Limitations of Choiceless Computation

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der
RWTH Aachen University zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Benedikt Pago, M.Sc.
aus München

Berichter: Universitätsprofessor Dr. Erich Grädel
Professor Dr. Anuj Dawar

Tag der mündlichen Prüfung: 12. Juli 2023

Abstract

One of the central open questions in finite model theory asks whether there exists a logic that captures polynomial time. This question is significant for several reasons, one of them being that a negative answer would separate P from NP, by Fagin's theorem. Yuri Gurevich, who was the first to make this question precise, conjectured that no logic captures polynomial time. This motivates the research on lower bounds against symmetric computation models contained in PTIME. One of the few such formalisms that have not yet been separated from PTIME is the logic Choiceless Polynomial Time (CPT). Establishing strong lower bounds for CPT has been a challenging problem ever since its invention by Blass, Gurevich and Shelah in 1999. This thesis focuses exactly on this goal. Our approach is mainly based on the famous Cai-Fürer-Immerman (CFI) query, which can be seen as an instance of the graph isomorphism problem but also as a linear equation system over the field $\mathbb{Z}_2$. Variants of this problem have been used to separate fixed-point logic with counting and rank logic from PTIME. Results by Dawar, Richerby and Rossman, and subsequently by Pakusa, Schalthöfer and Selman show that the CFI-query is definable in CPT in the following cases: over linearly ordered base graphs, preordered base graphs with colour classes of logarithmic size, and unordered base graphs of linear degree. However, the general unordered case remains open. In Chapter 7, we show that there is a family of unordered base graphs (namely, hypercubes) of sublinear degree which does not admit CPT-definable preorders with logarithmic colour classes. Consequently, none of the currently known CPT-algorithms for the CFI-query can be used to solve the unordered case. In Chapter 8, we go a step further: We define a general class of choiceless algorithms for the CFI-query that are based on the Dawar-Richerby-Rossman (DRR) technique; this encompasses all the known algorithms mentioned above. Then we show that the CFI-query on a class $\mathcal{K}$ of base graphs is not definable by a DRR-algorithm unless there exists a family of polynomial-size symmetric Boolean XOR-circuits with the same symmetries as the graphs in $\mathcal{K}$ and certain restrictions on the connectivity between the gates. In Chapter 9, we also present an almost sufficient lower bound against these circuits: If the connectivity and symmetry restrictions are slightly strengthened, and $\mathcal{K}$ is the class of n-dimensional hypercubes, then the required circuit family indeed does not exist. It remains as a problem for future work to lift this non-existence result also to the less restricted circuits – this would show that no DRR-algorithm decides the CFI-query on unordered hypercubes. Finally, in Chapter 10, we propose a different approach towards CPT lower bounds: We show that if CPT can distinguish all pairs of non-isomorphic graphs in a family $\mathcal{K}$, then this is also possible in a propositional proof system called the degree-3 extended polynomial calculus. Thus, potential future lower bounds for solving graph isomorphism in this proof system translate into CPT lower bounds and could also lead to a separation of CPT from PTIME.

Zusammenfassung

Eines der zentralen Probleme der endlichen Modelltheorie ist die Frage nach der Existenz einer Logik für Polynomialzeit. Eine negative Antwort würde wegen des Satzes von Fagin sofort die Komplexitätsklassen P und NP trennen. Tatsächlich vermutete Yuri Gurevich, der die Frage als erster in der Form formulierte, dass es keine Logik für P gibt. Dies motiviert die Forschung an unteren Schranken für Logiken, die in P enthalten sind. Eine der wenigen solchen Logiken, die bisher nicht von P getrennt werden konnte, ist Choiceless Polynomial Time (CPT), ein symmetrieinvariantes Berechnungsmodell, das 1999 von Blass, Gurevich und Shelah eingeführt wurde. Diese Arbeit ist vor allem dem Ziel gewidmet, die Grenzen der Ausdrucksstärke von CPT besser zu verstehen. Zu diesem Zweck betrachten wir hauptsächlich das Isomorphieproblem auf Cai-Fürer-Immerman (CFI) Graphen. Varianten dieses Problems sind bereits verwendet worden, um Fixpunktlogik mit Zählen und Ranglogik von P zu trennen. Resultate von Dawar, Richerby und Rossman, und darauf aufbauend von Pakusa, Schalthöfer und Selman zeigen, dass folgende Varianten des CFI-Problems in CPT definierbar sind: CFI über linear geordneten Basisgraphen, über prägeordneten Basisgraphen mit logarithmisch großen Farbklassen und Basisgraphen mit linearem Grad. Der allgemeine, ungeordnete Fall ist allerdings noch offen. In Kapitel 7 beweisen wir, dass eine Familie von ungeordneten Basisgraphen (nämlich Hyperwürfel) existiert, die sublinearen Grad haben und keine CPT-definierbaren Präordnungen mit logarithmischen Farbklassen erlauben. Daraus folgt, dass das CFI-Problem auf ungeordneten Hyperwürfeln mit keinem der bisher bekannten CPT-Algorithmen lösbar ist. In Kapitel 8 gehen wir noch einen Schritt weiter, indem wir eine allgemeine Klasse von CPT-Algorithmen für das CFI-Problem definieren, welche auf der Idee von Dawar, Richerby und Rossman (DRR) aufbauen; dies trifft insbesondere auf alle bisher überhaupt bekannten solchen Algorithmen zu. Wir zeigen, dass das CFI-Problem über einer Graphklasse $\mathcal{K}$ nur dann von einem DRR-Algorithmus gelöst werden kann, wenn eine Familie von symmetrischen XOR-Schaltkreisen existiert, die die gleichen Symmetrien wie die Graphen in $\mathcal{K}$ aufweisen und einige weitere Einschränkungen erfüllen. In Kapitel 9 zeigen wir dann, dass solche Schaltkreisfamilien mit nur geringfügig stärkeren Einschränkungen nicht existieren können, wenn wir für $\mathcal{K}$ wieder die Klasse der n-dimensionalen Hyperwürfel wählen. Dieses Resultat beweist also fast, dass kein DRR-Algorithmus das CFI-Problem über ungeordneten Hyperwürfeln lösen kann. In Kapitel 10 betrachten wir schließlich einen völlig anderen Ansatz und beweisen folgende Aussage: Jedes Paar nicht-isomorpher Graphen, das in CPT unterscheidbar ist, ist auch in einer Variante des *extended polynomial calculus* unterscheidbar. Als Konsequenz lassen sich potentielle zukünftige untere Schranken für die Komplexität des Graphisomorphieproblems in diesem algebraischen Beweiskalkül auf CPT übertragen, was ebenfalls zur Trennung von CPT und P führen könnte.

Contents

1 Introduction **11**

1.1 A hierarchy of logics in polynomial time 15
1.2 Contributions 21
1.3 Acknowledgements 28

2 Preliminaries **31**

2.1 Sets and linear orders 31
2.2 Descriptive complexity theory 32
2.3 Interpretations 33
2.4 Fixed-point logic with counting, pebble games, and the Weisfeiler-Leman algorithm 33
2.5 Treewidth 36
2.6 Permutation groups 37
2.7 Linear algebra 37

3 Choiceless Polynomial Time **39**

3.1 Comparing CPT with other logics 45
3.2 The power and limitations of CPT 46
3.2.1 Expressivity results 47
3.2.2 Non-definability results 49
3.3 Alternative presentations and variants of CPT 52
3.3.1 Polynomial-time Interpretation Logic 52
3.3.2 Choiceless Logarithmic Space 56
3.3.3 Deep Weisfeiler Leman 57
3.3.4 Choiceless Polynomial Time with Witnessed Symmetric Choice . . 58
3.3.5 Choiceless Polynomial Time as a fragment of an infinite-set based computation model 60
3.4 Questions that are not addressed in this thesis 61
3.4.1 Separating CPT from NP 61
3.4.2 A circuit characterisation of CPT 61

4 Symmetries and Permutation Groups **65**

4.1 Basic notions from group theory 65
4.2 Supports and supporting partitions 68
4.3 Every large permutation group contains a product of large alternating groups 72

4.4 Orbits, supports and Choiceless Polynomial Time 82
4.4.1 Groups with the (k,r)-support property 84
4.5 Outlook: Symmetries beyond automorphisms 87

5 The Cai-Fürer-Immerman graphs and their symmetries 89
5.1 The CFI-construction 90
5.2 Automorphisms of unordered CFI-graphs 92
5.3 CFI-structures over hypercubes 96

6 Prior research on the choiceless (non-)definability of Cai-Fürer-Immerman problems 99
6.1 Positive results: The super-symmetric object technique 99
6.1.1 Defining the CFI query on linearly ordered base graphs 99
6.1.2 Defining the CFI query on preordered base graphs 103
6.1.3 Defining the CFI query on base graphs of large degree 105
6.2 Negative results: Lower bounds on the support size 105

7 The non-definability of preorders with small colour classes in hypercubes 113
7.1 Using supporting partitions to estimate the stabiliser sizes 115
7.2 Proof of the Superpolynomial-Orbit Theorem 116
7.2.1 The case of sublinearly bounded supports 116
7.2.2 The case of linearly-sized supports 126
7.3 Conclusion and future research 136

8 Lower bounds for Choiceless Polynomial Time via Symmetric Circuits 139
8.1 Symmetries and supports of hereditarily finite sets over CFI graphs 141
8.2 CFI-symmetric hereditarily finite sets 145
8.3 Symmetric XOR-circuits 149
8.3.1 Symmetries of circuits 150
8.3.2 The parameter fan-in dimension 151
8.4 Constructing XOR-circuits from hereditarily finite objects 152
8.5 Applying the XOR-circuit construction to CFI-symmetric algorithms . . . 157
8.6 Extending the circuit construction to non-CFI-symmetric sets 159
8.6.1 Definition of the matrices 161
8.6.2 Construction of the circuit 173
8.6.3 Bounding the size of the circuit 177
8.6.4 Which vector spaces have symmetric bases? 182
8.7 Conclusion and future research 187

9 Lower bounds for families of symmetric XOR-circuits over hypercubes 189
9.1 Properties of hypercubes 189
9.2 Symmetric XOR-circuits over hypercubes 197
9.3 Conclusion and future research 210

10 Choiceless Polynomial Time and Propositional Proof Complexity 213
10.1 The (extended) polynomial calculus 219
10.2 Expressing graph isomorphism as a polynomial equation system 220
10.3 Separating the extended polynomial calculus from its non-extended version 222
10.4 Deep Weisfeiler Leman 222
10.4.1 Coherent configurations and algebraic sketches 223
10.4.2 The Deep Weisfeiler Leman computation model 224
10.4.3 Simplifications of DWL 226
10.4.4 Distinguishing graphs in Deep Weisfeiler Leman 228
10.4.5 Properties of coherent configurations 235
10.5 Refuting graph isomorphism in the extended polynomial calculus 237
10.6 Conclusion and future research 244

1 Introduction

The question whether there exists a logic that captures PTIME is one of the most prominent open problems in finite model theory. It was first brought up by Chandra and Harel in 1982; in [30], they posed the question whether there exists a database query language that can express precisely the polynomial time computable queries. A few years later, Yuri Gurevich rephrased this question and asked for a *logic* for polynomial time [66]. What this means is that evaluating each fixed sentence of such a logic should be a polynomial time problem (in the size of the given input structure) and conversely, any PTIME-decidable property of finite structures should be definable by a sentence in the logic. Gurevich also made precise how the term "logic" should be understood in this context. This is important for the question to make any sense at all because otherwise, one can easily find answers which are formally correct but obviously silly and beside the point of the question: For example, one could take the "logic" whose sentences are all polynomial time Turing machines. This would indeed be a logic for polynomial time in a very liberal sense of the word. However, according to Gurevich's definition, one of the key properties that sets logics apart from other classical computation models such as computers and Turing machines is *symmetry-invariance*. This means that logical sentences do not distinguish between isomorphic structures. With this definition, the question for a logic capturing PTIME can be seen as the question whether one can compute all PTIME-decidable properties of finite structures in a formalism that has no access to an ordered representation of the input structure. Turing machine computation is not symmetry-invariant: The perhaps subtle but important difference is that Turing machines simply do not perform computations on finite relational structures, but on binary strings encoding them. While it is not difficult to write down any given finite structure as a binary string (e.g. a graph can be encoded as its adjacency matrix), this string encoding necessarily removes all symmetries from the structure. For example, a graph $G = (V, E)$, viewed as a structure with one binary relation symbol E, has a priori no first and no last vertex; the universe is just an unordered set. But any string clearly is an ordered sequence of symbols, and so there is a first and a last vertex in the adjacency matrix. Thus, the seemingly harmless step of transforming a structure into a string defines a linear order on the universe. When we feed a graph $G = (V, E)$ to a Turing machine, the machine actually receives a *linearly ordered* structure $G_< = (V, E, <)$. Therefore, it can compute queries which are not invariant under the symmetries of the original unordered structure G. For example, it can output the degree of the "first vertex" of $G_<$, which is not even a meaningful query on G. So there is a mismatch between Turing machine computation and computation on structures, as we would naturally view it in finite model theory: Turing machines are inherently only applicable to ordered structures. Now one might say that computing the degree of the first vertex in $G_<$ is just a bad example because we could restrict ourselves to Turing machines that do compute isomorphism-invariant queries on G, even though they receive the input $G_<$. For example, we could focus on queries like "is G connected?", which only depend on G and not on the order. Indeed, the result of such queries is isomorphism-invariant, and a Turing machine computing them could use the linear order $<$, that is implicit in the string encoding of G, in the computation, but the final output "yes" or "no" would be independent of $<$. So then, why is the class of all polynomial time algorithms that decide isomorphism-invariant properties of graphs not a logic for PTIME? Besides being symmetry-invariant, a logic should also have a *decidable syntax*. Turing machines which compute order-invariant graph properties are not logical sentences because it is not decidable if their output is indeed order-invariant. Thus, a logic capturing PTIME must not be some undecidable symmetry-invariant fragment of an asymmetric computation model, but rather, it must be symmetry-invariant by design. To enforce this, one more additional requirement is needed: Namely, the syntax should be *effective*, meaning that there is an algorithm which computes, for any sentence ψ, a polynomial time Turing machine M_ψ that evaluates ψ in a given input structure. So there must be a meaningful relationship between the syntax and the algorithmic evaluation of the sentences – else, there exist again pathological solutions.

In total, the question for a PTIME-logic is really about what kind of computation is possible if no symmetry-breaking operations are allowed, or more precisely: What is the computational cost of symmetry-invariance? It depends on the answer of this question whether or not there exists a logic for PTIME. Note that if we disregard complexity issues, then one can simulate any asymmetric computation by a symmetric one: Given an input structure of size n, we can compute the set of all $n!$ many possible total orders on the universe, then perform a standard Turing machine computation on each of the ordered structures, and then check if the results are all the same. This way, the output is guaranteed to be isomorphism-invariant. In fact, such algorithms can be implemented in *BGS-logic*, so this is indeed possible in a symmetry-invariant computation model. However, this computation can obviously not be carried out in polynomial time because it requires space and time in the order of $n!$. In fact, Gurevich himself conjectured that the cost of symmetry-invariance is in general super-polynomial, i.e. that there exists no logic for PTIME. This conjecture is still open.

The aim of this thesis is to deepen the understanding of one of the most important candidate logics in PTIME, which have not yet been separated from it. This is *Choiceless Polynomial Time* (CPT), which was introduced by Blass, Gurevich, and Shelah in 1999 [21]. It can be seen as an extension of the better-understood *fixed-point logic with counting* by certain powerful data structures, namely *hereditarily finite sets*. For all we know, it may be the case that CPT captures PTIME but it may also be that we simply have not found sufficiently strong techniques yet for proving limitations of its expressive power. This is what we mainly focus on in this thesis. Before we give a more detailed account of Choiceless Polynomial Time and the other most important logics that have been studied in the quest for a PTIME-logic, let us take a look at a few other central questions from theoretical computer science; the (im)possibility of efficient symmetric computation plays an interesting role in all of them.

Classical complexity theory If Gurevich's conjecture about the non-existence of a logic for P were proven to be true, then it would mean that computation on ordered structures and strings is fundamentally easier than on unordered structures, which is maybe surprising from the point of view of classical complexity theory. But it would have an even more important consequence: It would entail that P $\neq$ NP. This is so because the complexity class NP can actually be captured by existentially second-order logic, which was proved by Ronald Fagin in 1974 [49]. The reason why symmetry-invariance comes at no extra cost in this case is that in existential second-order logic, it is possible to guess a linear order with the existential quantifier. Once an order is available, one can simulate a non-deterministic polynomial time Turing machine in $\exists$SO. If P = NP, then it is not difficult to show that $\exists$SO also captures P in the sense of Gurevich's definition. Therefore, the quest for a logic for P can also be seen as a programme towards separating P from NP by defining stronger and stronger logics within PTIME and then trying to disprove them as candidates for capturing P. Eventually, one might perhaps find out that indeed, no logic captures P, and hence, P $\neq$ NP, or one does find such a logic and can then compare it with $\exists$SO. The success chances of this programme are of course unclear, but certainly, the study of symmetric computation models is generally easier than that of Turing machines: There are many tools and techniques from finite model theory, such as Ehrenfeucht-Fraïssé style games or zero-one laws, that can be used to obtain inexpressibility results for logics. Also, there exist well-known "benchmark constructions" – usually based on the famous Cai-Fürer-Immerman graphs [29] – of structures on which many typical logics are particularly inexpressive. We introduce this CFI-construction in Chapter 5 but for the moment, it suffices to know that it can be applied to any family of connected graphs in order to obtain pairs of non-isomorphic structures, which are indistinguishable for many logics.
To study the limitations of symmetric computation, we can build on all this machinery from finite model theory, and hence, making progress with respect to lower bounds seems easier than in classical complexity theory.

