-1 monads (Theorem 7.8): We use M_1-algebras as modal operators; by Theorem 7.8, we can build an M_n-algebra out of n such operators. By Proposition 6.2, we can then use M_n-algebras (A_k) as formulas describing α-traces of length n: we fix a truth value, i.e. an element $\tau : 1 \to A_0$, and obtain a morphism $\tau^\#$ of M_n-algebras by free extension. In particular, the diagram

$$\begin{tikzcd}
M_1M_k1 \arrow[r, "M_1\tau_k^\#"] \arrow[d, "\mu^{1k}"'] & M_1A_k \arrow[d, "a^{1k}"] \\
M_{1+k}1 \arrow[r, "\tau_{1+k}^\#"'] & A_{1+k}
\end{tikzcd}$$

commutes for $1 + k \leq n$, thus precisely realizing the idea for a compositional semantics of operators that previously failed for ordinary monads (Remark 3.2). Before we introduce more specific syntax, we formally fix the semantics as just indicated:

▶ **Definition 8.1.** An *α-trace property* (A, τ) *of rank* n consists of an M_n-algebra $A = ((A_k), (a^{mk}))$ and a distinguished global element $\tau : 1 \to A_0$ called the *base*. We think of the elements of the A_k as truth values, and refer to A_n as the *type* of (A, τ). The *evaluation* of (A, τ) on a G-coalgebra $\gamma : C \to GC$ is the morphism

$$C \xrightarrow{\gamma^{(n)}} M_nC \xrightarrow{M_n!} M_n1 \xrightarrow{\tau_n^\#} A_n\,,$$

where $\tau^\#$ is the unique homomorphism from the free M_n-algebra on 1, $(M_k1)_{k\leq n}$, to A such that $\tau_0^\#\eta = \tau$. (In particular, α-trace properties are, by definition, α-trace invariant, i.e their evaluation factors through the α-trace sequence.)

We now develop a generic notion of α-trace formula as a syntax for α-trace properties, with a number of syntactic and semantic parameters that can be chosen freely. Given an α-trace property (A, τ), the A_i are, in principle, arbitrary M_0-algebras; however, the current set of examples suggests that it suffices to choose the A_i as powers of a fixed M_0-algebra Ω of truth values. We thus arrive at the following definition of generic α-trace logic.

Syntax. We parametrize the syntax over signatures Λ and Θ where Λ consists of modal operators with given finite arities and Θ of truth constants. α-Trace formulas ϕ of rank n are then defined by induction over n: the α-trace formulas of rank 0 are the truth constants; α-trace formulas ϕ of rank $n + 1$ have the form

$$\phi ::= L(\phi_1, \dots, \phi_k)$$

where $L \in \Lambda$ is k-ary, and $\phi_1, \dots, \phi_k$ are α-trace formulas of rank n. (Again, observe that this implies that when L is nullary, the formula L has rank n for every $n \geq 1$.)

Semantics. We assume from now on that $\mathbf{C}$ has finite products. As parameters of the semantics, we fix an M_0-algebra Ω with structure map $\omega : M_0\Omega \to \Omega$ serving as an object of truth values, and interpretations of the signature symbols. We let Ω^n denote the n-th Cartesian power of Ω as an M_0-algebra, with structure map $\omega^{(n)}$ (formed as in Proposition 6.3). An n-ary modal operator $L \in \Lambda$ is interpreted as a morphism $[\![L]\!] : M_1(\Omega^n) \to \Omega$ such that $(\omega^{(n)}, \omega, [\![L]\!])$ form an M_1-algebra with carriers $\Omega_0 = \Omega^n$, $\Omega_1 = \Omega$; explicitly, ω and $\omega^{(n)}$ are algebras for the monad M_0 and the diagrams

$$\begin{tikzcd}
M_1M_0(\Omega^n) \arrow[r, "M_1\omega^{(n)}"] \arrow[d, "\mu^{10}"'] & M_1(\Omega^n) \arrow[d, "{[\![L]\!]}"] \\
M_1\Omega^n \arrow[r, "{[\![L]\!]}"'] & \Omega
\end{tikzcd}
\qquad
\begin{tikzcd}
M_0M_1(\Omega^n) \arrow[r, "M_0{[\![L]\!]}"] \arrow[d, "\mu^{01}"'] & M_0\Omega \arrow[d, "\omega"] \\
M_1(\Omega^n) \arrow[r, "{[\![L]\!]}"'] & \Omega
\end{tikzcd}$$

commute. Finally, a truth constant $c \in \Theta$ is interpreted as a truth value $[\![c]\!] : 1 \to \Omega$.

The semantics of an α-trace formula ϕ is an α-trace property $[\![\phi]\!]$ of type Ω, defined recursively as follows. For $c \in \Theta$, we put (overloading notation)

$$[\![c]\!] = (\Omega, [\![c]\!]),$$

an α-trace property of rank 0. For an α-trace formula $L(\phi_1, \dots, \phi_k)$ of rank $n+1$, we form the product of the rank-n α-trace properties $[\![\phi_1]\!], \dots, [\![\phi_k]\!]$; explicitly, this product is formed by taking products of M_n-algebras as in Proposition 6.3, and by tupling the bases (observe that the evaluation of the product according to Definition 8.1 is the tuple formed from the evaluations of the component properties). We thus obtain a rank-n α-trace property $(((A_r)_{r\leq n}, (a^{mr})_{m+r\leq n}), \tau)$ of type Ω^k. Using Theorem 7.8, we then extend the latter to a rank-$(n+1)$ α-trace property $(((A_r)_{r\leq n+1}, (a^{mr})_{m+r\leq n+1}), \tau)$ of type Ω by taking $A_{n+1} = \Omega$, $a^{0,n+1} = \omega$, and $a^{1n} = [\![L]\!] : A_n = \Omega^k \to \Omega = A_{n+1}$.

▶ **Example 8.2.**

1. *Labelled transition systems.* As truth value object, we take $2 = \{\bot, \top\}$ with the usual join semilattice structure; we put $\Theta = \{\top\}$ and $[\![\top]\!] = \top : 1 \to 2$. We could then take Λ to consist just of unary modal operators of the form $\langle a \rangle$, interpreted as

$$[\![\langle a \rangle]\!] : \mathcal{P}(\Sigma \times 2) \to 2, \quad S \mapsto \begin{cases} \top & (a, \top) \in S \\ \bot & \text{otherwise} \end{cases}$$

   much as in Remark 3.2. This defines exactly the usual trace logic for LTS (in particular is already sufficient to distinguish states up to trace equivalence): $\langle a \rangle\phi$ says that there exists a trace that begins with a and continues with a trace satisfying ϕ.
   We obtain a slightly more interesting logic by extending Λ with operators of higher arity. Due to the equations imposed by the graded monad $(M_n) = (\mathcal{P}(\Sigma^n \times -))$, an M_1-algebra with carriers 2^k, 2 interprets $a \in \Sigma$ as a join-continuous map $2^k \to 2$; such maps have the form $(b_1, \dots, b_k) \mapsto \bigvee_{i\in I} b_i$ for some $I \subseteq \{1, \dots, k\}$. Thus, we can introduce k-ary operators L of the form

$$L(\phi_1, \dots, \phi_k) = \bigvee_{a\in\Sigma} \langle a \rangle \bigvee_{i\in I_a} \phi_i \qquad (I_a \subseteq \{1, \dots, k\});$$

   that is, we enrich the language with disjunction.

2. *Probabilistic trace logic:* Recall that generative probabilistic transition systems are coalgebras for $\mathcal{D}(\Sigma \times -)$, and their trace semantics is given by the graded monad $(M_n) = (\mathcal{D}(\Sigma^n \times -))$; in particular, $M_0 \cong \mathcal{D}$. To obtain a trace logic, we take $\Lambda = \{\langle \Sigma_0 \rangle \mid \Sigma_0 \subseteq \Sigma\}$, and $\Theta = \{1\}$. We choose $\Omega = [0, 1]$ as the object of truth values, made into a $\mathcal{D}$-algebra by taking expected values, i.e. a formal convex combination $\sum p_i q_i$ over $[0, 1]$ is mapped to the arithmetic sum $\sum p_i q_i$. We put $[\![1]\!] = 1 \in [0, 1]$. Finally, we interpret the modal operator $\langle \Sigma_0 \rangle$ by

   $$[\![\langle \Sigma_0 \rangle]\!] : \mathcal{D}(\Sigma \times [0, 1]) \to [0, 1], \qquad \mu \mapsto \textstyle\sum_{a \in \Sigma_0, p \in [0,1]} p\mu(\{(a, p)\}).$$

   Then a formula $\langle \Sigma_n \rangle \cdots \langle \Sigma_1 \rangle p$ evaluates, at a state c, to p times the probability that c takes a trace in $\Sigma_n \cdots \Sigma_1$; up to the slightly more general syntax, this is exactly the usual trace logic for generative probabilistic transition systems (see, e.g., [3]). Similarly as in the previous example, we can move to a richer language with higher-arity modal operators. As the distributive law behind the multiplication of $\mathcal{D}(\Sigma^n \times -)$ (Example 5.2.2) amounts to requiring that an M_1-algebra with carriers $[0, 1]^k$, $[0, 1]$ interprets every $a \in \Sigma$ as a morphism $[0, 1]^k \to [0, 1]$ of $\mathcal{D}$-algebras, we thus extend the language with affine maps (in analogy to adding disjunction in the case of LTS), i.e. with formulas $c + \sum_i q_i \phi_i$, subject to the proviso that $(x_i) \mapsto c + \sum q_i x_i$ defines a map $[0, 1]^k \to [0, 1]$. In particular, the extended language includes fuzzy negations $1 - \phi$.

3. *Coalgebraic modal logic:* Recall that finite-depth behavioural equivalence on G-coalgebras is α-trace equivalence for the graded monad $M_n X = G^n X$ (Example 5.2.4). Now finite-depth behavioural equivalence is precisely the equivalence described by coalgebraic modal logic for a separating set of predicate liftings (no assumptions are needed on the functor) [24, 28, 29]. The simplest example is Hennessy-Milner logic over labelled transition systems; other examples include probabilistic, graded, and neighbourhood-based logics [26]. In fact, coalgebraic modal logic can be seen as an α-trace logic. Specifically, let Λ be a signature of finitary (possibly nullary) modal operators L, with given interpretations as *predicate liftings* for G. The latter are equivalent to subsets of $G(2^k)$ where k is the arity, i.e. to maps $[\![L]\!] : G(2^k) \to 2$. Predicate liftings are closed under Boolean combination [28], so we can assume that Λ is closed under Boolean combinations. Therefore, we can restrict the syntax of coalgebraic modal logic to nothing but closed terms formed from the operations in Λ (the only other standard ingredient are Boolean operators, now absorbed by Λ).

   We define an α-trace logic by taking the same Λ, and $\Theta = \emptyset$; moreover, we take $\Omega = 2$ to be the truth value object. The interpretations Λ already have the required type $G(2^k) \to 2$; since M_0 is the identity monad, and the M_0-algebra structure on 2 is therefore trivial, there are no further conditions to check. Trace formulas over Λ and Θ are, then, exactly the same as formulas in coalgebraic modal logic over Λ, and the semantics is the same in both settings.

▶ **Remark 8.3.** The above examples seem to indicate that there is no single canonical choice for the truth value object Ω. In some cases, the free M_0-algebra $M_0 1$ will do, as in Example 8.2.1 or in a variant of Example 8.2.2 that uses subprobabilities instead of probabilities. As it stands, Ω is isomorphic to $M_0 2$ in Example 8.2.2, similarly in Example 8.2.3. Given Ω, a morphism $M_1(\Omega^n) \to \Omega$ corresponds to an n-ary lifting of Ω-valued predicates for M_1, i.e. a transformation $(\Omega^X)^n \to \Omega^{M_1 X}$, natural in X [28]; we leave the analysis of the predicate liftings arising from the interpretations $[\![L]\!] : M_1(\Omega^n) \to \Omega$ of n-ary modal operators L to future work.

▶ **Remark 8.4.** In all the above examples, the trace logic is *expressive*, i.e. logically equivalent states are α-trace equivalent. In the general case, it is trivial to come up with an expressive set of α-trace *properties*: just take, for each n, the free M_n-algebra over 1 (Proposition 6.2) as an α-trace property of rank n. Of course, this is uninteresting, as it amounts to just taking trace sets as logical formulas; also, it does not constitute a compositional syntax for α-trace *formulas*. We leave the identification of criteria for expressiveness of a given trace logic to future research.

9 Conclusions and Future Work

We have shown how many forms of trace semantics of coalgebras, including the usual trace semantics of nondeterministic and probabilistic labelled transition systems and Mazurkiewicz traces as well as finite-depth behavioural equivalence, can be modelled uniformly by embedding the coalgebraic type functor into a graded monad. A salient point about this approach is that it constitutes, to our best understanding, the first native semantic definition of generic trace equivalence, while existing approaches start either from a determinization procedure or a trace logic.

We have introduced a notion of graded algebras, which serve as trace-invariant properties. As our main technical result, we have shown that for the more restrictive class of depth-1 monads, graded algebras can be built in a modular fashion. This gives rise to a compositional syntax for trace-invariant logics. We have illustrated how such logics arise for our main examples of trace semantics, thus regaining and extending standard logics in the case of plain and probabilistic traces, and coalgebraic modal logic in the case of finite-depth behavioural equivalence.

Future investigations will be directed at analysing the expressivity as well as algorithmic aspects of trace logics, including the exploration of temporal extensions.

Acknowledgements. We wish to thank Alexander Kurz and Tadeusz Litak for useful discussions and pointers to the literature, and Erwin R. Catesbeiana for hints on inconsistent graded monads.

References

1 Luca Aceto, Anna Ingólfsdóttir, Kim Larsen, and Jiři Srba. *Reactive systems: modelling, specification and verification*. Cambridge Univ. Press, 2007.

2 JiříAdámek, JiříRosický, and Enrico Vitale. *Algebraic Theories*. Cambridge Univ. Press, 2011.

3 Marco Bernardo and Stefania Botta. A survey of modal logics characterising behavioural equivalences for non-deterministic and stochastic systems. *Math. Struct. Comput. Sci.*, 18:29–55, 2008.

4 Nick Bezhanishvili and Silvio Ghilardi. The bounded proof property via step algebras and step frames. *Ann. Pure Appl. Logic*, 165:1832–1863, 2014.

5 Marcello Bonsangue, Stefan Milius, and Alexandra Silva. Sound and complete axiomatizations of coalgebraic language equivalence. *ACM Trans. Comput. Log.*, 14, 2013.

6 Ivan Christoff. Testing equivalences and fully abstract models for probabilistic processes. In *Theories of Concurrency, CONCUR 1990*, volume 458 of *LNCS*, pages 126–140. Springer, 1990.

7 Corina Cîrstea. A coalgebraic approach to linear-time logics. In *Foundations of Software Science and Computation Structures, FoSSaCS 2014*, volume 8412 of *LNCS*, pages 426–440. Springer, 2014.

8 Corina Cîrstea. Canonical coalgebraic linear time logics. In *Proc. CALCO*, 2015. This volume.

9 Silvio Ghilardi. An algebraic theory of normal forms. *Ann. Pure Appl. Logic*, 71:189–245, 1995.

10 Valentin Goranko and Martin Otto. Model theory of modal logic. In P. Blackburn, J. van Benthem, and F. Wolter, editors, *Handbook of Modal Logic*, pages 249–329. Elsevier, 2006.

11 Daniel Gorín and Lutz Schröder. Simulations and bisimulations for coalgebraic modal logics. In *Algebra and Coalgebra in Computer Science, CALCO 2013*, volume 8089 of *LNCS*, pages 253–266. Springer, 2013.

12 Ichiro Hasuo, Bart Jacobs, and Ana Sokolova. Generic trace semantics via coinduction. *Log. Meth. Comput. Sci.*, 3, 2007.

13 Antony Hoare. *Communicating sequential processes.* Prentice Hall, 1985.

14 Bart Jacobs, Alexandra Silva, and Ana Sokolova. Trace semantics via determinization. In *Coalgebraic Methods in Computer Science, CMCS 2012*, volume 7399 of *LNCS*, pages 109–129. Springer, 2012.

15 Peter Johnstone. Adjoint lifting theorems for categories of algebras. *Bull. London Math. Soc.*, 7:294–297, 1975.

16 Shin-ya Katsumata. Parametric effect monads and semantics of effect systems. In *Principles of Programming Languages, POPL 2014*, pages 633–646. ACM, 2014.

17 Christian Kissig and Alexander Kurz. Generic trace logics. arXiv preprint 1103.3239, 2011.

18 Bartek Klin and Juriaan Rot. Coalgebraic trace semantics via forgetful logics. In *Foundations of Software Science and Computation Structures, FoSSaCS'15*, 2015.

19 Alexander Kurz, Stefan Milius, Dirk Pattinson, and Lutz Schröder. Simplified coalgebraic trace equivalence. In *Software, Services, and Systems*, volume 8950 of *LNCS*, pages 75–90. Springer, 2015.

20 A. Mazurkiewicz. *Concurrent Program Schemes and Their Interpretation.* Aarhus University, Comp. Sci. Depart., DAIMI PB-78, July 1977.

21 P.-A. Mellies. The parametric continuation monad. Preprint, 2015.

22 Eugenio Moggi. Notions of computation and monads. *Inf. Comput.*, 93:55–92, 1991.

23 Philip Mulry. Lifting theorems for Kleisli categories. In *Mathematical Foundations of Programming Semantics, MFPS 1993*, volume 802 of *LNCS*, pages 304–319. Springer, 1994.

24 D. Pattinson. Expressive logics for coalgebras via terminal sequence induction. *Notre Dame J. Formal Logic*, 45:19–33, 2004.

25 J. Rutten. Universal coalgebra: A theory of systems. *Theor. Comput. Sci.*, 249:3–80, 2000.

26 L. Schröder and D. Pattinson. PSPACE bounds for rank-1 modal logics. *ACM Trans. Comput. Log.*, 10:13:1–13:33, 2009.

27 L. Schröder and D. Pattinson. Rank-1 modal logics are coalgebraic. *J. Log. Comput.*, 20, 2010.

28 Lutz Schröder. Expressivity of coalgebraic modal logic: The limits and beyond. *Theor. Comput. Sci.*, 390:230–247, 2008.

29 Lutz Schröder and Dirk Pattinson. Coalgebraic correspondence theory. In *Foundations of Software Structures and Computer Science, FoSSaCS 2010*, volume 6014 of *LNCS*, pages 328–342. Springer, 2010.

30 Alexandra Silva, Filippo Bonchi, Marcello Bonsangue, and Jan Rutten. Generalizing determinization from automata to coalgebras. *Log. Meth. Comput. Sci*, 9(1:9), 2013.

31 A. Smirnov. Graded monads and rings of polynomials. *J. Math. Sci.*, 151:3032–3051, 2008.

32 Sam Staton. Relating coalgebraic notions of bisimulation. *Log. Meth. Comput. Sci.*, 7, 2011.

33 Rob van Glabbeek. The linear time-branching time spectrum (extended abstract). In *Theories of Concurrency, CONCUR'90*, volume 458 of *LNCS*, pages 278–297. Springer, 1990.

34 James Worrell. On the final sequence of a finitary set functor. *Theor. Comput. Sci.*, 338:184–199, 2005.

Open System Categorical Quantum Semantics in Natural Language Processing

Robin Piedeleu[1], Dimitri Kartsaklis[2], Bob Coecke[1], and Mehrnoosh Sadrzadeh[2]

1 Department of Computer Science, University of Oxford
Parks Road, Oxford OX1 3QD, UK
{robin.piedeleu;bob.coecke}@cs.ox.ac.uk

2 School of Electronic Engineering and Computer Science, Queen Mary University of London
Mile End Road, London E1 4NS, UK
{d.kartsaklis;m.sadrzadeh}@qmul.ac.uk

Abstract

Originally inspired by categorical quantum mechanics (Abramsky and Coecke, LiCS'04), the categorical compositional distributional model of natural language meaning of Coecke, Sadrzadeh and Clark provides a conceptually motivated procedure to compute the meaning of a sentence, given its grammatical structure within a Lambek pregroup and a vectorial representation of the meaning of its parts. Moreover, just like CQM allows for varying the model in which we interpret quantum axioms, one can also vary the model in which we interpret word meaning.

In this paper we show that further developments in categorical quantum mechanics are relevant to natural language processing too. Firstly, Selinger's CPM-construction allows for explicitly taking into account lexical ambiguity and distinguishing between the two inherently different notions of homonymy and polysemy. In terms of the model in which we interpret word meaning, this means a passage from the vector space model to density matrices. Despite this change of model, standard empirical methods for comparing meanings can be easily adopted, which we demonstrate by a small-scale experiment on real-world data. Secondly, commutative classical structures as well as their non-commutative counterparts that arise in the image of the CPM-construction allow for encoding relative pronouns, verbs and adjectives, and finally, iteration of the CPM-construction, something that has no counterpart in the quantum realm, enables one to accommodate both entailment and ambiguity.

1998 ACM Subject Classification I.2.7 Natural Language Processing

Keywords and phrases category theory, density matrices, distributional models, semantics

Digital Object Identifier 10.4230/LIPIcs.CALCO.2015.270

1 Introduction

Language serves to convey meaning. From this perspective, the ultimate and long-standing goal of any computational linguist is to capture and adequately represent the meaning of an utterance in a computer's memory. At word level, *distributional semantics* offers an effective way to achieve that goal; following the *distributional hypothesis* [11] which states that the meaning of a word is determined by its context, words are represented as vectors of co-occurrence statistics with all other words in the vocabulary. While models following this paradigm have been found very useful in a number of natural language processing tasks, they do not scale up to the level of phrases or sentences. This is due to the capacity of

6th International Conference on Algebra and Coalgebra in Computer Science (CALCO'15).
Editors: Lawrence S. Moss and Pawel Sobocinski; pp. 270–289
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

natural language to generate an infinite number of structures (phases and sentences) from finite means (words); no text corpus, regardless of its size, can provide reliable distributional statistics for a multi-word sentence. On the other hand, type-logical approaches conforming to the tradition of Lambek [16], Montague and other pioneers of language, are compositional and deal with the sentence at a more abstract level based on the syntactical rules that hold between the different text constituents, but in principle they do not provide a convincing model for word meaning.

The categorical compositional distributional model of Coecke, Sadrzadeh and Clark [7] addresses the challenge of combining these two orthogonal models of meaning in a unified setting. The model is based on the observation that a grammar expressed as a pregroup [15] shares the same structure with the category of finite dimensional vector spaces and linear maps, that of a *compact closed category* [14]. In principle, this offers a canonical way to express a grammatical derivation as a morphism that defines linear-algebraic manipulations between vector spaces, resulting in a sentence vector. The main characteristic of the model is that the grammatical type of a word determines the vector space in which it lives. Words with atomic types, such as nouns, are represented by vectors living in some basic vector space N; on the contrary, relational words such as verbs and adjectives live in tensor product spaces of higher order. An adjective, for example, is an element of $N \otimes N$, while a transitive verb lives in $N \otimes S \otimes N$. The relational tensors act on their argument by *tensor contraction*, a generalization of the familiar notion of matrix multiplication to higher order tensors.

Ambiguity is a dominant feature of language. At the lexical level, one can distinguish between two broad types of ambiguity: *homonymy* refers to cases in which, due to some historical accident, words that share exactly the same spelling and pronunciation are used to describe completely distinct concepts; such an example is ‘bank’, meaning a financial institution and a land alongside a river. On the other hand, the senses of a *polysemous* word are usually closely related with only small deviations between them; as an example, think of ‘bank’ again as a financial institution and the concrete building where that institution is accommodated. These two notions of ambiguity are inherently different; while a polysemous word still retains a certain level of semantic coherence, a homonymous word can be seen as an incoherent mixing due to coincidence. The issue of lexical ambiguity and the different levels of it is currently ignored from almost all attempts that aim to equip distributional models of meaning with compositionality.

The purpose of this paper is to provide the theoretical foundations for a compositional distributional model of meaning capable of explicitly dealing with lexical ambiguity. In the proposed model we exploit the observation that the compact closed structure on which the original model of Coecke et al. [7] was based provides an abstraction of the Hilbert space formulation used in the quantum theory, in terms of pure quantum states as vectors, which is known under the umbrella of categorical quantum mechanics [1]. In fact, the original model of Coecke et al. was itself greatly inspired by quantum theory, and in particular, by quantum protocols such as quantum teleportation. Importantly, vectors in a Hilbert space represent the states of a closed quantum system, also called *pure* states. Selinger’s *CPM-construction* [21], which maps any dagger compact closed category on another one, then adjoins *open system* states, also called *mixed states*. In the new model, these allow for a lack of knowledge on part of the system under consideration, which may be about an extended part of the quantum system, or uncertainty (read: ambiguity) regarding the preparation procedure.

The crucial distinction between homonymous and polysemous words is achieved as follows: while a polysemous word corresponds to a *pure* quantum state, a homonymous word is given by a *mixed* state that essentially embodies a probability distribution over all potential meanings of that word. Mathematically, a mixed states is expressed as a *density matrix*: a self-adjoint, positive semi-definite operator with trace one. The new formulation offers many opportunities for interesting and novel research. For instance, by exploiting the notion of *Von Neumann entropy* one can measure how ambiguity evolves from individual words to larger text constituents; we would expect that the level of ambiguity in word 'bank' is higher than that of the compound 'river bank'.

Furthermore, the richness of the new category in which the meanings of words now live offers interesting alternative design options. In the past, for example, Sadrzadeh, Kartsaklis and colleagues [19, 12] enriched the categorical compositional model with elements of classical processing, exploiting the fact that any basis of a finite-dimensional vector space induces a *commutative Frobenius algebra* over this space, which allows the uniform copying or deleting of the information relative to this basis [6]. As we will see in Sect. 4, the dagger compact closed categories arising from the CPM-construction also accommodate canonical non-commutative Frobenius algebras which have the potential to account for the non-commutativity of language.

Finally, we discuss how iterated application of the CPM-construction, which gives rise to states that have no interpretation in quantum theory, does have a natural application in natural language processing. It allows for simultaneous semantic representation of more than one language feature that can be represented by density matrices, for example, lexical entailment in conjunction with ambiguity.

Related work. The issue of lexical ambiguity in categorical compositional models of meaning has been previously experimentally investigated by Kartsaklis and Sadrzadeh [13], who present evidence that the introduction of an explicit disambiguation step on the word vectors prior to composition improves the performance of the models. Furthermore, the research presented here is not the only one that uses density matrices for linguistic purposes. Balkır [2] uses a form of density matrices in order to provide a similarity measure that can be used for evaluating hyponymy-hypernymy relations. In Sect. 5 we indicate how these two uses of density matrices can be merged into one. Finally, Blacoe et al. [3] describe a distributional (but not compositional) model of meaning based on density matrices created by grammatical dependencies.

2 Background

The field of *category theory* aims at identifying and studying connections between seemingly different forms of mathematical structures. A very representative example of its potency is the compositional categorical framework of Coecke et al. [7], which shows that a grammatical derivation defining the structure of a sentence is homomorphic to a linear-algebraic formula acting on a semantic space defined by a distributional model. The framework offers a concrete manifestation of the *rule-to-rule hypothesis* and a mathematical counterpart to the formal semantics perspective on language. As noted above, the main idea is based on the fact that both the type-logic of the model, a pregroup grammar, and the semantic category, namely **FHilb**, possess a compact-closed structure. Recall that a *compact closed category* is a monoidal category in which every object A has a left and right adjoint, denoted as A^l, A^r respectively, for which the following special morphisms exist:

$$\eta^l : I \to A \otimes A^l \qquad \eta^r : I \to A^r \otimes A \qquad \epsilon^l : A^l \otimes A \to I \qquad \epsilon^r : A \otimes A^r \to I \tag{1}$$

These maps need to satisfy certain conditions (known as *yanking equations*) which ensure that all relevant diagrams commute:

$$\begin{array}{ll} (1_A \otimes \epsilon^l_A) \circ (\eta^l_A \otimes 1_A) = 1_A & (\epsilon^r_A \otimes 1_A) \circ (1_A \otimes \eta^r_A) = 1_A \\ (\epsilon^l_A \otimes 1_{A^l}) \circ (1_{A^l} \otimes \eta^l_A) = 1_{A^l} & (1_{A^r} \otimes \epsilon^r_A) \circ (\eta^r_A \otimes 1_{A^r}) = 1_{A^r} \end{array} \tag{2}$$

Finally, the passage from syntax to semantics is carried out by a *strong monoidal functor* and, as a result, preserves the compact closed structure. Before we proceed to expand on the above constructions, we refer the reader to App. A for a brief introduction to the graphical calculus of monoidal categories which will be used throughout our exposition.

2.1 Pregroup grammars

A *pregroup algebra* [15] is a partially ordered monoid with unit 1, whose each element p has a left adjoint p^l and a right adjoint p^r, conforming to the following inequalities:

$$p^l \cdot p \leq 1 \leq p \cdot p^l \quad \text{and} \quad p \cdot p^r \leq 1 \leq p^r \cdot p \tag{3}$$

A *pregroup grammar* is a pregroup algebra freely generated over a set of basic types $\mathcal{B}$ including a designated end type and a type dictionary that assigns elements of the pregroup to the vocabulary of a language. For example, it is usually assumed that $\mathcal{B} = \{n, s\}$, where n is the type assigned to a noun or a well-formed noun phrase, while s is a designated type kept for a well-formed sentence. Atomic types can be combined in order to provide types for relational words; for example, an adjective has type $n \cdot n^l$, reflecting the fact that it is something that expects for a noun at its right-hand side in order to return another noun. Similarly, a transitive verb has type $n^r \cdot s \cdot n^l$, denoting something that expects two nouns (one at each side) in order to return a sentence. Based on (3), for this latter case the pregroup derivation gets the following form:

$$n \cdot (n^r \cdot s \cdot n^l) \cdot n = (n \cdot n^r) \cdot s \cdot (n^l \cdot n) \leq 1 \cdot s \cdot 1 \leq s \tag{4}$$

Let $\mathbf{C_F}$ denote the *free compact closed category* derived from the pregroup algebra of a pregroup grammar [18]; then, according to (1), the above type reduction corresponds to the morphism $\epsilon^r_n \cdot 1_s \cdot \epsilon^l_n : n \cdot n^r \cdot s \cdot n^l \cdot n \to s$ in $\mathbf{C_F}$.

2.2 From syntax to semantics

The type-logical approach presented in Sect. 2.1 is compositional, but unable to distinguish between words of the same type; even more importantly, the only information that a derivation such as the one in (4) can provide to us is whether the sentence is well-formed or not. Distributional models of meaning offer a solution to the first of these problems, by representing a word in terms of its distributional behaviour in a large corpus of text. While the actual methods for achieving this can vary (see App. D for a concrete implementation), the goal is always the same: to represent words as points of some metric space, where differences in semantic similarity can be detected and precisely quantified. The prime intuition is that words appearing in similar contexts must have a similar meaning [11]. The word vectors typically live in a highly dimensional semantic space with a fixed orthonormal basis, the elements of which correspond to content-bearing words. The values in the vector of a target word w_t express co-occurrence statistics extracted from some large corpus of text, showing how strongly w_t is associated with each one of the basis words. For a concise introduction to distributional models of meaning see [23].

We take $(\mathbf{FHilb}, \otimes)$, the category of finite dimensional Hilbert spaces and linear maps over the scalar field I, to be the semantic counterpart of $\mathbf{C_F}$ which, as we saw before, accommodates the grammar. **FHilb** is a *dagger compact closed* category (or, †-compact closed); that is, a *symmetric* compact closed category (so that $A^r \cong A^l = A^*$ for all A) equipped with an involutive contravariant functor $\dagger : \mathbf{FHilb} \to \mathbf{FHilb}$ that is the identity on objects. Concretely, in **FHilb**, for a morphism $f : A \to B$, its dagger $f^\dagger : B \to A$ is simply its adjoint. Furthermore, $\epsilon_A = \eta_A^\dagger \circ \sigma_{A^*,A}$ for all A.

Taking $|\psi\rangle$ and $|\phi\rangle$ to be two vectors in a Hilbert space $\mathcal{H}$, $\epsilon_A : A^* \otimes A \to I$ is the pairing $\epsilon_A(\langle\psi|, |\phi\rangle) = \langle\psi|(|\phi\rangle) = \langle\psi|\phi\rangle$ and $\eta_A = \epsilon_A^\dagger$. This allows the inner product to be categorically defined as $\langle\psi|\phi\rangle : I \xrightarrow{\psi} \mathcal{H} \xrightarrow{\phi^\dagger} I$. In practice it is often necessary to normalise in order to obtain the cosine of the angle between vectors as a measure of semantic similarity.

2.3 Quantizing the grammar

We now proceed to present a solution to the second problem posed above, that of providing a quantified semantic representation for a sentence by composing the representations of the words therein: in this paper we follow [17] and [12] and we achieve the transition from syntax to semantics via a *strong monoidal functor* $Q : \mathbf{C_F} \to \mathbf{FHilb}$ which can be shown to also preserve the compact structure so that $Q(p^l) = Q(p)^l$ and $Q(p^r) = Q(p)^r$ for p an object of $\mathbf{C_F}$. Since each object in **FHilb** is its own dual we also have $Q(p^l) \cong Q(p) \cong Q(p^r)$. Moreover, for basic types, we let $Q(n) = N$ and $Q(s) = S$. Note that since Q is strongly monoidal, complex types are mapped to tensor product of vector spaces:

$$Q(n \cdot n^r) = Q(n) \otimes Q(n^r) = N \otimes N \qquad Q(n^r \cdot s \cdot n^l) = Q(n^r) \otimes Q(s) \otimes Q(n^l) = N \otimes S \otimes N$$

Finally, each morphism in $\mathbf{C_F}$ is mapped to a linear map in **FHilb**. Equipped with such a functor, we can now define the meaning of a sentence as follows:

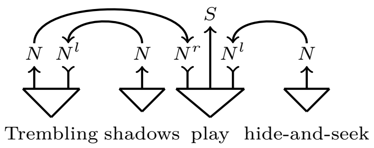
▶ **Definition 1.** Let $|w_i\rangle$ be a vector $I \to Q(p_i)$ corresponding to word w_i with type p_i in a sentence $w_1 w_2 \ldots w_n$. Given a type-reduction $\alpha : p_1 \cdot p_2 \cdot \ldots \cdot p_n \to s$, the meaning of the sentence is defined as:

$$|w_1 w_2 \ldots w_n\rangle := Q(\alpha)(|w_1\rangle \otimes \ldots \otimes |w_n\rangle) \tag{5}$$

Take as an example the sentence "Trembling shadows play hide-and-seek", with the standard types $n \cdot n^l$ and $n^r \cdot s \cdot n^l$ assigned to adjectives and verbs, respectively. Then the adjective 'trembling' will be a morphism $I \to Q(n \cdot n^l) = I \to N \otimes N$, that is, a state in the tensor product space $N \otimes N$. Note that this matrix defines a linear map $N \to N$, an interpretation that is fully aligned with the formal semantics perspective: an adjective is a function that takes a noun as input and returns a modified version of it. Similarly, the verb 'play' lives in $N \otimes S \otimes N$ or, equivalently, is a bi-linear map $N \otimes N \to S$ (with a subject and an object as arguments) which returns a sentence. In contrast to those two relational words, the nouns 'shadows' and 'hide-and-seek' are plain vectors in N. The syntax of the sentence conforms to the following type reduction:

$$(\epsilon_n^r \cdot 1_s) \circ (1_n \cdot \epsilon_n^l \cdot 1_{n^r} \cdot 1_s \cdot \epsilon_n^l) : n \cdot n^l \cdot n \cdot n^r \cdot s \cdot n^l \cdot n \to s \tag{6}$$

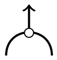
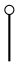
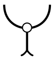
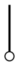
which, when transferred to **FHilb** via Q, yields the following diagrammatic derivation:

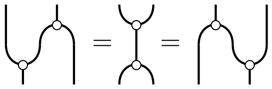


(7)

2.4 Using Frobenius algebras in language

Compact closed categories on their own do not have much structure. The expressive power of these categories can be increased using *Frobenius algebras*. Recall from [4] that a Frobenius algebra in a monoidal category is a quintuple $(A, \Delta, \iota, \mu, \zeta)$ such that:

- (A, μ, ζ) is a monoid, that is we have $\mu : A \otimes A \to A$  and $\zeta : I \to A$  satisfying associativity and unit conditions,
- (A, Δ, ι) is a co-monoid, so that $\Delta : A \to A \otimes A$  and $\iota : A \to I$  satisfy co-associativity and co-unit conditions;
- furthermore, Δ and μ adhere to the following *Frobenius condition*:

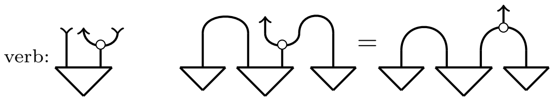


(8)

In a monoidal †-category, a †-Frobenius algebra is a Frobenius algebra whose co-monoid is adjoint to the monoid. As shown in [6], every finite dimensional Hilbert space $\mathcal{H}$ with orthonormal basis $\{|i\rangle\}$ has a †-Frobenius algebra associated to it, the co-multiplication and multiplication of which correspond to uniformly *copying* and *uncopying* the basis as follows:

$$\Delta :: |i\rangle \mapsto |i\rangle \otimes |i\rangle \qquad \iota :: |i\rangle \mapsto 1 \qquad \mu :: |i\rangle \otimes |j\rangle \mapsto \delta_{ij}|i\rangle := \begin{cases} |i\rangle & i = j \\ |0\rangle & i \neq j \end{cases} \qquad \zeta :: 1 \mapsto \sum_i |i\rangle$$

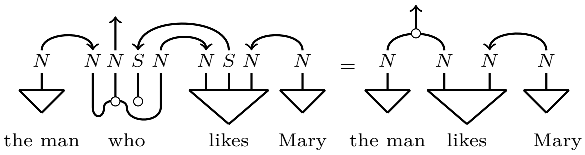
Abstractly, this enables us to copy and delete the (classical) information relative to the given basis. Concretely, the copying Δ-map amounts to encoding faithfully the components of a vector in $\mathcal{H}$ as the diagonal elements of a matrix in $\mathcal{H} \otimes \mathcal{H}$, while the "uncopying" operation μ picks out the diagonal elements of a matrix and returns them as a vector in $\mathcal{H}$. Kartsaklis et al. [12] use the Frobenius co-multiplication in order to faithfully encode tensors of lower order to higher order ones, thus restoring the proper functorial relation. An adjective, for example, is given as $\Delta(\sum_i |noun_i\rangle)$, where $|noun_i\rangle$ is a noun modified by the specific adjective in a training corpus. Furthermore, given a transitive verb constructed as $|verb\rangle = \sum_i |subj_i\rangle \otimes |obj_i\rangle$ [10], we can encode it to a tensor in $\mathcal{H} \otimes \mathcal{H} \otimes \mathcal{H}$ by either copying the row dimension (responsible for the interaction of the verb with the subject noun) or the column dimension (responsible for the interaction with the object). For the latter case, referred to by Copy-Object, the composition becomes as follows:



(9)

The composition for the case of copying the subject dimension proceeds similarly on the left-hand side. In practice, empirical work has shown that objects have stronger influence on the meaning of a transitive sentence than subjects [12], which suggests that the Frobenius structure of the Copy-Object approach is a more effective model of sentential compositionality.

Finally, Sadrzadeh et al. [19] exploit the abilities of Frobenius algebras in order to model relative pronouns. Specifically, *copying* is used in conjunction with *deleting* in order to allow the head noun of a relative clause to interact with its modifier verb phrase from the far left-hand side of the clause to its right-hand side. For the case of a relative clause modifying a subject this is achieved as follows:



(10)

3 Encoding ambiguity

The previous compositional model relies on a strong monoidal functor from a compact closed category, representing syntax, to **FHilb**, modelling a form of distributional semantics. In this section we will modify the functor to a new codomain category. To achieve our goal, we will explore a categorical construction, inspired from quantum physics and originally due to Selinger [21], in the context of the categorical model of meaning developed in the previous sections.

3.1 Mixing in FHilb

Although seemingly unrelated, quantum mechanics and linguistics share a common link through the framework of †-compact closed categories, an abstraction of the Hilbert space formulation, and have been used in the past [1] to provide structural proofs for a class of quantum protocols, essentially recasting the vector space semantics of quantum mechanics in a more abstract way. Shifting the perspective to the field of linguistics, we saw how the same formalism proposes a description of the semantic interactions of words at the sentence level. Here we make the connection between the two fields even more explicit, taking advantage of the fact that the ultimate purpose of quantum mechanics is to deal with uncertainty – and this is essentially what we need to achieve here in the context of language.

We start by observing that, in quantum physics, the Hilbert space model is insufficient to incorporate the epistemic state of the observer in its formalism: what if one does not have knowledge of a quantum system's initial state and can only attribute a probability distribution to a set of possible states? The answer is by considering a *statistical ensemble* of pure states: for example, one may assign a $\frac{1}{2}$ probability that the state vector of a system is $|\psi_1\rangle$ and a $\frac{1}{2}$ probability that it is in state $|\psi_2\rangle$. We say that this system is in a *mixed state*. In the Hilbert space setting, such a state cannot be represented as a vector. In fact, any normalised sum of pure states is again a pure state (by the vector space structure). Note that the state $(\psi_1 + \psi_2)/\sqrt{2}$ is a *quantum superposition* and not the mathematical representation of the mixed state above.

This situation is similar to the issue we face when trying to model ambiguity in distributional semantics: given two different meanings of a homonymous word and their relative weights (given as probabilities), simply looking at the convex composition of the associated vectors collapses the ambiguous meaning to a single vector, thereby fusing together the two senses of the word. The mathematical response to this problem is to move the focus away from states in a Hilbert space to a specific kind of operators on the same space: more specifically, to *density operators*, i.e., positive semi-definite, self-adjoint operators of trace one. The density operator formalism is our means to express a probability distribution over the potential meanings of a homonymous word in a distributional model (see App. C for a more detailed linguistic intuition). We formally define this as follows:

▶ **Definition 2.** Let a distributional model be given in the form of a Hilbert space M, in which every word w_t is represented by a statistical ensemble $\{(p_i, |w_t^i\rangle)\}_i$ – where $|w_t^i\rangle$ is a vector corresponding to a specific unambiguous meaning of the word that can occur with probability p_i. The distributional meaning of the word is defined as:

$$\rho(w_t) = \sum_i p_i |w_t^i\rangle\langle w_t^i| \tag{11}$$

Note that for the case of a non-homonymous word, the above formula reduces to $|w_t\rangle\langle w_t|$, with $|w_t\rangle$ corresponding to the state vector assigned to w_t. Now, if mixed states are density operators, we need a notion of morphism that preserves this structure, i.e., that maps states to states. In the Hilbert space model, the morphisms were simply linear maps. The corresponding notion in the mixed setting is that of *completely positive maps*, that is, positive maps that respect the monoidal structure of the underlying category.