Graph isomorphism Another interesting aspect of symmetric computation is its connection to the *graph isomorphism* problem and to graph *canonisation.* The former is the problem to decide, given two graphs G and H, if they are isomorphic or not, and the latter is the problem of computing a canonical form for a given graph G. A canonical form (also called canon) is a copy of the graph with a linearly ordered vertex set such that any two graphs are isomorphic if and only if they are mapped to the same canon. The canonisation problem is at least as hard as the isomorphism problem because a canonisation algorithm can be used to decide isomorphism: Simply compute the canons of the two given graphs G and H and check whether they are identical or not.
The graph isomorphism problem receives a lot of attention in theoretical computer science because its complexity status is still unresolved. It is contained in NP because

it admits an obvious certificate, namely the sought isomorphism, but it is probably not NP-complete: If it were, then the polynomial hierarchy would collapse [99]. Moreover, it would mean that all NP-complete problems can be solved in quasi-polynomial time by Babai's algorithm [15]. Neither of this seems very plausible. There are many graph classes for which isomorphism testing has been shown to be in P (see e.g. [104]), but a general polynomial time graph isomorphism test is not known. Graph isomorphism and canonisation play a central role when it comes to capturing complexity classes by logics. Namely, if a "computational logic", such as *fixed-point logic*, can canonise the graphs in a graph class $\mathcal{K}$, then the logic also captures PTIME on the class $\mathcal{K}$ by the famous *Immerman-Vardi theorem* [74] [103]. We elaborate further on this below, but essentially, this theorem formalises the intuition we have given so far, that on ordered structures, symmetric computation is the same as classical Turing machine computation. Thus, a standard approach for proving that a logic captures PTIME on a given restricted class of graphs is to show that canonisation (and hence also graph isomorphism) is definable in the logic. Interestingly, certain instances of the graph isomorphism problem (for example, the aforementioned Cai-Fürer-Immerman graphs) are provably hard for many important logics in P, such as *fixed-point logic with counting.* So, roughly speaking, the expressive power of a given logic seems to depend on which instances of the graph isomorphism problem can be solved in that logic. Very recently, Moritz Lichter and Pascal Schweitzer confirmed this intuition in some sense by presenting a logic which captures PTIME on a class of graphs if and only if it defines the isomorphism problem on that class [85]. This logic is an extension of *Choiceless Polynomial Time*, that we study in this thesis, with a *witnessed symmetric choice* operator. In Section 3.3.4, we review it in more detail.
Moreover, the graph distinguishing power of certain important logics in PTIME corresponds precisely to the power of certain well-known graph isomorphism heuristics. For example, *fixed-point logic with counting* distinguishes exactly those graphs which are also distinguishable by the k-dimensional *Weisfeiler Leman* algorithm, and Choiceless Polynomial Time corresponds to an extension thereof called *Deep Weisfeiler Leman*, introduced by Grohe, Schweitzer and Wiebking [63]. Deep Weisfeiler Leman can informally be seen as a framework that captures *all* combinatorial (i.e. choiceless) graph isomorphism algorithms, and we introduce it in detail in Chapter 10. Therefore, lower bounds for fixed-point logic with counting and Choiceless Polynomial Time also have implications for the limitations of certain classes of graph isomorphism tests.

Propositional proof complexity Besides classical complexity theory and the graph isomorphism problem, there is (at least) a third research topic that is connected to finite model theory, and this is *propositional proof complexity.* In this area, one is interested in the power of different propositional proof systems such as for example *resolution* or the *polynomial calculus* [32]. The overarching programme of proof complexity is to show lower bounds on the proof size that is required to prove/refute propositional tautologies/unsatisfiable formulas in different proof systems of increasing efficiency. Somewhat similarly to the quest for a logic capturing PTIME, one of the big goals of proof complexity is to find a proof system in which every tautology has a polynomial-size

proof, or to show that no such proof system exists. The existence of such an efficient proof system is equivalent to NP = coNP. So just like the quest for a PTIME-logic is motivated by the complexity-theoretic question "P = NP?", the search for a proof system with polynomial complexity is motivated by the question "NP = coNP?"
The connection between these two quests is that the methods for proving lower bounds can to some extent be transferred from one field to the other. For example, Atserias [9] and Atserias and Dalmau [11] observed a tight connection between the number of pebbles needed to win an existential pebble game from finite model theory on pairs of Boolean formulas and the width and space complexity of resolution.
In [90], it was shown that different variants of fixed-point logics have the same expressive power as bounded-width resolution and the bounded-degree polynomial calculus proof system. This shows that generally, lower bounds from finite model theory (for example obtained via pebble games) directly apply to these proof systems as well, and vice versa. Other results of a similar kind are [17] and [13]. In the former article, Berkholz and Grohe study the graph distinguishing power of the degree-k polynomial calculus and show that it distinguishes exactly the same graphs that are also distinguishable with the k-dimensional Weisfeiler Leman algorithm. In [13], Atserias and Ochremiak prove that the bounded-degree *sums of squares* proof system can be simulated in bounded-variable counting logic and thus, lower bounds for this logic imply lower bounds for the degree of sums of squares proofs. Hence, the search for a logic capturing PTIME is interesting not only for its own sake but the results and techniques that are encountered along the way can be inspiring in other contexts, too.

1.1 A hierarchy of logics in polynomial time

Now let us take a look at the logics and paradigms that have been discovered so far during this quest. For further reading, we recommend Martin Grohe's survey [59] from 2008 (even though it does not include the latest developments, naturally). The most basic logic in descriptive complexity theory is probably *first-order logic* (FO). Its model-checking problem is clearly in polynomial time (if the sentence is fixed) but it it also clear that FO is far from capturing PTIME: Due to its locality (as made precise by Hanf and Gaifman [47]), it cannot define, for example, if a graph is connected or not. A natural way to overcome this is adding an operator for defining transitive closures to FO, which results in *transitive closure logic* (TC). Different variants of this logic capture LOGSPACE and NLOGSPACE, respectively, on linearly ordered structures. However, on general unordered structures, transitive closure logic (even extended with a counting operator) fails to capture LOGSPACE because it cannot define the isomorphism problem on trees [48].

A more general recursion/iteration mechanism is given by *fixed-point operators*. The extension of first-order logic with such an operator is called *fixed-point logic*. There are different variants of fixed-point operators (least fixed-point, greatest fixed-point, inflationary fixed-point) but they are all equally expressive on finite structures [47] – hence the general term fixed-point logic usually refers to all these variants. To see

how fixed-point formulas work, consider for example the *inflationary* fixed-point operator $[\mathbf{ifp}\, X\overline{x}.\varphi(X,\overline{x})]$, where φ is a first-order formula (potentially containing also fixed-point operators). It defines in a given structure $\mathfrak{A}$ the fixed-point of the sequence $X^0 = \emptyset, ..., X^{i+1} = X^i \cup \{\overline{a} \in A^k \mid \mathfrak{A} \models \varphi(X^i, \overline{a})\}$, where k is the arity of the second-order variable X. With this operator, first-order logic is enriched by a quite natural iteration mechanism. Since the arity k of the fixed-point relation is fixed for any fixed formula, such sentences can be evaluated in polynomial time (at most $|A|^k$ tuples can be added to the relation in total). Note that this computation model is indeed symmetry-invariant because in each step of the fixed-point induction, a definable set of tuples is added to X, and such a set is always closed under the automorphisms of the structure.
The *Immerman-Vardi theorem* mentioned above states that fixed-point logic captures polynomial time on all linearly ordered finite structures. This is the case because in an ordered structure $\mathfrak{A}$, it is possible to logically define a binary string encoding of $\mathfrak{A}$. Then on this binary string encoding, one can simulate any polynomial time Turing machine with the help of the fixed-point operator. However, fixed-point logic is not strong enough to define all polynomial time properties on general *unordered* structures because it cannot count. Surprisingly, it cannot even define the seemingly simple query whether the size of the input structure is even or odd (known as the EVEN-query). This is the case because fixed-point logic can be embedded into infinitary first-order logic, which has a *zero-one law*. Therefore, every fixed-point sentence is true with probability either zero or one in random structures whose size tends towards infinity. Of course, the structure size being even has no fixed probability in the limit, and so, this property is not definable (see e.g. [81] or [65]).

The natural fix for this is adding counting terms, yielding *fixed-point logic with counting* (FPC). This logic is evaluated in two-sorted structures, where the first sort is the actual structure and the second sort is a linear order of the same size as the structure. The elements of the second sort are the numbers that the counting terms evaluate to. These terms can count the number of satisfying assignments for formulas. Fixed-point logic with counting is perhaps the most important logic of reference contained in PTIME, and it constitutes a robust complexity class. For example, we have already mentioned that its graph distinguishing power is the same as that of the famous k-dimensional Weisfeiler Leman algorithm. Another perhaps surprising characterisation of FPC is in terms of certain symmetric circuit families with Boolean and counting gates [6]. Many important classical algorithms can actually be implemented in FPC, for example the ellipsoid method for solving linear programs [7]. Moreover, FPC is powerful enough to capture PTIME on a large and natural graph class, namely on all graphs with an excluded minor. This result is due to Martin Grohe [60, 61] and builds on the graph structure theory developed by Robertson and Seymour. Essentially, graphs with an excluded minor possess enough structure that can be exploited by FPC in order to canonise them. All these results establish FPC as a robust and quite powerful symmetric computation model in PTIME. For more information on fixed-point logic with counting, we refer to the survey [34] or [86]. However, even though FPC distinguishes almost all graphs and captures PTIME on them [72], there is a prominent example of non-isomorphic graphs that cannot be distinguished in FPC. These are the aforementioned CFI graphs. In a sense, such graphs encode linear equation systems over the finite field $\mathbb{F}_2$ (or also other finite fields) [10]. Thus, FPC cannot define whether a given linear equation system over a finite field has a solution; this separates it from PTIME. It should be noted, though, that linear equation systems over the field $\mathbb{Q}$ can in fact be solved in FPC [73].

Symmetric choices There are basically three different ways to continue from here in the attempt to find a logic capturing P. Let us start with the probably least studied route, namely extending fixed-point logic with *witnessed symmetric choices.* Here, the idea is to allow an arbitrary non-deterministic choice from a definable choice set in every step of the fixed-point induction. In general, such choices may break symmetries, which would not be allowed in a logic. Therefore, all elements in the choice set are required to be related via an automorphism of the structure (i.e. the choice sets must be *orbits*). However, testing whether a given set is an orbit of the structure is as hard as the graph isomorphism problem and thus not known to be in PTIME. To ensure that the sentences of the logic can still be evaluated in polynomial time, the formulas must also define automorphisms which certify that the choice set is indeed an orbit. A fixed-point computation involving such non-deterministic symmetric choices is a branching computation tree, whose branches are all related via automorphisms of the structure. Therefore, the final result is in a sense independent of the non-deterministically chosen branch.
Variants of such fixed-point logics with symmetric choice were studied by Dawar and Richerby [93] [39], and also by Gire and Hoang [51]. It had been open whether Gire and Hoang's fixed-point logic with counting and witnessed symmetric choice captures PTIME, but there is a recent unpublished article by Lichter, in which this is disproved using a variant of the CFI-construction [83]. However, it remains open whether an extension of that logic by an *interpretation operator* is strong enough to capture P. Perhaps one of the downsides of the symmetric choice approach towards capturing PTIME is that witnessed symmetric choices are only useful on structures where orbits are definable and non-trivial: If witnesses for the choice sets being orbits cannot be defined in the logic, then the choice operator cannot be used, and if the orbits are trivial, i.e. singleton points, then each choice is from a singleton set, which renders the choice operator superfluous. These obstacles to the applicability of the witnessed choice operator might be a reason why this concept has been studied relatively little. The extension of Choiceless Polynomial Time with witnessed symmetric choice [85] mentioned earlier merges the symmetric choice approach with the paradigm of choiceless computation, which we introduce further below.

Linear-algebraic operators The second approach for extending FPC is based on linear algebra. As mentioned above, the famous counterexample separating FPC from PTIME can be seen as a linear equation system over a finite field. This called for adding suitable linear algebraic operators to FPC. Dawar, Grohe, Holm and Laubner introduced *rank logic* [36], which is the extension of FPC by an operator that evaluates to the rank of definable matrices over fields of different characteristics. Naturally, this logic can define the solvability of a given linear equation system over a finite field, and thereby, it can also distinguish CFI-graphs (over a fixed finite field). The original definition of rank logic was too weak, though, because it did not allow to dynamically choose in a formula the field over which a matrix is to be interpreted. Grädel and Pakusa [55] used Holm's CFI-structures over fields of different prime characteristics [73] to show that the original version of rank logic still fails to distinguish these if the characteristic is not fixed but varies with the input. At the same time, they proposed a straightforward solution for this problem by allowing a single rank logic formula to work over different prime characteristics – this is possible by letting the characteristic be a parameter of the rank operator. This generalised version of rank logic only survived a few years until Moritz Lichter was able to separate it from PTIME (actually even from *Choiceless Polynomial Time*) [82] as well. He considered an even more general CFI-construction that encodes linear equation systems over the finite rings $\mathbb{Z}_{2^i}$, where i varies with the input. Thus, rank logic is suited for solving linear equation systems over fields, but not over rings. Lichter's proof is via the construction of a winning strategy for Duplicator in the *invertible map* game [38], which captures the power of logics with linear algebraic operators: In [35], Dawar, Grädel and Pakusa defined an infinitary logic enriched with *all* possible isomorphism-invariant linear algebraic operators. This logic is not really natural but its expressive power is an upper bound for all conceivable linear-algebraic fixed-point logics, in particular also rank logic. Any two graphs for which Duplicator has a winning strategy in the invertible map game are indistinguishable in this "universal" linear-algebraic logic. Therefore, Lichter's construction not only rules out rank logic as a candidate for polynomial time, but in fact even the said linear-algebraic logic, in which all possible isomorphism-invariant linear-algebraic operators are available [37] . The linear-algebraic approach towards a logic for PTIME could hence be called a dead end. However, it might be that more general operators for dealing with equation systems over finite rings or perhaps even groups could be added to fixed-point logic in order to make further progress. Following this route to the end leads to the question if it is possible to define a relatively natural logic in which every constraint satisfaction problem in PTIME can be solved. One approach towards this would be to try and implement Bulatov's [27] or Zhuk's [105] polynomial time algorithm for CSPs in a symmetry-invariant way. At the moment, though, we know of no successful attempts in that direction.

Choiceless computation The third route towards capturing polynomial time is the main subject of this thesis. The logic *Choiceless Polynomial Time* with counting (CPT) can be thought of in two different ways: It is in a sense an extension of fixed-point logic by more complex data structures than just fixed-arity relations. Alternatively, one can also view CPT as a symmetry-invariant restriction of polynomial time Turing machines. This is the spirit in which it was first defined by Andreas Blass, Yuri Gurevich, and Saharon Shelah in 1999 [21]. They introduced CPT as an abstract state machine model. These machines are similar to polynomial time Turing machines but they receive as input a finite structure (not its representation as a string) and instead of symbols on a tape, they manipulate *hereditarily finite sets* with atoms from the universe of the structure. A hereditarily finite set (h.f. set, in short) is an arbitrarily nested finite set, like for example $\{a, b, \{c, d, \{a\}\}\}$, where a, b, c, d are elements of the input structure. The important difference to Turing machines is that the instructions of CPT enforce symmetry-invariance of the computation steps. So for instance, suppose the input structure $\mathfrak{A}$ has an automorphism that swaps a and c, and b and d. Then any CPT program (since CPT can be viewed both as a logic and a machine model, its sentences are often called programs or algorithms) that computes the set mentioned above also computes its automorphic image $\{c, d, \{a, b, \{c\}\}\}$ and stores it in the memory. Besides this symmetry-invariance, CPT programs are also restricted in the sense that the length of their runs and the total space used for the computed h.f. sets is always polynomially bounded in the size of the input structure (just like with classical PTIME-Turing machines). Blass, Gurevich, and Shelah considered two versions of CPT, one without and one with counting. The version with counting has an operator for defining the cardinality of a given set. In [21], the authors immediately separated CPT *without* counting from PTIME by showing that it cannot define the EVEN-query on structures without any relations, i.e. naked sets. So the cardinality operator is indeed necessary to have access to the size of sets in CPT. The inexpressibility result is based on a symmetry argument. Sets without relations are among the most symmetric structures possible because all $n!$ permutations of the universe are automorphisms. On such symmetric structures, every "too complex" h.f. set has a superpolynomial number of automorphic images and can therefore not be used in a CPT computation (all these automorphic images would also be computed and stored, and this is not possible in polynomial time). The second step is to show that defining the size of the input structure requires the computation of such a complex set. Of course, one has to make precise what a "too complex" h.f. set is. In this context, these are sets whose *minimum support* has super-constant size. We explain this in detail in Section 4.2; a support of a h.f. set x over a universe A is a subset $S \subseteq A$ such that every automorphism that fixes S pointwise also fixes x. The sets from the example above have the support $\{a, b, c, d\}$, and if the automorphism group of $\mathfrak{A}$ is the full symmetric group (and $|A| > 4$), then this is also the smallest possible support. So in short, on very symmetric structures, CPT can essentially only compute sets that contain a constant number of atoms. One can show using a pebble game argument that counting the size of the input structure requires sets with greater support. Alternatively, the inability of CPT to count can be seen from the fact that CPT without counting has a zero-one law, which was first proved by Shelah [101] and developed further by Blass and Gurevich [18].

So the most interesting variant of CPT is that with the counting operator. For this logic, it has been open for more than 20 years now whether it captures PTIME or not. Only few negative results concerning the power of CPT are known. Basically, there is the aforementioned support-argument separating CPT without counting from P, and a later variation of this technique by Benjamin Rossman [95], showing that CPT (with counting) cannot define the set of all hyperplanes in a given finite vector space. This is, however, a functional problem and not a *decision problem* in polynomial time. Proving limitations of CPT with regards to decision problems seems to be rather difficult. One thing we do know is that the nesting of sets indeed is a source of expressive power: There is a super-constant lower bound on the depth (or *rank*) of the h.f. sets that have to be computed by a CPT-program in order to distinguish certain CFI graphs (by Dawar, Richerby, and Rossman [40]). This rank lower bound follows from a lower bound on the minimum support size of a h.f. set that has to be computed (we will also build on this support lower bound for some of our negative results). Apart from that, the question for CPT lower bounds of any kind is wide open. It is therefore the main objective of this thesis to prove further insightful non-definability results and identify new potential approaches towards strong lower bounds or even the separation of P and CPT.

Research on positive results regarding the expressiveness has been somewhat more successful, although arguably, none of them provide strong evidence that CPT captures PTIME on *all* structures. The main theme controlling the expressive power of CPT seems to be the "amount of order" in the input structure: All important positive results concern classes of almost totally ordered structures. More precisely, these structures come with a *preorder* on their universe. A preorder is a partial order which is a total order on the set of *colour classes*; the term "colour class" refers to a set of pairwise incomparable elements in the preorder. The power of CPT on preordered structures depends on the properties of the colour classes, for all we know at the moment. The first expressiveness result that exploited such a preorder was the definability of the CFI-query (i.e. the problem to distinguish two given non-isomorphic CFI-graphs) with a preorder that orders the graphs up to their so-called CFI-gadgets [40]. This variant of the CFI-query is not definable in FPC (nor in rank logic if one excludes rank operators of characteristic two [55]). The CPT-algorithm developed by Dawar, Richerby, and Rossman in [40] crucially relies on the preorder in the CFI-graphs and could not directly be implemented in CPT without it. It is centred around the construction of a certain deeply nested h.f. object whose structure encodes the information needed to distinguish non-isomorphic CFI-graphs. In Chapter 6, we review their construction in detail because it is one of the main starting points for our own research. Later, Pakusa, Schalthöfer, and Selman took this algorithmic technique further and managed to define the CFI-query in CPT on preordered CFI-graphs with larger colour classes (containing a logarithmic number of gadgets each) [91]. Note that the larger the colour classes are, the more difficult it is to perform polynomial time choiceless computations because larger colour classes mean more symmetries – and generally, the amount of computational resources needed grows with the "amount of symmetry" in the input structure.
Preorders can be exploited by CPT not only to distinguish CFI-graphs, but also to capture polynomial time: Abu Zaid, Grädel, Grohe, and Pakusa showed in [4] that CPT captures PTIME on all preordered structures with constant colour class size where the automorphims group of each individual colour class is Abelian. This last condition may sound oddly specific but it comes from the fact that many typical benchmark structures, i.e. CFI-graphs and also the so-called *multipedes* [67], possess such Abelian colour classes. Both preordered CFI-graphs and multipedes are indistinguishable in FPC, but have an isomorphism problem in PTIME – therefore, the fact that CPT captures PTIME on these structures demonstrates once more that CPT is strictly more expressive than FPC.
Later on, this capturing result was slightly generalised by Lichter and Schweitzer to colour classes with dihedral automorphism groups [84]. This required considerable extra technical effort and according to personal communication with one of the authors, this

is more or less the end of the story. That is, there seems to be little hope to capture PTIME on preordered structures with less restricted colour classes – or this will become technically so tedious that it is not really worth the effort. So instead of continuing this route, we take the opposite approach and study the power of CPT on extremely symmetric structures (that have in particular no preorder at all) with the intention of proving non-definability results.

For completeness, we should also mention another aspect of structures – besides the existence of certain preorders – that makes CPT powerful on them, and this is *padding*. It is not very difficult to show that if one extends a given structure $\mathfrak{A}$ with a sufficient number of irrelevant elements, say $|A|!$, and marks these elements as padding with a special relation symbol, then CPT can perform any polynomial time computation on the relevant part of the structure: It is possible to recognise the true structure $\mathfrak{A}$ within the padded structure and then to compute the set of all linear orders of $\mathfrak{A}$. This does not violate any polynomial bounds because the padded structure is big enough. Then using these linear orders and the Immerman-Vardi theorem, we can carry out any polynomial time computation that we like, as already described earlier in this introduction (see also [22, 80]). This padding argument already suffices to show that CPT is strictly stronger than FPC because the power of FPC does not directly depend on the input size. Moreover, Duplicator's winning strategies in the pebble game that characterises the power of FPC are unaffected by padding in the structure. While this padding argument is a relatively simple way of separating CPT from the weaker fixed-point logic FPC, one can argue that it is rather superficial: The fact that CPT does benefit from padding is true just because the resources that any given CPT program may use depend directly on the structure size. However, this argument gives little insight into what is actually responsible for the power and limitations of symmetric computation with h.f. sets. A more extensive survey on CPT, including variants and alternative definitions, can be found in Chapter 3.

1.2 Contributions

Non-definability of preorders in hypercubes We focus almost exclusively on the limitations of Choiceless Polynomial Time or choiceless computation with h.f. sets in general. As outlined above, there is a series of positive results regarding the expressiveness of CPT on preordered structures. The first question we address is how much one can actually learn from these results for the unordered case. In other words, are suitably preordered structures just a very special class on which CPT happens to be particularly powerful, or can the unordered setting somehow be reduced to the preordered case? In Chapter 7, we answer this question by showing that preorders with sufficiently small colour classes are not CPT-definable in unordered structures in general. Therefore, being able to solve hard problems on preordered structures is not directly an indication that CPT is equally powerful on unordered structures. Concretely, what we study is the CFI-query on unordered n-dimensional hypercubes. The construction by Cai, Fürer and Immerman can be applied to any connected base graph in order to produce pairs of non-isomorphic graphs that are hard to distinguish by logics. So in particular, one can also define CFI-graphs on hypercubes. The motivation for studying these is that hypercubes are extremely symmetric; the automorphism group of the n-dimensional hypercube contains the symmetric group $\mathbf{Sym}_n$, albeit acting on the subsets of $[n]$ instead of the points. From a group-theoretic perspective, $\mathbf{Sym}_n$ is essentially the largest possible symmetry group because all other permutation groups are subgroups of $\mathbf{Sym}_n$, for some n. Apart from in the n-dimensional hypercube, $\mathbf{Sym}_n$ also appears as the automorphism group of the complete graph on n vertices (then acting on the points, not on subsets of $[n]$). Thus, CFI-graphs on complete graphs would be the more natural highly symmetric example. But it is known that CFI-graphs over graphs with vertices of linear degree can be distinguished in CPT [91]. This is basically due to a peculiarity of the CFI-construction, which increases the structure size exponentially in the degree, so the CFI-graphs over complete graphs are implicitly padded. Hence, CPT computations have more resources available on them, and therefore, CFI-graphs over complete graphs are not hard instances for CPT. The degree of hypercubes is only logarithmic in the structure size, which is why no implicit padding occurs when the CFI-construction is applied to them. So hypercube CFI-structures have the benefit that the full symmetric group $\mathbf{Sym}_n$ appears in the automorphism group, and at the same time, there is no padding that might "help" CPT. Indeed, our combinatorial result in Chapter 7 says that the n-dimensional hypercube $\mathcal{H}_n$ is so symmetric that any ordered partition of its vertex set into colour classes of logarithmic size has more than polynomially many (measured w.r.t. $|\mathcal{H}_n| = 2^n$) automorphic images. From this it follows with the automorphism-invariance of CPT that no CPT-program can compute any preorder with log-sized colour classes in $\mathcal{H}_n$. The strongest known CPT-algorithm (i.e. requiring the least preconditions) for distinguishing CFI-graphs is the one from [91] for preordered graphs with log-sized colour classes mentioned earlier. Our result shows that this algorithm cannot be applied to CFI-graphs over unordered n-dimensional hypercubes because the preorder that is necessary for the algorithm to work is itself not CPT-definable. Actually, one can check that the algorithm constructs a h.f. object which encodes such a preorder, so even if it were possible to implement this algorithm in such a way that it works on unordered structures, it could not succeed on hypercubes because it would implicitly define a preorder which we have proven to be undefinable.

So we can conclude that for this algorithm, it is essential that the preorder is supplied as part of the input. The result of Chapter 7 therefore establishes the CFI-query over unordered n-dimensional hypercubes as a potential candidate problem that might not be CPT-definable. This would separate CPT from PTIME, because, as already said, all kinds of CFI-graphs can be distinguished in polynomial time. Moreover, this also suggests that we should not be overly optimistic that CPT captures PTIME on *all* structures, just because it does so on certain classes of preordered structures with small colour classes. The content of Chapter 7 also appeared at CSL 2021 [87].

Symmetric XOR-circuits As a next step, we ask the question if there is a general class of combinatorial objects whose CPT-definability is a necessary condition for the definability of the CFI-query. For the specific preorder-based algorithm, this object was a preorder with log-sized colour classes. But the non-definability of these preorders in hypercubes does not mean that there is no CPT-algorithm at all for the CFI-query on these graphs. So it seems natural to search for a broader class of objects that characterise the power of more general CPT-algorithms for the CFI-query (i.e. hypothetical algorithms which are not yet known). Ideally, these should be objects which any conceivable CPT-algorithm for the CFI-query must implicitly define. If we find such a class of objects, this gives us a point of attack that we can use to show the non-definability of the CFI-query: Namely, then it suffices to show that these objects are not CPT-definable, just as the non-definability of preorders with log-sized colour classes was enough to rule out one particular algorithm.

It turns out that there is indeed such a class of "universal objects" that characterise a large class of choiceless algorithms for the CFI-query in the following sense. An algorithm from that class can only solve the CFI-query on a family $(\mathfrak{G}_n)_{n\in\mathbb{N}}$ of CFI-graphs over base graphs $(G_n)_{n\in\mathbb{N}}$ if there exists a corresponding family of symmetric XOR-circuits $(C_n)_{n\in\mathbb{N}}$. The input gates of each C_n are labelled with the edges of G_n, the internal gates are all XOR-gates, the size of C_n is polynomial in the size of $\mathfrak{G}_n$, the circuits satisfy a certain logarithmic bound on the fan-in of the gates, and they compute the XOR over a large number of their input bits (this is not trivially satisfied because the input gates may cancel themselves out in the circuit). Moreover, the circuits C_n are symmetric with respect to the automorphisms of G_n acting on the input gates. What we show in Chapter 8 is that only on families of CFI-structures, for which such symmetric circuits exist, it is possible to decide the CFI-query with a choiceless algorithm that is *CFI-symmetric* (see Theorem 8.0.1). The class of *CFI-symmetric* algorithms is one that we define; basically, it contains all CPT-algorithms which construct a h.f. set based on the technique developed by Dawar, Richerby, and Rossman in [40] (we will make this precise but the definition is too technical for now). All known CPT-algorithms for the CFI-query so far use this technique and hence belong to the class of algorithms whose power depends on the existence of the said symmetric circuit families. Currently, this technique is the only known design pattern for choiceless CFI-algorithms. Thus, lower bounds against this class of algorithms would mean that either CPT $\neq$ PTIME, or that we have to find a totally new construction principle for h.f. sets in order to solve the CFI-query on unordered graphs. The reason why these CFI-symmetric algorithms can only work on graphs for which there exist symmetric XOR-circuits is because the power of these algorithms is due to the properties of a specific h.f. set that they compute – and this h.f. set can quite naturally be viewed as an XOR-circuit with all the properties mentioned above.

We also study in Chapter 8 in how far the connection to symmetric XOR-circuits can be generalised to *all* possible CPT-algorithms for the CFI-query, rather than only the ones we call CFI-symmetric. This works out to an extent but is technically quite involved. What we find is that, generally, any h.f. set x over a CFI-graph can be translated into an XOR-circuit, as long as certain Boolean vector spaces appearing as stabiliser groups of the sub-objects of x admit a symmetric basis. It seems, however, not easy to understand which h.f. sets have this symmetric-basis property and which do not. Therefore, without putting in any further technical effort, we only have the connection to XOR-circuits for *CFI-symmetric* algorithms. We believe that this circuit-approach can be a fruitful method to prove lower bounds against CPT: The limitations of symmetric circuits have been and continue to be studied in different contexts in the literature. For example, there are works by Anderson and Dawar, and Dawar and Wilsenach on the connections between symmetric circuits and fixed-point logic with counting [6] and rank logic [42], and also on lower bounds for symmetric arithmetic circuits for the permanent [41] and determinant [43]. Furthermore, lower bounds for symmetric Boolean circuits computing the parity function or the product of permutation matrices have been shown by Rossman (and He) [96, 70].

We hope that future research will also lead to sufficiently strong lower bounds against the symmetric XOR-circuits we consider here. If we could disprove their existence for a given family of CFI-structures, then this would show that the CFI-query on these structures is undefinable in CPT, at least by CFI-symmetric algorithms.

Lower bounds against symmetric XOR-circuits over hypercubes Currently, we are not able to show for any family of CFI-graphs that the necessary symmetric XOR-circuits do not exist. However, in Chapter 9, we achieve this at least for a class of circuits with stronger restrictions. Again, we study the CFI-query over unordered hypercubes because this seems to be particularly hard for CPT, according to our results from Chapter 7. We know that if this version of the CFI-query can be solved in CPT by a CFI-symmetric algorithm, then there must exist a family $(C_n)_{n \in \mathbb{N}}$ of XOR-circuits such that the input gates of C_n are labelled with edges of the n-dimensional hypercube $\mathcal{H}_n$, and such that C_n is sufficiently symmetric with respect to the hypercube-automorphisms (i.e. $\mathbf{Sym}_n$) acting on the input gates. Additionally, they must satisfy the other properties mentioned in Theorem 9.1.5. In Chapter 9, we impose further restrictions on the circuits, for example that the number of parents and children of each gate (per orbit) be at most $\mathcal{O}(n)$, where n is the dimension of the hypercube, so this is logarithmic in the structure size. With these additional restrictions, we can indeed prove that such circuit families do not exist, or more precisely: Whenever a family of symmetric circuits satisfies these stronger properties, then most of their input bits cancel themselves out in the circuit, and thus, the output is not sensitive to as many input bits as required by Theorem 9.1.5. Intuitively, the idea is that very symmetric XOR-circuits are highly DAG-like (not tree-like), and have so many interconnections that many bits arrive at the output XOR-gate an even number of times, which leads to cancellation. In tree-like circuits, this problem could not arise because in trees, there is always a unique path from each input gate to the output gate at the root. However, trees seem to be too asymmetric. Thus, informally speaking, there appears to be a trade-off between symmetry of the circuit and avoiding self-cancellation of input bits – achieving both at the same time may be hard or even impossible. In the proof of our result, we use combinatorial and group-theoretic techniques, some of which come from Anderson and Dawar's analysis of symmetric circuits for FPC [6]. The main difficulty which prevented us from using all their techniques straight-away is that they study circuits which are symmetric with respect to $\mathbf{Sym}_n$ and have polynomial size in n – our circuits are also $\mathbf{Sym}_n$-symmetric but have size polynomial in 2^n, which is the size of the n-dimensional hypercube. This makes it more difficult to prove limitations on the circuits because it is of course easier for a circuit to be symmetric and exponentially large in n than being symmetric and of size polynomial in n. The main results of Chapters 8 and 9 are going to appear at MFCS 2023 [**?**].

Unfortunately, there is still some gap between the more restricted circuits whose existence we can disprove and the circuits whose existence is necessary for the choiceless definability of the CFI-query. Therefore, we cannot definitely say at the moment that the CFI-query over hypercubes is not definable by a CFI-symmetric algorithm. However, we conjecture that this is the case because it seems like our "symmetry vs. self-cancellation"-argument for these XOR-circuits still has some more potential that we have not completely used yet. Perhaps it is possible to make this argument even without imposing the extra restrictions on the circuits but this would probably require a much more technically involved analysis and maybe also some more techniques that we have not found yet.

In total, our line of research on symmetric XOR-circuits and the connection to the choiceless definability of the CFI-query leads to a conclusion that we might formulate as follows: If Choiceless Polynomial Time does capture PTIME, then at least one of the following two must be the case. Either, there exists a class of powerful CPT-algorithms for the CFI-query which are *not* CFI-symmetric, i.e. which construct some deeply nested set with large support according to a design principle that is currently not known (and probably less natural than the CFI-symmetric h.f. sets). Or there do exist highly symmetric XOR-circuits which avoid self-cancellation of too many input bits. Confirming either of these two seems to be hard and is probably not easier than trying to prove the impossibility of these two options.

Finally, we should remark that all the results we have mentioned so far are not very CPT-specific; they would in fact apply to (almost) *any* symmetry-invariant computation model working with h.f. sets. In particular, they would apply to *Choiceless Polynomial Space*, if one were to define this as the generalisation of CPT where the polynomial bound only restricts the size of the computed sets and not the running time. Also, they would apply to hypothetical non-deterministic versions of CPT where the h.f. sets are not constructed deterministically but could be guessed, as long as they are of polynomial size and symmetric. This is because our methods are just concerned with the existence or non-existence of certain symmetric objects, and we never actually think about their computability. Hence, if CPT can one day be separated from PTIME using our approach, then this would essentially separate *all* kinds of h.f.-set-based logics from P, thus showing that this entire route towards capturing PTIME cannot succeed. On the other hand, if it turns out that our methods are not strong enough to separate CPT from PTIME, then this could simply mean that they were too coarse. It might then still be that even though the required symmetric objects do exist, they are simply not computable in deterministic polynomial time – this would call for more sophisticated lower bound techniques.

Lower bounds for CPT via propositional proof complexity The last part of this thesis, Chapter 10 explores a different route towards CPT lower bounds. This chapter is based on a paper that appeared at CSL 2023 [88]. Instead of going via symmetric circuit complexity, we now compare CPT with a propositional proof system called the bounded-degree *extended polynomial calculus.* The polynomial calculus is a proof system for proving the unsatisfiability of polynomial equation systems with respect to $\{0,1\}$-assignments to the variables. From an initial set $\mathcal{P}$ of polynomial equations one can derive others which are implied by them. The proof rules of the polynomial calculus allow to form linear combinations of equations and to multiply them with variables. The system $\mathcal{P}$ is unsatisfiable if and only if the equation $1 = 0$ is derivable from it using these rules. In the *extended* polynomial calculus, one can additionally introduce *extension variables* as abbreviations for more complex polynomials in a derivation. This may in general allow for shorter proofs and is quite a powerful extension. In the *bounded-degree* extended polynomial calculus, there is a constant bound on the degree of the polynomials that may occur in a proof (in our case, this bound is three). In the non-extended polynomial calculus, bounding the degree of the polynomials appearing in a proof by a constant is a restriction commonly used to make proof search tractable. If the degree is at most k, and the number of variables is n, then at most $\approx n^{k+1}$ monomials can appear in the polynomials. The set of derivable polynomials then forms an n^{k+1}-dimensional vector space, and one can check in polynomial time whether the 1-polynomial is in this space. Therefore, the bounded-degree polynomial calculus is a natural proof system from the practical point of view. However, in contrast to the full polynomial calculus, it is incomplete – not every unsatisfiable polynomial equation system $\mathcal{P}$ admits a derivation of $1 = 0$ that only uses constant-degree polynomials. For the *extended* polynomial calculus, it is not known whether restricting the degree leads to incompleteness or has any computational advantages.