To constitute a compositional model of meaning, our construction also needs to respect our stated goals: specifically, the category of operator spaces and completely positive maps must be a †-compact closed category; furthermore, we need to identify the morphism that plays the part of the Frobenius algebra of the previous model. We start working towards these goals by describing a construction that builds a similar category, not only from **FHilb**, but, more abstractly, from any †-compact closed category.

3.2 Doubling and complete positivity

The category that we are going to build was originally introduced by Selinger [21] as a generalisation of the corresponding construction on Hilbert spaces. Conceptually, it corresponds to shifting the focus away from vectors or morphisms of the form $I \to A$ to operators on the same space or morphisms of type $A \to A$. We will formalise this idea by first introducing the category $\mathbf{D}(\mathcal{C})$ on a compact closed category $\mathcal{C}$, which can be perhaps better understood in its diagrammatic form as a *doubling* of the wires. In this context, we obtain a duality between states of $\mathbf{D}(\mathcal{C})$ and operators of $\mathcal{C}$, pictured by simple wire manipulations. As we will see, $\mathbf{D}(\mathcal{C})$ retains the compact closedness of $\mathcal{C}$ and is therefore a viable candidate for a semantic category in our compositional model of meaning. However, at this stage, states of $\mathbf{D}(\mathcal{C})$ do not yet admit a clear interpretation in terms of mixing. This is why we need to introduce the notion of completely positive morphisms, of which positive operators on a Hilbert space (mixed states in quantum mechanics) are a special case. This will allow us later to define the subcategory $\mathbf{CPM}(\mathcal{C})$ of $\mathbf{D}(\mathcal{C})$.

3.2.1 The D construction (doubling)

First, given a †-compact closed category[1] $\mathcal{C}$ we define:

▶ **Definition 3.** The category $\mathbf{D}(\mathcal{C})$ with

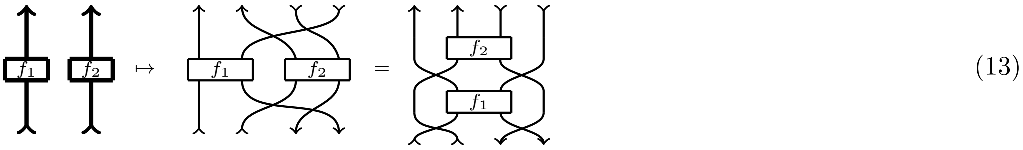
- the same objects as $\mathcal{C}$;
- morphisms between objects A and B of $\mathbf{D}(\mathcal{C})$ are morphisms $A \otimes A^* \to B \otimes B^*$ of $\mathcal{C}$.
- composition and dagger are inherited from $\mathcal{C}$ via the embedding $E : \mathbf{D}(\mathcal{C}) \hookrightarrow \mathcal{C}$ defined by $A \mapsto A \otimes A^*$ on objects and $f \mapsto f$ on morphisms.

In addition, we can endow the category $\mathbf{D}(\mathcal{C})$ of a monoidal structure by defining the tensor $\otimes_{\mathbf{D}}$ as $A \otimes_{\mathbf{D}} B = A \otimes B$ on objects A and B, and for morphisms $f_1 : A \otimes A^* \to B \otimes B^*$ and $f_2 : C \otimes C^* \to D \otimes D^*$, by:

$$f_1 \otimes_{\mathbf{D}} f_2 : A \otimes C \otimes C^* \otimes A^* \xrightarrow{\cong} A \otimes A^* \otimes C \otimes C^* \\ \xrightarrow{f_1 \otimes f_2} B \otimes B^* \otimes D \otimes D^* \xrightarrow{\cong} B \otimes D \otimes D^* \otimes B^* \quad (12)$$

[1] The construction works on any monoidal category with a dagger, i.e., an involution, but we will not need the additional generality.

Or graphically by,



where the arrow $\mapsto$ represents the functor E and we use the convention of depicting morphisms in $\mathbf{D}(\mathcal{C})$ with thick wires and boxes to avoid confusion. Note that the intuitive alternative of simply juxtaposing the two morphisms as we would in $\mathcal{C}$ fails to produce a completely positive morphism in general, as will become clearer when we define completely positivity in this context. This category carries all the required structure. We refer the reader to [21] for a proof of the following:

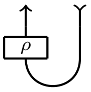
▶ **Proposition 4.** *The category $\mathbf{D}(\mathcal{C})$ inherits a $\dagger$-compact closed structure from $\mathcal{C}$ via the strict monoidal functor $M : \mathcal{C} \to \mathbf{D}(\mathcal{C})$ defined inductively by*

$$\begin{cases} f_1 \otimes f_2 & \mapsto M(f_1) \otimes_{\mathbf{D}} M(f_2) & ; \\ A & \mapsto A & \textit{on objects}; \\ f & \mapsto f \otimes f_* & \textit{on morphisms.} \end{cases}$$

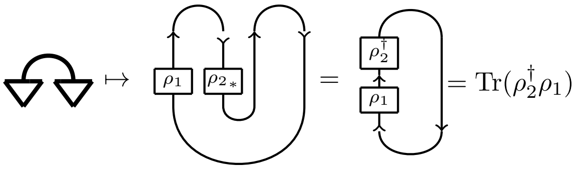
where $f_* = (f^\dagger)^*$ by definition.

The functor M shows that we are not losing any expressive power since unambiguous words (represented as maps of $\mathcal{C}$) still admit a faithful representation in doubled form. For reference, the reader can find in App. B a dictionary that translates useful diagrams from one category to the other. Now, notice that we have a bijective correspondence between states of $\mathbf{D}(\mathcal{C})$, i.e., morphisms $I \to A$ and operators on A in $\mathcal{C}$. Explicitly, the map $\mathcal{C}(A, A) \to \mathcal{C}(I, A \otimes A^*)$ is, for an operator $\rho : A \to A$,

$$\rho \mapsto \ulcorner \rho \urcorner = (\rho \otimes 1_{A^*}) \circ \eta_{A^*} = \tag{14}$$



that is easily seen to be an isomorphism by bending back the rightmost wire (by application of the yanking equations (2)). In the special case of states, the generalised inner product generated by the dagger functor can be computed in terms of the canonical trace induced by the compact closed structure (and reduces to the usual inner product on a space of operators in **FHilb**):

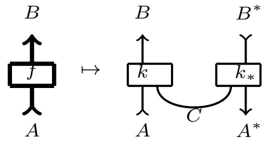


(15)

3.2.2 The CPM construction (complete positivity)

▶ **Definition 5.** A morphism $f : A \to B$ of $\mathbf{D}(\mathcal{C})$ is completely positive if there exists an object C and a morphism $k : C \otimes A \to B$, in $\mathcal{C}$, such that f embeds in $\mathcal{C}$ as $(k \otimes k_*) \circ (1_A \otimes \eta_{C^*} \otimes 1_{A^*})$

or, pictorially,

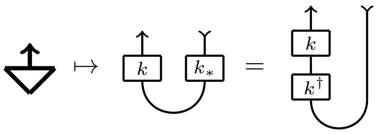


(16)

From this last representation, we easily see that the composition of two completely positive maps is completely positive. Similarly, the tensor product of two completely positive maps is completely positive. Therefore, we can define:

▶ **Definition 6.** The category $\mathbf{CPM}(\mathcal{C})$ is the subcategory of $\mathbf{D}(\mathcal{C})$ whose objects are the same and morphisms are completely positive maps.

$\mathbf{CPM}(\mathcal{C})$ is monoidal and $\otimes_{\mathbf{CPM}} = \otimes_{\mathbf{D}}$. We easily recover the usual notion of positive operator from this definition:



$= \ulcorner k \circ k^{\dagger} \urcorner$ (17)

with pure states corresponding to the disconnected case. Finally, from Def. 5 it is clear that, for a morphism f of $\mathcal{C}$, $M(f) = f \otimes f_*$ is completely positive. Thus,

▶ **Proposition 7.** *M factors through the embedding $I : \mathbf{CPM}(\mathcal{C}) \hookrightarrow \mathbf{D}(\mathcal{C})$, i.e., there exists a strictly monoidal functor $\tilde{M} : \mathcal{C} \to \mathbf{CPM}(\mathcal{C})$ such that $M = I\tilde{M}$.*

3.3 Categorical model of meaning: Reprise

We are now ready to put together all the concepts introduced above in the context of a compositional model of meaning. Our aim in this section is to reinterpret the previous model of [7] as a functor from a compact closed grammar to the category $\mathbf{CPM}(\mathcal{C})$, *for any compact closed category* $\mathcal{C}$. Given semantics in the form of a strong monoidal functor $Q : \mathbf{C_F} \to \mathcal{C}$, our model of meaning is defined by the composition:

$$\tilde{M}Q : \mathbf{C_F} \to \mathcal{C} \to \mathbf{CPM}(\mathcal{C}) \tag{18}$$

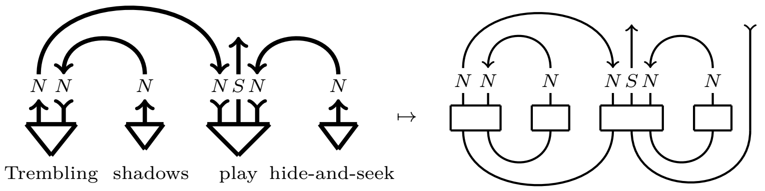
Since $\tilde{M}$ sends an object A to the same A in $\mathbf{CPM}(\mathcal{C})$, the mapping of atomic types, their duals and relational types of the grammar occurs in exactly the same fashion as in the previous model. Furthermore, note that Q is strongly monoidal and $\tilde{M}$ is strictly monoidal, so the resulting functor is strongly monoidal and, in particular, preserves the compact structure. Thus, we can perform type reductions in $\mathbf{CPM}(\mathcal{C})$ according to the grammatical structure dictated by the category $\mathbf{C_F}$.

Note that we have deliberately abstracted the model to highlight its richness – the category $\mathcal{C}$ could be any compact closed category: **FHilb**, the category **Rel** of sets and relations (in which case we recover a form of Montague semantics) or, as we will see in Sect. 5, even another iteration of the **CPM** construction.

▶ **Definition 8.** Let $\rho(w_i)$ be a meaning state $I \to \tilde{M}Q(p_i)$ corresponding to word w_i with type p_i in a sentence $w_1 \ldots w_n$. Given a type-reduction $\alpha : p_1 \cdot \ldots \cdot p_n \to s$, the meaning of the sentence is defined as:

$$\rho(w_1 \ldots w_n) := \tilde{M}Q(\alpha)\big(\rho(w_1) \otimes_{\mathbf{CPM}} \ldots \otimes_{\mathbf{CPM}} \rho(w_n)\big)$$

For example, assigning density matrix representations to the words in the previous example sentence "trembling shadows play hide and seek", we obtain the following meaning representation:

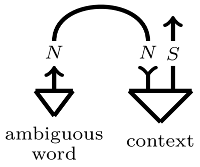


Diagrammatically, it is clear that in the new setting the partial trace implements meaning composition. Note that diagrams as the above illustrate the flow of ambiguity or information between words. The question of how does ambiguity evolve when composing words to form sentences is very hard to answer precisely in full generality. The key message is that (unambiguous) meaning emerges in the interaction of a word with its context, through the wires. This process of disambiguation is perhaps better understood by studying very simple examples, as we are going to do in the next section.

3.4 Introducing ambiguity in formal semantics

Here, we will work in the category **CPM**(**Rel**). We recall that **Rel** is the †-compact category of sets and relations. The tensor product is the Cartesian product and the dagger associates to a relation its opposite. Let our sentence set be $S = \{true, false\}$. In **Rel**, this means that we are only interested in the truth of a sentence, as in Montague semantics. In this context, nouns are subsets of attributes. Given a context to which we pass the meaning of a word, the meaning of the resulting sentence can be either $|false\rangle$, $|true\rangle$ or $|false\rangle + |true\rangle$, the latter representing superposition, i.e., the case for which the context is insufficient to determine the truth of all the attributes of the word (classically, this can be identified with $false$).

On the other hand, in the internal logic of **CPM**(**Rel**), mixing adds a second dimension that can be interpreted as ambiguous meaning, regardless of truth. The possible values are:



$$= \begin{cases} |\mathit{true}\rangle\langle\mathit{true}|, \\ |\mathit{false}\rangle\langle\mathit{false}|, \\ (|\mathit{true}\rangle + |\mathit{false}\rangle)(\langle\mathit{true}| + \langle\mathit{false}|), \\ 1_S \end{cases}$$

where the identity on S represents ambiguity. Note that we use Dirac notation in **Rel** rather than set theoretic union and cartesian product, since elements in finite sets can be seen as basis vectors of free modules over the semi-ring of Booleans; a binary relation can be expressed as an adjacency matrix. The trace of a square matrix picks out the elements for which the corresponding relation is reflexive.

Consider the phrase 'queen rules'. We allow a few highly simplifying assumptions: first, we restrict our set of nouns to the rather peculiar 'Freddy Mercury', 'Brian May', 'Elisabeth II', 'chess', 'England' and the empty word ϵ. Moreover, we consider the verb 'rule', supposed to have the following unambiguous meaning:

$$|\mathit{rule}\rangle = |\mathit{band}\rangle \otimes |\mathit{true}\rangle \otimes |\epsilon\rangle + |\mathit{chess}\rangle \otimes |\mathit{false}\rangle \otimes |\epsilon\rangle + |\mathit{elisabeth}\rangle \otimes |\mathit{true}\rangle \otimes |\mathit{england}\rangle$$

with the obvious $|\mathit{band}\rangle = |\mathit{freddy}\rangle + |\mathit{brian}\rangle$. This definition reflects the fact that a band can rule (understand "be the best") as well as a monarch. Finally, the ambiguous meaning of

Table 1 Computing entropy for nouns modified by relative clauses and adjectives.

Relative Clauses				Adjectives		
noun: v_1/v_2	*noun*	n that v_1	n that v_2	adj_1/adj_2	adj_1 n	adj_2 n
organ: enchant/ache	0.18	0.11	0.08	music/body	0.10	0.13
vessel: swell/sail	0.25	0.16	**0.01**	blood/naval	0.05	0.07
queen: fly/rule	0.28	0.14	0.16	fair/chess	0.05	0.16
nail: gleam/grow	0.19	0.06	0.14	rusty/finger	**0.04**	0.11
bank: overflow/loan	0.21	0.19	0.18	water/financial	0.20	0.16

'queen' is represented by the following operator:

$$\rho(queen) = |elisabeth\rangle\langle elisabeth| + |band\rangle\langle band| + |chess\rangle\langle chess|$$

A computation of the meaning of the sentence in algebraic form yields, $\mathrm{Tr}_N(|rule\rangle\langle rule| \circ (\mathrm{Tr}_{N'}(\rho(queen)) \otimes 1'_N)) = 1_S$. In other words, the meaning of the sentence is neither *true* nor *false* but still ambiguous. This is because the context that we pass to 'queen' is insufficient to disambiguate it (the band *or* the monarch can rule). Now, if we consider 'queen rules England', the only matching pattern in the definition of $|rule\rangle$ is $|elisabeth\rangle$ which corresponds to a *unique* and therefore unambiguous meaning of $\rho(queen)$. Hence, a similar calculation yields $\mathrm{Tr}_N(|rule\rangle\langle rule| \circ (\mathrm{Tr}_{N'}(\rho(queen)) \otimes |england\rangle\langle england|)) = |true\rangle\langle true|$ and the sentence is not only true but unambiguous. In this case, the context was sufficient to disambiguate the meaning of the word 'queen'.

3.5 Measuring ambiguity with real data

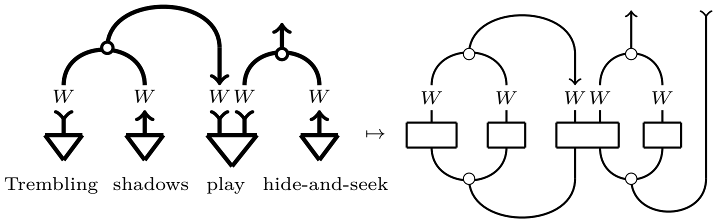
While a large-scale experiment is out of the scope of this paper, in this section we present some preliminary witnessing results that showcase the potential of the model. Using 2000-dimensional meaning vectors created by the procedure described in App. D, we show how ambiguity evolves for five ambiguous nouns when they are modified by an adjective or a relative clause. For example, 'nail' can appear as 'rusty nail' or 'nail that grows'; in both cases the modifier resolves part of the ambiguity, so we expect that the entropy of the larger compound would be lower than that of the original ambiguous noun. Both types of composition use the Frobenius framework described in Sect. 2.4; We further remind that for a density matrix ρ with eigen-decomposition $\rho = \sum e_i|e_i\rangle\langle e_i|$, Von Neumann entropy is given as $S(\rho) = -\,\mathrm{Tr}(\rho \ln \rho) = -\sum_i e_i \ln e_i$.

As Table 1 shows, the entropy of the compounds is always lower than that of the ambiguous noun. Even more interestingly, for some cases (e.g 'vessel that sails') the context is so strong that is capable to almost *purify* the meaning of the noun. This demonstrates an important aspect of the proposed model: *disambiguation = purification.*

3.6 Flow of information with †-Frobenius algebras

In the above examples we used the assumption that a verb tensor had been faithfully constructed according to its grammatical type. However, as we saw in Sect. 2.4, concrete constructions might yield operators on a space of tensor order lower than the space to which the functor $\tilde{M}Q$ maps their grammatical type. As before, †-Frobenius algebras can be used to solve this type mismatch and encode the information carried by an operator into tensors of higher order. Specifically, we will first consider the †-Frobenius algebra whose copying map is $M(\Delta)$ and whose deleting map is $M(\iota)$, as doubling preserves both operations.

In addition, the monoid operation is clearly completely positive. In more concrete terms, the monoid operation is precisely the point-wise (sometimes called Hadamard) product of matrices. Assuming we have a distributional model in the form of a vector space W with a distinguished basis and density matrices on W (to represent the meaning of our nouns and adjectives) and on $W \otimes W$ (for verbs), our example sentence is given by:



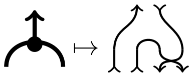
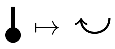
4 Non-commutativity

If the last section was concerned with applications of the CPM-construction to model ambiguity, here we discuss the role of the D-construction for the same purpose. Frobenius algebras on objects of $\mathbf{D}(\mathcal{C})$ are not necessarily commutative and thus their associated monoid is not a completely positive morphism. In the quantum physical literature, non-completely positive maps are not usually considered since they are not physically realisable. However, in linguistics, free from these constraints, we could theoretically venture outside of the subcategory $\mathbf{CPM}(\mathcal{C})$, deep into $\mathbf{D}(\mathcal{C})$.

For example, Coecke, Heunen and Kissinger [8] introduced the category $\mathbf{CP}^*(\mathcal{C})$ of †-Frobenius algebras (with additional technical conditions) and completely positive maps, over an arbitrary †-compact category $\mathcal{C}$, in order to study the interaction of classical and quantum systems in a single categorical setting: classical systems are precisely the commutative algebras and completely positive maps are quantum channels, that is, physically realisable processes between systems. Interestingly, in accordance with the content of the no-broadcasting theorem for quantum systems the multiplication of a commutative algebra is a completely positive morphism while the multiplication of a non-commutative algebra is not. It is clear that the meaning composition of words in a sentence is only commutative in exceptional cases; the non commutativity of the grammatical structure reflects this. However, in earlier methods of composition, this complexity was lost in translation when passing to semantics.

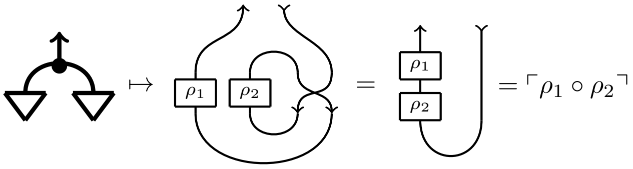
With linguistic applications in mind, the $\mathbf{CP}^*$ construction suggests various ways of composing the meaning of words, each corresponding to a specific Frobenius algebra operation. Conceptually, this idea makes sense since a verb does not compose with its subject in the same way that an adjective composes with the noun phrase to which it applies. The various ways of composing words may also offer a theoretical base for the introduction of logic in distributional models of natural language. This is where the richness of $\mathbf{D}(\mathcal{C})$ reveals itself: algebras in this category are more complex and, in particular, allow us to study the action of non-commutative structures – a topic of great interest to formal linguistics where the interaction of words is highly non-commutative. Hereafter we introduce a non-commutative †-Frobenius algebra that is not the doubled image of any algebra in $\mathcal{C}$.

▶ **Definition 9.** For every object A of $D(\mathcal{C})$, the morphisms of $D(\mathcal{C})$, $\mu : A \otimes_{\mathbf{D}} A \to A$ and $\iota : I \to A$ defined by the following diagrams in $\mathcal{C}$:

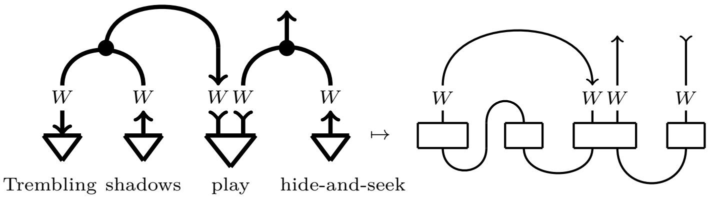
 $= (1_A \otimes \epsilon_A \otimes 1_{A^*}) \circ (1_{A\otimes A} \otimes \sigma_{A,A^*})$  $= \eta_{A^*}$

are the multiplication and unit of a †-Frobenius algebra $\mathcal{F}_{\mathbf{D}}$ – where σ is the natural swap isomorphism in $\mathcal{C}$.

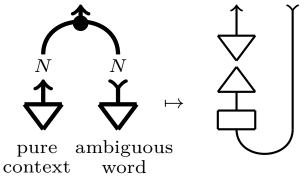
Proof that the above construction is indeed a †-Frobenius algebra can be found in [9]. The action of the Frobenius multiplication μ on states $I \to A$ of $\mathbf{D}(\mathcal{C})$ is particularly interesting; in fact, it implements the composition of operators of $\mathcal{C}$, in $\mathbf{D}(\mathcal{C})$:



The meaning of the "trembling shadows..." sentence using the algebra $\mathcal{F}_{\mathbf{D}}$ becomes:

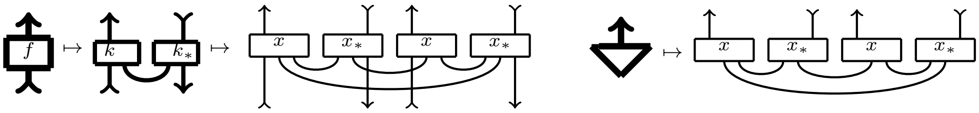


How does composition with the new algebra affect the flow of ambiguity in the simple case of an ambiguous word to which we pass an unambiguous context? Given a projection onto a one-dimensional subspace $|w\rangle\langle w|$ and a density operator ρ, the composition $|w\rangle\langle w|\rho$ is a (*not necessarily orthogonal*) projection. In a sense, the meaning of the pure word determines that of the ambiguous word as evidenced by the disconnected topology of the following diagram:

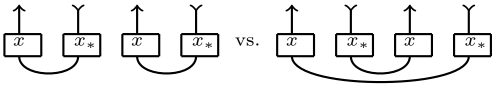


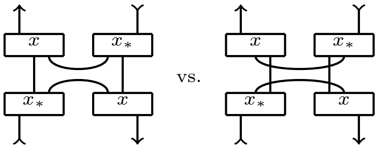
5 Adding lexical entailment

We now demonstrate the advantage of the fact that the CPM-construction is an abstract construction, and hence can be applied to any suitable (i.e. living in a †-compact closed category) model of word meaning. Besides ambiguity, another feature of language which is not captured by the distributional model is the fact that the meaning of one word (= *hypernym*) generalises that of another word (= *hyponym*). This points at a partial ordering of word meanings. For example, 'painter' generalises 'Brueghel'. Density matrices can be endowed with a partial ordering which could play that role, e.g. the *Bayesian ordering* [5]. This raises the question of how to accommodate both features together in a model of natural language meaning. Since $\mathbf{CPM}(\mathcal{C})$ is always †-compact closed, a canonical solution is obtained by iterating the CPM-construction:



Given a word/phrase/sentence meaning as above, lack of any ambiguity or generality correspond to distinct diagrams, respectively (left), and can be measured by taking the von Neumann entropy of different operators (right):





6 Conclusion and future work

In this paper we detailed a compositional distributional model of meaning capable of explicitly handling lexical ambiguity. We discussed its theoretical properties and demonstrated its potential for real-world natural language processing tasks by a small-scale experiment. A large-scale evaluation will be our challenging next step, aiming to provide empirical evidence regarding the effectiveness of the model in general and the performance of the different Frobenius algebras in particular. On the theoretical side, the logic of ambiguity in **CPM**(**Rel**), the non-commutative features of the D-construction as well as further exploration of nested levels of CPM, each deserve a separate treatment. In addition, one important weakness of distributional models is the representation of words that serve a purely logical role, like logical connectives or negation. Density operators support a form of logic whose distributional and compositional properties could be examined, potentially providing a solution to this long-standing problem of compositional distributional models.

References

1 Samson Abramsky and Bob Coecke. A categorical semantics of quantum protocols. In *19th Annual IEEE Symposium on Logic in Computer Science*, pages 415–425, 2004.

2 Esma Balkır. Using density matrices in a compositional distributional model of meaning. Master's thesis, University of Oxford, 2014.

3 William Blacoe, Elham Kashefi, and Mirella Lapata. A quantum-theoretic approach to distributional semantics. In *Proceedings of NACL 2013*, pages 847–857. Association for Computational Linguistics, June 2013.

4 A. Carboni and R.F.C. Walters. Cartesian Bicategories I. *Journal of Pure and Applied Algebra*, 49, 1987.

5 B. Coecke and K. Martin. A partial order on classical and quantum states. In B. Coecke, editor, *New Structures for Physics*, Lecture Notes in Physics, pages 593–683. Springer, 2011.

6 B. Coecke, D. Pavlovic, and J. Vicary. A New Description of Orthogonal Bases. *Mathematical Structures in Computer Science*, 1, 2008.

7 B. Coecke, M. Sadrzadeh, and S. Clark. Mathematical Foundations for a Compositional Distributional Model of Meaning. Lambek Festschrift. *Linguistic Analysis*, 36:345–384, 2010.

8 Bob Coecke, Chris Heunen, and Aleks Kissinger. Categories of quantum and classical channels. *arXiv preprint arXiv:1305.3821*, 2013.

9 Bob Coecke and Robert W Spekkens. Picturing classical and quantum Bayesian inference. *Synthese*, 186(3):651–696, 2012.

10 E. Grefenstette and M. Sadrzadeh. Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of the EMNLP 2011*, 2011.

11 Z. Harris. *Mathematical Structures of Language*. Wiley, 1968.

12 D. Kartsaklis, M. Sadrzadeh, S. Pulman, and B. Coecke. Reasoning about meaning in natural language with compact closed categories and Frobenius algebras. *arXiv preprint arXiv:1401.5980*, 2014.

13 Dimitri Kartsaklis and Mehrnoosh Sadrzadeh. Prior disambiguation of word tensors for constructing sentence vectors. In *Proceedings of EMNLP 2013*, pages 1590–1601, 2013.

14 G. M. Kelly and M. L. Laplaza. Coherence for compact closed categories. *Journal of Pure and Applied Algebra*, 19:193–213, 1980.

15 J. Lambek. *From Word to Sentence*. Polimetrica, Milan, 2008.

16 Joachim Lambek. The mathematics of sentence structure. *American mathematical monthly*, pages 154–170, 1958.

17 A. Preller and M. Sadrzadeh. Bell states and negative sentences in the distributed model of meaning. In P. Selinger B. Coecke, P. Panangaden, editor, *Electronic Notes in Theoretical Computer Science, Proceedings of the 6th QPL Workshop on Quantum Physics and Logic*. University of Oxford, 2010.

18 Anne Preller and Joachim Lambek. Free compact 2-categories. *Mathematical Structures in Computer Science*, 17(02):309–340, 2007.

19 M. Sadrzadeh, S. Clark, and B. Coecke. The Frobenius anatomy of word meanings I: subject and object relative pronouns. *Journal of Logic and Computation*, Advance Access, October 2013.

20 H. Schütze. Automatic Word Sense Discrimination. *Computational Linguistics*, 24:97–123, 1998.

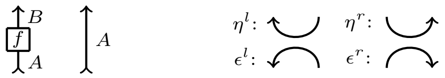
21 Peter Selinger. Dagger compact closed categories and completely positive maps. *Electronic Notes in Theoretical Computer Science*, 170:139–163, 2007.

22 Peter Selinger. A survey of graphical languages for monoidal categories. In Bob Coecke, editor, *New structures for physics*, pages 289–355. Springer, 2011.

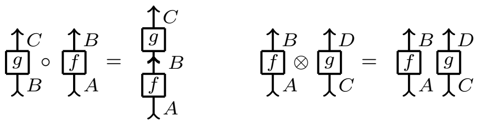
23 Peter D Turney and Patrick Pantel. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37(1):141–188, 2010.

A Graphical calculus

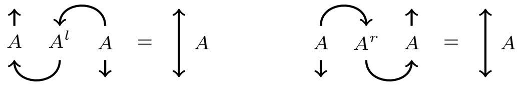
Monoidal categories are complete with regard to a graphical calculus [22] which depicts derivations in their internal language very intuitively, thus simplifying the reading and the analysis. Objects are represented as labelled wires, and morphisms as boxes with input and output wires. The η- and ϵ-maps are given as half-turns.



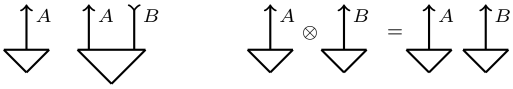
Composing morphisms amounts to connecting outputs to inputs, while the tensor product is simply juxtaposition:



In this language, the yanking equations (2) get an intuitive visual justification (here for the first two identities):

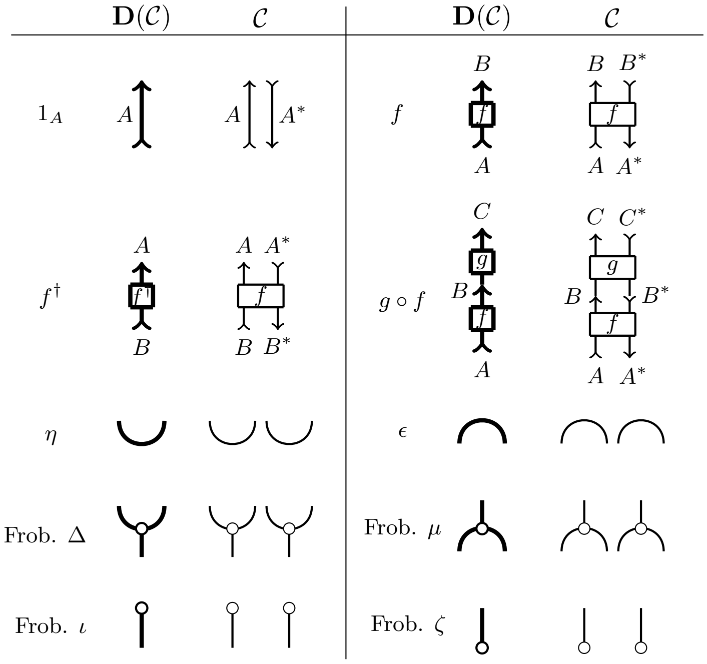


For a given object A, we define a *state* of A to be a morphism $I \to A$. If A denotes a vector space, we can think of a state as a specific vector living in that space. In our graphical language the unit object I can be omitted, leading to the following representation of states:



Note that the second diagram from the left depicts an *entangled* state of $A \otimes B$; product states (such as the rightmost one) are simple juxtapositions of two states.

B Translation from $\mathcal{C}$ to $\mathrm{D}(\mathcal{C})$



C Linguistic intuition

In order to deal with lexical ambiguity we firstly need to understand its nature. In other words, we are interested to study in what way an ambiguous word differs from an unambiguous one, and what is the defining quality that makes this distinction clear. On the surface, the answer to these questions seems straightforward: an ambiguous word is one with more than one lexicographic entries in the dictionary. However, this definition fits well only to homonymous cases, in which due to some historical accident words that share the same spelling and pronunciation refer to completely unrelated concepts. Indeed, while the number of meanings of a homonymous word such as ‘bank’ is almost fixed across different dictionaries, the same is not true for the small (and overlapping) variations of senses that might be listed under a word expressing a polysemous case.

The crucial distinction between homonymy and polysemy is that in the latter case a word still expresses a coherent and self-contained concept. Recall the example of the polysemous use of ‘bank’ as a financial institution and the building where the services of the institution are offered; when we use the sentence ‘I went to the bank’ (with the financial meaning of the word in mind) we essentially refer to *both* of the polysemous meanings of ‘bank’ at the same time – at a higher level, the word 'bank' expresses an abstract but concise concept that encompasses all of the available polysemous meanings. On the other hand, the fact that the same name can be used to describe a completely different concept (such as a river bank or a number of objects in a row) is nothing more than an unfortunate coincidence expressing lack of specification. Indeed, a listener of the above sentence can retain a small amount of uncertainty regarding the true intentions of the sayer; although her first guess would be that 'bank' refers to the dominant meaning of financial institution (including *all related polysemous meanings*), a small possibility that the sayer has actually visited a river bank still remains. Therefore, in the absence of sufficient context, the meaning of a homonymous word is more reliably expressed as a *probabilistic mixing* of the unrelated individual meanings.

In a distributional model of meaning where a homonymous word is represented by a single vector, the ambiguity in meaning has been collapsed into a convex combination of the relevant sense vectors; the result is a vector that can be seen as the average of all senses, inadequate to reflect the meaning of any of them in a reliable way. We need a way to avoid that. In natural language, ambiguities are resolved with the introduction of context (recall that meaning is use), which means that for a compositional model of meaning the resolving mechanism is the compositional process itself. We would like to retain the ambiguity of a homonymous word when needed (i.e. in the absence of appropriate context) and allow it to collapse only when the context defines the intended sense, during the compositional process.

In summary, we seek an appropriate model that will allows us: (a) to express homonymous words as probabilistic mixings of their individual meanings; (b) to retain the ambiguity until the presence of sufficient context that will eventually resolve it during composition time; (c) to achieve all the above in the multi-linear setting imposed by the vector space semantics of our original model.

D From Theory to Practice

The purpose of this appendix is to show how the theoretical ideas presented in this paper can take a concrete form using standard natural language processing techniques. The setting we present below has been used for the mini-experiments in Sect. 3.5. We approach the creation of density matrices as a three-step process: (a) we first produce an ambiguous semantic space; (b) we apply a word sense induction method on it in order to associate each word with a set of sense vectors; and finally (c) we use the sense vectors in order to create a density matrix for each word. These steps are described in separate sections below.

D.1 Creating a Concrete Semantic Space

We train our basic vector space using ukWaC, a corpus of English text with 2 billion words (100 million sentences). The basis of the vector space consists of the 2,000 most frequent content words (nouns, verbs, adjectives, and adverbs), excluding a list of *stop words*.[2] Furthermore, the vector space is lemmatized and unambiguous regarding syntactic information; in other words, each vector is uniquely identified by a (*lemma*,*pos-tag*) pair, which means for example that 'book' as a noun and 'book' as a verb are represented by different meaning vectors. The weights of each vector are set to the ratio of the probability of the context word c_i given the

[2] That is, very common words with low information content, such as the verbs 'get' and 'take' or adverbs like 'really' and 'always'.

Table 2 Derived Meanings for Word ‘Vessel’.

Meaning 1: 24070 contexts
port owner cargo fleet sailing ferry craft Navy merchant cruise navigation officer metre voyage authority deck coast launch fishery island charter Harbour pottery radio trip pay River Agency Scotland sell duty visit fish insurance skipper Roman sink War shore sail town Coastguard assistance Maritime registration call rescue bank Museum captain incident customer States yacht mooring barge comply landing Ireland sherd money Scottish tow tug maritime wreck board visitor tanker freight purchase lifeboat

Meaning 2: 5930 contexts
clot complication haemorrhage lymph stem VEGF Vitamin glucose penis endothelium retinopathy spasm antibody clotting AMD coagulation marrow lesion angina blindness medication graft vitamin vasoconstriction virus proliferation Ginkgo diabetic ventricle thickening tablet anaemia thrombus Vein leukocyte scleroderma stimulation degeneration homocysteine Raynaud breathe mediator Biloba Diabetes LDL metabolism Gene infiltrate atheroma arthritis lymphocyte lobe C’s histamine melanoma gut dysfunction vitro triglyceride infarction lipoprotein

target word t to the probability of the context word overall, as follows:

$$v_i(t) = \frac{p(c_i|t)}{p(c_i)} = \frac{\text{count}(c_i, t) \cdot \text{count}(total)}{\text{count}(t) \cdot \text{count}(c_i)}$$

where $\text{count}(c_i, t)$ refers to how many times c_i appears in the context of t (that is, in a 5-word window at either side of t) and $\text{count}(total)$ is the total number of word tokens in the corpus.

D.2 Word Sense Induction

The notion of word sense induction, that is, the task of detecting the different meanings under which a word appears in a text, is intimately connected with that of distributional hypothesis – that the meaning of a word is always context-dependent. If we had a way to create a vectorial representation for the contexts in which a specific word occurs, then, a clustering algorithm could be applied in order to create groupings of these contexts that hopefully reveal different usages of the word – *different meanings* – in the training corpus.

This intuitive idea was first presented by Schütze [20] in 1998, and more or less is the cornerstone of every unsupervised word sense induction and disambiguation method based on semantic word spaces up to today. The approach we use is a direct variation of this standard technique. For what follows, we assume that each word in the vocabulary has already been assigned to an *ambiguous* semantic vector by following typical distributional procedures, for example similar to the setting described in Sect. D.1.

We assume for simplicity that the context is defined at the sentence level. First, each context for a target word w_t is represented by a *context vector* of the form $\frac{1}{n}\sum_{i=1}^{n}|w_i\rangle$, where $|w_i\rangle$ is the semantic vector of some other word $w_i \neq w_t$ in the same context. Next, we apply hierarchical agglomerative clustering on this set of vectors in order to discover the latent senses of w_t. Ideally, the contexts of w_t will vary according to the specific meaning in which this word has been used. Table 2 provides a visualization of the outcome of this process for the ambiguous word ‘vessel’. Each meaning is visualized as a list of the most dominant words in the corresponding cluster, ranked by their TF-IDF values.

We take the centroid of each cluster as the vectorial representation of the corresponding sense/meaning. Thus, each word w is initially represented by a tuple $(|w\rangle, S_w)$, where $|w\rangle$ is the ambiguous semantic vector of the word as created by the usual distributional practice, and S_w is a set of *sense vectors* (that is, centroids of context vectors clusters) produced by the above procedure.

Note that our approach takes place at the vector level (as opposed to tensors of higher order), so it provides a natural way to create sets of meaning vectors for "atomic" words of the language, that is, for nouns. It turns out that the generalization of this to tensors of higher order is straightforward, since the clustering step has already equipped us with a number of sets consisting of context vectors, each one of which stands in one-to-one correspondence with a set of contexts reflecting a different semantic usage of the higher-order word. One then can use, for example, the argument "tensoring and summing" procedure of [10] (briefly described in Sect. 2.4) in order to compute the meaning of the ith sense of a word of arity n as:

$$|word\rangle_i = \sum_{c \in C_i} \bigotimes_{k=1}^{n} |arg_{k,c}\rangle \tag{19}$$

where C_i is the set of contexts associated with the ith sense, and $arg_{k,c}$ denotes the kth argument of the target word in context c. Of course, more advanced statistical methods could be also used for learning the sense tensors from the provided partitioning of the contexts, as long as these methods respect the multi-linear nature of the model. This completes the word sense induction step.

D.3 Creating Density Matrices

We have now managed to equip each word with a set of sense vectors (or higher-order tensors, depending on its grammatical type). Assigning a probability to each sense is trivial and can be directly derived by the number of times the target word occurs under a specific sense divided by the total occurrences of the word in the training corpus. This creates a statistical ensemble of state vectors and probabilities that can be used for computing a density matrix for the word according to Def. 2.

Modules Over Monads and Their Algebras

Maciej Piróg[1], **Nicolas Wu**[2], **and Jeremy Gibbons**[1]

1 **Department of Computer Science, University of Oxford**
Wolfson Building, Parks Rd, Oxford OX1 3QD, UK
`{firstname.lastname}@cs.ox.ac.uk`

2 **Department of Computer Science, University of Bristol**
Merchant Venturers Building, Woodland Rd, Bristol BS8 1UB, UK
`nicolas.wu@bristol.ac.uk`

Abstract

Modules over monads (or: actions of monads on endofunctors) are structures in which a monad interacts with an endofunctor, composed either on the left or on the right. Although usually not explicitly identified as such, modules appear in many contexts in programming and semantics. In this paper, we investigate the elementary theory of modules. In particular, we identify the monad freely generated by a right module as a generalisation of Moggi's resumption monad and characterise its algebras, extending previous results by Hyland, Plotkin and Power, and by Filinski and Støvring. Moreover, we discuss a connection between modules and algebraic effects: left modules have a similar feeling to Eilenberg–Moore algebras, and can be seen as handlers that are natural in the variables, while right modules can be seen as functions that run effectful computations in an appropriate context (such as an initial state for a stateful computation).

1998 ACM Subject Classification F.3.2 Semantics of Programming Languages, F.3.3 Studies of Program Constructs

Keywords and phrases monad, module over monad, algebraic data types, resumptions, free object

Digital Object Identifier 10.4230/LIPIcs.CALCO.2015.290

1 Introduction

Given a monad M, a *right module* over M (or: an M-module) is an endofunctor S together with a natural transformation (called an *action*)

$$\overrightarrow{\mu} : SM \to S$$

coherent with the monadic structure of M. Dually, a *left module* over M is an endofunctor L together with a natural transformation

$$\overleftarrow{\mu} : ML \to L$$

also coherent with the monadic structure of M.

Modules over monads are special cases of modules over monoids in monoidal categories (as monads are monoids in categories of endofunctors). They are discussed, for example, by Dubuc [10] and Mac Lane [23, Sec. VI.4]), as well as, in a more general setting, by Street [32]. In this paper, by developing some elementary theory of modules, we show their connections to some constructions in semantics of programming languages and the theory of algebraic data structures.