The polynomial calculus and its extensions can also be seen as a method for deciding graph isomorphism: For two given graphs G and H, it is not difficult to encode the existence of an isomorphism between them as a polynomial equation system whose $\{0,1\}$-solutions correspond exactly to the isomorphisms. Denote such an equation system as $P_{\text{iso}}(G,H)$. Then one can ask what the graph distinguishing power of, say, the degree-k polynomial calculus is, i.e. for which pairs of graphs it can prove the unsatisfiability of $P_{\text{iso}}(G,H)$. Berkholz and Grohe showed that the graphs distinguishable in degree-k PC are precisely the ones that are distinguishable with the well-known k-dimensional Weisfeiler Leman algorithm [17]. Building on this (and on the Deep Wesifeiler Leman framework [63]), we prove that whenever two graphs G and H are distinguishable in *Choiceless Polynomial Time* within a given polynomial resource bound $p(n)$, then $P_{\text{iso}}(G,H)$ has a degree-3 extended polynomial calculus (EPC_3) refutation of size $\approx p(n)$.

Moreover, this refutation uses extension variables only for polynomials that are either monomials of degree two or averaged sums of variables. Basically, the EPC$_3$-refutation of $P_{\text{iso}}(G, H)$ simulates the computation steps of the CPT-program that distinguishes G and H.
As a consequence, super-polynomial lower bounds for graph isomorphism on a graph class $\mathcal{K}$ in the degree-3 extended polynomial calculus (with this limited use of of extension variables) would imply that the graphs in $\mathcal{K}$ are not CPT-distinguishable. Here we mean CPT-distinguishability in quite a general sense: Namely, we say that CPT distinguishes all graphs in $\mathcal{K}$ if for every pair of non-isomorphic graphs, there exists some CPT-sentence with a fixed bound $p(n)$ that distinguishes them – this does not have to be the same sentence for all graphs in $\mathcal{K}$. Thus, lower bounds for the extended polynomial calculus would imply that one cannot distinguish all graphs in $\mathcal{K}$ even if one were allowed to guess the CPT-program that is to be executed, depending on the concrete instance (G, H).

Since the promising candidate problems for separating CPT from P are variants of the graph isomorphism problem which are in P (i.e. CFI-graphs or multipedes), a super-polynomial lower bound for EPC$_3$ for distinguishing such graphs would separate CPT from PTIME. Of course, the question is if such a lower bound in proof complexity is any easier to obtain than for CPT directly. It may be that in the area of proof complexity, successful methods will be developed that are not obvious from the perspective of finite model theory, so in this sense, the detour via proof complexity could be beneficial. There is indeed a recent lower bound for the extended polynomial calculus by Yaroslav Alekseev [5]. This is for the so-called binary-value principle, which unfortunately does not seem to be very related to graph isomorphism. The lower bound is also on the *bit-complexity* of the coefficients that must occur in any proof of the binary-value principle. This bit-complexity does not really play a role in our simulation of CPT by EPC$_3$, so it is probable that Alekseev's method cannot be used to obtain the desired lower bounds for graph isomorphism, which would help us to prove limitations of CPT. Hopefully, though, stronger techniques will be found in the future.

Our simulation of CPT in EPC$_3$ is naturally symmetric in some sense. Namely, in the resulting refutation of $P_{\text{iso}}(G, H)$, the set of extension variables that are used is closed under the automorphisms of G and H. So actually, a lower bound for such symmetric refutations would suffice to infer lower bounds for CPT. We intend to study the role of symmetry in proof complexity more systematically in the future, and our results from Chapter 10 might be a good starting point. It can be expected that a suitably defined symmetric proof system will admit similar lower bound techniques as logics such as CPT. We leave this as a topic for future work.

Organisation of the thesis Most of our contributions can be found in the **Chapters 7 to 10**. The necessary preliminaries from finite model theory are presented in **Chapter 2**. **Chapter 3** is an introduction to our object of study, the logic *Choiceless Polynomial Time.* There, we also review most of the research on CPT that has been carried out prior

to or simultaneously with the work on this thesis. **Chapter 4** is devoted to the relevant background from the theory of permutation groups, which is central for our symmetry-based lower bound techniques. In this chapter, we also prove a stronger version of a result from the literature [45] stating that every sufficiently large permutation group contains a large alternating group. This is used later in **Chapter 9**. After the group-theoretic part, we review the well-known *Cai-Fürer-Immerman construction* in **Chapter 5**. Specifically, we focus on *unordered* CFI-graphs and their symmetry groups, which have not received as much attention in the literature as ordered or preordered CFI-graphs so far. We also introduce CFI-graphs over unordered *n-dimensional hypercubes*, which we suggest as a candidate for separating CPT from PTIME. Finally, before we start with our own contributions, we review in **Chapter 6** the most important results and techniques from [40] by Dawar, Richerby and Rossman, and some extensions of this in [91] by Pakusa, Schalthöfer, and Selman. Concretely, we present the key ideas behind two choiceless algorithms for different variants of the preordered CFI-query, and a lower bound on the *support* size of a h.f. set that has to be activated by any CPT-program deciding the CFI-query.
Our first contribution is the non-definability of preorders with small colour classes in hypercubes, which we present in **Chapter 7**. This shows that the CFI-algorithms from [40] and [91] for ordered and preordered CFI-instances cannot be applied in the unordered case. In **Chapter 8**, we extract from these algorithms a structural design pattern for h.f. sets that characterises a more general class of choiceless algorithms for the CFI-query, and then show that the success of such algorithms depends on the existence of the aforementioned symmetric XOR-circuit families. The support lower bound from [40] is used in **Chapters 8** and **9**, where we naturally translate it into a property of the said circuits (sensitivity to a large number of input bits) and prove that under strict assumptions on the circuits (high symmetry and certain fan-in and fan-out bounds), this support/sensitivity lower bound cannot be met.
Finally, **Chapter 10** deals with the relationship between CPT and the extended polynomial calculus proof system, showing that super-polynomial lower bounds for distinguishing CFI-graphs or multipedes in this proof system would separate CPT from P.
The **Chapters 7** and **10** are based on [87] and [88]. The main contents of **Chapters 8** and **9** are summarised in [**?**] and are going to appear in August 2023.

1.3 Acknowledgements

First of all, I thank my supervisor Erich Grädel for his advice and support throughout this project. I especially enjoyed the atmosphere at the chair, which was characterised by a lot of freedom and independence for all PhD students, with respect to both teaching and research. The results of this thesis would not have been obtained, had I been forced to follow a predefined path. Moreover, I thank Erich for his sharing of inspiring maths-related literature such as Enzensberger's *Hommage an Gödel* or Dyson's talk on *Birds and Frogs*.
I would like to thank Anuj Dawar for agreeing to be a reviewer for this thesis and for

valuable impulses and discussions over the course of the years. I am also grateful to Wied Pakusa, who was an excellent supervisor for my Bachelor's thesis and introduced me to descriptive complexity theory in the first place. Without him, the topic of this thesis would probably have been a totally different one. Wied is also responsible for the idea to study CFI-graphs over hypercubes as a hard example for CPT, which was passed on to me by Patrick Landwehr. In this context, I would also like to thank Svenja Schalthöfer for briefing me on the whole subject of choiceless computation and in particular, for explaining potential approaches towards lower bounds to me. I owe special thanks to Daniel Wiebking for the countless discussions on Deep Weisfeiler Leman and computational group theory. Quite a few results in this thesis would not have been possible without Daniel's help. Also, I thank him for hosting our weekly Doppelkopf session – at least, until the pandemic broke out. I would also like to thank Leon Kullmann for spotting a mistake in the definition of the extended polynomial calculus in the non-final version of the thesis.
I am grateful to my colleagues and friends from the institute Richard, Katrin, Matthias and Matthias, and Lovro, for the great teamwork and the fun we had every summer running the course *Mathematische Logik*, and at the office in general. I also extend my thanks to my colleagues from the i7 chair, who contributed to the pleasant atmosphere at work, in particular to Patrick for his regular afternoon visits to everybody's offices. He should also be considered an honorary member of the Grädel group because it was mainly he who brought the group together outside of work.
Finally, I thank my friends and family for their support, and most importantly Monique for everything.

2 Preliminaries

Aber von der Logik läßt sich nicht einmal so viel rühmen. Sie ist nämlich bloß das Wissen in abstracto dessen, was Jeder in concreto weiß. Daher, so wenig als man sie braucht, einem falschen Räsonnement nicht beizustimmen, so wenig ruft man ihre Regeln zu Hülfe, um ein richtiges zu machen. [...] Und wer die Logik zu praktischen Zwecken erlernt, gleicht dem, der einen Bieber zu seinem Bau abrichten will. - Obgleich also ohne praktischen Nutzen, muß nichts destoweniger die Logik beibehalten werden, weil sie philosophisches Interesse hat.

Arthur Schopenhauer, Die Welt als Wille und Vorstellung

2.1 Sets and linear orders

Throughout the thesis, $[n]$ denotes the set $\{1, 2, ..., n\}$. The set of all subsets of a set A is denoted $\mathcal{P}(A)$. The central class of combinatorial objects that we study are *hereditarily finite sets* (also called h.f. *objects*) over a given finite set A of atoms. Usually, the atoms will be the universe of a structure. The set of hereditarily finite objects over A, $\mathrm{HF}(A)$, is defined as $\bigcup_{i\in\mathbb{N}} \mathrm{HF}_i(A)$, where $\mathrm{HF}_0(A) := A \cup \{\emptyset\}, \mathrm{HF}_{i+1}(A) := \mathrm{HF}_i(A) \cup 2^{\mathrm{HF}_i(A)}$. The size of a h.f. set $x \in \mathrm{HF}(a)$ is measured in terms of its *transitive closure* $\mathrm{tc}(x)$: The set $\mathrm{tc}(x)$ is the least transitive set such that $x \in \mathrm{tc}(x)$. Transitivity means that for every $a \in \mathrm{tc}(x)$, $a \subseteq \mathrm{tc}(x)$. Intuitively, one can view $\mathrm{tc}(x)$ as the set of all sets that appear as elements at some nesting depth within x. One can also naturally view any $x \in \mathrm{HF}(A)$ as a DAG with vertex set $V = \mathrm{tc}(x)$ and the element relation $\in$ as the edge relation.
The standard way to encode ordered tuples as h.f. sets is as Kuratowski-tuples, i.e. $(a_1, a_2, ..., a_k)$ is encoded as $\langle a_1, ..., a_k\rangle := \{a_1, \{a_2, \{a_3, \{..., \{a_k\}\}\}\}$. Thus, the operator $\langle\cdot\rangle$ maps tuples to their h.f.-set-representations.

A *linear order* (or total order) $<$ on a set A is as usual a binary relation that is irreflexive and transitive, and for any two distinct $a, b \in A$, it holds either $a < b$ or $b < a$. Often we will also be working with *preordered* sets. A binary relation $\preceq$ is a total preorder on A if it is reflexive and transitive, and any two elements are $\preceq$-comparable. A *colour class* of a preorder is a maximal subset $C \subseteq A$ on which $\preceq$ is symmetric. Intuitively, a colour class is a maximal set of incomparable elements. A total preorder induces a linear order on the set of colour classes, so it can be seen as an approximation to a linear order on A. Whenever we speak of preorders or preordered structures, we think of the preorder as the ordered partition $(C_1, C_2, ..., C_m)$ of A into its colour classes. Note that generally, preorders need not be total but in this thesis, the word “preorder” is to be read as “total preorder”.

2.2 Descriptive complexity theory

For a detailed introduction to the subject, there is a number of useful textbooks: [47, 65, 61, 75, 81, 86]. Here, we only review a few essentials and fix the notation.

A *vocabulary* or *signature* is a finite set $\tau = \{R_1, ..., R_\ell, f_1, ..., f_m\}$ of function and relation symbols, each with a fixed arity r_i or k_j, respectively, for $i \in [\ell], j \in [m]$. A τ-*structure* $\mathfrak{A} = (A, R_1^{\mathfrak{A}}, ..., R_\ell^{\mathfrak{A}}, f_1^{\mathfrak{A}}, ..., f_m^{\mathfrak{A}})$ is a set A, the *universe*, together with interpretations $R_i^{\mathfrak{A}} \subseteq A^{r_i}$ and $f_i^{\mathfrak{A}} : A^{k_i} \longrightarrow A$ for the relation and function symbols. Structures will often be denoted as fraktur letters, or in the case of graphs, simply as G. Graphs are structures with vocabulary $\{E\}$, where E is a binary relation symbol. We sometimes write $V(G)$ and $V(E)$ to denote the universe and the edge relation of a graph. If not stated otherwise, structures are assumed to be *finite*. Most of the time, they will also be relational, i.e. the vocabularies do not contain function symbols. An *isomorphism* between two relational τ-structures $\mathfrak{A}$ and $\mathfrak{B}$ is a bijection π between the universes such that $(a_1, ..., a_k) \in R^{\mathfrak{A}}$ iff $(\pi(a_1), ..., \pi(a_k)) \in R^{\mathfrak{B}}$ for every relation symbol $R \in \tau$. If such an isomorphism exists, we write $\mathfrak{A} \cong \mathfrak{B}$.

We assume familiarity with *first-order logic* (FO) and we briefly introduce some of its important extensions from algorithmic model theory below. When we compare the expressive power of two logics $\mathcal{L}_1$ and $\mathcal{L}_2$, we write $\mathcal{L}_1 \leq \mathcal{L}_2$ if every class of structures that is definable in $\mathcal{L}_1$ is also definable in $\mathcal{L}_2$, and we write $\mathcal{L}_1 \lneq \mathcal{L}_2$ to stress that there is at least one structure class definable in $\mathcal{L}_2$, but not in $\mathcal{L}_1$. If $\mathcal{L}_1$ and $\mathcal{L}_2$ can define exactly the same classes, then we write $\mathcal{L}_1 \equiv \mathcal{L}_2$. A structure class $\mathcal{K}$ over a vocabulary τ is definable in a logic $\mathcal{L}$ if there is an $\mathcal{L}$-sentence that is satisfied precisely by those τ-structures which are also in $\mathcal{K}$.

Capturing complexity classes with logics A classical complexity class $\mathcal{C}$ such as Logspace, Ptime or NP is *captured* by a logic $\mathcal{L}$ on a given class of structures $\mathcal{K}$ if the classes $\mathcal{K}' \subseteq \mathcal{K}$ that are decidable in $\mathcal{C}$ on inputs from $\mathcal{K}$ (suitably encoded as binary strings) are exactly those subclasses of $\mathcal{K}$ that are also definable in $\mathcal{L}$. This is not the whole truth, though: To give the open question "Is there a logic that captures Ptime?" a sensible meaning, one first of all has to make precise what a logic is, and secondly, impose a further condition that is required to capture a complexity class.

The two properties that constitute a logic $\mathcal{L}$ are first of all that $\mathcal{L}$, viewed as a set of sentences, is decidable, and secondly that for any two isomorphic structures $\mathfrak{A}$ and $\mathfrak{B}$, and every $\mathcal{L}$-sentence ψ, it holds $\mathfrak{A} \models \psi$ if and only if $\mathfrak{B} \models \psi$. The second property is commonly referred to as *symmetry-invariance* and is the main feature that distinguishes logics from classical algorithms. The first property rules out "logics" such as order-invariant FO or order-invariant fixed-point logic, which can make use of a linear order on the input structure but have to output an isomorphism-invariant result. These do not count as logics because it is undecidable if a given sentence is order-invariant or not.

A logic $\mathcal{L}$ captures the complexity class PTIME if two directions hold: Firstly, for every class $\mathcal{K}$ of τ-structures that is decidable in polynomial time, there must exist a sentence $\psi \in \mathcal{L}$ with vocabulary τ that is satisfied precisely by the structures in $\mathcal{K}$. Secondly, the data complexity of $\mathcal{L}$ must be in PTIME. This means that for every sentence $\psi \in \mathcal{L}$, there exists a polynomial time Turing machine M which decides, given as input a (binary string encoding of a) structure $\mathfrak{A}$, whether $\mathfrak{A} \models \psi$. Moreover, the machine M should be computable from ψ: That is, there must exist an algorithm which computes, given ψ, a pair (M, p) such that M decides whether $\mathfrak{A} \models \psi$ in time bounded by $p(|A|)$. This condition is called *effectiveness* and is needed to forbid certain artificial logics, whose sentences have no syntactic connection at all to their evaluation programs. For more background on the notion of capturing complexity classes by logics, see Gurevich's original definition [66] or the survey [59].

2.3 Interpretations

Interpretations are the logical analogue of reductions as we know them from complexity theory. Let σ, τ be vocabularies with $\tau = \{R_1, ..., R_m\}$ where each R_i is a relation symbol of arity r_i, and let $\mathcal{L}$ be a logic. An $\mathcal{L}[\sigma, \tau]$-interpretation $\mathcal{I}$ is an $\mathcal{L}$-definable mapping from σ-structures to τ-structures. The elements of the τ-structure are sets of k-tuples in the original σ-structure. The constant k is called the *dimension* of $\mathcal{I}$. Generally, interpretations can also take a tuple of parameters $\overline{z}$:

A k-dimensional $\mathcal{L}$-interpretation (with parameters) is a tuple

$$\mathcal{I}(\overline{z}) = (\varphi_\delta(\overline{x}, \overline{z}), \varphi_\approx(\overline{x}, \overline{y}, \overline{z}), \varphi_{R_1}(\overline{x}_1, ..., \overline{x}_{r_1}), ..., \varphi_{R_m}(\overline{x}_1, ..., \overline{x}_{r_m})),$$

where $\overline{x}, \overline{y}, \overline{x}_i$ are k-tuples of variables, and all formulas have the vocabulary σ. The interpretation $\mathcal{I}(\overline{z})$ defines a partial mapping from the set of σ-structures to the set of τ-structures. For a given σ-structure $\mathfrak{A}$ and an assignment $\overline{z} \mapsto \overline{a}$, we define $\mathfrak{B}$ be as a τ-structure with universe $B := \{\overline{b} \in A^k \mid \mathfrak{A} \models \varphi_\delta(\overline{b}, \overline{a})\}$ and relations $R_i^{\mathfrak{B}} := \{(\overline{b}_1, ..., \overline{b}_{r_i}) \in A^{kr_i} \mid \mathfrak{A} \models \varphi_{R_i}(\overline{b}_1, ..., \overline{b}_{r_i}, \overline{a})\}$, for all $i \in [m]$. From this structure, the "output" $\mathcal{I}(\mathfrak{A}, \overline{z} \mapsto \overline{a})$ is obtained by factoring out the equivalence classes defined by $\varphi_\approx$. Formally, let $\mathcal{E} := \{(\overline{b}_1, \overline{b}_2) \in A^{2k} \mid \mathfrak{A} \models \varphi_\approx(\overline{b}_1, \overline{b}_2, \overline{a})\}$. If $\mathcal{E}$ is not a congruence relation on $\mathfrak{B}$, then $\mathcal{I}(\mathfrak{A}, \overline{z} \mapsto \overline{a})$ is undefined. Else, $\mathcal{I}(\mathfrak{A}, \overline{z} \mapsto \overline{a})$ is defined to be the factor structure $\mathfrak{B}/\mathcal{E}$.