As our primary result, we describe the monad freely generated by a right M-module S. The functor part of this monad is given by the composition MS^*, where S^* is the free

6th International Conference on Algebra and Coalgebra in Computer Science (CALCO'15).
Editors: Lawrence S. Moss and Pawel Sobocinski; pp. 290–303
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

monad generated by S as an endofunctor. We also introduce the notion of *algebra for a module*, which is a coherent pair consisting of an S-algebra (for S as an endofunctor) and an Eilenberg–Moore M-algebra. We observe that algebras for a right M-module S coincide with Eilenberg–Moore algebras for the monad MS^*.

These considerations have some practical aspects as well. The monad MS^* is a generalisation of Moggi's [27] resumption monad $M(GM)^*$ for an endofunctor G, which has applications in semantics and functional programming. The universal property of MS^* subsumes Hyland, Plotkin, and Power's [20] result that $M(GM)^*$ is the coproduct of M and G^* in the category of monads, or Filinski and Støvring's [13] construction of data types that interleave data and monadic effects. Generalising the above constructions to the setting of modules gives us new, interesting instances.

In passing, we investigate more of the theory of modules. We give examples and general constructions, which suggest the ubiquity of modules. For instance, every (left or right) adjoint comonad is a module over its adjoint monad, and every endofunctor is a module over its codensity monad. We show that a large portion of the theory of monads can be transported to the theory of modules. For example, the connection between monads and adjunctions is lifted to the connection between modules and adjunctions paired with a functor, while the correspondence between distributive laws and liftings is extended to the correspondence between their obvious counterparts.

2 Modules over monads

2.1 Preliminaries

We work in a base category $\mathscr{C}$, which is locally small and complete. We reserve A, B, C, X, Y, Z to denote objects in categories, while f, g, h denote morphisms and natural transformations. Functors are usually denoted as F, U, G, H, M, T. We reserve M, T for monads, and F, U for left and right adjoints respectively. To avoid confusion, we sometimes add superscripts. Given functors G, G', H, H' and two natural transformations $g : G \to G'$ and $h : H \to H'$, we denote the composition of endofunctors by juxtaposition (for example, GH). The parallel composition of g and h is also denoted by juxtaposition, as in $gh : GH \to G'H'$.

For an endofunctor G, we write $\mathbf{Alg}(G)$ for the category of G-algebras, $\mathbf{Mnd}$ for the category of monads and monad morphisms over a base category, and $\mathbf{EM}(M)$ for the category of Eilenberg–Moore algebras for a monad M. We always denote the unit and the multiplication of a monad as η and μ respectively. If there is more than one monad in context, we add superscripts. We follow the standard abuse of notation and denote a monad by its underlying endofunctor.

Given an endofunctor $G : \mathscr{C} \to \mathscr{C}$, we denote the free monad generated by G (if it exists) as G^*, and the canonical injection by $\mathsf{emb} : G \to G^*$. For a monad T and a natural transformation $g : G \to T$, we denote by $[\![g]\!] : G^* \to T$ the monad morphism given by the freeness of G^*, that is, the unique monad morphism $[\![g]\!]$ with the property that $g = [\![g]\!] \cdot \mathsf{emb}$. Note that if the base category has binary coproducts, the functor part of G^* is given by $G^*A = \mu X.GX+A$, where $\mu X.HX$ denotes the carrier of the initial H-algebra (see Kelly [21]). In such a case, the free monad arises from the adjunction between $F^{\mathrm{Alg}} : \mathscr{C} \to \mathbf{Alg}(G)$ and $U^{\mathrm{Alg}} : \mathbf{Alg}(G) \to \mathscr{C}$, which we write as $F^{\mathrm{Alg}} \dashv U^{\mathrm{Alg}} : \mathscr{C} \rightharpoonup \mathbf{Alg}(G)$. This adjunction is strictly monadic, which means that the canonical comparison functor $K : \mathbf{Alg}(G) \to \mathbf{EM}(G^*)$ is an isomorphism.

2.2 Modules defined

▶ **Definition 1.** Let M be a monad. An endofunctor S together with a natural transformation (an *action*) $\overrightarrow{\mu} : SM \to S$ is called a *right M-module* (or: a *right module over M*) if the following diagrams commute:

$$\begin{tikzcd}
SMM \arrow[r, "S\mu"] \arrow[d, "\overrightarrow{\mu}M"] & SM \arrow[d, "\overrightarrow{\mu}"] \\
SM \arrow[r, "\overrightarrow{\mu}"] & S
\end{tikzcd}
\qquad
\begin{tikzcd}
S \arrow[r, "S\eta"] \arrow[dr, "\mathsf{id}"] & SM \arrow[d, "\overrightarrow{\mu}"] \\
 & S
\end{tikzcd}$$

We define a morphism between an M-module S and a M'-module S' as a pair $\langle m, s\rangle$, where $m : M \to M'$ is a monad morphism and $s : S \to S'$ is a natural transformation such that the following diagram commutes:

$$\begin{tikzcd}
SM \arrow[r, "\overrightarrow{\mu}^{S}"] \arrow[d, "sm"] & S \arrow[d, "s"] \\
S'M' \arrow[r, "\overrightarrow{\mu}^{S'}"] & S'
\end{tikzcd}$$

We refer to the category of all right modules over monads as **Mod**. Its objects are pairs $\langle M, S\rangle$, where M is a monad and S is an M-module. Arrows are given by morphisms between modules.

Similarly, we define a *left M-module* as an endofunctor L together with $\overleftarrow{\mu} : ML \to L$ such that the obvious analogues of the diagrams above commute.

▶ **Example 2.** The following are examples of general constructions of right modules (most of them dualise easily to the case of left modules):

1. Let M be a monad. Then, M is itself an M-module with the action given by the multiplication $\mu^M : MM \to M$.
2. Let $m : M \to T$ be a monad morphism. Then, T is an M-module with the action given by $\mu^T \cdot Tm : TM \to T$.
3. Let S be an M-module and G be an endofunctor. Then, GS is also an M-module with the action given by $G\overrightarrow{\mu} : GSM \to GS$. An important instance of this construction is when the original module is the monad itself, that is, when the module is given by GM.
4. With the definitions as above, let $\lambda : TM \to MT$ be a distributive law between monads. Then, the composition ST is a module of the induced monad MT. The action is given by $(\overrightarrow{\mu}\mu^T) \cdot S\lambda T : STMT \to MT$.
5. If S and Q are two M-modules, their coproduct $S + Q$ is also an M-module with the action defined componentwise.
6. Let F be an endofunctor with a right adjoint U. Then, F is an UF-module with the action given by $\varepsilon F : FUF \to F$, where ε is the counit of the adjunction.
7. Let M be a monad with a left adjoint W. In such a case, W is a comonad (the situation is dual to the one observed by Eilenberg and Moore [12], in which M has a right adjoint). Also, W is an M-module with the action given by $cW \cdot W\mu W \cdot WMu : WM \to W$, where $u : \mathsf{Id} \to MW$ is the unit and $c : WM \to \mathsf{Id}$ is the counit of the adjunction.

▶ **Example 3.** The last two constructions from Example 2 can be illustrated with the 'currying' adjunction native to cartesian closed categories, $A \times \text{-} \dashv (\text{-})^A$, for a fixed object A.

As for Example 2 (6), this adjunction gives rise to the state monad $(A \times \text{-})^A$, which can be seen as a model of stateful computations. The action of the module given by the left adjoint is equal to $\overrightarrow{\mu}_X = \mathsf{eval}^A_{A \times X} : A \times (A \times X)^A \to A \times X$, where $\mathsf{eval}^A_B : A \times B^A \to B$ is the evaluation morphism for exponentials. Alternatively, using the fact that simply-typed λ-calculus is the internal language of CCCs, one could say $\overrightarrow{\mu}_X = \lambda \langle a, f \rangle{:}A \times (A \times X)^A.\, f\, a$. Intuitively, $\overrightarrow{\mu}$ takes as its arguments an initial state and a stateful computation, and returns the final state paired with the final answer. In other words, it is a morphism that 'executes' the stateful computation.

Example 2 (7) comes from the fact that $(\text{-})^A$ is a monad, known in the functional programming community as the reader monad. Its multiplication $(X^A)^A \to X^A$ is given by the diagonal $\lambda f{:}(X^A)^A.\, \lambda a{:}A.\, f\, a\, a$. Its adjoint comonad (known as the environment comonad) $A \times \text{-}$ is a $(\text{-})^A$-module. The action of this module $\overrightarrow{\mu}_X : A \times X^A \to A \times X$ is given as $\lambda \langle a, f \rangle{:}A \times X^A.\, \langle a,\ f\, a \rangle$.

▶ **Example 4.** For all $n \geq 1$, the **Set** functor of lists with at least n elements is a module of the non-empty list monad (the free semigroup monad).

▶ **Example 5.** An Eilenberg–Moore algebra $\langle A,\ f : MA \to A \rangle$ can be understood as a left M-module given by the constant endofunctor C_A and a natural transformation $f : MC_A \to C_A$. Indeed, in the literature, Eilenberg–Moore algebras are sometimes called 'modules'.

We can consider **Cat** (the category of all categories up to a certain size) as a 2-category: 0-cells are categories, 1-cells are functors, and 2-cells are natural transformations. We consider different opposites of **Cat**: the op-dual $\mathbf{Cat}^{\mathsf{op}}$ obtained by reversing 1-cells, the co-dual $\mathbf{Cat}^{\mathsf{co}}$ obtained by reversing 2-cells, and the bidual $\mathbf{Cat}^{\mathsf{coop}}$ obtained by reversing both. For example, monads and comonads are mutually co-dual concepts (that is, a monad in $\mathbf{Cat}^{\mathsf{co}}$ is a comonad in **Cat**), while both are self-op-dual (that is, a monad is an opmonad, while a comonad is a co-opmonad). Left and right modules are mutually op-dual concepts, that is, a left module is a right opmodule, while a right module is a left opmodule. In the obvious way, co-duality gives us the concepts of left and right comodules over comonads.

▶ **Example 6.** Given an endofunctor $G : \mathscr{C} \to \mathscr{C}$, its codensity monad is given by the right Kan extension of G along itself:

$$\begin{tikzcd}
\mathscr{C} \arrow[r, "G"] \arrow[d, "G"'] \arrow[dr, phantom, "\Uparrow \kappa" near start] & \mathscr{C} \\
\mathscr{C} \arrow[ur, "\mathsf{Ran}_G G"'] &
\end{tikzcd}$$

In this case, G is a left module of $\mathsf{Ran}_G G$ with the natural transformation $\kappa : (\mathsf{Ran}_G G)G \to G$ being the module action. Intuitively, we can see the codensity monad as a generalised type of computations in continuation-passing style. The transformation κ executes the computation by supplying it with the identity continuation. Moreover, if G happens to have a left adjoint F, the codensity monad is equal to GF, and the situation simplifies to the op-dual of Example 2 (6).

Examples 3 and 6 suggest that some actions of modules can be intuitively seen as functions that *run* computations, while the functor parts provide *context* for the execution. We say more about this view on modules in Section 5.

2.3 Adjunctions paired with a functor

Monads and pairs of adjoint functors are closely related: every adjunction induces a monad, and every monad is induced by a number of adjunctions. This can be extended to modules over monads: M-modules are induced by adjunctions that induce M together with an additional functor. The only condition imposed on the functor is that it has the same type as the right adjoint (for right modules) or the left adjoint (for left modules). In detail:

▶ **Theorem 7.** *Let $F : \mathscr{C} \to \mathscr{D}$ be a functor with a right adjoint $U : \mathscr{D} \to \mathscr{C}$. We denote the unit and the counit as η and ε respectively. Then:*

- *Let $L : \mathscr{C} \to \mathscr{D}$ be a functor. Then, UL is a left UF-module with the action given as $U\varepsilon L : UFUL \to UL$.*
- *Let $R : \mathscr{D} \to \mathscr{C}$ be a functor. Then, RF is a right UF-module with the action given as $R\varepsilon F : RFUF \to RF$.*

Conversely, every M-module arises from an adjunction that induces M together with a functor of the appropriate type. In the case of left modules, this fact was noticed by Dubuc [10]; we complement it with a suitable counterpart of the Kleisli construction for right modules.

▶ **Theorem 8.** *Let $F^{\mathrm{EM}} \dashv U^{\mathrm{EM}}$ be the Eilenberg–Moore adjunction for a monad M on a category $\mathscr{C}$. For a left M-module S, we define a functor $L : \mathscr{C} \to \mathbf{EM}(M)$ as follows:*

$$LA = \langle SA,\ MSA \xrightarrow{\overleftarrow{\mu}_A} SA \rangle$$

$$L(f : A \to B) = SA \xrightarrow{Sf} SB$$

The induced module $U^{\mathrm{EM}}L$ is equal to S.

▶ **Theorem 9.** *Let $F^{\mathrm{Kl}} \dashv U^{\mathrm{Kl}}$ be the Kleisli adjunction for a monad M on a category $\mathscr{C}$. For a right M-module S, we define a functor $R : \mathbf{Kleisli}(M) \to \mathscr{C}$ as follows:*

$$RA = SA$$

$$R(f : A \to MB) = SA \xrightarrow{Sf} SMB \xrightarrow{\overrightarrow{\mu}_B} SB$$

The induced module RF^{Kl} is equal to S.

2.4 Distributive laws and liftings

Since most results about monads boil down to results about adjunctions, the construction above suggests that we can generalise much of the theory of monads to the theory of modules. As an example, we now consider distributive laws and liftings, introduced and proved equivalent by Beck [8] (see also Barr and Wells [7, Sec. 9.2] or Tanaka's PhD dissertation [33]):

▶ **Definition 10.** A *distributive law* of an endofunctor G over a monad M is a natural transformation $\lambda : GM \to MG$ such that $\lambda \cdot G\mu = \mu G \cdot M\lambda \cdot \lambda M$ and $\lambda \cdot G\eta = \eta G$. Similarly, a distributive law of a monad T over an endofunctor G is a natural transformation $\lambda : TG \to GT$ such that $\lambda \cdot \mu G = G\mu \cdot \lambda G \cdot M\lambda$ and $\lambda \cdot \eta G = G\eta$. A distributive law between monads T and M is a natural transformation $\lambda : TM \to MT$ that is both a distributive law of T as a endofunctor over M as a monad and T as a monad over M as an endofunctor.

▶ **Definition 11.** Given a monad T and an endofunctor G, we call an endofunctor $\overline{G} : \mathbf{EM}(T) \to \mathbf{EM}(T)$ a *lifting* of G to $\mathbf{EM}(T)$ if $U^{\mathrm{EM}}\overline{G} = GU^{\mathrm{EM}}$, where $U : \mathbf{EM}(T) \to \mathscr{C}$ is the forgetful functor. Let M be a monad. We call a monad $\overline{M} : \mathbf{EM}(T) \to \mathbf{EM}(T)$ a lifting of M (as a monad) if it is a lifting as an endofunctor and additionally the identities $\mu^M U^{\mathrm{EM}} = U^{\mathrm{EM}}\mu^{\overline{M}}$ and $\eta^M U^{\mathrm{EM}} = U^{\mathrm{EM}}\eta^{\overline{M}}$ hold.

▶ **Theorem 12.** *Let M and T be monads, and G be an endofunctor. Liftings of G to $\mathbf{EM}(T)$ are in 1–1 correspondence with distributive laws of T over G. Moreover, liftings of M (as a monad) are in 1–1 correspondence with distributive laws of T over M (as monads).*

We extend these notions and the correspondence to include modules:

▶ **Definition 13.** A *distributive law* of a monad T over a right M-module S consists of a distributive law between monads $\lambda : TM \to MT$ together with a distributive law of a monad over an endofunctor $\overrightarrow{\lambda} : TS \to ST$ such that $\overrightarrow{\lambda} \cdot T\overrightarrow{\mu} = \overrightarrow{\mu}T \cdot S\lambda \cdot \overrightarrow{\lambda}M$.

▶ **Definition 14.** Let T be a monad and S be a right M-module. An *Eilenberg–Moore lifting* of S as a module consists of $\overline{M}$ and $\overline{S}$ together with a natural transformation $\overrightarrow{\mu}^{\overline{S}} : \overline{S}\,\overline{M} \to \overline{S}$ such that:

- $\overline{M} : \mathbf{EM}(T) \to \mathbf{EM}(T)$ is a lifting of M as a monad,
- $\overline{S} : \mathbf{EM}(T) \to \mathbf{EM}(T)$ is a lifting of S as a functor,
- $\overline{S}$ together with $\overrightarrow{\mu}^{\overline{S}}$ form a right $\overline{M}$-module,
- it is the case that $U^{\mathrm{EM}}\overrightarrow{\mu}^{\overline{S}} = \overrightarrow{\mu}^{S}U^{\mathrm{EM}}$.

▶ **Theorem 15.** *Distributive laws of a monad T over an M-module S and Eilenberg–Moore liftings of S are in 1–1 correspondence.*

3 Resumptions: monads freely generated by modules

In this section, we introduce the monad MS^* freely induced by a right M-module S. Its monadic structure is an obvious generalisation of Hyland, Plotkin, and Power's construction [20] for Moggi's [27] monad $M(GM)^*$ for an endofunctor G.

Since in this and the next sections, we are interested only in right modules, when no direction (left or right) is given, we mean *right* modules.

▶ **Theorem 16.** *Given a monad M, let $\langle S, \overrightarrow{\mu}\rangle$ be an M-module. The functor MS^* can be given monadic structure via a distributive law $\lambda : S^*M \to MS^*$.*

Proof. First, consider the natural transformation $\delta = \eta\overrightarrow{\mu} : SM \to MS$. It is easy to verify that it is a distributive law of the functor S over the monad M. Such a distributive law gives us the following lifting $\overline{M} : \mathbf{Alg}(S) \to \mathbf{Alg}(S)$ of M to $\mathbf{Alg}(S)$:

$$\overline{M}\langle A,\, SA \xrightarrow{a} A\rangle = \langle MA,\, SMA \xrightarrow{\delta_A} MSA \xrightarrow{Ma} A\rangle$$
$$\overline{M}f = Mf$$

Since the category $\mathbf{Alg}(S)$ is monadic over $\mathscr{C}$ (hence, $\mathbf{Alg}(S) \cong \mathbf{EM}(S^*)$), this lifting can be seen as a lifting to $\mathbf{EM}(S^*)$:

$$\overline{M} : \mathbf{EM}(S^*) \to \mathbf{EM}(S^*)$$

Applying Theorem 12, we obtain a distributive law $\lambda : S^*M \to MS^*$, which gives a monadic structure to MS^*. ◀

Using the definitions of the appropriate isomorphisms and with some calculation, we can read off a direct definition of the distributive law in terms of a fold, that is, the unique algebra homomorphism from the initial $(S(\text{-}) + MA)$-algebra to the following algebra, where $\mathsf{cons}_A : SS^*A \to S^*A$ is the action of the free S-algebra generated by the object A:

$$\lambda_A = (\!|f|\!) : S^*MA \to MS^*A,$$

where $(\!|f|\!)$ is the unique algebra homomorphism from the initial algebra to $\langle MS^*A,\, f\rangle$ for

$$f = [\mu^M_{S^*A} \cdot \mathsf{cons}_A \cdot \overrightarrow{\mu}^S_{S^*A},\; M\eta^{S^*}_A] : SMS^*A + MA \to MS^*A.$$

▶ **Example 17.** The monad MS^* is a generalisation of Moggi's resumption monad $M(GM)^*$ for an endofunctor G. Moggi's monad arises as the special case for $S = GM$. It follows from Example 2 (3) that GM is an M-module. Using the 'rolling rule' [6], Moggi's monad can be rewritten as $A \mapsto \mu X.M(GX + A)$. A distinctive feature of our construction is that in general it is not given by an initial algebra.

Moggi's monad is an important data structure in functional programming, as it is often used to implement a form of algebraic effects. The endofunctor G represents a signature, while M is a background monad. Handling of the signature takes G to M, which in Haskell is often the *IO* monad. Important examples of this pattern are given by streaming I/O libraries, which help to manage resources efficiently without losing purity; see, for example, Kiselyov [22].

▶ **Example 18.** We instantiate our resumption monad with the reader monad $(\text{-})^A$ together with its module $A \times (\text{-})$ (see Example 3) to obtain $((A \times (\text{-}))^*)^A$. It is a version of the state monad that accumulates the intermediate states in a sequence. (We have previously [28] given a 'coinductive' version of this example.)

The monad MS^* is an important construction in the theory of modules, since it is freely generated by S understood as a module. First, we notice that the monad M can be seen as its own module with $\overrightarrow{\mu} = \mu$. Moreover, this construction is functorial:

▶ **Definition 19.** We define a functor $\Delta : \mathbf{Mnd} \to \mathbf{Mod}$ as follows:

$$\Delta M = \langle M, M\rangle$$
$$\Delta f = \langle f, f\rangle$$

The above functor can be seen as a form of a (dependent) diagonal, hence the notation Δ. Mac Lane [23] calls the module ΔM the *right regular representation* of M, referring to a similar concept from representation theory in abstract algebra. Hirschowitz and Maggesi [18] call ΔM the *tautological* module of M.

▶ **Theorem 20.** *The monad MS^* is the free object in the category* **Mnd** *generated by S with respect to the functor Δ. More precisely, this means that for monads M and T, an M-module S, and a module morphism $\langle m, f\rangle : \langle M, S\rangle \to \Delta T$, there exists a unique monad morphism $k : MS^* \to T$ such that the following diagram commutes:*

$$\begin{tikzcd}
M \arrow[r, "M\eta^{S^*}"] \arrow[dr, "m"'] & MS^* \arrow[d, "k"] & S^* \arrow[l, "\eta^M S^*"'] & S \arrow[l, "\mathsf{emb}^S"'] \arrow[dll, "f"] \\
& T & &
\end{tikzcd} \tag{1}$$

Proof. We define $k = \mu^T \cdot m[\![f]\!]$. Using the direct definition of λ, it can be shown using the properties of initial algebras that k is indeed a monad morphism.

It is easy to see that the diagram (1) commutes from the properties of monads and the freeness of S^*. To see that k is unique such a morphism, consider a monad morphism $r : MS^* \to T$ such that the diagram (1) commutes if we substitute r for k. Since $\eta^M S^* : S^* \to MS^*$ is a monad morphism, the composition $r \cdot \eta^M S^* : S^* \to T$ is a monad morphism, hence, from the freeness of S^*, we obtain that

$$r \cdot \eta^M S^* = [\![f]\!] \tag{2}$$

We calculate:

$$\begin{aligned}
r &= r \cdot \mu^{MS^*} \cdot \eta^{MS^*} MS^* && \text{(monads)}\\
&= r \cdot \mu^M \mu^{S^*} \cdot M\lambda S^* \cdot \eta^M \eta^{S^*} MS^* && \text{(def.)}\\
&= r \cdot \mu^M \mu^{S^*} \cdot \eta^M M \eta^{S^*} S^* && \text{(distr. law)}\\
&= r \cdot \mu^M \mu^{S^*} \cdot M\eta^M \eta^{S^*} S^* && \text{(monads)}\\
&= r \cdot \mu^M \mu^{S^*} \cdot M\lambda S^* \cdot M\eta^{S^*} \eta^M S^* && \text{(distr. law)}\\
&= r \cdot \mu^{MS^*} \cdot M\eta^{S^*} \eta^M S^* && \text{(def.)}\\
&= \mu^T \cdot rr \cdot M\eta^{S^*} \eta^M S^* && \text{(monad morphism)}\\
&= \mu^T \cdot m[\![f]\!] && \text{(LHS of (1) and (2))}\\
&= k && \text{(def.)}
\end{aligned}$$

◀

4 Algebras for modules

In this section, we introduce the notion of an algebra for a module. We show that the category of all such algebras for an M-module S coincides with the category of Eilenberg–Moore algebras for the monad MS^*.

▶ **Definition 21.** An algebra for an M-module S is a triple $\langle A,\ f : MA \to A,\ g : SA \to A\rangle$ such that:

1. The morphism f is an Eilenberg–Moore M-algebra.
2. The morphism g is an S-algebra.
3. The following coherence diagram commutes:

$$\begin{tikzcd}
SMA \arrow[r, "Sf"] \arrow[d, "\overrightarrow{\mu}_A"] & SA \arrow[d, "g"] \\
SA \arrow[r, "g"] & A
\end{tikzcd}$$

A morphism between two algebras $\langle A, f, g\rangle$ and $\langle B, f', g'\rangle$ is a morphism $h : A \to B$ that is both an S-algebra homomorphism $f \to f'$ and an M-algebra homomorphism $g \to g'$. We denote the category of algebras for an M-module S as $\mathbf{ModAlg}(M, S)$.

▶ **Theorem 22.** *Let S be an M-module. If S^* exists, the obvious forgetful functor U^{ModAlg} : $\mathbf{ModAlg}(M, S) \to \mathscr{C}$ has a left adjoint F^{ModAlg} given by:*

$$F^{\mathrm{ModAlg}} A = \langle MS^*A,\ f,\ g\rangle, \text{ where}$$

$$f = MMS^*A \xrightarrow{\mu^M_{S^*A}} MS^*A$$

$$g = SMS^*A \xrightarrow{\overrightarrow{\mu}_{S^*A}} SS^*A \xrightarrow{\mathsf{cons}_A} S^*A \xrightarrow{\eta^M_{S^*A}} MS^*A$$

$$F^{\mathrm{ModAlg}} h = MS^*h$$

The monad induced by this adjunction is equal to MS^*.

Proof. Consider the adjunction $F^{\mathrm{Alg}} \dashv U^{\mathrm{Alg}} : \mathscr{C} \rightharpoonup \mathbf{Alg}(S)$. The lifting $\overline{M}$ defined in the proof of Theorem 16 can be seen as a monad on $\mathbf{Alg}(S)$. It gives rise to an Eilenberg–Moore adjunction $F^{\mathrm{EM}} \dashv U^{\mathrm{EM}} : \mathbf{Alg}(S) \rightharpoonup \mathbf{EM}(\overline{M})$. The objects of $\mathbf{EM}(\overline{M})$ are algebras of the following shape:

$$\langle\langle A, g : SA \to A\rangle,\ f : \overline{M}\langle A, g\rangle \to \langle A, g\rangle\rangle$$

They satisfy the following conditions:

- The morphism g is an S-algebra (obviously).
- The morphism f has the Eilenberg–Moore property. Since $\overline{M}$ inherits its monadic structure from M, the morphism $f : MA \to A$ understood as a $\mathscr{C}$-morphism has the Eilenberg–Moore property for M.
- The morphism f is an algebra homomorphism between $\overline{M}\langle A, g\rangle = \langle MA,\ SMA \xrightarrow{\overrightarrow{\mu}_A} SA \xrightarrow{g} A \xrightarrow{\eta^M_A} MA\rangle$ and $\langle A, g\rangle$. The homomorphism diagram is then as follows:

$$\begin{tikzcd}
SMA \arrow[r, "\overrightarrow{\mu}_A"] \arrow[d, "Sf"] & SA \arrow[r, "g"] & A \arrow[r, "\eta^M_A"] \arrow[dr, dashed, "\mathsf{id}_A"] & MA \arrow[d, "f"] \\
SA \arrow[rrr, "g"] & & & A
\end{tikzcd}$$

  Since f has the Eilenberg–Moore property, it is the case that $f \cdot \eta^M_A = \mathsf{id}_A$ (as indicated by the dashed arrow).

These are exactly the conditions for $\langle A, f, g\rangle$ to be an algebra for the module S, which means that $\mathbf{ModAlg}(M, S) \cong \mathbf{EM}(\overline{M})$. The adjunction in question is then given by the following composite adjunction:

$$F^{\mathrm{EM}} F^{\mathrm{Alg}} \dashv U^{\mathrm{Alg}} U^{\mathrm{EM}} : \mathscr{C} \rightharpoonup \mathbf{ModAlg}(M, S) \cong \mathbf{EM}(\overline{M})$$

It is easy to see that U^{ModAlg} agrees with $U^{\mathrm{Alg}} U^{\mathrm{EM}}$, so its left adjoint is given by $F^{\mathrm{EM}} F^{\mathrm{Alg}}$ modulo the isomorphism. Simple unfolding of the definitions of F^{Alg} and F^{EM} gives us that the direct definition of their composition is as specified in the theorem. ◀

▶ **Example 23.** We can instantiate the theorem above to the 'bialgebraic' proof by Hyland, Plotkin, and Power's [20] that $M(GM)^*$ is a coproduct of M and G^* in **Mnd**. First, for two monads M and T, we define an $\langle M, T\rangle$-bialgebra as a triple $\langle A,\ f : MA \to A,\ g : TA \to A\rangle$, where f and g are Eilenberg–Moore algebra actions. All $\langle M, T\rangle$-bialgebras form a category, $\mathbf{BiAlg}(M, T)$, with morphisms given by $\mathscr{C}$-morphisms that are both M- and T-algebra homomorphisms. As shown by Kelly [21], in a category with coproducts, if the obvious forgetful functor from $\mathbf{BiAlg}(M, T)$ to the base category has a left adjoint, the induced monad is a coproduct of M and T in $\mathbf{Mnd}$. Indeed, for an M-module GM (see Example 2 (3) and (1)), one can prove that the category $\mathbf{ModAlg}(M, GM)$ is isomorphic to $\mathbf{BiAlg}(M, G^*)$ as follows.

Since $\mathbf{EM}(G^*) \cong \mathbf{Alg}(G)$, we can work with G-algebras (instead of Eilenberg–Moore G^*-algebras) in the third component of bialgebras. Given an algebra for a module $\langle A,\ f : MA \to A,\ g : GMA \to A\rangle$, we define the corresponding bialgebra as $\langle A,\ f,\ g \cdot G\eta_A : GA \to A\rangle$. Given a bialgebra $\langle A,\ f : MA \to A,\ g : GA \to A\rangle$, we define the corresponding algebra for a module as $\langle A,\ f,\ g \cdot Gf : GMA \to A\rangle$. The coherence condition follows easily from the fact that f is an Eilenberg–Moore algebra action. Simple calculation reveals that that the two transformations are mutual inverses. It is also easy to verify that a morphism between two algebras for a module is also a morphism between the corresponding bialgebras and *vice versa.*

Theorem 22 characterises the left adjoint to U^{ModAlg} (and so, to the forgetful functor $\mathbf{BiAlg}(M, G^*)$). The induced monad is indeed the free monad generated by the module GM, that is, $M(GM)^*$.

▶ **Example 24.** As defined by Atkey *et al.* [5], following Filinski and Støvring [13], a G-and-M-algebra is a triple $\langle A,\ m : MA \to A,\ f : GA \to A\rangle$, where M is a monad, G is an endofunctor, f is a morphism, and m is an Eilenberg–Moore algebra action. Morphisms between two G-and-M-algebras are $\mathscr{C}$-morphisms that are both G- and M-algebra homomorphisms. The initial G-and-M-algebra (whose carrier is given by $\mu MG \cong M(\mu GM)$) is used to model effectful datatypes, which interleave structure and monadic effects. Employing the isomorphism $\mathbf{EM}(G^*) \cong \mathbf{Alg}(G)$, one can easily see that the category of G-and-M-algebras is isomorphic to $\mathbf{BiAlg}(M, G^*)$, and so, as described in the previous example, isomorphic to $\mathbf{ModAlg}(M, GM)$. Since $F^{\mathrm{ModAlg}} : \mathscr{C} \to \mathbf{ModAlg}(M, GM)$ is cocontinuous (since it is a left adjoint), the initial G-and-M-algebra can be obviously reconstructed as $F^{\mathrm{ModAlg}}0$, where 0 is the initial object of $\mathscr{C}$.

▶ **Theorem 25.** *If S^* exists, the functor U^{ModAlg} is strictly monadic. This entails that the category $\mathbf{ModAlg}(M, S)$ is isomorphic to $\mathbf{EM}(MS^*)$.*

Proof. We use the strict version of Beck's monadicity theorem (see Mac Lane [23, Sec. VI.7]). We have already shown that U^{ModAlg} is a right adjoint, so it remains to show that it creates coequalisers for those parallel h_0, h_1 in $\mathbf{ModAlg}(M, S)$ for which $U^{\mathrm{ModAlg}}h_0$ and $U^{\mathrm{ModAlg}}h_1$ have a split coequaliser in $\mathscr{C}$.

Let $h_0, h_1 : \langle A, f^A, g^A\rangle \to \langle B, f^B, g^B\rangle$ be such a pair. Let c be a split coequaliser of $U^{\mathrm{ModAlg}}h_0$ and $U^{\mathrm{ModAlg}}h_1$. In other words, there exist morphisms s and t such that the following diagram commutes in $\mathscr{C}$ and in which the two horizontal compositions are the identities:

$$\begin{tikzcd}
B \arrow[r, "t"] \arrow[d, "c"] & A \arrow[r, "h_0"] \arrow[d, "h_1"] & B \arrow[d, "c"] \\
C \arrow[r, "s"] & B \arrow[r, "c"] & C
\end{tikzcd} \tag{3}$$

We need to show that there exist unique $f^C : MC \to C$ and $g^C : SA \to A$ such that $\langle C, f^C, g^C\rangle$ is an algebra for a module, and $c : \langle B, f^B, g^B\rangle \to \langle C, f^C, g^C\rangle$ is a homomorphism and a coequaliser of h_0 and h_1. From the monadicity of the forgetful functors $U^{\mathrm{EM}} : \mathbf{EM}(M) \to \mathscr{C}$ and $U^{\mathrm{Alg}} : \mathbf{Alg}(S) \to \mathscr{B}$, we obtain that there exist a unique Eilenberg–Moore

M-algebra $\langle C, f^C\rangle$ and a unique S-algebra $\langle C, g^C\rangle$, where

$$f^C = MC \xrightarrow{Ms} MB \xrightarrow{f^B} B \xrightarrow{c} C,$$

$$g^C = SC \xrightarrow{Ss} SB \xrightarrow{g^B} B \xrightarrow{c} C,$$

such that c is the coequaliser of $h_0, h_1 : \langle A, f^A\rangle \to \langle B, f^B\rangle$ understood as Eilenberg–Moore M-algebra homomorphisms and simultaneously the coequaliser of $h_0, h_1 : \langle A, g^A\rangle \to \langle B, g^B\rangle$ understood as S-algebra homomorphisms. Thus, it is left to check that $\langle C, f^C, g^C\rangle$ is an algebra for a module, that is, that the tuple f^C and g^C satisfy the condition (3) from Definition 21:

$$\begin{aligned}
g^C \cdot Sf^C &= c \cdot g^B \cdot Ss \cdot Sc \cdot Sf^B \cdot SMs && \text{(def.)}\\
&= c \cdot g^B \cdot Sh_1 \cdot St \cdot Sf^B \cdot SMs && \text{(diag. (3))}\\
&= c \cdot h_1 \cdot g^A \cdot St \cdot Sf^B \cdot SMs && \text{(h_1 homomorph.)}\\
&= c \cdot h_0 \cdot g^A \cdot St \cdot Sf^B \cdot SMs && \text{(c coequaliser)}\\
&= c \cdot g^B \cdot Sh_0 \cdot St \cdot Sf^B \cdot SMs && \text{(h_0 homomorph.)}\\
&= c \cdot g^B \cdot Sf^B \cdot SMs && \text{(diag. (3))}\\
&= c \cdot g^B \cdot \overrightarrow{\mu}_B \cdot SMs && \text{(coherence)}\\
&= c \cdot g^B \cdot Ss \cdot \overrightarrow{\mu}_C && \text{($\overrightarrow{\mu}$ nat.)}\\
&= g^C \cdot \overrightarrow{\mu}_C && \text{(def.)}
\end{aligned}$$

◀

5 Summary and future work

In this paper, we have taken a closer look at the notion of module over a monad, focusing mainly on right modules. We illustrate our results with a number of examples, some of them new, some just being a reformulation in the language of modules of previously known results.

One important application we hope for is in functional programming, where structures similar to resumptions appear in the form of streaming I/O libraries [22] and adaptation of algebraic effects [31]. Corollary 20 and Theorem 25 give universal properties of the monad MS^*, which can be used for equational reasoning about programs that utilise them [24]. Providing a simple description of an adjunction that gives rise to MS^* gives us a more efficient implementation via the codensity monad trick [17].

Another question is how modules relate to monadic effects, especially their algebraic presentations, extensively studied by Plotkin and Power [29], and handlers, in the sense of Plotkin and Pretnar [30]. An algebraic theory induces a monad M as the family of its free models, while handlers are given by other models, that is, Eilenberg–Moore algebras of the monad M. As mentioned in Example 5, every Eilenberg–Moore algebra is a module for a constant endofunctor, but there are families of models that are parametric in variables (for example, the free model). These can be modelled by general left modules.

As suggested by Example 3, the actions of right modules may represent functions that run the computations in some context. In these case, the context is a global state A; recall the types: $A \times (A \times X)^A \to A \times X$ and $A \times X^A \to A \times X$. Is this a more general situation? Below, we give another example:

▶ **Example 26.** Consider the monad T of binary trees on **Set** with variables in leaves and the monad multiplication given by substitution. Intuitively, we interpret them as choice trees of randomised computations, in which every choice is equally probable. The context of execution is given by $CX = X \times \mathbb{B}^\omega$, where $\mathbb{B}^\omega$ is the set of infinite binary streams, representing possible future sequences of coin tosses. Now, to run the computation in the context, we go down the tree, choosing a branch based on the front element of the stream (left upon 0, right upon 1), at each step discarding the front element. Then, the result of $\overrightarrow{\mu}_X : TX \times \{0,1\}^\omega \to X \times \mathbb{B}^\omega$ is a pair consisting of the variable in the leaf that is reached by going down the tree as specified in the prefix of an appropriate length paired with the unused 'tail' of the stream.

6 Related work

The research presented in this paper is inspired by our previous work [28], in which the *coinductive* resumption monad MS^∞ was studied, where S^∞ is the free completely iterative monad [3] defined as $S^\infty A = \nu X.SX + A$. There, we use an arbitrary right module S instead of GM mainly to simplify the presentation and the proofs, although the main result considers the monad $M(GM)^\infty$, which is similar to Moggi's monad.

Modules are used by Adámek, Milius, and Velebil [2, 25] to capture the notion of guardedness in their study of iterative monads. They define an *idealised* monad as a right M-module S together with a suitably coherent natural transformation $\sigma : S \to M$. The general intuition for idealised monads is that S is a 'subset' (especially if σ is monic) of computations that have some good properties, which are retained after composing with any other computation. For instance, consider Example 4, in which the 'ideal' of the non-empty list monad is given by lists of length at least n. An important example of idealised monads are *ideal* monads, which are defined by the property $M = S + \mathsf{Id}$; see also Ghani and Uustalu [15] for an extended discussion.

There are some obvious generalisations possible. For example, we can allow S to be a functor to a different category. This definition was used by Street [32] to define the Eilenberg–Moore object in a 2-category: it is a universal left module (in the generalised sense). Hirschowitz and Maggesi [18, 19] and Ahrens [4] use generalised left modules to capture the construction of higher-order syntax and semantics. They, too, discuss the elementary theory of modules, but from a slightly different angle: instead of **Mod**, they study the category of modules over a single monad M, that is, a fibre of **Mod** with respect to the functor $\mathbf{Mod} \to \mathscr{C}^\mathscr{C}$ that extracts the functor part of a module. This functor has some nice properties: it has a left adjoint given by $G \mapsto GM$ and reflects (co)limits, see Example 2 (5).

Resumptions were introduced by Milner [26] to capture the semantics of concurrency (see also Abramsky [1]). In programming, resumptions (known also as 'trampolined style' [14] or 'engines' [11, 16]) were first used to control program flow. The first use of resumptions (although, of course, not explicitly named so) was probably the famous result on structured programming by Böhm and Jacopini [9].

Acknowledgements. We would like to thank the Algebra of Programming group at Oxford University for discussions; the anonymous referees for their constructive comments, which helped us to improve this paper; and the EPSRC, whose grant on *Unifying Theories of Generic Programming* (EP/J010995/1) partially supported this work.

References

1 Samson Abramsky. Retracing some paths in process algebra. In Ugo Montanari and Vladimiro Sassone, editors, *CONCUR*, volume 1119 of *Lecture Notes in Computer Science (LNCS)*, pages 1–17. Springer, 1996.

2 Jiří Adámek, Stefan Milius, and Jiří Velebil. On rational monads and free iterative theories. *Electronic Notes in Theoretical Computer Science*, 69:23–46, 2002.

3 Jiří Adámek, Stefan Milius, and Jiří Velebil. Free iterative theories: A coalgebraic view. *Mathematical Structures in Computer Science*, 13(2):259–320, 2003.

4 Benedikt Ahrens. Modules over relative monads for syntax and semantics. *Mathematical Structures in Computer Science*, FirstView:1–35, 12 2014.

5 Robert Atkey, Neil Ghani, Bart Jacobs, and Patricia Johann. Fibrational induction meets effects. In Lars Birkedal, editor, *Foundations of Software Science and Computational Structures*, volume 7213 of *Lecture Notes in Computer Science (LNCS)*, pages 42–57. Springer, 2012.

6 Roland Carl Backhouse, Marcel Bijsterveld, Rik van Geldrop, and Jaap van der Woude. Categorical fixed point calculus. In David H. Pitt, David E. Rydeheard, and Peter Johnstone, editors, *Category Theory and Computer Science*, volume 953 of *Lecture Notes in Computer Science (LNCS)*, pages 159–179. Springer, 1995.

7 Michael Barr and Charles F. Wells. *Toposes, Triples, and Theories*. Grundlehren der mathematischen Wissenschaften. Springer-Verlag, 1985.

8 Jonathan M. Beck. Distributive laws. In *Seminar on Triples and Categorical Homology Theory*, volume 80 of *Lecture Notes in Mathematics*, pages 119–140. Springer Berlin / Heidelberg, 1969. 10.1007/BFb0083084.

9 Corrado Böhm and Giuseppe Jacopini. Flow diagrams, Turing machines and languages with only two formation rules. *Communications of the ACM*, 9(5):366–371, May 1966.

10 Eduardo J. Dubuc. *Kan extensions in enriched category theory*. Lecture Notes in Mathematics. Springer, Berlin, 1970.

11 R. Kent Dybvig and Robert Hieb. Engines from continuations. *Computer Languages*, 14(2):109–123, 1989.

12 Samuel Eilenberg and John C. Moore. Adjoint functors and triples. *Illinois Journal of Mathematics*, 9(3):381–398, 09 1965.

13 Andrzej Filinski and Kristian Støvring. Inductive reasoning about effectful data types. In *International Conference on Functional Programming*, pages 97–110. ACM, 2007.

14 Steven E. Ganz, Daniel P. Friedman, and Mitchell Wand. Trampolined style. In Didier Rémy and Peter Lee, editors, *International Conference on Functional Programming*, pages 18–27. ACM, 1999.

15 Neil Ghani and Tarmo Uustalu. Coproducts of ideal monads. *Theoretical Informatics and Applications*, 38(4):321–342, 2004.

16 Christopher T. Haynes and Daniel P. Friedman. Engines build process abstractions. In *LISP and Functional Programming*, pages 18–24, 1984.

17 Ralf Hinze. Kan extensions for program optimisation, or: Art and Dan explain an old trick. In Jeremy Gibbons and Pablo Nogueira, editors, *Mathematics of Program Construction – 11th International Conference, MPC 2012, Madrid, Spain, June 25-27, 2012. Proceedings*, volume 7342 of *Lecture Notes in Computer Science*, pages 324–362. Springer, 2012.

18 André Hirschowitz and Marco Maggesi. Modules over monads and linearity. In Daniel Leivant and Ruy J. G. B. de Queiroz, editors, *Workshop on Logic, Language, Information and Computation*, volume 4576 of *Lecture Notes in Computer Science (LNCS)*, pages 218–237. Springer, 2007.

19 André Hirschowitz and Marco Maggesi. Modules over monads and initial semantics. *Information and Computation*, 208(5):545–564, 2010.

20 Martin Hyland, Gordon D. Plotkin, and John Power. Combining effects: Sum and tensor. *Theoretical Computer Science*, 357(1-3):70–99, 2006.

21 Gregory M. Kelly. A unified treatment of transfinite constructions for free algebras, free monoids, colimits, associated sheaves, and so on. *Bulletin of the Australian Mathematical Society*, 22:1–83, 8 1980.

22 Oleg Kiselyov. Iteratees. In Tom Schrijvers and Peter Thiemann, editors, *Functional and Logic Programming*, volume 7294 of *Lecture Notes in Computer Science (LNCS)*, pages 166–181. Springer, 2012.

23 Saunders Mac Lane. *Categories for the Working Mathematician*. Graduate Texts in Mathematics. Springer, 1998.

24 Erik Meijer, Maarten M. Fokkinga, and Ross Paterson. Functional programming with bananas, lenses, envelopes and barbed wire. In John Hughes, editor, *Functional Programming Languages and Computer Architecture*, volume 523 of *Lecture Notes in Computer Science*, pages 124–144. Springer, 1991.

25 Stefan Milius. Completely iterative algebras and completely iterative monads. *Information and Computation*, 196:1–41, 2005.

26 Robin Milner. Processes: A mathematical model of computing agents. In *Logic Colloquium 1973*, Studies in logic and the foundations of mathematics, pages 157–173. North-Holland Pub. Co., 1975.

27 Eugenio Moggi. An abstract view of programming languages. Technical report, Edinburgh University, 1989.

28 Maciej Piróg and Jeremy Gibbons. The coinductive resumption monad. *Electronic Notes in Theoretical Computer Science*, 308:273–288, 2014. Mathematical Foundations of Programming Semantics (MFPS XXX).

29 Gordon D. Plotkin and A. John Power. Computational effects and operations: An overview. *Electronic Notes in Theoretical Computer Science*, 73:149–163, 2004.

30 Gordon D. Plotkin and Matija Pretnar. Handling algebraic effects. *Logical Methods in Computer Science*, 9(4), 2013.

31 Tom Schrijvers, Nicolas Wu, Benoit Desouter, and Bart Demoen. Heuristics entwined with handlers combined. In *PPDP 2014: Proceedings of the 16th International Symposium on Principles and Practice of Declarative Programming*. Association for Computing Machinery (ACM), 2014.

32 Ross Street. The formal theory of monads. *Journal of Pure and Applied Algebra*, 2(2):149–168, 1972.

33 Miki Tanaka. *Pseudo-Distributive Laws and a Unified Framework for Variable Binding*. PhD thesis, University of Edinburgh, 2005.

Revisiting the Institutional Approach to Herbrand's Theorem

Ionuţ Ţuţu[1,2] and José Luiz Fiadeiro[1]

1 Department of Computer Science, Royal Holloway University of London, UK
ittutu@gmail.com, jose.fiadeiro@rhul.ac.uk

2 Institute of Mathematics of the Romanian Academy, Research group of the project ID-3-0439, Romania

Abstract

More than a decade has passed since Herbrand's theorem was first generalized to arbitrary institutions, enabling in this way the development of the logic-programming paradigm over formalisms beyond the conventional framework of relational first-order logic. Despite the mild assumptions of the original theory, recent developments have shown that the institution-based approach cannot capture constructions that arise when service-oriented computing is presented as a form of logic programming, thus prompting the need for a new perspective on Herbrand's theorem founded instead upon a concept of generalized substitution system. In this paper, we formalize the connection between the institution- and the substitution-system-based approach to logic programming by investigating a number of features of institutions, like the existence of a quantification space or of representable substitutions, under which they give rise to suitable generalized substitution systems. Building on these results, we further show how the original institution-independent versions of Herbrand's theorem can be obtained as concrete instances of a more general result.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases Institution theory, Substitution systems, Herbrand's theorem

Digital Object Identifier 10.4230/LIPIcs.CALCO.2015.304

1 Introduction

The Fundamental Theorem of Herbrand [15] is a central result in proof theory that deals with the reduction of provability in first-order logic to provability in propositional logic. Its importance in the context of automated theorem proving was realized in the early 1960s, when, in combination with the theory of Horn-clause logic, it played a key role in establishing the mathematical foundations of logic programming (see e.g. [16]). In the conventional setting of relational first-order logic, Herbrand's theorem states that, for a set Γ of Horn clauses (i.e. for a logic program Γ), the answers to an existential query can be found simply by examining a term model – called the (least) Herbrand model – instead of all the models that satisfy Γ. Over the last three decades, the original result has been generalized to a variety of other logical systems, including Horn-clause logic with equality [13, 14], hidden algebra [12], and category-based constraint logic [3], culminating in [5] with an investigation of Herbrand's theorem in an arbitrary institution [11] – a categorical formalization of the intuitive notion of logical system put forward by Goguen and Burstall in the late 1970s.

The results presented in [5] are grounded on an institution-independent treatment of variables as signature morphisms (which correspond in concrete cases to extensions of signatures with new constant-operation symbols) that was first outlined in [24]. This enabled the development of fundamental semantic concepts to logic programming like query

6th International Conference on Algebra and Coalgebra in Computer Science (CALCO'15).
Editors: Lawrence S. Moss and Pawel Sobocinski; pp. 304–319
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

and solution in arbitrary institutions. Logic programs, for example, are given by theory presentations (usually universal Horn presentations), while queries are captured through existential quantifications of basic sentences. We will recall these concepts together with their corresponding properties one step at a time in the subsequent sections of this paper.

Thanks to its generality, the institution-based approach to Herbrand's theorem enabled the development of logic programming over a wide range of logical systems (see e.g. [10], and also [6]). All the same, certain institution-based forms of logic programming do not fit into the framework proposed in [5]. In particular, the logic-programming semantics of services [30] is grounded on a family of logical systems for which the concept of variable cannot be faithfully captured by means of representable extensions of signatures, thus failing to meet one of the most basic assumptions of the institution-independent variant of the theorem. This led us to advance in [29] a new abstract approach to logic programming and, implicitly, to Herbrand's theorem over a concept of generalized substitution system that extends institutions by allowing for direct representations of variables and substitutions – similarly to the context institutions of [23], though the latter are concrete, in the sense that the category of models of every signature is concrete over the category of indexed sets.

In the present paper, we continue the work reported in [29] with an investigation of the relationship between the institution-based and the substitution-system-based approach to logic programming. More specifically, we show that the hypotheses of the latter are indeed more general by examining the role of representability (of signature morphisms in arbitrary institutions) in the construction of a generalized substitution system. The main challenge here lies in the treatment of substitutions, which, in the institutional setting, are captured purely through their corresponding translations of sentences and reductions of models. This prevents us from using the additional information available when substitutions are regarded as mappings from variables to terms – which is only possible, however, in concrete examples of institutions such as first-order logic – thus making it difficult to translate substitutions along signature morphisms. For this reason, the main contributions of our paper, namely the derivation of a generalized substitution system from a given institution and the reformulation of the original institution-independent variants of Herbrand's theorem in the resulting framework, are parameterized by a class of general substitutions.

The paper is organized as follows: in Section 2 we review the concept of generalized substitution system and two well-known formalisms that have been studied in the context of institution-based logic-programming languages; in Section 3 we examine a class of substitution systems whose variables are defined through extensions of signatures (of a given institution), and whose substitutions correspond to the institution-independent notion of substitution; building on these results, in Section 4 we further investigate the translation of variables along signature morphisms and identify a set of sufficient conditions under which an institution can give rise to a generalized substitution system; lastly, in Section 5 we present a different perspective on the institution-independent versions of Herbrand's fundamental theorem.

2 Technical preliminaries

We generally assume familiarity with the theory of institutions, including its categorical underpinnings and the presentation of institutions as functors into the category $\mathbb{R}\mathrm{oom}$ of rooms and corridors (see, for example, the monographs [6, 25]). In terms of category-theoretic notations, we denote by $|\mathbb{C}|$ the collection of objects of a category $\mathbb{C}$, by $f \mathbin{⨟} g$ the composition of arrows f and g in diagrammatic order, and by 1_A the identity arrow of an object A.

Our work makes extensive use of comma categories. To this end, for any object A of a category $\mathbb{C}$, we denote the comma category of $\mathbb{C}$-objects under A by $A \,/\, \mathbb{C}$ and the forgetful functor $A \,/\, \mathbb{C} \to \mathbb{C}$ by $|\_|_A$. We also denote by $\mathbb{C}^{\to}$ the category of $\mathbb{C}$-arrows, and by *dom* the canonical projection functor $\mathbb{C}^{\to} \to \mathbb{C}$ that maps the arrows $f \colon A \to B$ in $\mathbb{C}$ to their domain.

2.1 Generalized substitution systems

Substitution systems are the most basic structures that underlie both the denotational and the operational semantics of the logic-independent approach to logic programming proposed in [29]. Since their definition relies technically on the category $\mathbb{R}\text{oom}$, we start by recalling that a *room* is a triple $\langle S, \mathbb{M}, \vDash \rangle$ consisting of a set S of *sentences*, a category $\mathbb{M}$ of *models*, and a *satisfaction relation* $\vDash \subseteq |\mathbb{M}| \times S$. Furthermore, a *corridor* (i.e. a morphism of rooms) $\langle \alpha, \beta \rangle \colon \langle S, \mathbb{M}, \vDash \rangle \to \langle S', \mathbb{M}', \vDash' \rangle$ is defined by a *sentence-translation function* $\alpha \colon S \to S'$ and a *model-reduction functor* $\beta \colon \mathbb{M}' \to \mathbb{M}$ such that, for all $M' \in |\mathbb{M}'|$ and $\rho \in S$,

$$M' \vDash' \alpha(\rho) \qquad \text{if and only if} \qquad \beta(M') \vDash \rho.$$

The following definitions originate from [29].

▶ **Definition 1** (Substitution system)**.** A *substitution system* is a triple $\langle \mathbb{S}\text{ubst}, G, \mathcal{S} \rangle$, usually denoted simply by $\mathcal{S}$, that consists of

- a category $\mathbb{S}\text{ubst}$ of (*abstract*) *signatures of variables* and *substitutions*,
- a room G of *ground sentences* and *models*, and
- a functor $\mathcal{S} \colon \mathbb{S}\text{ubst} \to G \,/\, \mathbb{R}\text{oom}$ defining, for every signature of variables X, the corridor $\mathcal{S}(X) \colon G \to G(X)$ from G to the room $G(X)$ of X*-sentences* and X*-models*.

A classical example can be obtained by defining $\mathbb{S}\text{ubst}$ as the category of (sets of) variables and substitutions over a fixed first-order signature Σ. In that case, G is the room of those sentences of Σ that are ground (i.e. without variables), and the corridors $\mathcal{S}(X)$ correspond to the signature morphisms that extend Σ by adding the variables of X as new constants.

Generalized substitution systems can be regarded as extensions of substitution systems that are parameterized by the signature used. In this sense, the connection between generalized substitutions systems and substitution systems is similar to that between institutions and rooms: generalized substitutions systems are functors into the category $\mathbb{S}\text{ubst}\mathbb{S}\text{ys}$ of substitutions systems. To make this definition more precise, we recall from [29] that a *morphism of substitution systems* between $\mathcal{S} \colon \mathbb{S}\text{ubst} \to G \,/\, \mathbb{R}\text{oom}$ and $\mathcal{S}' \colon \mathbb{S}\text{ubst}' \to G' \,/\, \mathbb{R}\text{oom}$ is a triple $\langle \Psi, \kappa, \tau \rangle$, where Ψ is a functor $\mathbb{S}\text{ubst} \to \mathbb{S}\text{ubst}'$, κ is a corridor $G \to G'$, and τ is a natural transformation $\mathcal{S} \Rightarrow \Psi \,\mathbin{⨾}\, \mathcal{S}' \,\mathbin{⨾}\, (\kappa \,/\, \mathbb{R}\text{oom})$.

▶ **Definition 2** (Generalized substitution system)**.** A *generalized substitution system* is a pair $\langle \mathbb{S}\text{ig}, \mathcal{GS} \rangle$ given by a category $\mathbb{S}\text{ig}$ of *signatures* and a functor $\mathcal{GS} \colon \mathbb{S}\text{ig} \to \mathbb{S}\text{ubst}\mathbb{S}\text{ys}$.

2.2 Equational logic programming

Before we embark on the study of institution-based abstract logic-programming languages, let us briefly survey the logical systems that underlie two of the most prominent examples of (concrete) logic-programming languages examined in the context of institutions: first-order and higher-order equational logic programming (see e.g. [13, 19], and also [22]). These will form the main reference points that we will use to illustrate the various concepts and properties discussed in the subsequent sections of our paper.

First-order equational logic

First-order equational logic programming is defined over the quantifier-free fragment of many-sorted first-order equational logic, whose institution we denote by QF-$\underline{\mathrm{FOL}_=}$. In what follows, we only give a succinct presentation of the most important first-order concepts needed for the purpose of our work. A more in-depth discussion of QF-$\underline{\mathrm{FOL}_=}$ can be found, for example, in [14, 29], or in the recent monographs [6, 25].

Signatures. The *signatures* of QF-$\underline{\mathrm{FOL}_=}$ are pairs $\langle S, F \rangle$, where S is a (finite) set of *sorts* and F is a family $(F_{w \to s})_{w \in S^*, s \in S}$ of (finite) sets of *operation symbols* indexed by *arities* and *sorts*. *Signature morphisms* $\varphi \colon \langle S, F \rangle \to \langle S', F' \rangle$ are defined by functions $\varphi^{\mathrm{st}} \colon S \to S'$ between the sets of sorts and by families of functions $\varphi^{\mathrm{op}}_{w \to s} \colon F_{w \to s} \to F'_{\varphi^{\mathrm{st}}(w) \to \varphi^{\mathrm{st}}(s)}$, for $w \in S^*$ and $s \in S$, between the sets of operation symbols.

Sentences, models, and the satisfaction relation. For every signature $\langle S, F \rangle$ and every sort $s \in S$, the set $\mathrm{T}_{F,s}$ of F*-terms* of sort s is the least set such that $\sigma(t_1, \ldots, t_n) \colon s \in \mathrm{T}_{F,s}$ for all $\sigma \in F_{s_1 \cdots s_n \to s}$ and $t_i \in \mathrm{T}_{F,s_i}$. The *sentences* over $\langle S, F \rangle$ are built from *equational atoms* $l = r$, where $l, r \in \mathrm{T}_{F,s}$ for some $s \in S$, by iteration of the usual Boolean connectives.

The *models*, or *algebras*, M of $\langle S, F \rangle$ interpret each sort $s \in S$ as a set M_s, called the *carrier* of s in M, and each operation symbol $\sigma \in F_{s_1 \cdots s_n \to s}$ as a function $M_\sigma \colon M_{s_1} \times \cdots \times M_{s_n} \to M_s$. *Homomorphisms* $h \colon M_1 \to M_2$ are families of functions $(h_s \colon M_{1,s} \to M_{2,s})_{s \in S}$ such that $h_s(M_{1,\sigma}(m_1, \ldots, m_n)) = M_{2,\sigma}(h_{s_1}(m_1), \ldots, h_{s_n}(m_n))$ for all $\sigma \in F_{s_1 \cdots s_n \to s}$ and $m_i \in M_{s_i}$.

Finally, the *satisfaction relation* is defined by induction on the structure of sentences, based on the evaluation of terms in models. For instance, an $\langle S, F \rangle$-model M satisfies an equational atom $l = r$ if and only if the terms l and r yield the same value in M.

Higher-order logic with Henkin semantics

Following the lines of [20], and also of more recent institution-theoretic works such as [25, 28], we define and study higher-order logic programming over a simplified version of higher-order logic with Henkin semantics that only takes into account λ-free terms. This does not limit the expressive power of the logic since for any term $\lambda(x \colon s).\, t$ one can define a new constant σ and a universal sentence of the form $\forall \{x \colon s\} \cdot \sigma\, x = t$.[1] Similarly to first-order equational logic programming, for the results presented in the following sections it suffices to consider the quantifier-free fragment of higher-order logic, whose institution we denote by QF-$\underline{\mathrm{HNK}}$.[2]

Signatures. A *higher-order signature* consists of a set S of *basic types*, or *sorts*, and a family $(F_s)_{s \in \vec{S}}$ of sets of *constant-operation symbols* indexed by S*-types*, where $\vec{S}$ is the least set for which $S \subseteq \vec{S}$ and $s_1 \to s_2 \in \vec{S}$ whenever $s_1, s_2 \in \vec{S}$. *Signature morphisms* $\varphi \colon \langle S, F \rangle \to \langle S', F' \rangle$ comprise functions $\varphi^{\mathrm{st}} \colon S \to \vec{S}'$ and $\varphi^{\mathrm{op}}_s \colon F_s \to F'_{\varphi^{\mathrm{type}}(s)}$, for $s \in \vec{S}$, where $\varphi^{\mathrm{type}} \colon \vec{S} \to \vec{S}'$ is the canonical extension of φ^{st} given by $\varphi^{\mathrm{type}}(s_1 \to s_2) = \varphi^{\mathrm{type}}(s_1) \to \varphi^{\mathrm{type}}(s_2)$.

Sentences, models, and the satisfaction relation. Given a signature $\langle S, F \rangle$, the family $\big(\mathrm{T}_{F,s}\big)_{s \in \vec{S}}$ of F*-terms* is the least family of sets such that $\sigma \colon s \in \mathrm{T}_{F,s}$ for all $s \in \vec{S}$

[1] A detailed presentation of this encoding, formalized as an institution comorphism, can be found in [9].

[2] Note that the universal sentences needed for encoding λ-terms can still be defined as Horn clauses of the logic programs under consideration.

and $\sigma \in F_s$, and $(t\,t_1) \in \mathrm{T}_{F,s_2}$ for all terms $t \in \mathrm{T}_{F,s_1\to s_2}$ and $t_1 \in \mathrm{T}_{F,s_1}$. As in the case of QF-$\underline{\mathrm{FOL}_=}$, the *sentences* over $\langle S, F\rangle$ are built from *equational atoms* $l = r$, where l and r are terms in $\mathrm{T}_{F,s}$ for some type $s \in \vec{S}$, by repeated applications of the Boolean connectives.

The *models* M of a higher-order signature $\langle S, F\rangle$ interpret the types $s \in \vec{S}$ as sets M_s, the constant symbols $\sigma \in F_s$ as elements $M_\sigma \in M_s$, and define injective maps $[\![\_]\!]^M_{s_1\to s_2} \colon M_{s_1\to s_2} \to [M_{s_1} \to M_{s_2}]$, where $[M_{s_1} \to M_{s_2}]$ denotes the set of functions from M_{s_1} to M_{s_2}, for any two types $s_1, s_2 \in \vec{S}$. *Model homomorphisms* $h\colon M_1 \to M_2$ are families of maps $(h_s \colon M_{1,s} \to M_{2,s})_{s\in\vec{S}}$ such that $h_s(M_{1,\sigma}) = M_{2,\sigma}$ for every type $s \in \vec{S}$ and operation $\sigma \in F_s$, and $[\![f]\!]^{M_1}_{s_1\to s_2} \mathbin{⨾} h_{s_2} = h_{s_1} \mathbin{⨾} [\![h_{s_1\to s_2}(f)]\!]^{M_2}_{s_1\to s_2}$ for all $s_1, s_2 \in \vec{S}$ and $f \in M_{1,s_1\to s_2}$.

The *satisfaction relation* relies once again on the interpretation of terms in models, which extends the interpretation of constant-operation symbols as follows: for every $\langle S, F\rangle$-model M and every pair of terms $t \in \mathrm{T}_{F,s_1\to s_2}$ and $t_1 \in \mathrm{T}_{F,s_1}$, $M_{(t\,t_1)} = [\![M_t]\!]^M_{s_1\to s_2}(M_{t_1})$.

3 Institution-independent substitutions

The institution-independent concept of substitution (see [5], and also [6]) generalizes first-order substitutions (as well as second-order and higher-order substitutions, among others) to arbitrary institutions by taking notice only of their syntactic and semantic effects: the translations of sentences and the reductions of models that they generate. A key step in arriving at this notion is the presentation of the extensions of signatures by sets of variables as particular cases of signature morphisms (along the lines of [24]). Thus, for any two signature morphisms $\chi_1 \colon \Sigma \to \Sigma_1$ and $\chi_2 \colon \Sigma \to \Sigma_2$ (two extensions of Σ) in an institution $\mathcal{I} = \langle \mathbb{S}\mathrm{ig}, \mathrm{Sen}, \mathrm{Mod}, \models\rangle$, a *substitution* $\psi\colon \chi_1 \to \chi_2$ is a pair $\langle \mathrm{Sen}_\Sigma(\psi), \mathrm{Mod}_\Sigma(\psi)\rangle$ given by

- a sentence-translation function $\mathrm{Sen}_\Sigma(\psi) \colon \mathrm{Sen}(\Sigma_1) \to \mathrm{Sen}(\Sigma_2)$ and
- a model-reduction functor $\mathrm{Mod}_\Sigma(\psi) \colon \mathrm{Mod}(\Sigma_2) \to \mathrm{Mod}(\Sigma_1)$

that preserve Σ, in the sense that $\mathrm{Sen}(\chi_1) \mathbin{⨾} \mathrm{Sen}_\Sigma(\psi) = \mathrm{Sen}(\chi_2)$ and $\mathrm{Mod}_\Sigma(\psi) \mathbin{⨾} \mathrm{Mod}(\chi_1) = \mathrm{Mod}(\chi_2)$, and satisfy the following condition:

$$M_2 \models_{\Sigma_2} \mathrm{Sen}_\Sigma(\psi)(\rho_1) \qquad \text{if and only if} \qquad \mathrm{Mod}_\Sigma(\psi)(M_2) \models_{\Sigma_1} \rho_1$$

for every Σ_2-model M_2 and every Σ_1-sentence ρ_1.

In this work, we take into consideration an equivalent formulation of the original definition that makes use of the category $\mathbb{R}\mathrm{oom}$ of rooms and corridors. In addition, we extend the fact that substitutions inherit the composition of their components – thus giving rise to a category – to derive a general substitution system of Σ-substitutions for each signature Σ.

▶ **Proposition 3.** *Let $\mathcal{Q} \subseteq \mathbb{S}\mathrm{ig}$ be a class of signature morphisms of an institution $\mathcal{I}$ regarded as a functor $\mathbb{S}\mathrm{ig} \to \mathbb{R}\mathrm{oom}$. For every $\mathcal{I}$-signature Σ we obtain a substitution system $\mathcal{SI}^{\mathcal{Q}}_\Sigma \colon \mathbb{S}\mathrm{ubst}^{\mathcal{Q}}_\Sigma \to \mathcal{I}(\Sigma) \,/\, \mathbb{R}\mathrm{oom}$ defined as follows:*

- *The objects of the category $\mathbb{S}\mathrm{ubst}^{\mathcal{Q}}_\Sigma$ – i.e. the* signatures of Σ-variables *– are signature morphisms $\chi\colon \Sigma \to \Sigma(\chi)$*[3] *belonging to the class $\mathcal{Q}$. Their corresponding corridors via the functor $\mathcal{SI}^{\mathcal{Q}}_\Sigma$ are given simply by $\mathcal{I}(\chi)\colon \mathcal{I}(\Sigma) \to \mathcal{I}(\Sigma(\chi))$.*

[3] For convenience, we denote the codomain of the signature morphism χ by $\Sigma(\chi)$; this reflects the intuition that $\Sigma(\chi)$ is an extension of the signature Σ with variables defined by χ.

- *For every two signatures of* Σ*-variables* $\chi_1 \colon \Sigma \to \Sigma(\chi_1)$ *and* $\chi_2 \colon \Sigma \to \Sigma(\chi_2)$*, a* Σ-substitution $\psi \colon \chi_1 \to \chi_2$ *(i.e. a morphism in* $\mathbb{S}\mathrm{ubst}^{\mathcal{Q}}_{\Sigma}$*) consists of a corridor* $\langle \mathrm{Sen}_{\Sigma}(\psi), \mathrm{Mod}_{\Sigma}(\psi) \rangle$ *between* $\mathcal{I}(\Sigma(\chi_1))$ *and* $\mathcal{I}(\Sigma(\chi_2))$ *such that* $\mathcal{I}(\chi_1) \mathbin{⨟} \langle \mathrm{Sen}_{\Sigma}(\psi), \mathrm{Mod}_{\Sigma}(\psi) \rangle = \mathcal{I}(\chi_2)$*.*

$$\begin{tikzcd}
\mathcal{I}(\Sigma(\chi_1)) \arrow[rr, bend right, "{\langle \mathrm{Sen}_{\Sigma}(\psi), \mathrm{Mod}_{\Sigma}(\psi) \rangle}"'] & \mathcal{I}(\Sigma) \arrow[l, "\mathcal{I}(\chi_1)"'] \arrow[r, "\mathcal{I}(\chi_2)"] & \mathcal{I}(\Sigma(\chi_2))
\end{tikzcd}$$

As such, Σ*-substitutions are merely arrows in the comma category* $\mathcal{I}(\Sigma) \,/\, \mathbb{R}\mathrm{oom}$*, meaning that they are identified with their images under the functor* $\mathcal{SI}^{\mathcal{Q}}_{\Sigma}$*. The composition of substitutions is defined accordingly.*

▶ **Example 4.** In QF-$\underline{\mathrm{FOL}}_{=}$, a *variable* over a signature $\langle S, F \rangle$ is a triple $(x, s, F_{\varepsilon \to s})$,[4] often denoted simply by $x \colon s$, where x is the name of the variable and s is its sort. Thus, QF-$\underline{\mathrm{FOL}}_{=}$-signatures of $\langle S, F \rangle$-variables X are S-indexed families of sets X_s of variables of sort s such that different variables have different names. *First-order substitutions* $\psi \colon X \to Y$ can be further defined as S-indexed families of maps $\psi_s \colon X_s \to T_{F \cup Y, s}$ that assign a term over the extended signature $\langle S, F \cup Y \rangle$ to every variable of X.

One can easily check that first-order substitutions $\psi \colon X \to Y$ indeed give rise to general substitutions between $\langle S, F \rangle \subseteq \langle S, F \cup X \rangle$ and $\langle S, F \rangle \subseteq \langle S, F \cup Y \rangle$ (see e.g. [6]). For instance, the reduct $\mathrm{Mod}_{\langle S,F \rangle}(\psi)(N)$ of an $\langle S, F \cup Y \rangle$-model N is the $\langle S, F \cup X \rangle$-expansion of $N{\upharpoonright}_{\langle S,F \rangle}$ given by $\mathrm{Mod}_{\langle S,F \rangle}(\psi)(N)_{x \colon s} = N_{\psi(x)}$ for every variable $x \colon s$ of X. Note, however, that not every general substitution between $\langle S, F \rangle \subseteq \langle S, F \cup X \rangle$ and $\langle S, F \rangle \subseteq \langle S, F \cup Y \rangle$ corresponds to a first-order substitution; we will discuss this aspect to a greater extent in Section 4.2.

Higher-order signatures of variables and *substitutions* can be defined likewise, by recalling that a *higher-order variable* over a signature $\langle S, F \rangle$ is a triple of the form (x, s, F_s), where x and s correspond to the name and the type of the variable (see e.g. [28], and also [2]). As expected, this means that we allow higher-order variables to range over arbitrary functions.

4 Quantification spaces

Due to its mild assumptions, the construction outlined in Propostition 3 cannot be easily generalized to accommodate signature morphisms. More precisely, one cannot guarantee that signature morphisms $\varphi \colon \Sigma \to \Sigma'$ lead to adequate morphisms between the substitution systems associated with Σ and Σ': it would suffice, for example, to consider a class $\mathcal{Q}$ of signature morphisms that consists only of extensions of Σ, thus preventing the translation of the signatures of Σ-variables along φ. To overcome this limitation, we take into account only those extensions of signatures that belong to a quantification space – a notion introduced in [7] in the context of quasi-Boolean encodings[5] and utilized in a series of papers on hybridization and many-valued institutions (see e.g. [17, 8]). For the purpose of our work, it will be convenient to consider a more categorical formulation of the original definition of quantification spaces, based on Fact 5 below.

▶ **Fact 5.** *Consider a category* $\mathbb{C}$ *and a subcategory* $\mathbb{Q}$ *of the category* $\mathbb{C}^{\to}$ *of* $\mathbb{C}$*-arrows. The domain functor dom:* $\mathbb{Q} \to \mathbb{C}$ *gives rise to a natural transformation* $\iota_{\mathbb{Q}} \colon (\_ \,/\, \mathbb{Q}) \Rightarrow$

[4] We denote the empty arity by ε; hence, $F_{\varepsilon \to s}$ is the set of *constants* of sort s of the signature $\langle S, F \rangle$.

[5] It should be noted, however, that the ideas that underlie quantification spaces can be traced back to [27] – one of the earliest works in which open formulae are treated in arbitrary institutions.

$dom^{\mathrm{op}} \fatsemi (\_ \,/\, \mathbb{C})$ where $(\_ \,/\, \mathbb{Q})\colon \mathbb{Q}^{\mathrm{op}} \to \mathbb{C}\mathrm{at}$ and $(\_ \,/\, \mathbb{C})\colon \mathbb{C}^{\mathrm{op}} \to \mathbb{C}\mathrm{at}$ are the canonical comma-category functors and, for every triple $\langle A_1, f, A_2\rangle$ in $|\mathbb{Q}|$ (i.e. arrow $f\colon A_1 \to A_2$ in $\mathbb{C}$), $\iota_{\mathbb{Q},f}\colon f \,/\, \mathbb{Q} \to A_1 \,/\, \mathbb{C}$ is the functor that maps the morphisms $\langle g_1, g_2\rangle\colon \langle A_1, f, A_2\rangle \to \langle A_1', f', A_2'\rangle$ in $\mathbb{Q}$ (corresponding to commutative squares in $\mathbb{C}$) to $g_1\colon A_1 \to A_1'$.

$$\begin{tikzcd}
A_1 \arrow[r, "g_1"] \arrow[d, "f"'] & A_1' \arrow[d, "f'"] & & & \\
& & \mapsto & A_1 \arrow[r, "g_1"] & A_1' \\
A_2 \arrow[r, "g_2"'] & A_2' & & &
\end{tikzcd}$$

▶ **Definition 6** (Quantification space)**.** For any institution $\mathcal{I}\colon \mathbb{S}\mathrm{ig} \to \mathbb{R}\mathrm{oom}$, a *quantification space* consists of a subcategory $\mathbb{Q}$ of $\mathbb{S}\mathrm{ig}^{\to}$ such that

1. every arrow in $\mathbb{Q}$ corresponds to a pushout in $\mathbb{S}\mathrm{ig}$, and
2. the transformation $\iota_{\mathbb{Q}}\colon (\_ \,/\, \mathbb{Q}) \Rightarrow dom^{\mathrm{op}} \fatsemi (\_ \,/\, \mathbb{S}\mathrm{ig})$ is a natural isomorphism.

This means that for every extension of signatures $\chi\colon \Sigma \to \Sigma(\chi)$ in $|\mathbb{Q}|$ and every signature morphism $\varphi\colon \Sigma \to \Sigma'$ there exist a unique extension $\chi'\colon \Sigma' \to \Sigma'(\chi')$ in $|\mathbb{Q}|$ and a unique signature morphism $\phi\colon \Sigma(\chi) \to \Sigma'(\chi')$ such that the pair $\langle \varphi, \phi\rangle$ defines a morphism in $\mathbb{Q}$ between the arrows χ and χ'.[6] We will henceforth denote the signature extension χ' and the signature morphism ϕ by $\chi^{\varphi}\colon \Sigma' \to \Sigma'(\chi^{\varphi})$ and $\varphi^{\chi}\colon \Sigma(\chi) \to \Sigma'(\chi^{\varphi})$, respectively.

$$\begin{tikzcd}
\Sigma \arrow[r, "\varphi"] \arrow[d, "\chi"'] & \Sigma' \arrow[d, "\chi^{\varphi}"] \\
\Sigma(\chi) \arrow[r, "\varphi^{\chi}"'] & \Sigma'(\chi^{\varphi})
\end{tikzcd}$$

▶ **Example 7.** For both $\textsc{qf}\text{-}\underline{\mathrm{FOL}_{=}}$ and $\textsc{qf}\text{-}\underline{\mathrm{HNK}}$, the extensions of signatures $\chi\colon \langle S, F\rangle \to \langle S, F \cup X\rangle$ defined by (families of) finite sets of first-order/higher-order $\langle S, F\rangle$-variables X form a quantification space. More precisely, for every signature morphism $\varphi\colon \langle S, F\rangle \to \langle S', F'\rangle$,

- $\chi^{\varphi}\colon \langle S', F'\rangle \to \langle S', F' \cup X^{\varphi}\rangle$ is the extension of $\langle S', F'\rangle$ given by the sets of variables $X^{\varphi}_{s'} = \{x\colon s' \mid x\colon s \in X_s$ for some sort $s \in S$ (or type $s \in \vec{S}$) such that $\varphi(s) = s'\}$, and
- $\varphi^{\chi}\colon \langle S, F \cup X\rangle \to \langle S', F' \cup X^{\varphi}\rangle$ is the canonical extension of φ that maps each $\langle S, F\rangle$-variable $x\colon s$ in X to the $\langle S', F'\rangle$-variable $x\colon \varphi(s)$ in X^{φ}.

▶ **Definition 8** (Adequacy)**.** For any institution, a quantification space $\mathbb{Q}$ is said to be *adequate* if every arrow $\langle \varphi, \varphi^{\chi}\rangle\colon \chi \to \chi^{\varphi}$ in $\mathbb{Q}$ corresponds to a model-amalgamation square: for every Σ'-model M' and $\Sigma(\chi)$-model N such that $M'\!\upharpoonright_{\varphi} = N\!\upharpoonright_{\chi}$ there exists a unique model N' of $\Sigma'(\chi^{\varphi})$ – the *amalgamation* of M' and N – such that $N'\!\upharpoonright_{\chi^{\varphi}} = M'$ and $N'\!\upharpoonright_{\varphi^{\chi}} = N$.

In semi-exact institutions[7] – like $\textsc{qf}\text{-}\underline{\mathrm{FOL}_{=}}$ (see [18]) – all quantification spaces are adequate. This is not the case of $\textsc{qf}\text{-}\underline{\mathrm{HNK}}$, for which it is known that, due to the presence of higher-order types, not every pushout square of signature morphisms is a model-amalgamation square (see e.g. [2]). Nonetheless, the quantification space for $\textsc{qf}\text{-}\underline{\mathrm{HNK}}$ outlined in Example 7 is adequate: the amalgamation N' of any two given models M' of $\langle S', F'\rangle$ and N of $\langle S, F \cup X\rangle$ is the unique χ^{φ}-expansion of M' that satisfies $N'_{x\colon \varphi(s)} = N_{x\colon s}$ for each variable $x\colon s$ in X.

▶ **Remark 9.** *Since the morphisms of any quantification space $\mathbb{Q}$ are required to form a category (by definition, $\mathbb{Q}$ is a subcategory of $\mathbb{S}\mathrm{ig}^{\to}$), for every signature extension $\chi\colon \Sigma \to \Sigma(\chi)$*

[6] Moreover, the signature morphisms χ' and ϕ correspond to a pushout of φ and χ.

[7] We recall that an institution is semi-exact if its model functor preserves pullbacks.

in $|\mathbb{Q}|$ *and every pair of composable signature morphisms* $\varphi\colon \Sigma \to \Sigma'$ *and* $\varphi'\colon \Sigma' \to \Sigma''$*, we have* $(\chi^{\varphi})^{\varphi'} = \chi^{\varphi \fatsemi \varphi'}$ *and* $\varphi^{\chi} \fatsemi (\varphi')^{\chi^{\varphi}} = (\varphi \fatsemi \varphi')^{\chi}$*. Moreover,* $\chi^{1_{\Sigma}} = \chi$ *and* $1_{\Sigma}^{\chi} = 1_{\Sigma(\chi)}$*.*

$$\begin{tikzcd}
\Sigma \arrow[r, "\varphi"] \arrow[d, "\chi"'] & \Sigma' \arrow[r, "\varphi'"] \arrow[d, "\chi^{\varphi}"] & \Sigma'' \arrow[d, "(\chi^{\varphi})^{\varphi'} = \chi^{\varphi \fatsemi \varphi'}"] \\
\Sigma(\chi) \arrow[r, "\varphi^{\chi}"] \arrow[rr, bend right, "(\varphi \fatsemi \varphi')^{\chi}"'] & \Sigma'(\chi^{\varphi}) \arrow[r, "(\varphi')^{\chi^{\varphi}}"] & \Sigma''((\chi^{\varphi})^{\varphi'})
\end{tikzcd}$$

Quantification spaces thus provide adequate support for translating abstract signature extensions along morphisms of signatures in a functorial manner.

4.1 Representable signature extensions

Since the institution-independent substitutions of Proposition 3 correspond to a semantic concept, we cannot expect to translate them along signature morphisms in the same manner as the extensions of signatures. The solution that we propose herein relies on an important characterization of the first-order signature extensions with new constant-operation symbols: for every QF-$\underline{\text{FOL}_{=}}$-signature extension $\langle S, F\rangle \subseteq \langle S, F \cup X\rangle$ there is a one-to-one correspondence between the models of $\langle S, F \cup X\rangle$ and the model homomorphisms defined on the free $\langle S, F\rangle$-algebra $\mathrm{T}_F(X)$ over the set of variables X; in particular, every $\langle S, F \cup X\rangle$-model N determines the homomorphism $h\colon \mathrm{T}_F(X) \to N{\upharpoonright_{\langle S,F\rangle}}$ given by $h(x) = N_x$ for every variable x in X. In this context, $\mathrm{T}_F(X)$ is said to be a *representation* of the inclusion of signatures $\langle S, F\rangle \subseteq \langle S, F \cup X\rangle$. The following definition originates from [4].

▶ **Definition 10** (Representable signature morphism)**.** In any institution, a signature morphism $\chi\colon \Sigma \to \Sigma(\chi)$ is *representable* if there exist a Σ-model M_{χ}, called the *representation* of χ, and an isomorphism of categories i_{χ} between $\mathrm{Mod}(\Sigma(\chi))$ and $M_{\chi} \,/\, \mathrm{Mod}(\Sigma)$ such that the following diagram commutes.

$$\begin{tikzcd}
\mathrm{Mod}(\Sigma) & \\
\mathrm{Mod}(\Sigma(\chi)) \arrow[u, "\_{\upharpoonright_{\chi}}"] \arrow[r, "i_{\chi}"'] & M_{\chi} \,/\, \mathrm{Mod}(\Sigma) \arrow[ul, "|\_|_{M_{\chi}}"']
\end{tikzcd}$$

Representable first-order signature morphisms were studied in depth in [26], from where we recall Proposition 11 below (see also [6]).

▶ **Proposition 11.** *A first-order signature morphism is representable if and only if it is bijective on all symbols, except constant-operation symbols.*

Consequently, all QF-$\underline{\text{FOL}_{=}}$-signature extensions with constants are representable.

A similar result can be obtained for QF-$\underline{\text{HNK}}$. In that case, however, the signature extensions with constants $\langle S, F\rangle \subseteq \langle S, F \cup X\rangle$ can only be guaranteed to be quasi-representable, in the sense that, for every $\langle S, F \cup X\rangle$-model N, the canonical functor $N \,/\, \mathrm{Mod}(S, F \cup X) \to N{\upharpoonright_{\langle S,F\rangle}} \,/\, \mathrm{Mod}(S, F)$ determined by the model-reduct functor $\_{\upharpoonright_{\langle S,F\rangle}}$ is an isomorphism (see, for example, [2] for more details). Representability further requires that the resulting higher-order signatures $\langle S, F \cup X\rangle$ have initial models, a property which holds whenever

$\langle S, F \cup X\rangle$ has at least one constant-operation symbol for each type.[8]

▶ **Remark 12.** *Let $\chi\colon \Sigma \to \Sigma(\chi)$ and $\chi'\colon \Sigma' \to \Sigma'(\chi')$ be a pair of representable signature extensions defined by a quantification space $\mathbb{Q}$, and let β and β' be two functors as depicted below such that $\mathrm{Mod}(\chi') \fatsemi \beta = \beta' \fatsemi \mathrm{Mod}(\chi)$.*

$$\begin{tikzcd}
\mathrm{Mod}(\Sigma) & & \mathrm{Mod}(\Sigma') & \\
\mathrm{Mod}(\Sigma(\chi)) \arrow[u, "-\upharpoonright_{\chi}"] \arrow[dr, "i_\chi"'] & & \mathrm{Mod}(\Sigma'(\chi')) \arrow[ll, "\beta'"'] \arrow[u, "-\upharpoonright_{\chi'}"] \arrow[dr, "i_{\chi'}"'] & \\
& M_\chi \,/\, \mathrm{Mod}(\Sigma) \arrow[uul, "\lvert\_\rvert_{M_\chi}"'] & & M_{\chi'} \,/\, \mathrm{Mod}(\Sigma') \arrow[ll, "U"] \arrow[uul, "\lvert\_\rvert_{M_{\chi'}}"']
\arrow[from=1-3, to=1-1, "\beta"']
\end{tikzcd}$$

The composition $i_{\chi'}^{-1} \fatsemi \beta' \fatsemi i_\chi$ gives rise to a functor U between the comma categories $M_{\chi'}$ / $\mathrm{Mod}(\Sigma')$ and M_χ / $\mathrm{Mod}(\Sigma)$, where

- *for every Σ'-model homomorphism $h'\colon M_{\chi'} \to M'$, $U(h')$ is the Σ-model homomorphism $(i_{\chi'}^{-1} \fatsemi \beta' \fatsemi i_\chi)(h')\colon M_\chi \to \beta(M')$, and*
- *for every arrow $f'\colon h'_1 \to h'_2$ between model homomorphisms $h'_1\colon M_{\chi'} \to M'_1$ and $h'_2\colon M_{\chi'} \to M'_2$, $U(f')$ is just the β-reduct of f', $\beta(f')\colon \beta(M'_1) \to \beta(M'_2)$.*

When β is the model-reduct functor $\mathrm{Mod}(\varphi)$ of a signature morphism $\varphi\colon \Sigma \to \Sigma'$, and when χ' and β' are χ^φ and $\mathrm{Mod}(\varphi^\chi)$, respectively, we will denote the functor $U\colon M_{\chi'} \,/\, \mathrm{Mod}(\Sigma') \to M_\chi \,/\, \mathrm{Mod}(\Sigma)$ by $U_{\varphi,\chi}$. Similarly, when β is the identity of $\mathrm{Mod}(\Sigma)$ and β' is the underlying model functor of a substitution $\psi\colon \chi \to \chi'$, we will denote the functor U by U_ψ.