2.4 Fixed-point logic with counting, pebble games, and the Weisfeiler-Leman algorithm

This is a brief introduction to some of the most important extensions of FO that we need in this thesis (besides Choiceless Polynomial Time, which is presented in detail in the next chapter). *Fixed-point logics* can be defined with different kinds of operators, such as

least or inflationary fixed-points, but they all have the same expressive power. Anyway, more relevant for us than plain fixed-point logic is *fixed-point logic with counting* (FPC). The following presentation follows the one in Dawar's survey [34]. We also recommend Otto's monograph [86] for further background information.
Sentences of FPC are evaluated in two-sorted structures. The first sort is the domain of the structure itself and the second sort contains all natural numbers (in alternative definitions, the second sort is assumed to be an initial segment of $\mathbb{N}$ that is as long as the size of the first sort). Accordingly, FPC has two sorts of variables, domain variables $x_1, x_2, ...$ and number variables $\nu_1, \nu_2, ...$ Since the number variables range over all of $\mathbb{N}$, they may only be quantified when they are bounded by a term because otherwise, it would not be possible to evaluate FPC-sentences in finite time. In short, FPC extends FO with counting terms and fixed-point operators. The set of terms and formulas of FPC is defined as follows:

- Numeric variables ν are *counting terms.*
- If φ is a formula and x is a variable, then $\#\varphi$ is a *counting term.* It evaluates to the number of domain elements that satisfy $\varphi(x)$.
- All FO-formulas are also FPC-formulas.
- If t_1 and t_2 are counting terms, then $t_1 = t_2$ and $t_1 < t_2$ are FPC-formulas.
- If φ is a formula, ν is a numeric variable and t is a counting term, then $\exists \nu \leq t \varphi$ is a formula.
- If φ is a formula, X is a second-order variable, $\overline{z}$ is a tuple of variables whose sorts match the sorts of X, and $\overline{t}$ is a tuple of terms with the same sorts as $\overline{z}$, then $[\mathbf{fp}_{X,\overline{z}}\ \varphi](\overline{t})$ is a formula. It evaluates to true if and only if the tuple of elements denoted by $\overline{t}$ is in the *inflationary fixed-point* of the operator defined by φ.

The inflationary fixed-point of φ in $\mathfrak{A}$ is the fixed-point of the sequence $X^0 = \emptyset, ..., X^{i+1} = X^i \cup \{\overline{a} \in A^k \mid \mathfrak{A} \models \varphi(X^i, \overline{a})\}$, where k is the arity of X. Such a fixed-point is always reached after polynomially many steps.

Counting logic and the bijection game The expressive power of FPC can be bounded in terms of the power of $\mathcal{C}^k$. For $k \in \mathbb{N}$, $\mathcal{C}^k$ denotes the k-variable fragment of first-order logic with *counting quantifiers.* The counting quantifiers in this logic are of the form $\exists^i x \varphi(x)$, for every $i \in \mathbb{N}$, expressing that at least i elements of the structure satisfy $\varphi(x)$. Note that such counting quantifiers can be simulated in ordinary FO but this requires more than one variable. Proving inexpressibility results for the logic $\mathcal{C}^k$ is usually done by exhibiting a pair $(\mathfrak{A}, \mathfrak{B})$ of structures such that $\mathfrak{A}$ has a certain property whose inexpressibility we want to show and $\mathfrak{B}$ does not have the property. Then the aim is to prove that $\mathfrak{A}$ and $\mathfrak{B}$ satisfy exactly the same $\mathcal{C}^k$-sentences. If this is the case, $\mathfrak{A}$ and $\mathfrak{B}$ are called *$\mathcal{C}^k$-equivalent*, denoted $\mathfrak{A} \equiv_{\mathcal{C}^k} \mathfrak{B}$. This is a relaxation of the isomorphism-relation. As shown by Immerman and Lander, $\mathcal{C}^k$-equivalent (finite) structures are also indistinguishable in FPC because if the structures are fixed, then any fixed-point sentence can be unravelled into a $\mathcal{C}^k$-formula.

Theorem 2.4.1 ([76]). *For every sentence $\varphi \in$ FPC, there is a k such that if $\mathfrak{A} \equiv_{\mathcal{C}^k} \mathfrak{B}$, then $\mathfrak{A} \models \varphi$ iff $\mathfrak{B} \models \varphi$.*

Therefore, inexpressibility results for FPC can be shown by establishing $\mathcal{C}^k$-equivalence of suitable structures. The most important tool for this is a game-theoretic characterisation of $\mathcal{C}^k$-equivalence: The *bijective k-pebble game* is played on a pair of structures $(\mathfrak{A}, \mathfrak{B})$ by two players, Spoiler and Duplicator. Duplicator has a winning strategy if and only if $\mathfrak{A} \equiv_{\mathcal{C}^k} \mathfrak{B}$ [71]. The game proceeds as follows: A position in the game is a set of pebble-pairs $\pi \subseteq A \times B$ of size at most k. In each round, Spoiler may pick up any number of pebble-pairs and remove them from the board such that in the resulting position π', less than k pebble-pairs remain. Then Duplicator specifies a bijection $f : A \longrightarrow B$ such that for every $(a, b) \in \pi'$, $f(a) = b$. Spoiler now puts down a new pebble on some element $a \in A$ of his choice, and the corresponding pebble in B is placed on $f(a)$. If the resulting set of pebble-pairs does not induce a local isomorphism, then Spoiler wins. Duplicator has a winning strategy if she can enforce to play forever without losing. A position $\pi = \{(a_1, b_1), ..., (a_k, b_k)\}$ is said to induce a local isomorphism if the mapping g that maps each a_i to b_i, for $i \in [k]$, is an isomorphism from the induced substructure of $\mathfrak{A}$ with universe $\{a_1, ..., a_k\}$ into the induced substructure of $\mathfrak{B}$ with universe $\{b_1, ..., b_k\}$.

The positions π from which Duplicator has a winning strategy are given by those tuples that have the same *$\mathcal{C}^k$-type* in $\mathfrak{A}$ and $\mathfrak{B}$. The *$\mathcal{C}^k$-type* of a tuple $\overline{a} \in A^{\leq k}$ is the collection of all $\mathcal{C}^k$-formulas that are satisfied by $\overline{a}$ in $\mathfrak{A}$. It is known that in each finite structure, every $\mathcal{C}^k$-type is definable by a single $\mathcal{C}^k$-formula, so even though a type is an infinite collection of formulas, it is semantically equivalent to one finite $\mathcal{C}^k$-formula, if the structure is fixed [54].

The Weisfeiler-Leman algorithm The expressive power of the logic $\mathcal{C}^k$ on finite structures corresponds exactly to the power of the well-known $(k-1)$-dimensional Weisfeiler-Leman algorithm. This polynomial time algorithm is a typical heuristic for the graph isomorphism problem. On a given graph G, the k-dimensional Weisfeiler-Leman algorithm (k-WL) computes a partition of $V(G)^k$ into colour classes by starting with a single colour class, which is iteratively refined. For details of this refinement procedure, see for example Kiefer's survey [78]. To find out whether two given graphs G and H are isomorphic or not, one can run the k-WL algorithm on each of them and compare the computed colourings. If there is some colour c such that the number of tuples with colour c in G is different from the number of tuples with colour c in H, then $G \not\cong H$. If the colour class sizes all agree, then the graphs might be isomorphic or not, so k-WL is an incomplete graph isomorphism test. Generally, increasing the dimension k will lead to WL distinguishing more graphs but will also increase the running time.

What matters most for us is that the colours computed by k-WL are exactly the $\mathcal{C}^{k+1}$-types. Therefore, for two graphs G and H, and tuples $\overline{a} \in V(G)^k, \overline{b} \in V(H)^k$, it holds:

The tuples $\overline{a}$ and $\overline{b}$ receive the same colour in the k-dimensional WL-colouring computed on G and H, respectively, if and only if $\overline{a}$ and $\overline{b}$ satisfy the same $\mathcal{C}^{k+1}$-formulas in G and H, respectively (see Theorem 2.2 in [78]). As described above, $\overline{a}$ and $\overline{b}$ have the same $\mathcal{C}^{k+1}$-type if and only if Duplicator has a winning strategy in the bijective $(k+1)$-pebble game starting from the position given by $(\overline{a}, \overline{b})$. Thus, k-WL distinguishes the graphs G and H if and only if Spoiler has a winning strategy in the bijective $(k+1)$-pebble game on G and H starting with no pebbles on the board.

Partial fixed-point logic Another fixed-point logic that has connections with Choiceless Polynomial Time is *partial fixed-point logic*, denoted PFP. In this logic, the fixed-point inductions need not be monotone or inflationary, so tuples can be removed again from the fixed-point. This also means that the fixed-point of a PFP-formula in a structure does not always exist. Partial fixed-points are computable in polynomial space. There exists also the restriction $\text{PFP}|_{\text{PTIME}}$, in which the fixed-point inductions may only have polynomial length. Partial fixed-point logic is equivalent to a language of while-programs [1].

2.5 Treewidth

Inexpressibility results for the logic $\mathcal{C}^k$ are often related to the well-known graph parameter *treewidth*. The "standard" way to define structures $(\mathfrak{A}, \mathfrak{B})$, which are non-isomorphic but $\mathcal{C}^k$-equivalent is the so-called Cai-Fürer-Immerman construction [29]; we review this in Chapter 5. The CFI-construction takes some connected undirected base graph G and transforms it into such a pair $(\mathfrak{A}, \mathfrak{B})$ of non-isomorphic $\mathcal{C}^k$-equivalent structures. The parameter k in the $\mathcal{C}^k$-equivalence is equal to the *treewidth* of the base graph G. The treewidth of a graph G is defined as the smallest natural number t such that G has a *tree decomposition* of *width t*. The following is taken from a survey by Bodlaender [24], where more details can be found:
A tree decomposition of $G = (V, E)$ is a tree $T = (I, F)$ each of whose nodes $i \in I$ is associated with a *bag* $X_i \subseteq V$. The tree T and its bags satisfy:

- $\bigcup_{i \in I} X_i = V$,
- for each edge $\{v, w\} \in E$, there exists an $i \in I$ with $\{v, w\} \subseteq X_i$,
- for all $v \in V$, the set $\{i \in I \mid v \in v_i\}$ induces a subtree of T.

The *width* of the tree decomposition is $\max_{i \in I} |X_i| - 1$. Thus, trees have treewidth 1, and the less treelike a graph is, the higher is the treewidth.

Treewidth also has a game characterisation. A graph G has treewidth at most k if the Cops have a winning strategy with $k+1$ cops in the *Cops and Robber game*. This game is played on G by two players, the Robber versus the Cops. The robber is always on a vertex and each cop can either be on a vertex or in a helicopter. The goal of the Cops is to land a cop on the field where the robber stands. The robber can, however, move

at infinite speed along any path that is not blocked by a cop. In particular, he can see where a cop is about to land and can make his move before the landing is completed. The cops move in their helicopters (also at infinite speed).
Essentially, the connection between treewidth and the number of variables needed to distinguish CFI-graphs in counting logic is due to a connection between the games: The Robber's winning strategy in the Cops and Robber game on the base graphs translates to a winning strategy for Spoiler in the bijective pebble game on the CFI-graphs [10].

2.6 Permutation groups

The entire Chapter 4 is devoted to the group-theoretic background and tools that we make extensive use of in our lower bound considerations. Therefore, we will be very brief here. A group is an algebraic structure $(G, \cdot)$ with a binary operation that is associative, has a unique neutral element, and unique inverses. The set of all permutations of some set A is denoted $\mathbf{Sym}(A)$ and has a group structure with respect to the operation $\circ$, the composition of two permutations. The *automorphism group* $\mathbf{Aut}(\mathfrak{A})$ of a structure is the subgroup of $\mathbf{Sym}(A)$ consisting of all permutations which preserve the relations of $\mathfrak{A}$. The *orbit* of an element $a \in A$ with respect to the group $\mathbf{Aut}(\mathfrak{A})$ is the set $\{\pi(a) \mid \pi \in \mathbf{Aut}(\mathfrak{A})\}$ of possible images of a. Similarly, one can define the orbits of tuples, sets, and generally, elements of $\text{HF}(A)$. In the language of groups, the symmetry-invariance of logics means that any set or relation or h.f. set that is defined by a logical formula in a structure $\mathfrak{A}$ is closed under orbits.

We will frequently estimate the asymptotic sizes of certain permutation groups. Let $n = |A|$. Then $|\mathbf{Sym}(A)| = n!$. Asymptotically, factorials can be approximated with the *Stirling approximation* $n! \approx \sqrt{2\pi n} \cdot \left(\frac{n}{e}\right)^n$. At this point, we also recall the standard notation for the asymptotic behaviour of functions. Let $f(n), g(n) : \mathbb{N} \longrightarrow \mathbb{N}$ be monotonously increasing functions. Then $f(n) \in o(g(n))$ if $\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$. Conversely, $f(n) \in \omega(g(n))$ if $\lim_{n \to \infty} \frac{f(n)}{g(n)} = \infty$. We have $f(n) \in \Theta(g(n))$ if $\lim_{n \to \infty} \frac{f(n)}{g(n)}$ is some non-zero constant. Finally, $f(n) \in \mathcal{O}(g(n))$ if $f(n) \in o(n)$ or $f(n) \in \Theta(g(n))$, and we write $f(n) \in \Omega(g(n))$ if $f(n) \in \omega(n)$ or $f(n) \in \Theta(g(n))$.

2.7 Linear algebra

We denote by $\mathbb{F}_2$ the finite field with two elements $\{0, 1\}$, and by $\mathbb{F}_2^n$ the set of n-tuples over $\mathbb{F}_2$, viewed as an n-dimensional vector space. We will also be dealing with spaces indexed by some finite (unordered) set J, in which case we write $\mathbb{F}_2^J$ for the $|J|$-dimensional vector space whose coordinates are the elements of J. Similarly, the rows and columns of a matrix can be indexed with such finite unordered sets, so a matrix $M \in \mathbb{F}_2^{I \times J}$ describes a linear transformation $M : \mathbb{F}_2^J \longrightarrow \mathbb{F}_2^I$. The *image* of M is $\mathbf{Im}(M) = \{M \cdot \mathbf{v} \mid \mathbf{v} \in \mathbb{F}_2^J\}$, and the *kernel* is $\mathbf{Ker}(M) = \{\mathbf{v} \in \mathbb{F}_2^J \mid M \cdot \mathbf{v} = \mathbf{0}\}$. The *Rank Theorem* states that $|J| = \mathbf{rk}(M) + \dim(\mathbf{Ker}(M))$. The *rank* $\mathbf{rk}(M)$ denotes the dimension of $\mathbf{Im}(M)$. It is

equal both to the dimension of the space spanned by the column vectors of M and the dimension of the space spanned by the row vectors.
The notion of a *kernel* also appears in connection with *group homomorphisms* $h : G \longrightarrow H$. The kernel of h is the subgroup of G that is mapped to the neutral element of H.

3 Choiceless Polynomial Time

Choiceless Polynomial Time was introduced in 1999 by Blass, Gurevich, and Shelah [21]. The authors devised it as an abstract state machine model, rather than a typical logic, so the original syntax of CPT looks like a programming language. These abstract state machines operate on finite structures, which are expanded by so-called dynamic functions. The dynamic functions assign to each k-tuple in the structure a hereditarily finite set; as the name suggests, these functions can be modified by the abstract state machine and so they serve as the "work tape" of the machine. The update operations of the dynamic functions are by design isomorphism-invariant, and time and space of the computation are restricted polynomially in the size of the input structure. However, the abstract state machine presentation of CPT has a lot of overhead that makes it unnecessarily complicated to use and obfuscates a bit what this model really is: A fixed-point logic with h.f. sets. Therefore, in 2010, Rossman proposed a leaner and more logic-like definition of CPT, called BGS-*logic* (after the inventors of CPT) [95]. We present here a later variant of BGS-logic due to Grädel and Grohe [53]. Our presentation follows the one in Schalthöfer's PhD thesis [97].

Definition 3.0.1 (Syntax of BGS-logic)**.** *Let σ be a relational vocabulary. The set* BGS$[\sigma]$ *of BGS-terms and -formulas is defined inductively as follows:*

- *Every variable x is a term.*
- *$\emptyset$ and* Atoms *are terms.*
- *If r, s are terms, then* $\mathsf{Unique}(s), \mathsf{Pair}(r, s), \mathsf{Union}(s)$ *and* $\mathsf{Card}(s)$ *are terms.*
- *If $s_1, ..., s_k$ are terms and $R \in \sigma$ is a k-ary relation symbol, then $Rs_1...s_k$ is a formula.*
- *If φ, ψ are formulas, then so are $\varphi \wedge \psi$ and $\varphi \vee \psi$.*
- *If s, t are terms, then $s = t$ and $s \in t$ are formulas.*
- *If r, s are terms, x is a variable that is not free in r, and φ is a formula, then $\{s : x \in r : \varphi\}$ is a term (called* comprehension term*).*
- *If s is a term with only one free variable, then s^* is a term (called* iteration term*).*

The free variables of terms and formulas are defined in the usual way. The comprehension term $\{s : x \in r : \varphi\}$ binds x in s and φ, and the iteration term s^* binds the unique free variable of s. Sentences are formulas without free variables and are often also called programs or algorithms in the context of BGS and CPT.