Model homomorphisms $h'\colon M_{\chi'} \to M'$ can be regarded both as objects and as arrows (between $1_{M_{\chi'}}$ and h') in the comma category $M_{\chi'} \,/\, \mathrm{Mod}(\Sigma')$. In combination with the definition of $U\colon M_{\chi'} \,/\, \mathrm{Mod}(\Sigma') \to M_\chi \,/\, \mathrm{Mod}(\Sigma)$, the arrow view provides us a useful factorization of $U(h')$ as $U(1_{M_{\chi'}}) \fatsemi \beta(h')$.

$$\begin{tikzcd}
& M_{\chi'} \arrow[dl, "1_{M_{\chi'}}"'] \arrow[dr, "h'"] & & & & M_\chi \arrow[dl, "U(1_{M_{\chi'}})"'] \arrow[dr, "U(h')"] & \\
M_{\chi'} \arrow[rr, "h'"'] & & M' & \mapsto & \beta(M_{\chi'}) \arrow[rr, "\beta(h')"'] & & \beta(M')
\end{tikzcd}$$

▶ **Fact 13.** *Under the notation and hypotheses of Remark 12, for every Σ'-model homomorphism $h'\colon M_{\chi'} \to M'$, $U(h') = U(1_{M_{\chi'}}) \fatsemi \beta(h')$.*

4.2 Representable substitutions

Quantification spaces that have representable extensions of signatures, meaning that every extension $\chi\colon \Sigma \to \Sigma(\chi)$ defined by the quantification space is representable, allow us to extend the concept of representability from signature extensions (i.e. signatures of variables) to substitutions, leading to a purely model-theoretic view of the categories of substitutions.

▶ **Proposition 14.** *For any signature Σ in an institution with a quantification space $\mathbb{Q}$, the representation of signature extensions generalizes to a functor $R_\Sigma^{\mathbb{Q}}\colon \mathbb{S}\mathrm{ubst}_\Sigma^{\mathbb{Q}} \to \mathrm{Mod}(\Sigma)$, where*

[8] This property is commonly achieved by assuming that, for each type $s \in \vec{S}$, the set F_s contains an implicit constant-operation symbol.

- *for every extension of signatures* $\chi\colon \Sigma \to \Sigma(\chi)$ *in* $|\mathbb{Q}|$, $R^{\mathbb{Q}}_{\Sigma}(\chi) = M_{\chi}$, *and*
- *for every substitution* $\psi\colon \chi_1 \to \chi_2$, $R^{\mathbb{Q}}_{\Sigma}(\psi) = U_{\psi}(1_{M_{\chi_2}})\colon M_{\chi_1} \to M_{\chi_2}$.

Moreover, for every Σ*-substitution* ψ, $\mathrm{Mod}_{\Sigma}(\psi)$ *is uniquely determined by* $R^{\mathbb{Q}}_{\Sigma}(\psi)$.

When the quantification space $\mathbb{Q}$ and the signature Σ are clear from the context, we may also denote the representation $R^{\mathbb{Q}}_{\Sigma}(\psi)$ of a substitution $\psi\colon \chi_1 \to \chi_2$ by $h_{\psi}\colon M_{\chi_1} \to M_{\chi_2}$.

Note that, in general, the functor $R^{\mathbb{Q}}_{\Sigma}\colon \mathbb{S}\mathrm{ubst}^{\mathbb{Q}}_{\Sigma} \to \mathrm{Mod}(\Sigma)$ of Proposition 14 need be neither full nor faithful. For example, in the case of QF-$\underline{\mathrm{FOL}}_{=}$, for every general substitution ψ we can define another substitution ψ' (with the same domain and codomain as ψ) such that, for every atomic sentence $l = r$ that is not ground, $\mathrm{Sen}_{\Sigma}(\psi')(l = r)$ corresponds to $\mathrm{Sen}_{\Sigma}(\psi)(r = l)$. In consequence, we can obtain distinct institution-independent substitutions having the same underlying model functor – and thus, the same representation. This is contrary to our intuition concerning first-order substitutions, where, given a signature $\langle S, F\rangle$, every substitution $\psi\colon X_1 \to X_2$, i.e. every S-indexed family of maps $\psi_s\colon X_{1,s} \to \mathrm{T}_{F \cup X_2, s}$, is determined uniquely by its representation $R^{\mathbb{Q}}_{\langle S,F\rangle}(\psi)\colon \mathrm{T}_F(X_1) \to \mathrm{T}_F(X_2)$. As we will see later, the full and faithful representation of substitutions as model homomorphisms is essential for translating substitutions along signature morphisms.

▶ **Definition 15** (Representable substitution)**.** Let Σ be a signature in an institution equipped with a quantification space $\mathbb{Q}$. For every subcategory $\mathbb{S}\mathrm{ubst}_{\Sigma} \subseteq \mathbb{S}\mathrm{ubst}^{\mathbb{Q}}_{\Sigma}$, a substitution $\psi\colon \chi_1 \to \chi_2$ in $\mathbb{S}\mathrm{ubst}_{\Sigma}$ is said to be $\mathbb{Q}$-*representable* if it is uniquely determined by its image under $R^{\mathbb{Q}}_{\Sigma}$. In addition, $\mathbb{S}\mathrm{ubst}_{\Sigma}$ forms a category of $\mathbb{Q}$-*representable* Σ-*substitutions* if the restriction of the functor $R^{\mathbb{Q}}_{\Sigma}\colon \mathbb{S}\mathrm{ubst}^{\mathbb{Q}}_{\Sigma} \to \mathrm{Mod}(\Sigma)$ to $\mathbb{S}\mathrm{ubst}_{\Sigma}$ is both full and faithful.

▶ **Example 16.** Let $\mathbb{Q}$ be the quantification space for QF-$\underline{\mathrm{FOL}}_{=}$ presented in Example 7. For every signature $\langle S, F\rangle$, the subcategory $\mathbb{S}\mathrm{ubst}_{\langle S,F\rangle} \subseteq \mathbb{S}\mathrm{ubst}^{\mathbb{Q}}_{\langle S,F\rangle}$ whose arrows correspond to the corridors induced by first-order substitution forms a category of $\mathbb{Q}$-representable substitutions. A similar property can be formulated for higher-order substitutions.

For the remaining part of this paper we will assume that $\mathcal{I}$ *is an arbitrary but fixed institution* $\langle \mathbb{S}\mathrm{ig}, \mathrm{Sen}, \mathrm{Mod}, \models\rangle$ *equipped with*

- *an adequate quantification space* $\mathbb{Q} \subseteq \mathbb{S}\mathrm{ig}^{\to}$ *of representable signature extensions, and*
- *a broad subcategory* $\mathbb{S}\mathrm{ubst}_{\Sigma} \subseteq \mathbb{S}\mathrm{ubst}^{\mathbb{Q}}_{\Sigma}$ *(i.e. with the same objects as* $\mathbb{S}\mathrm{ubst}^{\mathbb{Q}}_{\Sigma}$*), for every signature* $\Sigma \in |\mathbb{S}\mathrm{ig}|$, *of* $\mathbb{Q}$*-representable* Σ*-substitutions.*

▶ **Lemma 17.** *Under the above assumptions, for every morphism* $\varphi\colon \Sigma \to \Sigma'$ *in* $\mathbb{S}\mathrm{ig}$ *and every signature extension* $\chi\colon \Sigma \to \Sigma(\chi)$ *in* $|\mathbb{Q}|$, *the homomorphism* $U_{\varphi,\chi}(1_{M_{\chi^{\varphi}}})\colon M_{\chi} \to M_{\chi^{\varphi}}\!\upharpoonright_{\varphi}$ *is a universal arrow from* M_{χ} *to* $\mathrm{Mod}(\varphi)$.

The lemma above enables us to make use of a well-known construction of adjoint functors from universal arrows (see e.g. [1]) to derive translations between categories of substitutions.

▶ **Proposition 18.** *Every morphism of signatures* $\varphi\colon \Sigma \to \Sigma'$ *gives rise to a functor* $\Psi_{\varphi}\colon \mathbb{S}\mathrm{ubst}_{\Sigma} \to \mathbb{S}\mathrm{ubst}_{\Sigma'}$ *that maps*

$$\begin{tikzcd}
 & \mathcal{I}(\Sigma) \arrow[rr, "\varphi"] \arrow[ld, "\mathcal{I}(\chi_1)"'] \arrow[d, "\mathcal{I}(\chi_2)"] & & \mathcal{I}(\Sigma') \arrow[ld, "\mathcal{I}(\chi_1^{\varphi})"'] \arrow[d, "\mathcal{I}(\chi_2^{\varphi})"] \\
\mathcal{I}(\Sigma(\chi_1)) \arrow[rr, "\mathcal{I}(\varphi^{\chi_1})"'] \arrow[rd, "\psi"'] & & \mathcal{I}(\Sigma'(\chi_1^{\varphi})) \arrow[rd, "\left(R^{\mathbb{Q}}_{\Sigma'}\right)^{-1}(h^{\varphi}_{\psi})"'] & \\
 & \mathcal{I}(\Sigma(\chi_2)) \arrow[rr, "\mathcal{I}(\varphi^{\chi_2})"'] & & \mathcal{I}(\Sigma'(\chi_2^{\varphi}))
\end{tikzcd}$$

- *every signature extension* $\chi\colon \Sigma \to \Sigma(\chi)$ *to* $\chi^\varphi\colon \Sigma' \to \Sigma'(\chi^\varphi)$*, and*
- *every* Σ*-substitution* $\psi\colon \chi_1 \to \chi_2$ *to* $\psi^\varphi = \big(R^{\mathbb{Q}}_{\Sigma'}\big)^{-1}(h^\varphi_\psi)$*, where* h^φ_ψ *is the unique* Σ'*-homomorphism* $M_{\chi_1^\varphi} \to M_{\chi_2^\varphi}$ *for which the diagram below commutes.*

$$\begin{tikzcd}
M_{\chi_1} \arrow[rr, "U_{\varphi,\chi_1}(1_{M_{\chi_1^\varphi}})"] \arrow[dr, "h_\psi"'] & & M_{\chi_1^\varphi}\restriction_\varphi \arrow[dr, "h^\varphi_\psi\restriction_\varphi"] & & M_{\chi_1^\varphi} \arrow[dr, "h^\varphi_\psi"] & \\
& M_{\chi_2} \arrow[rr, "U_{\varphi,\chi_2}(1_{M_{\chi_2^\varphi}})"'] & & M_{\chi_2^\varphi}\restriction_\varphi & & M_{\chi_2^\varphi}
\end{tikzcd}$$

Moreover, Ψ *itself is functorial, in the sense that* $\Psi_{\varphi \mathbin{;} \varphi'} = \Psi_\varphi \mathbin{;} \Psi_{\varphi'}$ *for every pair of composable signature morphisms* $\varphi\colon \Sigma \to \Sigma'$ *and* $\varphi'\colon \Sigma' \to \Sigma''$*, and* $\Psi_{1_\Sigma} = 1_{\mathbb{S}\mathrm{ubst}_\Sigma}$*.*

Proof. The first part of the statement follows by Lemma 17 as a direct consequence of the universal property of the homomorphism $U_{\varphi,\chi_1}(1_{M_{\chi_1^\varphi}})$ – note that, since the functor $R^{\mathbb{Q}}_\Sigma$ is assumed to be both full and faithful, for every signature Σ, it suffices to reason about the representations of substitutions. With respect to the second part of the statement, notice first that, by the definition of quantification spaces, the translation of signature extensions along signature morphisms is functorial (see Remark 9). In addition, by Remark 12, for every signature extension $\chi\colon \Sigma \to \Sigma(\chi)$, $U_{\varphi \mathbin{;} \varphi',\chi} = U_{\varphi',\chi^\varphi} \mathbin{;} U_{\varphi,\chi}$. This allows us to deduce, according to Fact 13, that $U_{\varphi \mathbin{;} \varphi',\chi}(1_{M_{\chi^\varphi}}) = U_{\varphi,\chi}(1_{M_{\chi^\varphi}}) \mathbin{;} U_{\varphi',\chi^\varphi}(1_{M_{\chi^{\varphi \mathbin{;} \varphi'}}})$. Hence, by the general properties of composing universal arrows, we can further conclude that the translation of substitutions along signature morphisms is also functorial. ◀

4.3 Deriving generalized substitution systems

For any signature morphism $\varphi\colon \Sigma \to \Sigma'$, the functor $\Psi_\varphi\colon \mathbb{S}\mathrm{ubst}_\Sigma \to \mathbb{S}\mathrm{ubst}_{\Sigma'}$ discussed in Proposition 18 can be extended in a straightforward manner to a morphism $\langle \Psi_\varphi, \kappa_\varphi, \tau_\varphi \rangle$ between the substitution systems $\mathcal{SI}_\Sigma$ and $\mathcal{SI}_{\Sigma'}$ obtained by restricting the functors $\mathcal{SI}^{\mathbb{Q}}_\Sigma$ and $\mathcal{SI}^{\mathbb{Q}}_{\Sigma'}$ of Proposition 3 to the subcategories $\mathbb{S}\mathrm{ubst}_\Sigma$ and $\mathbb{S}\mathrm{ubst}_{\Sigma'}$ of Σ- and Σ'-substitutions.

$$\begin{tikzcd}
\mathbb{S}\mathrm{ubst}_\Sigma \arrow[r, "\mathcal{SI}_\Sigma"] \arrow[d, "\Psi_\varphi"'] \arrow[dr, phantom, "\Downarrow\tau_\varphi"] & \mathcal{I}(\Sigma) \,/\, \mathbb{R}\mathrm{oom} \\
\mathbb{S}\mathrm{ubst}_{\Sigma'} \arrow[r, "\mathcal{SI}_\Sigma"'] & \mathcal{I}(\Sigma') \,/\, \mathbb{R}\mathrm{oom} \arrow[u, "\mathcal{I}(\varphi)/\mathbb{R}\mathrm{oom}"']
\end{tikzcd}$$

To be more specific, κ_φ is the corridor $\langle \mathrm{Sen}(\varphi), \mathrm{Mod}(\varphi) \rangle$ obtained by taking the image $\mathcal{I}(\varphi)$ of φ under the institution $\mathcal{I}$, regarded as a functor into $\mathbb{R}\mathrm{oom}$. Furthermore, for every signature extension $\chi\colon \Sigma \to \Sigma(\chi)$, the corridor $\tau_{\varphi,\chi}\colon \mathcal{I}(\Sigma(\chi)) \to \mathcal{I}(\Sigma'(\chi^\varphi))$ is simply $\mathcal{I}(\varphi^\chi)$. It should be noted, however, that the naturality of τ_φ holds in general only up to semantic equivalence (see Proposition 19 below): this means that we can only guarantee that $\mathrm{Mod}(\varphi^\chi)$ is natural in χ, and thus that, for every substitution $\psi\colon \chi_1 \to \chi_2$ and sentence ρ over $\Sigma(\chi_1)$, $\varphi^{\chi_2}(\psi(\rho))$ and $\psi^\varphi(\varphi^{\chi_1}(\rho))$ are satisfied by the same class of models. In concrete cases like QF-$\underline{\mathrm{FOL}_=}$, the equality $\varphi^{\chi_2}(\psi(\rho)) = \psi^\varphi(\varphi^{\chi_1}(\rho))$ is usually due to the careful choice of the categories of substitutions; other choices, which may involve, for example, swapping the left- and the right-hand side of non-ground equational atoms, do not necessarily give rise to natural transformations τ_φ. For this reason, in what follows, we will implicitly assume that the categories $\mathbb{S}\mathrm{ubst}_\Sigma$ of Σ-substitutions are *compatible* with respect to signature morphisms, meaning that $\psi \mathbin{;} \mathcal{I}(\varphi^{\chi_2}) = \mathcal{I}(\varphi^{\chi_1}) \mathbin{;} \psi^\varphi$ for every substitution $\psi\colon \chi_1 \to \chi_2$.

▶ **Proposition 19.** *For every signature morphism* $\varphi\colon \Sigma \to \Sigma'$ *and every* Σ*-substitution* $\psi\colon \chi_1 \to \chi_2$*,* $\mathrm{Mod}(\varphi^{\chi_2}) \mathbin{;} \mathrm{Mod}_\Sigma(\psi) = \mathrm{Mod}_{\Sigma'}(\psi^\varphi) \mathbin{;} \mathrm{Mod}(\varphi^{\chi_1})$*.*

We can now conclude the construction of a generalized substitution system $\mathcal{SI}\colon \mathbb{S}\mathrm{ig} \to \mathbb{S}\mathrm{ubst}\mathbb{S}\mathrm{ys}$ from an arbitrary institution $\mathcal{I}\colon \mathbb{S}\mathrm{ig} \to \mathbb{R}\mathrm{oom}$ that satisfies the hypotheses laid out in Section 4.2 by noticing that, according to Proposition 18, to the fact that $\mathcal{I}$ is a functor, and to Remark 9, all components of the morphism of substitution systems $\langle \Psi_\varphi, \kappa_\varphi, \tau_\varphi \rangle$ presented above are functorial in φ. Moreover, since the quantification space of $\mathcal{I}$ is assumed to be adequate, the generalized substitution system $\mathcal{SI}$ has model amalgamation.[9]

▶ **Theorem 20.** *For every institution $\mathcal{I}\colon \mathbb{S}\mathrm{ig} \to \mathbb{R}\mathrm{oom}$ equipped with an adequate quantification space $\mathbb{Q}$ of representable signature extensions and with compatible categories $\mathbb{S}\mathrm{ubst}_\Sigma$ of $\mathbb{Q}$-representable Σ-substitutions, $\mathcal{SI}\colon \mathbb{S}\mathrm{ig} \to \mathbb{S}\mathrm{ubst}\mathbb{S}\mathrm{ys}$ is a generalized substitution system that has model amalgamation.*

▶ **Example 21.** Both institutions QF-$\underline{\mathrm{FOL}_=}$ and QF-$\underline{\mathrm{HNK}}$, in combination with the extensions of signatures with constants and with the first-order and higher-order substitutions outlined in Example 4, give rise to generalized substitution systems.

5 Logic programming over an arbitrary institution

The view we take here is that the logic programming paradigm can be developed over an arbitrary institution $\mathcal{I}\colon \mathbb{S}\mathrm{ig} \to \mathbb{R}\mathrm{oom}$ by considering logic-programming frameworks and languages as in [29] defined over the generalized substitution system $\mathcal{SI}\colon \mathbb{S}\mathrm{ig} \to \mathbb{S}\mathrm{ubst}\mathbb{S}\mathrm{ys}$ introduced in Section 4.3. To this end, we assume that $\mathcal{I}\colon \mathbb{S}\mathrm{ig} \to \mathbb{R}\mathrm{oom}$ is an institution that satisfies the hypotheses of Theorem 20, and we let $\mathcal{L}$ be a logic-programming language whose underlying generalized substitution system is derived from $\mathcal{I}$.[10]

Under the additional assumption that, for every signature Σ, the identity 1_Σ is a signature of variables – the 'empty' signature of Σ-variables – the general institution-independent versions of Herbrand's theorem presented in [5, 6] can be obtained as concrete instances of Herbrand's theorem for abstract logic-programming languages. In particular, the equivalence 1 ⇔ 2 of Theorem 25 below captures the denotational aspect of the result – how the problem of checking whether a logic program entails a given query (formalized as an existential sentence) can be reduced from all models of the program to those that are initial; on the other hand, the equivalence 2 ⇔ 3 emphasizes the operational aspect of the theorem – the correspondence between those expansions of the program's initial model that satisfy the underlying (quantifier-free) sentence of the query and the possible solutions to the query.

To start with, let us recall that, in any category, an object A is *projective* with respect to an arrow $e\colon B \to C$ provided that every other arrow $f\colon A \to C$ can be factored through e as $f = h \mathbin{⨾} e$, for some arrow $h\colon A \to B$. For instance, as a consequence of the axiom of choice, for every QF-$\underline{\mathrm{FOL}_=}$-signature extension $\langle S, F \rangle \subseteq \langle S, F \cup X \rangle$, the free algebra $T_F(X)$ – that is the representation of the inclusion $\langle S, F \rangle \subseteq \langle S, F \cup X \rangle$ – is projective with respect to all epimorphisms, and in particular with respect to all quotient homomorphisms between the initial models $0_{\langle S, F \rangle}$ of the signature $\langle S, F \rangle$ and $0_{\langle S, F \rangle, \Gamma}$ of sets of $\langle S, F \rangle$-clauses Γ.

The concept of *basic sentence* (see [4], and also [27], where it was studied under the name of *ground positive elementary sentence*) captures the satisfaction of the conjunctions of atomic sentences that are usually involved in defining logic-programming queries.

[9] This property is essential for ensuring that the satisfaction of clauses and queries is invariant under change of notation; in general, it means that, for every signature morphism $\varphi\colon \Sigma \to \Sigma'$ and signature of Σ-variables X, we can amalgamate those models M' of Σ' and N of X that have the same Σ-reduct.

[10] It should be noted that, in the present paper, we do not fully address the operational semantics of $\mathcal{L}$. A detailed presentation of the goal-directed rules – which, for first-order and higher-order equational logic programming, correspond to paramodulation – can be found in [29].

▶ **Definition 22** (Basic sentence)**.** For any signature Σ, a sentence ρ is said to be *basic* if there exists a model M_ρ such that, for every Σ-model M, $M \models_\Sigma \rho$ if and only if there exists a model homomorphism $M_\rho \to M$.

▶ **Example 23.** In the institution QF-$\underline{\mathrm{FOL}_=}$, every (finite) conjunction of first-order equational atoms forms a basic sentence (see, for example, [6]). This property does not hold in general for QF-$\underline{\mathrm{HNK}}$, for which one can define higher-order equational atoms of the form $\sigma_1\, f = \sigma_2\, f$, with $f\colon s \to s$ and $\sigma_1, \sigma_2\colon (s \to s) \to s'$, that are not basic (see [2, 6]).

We also recall from [29] the following concept of reachability.

▶ **Definition 24** (Reachable model)**.** Given an extension $\chi\colon \Sigma \to \Sigma(\chi)$, a Σ-model M is *χ-reachable* if for every χ-expansion N of M there exists a substitution $\psi\colon \chi \to \chi'$ such that

- χ' is conservative, in the sense that every Σ-model admits a χ'-expansion, and
- the canonical map $\_\!\upharpoonright_\chi\colon N \,/\, \mathrm{Mod}_\Sigma(\psi) \to M \,/\, \mathrm{Mod}(\chi')$ determined by the reduct functor $\mathrm{Mod}(\chi)$ is surjective on objects.

The above preliminaries enable us to recast the institution-independent versions of Herbrand's theorem in the context of abstract logic-programming languages.

▶ **Theorem 25.** *Consider a logic program $\langle \Sigma, \Gamma \rangle$*[11] *and a Σ-query $\exists\chi \cdot \rho$ such that*

- *both the signature Σ and the program $\langle \Sigma, \Gamma \rangle$ have initial models 0_Σ and $0_{\Sigma,\Gamma}$,*
- *M_χ is projective with respect to the unique homomorphism $!_\Gamma\colon 0_\Sigma \to 0_{\Sigma,\Gamma}$, and*
- *the $\Sigma(\chi)$-sentence ρ is basic.*

Then the following statements are equivalent:

1. $\Gamma \models_\Sigma \exists\chi \cdot \rho$.
2. $0_{\Sigma,\Gamma} \models_\Sigma \exists\chi \cdot \rho$.
3. *There exists a substitution $\psi\colon \chi \to \chi'$ such that χ' is conservative and $\Gamma \models_\Sigma \forall\chi' \cdot \psi(\rho)$.*

Proof. According to [29, Theorem 5.12], it suffices to prove that $0_{\Sigma,\Gamma}$ is χ-reachable and that ρ is preserved by χ-homomorphisms. The latter property follows from the assumption that ρ is basic (see [4]). Therefore, we will focus solely on proving that $0_{\Sigma,\Gamma}$ is χ-reachable.

Let $N_{\Sigma,\Gamma}$ be a χ-expansion of $0_{\Sigma,\Gamma}$. Since χ is a representable extension of signatures (by hypothesis), we know that $i_\chi(N_{\Sigma,\Gamma})\colon M_\chi \to 0_{\Sigma,\Gamma}$ is an object of the comma category $M_\chi/\mathrm{Mod}(\Sigma)$; and because its representation, M_χ, is projective with respect to $!_\Gamma\colon 0_\Sigma \to 0_{\Sigma,\Gamma}$, it follows that there exists a homomorphism $h\colon M_\chi \to 0_\Sigma$ such that $h \fatsemi !_\Gamma = i_\chi(N_{\Sigma,\Gamma})$. Moreover, since the identity 1_Σ is a signature of Σ-variables – which, by hypothesis, is also representable – we deduce that 0_Σ is (isomorphic to) the representation M_{1_Σ} of 1_Σ. By the representability of Σ-substitutions, we further obtain the substitution $\left(R^{\mathbb{Q}}_{\Sigma'}\right)^{-1}(h)\colon \chi \to 1_\Sigma$, which we will henceforth denote by ψ. All we need to show now is that the canonical map $\_\!\upharpoonright_\Sigma\colon N_{\Sigma,\Gamma} \,/\, \mathrm{Mod}_\Sigma(\psi) \to 0_{\Sigma,\Gamma} \,/\, \mathrm{Mod}(\Sigma)$ is surjective on objects.

To this end, notice that every Σ-model homomorphism $g\colon 0_{\Sigma,\Gamma} \to M$ can be viewed as an arrow in $M_\chi \,/\, \mathrm{Mod}(\Sigma)$ between $i_\chi(N_{\Sigma,\Gamma})$ and $i_\chi(N_{\Sigma,\Gamma}) \fatsemi g$, from which we deduce that $i_\chi^{-1}(g)$ is a $\Sigma(\chi)$-model homomorphism between $N_{\Sigma,\Gamma}$ and $N = i_\chi^{-1}(i_\chi(N_{\Sigma,\Gamma}) \fatsemi g)$. In addition, by Proposition 14 and the commutativity of the diagram below, we obtain $M\!\upharpoonright_\psi = i_\chi^{-1}(h \fatsemi !_M) = i_\chi^{-1}(i_\chi(N_{\Sigma,\Gamma}) \fatsemi g) = N$, thus confirming that $i_\chi^{-1}(g)\colon N_{\Sigma,\Gamma} \to M\!\upharpoonright_\psi$

[11] For simplicity, we only consider here logic programs defined as theory presentations. The same result can still be stated for more complex, structured logic programs as in [29].

is an object of $N_{\Sigma,\Gamma} \mathbin{/} \mathrm{Mod}_{\Sigma}(\psi)$. The conclusion of the theorem follows by observing that $N{\upharpoonright}_{\Sigma} = |i_{\chi}(N_{\Sigma,\Gamma}) \mathbin{;} g|_{M_{\chi}} = M$ and $i_{\chi}^{-1}(g){\upharpoonright}_{\Sigma} = |g|_{M_{\chi}} = g$.

$$\begin{tikzcd}
M_{\chi} \arrow[rr, "h"] \arrow[dr, "i_{\chi}(N_{\Sigma,\Gamma})"'] & & 0_{\Sigma} \arrow[dl, "!_{\Gamma}"] \arrow[dr, "!_{M}"] & \\
& 0_{\Sigma,\Gamma} \arrow[rr, "g"'] & & M
\end{tikzcd}$$

◀

6 Conclusions

In this paper, we have examined the connection between the institution-independent approach to Herbrand's theorem reported in [5] and the abstract axiomatic theory of logic programming that we previously proposed in [29]. We have first shown that, for an arbitrary but fixed signature Σ of an institution $\mathcal{I}$, any class of $\mathcal{I}$-signature morphisms gives rise to a canonical Σ-substitution system whose substitutions correspond precisely to the institution-independent concept of substitution. Lifting this result to institutions and generalized substitution systems – so as to enable the application of the general variant of Herbrand's theorem from [29] – proved to be much more difficult, and it required the development of a number of new properties and results concerning quantification spaces and the representability of signature morphisms. To summarize, we have determined that any institution equipped with an adequate quantification space of representable signature extensions and with compatible categories of representable substitutions leads to a generalized substitution system. Moreover, we showed that the resulting generalized substitution system has model amalgamation, and thus that it forms a suitable foundation for defining logic-programming languages.

The most problematic aspect of the derivation of a generalized substitution system is the translation of institution-independent substitutions along signature morphisms, for which one still has to check properties such as compatibility for each particular institution of interest. For this reason, a promising line of research would be to explore alternative, more syntactic, and also more specific notions of substitution, inspired for example by the recent study [21] on derived signature morphisms and substitutions in the context of institutional monads.

Acknowledgements. This research has been supported by a grant of the Romanian National Authority for Scientific Research, CNCS-UEFISCDI, project PN-II-ID-PCE-2011-3-0439.

References

1 Jiri Adámek, Horst Herrlich, and George Strecker. *Abstract and Concrete Categories: The Joy of Cats*. Dover Publications, 2009. reprint.

2 Mihai Codescu. The model theory of higher order logic. Master's thesis, Şcoala Normală Superioară Bucureşti, 2007.

3 Răzvan Diaconescu. Category-based constraint logic. *Mathematical Structures in Computer Science*, 10(3):373–407, 2000.

4 Răzvan Diaconescu. Institution-independent ultraproducts. *Fundamenta Informaticae*, 55(3–4):321–348, 2003.

5 Răzvan Diaconescu. Herbrand theorems in arbitrary institutions. *Information Processing Letters*, 90(1):29–37, 2004.

6 Răzvan Diaconescu. *Institution-Independent Model Theory*. Birkhäuser, 2008.

7 Răzvan Diaconescu. Quasi-Boolean encodings and conditionals in algebraic specification. *Journal of Logic and Algebraic Programming*, 79(2):174–188, 2010.

8 Răzvan Diaconescu. Institutional semantics for many-valued logics. *Fuzzy Sets and Systems*, 218:32–52, 2013.

9 Daniel Găină. Forcing, Downward Löwenheim-Skolem and Omitting Types theorems, institutionally. *Logica Universalis*, 8(3-4):469–498, 2014.

10 Daniel Găină. Foundations of logic programming in hybridised logics. In *Recent Trends in Algebraic Development Techniques*, Lecture Notes in Computer Science. Springer, in press.

11 Joseph A. Goguen and Rod M. Burstall. Institutions: abstract model theory for specification and programming. *Journal of the* ACM, 39(1):95–146, 1992.

12 Joseph A. Goguen, Grant Malcolm, and Tom Kemp. A hidden Herbrand theorem: combining the object and logic paradigms. *Journal of Logic and Algebraic Programming*, 51(1):1–41, 2002.

13 Joseph A. Goguen and José Meseguer. EQLOG: Equality, types, and generic modules for logic programming. In *Logic Programming: Functions, Relations, and Equations*, pages 295–363. Prentice Hall, 1986.

14 Joseph A. Goguen and José Meseguer. Models and equality for logical programming. In Hartmut Ehrig, Robert A. Kowalski, Giorgio Levi, and Ugo Montanari, editors, *Theory and Practice of Software Development*, volume 250 of *Lecture Notes in Computer Science*, pages 1–22. Springer, 1987.

15 Jacques Herbrand. Investigations in proof theory. In *From Frege to Gödel: A Source Book in Mathematical Logic*, pages 525–581. Harvard University Press, 1967.

16 John W. Lloyd. *Foundations of Logic Programming.* Springer, 1987.

17 Manuel A. Martins, Alexandre Madeira, Răzvan Diaconescu, and Luís Soares Barbosa. Hybridization of institutions. In Andrea Corradini, Bartek Klin, and Corina Cîrstea, editors, *Algebra and Coalgebra in Computer Science*, volume 6859 of *Lecture Notes in Computer Science*, pages 283–297. Springer, 2011.

18 José Meseguer. General logics. In Heinz-Dieter Ebbinghaus, José Fernández-Prida, Manuel Garrido, Daniel Lascar, and Mario Rodriquez-Artalejo, editors, *Logic Colloquium 1987*, volume 129, pages 275–329. Elsevier, 1989.

19 José Meseguer. Multiparadigm logic programming. In Hélène Kirchner and Giorgio Levi, editors, *Algebraic and Logic Programming*, volume 632 of *Lecture Notes in Computer Science*, pages 158–200. Springer, 1992.

20 Bernhard Möller, Andrzej Tarlecki, and Martin Wirsing. Algebraic specifications of reachable higher-order algebras. In Donald Sannella and Andrzej Tarlecki, editors, *Abstract Data Types*, volume 332 of *Lecture Notes in Computer Science*, pages 154–169. Springer, 1987.

21 Till Mossakowski, Ulf Krumnack, and Tom Maibaum. What is a derived signature morphism? In *Recent Trends in Algebraic Development Techniques*, Lecture Notes in Computer Science. Springer, in press.

22 Fernando Orejas, Elvira Pino, and Hartmut Ehrig. Institutions for logic programming. *Theoretical Computer Science*, 173(2):485–511, 1997.

23 Wieslaw Pawlowski. Context institutions. In Magne Haveraaen, Olaf Owe, and Ole-Johan Dahl, editors, *Specification of Abstract Data Types*, volume 1130 of *Lecture Notes in Computer Science*, pages 436–457. Springer, 1995.

24 Donald Sannella and Andrzej Tarlecki. Building specifications in an arbitrary institution. In Gilles Kahn, David B. MacQueen, and Gordon D. Plotkin, editors, *Semantics of Data Types*, volume 173 of *Lecture Notes in Computer Science*, pages 337–356. Springer, 1984.

25 Donald Sannella and Andrzej Tarlecki. *Foundations of Algebraic Specification and Formal Software Development.* Springer, 2011.

26 Traian Florin Şerbănuţă. Institutional concepts in first-order logic, parameterized specification, and logic programming. Master's thesis, University of Bucharest, 2004.

27 Andrzej Tarlecki. Quasi-varieties in abstract algebraic institutions. *Journal of Computer and System Sciences*, 33(3):333–360, 1986.

28 Ionuţ Ţuţu. Comorphisms of structured institutions. *Information Processing Letters*, 113(22–24):894–900, 2013.

29 Ionuţ Ţuţu and José L. Fiadeiro. From conventional to institution-independent logic programming. *Journal of Logic and Computation*, in press.

30 Ionuţ Ţuţu and José L. Fiadeiro. Service-oriented logic programming. *Logical Methods in Computer Science*, in press.

Coalgebraic Infinite Traces and Kleisli Simulations

Natsuki Urabe and Ichiro Hasuo

Department of Computer Science, The University of Tokyo
Hongo 7-3-1, Tokyo 113-8656, Japan
`{urabenatsuki,ichiro}@is.s.u-tokyo.ac.jp`

Abstract

Kleisli simulation is a categorical notion introduced by Hasuo to verify finite trace inclusion. They allow us to give definitions of *forward and backward simulation* for various types of systems. A generic categorical theory behind Kleisli simulation has been developed and it guarantees the soundness of those simulations wrt. *finite* trace semantics. Moreover, those simulations can be aided by *forward partial execution* (FPE) – a categorical transformation of systems previously introduced by the authors.

In this paper, we give Kleisli simulation a theoretical foundation that assures its soundness also wrt. *infinite* trace. There, following Jacobs' work, infinite trace semantics is characterized as the "largest homomorphism." It turns out that soundness of forward simulations is rather straightforward; that of backward simulation holds too, although it requires certain additional conditions and its proof is more involved. We also show that FPE can be successfully employed in the infinite trace setting to enhance the applicability of Kleisli simulations as witnesses of trace inclusion. Our framework is parameterized in the monad for branching as well as in the functor for linear-time behaviors; for the former we use the powerset monad (for nondeterminism) as well as the sub-Giry monad (for probability).

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases category theory, coalgebra, simulation, verification, trace semantics

Digital Object Identifier 10.4230/LIPIcs.CALCO.2015.320

1 Introduction

Language inclusion of transition systems is an important problem in both qualitative and quantitative verification. In a qualitative setting the problem is concretely as follows: for given two nondeterministic systems $\mathcal{X}$ and $\mathcal{Y}$, check if $L(\mathcal{X}) \subseteq L(\mathcal{Y})$ – that is, if the set of words generated by $\mathcal{X}$ is included in the set of words generated by $\mathcal{Y}$. In a typical usage scenario, $\mathcal{X}$ is a model of the *implementation* in question while $\mathcal{Y}$ is a model that represents the *specification* of $\mathcal{X}$. More concretely, $\mathcal{Y}$ is a system such that $L(\mathcal{Y})$ is easily seen not to contain anything "dangerous" – therefore the language inclusion $L(\mathcal{X}) \subseteq L(\mathcal{Y})$ immediately implies that $L(\mathcal{X})$ contains no dangerous output, either. Such a situation can also arise in a *quantitative setting* where a specification is about *probability*, *reward*, and so on.

▶ **Example 1.1.** In Fig. 1 are four examples of transition systems; $\mathcal{X}$ and $\mathcal{Y}$ are qualitative/nondeterministic; $\mathcal{Z}$ and $\mathcal{W}$ exhibit probabilistic branching. We shall denote the *finite* language of a system $\mathcal{A}$ by $L^*(\mathcal{A})$ and the *infinite* one by $L^\infty(\mathcal{A})$. We define that a generated finite word is one with a run that ends with the termination symbol ✓.

In the nondeterministic setting, languages are *sets* of words. We have $L^*(\mathcal{X}) = \{b\} \subseteq \{b, ab, aab, \dots\} = L^*(\mathcal{Y})$, i.e. *finite* language inclusion from $\mathcal{X}$ to $\mathcal{Y}$. However $abb\ldots \in L^\infty(\mathcal{X})$ while $abb\ldots \notin L^\infty(\mathcal{Y})$, hence *infinite* language inclusion fails.

6th International Conference on Algebra and Coalgebra in Computer Science (CALCO'15).
Editors: Lawrence S. Moss and Pawel Sobocinski; pp. 320–335
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

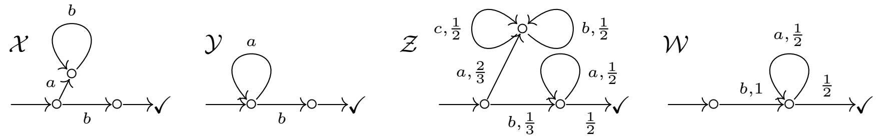


Figure 1 Examples of nondeterministic and probabilistic automata.

In the probabilistic setting, languages are naturally *probability distributions* over words; and language inclusion refers to the pointwise order between probabilities. For example $L^*(\mathcal{Z}) = [b \mapsto \frac{1}{6}, ba \mapsto \frac{1}{12}, baa \mapsto \frac{1}{24}, \dots]$ and $L^*(\mathcal{W}) = [b \mapsto \frac{1}{2}, ba \mapsto \frac{1}{4}, baa \mapsto \frac{1}{8}, \dots]$; since the former assigns no greater probabilities to all the words, we say that the finite language of $\mathcal{Z}$ is *included* in that of $\mathcal{W}$. This quantitative notion of trace inclusion is also useful in verification: it gives e.g. an upper bound for the probability for something bad.

Finally, the *infinite* languages for probabilistic systems call for measure-theoretic machinery since, in most of the cases, any infinite word gets assigned the probability 0 (which is also the case in $\mathcal{Z}$ and $\mathcal{W}$). Here it is standard to assign probabilities to *cylinder sets* rather than to individual words; see e.g. [2]. An example of a cylinder set is $\{aw \mid w \in \{b, c\}^\omega\}$. The language $L^\infty(\mathcal{Z})$ assigns $\frac{2}{3}$ to it, while $L^\infty(\mathcal{W})$ assigns 0; therefore we do not have *infinite* language inclusion from $\mathcal{Z}$ to $\mathcal{W}$.

There are many known algorithms for checking language inclusion. A well-known one for NFA is a complete one that reduces the problem to emptiness check; however it involves complementation, hence determinization, that incurs an exponential blowup.

One of the alternative approaches to language inclusion is by *simulation*. In the simulation-based verification we look for a simulation, that is, a witness for *stepwise* language inclusion. The notion of simulation is commonly defined so that it implies (proper, global) language inclusion – a property called *soundness*. Although its converse (*completeness*) fails in many settings, such simulation-based approaches tend to have an advantage in computational cost. One prototype example of such simulation notions is *forward* and *backward simulation* [14], by Lynch and Vaandrager, for nondeterministic automata. They are shown in [14] to satisfy soundness wrt. finite trace: explicitly, existence of a forward (or backward) simulation from $\mathcal{X}$ to $\mathcal{Y}$ implies $L(\mathcal{X}) \subseteq L(\mathcal{Y})$, where the languages collects all the *finite* words generated.

Kleisli simulation [8, 9, 18] is a categorical generalization of these notions of forward and backward simulation by Lynch and Vaandrager. It builds upon the use of coalgebras in a *Kleisli category*, in [10], where they are used to characterize finite traces. Specifically:

- A branching system $\mathcal{X}$ is represented as an F*-coalgebra* $c : X \nrightarrow FX$ in the Kleisli category $\mathcal{K}\ell(T)$, for a suitable choice of a functor F and a monad T. Here F and T are parameters that determine the (linear-time) *transition type* and the *branching type*, respectively, of the system $\mathcal{X}$. Examples are:
  - $F = 1 + \Sigma \times (\_)$ (terminate, or (output and continue)) and the *powerset monad* $T = \mathcal{P}$ on **Sets** (nondeterminism), if $\mathcal{X}$ is a nondeterministic automaton (with explicit termination); and
  - the same functor $F = 1 + \Sigma \times (\_)$ and the *sub-Giry monad* $T = \mathcal{G}$ [7] on the category **Meas** of measurable spaces and measurable functions, for their probabilistic variant.
- In [10], under certain conditions on F and T, it is shown that a *final* F*-coalgebra* in $\mathcal{K}\ell(T)$ arises as a lifting of an initial F-algebra in **Sets**. Moreover, it is observed that the natural notion of "finite trace semantics" or "(finite) languages" is captured by a unique homomorphism via finality. This works uniformly for a wide variety of systems, by changing F and T.

It is shown in [8] that, with respect to this categorical characterization of finite trace [10], both forward and backward Kleisli simulation are indeed sound. This categorical background allows us to instantiate Kleisli simulation for various concrete systems – including both qualitative and quantitative ones – and obtain simulation notions whose soundness wrt. *finite* traces comes for free [8, 9]. Like many other notions of simulation, the resulting simulation sometimes fails to be complete. This drawback of incompleteness wrt. finite trace can be partly mended by *forward partial execution* (FPE), a transformation of coalgebraic systems introduced in [18] that potentially increases the likelihood of existence of simulations.

Contributions. In this paper we continue our series of work [8, 9, 18] and study the relationship between Kleisli simulations and *infinite* traces. This turns out to be more complicated than we had expected, a principal reason being that *infinite* traces are less well-behaved than finite traces (that are characterized simply by finality).

For a suitable coalgebraic characterization of infinite traces we principally follow [11] – also relying on observations in [4, 12] – and characterize infinite traces in terms of *largest homomorphisms.* More specifically, we lift a final F-coalgebra in **Sets** to the Kleisli category $\mathcal{K}\ell(T)$ and exhibit that the latter admits a largest homomorphism. In this paper we (principally) work with: the powerset monad $\mathcal{P}$ (on **Sets**) and the sub-Giry monad $\mathcal{G}$ (on **Meas**), as a monad T for branching; and a polynomial functor F for linear-time behaviors.

Here are our concrete contributions. For each of the above combinations of T and F:

- We show that forward Kleisli simulations are sound with respect to inclusion of *infinite* languages. The proof of this general result is not hard, exploiting the above coalgebraic characterization of infinite languages as largest homomorphisms.
- We show that backward simulations are sound too, although here we have to impose suitable restrictions, like *totality* and *image-finiteness.* The soundness proofs are much more involved, too, and calls for careful inspection of the construction of infinite trace semantics. The proofs are separately for $T = \mathcal{P}$ and for $\mathcal{G}$.
- We show that *forward partial execution* (FPE) – a transformation from [18] that aids discovery of fwd./bwd. simulations – is applicable also to the current setting of *infinite* trace inclusion. More specifically we prove: *soundness* of FPE (discovery of a simulation after FPE indeed witnesses infinite language inclusion); and its *adequacy* (FPE does not destroy simulations that are already there).

Organization. §2 is devoted to categorical preliminaries; we fix notations there. In §3 we review the previous works that we rely on, namely coalgebraic infinite trace semantics [11], Kleisli simulation [8, 9, 18], and FPE [18]. Our technical contributions are in the subsequent sections: in §4 we study the nondeterministic setting (i.e. the powerset monad $\mathcal{P}$ on **Sets** and a polynomial functor F); §5 is for the probabilistic setting (where the monad T is the sub-Giry monad $\mathcal{G}$). In §6 we briefly discuss other monads like the *lift monad* $\mathcal{L}$ (for divergence) and the *subdistribution monad* $\mathcal{D}$ on **Sets** (for discrete probabilities).

Some definitions and results in §4–5 are marked with †. Those marked ones are essentially proofs of the results for specific settings (namely $T = \mathcal{P}$ and $T = \mathcal{G}$) but formulated in general terms with a general T. We do so in the hope that the axioms thus identified will help to discover new instances.

Most proofs are deferred to the appendices, that are found in the extended version [19] of this paper. Auxiliary definitions and examples are also found there.

2 Preliminaries

▶ **Definition 2.1.** A *polynomial functor* F on **Sets** is defined by the following BNF notation: $F ::= \mathrm{id} \mid A \mid F_1 \times F_2 \mid \coprod_{i\in I} F_i$. Here $A \in$ **Sets** and I is a countable set.

The notion of polynomial functor can be also defined for **Meas** – the category of measurable spaces and measurable functions between them.

▶ **Definition 2.2.** A *(standard Borel) polynomial functor* F on **Meas** is defined by the following BNF notation: $F ::= \mathrm{id} \mid (A, \mathfrak{F}_A) \mid F_1 \times F_2 \mid \coprod_{i\in I} F_i$. Here I is a countable set; and we require that $(A, \mathfrak{F}_A) \in$ **Meas** is a *standard Borel space* (see e.g. [6]). The σ-algebra $\mathfrak{F}_{FX}$ associated to FX is defined in the obvious manner. Namely: for $F = \mathrm{id}$, $\mathfrak{F}_{FX} = \mathfrak{F}_X$; for $F = (A, \mathfrak{F}_A)$, $\mathfrak{F}_{FX} = \mathfrak{F}_A$; for $F = F_1 \times F_2$, $\mathfrak{F}_{FX}$ is the smallest σ-algebra that contains $A_1 \times A_2$ for all $A_1 \in \mathfrak{F}_{F_1X}$ and $A_2 \in \mathfrak{F}_{F_2X}$; for for $F = \coprod_{i\in I} F_i$, $\mathfrak{F}_{FX} = \{\coprod_{i\in I} A_i \mid A_i \in \mathfrak{F}_{F_iX}\}$.

For arrows, F works in the same manner as a polynomial functor on **Sets**.

In what follows, a standard Borel polynomial functor is often called simply a *polynomial functor*.

The technical requirement of being standard Borel in the above will be used in the probabilistic setting of §5 (it is also exploited in [4, 17]). A standard Borel space is a measurable space induced by a Polish space; for further details see e.g. [6].

There is a natural correspondence between polynomial functors and *ranked alphabets*. In this paper a functor F for the (linear-time) transition type is restricted to a polynomial one; this means that we are dealing with (T-branching) systems that generate *trees* over some ranked alphabet. We collect some standard notions and notations for such trees in Appendix A.1; they will be used later in showing that our coalgebraic infinite traces indeed capture infinite tree languages of such systems.

We go on to introduce monads T for branching. We principally use two monads – the *powerset monad* $\mathcal{P}$ on **Sets** and the *sub-Giry* monad $\mathcal{G}$ on **Meas**. The latter is an adaptation of the *Giry monad* [7] and inherits most of its structure from the Giry monad; see Rem. 2.6.

▶ **Definition 2.3** (monads $\mathcal{P}$ and $\mathcal{G}$)**.** The *powerset monad* is the monad $(\mathcal{P}, \eta^{\mathcal{P}}, \mu^{\mathcal{P}})$ on **Sets** such that $\mathcal{P}X = \{A \subseteq X\}$ and $\mathcal{P}f(A) = \{f(x) \mid x \in A\}$. Its unit is given by the singleton set $\eta^{\mathcal{P}}_X(x) = \{x\}$ and its multiplication is given by $\mu^{\mathcal{P}}_X(M) = \bigcup_{A\in M} A$.

The *sub-Giry monad* is the monad $(\mathcal{G}, \eta^{\mathcal{G}}, \mu^{\mathcal{G}})$ on **Meas** such that

- $\mathcal{G}(X, \mathfrak{F}_X) = (\mathcal{G}X, \mathfrak{F}_{\mathcal{G}X})$, where the underling set $\mathcal{G}X$ is the set of all *subprobability measures* on $(X, \mathfrak{F}_X)$. The latter means those measures which assign to the whole space X a value in the unit interval $[0, 1]$.
- The σ-algebra $\mathfrak{F}_{\mathcal{G}X}$ on $\mathcal{G}X$ is the smallest σ-algebra such that, for all $S \in \mathfrak{F}_X$, the function $\mathrm{ev}_S : \mathcal{G}X \to [0, 1]$ defined by $\mathrm{ev}_S(P) = P(S)$ is measurable.
- $\mathcal{G}f(\nu)(S) = \nu(f^{-1}(S))$ where $f : (X, \mathfrak{F}_X) \to (Y, \mathfrak{F}_Y)$ is measurable, $\nu \in \mathcal{G}X$, and $S \in \mathfrak{F}_Y$.
- $\eta^{\mathcal{G}}_{(X,\mathfrak{F}_X)(x)}$ is given by the *Dirac measure*: $\eta^{\mathcal{G}}_{(X,\mathfrak{F}_X)}(x)(S)$ is 1 if $x \in S$ and 0 otherwise.
- $\mu^{\mathcal{G}}_{(X,\mathfrak{F}_X)}(\Psi)(S) = \int_{\mathcal{G}(X,\mathfrak{F}_X)} \mathrm{ev}_S \, d\Psi$ where $\Psi \in \mathcal{G}^2X$, $S \in \mathfrak{F}_X$ and ev_S is defined as above.

A monad gives rise to a category called its *Kleisli category* (see e.g. [15]).

▶ **Definition 2.4** (Kleisli category $\mathcal{K}\ell(T)$)**.** Given a monad (T, η, μ) on a category $\mathbb{C}$, the *Kleisli category* for T is the category $\mathcal{K}\ell(T)$ whose objects are the same as $\mathbb{C}$, and for each pair of objects X, Y, the homset $\mathcal{K}\ell(T)(X, Y)$ is given by $\mathbb{C}(X, TY)$. An arrow in $\mathcal{K}\ell(T)$ is referred to as a *Kleisli arrow*, and depicted by $X \mathrel{\ooalign{\hfil$\mapstochar\mkern5mu$\hfil\cr$\to$\cr}} Y$ for distinction. Note that it is nothing but an arrow $X \to TY$ in the base category $\mathbb{C}$.

Moreover, for two sequential Kleisli arrows $f : X \relbar\joinrel\mapstochar\rightarrow Y$ and $g : Y \relbar\joinrel\mapstochar\rightarrow Z$, their composition is given by $\mu_Z \circ Tg \circ f$ and denoted by $g \odot f$. The *Kleisli inclusion functor* is the functor $J : \mathbb{C} \to \mathcal{K}\ell(T)$ such that $JX = X$ and $Jf = \eta_Y \circ f$ for $f : X \to Y$ in $\mathbb{C}$.

It is known that a functor $F : \mathbb{C} \to \mathbb{C}$ canonically lifts to a functor $\overline{F} : \mathcal{K}\ell(T) \to \mathcal{K}\ell(T)$, given that there exists a natural transformation $\lambda : FT \Rightarrow TF$ that is compatible with the unit and the multiplication of T. Such a natural transformation is called a *distributive law*. For more details, see [16].

Throughout this paper, we fix the orders on the homsets of $\mathcal{K}\ell(\mathcal{P})$ and $\mathcal{K}\ell(\mathcal{G})$ as follows.

▶ **Definition 2.5** (order enrichment of $\mathcal{K}\ell(\mathcal{P})$ and $\mathcal{K}\ell(\mathcal{G})$). We define an order on $\mathcal{K}\ell(\mathcal{P})(X, Y)$ by $f \sqsubseteq g \overset{\text{def}}{\Leftrightarrow} \forall x \in X.\, f(x) \subseteq g(x)$. We define an order on $\mathcal{K}\ell(\mathcal{G})(X, Y)$ by $f \sqsubseteq g \overset{\text{def}}{\Leftrightarrow} \forall x \in X.\, \forall A \in \mathfrak{F}_Y.\, f(x)(A) \leq g(x)(A)$. Here the last $\leq$ is the usual order in the unit interval $[0, 1]$.

▶ **Remark 2.6.** *The sub-Giry monad $\mathcal{G}$ is an adaptation of the* Giry monad *from [7]; in the original Giry monad we only allow (proper)* probability measures*, i.e. measures that map the whole space to* 1*. We work with the sub-Giry monad because, without this relaxation from probability to subprobability, the order structure in Def. 2.5 is reduced to the equality.*

3 Infinite Traces, Kleisli Simulations and Coalgebras in $\mathcal{K}\ell(T)$

In this section we review the categorical constructs, the relationship among which lies at the heart of this paper. They are namely: coalgebraic infinite trace semantics [11], Kleisli simulation [8, 9, 18] and forward partial execution (FPE) [18].

The following situation is identified in [11], (see also §A.2 and §A.5.3): the largest homomorphism to a certain coalgebra that we describe below is observed to coincide with the standard, conventionally defined notion of infinite language, for a variety of systems. An instance of it is shown to arise, in [11], when $\mathbb{C} = \mathbf{Sets}$, $T = \mathcal{P}$ and F is a polynomial functor. In §4 we will give another proof for this fact; the new proof will serve our goal of showing soundness of backward simulations.

▶ **Definition 3.1** (infinite trace situation). Let F be an endofunctor and T be a monad on a category $\mathbb{C}$. We assume that each homset of the Kleisli category $\mathcal{K}\ell(T)$ carries an order $\sqsubseteq$. A functor F and a monad T constitute an *infinite trace situation* with respect to $\sqsubseteq$ if they satisfy the following conditions.

- There exists a final F-coalgebra $\zeta : Z \to FZ$ in $\mathbb{C}$.
- There exists a distributive law $\lambda : FT \Rightarrow TF$, yielding a lifting $\overline{F}$ on $\mathcal{K}\ell(T)$ of F.
- For each coalgebra $c : X \relbar\joinrel\mapstochar\rightarrow \overline{F}X$ in $\mathcal{K}\ell(T)$, the lifting $J\zeta : Z \relbar\joinrel\mapstochar\rightarrow \overline{F}Z$ of ζ admits the largest homomorphism. That is, there exists a homomorphism $\mathsf{tr}^\infty(c) : X \relbar\joinrel\mapstochar\rightarrow Z$ from c to $J\zeta$ such that, for any homomorphism f from c to $J\zeta$, $f \sqsubseteq \mathsf{tr}^\infty(c)$ holds.

In [8, 9, 18] we augment a coalgebra with an explicit arrow for initial states. The resulting notion is called a (T, F)-system.

▶ **Definition 3.2** (infinite trace semantics for (T, F)-systems [10, 11]). Let $\mathbb{C}$ be a category with a final object $1 \in \mathbb{C}$. A (T, F)*-system* is a triple $\mathcal{X} = (X, s, c)$ consisting of a *state space* $X \in \mathbb{C}$, a Kleisli arrow $s : 1 \relbar\joinrel\mapstochar\rightarrow X$ for *initial states*, and $c : X \relbar\joinrel\mapstochar\rightarrow \overline{F}X$ for *transition.* Let us assume that the endofunctor F and the monad T on $\mathbb{C}$ constitute an infinite trace situation. The *coalgebraic infinite trace semantics* of a (T, F)-system $\mathcal{X} = (X, s, c)$ is the Kleisli arrow $\mathsf{tr}^\infty(c) \odot s : 1 \relbar\joinrel\mapstochar\rightarrow Z$ (see the diagram, in $\mathcal{K}\ell(T)$, on the right).

$$\begin{tikzcd}
\overline{F}X \arrow[r, rightsquigarrow, "\overline{F}(\mathsf{tr}^\infty(c))"] & \overline{F}Z \\
X \arrow[u, "c"] \arrow[r, rightsquigarrow, "\mathsf{tr}^\infty(c)"] \arrow[ur, phantom, "="] & Z \arrow[u, "J\zeta"'] \\
1 \arrow[u, "s"] &
\end{tikzcd}$$

Suppose that we are given two (T, F)-systems $\mathcal{X} = (X, s, c)$ and $\mathcal{Y} = (Y, t, d)$. Let us say we aim to prove the inclusion between infinite trace semantics, that is, to show $\mathsf{tr}^{\infty}(c) \odot s \sqsubseteq \mathsf{tr}^{\infty}(d) \odot t$ with respect to the order in the homset of $\mathcal{K}\ell(T)$. Our goal in this paper is to offer *Kleisli simulations* as a sound means to do so.

The notions of *forward* and *backward Kleisli simulation* are introduced in [8] as a categorical generalization of fwd./bwd. simulations in [14]. They are defined as Kleisli arrows between (the state spaces of) two (T, F)-system that are subject to certain inequalities – in short they are *lax/oplax coalgebra homomorphisms.* In [8] they are shown to be sound with respect to *finite* trace semantics – the languages of finite words, concretely; and the unique arrow to a lifted initial algebra (that is a final coalgebra, see [10] and the introduction), abstractly. In this paper we are interested in their relation to *infinite* trace semantics.

▶ **Definition 3.3** (fwd./bwd. Kleisli simulation [8])**.** Let F be an endofunctor and T be a monad on $\mathbb{C}$ such that each homset of $\mathcal{K}\ell(T)$ carries an order $\sqsubseteq$. Let $\mathcal{X} = (X, s, c)$ and $\mathcal{Y} = (Y, t, d)$ be (T, F)-systems.
A *forward Kleisli simulation* from $\mathcal{X}$ to $\mathcal{Y}$ is a Kleisli arrow $f : Y \rightarrow\!\!\!\!\!\!+\; X$ that satisfies the following conditions (see the diagram).

$$s \sqsubseteq f \odot t, \quad \text{and} \quad c \odot f \sqsubseteq \overline{F}f \odot d.$$

$$\begin{tikzcd}
\overline{F}X & \sqsubseteq & \overline{F}Y \arrow[ll, "|" marking, "\overline{F}f"'] \\
X \arrow[u, "|" marking, "c"'] & \sqsubseteq & Y \arrow[ll, "|" marking, "f"'] \arrow[u, "|" marking, "d"'] \\
& 1 \arrow[ul, bend left, "s"] \arrow[ur, bend right, "t"'] &
\end{tikzcd}$$

We write $\mathcal{X} \sqsubseteq_{\mathbf{F}} \mathcal{Y}$ if there exists a forward simulation from $\mathcal{X}$ to $\mathcal{Y}$.
A *backward Kleisli simulation* from $\mathcal{X}$ to $\mathcal{Y}$ is a Kleisli arrow $b : X \rightarrow\!\!\!\!\!\!+\; Y$ that satisfies the following conditions (see the diagram).

$$b \odot s \sqsubseteq t, \quad \text{and} \quad \overline{F}b \odot c \sqsubseteq d \odot f.$$

$$\begin{tikzcd}
\overline{F}X \arrow[rr, "|" marking, "\overline{F}b"] & \sqsubseteq & \overline{F}Y \\
X \arrow[u, "|" marking, "c"'] \arrow[rr, "|" marking, "b"] & \sqsubseteq & Y \arrow[u, "|" marking, "d"'] \\
& 1 \arrow[ul, bend left, "s"] \arrow[ur, bend right, "t"'] &
\end{tikzcd}$$

We write $\mathcal{X} \sqsubseteq_{\mathbf{B}} \mathcal{Y}$ if there exists a backward simulation from $\mathcal{X}$ to $\mathcal{Y}$.

Forward partial execution (FPE) is a transformation of a (T, F)-system introduced in [18] for the purpose of aiding discovery of Kleisli simulations. Intuitively, it "executes" the given system by one step.

▶ **Definition 3.4** (FPE [18])**.** Let F be an endofunctor and T be a monad on $\mathbb{C}$. *Forward partial execution* (FPE) is a transformation that takes a (T, F)-system $\mathcal{X} = (X, s, c)$ as an input and returns a (T, F)-system $\mathcal{X}_{\mathsf{FPE}} = (\overline{F}X, c \odot s, \overline{F}c)$ as an output.

It is shown in [18] that FPE is a valid technique for establishing inclusion of *finite* trace semantics, in the technical senses of *soundness* and *adequacy.* Soundness asserts that discovery of a Kleisli simulation after applying FPE indeed witnesses trace inclusion between the original systems; adequacy asserts that if there is a Kleisli simulation between the original systems, then there is too between the transformed systems. In this paper, naturally, we wish to establish the same results for *infinite* trace semantics.

4 Systems with Nondeterministic Branching

In the rest of the paper we develop a coalgebraic theory of infinite traces and (Kleisli) simulations – the main contribution of the paper. We do so separately for the nondeterministic setting ($T = \mathcal{P}$) and for the probabilistic one ($T = \mathcal{G}$). This is because of the difference in the constructions of infinite traces, and consequently in the soundness proofs.

In this section we focus on the nondeterministic setting; we assume that F is a polynomial functor.

4.1 Construction of Infinite Traces

The following is already known from [11].

▶ **Theorem 4.1.** *The combination of polynomial F and $T = \mathcal{P}$ constitute an infinite trace situation (Def. 3.1).*

The proof in [11] combines fibrational intuitions with some constructions that are specific to **Sets**. Here we present a different proof. It exploits an order-theoretic structure of the Kleisli category $\mathcal{K}\ell(\mathcal{P})$; this will be useful later in showing soundness of (restricted) backward simulations. Our proof also paves the way to the probabilistic case in §5.

In fact, our proof of Thm. 4.1 is stated axiomatically, in the form of the following proposition. This is potentially useful in identifying new examples other than the combination of polynomial F and $T = \mathcal{P}$ (although we have not yet managed to do so). It is essentially the construction of a greatest fixed point by transfinite induction [5].

▶ **Proposition 4.2.**[†] *Let $\mathbb{C}$ be a category, F be an endofunctor on $\mathbb{C}$, and T be a monad on $\mathbb{C}$. Assume the following conditions.*

1. *There exists a final F-coalgebra $\zeta : Z \to FZ$ in $\mathbb{C}$.*
2. *There exists a distributive law $\lambda : FT \Rightarrow TF$, yielding a lifting $\overline{F}$ on $\mathcal{K}\ell(T)$ of F.*
3. *For each $X, Y \in \mathcal{K}\ell(T)$, the homset $\mathcal{K}\ell(T)(X, Y)$ carries a partial order $\sqsubseteq$. Moreover, $\overline{F}$'s action on arrows, as well as composition of arrows in $\mathcal{K}\ell(T)$, is monotone with respect to this order.*
4. *For each $X, Y \in \mathcal{K}\ell(T)$, every (possibly transfinite) decreasing sequence in $\mathcal{K}\ell(T)(X, Y)$ has the greatest lower bound. That is: let $\mathfrak{a}$ be a limit ordinal and $(g_\mathfrak{i} : X \nrightarrow Y)_{\mathfrak{i}<\mathfrak{a}}$ be a family of arrows such that $\mathfrak{i} \le \mathfrak{j}$ implies $g_\mathfrak{i} \sqsupseteq g_\mathfrak{j}$. Then $\bigsqcap_{\mathfrak{i}<\mathfrak{a}} g_\mathfrak{i}$ exists.*
5. *For each $X \in \mathbb{C}$, the homset $\mathcal{K}\ell(T)(X, Z)$ has the largest element $\top_{X,Z}$.*

Then T and F constitute an infinite trace situation with respect to $\sqsubseteq$.

Proof. Let $c : X \nrightarrow \overline{F}X$ be an $\overline{F}$-coalgebra in $\mathcal{K}\ell(T)$. We shall construct the largest homomorphism $\mathsf{tr}^\infty(c) : X \nrightarrow Z$ from c to $J\zeta$, by transfinite induction.
We define an endofunction $\Phi_X : \mathcal{K}\ell(T)(X, Z) \to \mathcal{K}\ell(T)(X, Z)$ by $\Phi_X(f) = J\zeta^{-1} \odot \overline{F}f \odot c$. By the monotonicity of $\odot$ and $\overline{F}$ (Assumption 3), Φ_X is also monotone. For each ordinal $\mathfrak{a}$, we define $\Phi^\mathfrak{a}_X(\top_{X,Z}) \in \mathcal{K}\ell(T)(X, Z)$ by the following transfinite induction.

$$\begin{tikzcd}
\overline{F}X \arrow[r, "\overline{F}\top_{X,Z}", "|" marking] \arrow[dr, phantom, "\sqsubseteq"] & \overline{F}Z \\
X \arrow[u, "c", "|" marking] \arrow[r, "\top_{X,Z}"', "|" marking] & Z \arrow[u, "J\zeta"', "\cong", "|" marking]
\end{tikzcd}$$

- $\Phi^0_X(\top_{X,Z}) = \top_{X,Z}$.
- For a successor ordinal $\mathfrak{a}$, $\Phi^\mathfrak{a}_X(\top_{X,Z}) = \Phi_X(\Phi^{\mathfrak{a}-1}_X(\top_{X,Z}))$.
- For a limit ordinal $\mathfrak{a}$, $\Phi^\mathfrak{a}_X(\top_{X,Z}) = \bigsqcap_{\mathfrak{i}<\mathfrak{a}} \Phi^\mathfrak{i}_X(\top_{X,Z})$. (cf. Assumption 4)

We define $\mathfrak{l}$ to be the smallest ordinal such that the cardinality of $\mathfrak{l}$ is greater than that of $\mathcal{K}\ell(T)(X, Z)$. Then from [5], $\Phi^\mathfrak{l}_X(\top_{X,Z})$ is the greatest fixed point of Φ_X. This immediately implies that $\Phi^\mathfrak{l}_X(\top_{X,Z})$ is the largest homomorphism from c to $J\zeta$. ◀

Note that the local continuity of composition in $\mathcal{K}\ell(T)$ is not assumed. This is because $\mathcal{P}$ – our choice for T in this section – does not satisfy it. Indeed, consider $f : X \nrightarrow Y$ and a decreasing sequence $(g_i : Y \nrightarrow Z)_{i\in\omega}$, both in $\mathcal{K}\ell(\mathcal{P})$. Then we have $\big(\bigsqcap_{i\in\omega} g_i\big) \odot f(x) = \bigcup_{y\in f(x)} \bigcap_{i\in\omega} g_i(y)$ while $\bigsqcap_{i\in\omega}(g_i \odot f)(x) = \bigcap_{i\in\omega} \bigcup_{y\in f(x)} g_i(y)$, and these two are not equal in general (e.g. Example A.31). This failure of continuity prevents us from applying the (simpler) *Kleene fixed-point theorem*, in which induction terminates after ω steps.

There does exist a nondeterministic automaton for which the largest homomorphism is obtained after steps bigger than ω; see Example A.31.

It is easy to check that all the assumptions in Prop. 4.2 are satisfied by polynomial F and $T = \mathcal{P}$. This yields Thm. 4.1. We can also show that the resulting coalgebraic infinite trace semantics coincides with the usual definition of (infinite) tree languages for nondeterministic systems. See §A.2.1 for details.

4.2 Kleisli Simulations for Nondeterministic Systems

4.2.1 Forward Simulations

Soundness of forward simulation is not hard; we do not have to go into the construction in Prop. 4.2.

▶ **Theorem 4.3.** *Given two $(\mathcal{P}, F)$-systems $\mathcal{X} = (X, s, c)$ and $\mathcal{Y} = (Y, t, d)$, $\mathcal{X} \sqsubseteq_{\mathbf{F}} \mathcal{Y}$ implies $\mathsf{tr}^\infty(c) \odot s \sqsubseteq \mathsf{tr}^\infty(d) \odot t$.*

The proof, again, is formulated as a general result, singling out some sufficient axioms.

▶ **Lemma 4.4.**[†] *Let F be an endofunctor and T be a monad on $\mathbb{C}$; assume further that they constitute an infinite trace situation (with respect to $\sqsubseteq$). We assume the following conditions.*

1. *Each homset of $\mathcal{K}\ell(T)$ is ω-complete, that is, each increasing ω-sequence in it has the lub.*

2. *Composition $\odot$ of arrows in $\mathcal{K}\ell(T)$ and $\overline{F}$'s action on arrows are both ω-continuous (i.e. they preserve the lub. of an increasing ω-sequence). It follows that they are both monotone.*

For two (T, F)-systems $\mathcal{X} = (X, s, c)$ and $\mathcal{Y} = (Y, t, d)$, if $f : Y \nrightarrow X$ is a forward simulation from $\mathcal{X}$ to $\mathcal{Y}$, then $\mathsf{tr}^\infty(c) \odot f \sqsubseteq \mathsf{tr}^\infty(d)$. As a consequence we have $\mathsf{tr}^\infty(c) \odot s \sqsubseteq \mathsf{tr}^\infty(d) \odot t$.

Proof. Let $\zeta : Z \to FZ$ be a final F-coalgebra in $\mathbb{C}$. We define a function $\Phi_Y : \mathcal{K}\ell(T)(Y, Z) \to \mathcal{K}\ell(T)(Y, Z)$ by $\Phi_Y(g) = J\zeta^{-1} \odot \overline{F}g \odot d$; note that ζ is a final coalgebra and hence an isomorphism. Then

$$\begin{aligned} \mathsf{tr}^\infty(c) \odot f &= J\zeta^{-1} \odot \overline{F}(\mathsf{tr}^\infty(c)) \odot c \odot f && (\mathsf{tr}^\infty(c) \text{ is a homomorphism}) \\ &\sqsubseteq \Phi_Y(\mathsf{tr}^\infty(c) \odot f) && (f \text{ is a fwd. sim., and the definition of } \Phi_Y). \end{aligned}$$

By the assumption that $\overline{F}$ and the composition are monotone, Φ_Y is also monotone. Therefore by repeatedly applying Φ_Y to the both sides of the above inequality, we obtain an increasing sequence $\mathsf{tr}^\infty(c) \odot f \sqsubseteq \Phi_Y(\mathsf{tr}^\infty(c) \odot f) \sqsubseteq \Phi_Y^2(\mathsf{tr}^\infty(c) \odot f) \sqsubseteq \cdots$ in $\mathcal{K}\ell(T)(Y, Z)$. As $\mathcal{K}\ell(T)(Y, Z)$ is ω-complete, the least upper bound $\bigsqcup_{i<\omega} \Phi^i(\mathsf{tr}^\infty(c) \odot f)$ exists. By the assumption that $\overline{F}$ and $\odot$ are both locally ω-continuous, Φ_Y is also ω-continuous. Therefore $\bigsqcup_{i<\omega} \Phi^i(\mathsf{tr}^\infty(c) \odot f)$ is a fixed point of Φ_Y, and hence a homomorphism from d to $J\zeta$. As $\mathsf{tr}^\infty(d)$ is the largest homomorphism from d to $J\zeta$, this implies $\mathsf{tr}^\infty(c) \odot f \sqsubseteq \bigsqcup_{i<\omega} \Phi^i(\mathsf{tr}^\infty(c) \odot f) \sqsubseteq \mathsf{tr}^\infty(d)$. Combining with the assumption that f is a forward simulation (its condition on initial states), we have $\mathsf{tr}^\infty(c) \odot s \sqsubseteq \mathsf{tr}^\infty(c) \odot f \odot t \sqsubseteq \mathsf{tr}^\infty(d) \odot t$. ◀

$$\begin{tikzcd}
\overline{F}Y \arrow[rr, "\shortmid" marking, "\overline{F}f"'] \arrow[rrrr, bend left, "\shortmid" marking, "\overline{F}(\mathsf{tr}^\infty(d))"] & & \overline{F}X \arrow[rr, "\shortmid" marking, "\overline{F}(\mathsf{tr}^\infty(c))"'] & & \overline{F}Z \\
& \sqsupseteq & & = & \\
Y \arrow[uu, "\shortmid" marking, "d"'] \arrow[rr, "\shortmid" marking, "f"] \arrow[rrrr, bend right, "\shortmid" marking, "\mathsf{tr}^\infty(d)"'] & & X \arrow[uu, "\shortmid" marking, "c"'] \arrow[rr, "\shortmid" marking, "\mathsf{tr}^\infty(c)"] & & Z \arrow[uu, "\shortmid" marking, "J\zeta"'] \\
& \sqsupseteq & & & \\
& 1 \arrow[uul, "\shortmid" marking, "t"] \arrow[uur, "\shortmid" marking, "s"'] & & &
\end{tikzcd}$$

It is known from [10] that the combination of polynomial F and $T = \mathcal{P}$ satisfy the conditions of Lem. 4.4. Hence we obtain Thm. 4.3, i.e. soundness of fwd. simulation in the nondeterministic setting.

4.2.2 Backward Simulations

Next we wish to establish soundness of *backward* Kleisli simulations with respect to *infinite* traces (for finite traces it is shown in [8]). In fact, the desired soundness fails in general – a counterexample is in Example A.32. It turns out that we can impose certain restrictions on backward Kleisli simulations and ensure soundness.

▶ **Definition 4.5** (totality, image-finiteness, TIF-backward simulation)**.** Let $\mathcal{X} = (X, s, c)$ and $\mathcal{Y} = (Y, t, d)$ be $(\mathcal{P}, F)$-systems. A backward simulation $b : X \mathrel{\ooalign{\hfil$\mapstochar\mkern5mu$\hfil\cr$\to$\cr}} Y$ from $\mathcal{X}$ to $\mathcal{Y}$ is *total* if $b(x) \neq \emptyset$ for all $x \in X$; it is *image-finite* if $b(x)$ is finite for all $x \in X$. If b satisfies both of the two conditions, it is called a *TIF-backward simulation.* We write $\mathcal{X} \sqsubseteq^{\mathrm{TIF}}_{\mathbf{B}} \mathcal{Y}$ if there exists a TIF-backward simulation from $\mathcal{X}$ to $\mathcal{Y}$.

▶ **Theorem 4.6** (soundness of $\sqsubseteq^{\mathrm{TIF}}_{\mathbf{B}}$)**.** *For two $(\mathcal{P}, F)$-systems $\mathcal{X} = (X, s, c)$ and $\mathcal{Y} = (Y, t, d)$, $\mathcal{X} \sqsubseteq^{\mathrm{TIF}}_{\mathbf{B}} \mathcal{Y}$ implies $\mathsf{tr}^{\infty}(c) \odot s \sqsubseteq \mathsf{tr}^{\infty}(d) \odot t$.*

The proof of Thm. 4.6 is, yet again, via the following axiomatic development.

▶ **Definition 4.7** (TIF-backward simulation, generally)**.**[†] Let F be an endofunctor and T be a monad on $\mathbb{C}$ that satisfy the conditions in Prop. 4.2 wrt. $\sqsubseteq$. For two (T, F)-systems $\mathcal{X} = (X, s, c)$ and $\mathcal{Y} = (Y, t, d)$, a *TIF-backward simulation* from $\mathcal{X}$ to $\mathcal{Y}$ is a backward simulation $b : X \mathrel{\ooalign{\hfil$\mapstochar\mkern5mu$\hfil\cr$\to$\cr}} Y$ that satisfies the following conditions.

1. The arrow $b : X \mathrel{\ooalign{\hfil$\mapstochar\mkern5mu$\hfil\cr$\to$\cr}} Y$ satisfies $\top_{Y,Z} \odot b = \top_{X,Z}$.
2. Precomposing $b : X \mathrel{\ooalign{\hfil$\mapstochar\mkern5mu$\hfil\cr$\to$\cr}} Y$ preserves the greatest lower bound of any decreasing transfinite sequence. That is, let $A \in \mathcal{K}\ell(T)$, $\mathfrak{a}$ be a limit ordinal, and $(g_{\mathfrak{i}} : Y \mathrel{\ooalign{\hfil$\mapstochar\mkern5mu$\hfil\cr$\to$\cr}} A)_{\mathfrak{i} < \mathfrak{a}}$ be a family of Kleisli arrows such that $\mathfrak{i} \leq \mathfrak{j}$ implies $g_{\mathfrak{i}} \sqsupseteq g_{\mathfrak{j}}$. Then we have $\bigsqcap_{\mathfrak{i} \in \mathfrak{a}} (g_{\mathfrak{i}} \odot b) = (\bigsqcap_{\mathfrak{i} \in \mathfrak{a}} g_{\mathfrak{i}}) \odot b$.

We write $\mathcal{X} \sqsubseteq^{\mathrm{TIF}}_{\mathbf{B}} \mathcal{Y}$ if there exists a TIF-backward simulation from $\mathcal{X}$ to $\mathcal{Y}$.

Assumption 2 of Def. 4.7 resembles how "finiteness" is formulated in category theory, e.g. in the definition of *finitary* objects.

This general TIF-backward simulation satisfies soundness. For its proof we have to look into the inductive construction of the largest homomorphism in §4.1.

▶ **Lemma 4.8.**[†] *Let F and T be as in Prop. 4.2. For two (T, F)-systems $\mathcal{X} = (X, s, c)$ and $\mathcal{Y} = (Y, t, d)$, $\mathcal{X} \sqsubseteq^{\mathrm{TIF}}_{\mathbf{B}} \mathcal{Y}$ (in the sense of Def. 4.7) implies $\mathsf{tr}^{\infty}(c) \sqsubseteq \mathsf{tr}^{\infty}(d) \odot b$. Furthermore it follows that $\mathsf{tr}^{\infty}(c) \odot s \sqsubseteq \mathsf{tr}^{\infty}(d) \odot t$.*

Proof.

Let $\zeta : Z \to FZ$ be a final F-coalgebra in $\mathbb{C}$. We define $\Phi_X : \mathcal{K}\ell(T)(X, Z) \to \mathcal{K}\ell(T)(X, Z)$ and $\Phi_Y : \mathcal{K}\ell(T)(Y, Z) \to \mathcal{K}\ell(T)(Y, Z)$ as in the proof of Prop. 4.2. Moreover, in the same manner as in the proof of Prop. 4.2, for each ordinal $\mathfrak{a}$, we define $\Phi^{\mathfrak{a}}_X(\top_{X,Z}) : X \mathrel{\ooalign{\hfil$\mapstochar\mkern5mu$\hfil\cr$\to$\cr}} Z$ and $\Phi^{\mathfrak{a}}_Y(\top_{Y,Z}) : Y \mathrel{\ooalign{\hfil$\mapstochar\mkern5mu$\hfil\cr$\to$\cr}} Z$ by the transfinite induction on $\mathfrak{a}$. As we have seen in the proof of Prop. 4.2, there exist ordinals $\mathfrak{l}_X$ and $\mathfrak{l}_Y$ s.t. $\mathsf{tr}^{\infty}(c) = \Phi^{\mathfrak{l}_X}_X(\top_{X,Z})$ and $\mathsf{tr}^{\infty}(d) = \Phi^{\mathfrak{l}_Y}_Y(\top_{Y,Z})$. Let $\mathfrak{l} = \max(\mathfrak{l}_X, \mathfrak{l}_Y)$. We shall now prove by transfinite induction that, for each $\mathfrak{a}$, we have $\Phi^{\mathfrak{a}}_X(\top_{X,Z}) \sqsubseteq \Phi^{\mathfrak{a}}_Y(\top_{Y,Z}) \odot b$; this will yield our goal by taking $\mathfrak{a} = \mathfrak{l}$.

$$\begin{tikzcd}
\overline{F}X \arrow[r, "|" marking, "\overline{F}b"'] \arrow[rr, bend left, "|" marking, "\overline{F}\top_{X,Z}"] & \overline{F}Y \arrow[r, "|" marking, "\overline{F}\top_{Y,Z}"'] & \overline{F}Z \\
X \arrow[u, "|" marking, "c"] \arrow[r, "|" marking, "b"] \arrow[rr, bend right, "|" marking, "\top_{X,Z}"'] \arrow[ur, phantom, "\sqsubseteq"] & Y \arrow[u, "|" marking, "d"] \arrow[r, "|" marking, "\top_{Y,Z}"] \arrow[ur, phantom, "\sqsubseteq"] & Z \arrow[u, "|" marking, "J\zeta", "\cong"']
\end{tikzcd}$$

For $\mathfrak{a} = 0$, from Assumption 1 of Def. 4.7, we have $\Phi^{\mathfrak{a}}_X(\top_{X,Z}) = \top_{X,Z} = \top_{Y,Z} \odot b = \Phi^{\mathfrak{a}}_Y(\top_{Y,Z}) \odot b$.

Assume that $\mathfrak{a}$ is a successor ordinal and $\Phi^{\mathfrak{a}-1}_X(\top_{X,Z}) \sqsubseteq \Phi^{\mathfrak{a}-1}_Y(\top_{Y,Z}) \odot b$. Then

$$\begin{aligned}
\Phi^{\mathfrak{a}}_X(\top_{X,Z}) &\sqsubseteq J\zeta^{-1} \odot \overline{F}(\Phi^{\mathfrak{a}-1}_Y(\top_{Y,Z})) \odot \overline{F}b \odot c && \text{(by induction hypothesis)} \\
&\sqsubseteq \Phi^{\mathfrak{a}}_Y(\top_{Y,Z}) \odot b && \text{(b is a bwd. simulation)}\,.
\end{aligned}$$

Let $\mathfrak{a}$ be a limit ordinal and assume that $\Phi^{\mathfrak{i}}_X(\top_{X,Z}) \sqsubseteq \Psi^{\mathfrak{i}}_Y(\top_{Y,Z}) \odot b$ for all $\mathfrak{i} < \mathfrak{a}$. Then

$$\begin{aligned} \Phi^{\mathfrak{a}}_X(\top_{X,Z}) &\sqsubseteq \textstyle\bigsqcap_{\mathfrak{i}<\mathfrak{a}} \bigl(\Phi^{\mathfrak{i}}_Y(\top_{Y,Z}) \odot b\bigr) && \text{(by induction hypothesis)} \\ &= \Phi^{\mathfrak{a}}_Y(\top_{Y,Z}) \odot b && \text{(by Assumption 2 of Def. 4.7)}\,. \end{aligned}$$

Thus $\mathsf{tr}^\infty(c) \sqsubseteq \mathsf{tr}^\infty(d) \odot b$. The last claim follows from b's condition on initial states. ◀

Proof of Thm. 4.6. In Lem. A.17 we prove that a TIF-backward simulation in the specific sense of Def. 4.5 is also a TIF-backward simulation in the general sense of Def. 4.7. Therefore Lem. 4.8 yields trace inclusion. ◀

Even with the additional constraints of totality and image-finiteness, backward Kleisli simulations are a viable method for establishing infinite trace inclusion. An example is in Example A.33 where a fwd. simulation does not exist but a TIF-bwd. simulation does.

4.3 Forward Partial Execution for Nondeterministic Systems

We now apply *forward partial execution* (FPE) [18] – a transformation of coalgebraic systems that potentially increases the likelihood of existence of simulations – in the current setting of nondeterminism and infinite traces. We follow the setting in [18] for the *finite* traces, and formulate FPE's "correctness" in the following theorem.

▶ **Theorem 4.9.** *Let F be a polynomial functor on* **Sets**. *For $(\mathcal{P}, F)$-systems $\mathcal{X} = (X, s, c)$ and $\mathcal{Y} = (Y, t, d)$, the following hold.*

1. **a.** (soundness of FPE for fwd. sim.) $\mathcal{X}_{\mathsf{FPE}} \sqsubseteq_{\mathbf{F}} \mathcal{Y}$ *implies* $\mathsf{tr}^\infty(c) \odot s \sqsubseteq \mathsf{tr}^\infty(d) \odot t$.
   b. (adequacy of FPE for fwd. sim.) $\mathcal{X} \sqsubseteq_{\mathbf{F}} \mathcal{Y}$ *implies* $\mathcal{X}_{\mathsf{FPE}} \sqsubseteq_{\mathbf{F}} \mathcal{Y}$.
2. **a.** (soundness of FPE for bwd. sim.) $\mathcal{X} \sqsubseteq^{\mathrm{TIF}}_{\mathbf{B}} \mathcal{Y}_{\mathsf{FPE}}$ *implies* $\mathsf{tr}^\infty(c) \odot s \sqsubseteq \mathsf{tr}^\infty(d) \odot t$.
   b. (adequacy of FPE for bwd. sim.) $\mathcal{X} \sqsubseteq^{\mathrm{TIF}}_{\mathbf{B}} \mathcal{Y}$ *implies* $\mathcal{X} \sqsubseteq^{\mathrm{TIF}}_{\mathbf{B}} \mathcal{Y}_{\mathsf{FPE}}$, *assuming that the following hold.*
      i. $d(y) \neq \emptyset$ *for all* $y \in Y$.
      ii. $d(y)$ *is finite for all* $y \in Y$.