In a given structure $\mathfrak{A}$, the BGS terms evaluate to hereditarily finite sets over the universe of $\mathfrak{A}$. Since the operation $\mathsf{Card}(s)$ evaluates to the cardinality of the given set, it is also necessary to encode natural numbers as elements of $\mathrm{HF}(A)$. This can be done with von Neumann ordinals, i.e. 0 is $\emptyset$, 1 is $\{\emptyset\}$, 2 is $\{\emptyset, \{\emptyset\}\}$, etc. In the following definition, we write $[n]$ to denote the von Neumann ordinal that encodes the number n.

Definition 3.0.2 (Semantics of BGS)**.** *Let $\mathfrak{A}$ be a finite σ-structure and β be an assignment that maps the free variables of the following terms and formulas to values in* $\mathrm{HF}(A)$*. The evaluation of terms and formulas in* $(\mathfrak{A}, \beta)$ *is defined as follows:*

- $[\![x]\!]^{(\mathfrak{A},\beta)} = \beta(x)$.
- $[\![\emptyset]\!]^{\mathfrak{A}} = \emptyset$ *and* $[\![\mathsf{Atoms}]\!]^{\mathfrak{A}} = A$.
- $\mathfrak{A}, \beta \models Rs_1 ... s_k$ *if and only if* $([\![s_1]\!]^{\mathfrak{A},\beta}, ..., [\![s_k]\!]^{\mathfrak{A},\beta}) \in R^{\mathfrak{A}}$.
- *Boolean combinations of formulas are interpreted as usual.*
- $[\![\mathsf{Unique}(s)]\!]^{\mathfrak{A},\beta} = \begin{cases} a, \textit{ if } [\![s]\!]^{\mathfrak{A},\beta} = \{a\}, \\ \emptyset, \textit{ otherwise.} \end{cases}$
- $[\![\mathsf{Pair}(r, s)]\!]^{\mathfrak{A},\beta} = \{[\![r]\!]^{\mathfrak{A},\beta}, [\![s]\!]^{\mathfrak{A},\beta}\}$.
- $[\![\mathsf{Union}(s)]\!]^{\mathfrak{A},\beta} = \bigcup_{b \in [\![s]\!]^{\mathfrak{A},\beta}} b$.
- $[\![\mathsf{Card}(s)]\!]^{\mathfrak{A},\beta} = [|[\![s]\!]^{\mathfrak{A},\beta}|]$.
- $\mathfrak{A}, \beta \models s = t$ *if and only if* $[\![s]\!]^{\mathfrak{A},\beta} = [\![t]\!]^{\mathfrak{A},\beta}$.
- $\mathfrak{A}, \beta \models s \in t$ *if and only if* $[\![s]\!]^{\mathfrak{A},\beta} \in [\![t]\!]^{\mathfrak{A},\beta}$.
- $[\![\{s : x \in r : \varphi\}]\!]^{\mathfrak{A},\beta} = \{[\![s]\!]^{\mathfrak{A},\beta[x\mapsto b]} \mid b \in [\![r]\!]^{\mathfrak{A},\beta} \textit{ with } \mathfrak{A}, \beta[x \mapsto b] \models \varphi\}$.
- *For the evaluation of an iteration term t^* in $\mathfrak{A}$, define the sequence $(t^i)_{i\in\mathbb{N}}$ as $t^0 = \emptyset$ and $t^{i+1} = t[x/t^i]$, where x is the free variable of t. Then*

$$[\![t]\!]^{\mathfrak{A}} = \begin{cases} [\![t^\ell]\!]^{\mathfrak{A}}, \textit{ for the least } \ell \textit{ with } [\![t^\ell]\!]^{\mathfrak{A}} = [\![t^{\ell+1}]\!]^{\mathfrak{A}}, \textit{ if such an } \ell \textit{ exists,} \\ \emptyset, \textit{ otherwise.} \end{cases}$$

The logic CPT is obtained from BGS by putting an explicit polynomial bound on every BGS-term and -formula. So CPT is the set of all pairs (t, p) and (φ, p), where t (respectively φ) is a BGS-term/-formula, and $p : \mathbb{N} \longrightarrow \mathbb{N}$ is a polynomial. We also write $\mathrm{CPT}(p(n))$ for the set of all CPT sentences whose polynomial bound is at most $p(n)$. As shown in [97], it also possible to drop this explicit polynomial bound and to let the CPT-programs themselves track their resource consumption.
The maximum number of free variables in any subterm of a CPT sentence is called its

variable rank. Sometimes, we are also interested in CPT *without counting*, which we denote CPT$^-$. This logic is like CPT but without the cardinality operator. Choiceless Polynomial Time without counting was already separated from PTIME in the article that introduced CPT [21] (see Theorem 3.2.2); therefore, we will not deal with CPT$^-$ too much, but the proof of this separation result is very instructive. We review it in more detail in Sections 3.2.2 and 4.4.

The semantics of CPT-terms and -formulas is the same as in BGS, with the exception of iteration terms. Their evaluation in a structure $\mathfrak{A}$ is restricted to remain within the polynomial bound $p(|A|)$, both with respect to the length of the iteration and the space required to store the computed h.f. sets. This space is taken to be the size of the *transitive closure* of a h.f. set (see Section 2.1).

Definition 3.0.3 (Semantics of CPT)**.** *Let* $(t,p) \in$ CPT *be a term. If t is not an iteration term, then the semantics is as in* BGS*-logic. For an iteration term* (t^*,p) *and a given structure* $\mathfrak{A}$ *of matching vocabulary, we have*

$$[\![(t^*,p)]\!]^{\mathfrak{A}} = \begin{cases} [\![t^\ell]\!]^{\mathfrak{A}}, & \text{for the least } \ell \le p(|A|) \text{ with } t^\ell = t^{\ell+1} \text{ and} \\ & \left|\mathrm{tc}([\![t^i]\!]^{\mathfrak{A}})\right| \le p(|A|) \text{ for all } i \le \ell, \text{ if such an } \ell \text{ exists,} \\ & \emptyset, \text{ otherwise.} \end{cases}$$

The *length of the iteration* $\mathrm{len}(t^*,p,\mathfrak{A})$ is the least ℓ with $[\![t^*]\!]^{\mathfrak{A}} = [\![t^\ell]\!]^{\mathfrak{A}}$. In particular, it is zero in the second case.

In this thesis, we are not concerned with writing concrete CPT-programs, but we are rather interested in lower bounds and inexpressibility results. For this, we will look at the h.f. sets that are created during CPT computations. These are usually called the *active objects* in the run of a CPT program on a given structure. Depending on the concrete formulation of CPT, there are different ways to define the set of active objects but the precise definition does not matter for the lower bound results. Intuitively, the active objects encompass all sets that appear in the transitive closures of the stages of an iteration term. As in [97], it suffices here to focus on iteration terms because the other terms can only produce sets of "constant complexity" anyway.

Definition 3.0.4 (Active objects)**.** *Let* (t,p) *be a* CPT*-term and* $\mathfrak{A}$ *a structure of matching vocabulary. Let S be the set of all terms s such that the iteration term s^* is a subterm of t. The* active objects *of* (t,p) *over* $\mathfrak{A}$ *are*

$$\mathrm{act}(t,p,\mathfrak{A}) = A \cup \bigcup_{s \in S} \bigcup_{i=0}^{len(s^*,p,\mathfrak{A})} \mathrm{tc}\left([\![s^i]\!]^{\mathfrak{A}}\right).$$

The definition is the same if t is not a term but a formula. By definition of CPT, the set of active objects is closed under the automorphisms of $\mathfrak{A}$. This is folklore and we do not prove it explicitly. Essentially, this is based on the fact that the comprehension terms can only select definable – and hence isomorphism-invariant – subsets of already computed sets. This property is also what makes CPT a logic in the sense of Gurevich's definition (see Section 2.2).

Lemma 3.0.5. *Let (t, p) be a* CPT*-term and $\mathfrak{A}$ a structure of matching vocabulary. Then* $\text{act}(t, p, \mathfrak{A})$ *is closed under the automorphisms of $\mathfrak{A}$, that is, for every $a \in \text{act}(t, p, \mathfrak{A})$, and $\pi \in \mathbf{Aut}(\mathfrak{A})$, $\pi(a)$ is also in* $\text{act}(t, p, \mathfrak{A})$*.*

Here, $\mathbf{Aut}(\mathfrak{A})$ denotes the *automorphism group* of the structure $\mathfrak{A}$. For details on automorphism groups and their action on h.f. sets, we refer to Chapter 4. A straightforward consequence of the lemma is that objects with super-polynomially large *orbits* cannot be among the active objects of any CPT-sentence, because the size of definable objects is polynomially bounded. The orbit of an object with respect to a permutation group is the set of all its images under the action of the group (again, see Chapter 4 for more details).

Corollary 3.0.6. *Let $(\mathfrak{A}_n)_{n\in\mathbb{N}}$ be a sequence of structures and $(a_n)_{n\in\mathbb{N}}$ with $a_n \in \text{HF}(A_n)$ be a sequence of h.f. sets over them. If the orbit-size of these objects*

$$|\mathbf{Orbit}(a_n)| = |\{\pi(a_n) \mid \pi \in \mathbf{Aut}(\mathfrak{A}_n)\}|$$

is not bounded by any polynomial in $|A_n|$, then there exists no CPT*-sentence or -term that activates a_n on input $\mathfrak{A}_n$, for all but finitely many $n \in \mathbb{N}$.*

Proof. Suppose for a contradiction that (φ, p) is a CPT-sentence which activates a_n on input $\mathfrak{A}_n$, for infinitely many $n \in \mathbb{N}$. For all these n, we have: $\text{act}(\varphi, p, \mathfrak{A}_n) \supseteq \mathbf{Orbit}(a_n)$ according to Lemma 3.0.5. At the same time, $|\text{act}(t, p, \mathfrak{A}_n)| \leq p^2(|A_n|)$ by Definitions 3.0.3 and 3.0.4: The transitive closure of every stage of an iteration term is bounded by $p(|A_n|)$, and the number of stages is also bounded by $p(|A_n|)$. Actually, we could say more precisely that $\mathbf{Orbit}(a_n)$ must even be contained within a single stage of an iteration term, and can therefore not be larger than $p(|A_n|)$. In any case, this contradicts the assumption that there exists no polynomial bound for $|\mathbf{Orbit}(a_n)|$ in terms of $|A_n|$. □

This corollary is actually the basis for most non-definability arguments in this thesis. Since all our new contributions are about lower bounds, it more or less suffices to view CPT as a h.f.-set-based computation model with the above symmetry limitation – the exact syntax and semantics of CPT are irrelevant most of the time. In fact, our lower bound from Chapter 7 applies to any computation model in which the stages of computation are automorphism-closed and polynomially bounded in size. Thus, the symmetry limitation could not even be overcome if the h.f. sets were guessed non-deterministically or if we dropped the polynomial bound on the *length* of the computation and considered a formalism like "Choiceless Polynomial Space".

Example: Defining linear orders in CPT

Let us now look at an example to get a feeling for how CPT works. For a start, we can quickly convince ourselves that CPT simulates FO and also fixed-point logic with

counting: Essentially, comprehension terms can be used to express quantifiers by defining the set of satisfying assignments for a formula and comparing it to $\emptyset$. To simulate FPC, the stages of a classical fixed-point induction can be encoded as sets, where we use e.g. Kuratowski pairs $\{x, \{x, y\}\}$ to represent ordered tuples in a fixed-point relation. Then it is not hard to simulate fixed-point operators with iteration terms. Counting can be done with the Card-operator.
It is also straightforward to verify that the usual set-theoretic operations such as union, intersection, and difference can be implemented in CPT (without iteration). For example, $a \cup b = \mathsf{Union}(\mathsf{Pair}(a, b))$, and $a \setminus b = \{x : x \in a : x \notin b\}$. Also, formation of singleton sets is possible via $\{a\} = \mathsf{Pair}(a, a)$. Therefore, in the following example, we will simply use these set-theoretic operations as if they were part of the syntax.

Our example will be the construction of a *linear order* on a given unordered input structure $\mathfrak{A}$. On structures where this is possible, CPT captures PTIME by the Immerman-Vardi Theorem. Without the polynomial bound, i.e. in BGS-logic, defining a linear order, or rather, the set of all possible linear orders of $\mathfrak{A}$, is not a problem. We encode orders such as $a < b < c$ as $\{c, \{b, \{a\}\}\}$. Moreover, we will always pair a linear order with its domain, i.e. the set of its elements. So we represent $a < b < c$ as $\{\{c, \{b, \{a\}\}\}, a, b, c\}$. This is necessary for technical reasons, so we can avoid adding the same element twice. The following term extends a given linear order x by the element y:

$$t_{\text{ext}}(x, y) := \mathsf{Pair}\Big((x \setminus \mathsf{Atoms}) \cup \{y\}\}, (x \cap \mathsf{Atoms}) \cup \{y\}\Big).$$

Let, for example, $x = \{\{c, \{b, \{a\}\}\}, a, b, c\}$. Then $x \setminus \mathsf{Atoms} = \{\{c, \{b, \{a\}\}\}\}$, and taking the union with $\{y\}$ yields the new extended linear order $\{y, \{c, \{b, \{a\}\}\}\}$. Next, using the Pair-operation, we put the resulting order in a set together with the set of atoms in its domain.

Our aim is to use this inside an iteration term whose stages are the sets of all partial orders of length one, then length two, then three, etc. The fixed-point of this term will be the set of all linear orders of the input structure. The update step of this iteration term looks like this:

$$t_{\text{update}}(x) := \mathsf{Union}\big(\{\{t_{\text{ext}}(y, z) : z \in \mathsf{Atoms} : \neg\varphi_{\text{contains}}(y, z)\} : y \in x : \text{true}\}\big).$$

Here, we used "true" as an abbreviation for $\emptyset = \emptyset$. The formula $\varphi_{\text{contains}}(y, z)$ still has to be defined. It checks if the order y already contains the element z. In total, $t_{\text{update}}(x)$ takes as input a set x of partial linear orders and extends every order by each element of the universe that is not yet contained. The result is again a set of partial linear orders (we need to apply Union, because else, it would be a set of sets of orders). The missing containment-check can be implemented easily because our representation of a linear order contains its domain as a subset:

$$\varphi_{\text{contains}}(y, z) := z \in y.$$

Finally, we put all this together into an iteration term. Since the first stage is always the empty set, we have to implement a case distinction for the initialisation step:

$$t_<(x)^* := \big(\{\mathsf{Pair}(\{y\}, y) : y \in \mathsf{Atoms} : x = \emptyset\} \cup t_{\text{update}}(x)\big)^*.$$

In the first application of $t_<(x)$, we have $x = \emptyset$, so the comprehension term will yield the set $\{\{\{a\}, a\} \mid a \in A\}$, which contains the representations of all 1-element orders (and $t_{\text{update}}(x) = \emptyset$). If $x \neq \emptyset$, then the term $t_{\text{update}}(x)$ evaluates to the set of all linear orders that are one element longer than the orders in the current stage x. An example run on a structure $\mathfrak{A}$ with universe $\{a, b, c\}$ is:

$$\begin{aligned}
[\![t_<^0]\!]^{\mathfrak{A}} &= && \emptyset \\
[\![t_<^1]\!]^{\mathfrak{A}} &= && \{\{\{a\}, a\}, \{\{b\}, b\}, \{\{c\}, c\}\} \\
[\![t_<^2]\!]^{\mathfrak{A}} &= && \{\{\{b, \{a\}\}, a, b\}, \{\{a, \{b\}\}, a, b\}, \{\{c, \{a\}\}, a, c\}, \{\{a, \{c\}\}, a, c\}, \\
& && \{\{b, \{c\}\}, b, c\}, \{\{c, \{b\}\}, b, c\}\}. \\
[\![t_<^3]\!]^{\mathfrak{A}} &= && ...
\end{aligned}$$

The stage $[\![t_<^3]\!]^{\mathfrak{A}}$ is the result of the computation and contains all $3! = 6$ possible linear orders of $\{a, b, c\}$. Now given this set, it is possible to decide any PTIME-property of $\mathfrak{A}$ by applying the Immerman-Vardi Theorem and running the corresponding fixed-point computation for each of these linear orders. Since the result of each of these computations is independent of the chosen order of $\mathfrak{A}$, we obtain a single yes/no-answer in this way.

At this point, we can do a quick sanity check of Lemma 3.0.5 and pick any set that is *active* in the computation above. For example, $x := \{a, \{b\}\}$ appears in $\text{tc}(t_<^2)$. Even if $\mathbf{Aut}(\mathfrak{A}) = \mathbf{Sym}(A)$, so if $\mathfrak{A}$ is as symmetric as possible, we see that indeed, for every $\pi \in \mathbf{Aut}(\mathfrak{A})$, $\pi(x)$ also appears as an active object: We have $\{a, \{c\}\}$, $\{b, \{a\}\}$, etc. in $\text{tc}([\![t_<^2]\!]^{\mathfrak{A}})$. As one can see in this example, each stage of the iteration is itself automorphism-closed. Lemma 3.0.5 implies that the above computation is the best we can hope for if $\mathbf{Aut}(\mathfrak{A}) = \mathbf{Sym}(A)$, i.e. we cannot compute fewer than *all* possible linear orders in that case. Of course, for structures with more restricted symmetries (for instance, rigid structures), it may be possible to compute in BGS a much smaller set of linear orders. This would require a different BGS-term than the one we used, because our $t_<$ does not access the relations of the structure, and so, any structure looks like a fully symmetric naked set to our term. In Section 3.2.1 below, we review what is known on the power of CPT to compute linear orders on certain *restricted classes* of structures. On fully symmetric structures, however, computing orders is only possible in BGS but not in CPT (at least not in the naive way as in the example), since $|A|!$ is not polynomially bounded. Observe that the problem is not the *length* of the iteration but the *size* of the stages.