Informally: *soundness* means that discovery after applying FPE still witnesses the trace inclusion between the original systems; and *adequacy* means that the relationship $\sqsubseteq_{\mathbf{F}}$ (or $\sqsubseteq^{\mathrm{TIF}}_{\mathbf{B}}$) is not destroyed by application of FPE. The theorem also implies that FPE must be applied to the "correct side" of the desired trace inclusion: $\mathcal{X}$ in the search for a fwd. simulation; and $\mathcal{Y}$ in the search for a bwd. one.

Note that the adequacy property is independent from the choice of trace semantics (finite or infinite). Therefore the statement 1b of Thm. 4.9 is the same as its counterpart in [18]. For the statement 2b, however, we have to check that the TIF restriction (that is absent in [18]) is indeed carried over.

In [18] it is shown that FPE can indeed create a simulation that does not exist between the original systems. Its practical use is witnessed by experimental results in [18], too. It would not be hard to observe the same in the current setting for *infinite* traces.

For the proof of Thm. 4.9, once again, we turn to an axiomatic development.

▶ **Theorem 4.10** (FPE and fwd. sim.).† *Let F be an endofunctor and T be a monad on $\mathbb{C}$, as in Lem. 4.4 (that is, they constitute an infinite trace situation and satisfy the two additional assumptions.) Let $\mathcal{X} = (X, s, c)$ and $\mathcal{Y} = (Y, t, d)$ be (T, F)-systems. Then we have:*

1. (soundness for fwd. sim.) $\mathcal{X}_{\mathsf{FPE}} \sqsubseteq_{\mathbf{F}} \mathcal{Y}$ *implies* $\mathsf{tr}^\infty(c) \odot s \sqsubseteq \mathsf{tr}^\infty(d) \odot t$.
2. (adequacy for fwd. sim.) $\mathcal{X} \sqsubseteq_{\mathbf{F}} \mathcal{Y}$ *implies* $\mathcal{X}_{\mathsf{FPE}} \sqsubseteq_{\mathbf{F}} \mathcal{Y}$.

▶ **Theorem 4.11** (FPE and bwd. sim.).[†] *Let F be an endofunctor and T be a monad on $\mathbb{C}$ that satisfy the conditions in Prop. 4.2 (hence those in Lem. 4.8). Let $\mathcal{X} = (X, s, c)$ and $\mathcal{Y} = (Y, t, d)$ be (T, F)-systems.*

1. (soundness for bwd. sim.) $\mathcal{X} \sqsubseteq_{\mathbf{B}}^{\mathrm{TIF}} \mathcal{Y}_{\mathsf{FPE}}$ *implies* $\mathsf{tr}^\infty(c) \odot s \sqsubseteq \mathsf{tr}^\infty(d) \odot t$.
2. (adequacy for bwd. sim.) $\mathcal{X} \sqsubseteq_{\mathbf{B}}^{\mathrm{TIF}} \mathcal{Y}$ *implies* $\mathcal{X} \sqsubseteq_{\mathbf{B}}^{\mathrm{TIF}} \mathcal{Y}_{\mathsf{FPE}}$ *if the following conditions are satisfied.*
   a. *The coalgebra $d : Y \nrightarrow \overline{F}Y$ satisfies $\top_{\overline{F}Y,Z} \odot d = \top_{Y,Z}$.*
   b. *Precomposing d preserves the glb. of a decreasing transfinite sequence.*

Proof of Thm. 4.9. 1 is immediate from Thm. 4.10. In a similar manner to Lem.A.17, we can prove 2 using Thm. 4.11. ◀

5 Systems with Probabilistic Branching

We now turn to probabilistic systems. They are modeled as $(\mathcal{G}, F)$-systems in the category **Meas**. Here we establish largely the same statements as in §4, but many constructions and proofs are different. Throughout this section F is assumed to be a (standard Borel) polynomial functor on **Meas** (Def. 2.2).

5.1 Construction of Infinite Traces

▶ **Theorem 5.1.** *The combination of polynomial F and $T = \mathcal{G}$ constitute an infinite trace situation (Def. 3.1).*

Our basic idea of the construction is similar to that for $\mathcal{P}$ (§4.1). Our goal is to construct the largest homomorphism from an $\overline{F}$-coalgebra c in to the lifted final coalgebra $J\zeta : Z \nrightarrow \overline{F}Z$; we do so inductively, starting from the top element and going down along a decreasing sequence. Compared to the nondeterministic case ($T = \mathcal{P}$), major differences are as follows.

- Composition of Kleisli arrows is ω^{op}-continuous in $\mathcal{K}\ell(\mathcal{G})$. This is an advantage, because we can appeal to the Kleene fixed point theorem and we only need inductive construction up-to ω steps (while, for $\mathcal{P}$, we needed transfinite induction).
- A big disadvantage, however, is the absence of the top element $\top_{X,Z}$ in $\mathcal{K}\ell(T)(X, Z)$. One can imagine a top element $\top_{X,Z}$ to assign 1 to every event – this is however not a (probability) measure.

To cope with the latter challenge, we turn to the *final F-sequence* in **Meas** that yields a final F-coalgebra as its limit. Instead of using a sequence like $\top \sqsupseteq \Phi(\top) \sqsupseteq \cdots$ in $\mathcal{K}\ell(T)(X, Z)$ (where the largest element $\top$ does not exist anyway), we use a decreasing sequence that goes along the final sequence.

$$\begin{tikzcd}
\overline{F}X \arrow[dd, "\overline{F}(\mathrm{tr}^\infty(c))"', "|" marking] \arrow[ddr, phantom, "="] & X \arrow[l, "c"', "|" marking] \arrow[dd, dashed, "\mathrm{tr}^\infty(c)"', "|" marking] \arrow[dr, "\pi_0", "|" marking] \arrow[drr, bend left, "\pi_1", "|" marking] \arrow[drrr, bend left, "\pi_2", "|" marking] & & & & \cdots \\
 & & 1 & \overline{F}1 \arrow[l, "J!"', "|" marking] & \overline{F}^2 1 \arrow[l, "JF!"', "|" marking] & \cdots \arrow[l, "JF^2!"', "|" marking] \\
\overline{F}Z & Z \arrow[l, "J\zeta"', "\cong", "|" marking] \arrow[ur, "J\gamma_0"', "|" marking] \arrow[urr, bend right, "J\gamma_1"', "|" marking] \arrow[urrr, bend right, "J\gamma_2"', "|" marking] & & & & \cdots
\end{tikzcd}$$

The precise construction is found in the proof of the following proposition (the proof is in Appendix A.4.

▶ **Proposition 5.2.**[†] *Let $\mathbb{C}$ be a category, F be an endofunctor on $\mathbb{C}$, and T be a monad on $\mathbb{C}$ where each homset of $\mathcal{K}\ell(T)$ carries an order $\sqsubseteq$. We assume the following conditions.*

1. *The category $\mathbb{C}$ has a final object 1; the final sequence $1 \xleftarrow{!_{F1}} F1 \xleftarrow{F!_{F1}} F^2 1 \xleftarrow{F^2!_{F1}} \ldots$ has a limit $(Z, (\gamma_i : Z \to F^i 1)_{i \in \omega})$; and moreover, F preserves this limit. (Hence the limit carries a final F-coalgebra [1].)*
2. *There exists a distributive law $\lambda : FT \Rightarrow TF$, yielding a lifting $\overline{F}$ on $\mathcal{K}\ell(T)$ of F.*

3. *For $X, Y \in \mathcal{K}\ell(T)$, every decreasing ω-sequence $f_0 \sqsupseteq f_1 \sqsupseteq \dots$ in $\mathcal{K}\ell(T)(X,Y)$ has the greatest lower bound $\bigsqcap_{i\in\omega} f_i$. Moreover, composition of arrows in $\mathcal{K}\ell(T)$ and $\overline{F}$'s action on arrows are both ω^{op}-continuous. That is, for each $g : Z \nrightarrow X$ and $h : Y \nrightarrow W$, we have $g \odot (\bigsqcap_{i\in\omega} f_i) = \bigsqcap_{i\in\omega}(g \odot f_i)$, $(\bigsqcap_{i\in\omega} f_i) \odot h = \bigsqcap_{i\in\omega}(f_i \odot h)$, and $\overline{F}(\bigsqcap_{i\in\omega} f_i) = \bigsqcap_{i\in\omega}(\overline{F} f_i)$.*
4. *The lifting $J(!_X)$ of the unique arrow to 1 is the largest element of $\mathcal{K}\ell(T)(X,1)$.*
5. *The functor J lifts the limit in Assumption 1 to a 2-limit. Namely, for any cone $(X, (\pi_i : X \nrightarrow F^i 1)_{i\in\omega})$ over the sequence $1 \overset{J!_{F1}}{\nleftarrow} \overline{F}1 \overset{JF!_{F1}}{\nleftarrow} \overline{F}^2 1 \overset{JF^2!_{F1}}{\nleftarrow} \cdots$, there uniquely exists $l : X \nrightarrow Z$ s.t. $\pi_i = J\gamma_i \odot l$ holds for each $i \in \omega$. Moreover, if $l' : X \nrightarrow Z$ satisfies $J\gamma_i \odot l' \sqsubseteq J\gamma_i \odot l$ for each $i \in \omega$, then $l' \sqsubseteq l$ holds.*

Then F and T constitute an infinite trace situation with respect to $\sqsubseteq$.

In more elementary terms, Assumption 5 asserts that: J lifts the limit Z; and the lifted limit satisfies a stronger condition of “carrying over” the order between cones to the order between mediating maps.

Proof of Thm. 5.1. We have to check that polynomial F and $T = \mathcal{G}$ satisfy the assumptions in Prop. 5.2. The most nontrivial is Assumption 5; there we rely on Kolmogorov's consistency theorem, for the fact that a limit is lifted to a limit. That the latter is indeed a 2-limit is not hard, exploiting suitable monotonicity. Details are found in Lem. A.18. ◀

We can also show that the resulting coalgebraic infinite trace semantics coincides with the usual definition of (infinite) tree languages for probabilistic systems. See §A.2.2 for details.

5.2 Kleisli Simulations for Probabilistic Systems

5.2.1 Forward Simulations

Soundness of forward simulation, in the current probabilistic setting, follows immediately from the the axiomatic development in Lem. 4.4.

▶ **Theorem 5.3.** *Given two $(\mathcal{G}, F)$-systems $\mathcal{X} = (X, s, c)$ and $\mathcal{Y} = (Y, t, d)$, $\mathcal{X} \sqsubseteq_{\mathbf{F}} \mathcal{Y}$ implies $\mathsf{tr}^\infty(c) \odot s \sqsubseteq \mathsf{tr}^\infty(d) \odot t$.*

5.2.2 Backward Simulations

Next we turn to backward simulations. Similarly to nondeterministic setting (§4.2.2), we have to impose a certain restriction on backward Kleisli simulations to ensure soundness. By the feature of $\mathcal{G}$ that composition in $\mathcal{K}\ell(\mathcal{G})$ is ω-continuous, the image-finiteness condition is no longer needed.

▶ **Definition 5.4** (totality, T-backward simulation)**.** Let $\mathcal{X} = (X, s, c)$ and $\mathcal{Y} = (Y, t, d)$ be $(\mathcal{G}, F)$-systems. A backward simulation $b : X \nrightarrow Y$ from $\mathcal{X}$ to $\mathcal{Y}$ is *total* if $b(x)(Y) = 1$ for all $x \in X$. If b is total, it is called a *T-backward simulation*. We write $\mathcal{X} \sqsubseteq^{\mathrm{T}}_{\mathbf{B}} \mathcal{Y}$ if there exists a T-backward simulation from $\mathcal{X}$ to $\mathcal{Y}$.

▶ **Theorem 5.5** (soundness of $\sqsubseteq^{\mathrm{T}}_{\mathbf{B}}$)**.** *For two $(\mathcal{G}, F)$-systems $\mathcal{X} = (X, s, c)$ and $\mathcal{Y} = (Y, t, d)$, $\mathcal{X} \sqsubseteq^{\mathrm{T}}_{\mathbf{B}} \mathcal{Y}$ implies $\mathsf{tr}^\infty(c) \odot s \sqsubseteq \mathsf{tr}^\infty(d) \odot t$.*

The proof of Thm. 5.5 is via the following axiomatic development.

▶ **Definition 5.6** (T-backward simulation, generally)**.**[†] Let F be an endofunctor and T be a monad on $\mathbb{C}$ that satisfy the conditions in Prop. 5.2 wrt. $\sqsubseteq$. For two (T, F)-systems $\mathcal{X} = (X, s, c)$ and $\mathcal{Y} = (Y, t, d)$, a *T-backward simulation* from $\mathcal{X}$ to $\mathcal{Y}$ is a backward simulation $b : X \nrightarrow Y$ that satisfies the following condition:

1. The arrow $b : X \nrightarrow Y$ satisfies $J!_Y \odot b = J!_X$. Here $!_Y : Y \to 1$ is the unique function.
We write $\mathcal{X} \sqsubseteq^{\mathrm{T}}_{\mathbf{B}} \mathcal{Y}$ if there exists a T-backward simulation from $\mathcal{X}$ to $\mathcal{Y}$.

This general T-backward simulation satisfies soundness. For its proof we have to look into the inductive construction of the largest homomorphism in §5.1 (Prop. 5.2).

▶ **Lemma 5.7.**† *Let F and T be as in Prop. 5.2. For two (T, F)-systems $\mathcal{X} = (X, s, c)$ and $\mathcal{Y} = (Y, t, d)$, $\mathcal{X} \sqsubseteq^{\mathrm{T}}_{\mathbf{B}} \mathcal{Y}$ (in the sense of Def. 5.6) implies $\mathsf{tr}^{\infty}(c) \sqsubseteq \mathsf{tr}^{\infty}(d) \odot b$. Furthermore it follows that $\mathsf{tr}^{\infty}(c) \odot s \sqsubseteq \mathsf{tr}^{\infty}(d) \odot t$.*

Proof of Thm. 5.5. In Lem. A.19 we prove that a T-backward simulation in the specific sense of Def. 5.4 is also a T-backward simulation in the general sense of Def. 5.4. Therefore Lem. 5.7 yields trace inclusion. ◀

5.3 Forward Partial Execution for Probabilistic Systems

We show that FPE can be used to aid discovery of forward and backward simulations, also in the current probabilistic setting.

▶ **Theorem 5.8.** *Let F be a polynomial functor on* **Meas**. *For $(\mathcal{G}, F)$-systems $\mathcal{X} = (X, s, c)$ and $\mathcal{Y} = (Y, t, d)$, the following hold.*

1. **a.** (soundness of FPE for fwd. sim.) $\mathcal{X}_{\mathsf{FPE}} \sqsubseteq_{\mathbf{F}} \mathcal{Y}$ *implies* $\mathsf{tr}^{\infty}(c) \odot s \sqsubseteq \mathsf{tr}^{\infty}(d) \odot t$.
b. (adequacy of FPE for fwd. sim.) $\mathcal{X} \sqsubseteq_{\mathbf{F}} \mathcal{Y}$ *implies* $\mathcal{X}_{\mathsf{FPE}} \sqsubseteq_{\mathbf{F}} \mathcal{Y}$.

2. **a.** (soundness of FPE for bwd. sim.) $\mathcal{X} \sqsubseteq^{\mathrm{T}}_{\mathbf{B}} \mathcal{Y}_{\mathsf{FPE}}$ *implies* $\mathsf{tr}^{\infty}(c) \odot s \sqsubseteq \mathsf{tr}^{\infty}(d) \odot t$.
b. (adequacy of FPE for bwd. sim.) $\mathcal{X} \sqsubseteq^{\mathrm{T}}_{\mathbf{B}} \mathcal{Y}$ *implies* $\mathcal{X} \sqsubseteq^{\mathrm{T}}_{\mathbf{B}} \mathcal{Y}_{\mathsf{FPE}}$, *assuming that:* $d(y)(FY) = 1$ *for all* $y \in Y$.

The item 1 for forward simulations follows immediately from Thm. 4.10. For the relationship to backward simulations, we develop another general result.

▶ **Theorem 5.9** (FPE and bwd. sim.).† *Let F be an endofunctor and T be a monad on $\mathbb{C}$ that satisfy the conditions in Prop. 5.2 (hence those in Lem. 5.7). Let $\mathcal{X} = (X, s, c)$ and $\mathcal{Y} = (Y, t, d)$ be (T, F)-systems.*

1. (soundness for bwd. sim.) $\mathcal{X} \sqsubseteq^{\mathrm{T}}_{\mathbf{B}} \mathcal{Y}_{\mathsf{FPE}}$ *implies* $\mathsf{tr}^{\infty}(c) \odot s \sqsubseteq \mathsf{tr}^{\infty}(d) \odot t$.
2. (adequacy for bwd. sim.) $\mathcal{X} \sqsubseteq^{\mathrm{T}}_{\mathbf{B}} \mathcal{Y}$ *implies* $\mathcal{X} \sqsubseteq^{\mathrm{T}}_{\mathbf{B}} \mathcal{Y}_{\mathsf{FPE}}$, *assuming that: the coalgebra $d : Y \nrightarrow \overline{F}Y$ satisfies $J!_{FY} \odot d = J!_Y$.*

Proof of Thm. 5.8. The item 1 is immediate from Thm. 4.10. In a similar manner to Lem. A.19, we can prove the item 2 using Thm. 5.9. ◀

6 Systems with Other Branching Types

In this section we briefly discuss two more pairs of F and T that constitute infinite trace situations.

The first pair is a polynomial functor F on **Sets** and the *lift monad* $\mathcal{L}$. For a given set $X \in$ **Sets**, $\mathcal{L}X$ is given by $\{\perp\} + X$. The added element $\perp$ represents the aborting or non-termination of the program, and hence an $(\mathcal{L}, F)$-system can be regarded as a *tree automaton with exception*. To show that F and $\mathcal{L}$ constitute an infinite trace situation, we rely on Prop. 5.2 (but not Prop. 4.2, since $\mathcal{L}X$ does not have the greatest element). Therefore, much like for $\mathcal{G}$, we can check trace inclusion by forward or T-backward simulations (see §5.2). More details are found in §A.5.

The second pair is that of polynomial F on **Sets** and the *subdistribution monad* $\mathcal{D}$. For a given set $X \in$ **Sets**, $\mathcal{D}X$ is the set $\{d\colon X \to [0,1] \mid \sum_{x\in X} d(x) \le 1\}$ of (discrete) subdistributions over X. The subdistribution monad $\mathcal{D}$ is similar to the sub-Giry monad $\mathcal{G}$, and a $(\mathcal{D}, F)$-system can be also regarded as a probabilistic tree automaton. We can prove that F and $\mathcal{D}$ constitute an infinite trace situation. The resulting infinite trace semantics has limited use, however, due to the discrete nature of an arrow $X \nrightarrow \mathcal{D}Z$ (it assigns a probability to a single tree and the probability is most of the time 0; see Example 1.1). Another difficulty is that infinite traces for $T = \mathcal{D}$ does not follow from either of our general results (Prop. 4.2 or Prop. 5.2) – in §A.6 we construct infinite traces for $T = \mathcal{D}$ in concrete terms. This prevents us from applying the general theories for Kleisli simulations in §4–5. For more details, see §A.6.

7 Related Work

The construction of the largest homomorphism given in Prop. 5.2 is based on the one in [4]. The latter imposes some technical conditions on a monad T, including a "totality" condition that excludes $T = \mathcal{P}$ from its instances (the nonempty powerset monad is an instance). Our assumption of lifting to a 2-limit (Assumption 5 in Prop. 5.2) is inspired by a condition in [4], namely that the limit Z is lifted to a *weak* limit in $\mathcal{K}\ell(T)$. It is not the case that Prop. 5.2 subsumes the construction in [4]: the former does not apply to the nonempty powerset monad (but our Prop. 4.2 does apply to it).

In [12], an explicit description of a (proper, not weakly) final $\overline{F}$-coalgebra is given for $F \in \{\, \Sigma \times (\_),\, 1 + \Sigma \times (\_) \,\}$ and $T \in \{\mathcal{G}, \mathcal{G}_{=1}\}$. Here $\mathcal{G}_{=1}$ is the *Giry monad* and restricts $\mathcal{G}$ to proper, not sub-, distributions. We do not use their results (proper finality) for characterization of infinite traces, because: 1) if $T = \mathcal{G}$ then the final coalgebras do not coincide with the set of possibly infinite words; and 2) if $T = \mathcal{G}_{=1}$ then language inclusion is reduced to the equality. We doubt about the value of developing simulation-based methods for the latter degenerate case, one reason being that trace inclusion is often a more difficult problem than trace equivalence. For example, finite trace inclusion for probabilistic systems is undecidable [3] while trace equivalence is decidable [13].

In [17], it is shown that: a limit of a ω^{op}-sequence consisting of standard Borel spaces and surjective measurable functions is preserved by a polynomial functor F (where constants are restricted to standard Borel spaces), and also by $\mathcal{G}$. It is also shown there that such a polynomial functor F preserves standard Borel spaces, and so does $\mathcal{G}$. These facts imply the existence of a final $\mathcal{G}F$-coalgebra in **Meas** for every polynomial functor F. Note however that this final $\mathcal{G}F$-coalgebra captures (probabilistic) bisimilarity, not trace semantics.

8 Conclusions and Future Work

We have shown that the technique forward and backward Kleisli simulations [8] and that of FPE [18] – techniques originally developed for witnessing *finite* trace inclusion – are also applicable to *infinite* trace semantics. We followed [11] (and also [4, 12]) to characterize infinite trace semantics in coalgebraic terms, on which we established properties of Kleisli simulations such as soundness. We developed our theory for two classes of instances: nondeterministic systems and probabilistic ones.

There are some directions for a future work. In [18], in addition to FPE, a transformation called *backward partial execution* (BPE) is introduced. Similarly to FPE, BPE can also aid forward and backward Kleisli simulation for finite trace in the sense that it satisfy soundness and adequacy. However, BPE is only defined for word automata (with T-branching) and not generally for (T, F)-systems. Defining BPE categorically and proving its soundness and adequacy with respect to infinite trace, possibly restricting to word automata, is one of the future work.

Another direction is implementation and experiments. As forward and backward Kleisli simulations in this paper are defined in almost the same way as [18], we can use the implementation already developed there to check infinite trace inclusion.

Acknowledgments. The authors are supported by Grants-in-Aid No. 24680001 & 15K11984, JSPS.

References

1 Jirí Adámek and Václav Koubek. Least fixed point of a functor. *J. Comput. Syst. Sci.*, 19(2):163–178, 1979.

2 Christel Baier and Joost-Pieter Katoen. *Principles of model checking.* MIT Press, 2008.

3 Vincent D. Blondel and Vincent Canterini. Undecidable problems for probabilistic automata of fixed dimension. *Theory Comput. Syst.*, 36(3):231–245, 2003.

4 Corina Cîrstea. Generic infinite traces and path-based coalgebraic temporal logics. *Electr. Notes Theor. Comput. Sci.*, 264(2):83–103, 2010.

5 P. Cousot and R. Cousot. Constructive versions of Tarski's fixed point theorems. *Pacific Journal of Mathematics*, 81(1):43–57, 1979.

6 J.L. Doob. *Measure Theory.* Graduate Texts in Mathematics. Springer New York, 1994.

7 Michele Giry. A categorical approach to probability theory. In *Proc. Categorical Aspects of Topology and Analysis*, volume 915 of *Lect. Notes Math.*, pages 68–85, 1982.

8 Ichiro Hasuo. Generic forward and backward simulations. In Christel Baier and Holger Hermanns, editors, *CONCUR*, volume 4137 of *Lect. Notes Comp. Sci.*, pages 406–420. Springer, 2006.

9 Ichiro Hasuo. Generic forward and backward simulations II: Probabilistic simulation. In Paul Gastin and François Laroussinie, editors, *CONCUR*, volume 6269 of *Lecture Notes in Computer Science*, pages 447–461. Springer, 2010.

10 Ichiro Hasuo, Bart Jacobs, and Ana Sokolova. Generic trace semantics via coinduction. *Logical Methods in Computer Science*, 3(4), 2007.

11 Bart Jacobs. Trace semantics for coalgebras. *Electr. Notes Theor. Comput. Sci.*, 106:167–184, 2004.

12 Henning Kerstan and Barbara König. Coalgebraic trace semantics for continuous probabilistic transition systems. *Logical Methods in Computer Science*, 9(4), 2013.

13 Stefan Kiefer, Andrzej S. Murawski, Joël Ouaknine, Björn Wachter, and James Worrell. Language equivalence for probabilistic automata. In Ganesh Gopalakrishnan and Shaz Qadeer, editors, *CAV*, volume 6806 of *Lecture Notes in Computer Science*, pages 526–540. Springer, 2011.

14 Nancy A. Lynch and Frits W. Vaandrager. Forward and backward simulations: I. Untimed systems. *Inf. Comput.*, 121(2):214–233, 1995.

15 Saunders Mac Lane. *Categories for the working mathematician*, volume 5. Springer verlag, 1998.

16 Philip S. Mulry. Lifting theorems for kleisli categories. In Stephen D. Brookes, Michael G. Main, Austin Melton, Michael W. Mislove, and David A. Schmidt, editors, *Mathematical Foundations of Programming Semantics, 9th International Conference, New Orleans, LA, USA, April 7–10, 1993, Proceedings*, volume 802 of *Lecture Notes in Computer Science*, pages 304–319. Springer, 1993.

17 Christoph Schubert. Terminal coalgebras for measure-polynomial functors. In Jianer Chen and S. Barry Cooper, editors, *Theory and Applications of Models of Computation, 6th Annual Conference, TAMC 2009, Changsha, China, May 18–22, 2009. Proceedings*, volume 5532 of *Lecture Notes in Computer Science*, pages 325–334. Springer, 2009.

18 Natsuki Urabe and Ichiro Hasuo. Generic forward and backward simulations III: quantitative simulations by matrices. In Paolo Baldan and Daniele Gorla, editors, *CONCUR 2014 – Concurrency Theory – 25th International Conference, CONCUR 2014, Rome, Italy, September 2–5, 2014. Proceedings*, volume 8704 of *Lecture Notes in Computer Science*, pages 451–466. Springer, 2014.

19 Natsuki Urabe and Ichiro Hasuo. Coalgebraic infinite traces and kleisli simulations. *CoRR*, abs/1505.06819, 2015.

Finitary Corecursion for the Infinitary Lambda Calculus*

Stefan Milius and Thorsten Wißmann

Lehrstuhl für Theoretische Informatik, FAU Erlangen-Nürnberg, Germany

Abstract

Kurz et al. have recently shown that infinite λ-trees with finitely many free variables modulo α-equivalence form a final coalgebra for a functor on the category of nominal sets. Here we investigate the rational fixpoint of that functor. We prove that it is formed by all rational λ-trees, i.e. those λ-trees which have only finitely many subtrees (up to isomorphism). This yields a corecursion principle that allows the definition of operations such as substitution on rational λ-trees.

1998 ACM Subject Classification F.3.2 Semantics of Programming Languages, F.4.1 Mathematical Logic, D.3.1 Formal Definitions and Theory

Keywords and phrases rational trees, infinitary lambda calculus, coinduction

Digital Object Identifier 10.4230/LIPIcs.CALCO.2015.336

1 Introduction

One of the most important concepts in computer science is the λ-calculus. It is a very simple notion of computation because its syntax consists only of three constructs: variables, λ-abstraction and function application, and its semantics consists of only two concepts α-conversion for renaming of bound variables and β-conversion for executing function applications. Yet it is very powerful since it is Turing complete and allows to define many notions of higher level programming languages such as booleans, if-then-else, natural numbers, arithmetic operations, lists including mapping and folding, recursion etc.[1]

However, whenever one wants to deal with inductive and coinductive definitions in the presence of variable binding subtle issues arise and one has to be careful not to mess up the variable binding. One solution to these problems has been proposed by Gabbay and Pitts [12]. They use *nominal sets* as a framework for dealing with binding operators, abstraction and structural induction. Nominal sets go back to Fraenkel's and Mostowski's permutation model for set theory devised in the 1920s and 1930s. They are sets equipped with an action of the group of finite permutations on a given fixed set $\mathcal{V}$ of atoms (here these play the role of variables). For an arbitrary nominal set one can then define the notions of "free" and "bound" variables using the notion of support (we recall this in Section 2.2). Gabbay and Pitts then consider the functor

$$L_\alpha X = \mathcal{V} + [\mathcal{V}]X + X \times X$$

expressing the type of the term constructors of the λ-calculus (note that the abstraction functor $[\mathcal{V}]X$ is a quotient of $\mathcal{V} \times X$ modulo renaming "bound" variables). And they prove that the initial algebra for L_α is formed by all λ-terms modulo α-equivalence.

[*] This work is supported by the Deutsche Forschungsgemeinschaft (DFG) under project MI 717/5-1.

[1] Depending on the application a third semantic concept, η conversion, may be of interest. But this is neither needed for Turing completeness nor for our work.

6th International Conference on Algebra and Coalgebra in Computer Science (CALCO'15).
Editors: Lawrence S. Moss and Pawel Sobocinski; pp. 336–351
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Recently, Kurz et al. [18] have characterized the final coalgebra for L_α (and more generally, for functors arising from so-called binding signatures): it is carried by the set of all infinitary λ-terms (i.e. finite or infinite λ-trees) with finitely many free variables modulo α-equivalence. This then allows to define operations on infinitary λ-terms by coinduction, for example substitution and operations that assign to an infinitary λ-term its normal form computations (e.g. the Böhm, Levy-Longo, and Berarducci trees of a given infinitary λ-term).

Our contribution in this paper is to give a characterization of the *rational fixpoint* of the functor L_α. In general, the rational fixpoint for a functor F lies between the initial algebra and the final coalgebra for F. If one thinks of it as a coalgebra, it is characterized as the final locally finitely presentable F-coalgebra. Intuitively, one may think of it as collecting all behaviours of "finite" (more technically, finitely presentable carried) F-coalgebras. Examples include regular languages, eventually periodic and rational streams, rational formal power-series etc. For a polynomial endofunctor F_Σ on sets associated to the signature Σ, the rational fixpoint consists of regular Σ-trees of Elgot [10], i.e. those (finite and infinite) Σ-trees having only finitely many different subtrees (up to isomorphism). We will prove in Section 3 that the rational fixpoint for L_α on Nom is carried by all rational λ-trees modulo α-equivalence. Before that we recall in Section 2 preliminaries on the infinitary λ-calculus, nominal sets and the rational fixpoint. The finality principle of the rational fixpoint may be understood as a finitary corecursion principle. In Section 4 we show applications of our main result, in particular, that the coinductive definition of substitution given in [18] restricts to rational trees. We also discuss coinductive definitions concerning normal form computations. We conclude in Section 5.

Related work. The work presented here is based on the second author's student project reported in [28].

A related approach to variable binding operations which uses presheaves over finite sets was proposed by Fiore, Plotkin and Turi [11]. By now this has developed into a respectable body of work by these and other authors. Most related to our work here is the coinductive approach to infinitary and rational λ-terms studied by Adámek, Milius and Velebil [3]. This work considers an endofunctor very similar to L_α but on the category of presheafes on finite sets. Its final coalgebra is shown to be the presheaf of all infinite λ-trees and the rational fixpoint the presheaf of all rational trees – each of them modulo α-equivalence.

Omitted proofs and details may be found in the full version [22] of our paper.

2 Preliminaries

We assume that readers are familiar with basic notions of category theory and with algebras and coalgebras for an endofunctor. For a given endofunctor F on the category $\mathcal{C}$ we will write $t : \nu F \to F(\nu F)$ for the final coalgebra (assuming that it exists). Given an F-coalgebra (C, c) we write $c^\dagger : (C, c) \to (\nu F, t)$ for the unique F-coalgebra homomorphism from C to νF. The category of coalgebras for an endofunctor F is denoted by $\mathsf{Coalg}\, F$. For introductory texts on coalgebras see [26, 16, 1].

We will now give some background on the (infinitary) λ-calculus, on nominal sets and on the rational fixpoint of a functor as needed in the present paper.

2.1 Infinitary λ-Calculus and Rational Trees

Before we talk about infinitary λ-terms (aka λ-trees) first recall that ordinary λ-terms are defined starting from a fixed countable set of variables $\mathcal{V}$ by the grammar

$$T ::= x \mid \lambda x.T \mid TT,$$

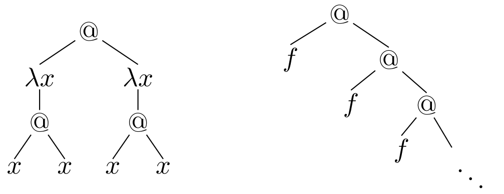
where x ranges over $\mathcal{V}$. We denote the set of all λ-terms by Λ. Free and bound variables and substitution are defined as usual with the operator $\lambda x.(-)$ binding x in its argument. Often one considers λ-terms modulo *α-equivalence*, i.e., the least equivalence relation on λ-terms identifying two terms that arise by consistently renaming bound variables. One can think of a term $\lambda x.T$ as representing a computation that takes a parameter P that is used in all free occurences of x in T. Hence, the main computation rule of the λ-calculus is β-reduction, i.e. the rule

$$(\lambda x.T)P \to_\beta T[x \mapsto P].$$

For example we have $(\lambda x.\lambda y.x)\, a\, b \to_\beta (\lambda y.a)\, b \to_\beta a$, where a cannot be reduced further. However, terms may have infinite reduction sequences; a prominent example is Yf for the Y-combinator defined as $Y := \lambda g.(\lambda x.g(x\,x))\,(\lambda x.g(x\,x))$ we have:

$$\begin{aligned} Yf &= (\lambda g.(\lambda x.g(x\,x))\,(\lambda x.g(x\,x)))f \\ &\to_\beta (\lambda x.f(x\,x))\,(\lambda x.f(x\,x)) \to_\beta f((\lambda x.f(x\,x))\,(\lambda x.f(x\,x))) \\ &\to_\beta f(f((\lambda x.f(x\,x))\,(\lambda x.f(x\,x)))) \to_\beta \cdots \end{aligned}$$

Informally speaking, this "converges" to the infinite term $f(f(f(\cdots)))$. If one takes such infinite terms as legal objects of the λ-calculus one is led to *infinitary* λ-calculus. There one replaces λ-terms by (finite and infinite) λ-trees. A *λ-tree* is a rooted and ordered tree with leaves labelled by variables in $\mathcal{V}$ and with two sorts of inner nodes: nodes with one successor labelled by λx for some variable $x \in \mathcal{V}$ and nodes with two successors labelled by @. For example, we have the λ-trees



(1)

representing the λ-term $(\lambda x.xx)(\lambda x.xx)$ and the infinite term $f(f(f(\cdots)))$, respectively. Let Λ^∞ be the set of all λ-trees. The notions of free and bound variables of a λ-tree are clear: a variable x is bound in a λ-tree t if there is a path from a leave labelled by x to the root of t that contains a node labelled by λx, and x is free in t if there is a path from an x-labelled leaf to the root of t that does not contain any node labelled by λx.

The classic approach to defining operations such a substitution on λ-trees uses that Λ^∞ is the metric completion of Λ under a natural metric; this idea of using a metric approach to dealing with infinite trees goes at least back to Arnold and Nivat [5]. Thus, every infinite λ-tree is regarded as the limit of the Cauchy sequence of its truncations at level n. Notions such as α-equivalence and substitution of λ-trees are then defined by extending the corresponding notions on finite λ-trees (i.e. λ-terms) continuously. More concretely, two λ-trees s and t are α-equivalent iff for every natural number n the pair of truncations at level n of s and t are α-equivalent λ-terms (see [18, Definition 5.17]).

Our aim in this paper is to give a coalgebraic characterization of an important subclass of all λ-trees, the so called *rational* λ-trees. The following definition follows Ginali's characterization [15] of regular Σ-trees for a signature Σ:

▶ **Definition 2.1.** A λ-tree having only finitely many subtrees (up to isomorphism) is called *rational.* A λ-tree modulo α-equivalence, i.e. an α-equivalence class of λ-trees, is called *rational* if it contains at least one rational λ-tree.

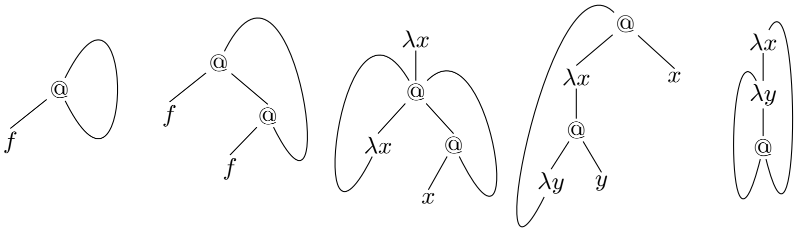


Figure 1 Finite representations of rational λ-trees.

Intuitively, the rational λ-trees are those λ-trees that admit a finite representation as a λ-tree with "uplinks". All finite λ-trees are, of course, rational, and so is the right-hand λ-tree in (1). Other examples are in Figure 1.

The uplink from some node s to some other node r indicates that the entire tree starting at r occurs as a subtree of s. In other words, such a λ-tree with uplinks represents its tree unravelling, i.e. the first and second tree on the left both represent the rational infinite λ-tree shown in (1) on the right.

Things get more complicated, if abstractions come into play, as in the third tree. Here the x clearly refers to the λx in the root, but some of the "copies" of x are bound by the λx in the left branch and other copies are bound to the abstraction in the root. Something similar can be observed in the last but one tree, which has two free variables x, y, but all "copies" of x and y are bound by the previous copy of λx and λy respectively. Finally, the rightmost tree represents a λ-tree that consists of applications and abstractions only:

$$\lambda xy.(\lambda y \lambda y \ldots)(\lambda xy.(\lambda y \lambda y \ldots)(\lambda xy.(\lambda y \lambda y \ldots) \ldots)).$$

2.2 Nominal Sets

It was the idea of Gabbay and Pitts [12] to use nominal sets as a category-theoretic framework in which to describe λ-terms modulo α-equivalence as the initial algebra for a functor L_α. One can then use its universal property to define operations such as substitution of λ-terms. And Kurz et al. [18] characterized the final coalgebra for L_α; it is carried by the set of λ-trees with finitely many free variables modulo α-equivalence. Again, the universal property allows one to define operations such as substitution – this time by corecursion. We will now recall some background material on nominal sets and the main result of [18].

We fix a countable set $\mathcal{V}$ of variable names. Let $\mathfrak{S}(\mathcal{V})$ be the group of *finite permutations* of $\mathcal{V}$, where a permutation $\pi \in \mathfrak{S}(\mathcal{V})$ is called finite iff $\{v \in \mathcal{V} \mid \pi(v) \neq v\}$ is a finite set. Now consider a set X together with a group action $\cdot : \mathfrak{S}(\mathcal{V}) \times X \to X$. Intuitively, one should think of X as a set of terms, and for a finite permutation of variable names π and some term x, $\pi \cdot x$ denotes the new term obtained after renaming the variables in x according to π. In order to talk about variables "occurring" in $x \in X$ we can check which variable renamings fix the term x. This is captured by the notion of *support*: a set $S \subseteq \mathcal{V}$ *supports* $x \in X$ if for all $\pi \in \mathfrak{S}(\mathcal{V})$ with $\pi(v) = v$ for all $v \in S$ we have $\pi \cdot x = x$. Some $x \in X$ is *finitely supported* if there is a finite $S \subseteq \mathcal{V}$ supporting x.

A *nominal set* is a set X together with a $\mathfrak{S}(\mathcal{V})$-action such that all elements of X are finitely supported.

▶ **Example 2.2.**

1. The set $\mathcal{V}$ of variable names with the group action given by $\pi \cdot v = \pi(v)$ is a nominal set; for each $v_i \in \mathcal{V}$ the singleton $\{v_i\}$ supports v_i.

2. Every ordinary set X can be made a nominal set by equipping it with the trivial action $\pi \cdot x = x$ for all $x \in X$ and $\pi \in \mathfrak{S}(\mathcal{V})$. So each $x \in X$ can be thought of a term not containing any variable, i.e. the empty set supports x.
3. The set Λ of all λ-terms forms a nominal set with the group action given by renaming of free variables. Every λ-term is supported by the set of its free variables. In contrast the set Λ^∞ of all λ-trees is not nominal since λ-trees with infinitely many free variables do not have finite support. However, the set $\Lambda^\infty_{\mathsf{ffv}}$ of all λ-trees with finitely many free variables is nominal.

Notice that if $S \subseteq V$ supports $x \in X$, then $S' \supseteq S$ also supports $x \in X$. So S supporting x only means that by not touching the members of S one does not modify the term x. But it is more interesting to talk about the variables actually occurring in x. This is achieved by considering the smallest set supporting x, which is denoted by $\mathsf{supp}(x)$. If $v \in \mathcal{V} \setminus \mathsf{supp}(x)$, we say that v is *fresh for* x, denoted by $v \mathrel{\#} x$.

▶ **Example 2.3.** The set $\mathcal{P}_f(\mathcal{V})$ of finite subsets of $\mathcal{V}$, together with the point-wise action is a nominal set. The support of each $u \in \mathcal{P}_f(\mathcal{V})$ is u itself: $\pi \cdot u = \{\pi \cdot x \mid x \in u\}$ and $\mathsf{supp}(u) = u$. Note that $\mathcal{P}(\mathcal{V})$ with the point-wise action is not a nominal set because the infinite $\{v_0, v_2, v_4, \ldots\}$ does not have any finite support.

The morphisms of nominal sets are those maps which are equivariant: an *equivariant map* $f : (X, \cdot) \to (Y, \star)$ is a map $f : X \to Y$ with $f(\pi \cdot x) = \pi \star f(x)$ for all $\pi \in \mathfrak{S}(\mathcal{V}), x \in X$.

For example, the function $\mathsf{supp} : X \to \mathcal{P}_f(\mathcal{V})$ mapping each element to its (finite) support is an equivariant map.

▶ **Remark 2.4.** For any equivariant $f : (X, \cdot) \to (Y, \star)$, we have $\mathsf{supp}(f(x)) \subseteq \mathsf{supp}(x)$ for any $x \in X$.

The nominal sets – together with the equivariants as morphisms – form a category, denoted by Nom. As shown in [14], this category is (equivalent to) a Grothendieck topos (the so-called Shanuel topos), and so it has rich categorical structure. We only mention some facts needed for the current paper.

Monomorphisms and epimorphisms in Nom are precisely the injective and surjective equivariant maps, respectively. It is not difficult to see that every epimorphism in Nom is strong, i.e., it has the unique diagonalization property w.r.t. any monomorphism: given an epimorphism $e : A \twoheadrightarrow B$, a monomorphism $m : C \hookrightarrow D$ and $f : A \to C$, $g : B \to D$ with $g \cdot e = m \cdot f$, there exists a unique diagonal $d : B \to C$ with $d \cdot e = f$ and $m \cdot d = g$.

Furthermore, Nom has image-factorizations; that means that every equivariant map $f : A \to C$ factorizes as $f = m \cdot e$ for an epimorphism $e : A \twoheadrightarrow B$ and a monomorphism $m : B \hookrightarrow C$. Note that the intermediate object B is (isomorphic to) the image $f[A]$ in B with the restricted action. For an endofunctor F on Nom preserving monos this factorization systems lifts to $\mathsf{Coalg}\, F$: every F-coalgebra homomorphism f has a factorization $f = m \cdot e$ where e and m are F-coalgebra homomorphisms that are epimorphic and monomorphic in Nom, respectively.

Recall from [24, Section 2.2] that Nom is complete and cocomplete with colimits and finite limits formed as in Set. In fact, Nom is a locally finitely presentable category in the sense of Gabriel and Ulmer [13] (see also Adámek and Rosický [4]). We shall not recall that notion here as it is not needed in the current paper; intuitively, a locally finitely presentable category is a category with a well behaved “finite” objects (called *finitely presentable* objects) such that every object can be build (as a filtered colimit) from these. Petrişan [23, Proposition 2.3.7] has shown that the finitely presentable objects of Nom are precisely the orbit-finite nominal sets.

▶ **Definition 2.5.** For a nominal set $(X, \cdot)$ and $x \in X$ the set $\{\pi \cdot x \mid \pi \in \mathfrak{S}(\mathcal{V})\}$ is called the *orbit* of x. A nominal set $(X, \cdot)$ is said to be *orbit-finite* if it has only finitely many orbits.

The notion of orbit-finiteness plays a central role in our paper since the rational λ-trees modulo α-equivalence are described by precisely all the coalgebras with an orbit-finite carrier for the functor L_α further below (cf. Proposition 2.11 and Theorem 3.4).

We now collect a few easy properties of orbit-finite sets that we are going to need.

▶ **Lemma 2.6.** *For any $x_1, x_2 \in X$ in the same orbit, we have $|\,\mathsf{supp}(x_1)| = |\,\mathsf{supp}(x_2)|$.*

▶ **Lemma 2.7.** *For an element x of a nominal set X, there are at most $|\,\mathsf{supp}(x)|!$ many elements with support $\mathsf{supp}(x)$ in the orbit of x.*

▶ **Lemma 2.8.** *For a finite set $W \subseteq \mathcal{V}$ and an orbit $\mathcal{O}$ of the nominal set X there are only finitely many elements in $\mathcal{O}$ whose support is contained in W.*

Let us now recall from Kurz et al. [18] how all λ-trees form a final coalgebra in Nom. First consider the following endofunctor on Nom:

$$LX = \mathcal{V} + \mathcal{V} \times X + X \times X;$$

its coproduct components describe the type of the term constructors of the λ-calculus (variables, λ-abstraction and application, respectively). As shown in [18], the final coalgebra for this functor is carried by the set of all λ-trees containing finitely many (free and bound) variables.[2] Its coalgebra structure is the obvious map decomposing a λ-tree at the root: a single node λ-tree is mapped to its node label in $\mathcal{V}$, a λ-tree whose root is labeled by λx to (x, t), where t is the λ-tree defined by the successor of the root and a λ-tree with root label @ to the pair of λ-trees defined by the successors of the root.

Since this final coalgebra completely disregards α-equivalence it is not possible to define substitution as a total operation on it. The solution is to replace the second component $\mathcal{V} \times X$ of L by Gabbay and Pitts abstraction functor [12, Lemma 5.1] that takes α-equivalence into account:

▶ **Definition 2.9.** Let $(X, \cdot)$ be a nominal set. We define α-equivalence $\sim_\alpha$ as the relation on $\mathcal{V} \times X$ as

$$(v_1, x_1) \sim_\alpha (v_2, x_2) \text{ if there exists } z \# \{v_1, v_2\}, z \# x_1, z \# x_2 \text{ with } (v_1\ z)x_1 = (v_2\ z)x_2.$$

The $\sim_\alpha$-equivalence class of (v, x) is denoted by $\langle v \rangle x$. The abstraction $[\mathcal{V}]X$ of the nominal X is the quotient $(\mathcal{V} \times X)/{\sim_\alpha}$ with the group action defined by

$$\pi \cdot \langle v \rangle x = \langle \pi(v) \rangle (\pi \cdot x).$$

For an equivariant map $f : X \to Y$, $[\mathcal{V}]f : [\mathcal{V}]X \to [\mathcal{V}]Y$ is defined by $\langle v \rangle x \mapsto \langle v \rangle (f(x))$.

Note that the abstraction functor $[\mathcal{V}](-)$ is *strong*, i.e., we have a natural transformation τ with components $\tau_{X,Y} : [\mathcal{V}]X \times Y \to [\mathcal{V}](X \times Y)$ given by $\tau_{X,Y}(\langle v \rangle x, y) = \langle v \rangle (x, y)$; we will need the strength τ in Section 4.1. Now one considers the endofunctor L_α on Nom given by

$$L_\alpha X = \mathcal{V} + [\mathcal{V}]X + X \times X.$$

[2] Note that this is different from the set $\Lambda^\infty_{\mathsf{ffv}}$ mentioned in Example 2.10.3; λ-trees in the latter may have infinitely many bound variables.

Gabbay and Pitts [12] showed that its initial algebra consists of all λ-terms modulo α-equivalence, and the main result of Kurz et al. [18] is that the final coalgebra νL_α is carried by the set $\Lambda^\infty_{\mathsf{ffv}}$ of all λ-trees with finitely many free variables quotiented by α-equivalence. The coalgebra structure is the same as on the final coalgebra for L – one can show that this is well-defined on equivalence classes modulo α-equivalence.

2.3 The Rational Fixpoint

Recall that by Lambek's Lemma [19], the structure of an initial algebra and a final coalgebra for a functor F are isomorphisms, so both yield *fixpoints* of F. Here we shall be interested in a third fixpoint that lies in between initial algebra and final coalgebra called the *rational fixpoint* of F. This can on the one hand be characterized as the initial iterative algebra for F (see [2]) or as the final locally finitely presentable coalgebra for F (see [20]). We will only recall the latter description since the former will not be needed in this paper.

The rational fixpoint can be defined for any *finitary* endofunctor F on a locally finitely presentable (lfp, for short) category $\mathcal{C}$, i.e. F is an endofunctor on $\mathcal{C}$ that preserves filtered colimits. Examples of lfp categories are Set, the categories of posets and of graphs, every finitary variety of algebras (such as groups, rings, and vector spaces) and every Grothendieck topos (such as Nom). The finitely presentable objects in these categories are: all finite sets, posets or graphs, those algebras presented by finitely many generators and relations, and, as we mentioned before, the orbit-finite nominal sets.

Now let $F : \mathcal{C} \to \mathcal{C}$ be finitary on the locally finitely presentable category $\mathcal{C}$ and consider the full subcategory $\mathsf{Coalg}_f\, F$ of $\mathsf{Coalg}\, F$ given by all F-coalgebras with a finitely presentable carrier. In [20] the *locally finitely presentable* F-coalgebras were characterized as precisely those coalgebras that arise as a colimit of a filtered diagram of coalgebras from $\mathsf{Coalg}_f\, F$. It follows that the *final* locally finitely presentable coalgebra can be constructed as the colimit of *all* coalgebras from $\mathsf{Coalg}_f\, F$. More precisely, one defines a coalgebra $r : \varrho F \to F(\varrho F)$ as the colimit of the inclusion functor of $\mathsf{Coalg}_f\, F$: $(\varrho F, r) := \operatorname{colim}(\mathsf{Coalg}_f\, F \hookrightarrow \mathsf{Coalg}\, F)$. Note that since the forgetful functor $\mathsf{Coalg}\, F \to \mathcal{C}$ creates all colimits this colimit is actually formed on the level of $\mathcal{C}$. The colimit ϱF then carries a uniquely determined coalgebra structure r making it the colimit above.

As shown in [2], ϱF is a fixpoint for F, i.e. its coalgebra structure r is an isomorphism. From [20] we obtain that local finite presentability of a coalgebra (C, c) has the following concrete characterizations: (1) for $\mathcal{C} = \mathsf{Set}$ local finiteness, i.e. every element of C is contained in a finite subcoalgebra of C; (2) for $\mathcal{C} = \mathsf{Nom}$, local orbit-finiteness, i.e. every element of C is contained in an orbit-finite subcoalgebra of C; (3) for $\mathcal{C}$ the category of vector spaces over a field K, local finite dimensionality, i.e., every element of C is contained in a subcoalgebra of C carried by a finite dimensional subspace of C.

▶ **Example 2.10.** We list only a few examples of rational fixpoints; for more see [2, 20, 8].

1. Consider the functor $FX = 2 \times X^A$ on Set where A is an input alphabet and $2 = \{0, 1\}$. The F-coalgebras are precisely the deterministic automata over A (without initial states). The final coalgebra is carried by the set $\mathcal{P}(A^*)$ of all formal languages and the rational fixpoint is its subcoalgebra of regular languages over A.
2. For $FX = \mathbb{R} \times X$ on Set the final coalgebra is carried by the set $\mathbb{R}^\omega$ of all real streams and the rational fixpoint is its subcoalgebra of all eventually periodic streams, i.e. streams $uvvv\cdots$ with $u, v \in \mathbb{R}^*$. Taking the same functor on the category of real vector spaces we get the same final coalgebra $\mathbb{R}^\omega$ with the componentwise vector space structure, but this time the rational fixpoint is formed by all rational streams (see [27, 20]).

3. Let Σ be a signature of operation symbols with prescribed arity, i.e. a sequence $(\Sigma_n)_{n<\omega}$ of sets. This give rise to an associated polynomial endofunctor F_Σ on Set given by $F_\Sigma X = \coprod_{n<\omega} \Sigma_n \times X^n$. Its initial algebra is formed by all Σ-terms and its final coalgebra by all (finite and infinite) Σ-trees, i.e. rooted and ordered trees such that every node with n children is labelled by an n-ary operation symbol. And the rational fixpoint consists precisely of all regular Σ-trees of Elgot [10] (see also Courcelle [9]), i.e. those Σ-trees having only finitely many different subtrees up to isomorphism (see Ginali [15]).

Note that in all the above examples the rational fixpoint ϱF allways occurs as a subcoalgebra of the final coalgebra νF. But this need not be the case in general (see [8, Example 3.15] for a counterexample). However, we have the following result:

▶ **Proposition 2.11** ([8, Proposition 3.12])**.** *Suppose that in $\mathcal{C}$ finitely presentable objects are closed under strong quotients and that F is finitary and preserves monomorphisms. Then the rational fixpoint ϱF is the subcoalgebra of νF given by the union of images of all coalgebra homomorphisms $c^\dagger : (C, c) \to (\nu F, t)$ where (C, c) ranges over $\mathsf{Coalg}_f\, F$.*[3]

In particular, for a finitary functor F on Set (or Nom resp.) preserving monomorphisms the rational fixpoint is the union in νF of images of all finite (or orbit-finite resp.) coalgebras; in symbols:

$$\varrho F = \bigcup_{(C,c) \text{ in } \mathsf{Coalg}_f\, F} c^\dagger[C] \;\subseteq\; \nu F.$$

Note that it is sufficient to let (C, c) range over those coalgebras in $\mathsf{Coalg}_f\, F$ where $c^\dagger$ is injective (or an inclusion map) because for an arbitrary (orbit-)finite (C, c) in $\mathsf{Coalg}_f F$ its image $c^\dagger[C]$ is an (orbit-)finite F-coalgebra, too.

3 The Rational Fixpoint in Nominal Sets

In this section we are going to prove the main result of our paper, a characterization of the rational fixpoint for the functor L_α as the rational λ-trees modulo α-equivalence.

But we start with the rational fixpoint of the functor $LX = \mathcal{V} + \mathcal{V} \times X + X \times X$. Note that both functors L and L_α are finitary and preserve monomorphisms (to see this use [18, Proposition 5.6] and the fact the forgetful functor from Nom to Set creates colimits).

▶ **Proposition 3.1.** *The rational fixpoint of the functor $L : \mathsf{Nom} \to \mathsf{Nom}$ is formed by all rational λ-trees.*

In the proof of the following theorem we will slightly abuse notation and consider $LX = \mathcal{V} + \mathcal{V} \times X + X \times X$ as an endofunctor on Set. Note that its final coalgebra is formed by the set Λ^∞ of all λ-trees and its rational fixpoint by all rational λ-trees (this follows from Example 2.10.3).

▶ **Theorem 3.2.** *Let $X \xrightarrow{a} L_\alpha X$ be an orbit-finite coalgebra. Then for all* $\mathsf{root} \in X$*, $a^\dagger(\mathsf{root}) \in \varrho L_\alpha$ is a rational λ-tree.*

[3] In a general lfp category the *image* of $c^\dagger$ is obtained by taking a strong epi-mono factorization of $c^\dagger$, and the *union* is then obtained as a directed colimit of the resulting subobjects of $(\nu F, t)$.

Proof sketch. Let $m := \max_{x\in X} \big|\mathsf{supp}(x)\big|$ be the maximal number of free variables in any element of X. This exists by Theorem 2.6 since X is orbit-finite. Let $W \subseteq \mathcal{V}$ be some set of $m+1$ variables containing $\mathsf{supp}(\mathsf{root})$. Hence for all $x \in X$ there exists a $w \in W$ with $w \# x$.

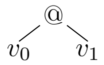
In the following, one constructs a rational λ-tree in the α-equivalence class of $a^\dagger(\mathsf{root})$. First, define an L-coalgebra $C \xrightarrow{c} LC = \mathcal{V} + \mathcal{V} \times C + C \times C$ in Set with $C := \{x \in X \mid \mathsf{supp}(x) \subseteq W\}$ and

$$c(x) = \begin{cases} w & \text{if } a(x) = w \in W \subseteq \mathcal{V} \\ (\ell, r) & \text{if } a(x) = (\ell, r) \in X \times X \\ (w, y) & \text{if } a(x) = \langle v \rangle y' \text{ and } y = (v\ w)y' \text{ for some } w \in W \setminus \mathsf{supp}(x) \end{cases}$$

Next one readily verifies that c is well-defined, i.e., its image really lies in LC.

Furthermore, C is finite because X is orbit-finite and within any orbit there are only finitely many elements with a support contained in W by Theorem 2.8. Let $c^\dagger$ denote the unique L-coalgebra homomorphism into the final L-coalgebra in Set. Since C is finite, we know that $c^\dagger : C \to \nu L$ factors through the rational fixpoint, i.e. for every $x \in C$, $c^\dagger(x)$ is a rational λ-tree. One then proves that $[c^\dagger(x)]_\alpha = a^\dagger(x)$ for all $x \in C \subseteq X$, where $[-]_\alpha$ denotes α-equivalence classes. This involves a non-trivial induction argument using the *final chains* of the set functor L and the functor L_α on Nom as well as technical details from [18]. It follows that $a^\dagger(\mathsf{root})$ is rational. ◀

For the L-coalgebra (C, c) in Set from the previous proof, we know that for any $x \in C$, the resulting tree $c^\dagger(x)$ has at most $|C|$ subtrees. This does not hold for an L_α-coalgebra (X, a) in Nom: if X has a non-trivial action, then the cardinality of X is at least infinite, i.e. the cardinality does not give a reasonable bound for the number of subtrees. And the number of orbits n is not a bound either. The problem is that multiple elements from the same orbit may represent different subtrees. For example, consider the rational tree

$t :=$  ,

and let $X := \mathcal{V} + \{(\ell, r) \in \mathcal{V} \times \mathcal{V} \mid \ell \neq r\}$ be equipped with the coalgebra structure

$$X \xrightarrow{a} L_\alpha X, \quad a(x) = \begin{cases} v & \text{if } x = v \in \mathcal{V} \\ (\ell, r) & \text{if } x = (\ell, r) \in \mathcal{V} \times \mathcal{V}. \end{cases}$$

X is constructed to have two orbits: one consisting of single variables and one consisting of unequal ordered pairs of variables. We have $(v_0, v_1) \in X$ and $a^\dagger\big((v_0, v_1)\big) = [t]_\alpha$. But t has three subtrees, namely t itself, v_0, and v_1. This example is expanded later in Example 3.5.

But when looking closer at the construction of C in the previous proof, we can give a bound on the number of its elements, i.e. the number of (up to isomorphism) different subtrees of the rational tree $c^\dagger(\mathsf{root})$.

▶ **Proposition 3.3.** *Let (X, a) be an orbit-finite L_α-coalgebra with n orbits and let $m = \max_{x\in X} \big|\mathsf{supp}(x)\big|$. Then the number of elements of the coalgebra C (as constructed in the previous proof) is bounded by $n \cdot (m+1)!$.*

Proof. Recall from the proof of Theorem 3.2 that $C := \{x \in X \mid \mathsf{supp}(x) \subseteq W\}$, where W is a set of $m+1$ variables. Consider a fixed orbit O whose elements have a support of cardinality k: there are at most $\binom{m+1}{k}$ possibilities of choosing a k-element subset S of

W and for any such S there are at most $k!$ elements in O with support S, by Theorem 2.7. Thus, the number of elements of O in C is at most

$$\binom{m+1}{k} \cdot k! = \frac{(m+1)!}{k! \cdot (m+1-k)!} \cdot k! = \frac{(m+1)!}{(m+1-k)!} \overset{k \le m}{\le} \frac{(m+1)!}{(m+1-m)!} = (m+1)!.$$

In total, the cardinality of C is bounded by $n \cdot (m+1)!$. ◀

That the number of orbits n occurs linearly is not surprising, because if we have some finite carrier set X with the trivial action that "uses" all its elements for the coalgebra structure, we have exactly one subtree per element of X. In Example 3.5 we shall see that the factor is $(m+1)!$ is necessary. But before that we state and prove our main result:

▶ **Theorem 3.4.** *The rational fixed point ϱL_α contains precisely the rational λ-trees modulo α-equivalence.*

Proof. After Theorem 3.2 it only remains to show that all α-equivalence classes of rational λ-trees are in ϱL_α. Let $u_\alpha \in \nu L_\alpha$ be rational, witnessed by some rational representative $u \in u_\alpha$ with only finitely many subtrees (up to isomorphism). Let C be the finite set of all subtrees of u and define the nominal set X as

$$X := \bigcup_{s \in C} O([s]_\alpha) \ \subseteq \ \nu L_\alpha,$$

where $[s]_\alpha \in \nu L_\alpha$ is the α-equivalence class of the subtree s and $O(y) \subseteq \nu L_\alpha$ denotes the orbit of a given element $y \in \nu L_\alpha$. Note that the group action of νL_α restricts to X since it is a union of orbits.

Next we define a coalgebra structure $a : X \to L_\alpha X$ by restriction of the structure $t : \nu L_\alpha \to L_\alpha(\nu L_\alpha)$: set $a(x) := t(x)$ for all $x \in X$. We need to check that this is well-defined, i.e. that $t(x)$ really lies in $L_\alpha X$ for every $x \in X$. For this we consider three cases:

1. The case $t(x) \in \mathcal{V}$ is clear;
2. Suppose that $t(x) = \langle v \rangle y \in [\mathcal{V}](\nu L_\alpha)$ where $x = \pi \cdot [\lambda w.s]_\alpha$ for some $\pi \in \mathfrak{S}(\mathcal{V})$ and some subtree $\lambda w.s$ of u. Then we have

   $$\langle w \rangle [s]_\alpha = t([\lambda w.s]_\alpha) = t(\pi^{-1} \cdot x) = \pi^{-1} \cdot t(x) = \pi^{-1} \cdot \langle v \rangle y = \big\langle \pi^{-1}(v) \big\rangle (\pi^{-1} \cdot y).$$

   By the definition of abstraction, we have some $z \in \mathcal{V}$ with

   $$(w\ z) \cdot [s]_\alpha = (\pi^{-1}(v)\ z) \cdot \pi^{-1} \cdot y.$$

   Hence, y is in the orbit of $[s]_\alpha$ and therefore $t(x)$ is in $[\mathcal{V}]X$.
3. For $t(x) = (\ell, r) \in \nu L_\alpha \times \nu L_\alpha$, let $x \in X$ be $\pi \cdot [(s_\ell, s_r)]_\alpha$ for some $\pi \in \mathfrak{S}(\mathcal{V})$ and subtrees s_ℓ, s_r of u. Analogously to the previous case, we have

   $$(\ell, r) = t(x) = t(\pi \cdot [(s_\ell, s_r)]_\alpha) = \pi \cdot ([s_\ell]_\alpha, [s_r]_\alpha) = (\pi \cdot [s_\ell]_\alpha, \pi \cdot [s_r]_\alpha) \in X \times X.$$

By construction, $u_\alpha \in X$ and $a^\dagger(x) = x$ holds for all $x \in X$. By the finiteness of C, X is orbit-finite and thus $a^\dagger[X] \subseteq \varrho L_\alpha$, and in particular $u_\alpha \in \varrho L_\alpha$. ◀

The following example shows that the bound in Theorem 3.3 on the number of elements of the L-coalgebra C from the proof of Theorem 3.2 can essentially not be improved even if we omit the usage of λ-abstraction.

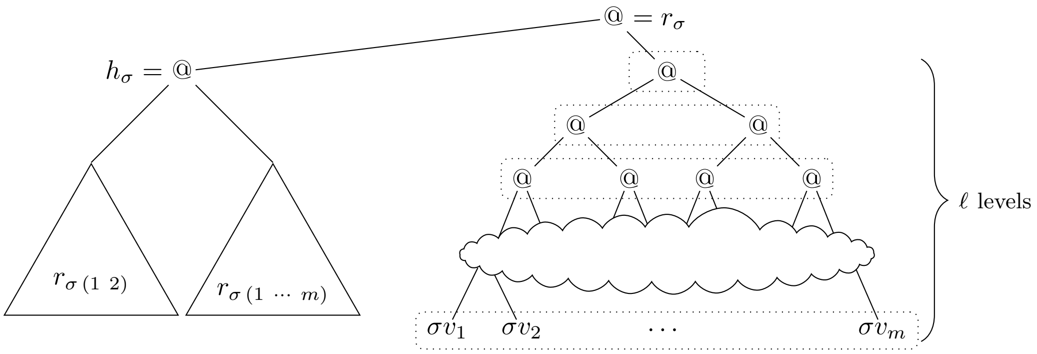


Figure 2

▶ **Example 3.5.** Let $\ell \geq 1$ be a fixed natural number and let $m = 2^{\ell-1}$. Further let $V = \{v_1, \ldots, v_m\}$ be a set of m variables and consider the rational λ-trees parametrized by permutations $\sigma \in \mathfrak{S}(\mathcal{V})$ shown in Figure 2.

With r_σ and h_σ we denote the corresponding subtrees rooted at the indicated nodes. Note that the α-equivalence classes in ϱL_α of each r_σ and of any of its subtrees are singletons since r_σ does not contain any λ-abstraction. For this reason we shall henceforth abuse notation and denote those equivalence classes by their representatives. Observe further that the group action on ϱL_α satisfies $\tau \cdot r_\sigma = r_{\tau\sigma}$ for any $\tau, \sigma \in \mathfrak{S}(\mathcal{V})$. This implies that all r_σ and h_σ, respectively, lie in the same orbit. Similarly, one can see that all nodes on the same level in the right-hand maximal subtrees of every r_σ (indicated by the dotted rectangles) lie in the same orbit.

Now consider r_{id} and the corresponding orbit-finite subcoalgebra X of ϱL_α from the proof of Theorem 3.4. The elements of X are all subtrees of r_{id} with the group action inherited from ϱL_α. By the above reasoning we see that X has precisely $\ell + 2$ orbits. Hence, the number of orbits of X is logarithmic in m. But the number of subtrees of r_{id} grows faster than $m!$. To see this, notice first that the permutations $(1\ 2)$ and $(1\ 2\ \cdots\ m)$ generate the group of all permutations of m elements. Thus, we see that r_{id} has all r_σ as subtrees where σ is any permutation that fixes $\mathcal{V} \setminus V$. For different σ and τ we have that r_σ, h_σ, r_τ and h_τ are pairwise non-isomorphic. Thus, we see that r_{id} has at least $2 \cdot m!$ subtrees. In addition, consider the i-th level (from 1 at the bottom to ℓ at the top) on the right in Figure 2. Each node on that level covers $k = 2^{i-1}$ variables. Then for each permutation of those k variables there exists a subtree of the right-hand successor of some subtree r_σ of r_{id} which is a complete binary tree of height i with the front given by the permutation. Thus, the total number of subtrees of r_{id} is precisely

$$2 \cdot m! + \sum_{i=1}^{\ell} \left(\binom{m}{2^{i-1}} \cdot (2^{i-1})! \right) = 2 \cdot m! + \sum_{i=1}^{\ell} \frac{m!}{(m - 2^{i-1})!}.$$

4 Application: Corecursive Definitions on Rational λ-trees

Our result in Theorem 3.4 that rational λ-trees modulo α-equivalence form the final locally orbit-finite L_α-coalgebra yields a corecursion principle. In this section we shall demonstrate this principle by considering two easy applications. First we show that substitution as defined corecursively for all λ-trees in [18] restricts to rational λ-trees. Secondly, we discuss the corecursive definition of the computation of the Böhm tree of a given rational λ-tree.

4.1 Substitution on Rational λ-trees

When performing operations known from (infinitary) λ-calculus on rational λ-trees, it is not clear whether the resulting λ-tree still is rational in general. One such operation is the substitution function $\mathsf{subs} : \nu L_\alpha \times \mathcal{V} \times \nu L_\alpha \longrightarrow \nu L_\alpha$, which for a given triple (t, v, s) replaces each occurence of the variable v in t by s. Kurz et al. [18] show how to define subs by coinduction: to do this they define an L_α-coalgebra whoose unique homomorphism into the final coalgebra yields subs. It is possible to adapt this to the orbit-finite case as follows.

For arbitrary coalgebras $A \xrightarrow{a} L_\alpha A$ and $B \xrightarrow{b} L_\alpha B$ with orbit-finite carriers, one defines $\mathsf{subs}_{A,B} : A \times \mathcal{V} \times B \to \varrho L_\alpha$. This map $\mathsf{subs}_{A,B}$ describes the substitution of a variable within an element of the coalgebra A by some element of the coalgebra B. In the following

$$B \xrightarrow{\mathsf{inl}} B + A \times \mathcal{V} \times B \xleftarrow{\mathsf{inr}} A \times \mathcal{V} \times B \qquad \text{and} \qquad \mathcal{V} \xrightarrow{\mathsf{in}_1} L_\alpha X \xleftarrow{\mathsf{in}_2} [\mathcal{V}]X$$

denote coproduct injections. Now define $B + A \times \mathcal{V} \times B \xrightarrow{[g,h]} L_\alpha(B + A \times \mathcal{V} \times B)$ with $g = L_\alpha(\mathsf{inl}) \circ b$ and $h = [h_{\mathrm{Var}}, h_{\mathrm{Abs}}, h_{\mathrm{App}}] \circ (a \times \mathrm{id} \times \mathrm{id})$ using distributivity and

- $h_{\mathrm{Var}} : \mathcal{V} \times \mathcal{V} \times B \to L_\alpha(B + A \times \mathcal{V} \times B)$, $h_{\mathrm{Var}}(v, w, x) = \begin{cases} \mathsf{in}_1 v & \text{if } v \neq w \\ L_\alpha(\mathsf{inl})(\underbrace{b(x)}_{\in L_\alpha(B)}) & \text{if } v = w \end{cases}$
- $h_{\mathrm{Abs}} : ([\mathcal{V}]A) \times \mathcal{V} \times B \to L_\alpha(B + A \times \mathcal{V} \times B)$, $h_{\mathrm{Abs}} := \mathsf{in}_2 \circ \tau_{A,\mathcal{V}\times B}$, using the strength τ of the functor $[\mathcal{V}]$ with $\tau_{A,\mathcal{V}\times B} : ([\mathcal{V}]A) \times \mathcal{V} \times B \to [\mathcal{V}](A \times \mathcal{V} \times B)$.
- $h_{\mathrm{App}} : A \times A \times \mathcal{V} \times B \to L_\alpha(B + A \times \mathcal{V} \times B)$ is the composition of the obvious steps $A \times A \times \mathcal{V} \times B \to (A \times \mathcal{V} \times B) \times (A \times \mathcal{V} \times B) \to L_\alpha(B + A \times \mathcal{V} \times B)$.

$B + A \times \mathcal{V} \times B$ is orbit-finite, so we get a unique L_α-coalgebra homomorphism $[g', h']$ into ϱL_α, making the following diagram commute:

$$\begin{tikzcd}
B + A \times \mathcal{V} \times B \arrow[r, "{[g,h]}"] \arrow[d, "{[g',h']}"'] & L_\alpha(B + A \times \mathcal{V} \times B) \arrow[d, "{L_\alpha([g',h'])}"] \\
\varrho L_\alpha \arrow[r, "f_\alpha"] & L_\alpha(\varrho L_\alpha)
\end{tikzcd}$$

Now we define $\mathsf{subs}_{A,B} := h'$. It remains to extend this to the desired domain $\varrho L_\alpha \times \mathcal{V} \times \varrho L_\alpha$. Let I be the set of all orbit-finite subcoalgebras (X, a) of $(\nu L_\alpha, t_\alpha)$. Then we know from (the discussion following) Proposition 2.11 that

$$\varrho L_\alpha = \bigcup_{(X,a) \text{ in } I} X.$$

Hence, we can define substitution $\mathsf{subs}_{\mathrm{rat}} : \varrho L_\alpha \times \mathcal{V} \times \varrho L_\alpha \to \varrho L_\alpha$ on rational λ-trees by

$$\mathsf{subs}_{\mathrm{rat}}(x, v, y) = \mathsf{subs}_{A,B}(x, v, y), \text{for some } A, B \in I \text{ with } x \in A \text{ and } y \in B.$$

It remains to prove that the result does not depend on the choice of A and B. But if we have any other (A', a') in I with $x \in A'$ then since both A and A' are subcoalgebras we have $a(x) = t_\alpha(x) = a'(x)$. Similarly for B and $y \in B$.

Thus, since the function h is defined by pattern matching, i.e. on the alternatives indicated by a, it behaves independently from the choice of A and B as desired.

To summarize, we can say that one can define operations on ϱL_α if these n-ary operations can be defined restricted to n orbit-finite subcoalgebras of νL_α as we have just seen for the operation of substitution.

4.2 Normalization of Rational λ-trees

In the λ-calculus different kinds of normal forms play an important role. One of them is the *head normal form* (*hnf*, for short). A λ-term is in hnf if it is of the form $\lambda x_1 \dots \lambda x_n.yN_1 \dots N_m$, where y is a variable and the N_i are arbitrary terms. If one recursively requires the N_i to be in hnf as well, one gets the definition of *Böhm trees*. Adding an additional constant symbol $\bot$ to the syntax of the λ-calculus allows the following corecursive definition of the Böhm tree $\mathsf{BT}(M)$ of a λ-term M (see [18]):

$$\mathsf{BT}(M) = \begin{cases} \lambda x_1 \dots \lambda x_n.y\, \mathsf{BT}(N_1) \dots \mathsf{BT}(N_m) & \text{if } M \twoheadrightarrow_\beta \lambda x_1 \dots \lambda x_n.yN_1 \dots N_m \\ \bot & \text{otherwise,} \end{cases} \tag{2}$$

where $\twoheadrightarrow_\beta$ denoted the reflexive, transitive closure of β-reduction $\rightarrow_\beta$. So a Böhm tree of a term M is the normal form of M in the infinitary λ-calculus, or $\bot$ if there is no normal form [6, 17].

Kurz et al. [18] obtained the operation BT by using the final coalgebra Λ_α^∞ of the following endofunctor Lb_α on Nom expressing the syntax of the λ-calculus extended by $\bot$:

$$\mathsf{Lb}_\alpha X = \mathcal{V} + \{\bot\} + [\mathcal{V}]X + X \times X.$$

The nominal set Λ_α^∞ consists of all λ-trees over $\bot$ modulo α-equivalence, i.e. λ-trees where some leaves are labelled by $\bot$ in lieu of a variable from $\mathcal{V}$. So $\mathsf{BT} : \Lambda_\alpha^\infty \to \Lambda_\alpha^\infty$ is defined as the unique coalgebra homomorphism from a coalgebra $b : \Lambda_\alpha^\infty \to \mathsf{Lb}_\alpha(\Lambda_\alpha^\infty)$, where b is defined by

$$b(M) = \begin{cases} t(N) & \text{if } M \twoheadrightarrow_\beta N \text{ and } N \text{ is in hnf} \\ \bot & \text{else,} \end{cases}$$

into the final coalgebra $\nu\mathsf{Lb}_\alpha$.

The rational fixpoint $\varrho\mathsf{Lb}_\alpha$ consists of the rational λ-trees over $\bot$ modulo α-equivalence; in fact, it is easy to extend to the proof of Theorem 3.4 to the functor Lb_α.

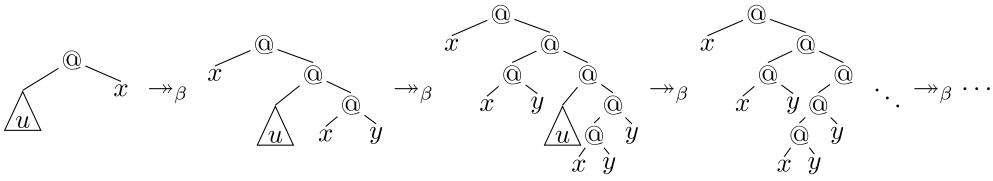
However, BT does not restrict to $\varrho\mathsf{Lb}_\alpha$. Consider the λ-term

$$u := Y_{(\lambda g.\lambda x.x\,(g(x\,y)))}, \text{ with } Y_f := (\lambda z.f(z\ z))(\lambda z.f(z\ z)). \tag{3}$$

which is finite and therefore rational. In other words, u represents an element of $\varrho\mathsf{Lb}_\alpha$. Let us look at its Böhm tree, by considering the β-reduction sequence of u using $Y_f \rightarrow_\beta f(Y_f f)$:

$$u\,x = Y_{(\lambda g.\lambda x.x\,(g(x\,y)))}\,x \rightarrow_\beta (\lambda g.\lambda x.x\,(g(x\,y)))\,\overbrace{Y_{(\lambda g.\lambda x.x\,(g(x\,y)))}}^{u}\,x \twoheadrightarrow_\beta x\,(u\,(x\,y))$$

Applying $ux \twoheadrightarrow_\beta x\,(u\,(x\,y))$ multiple times yields the following sequence:

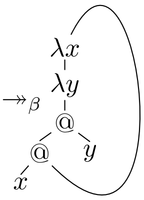


The resulting infinite λ-tree is clearly not rational; in fact, consider the subtrees defined by the left-hand cildren of every node on the right-most path. Then the subtree rooted at the left-hand sucessor of the n-th node on that path has the list xy^n as its front of leaf labels.

And since this tree does not contain any λ-operators its α-equivalence class is a singleton, whence $\mathsf{BT}(u) \notin \varrho\mathsf{Lb}_\alpha$.

Of course, there are also λ-terms, whose Böhm tree is infinitely large but stays rational, for example:

$$s := Y_{(\lambda g.\lambda x.\lambda y.x\,g\,y)))}, \text{ with } s \to_\beta (\lambda g.\lambda x.\lambda y.x\,g\,y))) \, s \to_\beta \lambda x.\lambda y.x\,s\,y. \twoheadrightarrow_\beta \cdots \twoheadrightarrow_\beta$$



The rational λ-tree on the right above, call it r, is the Böhm tree for s, i.e. $\mathsf{BT}(s) = r \in \varrho\mathsf{Lb}_\alpha$. In other words the subcoalgebra S of $(\Lambda^\infty_\alpha, b)$ above generated by $[s]_\alpha$ is orbit-finite, hence the restriction of BT to S factorizes through $\varrho\mathsf{Lb}_\alpha$.

Can one characterize the largest subcoalgebra of $(\Lambda^\infty_\alpha, b)$ whose image under BT lies in $\varrho\mathsf{Lb}_\alpha$? We leave this question for further work.

5 Conclusion and Future Work

We have contributed to the abstract algebraic study of variable binding using nominal sets. In particular, we have extended a recent coalgebraic approach to infinitary λ-calculus due to Kurz et al. Whereas they proved in [18] that λ-trees with finitely many variables modulo α-equivalence form the final coalgebra for the functor L_α on Nom we have given a characterization of the rational fixpoint of that functor. It contains precisely the rational λ-trees modulo α-equivalence.

This characterization entails a corecursion principle for rational λ-trees because the rational fixpoint is the final locally orbit-finite coalgebra for L_α. In this sense we have achieved finitary corecursion for the infinitary λ-calculus. We have demonstrated the new principle and its limitations with two applications: a corecursive definition of substitution and of a normalform computation.

Our work is only a first step in the study of the coalgebraic approach to finitary coinduction for infinitary terms with variable binding operators. First, it should be clear that our results generalize from λ-terms to the rational fixpoint for endofunctors on Nom associated to a binding signature. Other points for future work are: (1) the extension of the coalgebraic approach to rational and infinitary λ-terms using nominal sets to treat the solutions of higher-order recursion schemes as was done in the setting of presheaves on finite sets in [3], and (2) the study of specification formats that extend our simple corecursion principle that follows from finality; more precisely, Bonsague et al. [7, 21] have proposed bipointed specifications as an abstract format (by restricting Turi's and Plotkin's abstract GSOS rules [25]) to specify algebraic operations on the rational fixpoint of an endofunctor. It should be interesting to work out a concrete rule format corresponding to bipointed specifications for rational λ-terms and rational terms for arbitrary binding signatures. Last, but not least, the similarity of the results in [3] on the one hand and those in [18] and here on the other hand is so striking that there should be a formal connection; however, to our knowledge this has not been worked out in the literature yet.

References

1 Jiří Adámek. Introduction to coalgebra. *Theory Appl. Categ.*, 14:157–199, 2005.

2 Jiří Adámek, Stefan Milius, and Jiri Velebil. Iterative algebras at work. *Math. Structures Comput. Sci*, 16(6):1085–1131, 2006.

3 Jiří Adámek, Stefan Milius, and Jiri Velebil. Semantics of higher-order recursion schemes. *Log. Methods Comput. Sci.*, 7(1), 2011.

4 Jiří Adámek and Jiří Rosický. *Locally presentable and accessible categories.* Cambridge University Press, 1994.

5 André Arnold and Maurice Nivat. The metric space of infinite trees. algebraic and topological properties. *Fundam. Inform.*, 3(4):445–476, 1980.

6 Hendrik Pieter Barendregt. *The Lambda calculus: Its syntax and semantics.* North-Holland, Amsterdam, 1984.

7 Marcello M. Bonsangue, Stefan Milius, and Jurriaan Rot. On the specification of operations on the rational behaviour of systems. In *Proc. EXPRESS/SOS'12*, volume 89 of *Electron. Proc. Theoret. Comput. Sci.*, pages 3–18, 2012.

8 Marcello M. Bonsangue, Stefan Milius, and Alexandra Silva. Sound and complete axiomatizations of coalgebraic language equivalence. *ACM Trans. Comput. Log.*, 14(1:7), 2013.

9 Bruno Courcelle. Fundamental properties of infinite trees. *Theoret. Comput. Sci.*, 25:95–169, 1983.

10 Calvin C. Elgot. Monadic computation and iterative algebraic theories. In H. E. Rose and J. C. Sheperdson, editors, *Logic Colloquium 1973*, volume 80, pages 175–230, Amsterdam, 1975. North-Holland Publishers.

11 Marcelo Fiore, Gordon D. Plotkin, and Daniele Turi. Abstract syntax and variable binding. In *Proc. Logic in Computer Science 1999*, pages 193–202. IEEE Press, 1999.

12 Murdoch Gabbay and Andrew M. Pitts. A new approach to abstract syntax involving binders. In *Proc. LICS'99*, pages 214–224. IEEE Computer Society Press, 1999.

13 Peter Gabriel and Friedrich Ulmer. *Lokal präsentierbare Kategorien*, volume 221 of *Lecture Notes Math.* Springer-Verlag, 1971.

14 Fabio Gaducci, Marino Miculan, and Ugo Montanari. About permutation algebras, (pre)sheaved and names sets. *Higher-Order Symb. Comput.*, 19:283–304, 2006.

15 Susanna Ginali. Regular trees and the free iterative theory. *J. Comput. System Sci.*, 18:228–242, 1979.

16 Bart Jacobs and Jan Rutten. A tutorial on (co)algebras and (co)induction. *EATCS Bulletin*, 62:62–222, 1997.

17 Richard Kennaway, Jan Willem Klop, Ronan Sleep, and Fer-Jan de Vries. Infinitary Lambda Calculus. *Theoret. Comput. Sci.*, 175(1):93–125, 1997.

18 Alexander Kurz, Daniela Petrisan, Paula Severi, and Fer-Jan de Vries. Nominal coalgebraic data types with applications to lambda calculus. *Log. Methods Comput. Sci.*, 9(4), 2013.

19 Joachim Lambek. A fixpoint theorem for complete categories. *Math. Z.*, 103:151–161, 1968.

20 Stefan Milius. A sound and complete calculus for finite stream circuits. In *Proc. LICS'10*, pages 449–458. IEEE Computer Society, 2010.

21 Stefan Milius, Marcello M. Bonsangue, Robert S.R. Myers, and Jurriaan Rot. Rational operation models. In *Proc. MFPS XXIX*, volume 298 of *Electron. Notes Theor. Comput. Sci.*, pages 257–282, 2013.

22 Stefan Milius and Thorsten Wißmann. Finitary corecursion for the infinitary lambda calculus. full version; available at `http://arxiv.org/abs/1505.07736`, 2015.

23 Daniela Petrişan. *Investigations into Algebra and Topology over Nominal Sets.* dissertation, University of Leicester, 2011.

24 Andrew M. Pitts. *Nominal Sets: Names and Symmetry in Computer Science*, volume 57 of *Cambridge Tracts in Theoretical Computer Science.* Cambridge University Press, 2013.

25 Gordon D. Plotkin and Daniele Turi. Towards a mathematical operational semantics. In *Proc. Logic in Computer Science (LICS'97)*, pages 280–291, 1997.

26 Jan Rutten. Universal coalgebra: a theory of systems. *Theoret. Comput. Sci.*, 249(1):3–80, 2000.

27 Jan Rutten. Rational streams coalgebraically. *Log. Methods Comput. Sci.*, 4(3:9):22 pp., 2008.

28 Thorsten Wißmann. The rational fixed point in nominal sets and its application to infinitary lambda-calculus. Project report, available at `http://thorsten-wissmann.de/theses/project-wissmann.pdf`, October 2014.