3.1 Comparing CPT with other logics

The previous example brings us to a nice and relatively simple argument which shows that CPT is strictly more powerful than FPC. The idea is that $|A|!$ can artificially be made polynomial – if A is not the whole universe of the input structure. So we simply augment the structure $\mathfrak{A}$ that we are interested in with padding, i.e. useless elements in the universe. This way, we obtain a structure $\mathfrak{B}$ that is the disjoint union of $\mathfrak{A}$ with a large number of new points that are marked as padding with a special unary relation symbol. If the padding is large enough, then $|A|!$ is polynomially bounded in $|B|$, and so, it is easy to compute in CPT the set of all linear orders of A in the input structure $\mathfrak{B}$. With this, any PTIME-property of $\mathfrak{A}$ is CPT-definable – in $\mathfrak{B}$. However, it can be seen that FPC does not benefit from padding. If the structure is fixed, then any FPC-sentence φ can be written as a $\mathcal{C}^k$-sentence, where k depends only on φ. If two structures $\mathfrak{A}_1, \mathfrak{A}_2$ are $\mathcal{C}^k$-equivalent, then they will still be $\mathcal{C}^k$-equivalent after padding has been added because Spoiler cannot use the padding to suddenly have a winning strategy in the bijective k-pebble game. With these considerations one can separate CPT from FPC using the Cai-Fürer-Immerman (CFI) query, which we introduce in detail in Chapter 5. Basically, the CFI-construction [29] can be applied to any family of connected "*base graphs*" and yields pairs of non-isomorphic *CFI-graphs* over these base graphs. Such non-isomorphic CFI-graphs can be distinguished in PTIME but not in the logic $\mathcal{C}^k$ for any fixed k; hence, they are a typical benchmark for PTIME-logics and are in particular indistinguishable in FPC. However, using the above argument and the fact that CFI-graphs can be distinguished with a polynomial time algorithm, one can show that CFI-graphs with sufficient padding are distinguishable in CPT. The situation is similar with multipedes. These are also structures whose isomorphism problem is in polynomial time but not definable in $\mathcal{C}^k$ [67]. In [22], Blass, Gurevich and Shelah used the padding argument on CFI-graphs/multipedes to show that FPC $\lneq$ CPT.

Later on, it was proven by Dawar, Richerby and Rossman [40] that padding is not actually necessary: CPT can distinguish non-isomorphic CFI graphs over linearly ordered base graphs even if there is no padding in the input. Again, FPC fails at this task (the Immerman-Vardi Theorem does not apply because only the base graphs are linearly ordered; the CFI-graphs only inherit a *preorder* from this linear order). The CPT-algorithm given in [40] constructs deeply nested h.f. sets, which take the power of CPT beyond that of FPC. The authors also prove that sets of greater than constant nesting depth are necessary to solve the CFI-query (unless the instances are padded). So we know two independent mechanisms of CPT that are responsible for its high expressiveness, compared to FPC: Firstly, the polynomial resource bound that takes into account the size of the input structure and hence lets CPT benefit from padding; secondly, the possibility to use *deeply nested* sets as "data structures". In Section 6, we review the content of [40] in more detail, since it is the main starting point for the investigations in this thesis. To summarise, we have:

Theorem 3.1.1 ([22], [40])**.** FPC $\lneq$ CPT *(even on unpadded structures).*

More recently, Moritz Lichter was able to separate CPT also from *rank logic* [82], which is an extension of FPC by an operator to compute the rank of definable matrices [55]. The separating example is a variant of the CFI-query on linearly ordered base graphs over rings $\mathbb{Z}_{2^i}$, for varying $i \in \mathbb{N}$. Lichter showed that these CFI-structures are indistinguishable in rank logic and even in the *infinitary linear-algebraic logic* by Dawar, Grädel, and Pakusa [35], an extension of infinitary FO with *all* isomorphism-invariant linear-algebraic operators. The proof is a construction of a winning strategy for Duplicator in the *invertible-map game*, which characterises the distinguishing power of this linear-algebraic logic. In CPT, these CFI-graphs over rings can be canonised using a known canonisation procedure for structures with Abelian colour classes (see Theorem 3.2.1 below).

Theorem 3.1.2 ([82], [37])**.** *There is a query that is definable in* CPT *but not in rank logic, nor in infinitary linear-algebraic logic.*

It is open whether this means that rank logic is a fragment of CPT or if these two logics are simply incomparable. We suspect the latter because rank logic can easily solve linear equation systems over finite fields on *unordered* structures. As we outline later on, for CPT, this is only known to be possible on certain families of *preordered* structures. Our results from Chapter 7 indicate that positive results for CPT on preordered structures do not imply that CPT should be equally powerful in the unordered case. In fact, we conjecture that CPT cannot define the CFI-query on general unordered graphs, which would mean that it cannot solve linear equation systems over finite fields, either.

3.2 The power and limitations of CPT

We now turn our attention to the most important positive and negative results that are known concerning the power of CPT. On the negative side, we know of no decision problem in polynomial time that is provably inexpressible in CPT. However, the foundations for proving lower bounds against CPT have been laid: Classical tools from finite model theory such as the bijective k-pebble game and zero-one laws have been studied in the context of CPT (or CPT without counting, in the case of zero-one laws) and are applicable to an extent. In Chapter 6, we deal with some of these tools in detail. But before we come to these lower bound techniques, let us focus on the expressivity of CPT: There has been a line of research that, one might say, is devoted to understanding in what relaxed situations the Immerman-Vardi Theorem remains valid for CPT. Namely, it has been shown that – in contrast to FPC – CPT can define certain preorderd variants of the difficult CFI-query, and, more generally, it captures PTIME on certain classes of preordered structures. So while the Immerman-Vardi Theorem requires the structures to be *totally ordered* in order for FPC to capture PTIME on them, the total order can be relaxed to a preorder in the case of CPT. The colour classes of such a preorder, however, need to have certain properties, for example, be small enough, or have automorphism groups of a particular kind.

3.2.1 Expressivity results

The earliest positive results on the power of CPT are due to Blass, Gurevich, and Shelah themselves, but they do not go much beyond the power of FPC. As already said, in [22], they show that CPT can define the CFI-query on padded structures. Moreover, it is shown in [22] that it can be decided in CPT whether a given bipartite graph has a perfect matching – however, the authors also give an algorithm that can be implemented in FPC, so this is not an example for the power of CPT, in particular. A few years later, in a follow-up paper [20], Blass and Gurevich sketched a CPT-algorithm for computing the *determinant* of a given matrix over a finite field, which is a choiceless implementation of an older parallel algorithm by Csanky [33].
The first CPT-algorithm that separates CPT from FPC on *unpadded* structures is the one by Dawar, Richerby and Rossman [40] mentioned above, which distinguishes CFI-structures over linearly ordered base graphs using a deeply nested h.f. set construction. These specific h.f. sets encode the relevant information about the input structure in a succinct and algorithmically useful way. The idea behind it has subsequently been generalised and led to stronger definability results for CPT: In [91], Pakusa, Schalthöfer, and Selman extended the algorithm from [40] to CFI-graphs over *preordered base graphs* with colour classes of *logarithmic size*, and to unordered base graphs that contain at least one vertex of *linear degree*. The latter result exploits implicit padding that occurs in the construction of CFI-graphs over base graphs of large degree. In particular, it means that CPT can define the CFI-query over complete base graphs. In Chapter 6, we present all these algorithms for different variants of the CFI-query in more detail.

The technique of using a particular deeply nested h.f. set construction to encode information has also been used to solve a more general problem in CPT: Namely, Abu Zaid, Grädel, Grohe, and Pakusa showed in [4] that CPT can *canonise* all structures that come with a preorder whose colour classes have *Abelian automorphisms*. This means that on such structures, CPT can define a linearly ordered copy of the original structure, and hence perform any PTIME-computation on it using the Immerman-Vardi Theorem:

Theorem 3.2.1 (Corollary 19 in [4])**.** *CPT captures* PTIME *on classes of q-bounded structures with Abelian colours, and specifically, on* 2*-bounded structures.*

The term *q-bounded* means that the structures have a preorder whose colour classes are each of size at most q, where q is some constant. The colour classes being Abelian means that the induced substructure on each colour class has an Abelian automorphism group (Abelian means commutative – see Chapter 4 for the background from group theory). In the case of preordered CFI-graphs, for instance, the automorphism group of each colour class is isomorphic to $(\mathbb{Z}_2, +)$, which is Abelian (or to another finite field, for generalised CFI-structures). Extreme examples of non-Abelian groups are $\mathbf{Sym}_n$ or $\mathbf{Alt}_n$, the symmetric or the alternating group, which are essentially the largest possible permutation groups. The definability of the CFI-query on linearly ordered base graphs is also a consequence of Theorem 3.2.1. This theorem additionally solves another problem that was previously not known to be CPT-definable, namely the isomorphism problem of *multipedes* (without padding). Multipedes are preordered structures with colour classes of size 2 (the pairs of feet, which are indistinguishable in $\mathcal{C}^k$), so CPT captures PTIME on them by the theorem, and isomorphism of multipedes is computable in polynomial time.

The proof of Theorem 3.2.1 is an interesting example of the connections between linear algebra and the graph isomorphism/canonisation problem: On structures with Abelian colours, the canonisation problem essentially reduces to solving linear equation systems over a finite ring $\mathbb{Z}_d$. The linear equation systems that arise from q-bounded Abelian structures have very specific properties and are called *cyclic* in [4]. Basically, this means that they have a preorder on the set of variables such that the variables in each colour class are related via "additive shifts", i.e. they satisfy constraints of the form $x = y+1$, for example. The main technical contribution of [4] is to show that such cyclic linear equation systems are solvable in CPT. This works by encoding the equation systems as h.f. sets in such a way that each colour class of the variables corresponds to a single h.f. object and can hence be "factored out". The resulting equation system is then not only preordered, but totally ordered because each colour class is treated as one object. Totally ordered equation systems over finite fields can be solved using for example Gaussian elimination (and the Immerman-Vardi Theorem). So the mechanism that is responsible for the power of CPT on structures with Abelian colours is intuitively speaking the ability to encode the colour classes of a preorder as h.f. objects in such a way that each colour class appears as a single object but its relevant information is still implicit in the structure of the h.f. set. Once each colour class is compressed into a single object in this way, the preorder on the elements reduces to a total order on the set of "compressed colour classes". Then on this totally ordered set of compressed colour classes, the Immerman-Vardi Theorem allows us to carry out any polynomial time computation. This is why one might say that the power of CPT lies in the fact that it allows to lift the Immerman-Vardi Theorem from totally ordered to preordered structures with suitable colour classes. What this means for the power of CPT on *unordered* structures seems unclear, though. For example, it is open whether CPT can solve *unordered* linear equation systems over finite fields – the *cyclic* ones that it can solve are very specific *preordered* equation systems. If CPT could solve unordered ones, too, then it could also generally define the CFI-query on unordered structures, which seems to be a very challenging task.

More recently, Lichter and Schweitzer generalised Theorem 3.2.1 to q-bounded structures with *dihedral* automorphism groups (Theorem 1 in [84]). A dihedral group is the automorphism group of a regular n-gon and consists of rotations and reflections. For $n > 2$, dihedral groups are non-Abelian because of the reflections – the rotations correspond to Abelian groups $\mathbb{Z}_d$. One could say that dihedral groups are just "slightly non-Abelian". While from Abelian to dihedral is a small step for a group, it is a big step on the technical side, and the proof of the canonisation result is considerably more involved. Thus, continuing on this route would probably be difficult, and as already mentioned, it is not at all clear that such capturing results for preordered structures will eventually also extend to unordered structures.

3.2.2 Non-definability results

The automorphism-based symmetry approach Proving non-definability results for CPT (with counting) is not easy and there is basically only one established approach for this, which we also follow most of the time in this thesis. The general idea is to use the fact that every h.f. set that is activated during a CPT-computation on a given structure $\mathfrak{A}$ is activated together with its entire $\mathbf{Aut}(\mathfrak{A})$-orbit (Lemma 3.0.5). If one can prove that a certain h.f. set has an orbit of super-polynomial size with respect to the structure in question, then this object is not CPT-definable because this would violate the polynomial resource bound (Corollary 3.0.6). This argument by itself can be used to show the non-definability of *functional problems*, i.e. that certain objects are not definable on certain input structures. Ultimately, the aim is to show the non-definability of *decision problems*, preferably ones in PTIME. A potential way to lift functional non-definability results to decision problems is to prove that solving a given decision problem requires the activation of some h.f. set x with certain specific properties – and then to show that these properties necessarily entail a super-polynomial orbit size of x. We are only aware of three examples where such super-polynomial orbit arguments have been employed in the literature. The first is the proof that CPT without counting cannot define the EVEN-query (nor the bipartite perfect matching problem) by Blass, Gurevich, and Shelah.

Theorem 3.2.2 (Theorems 42 and 43 in [21])**.** *The following two problems are not definable in* CPT^-*:*

- *The parity of the structure size in a given finite structure with empty vocabulary.*
- *The existence of a perfect matching in a given undirected bipartite graph.*

As we have already mentioned in the introduction, the proof is based on a property of h.f. sets that is called their *minimum support size*. For a h.f. set x over a structure $\mathfrak{A}$, the minimum support size is the least number of elements in the universe A such that fixing these elements also fixes x. More precisely, a support of x is a subset $S \subseteq A$ such that every permutation in $\mathbf{Aut}(\mathfrak{A})$ which fixes each point in S also fixes the object x. It can be shown that deciding either of the two problems mentioned in the theorem requires the activation of a set with a super-constant support (unless we have the cardinality operator available). The second step is then to show that all objects whose minimum support size is super-constant have a super-polynomial orbit. This relies on the fact that the structures we are considering in the theorem are basically as symmetric as possible: For instance, structures with empty vocabulary are just sets, and their automorphism group is the full symmetric group acting on the universe. When so many automorphisms are present, it is relatively easy to prove that super-constant support implies super-polynomial orbit size. In Chapter 4, we provide more details about the theory of supports and also sketch why the connection between support-size and orbit-size exists on highly symmetric structures. A central technical challenge in this thesis is to prove such super-polynomial orbit theorems also on structures whose automorphism groups are smaller compared to the structure size (i.e. on n-dimensional hypercubes – see the results in Chapter 7).

This seems to be necessary because such excessively symmetric structures as edgeless or complete graphs are probably not suitable candidate instances to separate CPT (*with counting*) from PTIME.

The second result in the literature which makes a connection between the support size of the h.f. sets activated in a CPT computation and the expressive power is due to Dawar, Richerby and Rossman [40]. They studied more complex structures than just naked sets, namely CFI-graphs. As long as these satisfy a certain homogeneity condition (meaning that $\mathcal{C}^k$-types coincide with orbits), it can be shown that distinguishing non-isomorphic CFI-graphs in CPT requires the activation of a h.f. set whose minimum support is linear in the treewidth of the base graph (see Theorem 6.2.7). This is actually a much stronger support lower bound than the super-constant one that appears in the proof of Theorem 3.2.2. However, turning this into a non-definability result for the CFI-query would also require to show that objects with linear support over suitable CFI-graphs necessarily have a super-polynomial orbit. This is much harder to prove for CFI-graphs than for naked sets (or complete graphs) because the automorphism group of interesting CFI-graphs is not the full symmetric group acting on the universe (as already mentioned, the CFI-query over complete base graphs is indeed definable in CPT, so these are not useful for lower bounds). In Chapter 9, we study this problem on n-dimensional hypercubes as base graphs and prove structural restrictions on h.f. sets with large support and polynomial orbit size over hypercube CFI-structures. We are unfortunately not able to show that large support size entails super-polynomial orbit size on these structures (which would separate CPT from PTIME) but we show that large support and small orbit can only go together if the h.f. set satisfies particular conditions.

Dawar, Richerby and Rossman also prove a structural limitation of h.f. sets that have linear support and a polynomial orbit, namely they show that such sets must have a super-constant rank (i.e. nesting depth). In total, this shows that the nesting depth of h.f. sets is indeed necessary to get the full power of CPT because defining the CFI-query in CPT requires the activation of a nested object (see Theorem 6.2.8). We present the support and the rank lower bound from [40] in detail in Chapter 6.

The third non-definability result that has been obtained with such orbit-support arguments is not about a decision problem but a functional problem in PTIME. The problem is: Given a finite vector space V over some finite field $\mathbb{F}$, compute the set $\mathcal{H}(V)$ of all hyperplanes in V. This is the set of all subspaces with codimension 1 in V. The vector space is presented as a structure whose elements are the vectors. It has a binary $+$ operation and a unary scalar multiplication for every field element in $\mathbb{F}$. The problem asks to define $\mathcal{H}(V)$ as a h.f. set over V. More precisely, this is a set of subsets of V and its transitive closure has polynomial size in $|V|$, so it is indeed PTIME-computable. Benjamin Rossman showed that the orbit size of $\mathcal{H}(V)$ is super-polynomial in $|V|$ and therefore, this set of all hyperplanes is not CPT-definable.

Theorem 3.2.3 (Theorem 6.1 in [95])**.** *No* CPT*-program computes the operation* $V \longrightarrow$

$\mathcal{H}(V)$ over finite vector spaces over a fixed finite field $\mathbb{F}$.

The proof technique is similar to the one of Theorem 3.2.2: Hyperplanes have a large support, and Rossman shows that this entails a super-polynomial orbit. The reason why this support-orbit connection works out nicely again in this case is because vector spaces are sufficiently symmetric (the automorphism group is the general linear group $\mathbf{GL}(V)$). So in summary, the support-orbit argument that was first developed in [21] works best on structures whose automorphism group is extremely large compared to the structure size, e.g. $\mathbf{Sym}_n$ or the general linear group. In fact, Rossman also identified a property of automorphism groups (called the *(k,r)-support property*) that makes such an argument applicable. In the group theory chapter (Chapter 4), we will get back to this.

Zero-one laws The second approach towards CPT lower bounds is via zero-one laws; however, this is perhaps a bit less significant than the automorphism-based symmetry approach because it only applies to CPT$^-$, which does not capture PTIME, anyway. Zero-one laws exist also for weaker logics, for example fixed-point logic (*without counting*) and infinitary FO (see for example in [81]). The meaning of a zero-one law is that on families of random structures whose size tends towards infinity, every sentence in the logic holds with asymptotic probability 0 or 1 in these structures. The first proof of the zero-one law for CPT$^-$ is due to Shelah [101]. Subsequently, this was generalised and presented differently by Blass and Gurevich [18] [19].

Theorem 3.2.4 (Theorem 2.2 in [18])**.** *Let a BGS-program Π with Boolean output and a polynomial bound for the number of active elements be given. There exists a number m, an output value v, and a class $\mathcal{C}$ of undirected graphs such that $\mathcal{C}$ has asymptotic probability one and such that, for each $G \in \mathcal{C}$, either*

- *Π on input G halts after exactly m steps with output value v and without exceeding the bound on the active elements, or*
- *Π on input G either never halts or exceeds the bound on active elements.*

The BGS-computation model that Blass and Gurevich are referring to is CPT$^-$ without an explicit bound on the length of the computations, so it is possible that a computation never terminates. Thus, the theorem states that all successful computations of Π on graphs in $\mathcal{C}$ output the same result. The class $\mathcal{C}$ is a family of all random graphs of size greater than some fixed number satisfying certain *strong extension axioms* (and the structures do satisfy these axioms with probability one in the limit). The proof of the classical zero-one law for FO also makes use of so-called extension axioms. Roughly speaking, they state that any k-tuple of a given type that is realised in the random structure can be extended to a $(k+1)$-tuple of any other type in the structure. These standard extension axioms are too weak to prove the zero-one law for CPT, which is why the authors use *strong* versions of them. Strong extension axioms additionally require that there exist sufficiently many realisations of the respective types in the random structure. The zero-one law for CPT then follows by the construction of a winning strategy for Duplicator, which exploits symmetries of the random graphs. This is quite remarkable because random graphs are rigid with high probability, and so they have no non-trivial automorphisms. However, the strong extension axioms guarantee enough "local symmetry": The authors consider *motions*, which are partial automorphisms of the structure, and they also define a suitable notion of supports for h.f. sets with respect to the action of these motions. This actually looks like a quite promising tool in general because it can be applied even on structures with very few or no non-trivial automorphisms, whereas the techniques described in the previous paragraph always require structures with extremely many automorphisms. However, the disadvantage of the motions is that they are more awkward to work with and have a less developed theory than permutation groups.

An immediate consequence of Theorem 3.2.4 is the non-definability of the EVEN-query in CPT^-: For any program Π which supposedly defines it, the theorem gives us a class $\mathcal{C}$ of suitable random graphs on which Π always outputs the same value, even though $\mathcal{C}$ contains graphs of both parities.

There is a second interesting application of this zero-one law (or rather, of its proof) in the literature. Namely, Rossman had proposed (see [20]) the *Abelian Semigroup Subset Sum* problem as a surprisingly simple candidate for which it was unclear whether it is CPT- or even FPC-definable. The problem asks, given a finite Abelian semigroup $(G, +)$, and a subset $X \subseteq G$, to compute the sum $\sum X$ (since G is Abelian, the value of the sum is well-defined). Clearly, one can compute this in PTIME, but it is not obvious how to do it in a choiceless fashion. More than 10 years later, the descriptive complexity status of this problem was resolved by Abu Zaid, Dawar, Grädel, and Pakusa [3]. They presented an FPC-definable dynamic programming algorithm for it, which also means that it is in CPT. At the same time, they showed that counting is necessary: A variation of the proof of Theorem 3.2.4 leads to the result that CPT^- fails to solve the Abelian Semigroup Subset Sum problem.

3.3 Alternative presentations and variants of CPT

In addition to the original formulation of CPT as an abstract state machine model, and the later developed BGS-logic, there is also a neat way to define Choiceless Polynomial Time via iterated first-order interpretations. For our lower bound considerations, the approach via BGS-logic is often more useful because it exposes the connection to symmetric hereditarily finite sets. However, *polynomial-time interpretation logic* (PIL) has the advantage that it naturally gives rise to certain interesting fragments and extensions of CPT. This logic was first studied by Svenja Schalthöfer in her Master's thesis [98] and in [52] together with Erich Grädel, Łukasz Kaiser, and Wied Pakusa.

3.3.1 Polynomial-time Interpretation Logic

Choiceless Polynomial Time can alternatively be defined as the polynomial-time fragment of *interpretation logic* (IL). Sentences in interpretation logic are essentially $(\mathrm{FO} + \mathrm{H})$-interpretations that are iteratively applied to the input structure. The resulting sequence

of structures is the computation; it plays the same role as the sequence of hereditarily finite sets that we get when we evaluate BGS iteration terms.
The logic FO + H is first-order logic extended with the *Härtig quantifier* H. If φ and ψ are formulas, then so is $\mathsf{H}x.\varphi(x).\psi(x)$. The Härtig quantifier evaluates to true in a structure $\mathfrak{A}$ iff the number of assignments to x that satisfy φ is the same as for ψ. This is needed to emulate the cardinality operator of CPT.

Definition 3.3.1 (Interpretation logic, [97]). *Let σ, τ be relational vocabularies. An* $\mathrm{IL}[\sigma,\tau]$*-sentence is a tuple* $\Pi = (\mathcal{I}_{init}, \mathcal{I}_{step}, \psi_{halt}, \psi_{out})$*, where* $\mathcal{I}_{init}$ *is an* $(\mathrm{FO}+\mathrm{H})[\sigma,\tau]$*-interpretation,* $\mathcal{I}_{step}$ *is an* $(\mathrm{FO}+\mathrm{H})[\tau,\tau]$*-interpretation, both* $\mathcal{I}_{init}$ *and* $\mathcal{I}_{step}$ *are parameter-free, and* ψ_{halt} *and* ψ_{out} *are* $(\mathrm{FO}+\mathrm{H})[\tau]$*-sentences.*

Given a σ-structure $\mathfrak{A}$, we evaluate an IL-sentence Π by considering the *run* of Π on $\mathfrak{A}$. This is the sequence given by $\mathfrak{A}_0 = \mathcal{I}_{\text{init}}(\mathfrak{A}), \mathfrak{A}_{i+1} = \mathcal{I}_{\text{step}}(\mathfrak{A}_i)$. The run either does not terminate, or it halts at the first $\mathfrak{A}_i$ that satisfies ψ_{halt}. This $\mathfrak{A}_i$ is also denoted $\Pi(\mathfrak{A})$, and we have $\mathfrak{A} \models \Pi$ if and only if the run of Π on $\mathfrak{A}$ halts and $\Pi(\mathfrak{A}) \models \psi_{\text{out}}$.

Now *polynomial-time interpretation logic* (PIL) is defined by pairing IL-sentences with polynomials, so PIL-sentences are of the form (Π, p). A structure $\mathfrak{A}$ satisfies (Π, p) if the run of Π on $\mathfrak{A}$ is accepting, where the run now terminates not only if the current stage $\mathfrak{A}_i$ satisfies ψ_{halt}, but also automatically after $p(|A|)$ many steps or if in some step i, $|A_i| > p(|A|)$.

The reason why PIL and CPT mutually simulate each other is that the interpretations in PIL can define a congruence relation $\approx$ on the structure and contract the congruence classes. This corresponds to the construction of unordered sets in CPT. In addition to that, PIL can construct ordered tuples over the given structure because the interpretations may be multidimensional. So the iterated application of an interpretation to the input structure produces a new structure whose elements are implicitly sets of tuples of sets of tuples, and so on. However, this nesting structure is less transparent in PIL than in BGS logic. With the above reasoning, it can be shown that PIL and CPT have exactly the same expressive power:

Theorem 3.3.2 (Theorem 1 in [52]). PIL $\equiv$ CPT.

Besides being sometimes more convenient for "programming" in, PIL is also interesting because it exposes certain natural fragments of CPT: One can simply restrict the interpretations that are allowed in the PIL-programs. This leads to the following variants. All the results are from [52].

- PIL^-: In PIL^-, all interpretations are FO-interpretations instead of FO + H-interpretations. It holds $\mathrm{PIL}^- \equiv \mathrm{CPT}^-$.
- *One-dimensional* PIL^*: Here, all interpretations are one-dimensional, and for technical reasons, the logic is always evaluated in two-sorted structures where the second sort is a linear order of the same size as the first sort. It holds $\mathrm{PIL}^* \equiv \mathrm{FPC}$.

- *One-dimensional* PIL^-. The one-dimensional fragment without counting has the same expressive power as $\text{PFP}|_{\textsc{Ptime}}$, the polynomial time restriction of partial fixed-point logic. It follows with a result due to Abiteboul and Vianu [2]: One-dimensional $\text{PIL}^- \equiv \text{LFP}$ if and only if $\textsc{Ptime} = \textsc{Pspace}$.
- *Two-dimensional* PIL $\equiv$ PIL, so the ability to create pairs is enough to get the full power of PIL.
- *Congruence-free* PIL (without counting): In this variant, the interpretations must not define non-trivial congruence relations, so the congruence formula is always $\varphi_{\approx}(x, y) = (x = y)$. For the version without counting, it holds ($\approx$-free PIL^-) $\equiv$ $\text{while}_{\text{new}}|_{\textsc{Ptime}}$. The language $\text{while}_{\text{new}}$ is an extension of a while-programming language equivalent to PFP. In $\text{while}_{\text{new}}$, one can additionally construct new elements corresponding to tuples. It was shown in [23] that $\text{while}_{\text{new}}|_{\textsc{Ptime}}$ is strictly weaker than CPT^-. Thus, it holds: ($\approx$-free PIL^-) $\lneq$ PIL^-.
- For $\approx$-free PIL *with counting*, it holds FPC $\lneq$ ($\approx$ -free PIL) $\lneq$ PIL. The separation between FPC and $\approx$ -free PIL can be shown with a padding argument, and the separation between ($\approx$ -free PIL) and PIL follows because as shown in [52], there are classes of structures (CFI-graphs), on which $\approx$ -free PIL can be simulated in CPT using only sets of bounded rank. But according to Theorem 6.2.8, which was proved in [40], there are queries on these structures that are definable in CPT, but not with sets of bounded rank. Furthermore, in [97], a separation between $\approx$-free PIL and bounded-rank CPT is shown: The CFI-query on complete graphs is definable in CPT with sets of constant rank, but not in $\approx$ -free PIL.

These results highlight that in order to obtain the full power of CPT, we need both the ability to construct new objects from tuples, and also the ability to form unordered sets of these tuples. It suffices if the tuples have arity two because by iterated pairing, we can also construct longer tuples. The necessity to construct sets, i.e. to factor out congruence classes, has to do with the polynomial bound: It simply saves space, which is sometimes necessary in order to stay within polynomial bounds.

Non-deterministic extensions of PIL

Polynomial-time interpretation logic can be quite naturally extended with non-deterministic mechanisms. In Schalthöfer's PhD thesis [97], the logic $\exists$PIL is defined: The sentences of this logic are PIL-sentences (Π, p) where the $\mathcal{I}_{\text{step}}$-interpretation in Π may have parameters. Such a PIL-sentence does not define a unique run on a given structure $\mathfrak{A}$. Any sequence of structures $\mathfrak{A}_0 = \mathfrak{A}, \mathfrak{A}_1, \mathfrak{A}_2, ...$ is a run of (Π, p) on $\mathfrak{A}$ if, for each i, there exists a parameter tuple $\overline{a}_i$ in $\mathfrak{A}_i$ such that $\mathfrak{A}_{i+1} = \mathcal{I}_{\text{step}}(\mathfrak{A}_i, \overline{a}_i)$. Then a structure $\mathfrak{A}$ satisfies an $\exists$PIL-sentence (Π, p) if there exists at least one accepting run. Theorem 4.29 in [97] states that

$$\exists\text{PIL} \equiv \exists\text{SO}$$
$$\forall\text{PIL} \equiv \forall\text{SO}.$$

In ∃PIL, it is possible to guess any k-ary relation X by constructing it tuple by tuple, adding the guessed parameter tuple of $\mathcal{I}_{\text{step}}$ to X each time. To simulate ∃PIL in ∃SO, one can observe that it can be decided in NP if a given input structure satisfies a fixed ∃PIL-sentence. Then Fagin's Theorem can be used to simulate this in ∃SO. The equivalence ∀PIL ≡ ∀SO then follows because these can define precisely the complementary queries. A generalisation of ∃PIL and ∀PIL to *alternating* PIL is also presented in [97]. We omit the technical details of this logic but essentially this is the PIL-version of alternating Turing machines: The set of possible runs defines a computation tree with existential and universal nodes (nullary relations in the respective structure $\mathfrak{A}_i$ determine which node has which type). Alternating PIL has the same power as full second-order logic (Theorem 4.29 in [97]).

It is not clear if ∃PIL is the "right" non-deterministic version of Choiceless Polynomial Time. As Schalthöfer points out in [97], there is also the option to consider non-deterministic choices with regards to the $\mathcal{I}_{\text{step}}$-interpretation instead of its parameters. Such a logic has not been defined or studied anywhere to our knowledge but one could for example define it as a version of PIL where each program has two $\mathcal{I}_{\text{step}}$-interpretations, $\mathcal{I}^0_{\text{step}}$ and $\mathcal{I}^1_{\text{step}}$. Then in every step, one of them is non-deterministically selected and applied to the current structure. The input structure is accepted if there exists an accepting run. Let us call this logic, that we have only sketched, NPIL. It can be simulated by ∃PIL, which is why in [97], only ∃PIL is studied. However, our investigations in Chapter 10 indicate that this probably weaker NPIL perhaps deserves more attention. The main reason why NPIL may be seen as a more natural non-deterministic variant of CPT is that it is still choiceless: It cannot break symmetries of the input structure, while ∃PIL can. The latter can guess an arbitrary tuple in each step (and hence easily define a linear order, for example), while NPIL can only guess which interpretation to execute in each step. No matter which of the two interpretations is guessed, the update of the structure will be symmetry-invariant because the interpretations do not take parameters. In Chapter 10, we study the power of CPT to distinguish non-isomorphic graphs, specifically in comparison with the computation model *Deep Weisfeiler Leman* and a propositional proof system called the *extended polynomial calculus*. In this proof system, two graphs G and H are distinguishable if there exists a proof for them being non-isomorphic. As we show there, such a proof for $G \not\cong H$ essentially plays the role of a CPT-program that, on input $G \uplus H$, constructs two disjoint higher-order objects, one on G and one on H, which are "obviously non-isomorphic"; this means, their non-isomorphism can be witnessed by a $\mathcal{C}^3$-formula. So in this view, the question whether CPT can distinguish two graphs is about the *existence* of a sequence of symmetry-invariant operations that construct a suitable higher-order object, from which the potentially hard-to-spot non-isomorphism of G and H can be read off easily. Now intuitively, it should be the case that if such a sequence of operations exists, then it can be guessed by some universal NPIL-program for graph non-isomorphism. Of course, ∃PIL can also guess such constructions of higher-order objects but this is beside the point: Arguably, the interesting question is whether *symmetric* higher-order objects can distinguish the graphs, and ∃PIL is simply not limited to symmetry-invariant computation.

In short, we suspect that the power of NPIL can be understood like this: For every polynomial bound $p(n)$, there is a universal NPIL-program with that bound, which essentially guesses a deterministic CPT-program with bound $p(n)$, and applies it to the given input structure. This universal NPIL-program in a sense characterises the highest expressive power one can possibly achieve within the bound $p(n)$ in CPT, if one ignores the question of computability of the h.f. sets. Our lower bound results in Chapter 7 and Chapter 9 are based on the *non-existence* of certain symmetric h.f. sets, not on their non-computability in CPT, so our lower bound techniques would even apply to the non-deterministic logic NPIL.

3.3.2 Choiceless Logarithmic Space

A less natural fragment of CPT is Choiceless Logspace. This logic was developed by Grädel and Schalthöfer ([97], [56]) as a h.f.-set-based computation model for the complexity class LOGSPACE. It is a fragment of CPT and the most powerful known logic that is contained in LOGSPACE. However, it can provably not define the Cai-Fürer-Immerman query (see Chapter 5), which is possible in LOGSPACE, and so, it does not capture the complexity class.

The definition of CLogspace is quite involved, so we only explain its most important aspects. The syntax is very similar to that of BGS logic, extended with a variant of the lrec-operator. This *limited recursion* operator comes from the logic LREC, invented by Grohe, Grußien, Hernich, and Laubner ([62], [64]). LREC is an extension of transitive closure logic that captures logspace on directed trees and interval graphs. Its recursion operator allows for more powerful recursion than the transitive closure operator in TC-logic, while ensuring logspace data complexity. It is explicitly added to CLogspace because it is not known whether the iteration terms of CLogspace can simulate it.
To make sure that CLogspace programs can be evaluated in logarithmic space, there is a logarithmic bound on the size of the activated h.f. sets. Now the size of these sets is measured a bit differently than in CPT. In CPT, it is just the sum over the elements in the transitive closure. Now we have to be more careful: For example, storing a singleton set $\{a\}$ naively requires not just one bit, but logarithmically many, because we need to save the name of the atom a. However, one can be more clever: If the atom a is, say FO-definable in $\mathfrak{A}$, then it suffices to know the defining formula, which is part of the CLogspace program and hence has constant size. Generally, indexing the elements of any FO-definable subset $B \subseteq A$ of the universe requires only $\log |B|$ many bits, if the defining formula is known (then our "address space" is just B instead of A). This trick is incorporated into the syntax and semantics of CLogspace, which is a bit technical. Essentially, instead of the term Atoms in BGS, we now have Atoms.φ, which evaluates to the set B of atoms satisfying φ. Each element of this set is then stored using $\log |B|$ many bits. The logic CLogspace can be simulated in CPT and is therefore choiceless. Moreover, it is contained in LOGSPACE, and the inclusion is strict.

Theorem 3.3.3 ([56]).
CLOGSPACE $\leq$ CPT.
LREC $\lneq$ CLOGSPACE $\lneq$ LOGSPACE.

The separation between LREC and CLOGSPACE is shown with a padding argument because LREC $\leq$ FPC, and hence, it does not benefit from padding, but the power of CLogspace depends on the input size, like in CPT. The reason why CLogspace cannot define the CFI-query and hence does not capture logspace is Theorem 6.2.7, which was shown in [40]. It says that deciding the CFI-query in CPT requires the activation of a set with *linear support*. We introduce this notion in the next chapter but for now, it suffices to say that any set activated in CLogspace can have at most logarithmic support. This is because the number of occurring atoms in a set is an upper bound for its support, and every atom requires at least one bit of memory to store (even with the trick we explained above). Thus, the question whether there exists a logic that captures LOGSPACE remains open. As far as we know, there is currently not even a candidate logic that has not been separated from logspace. Defining a suitable logspace-logic might be even harder than the task for polynomial time. As the support argument shows, h.f. sets as data structures for choiceless logspace computation are probably not powerful enough, even though CLogspace already involves a lot of technical fine-tuning to be more space-efficient. Perhaps, progress could be made by using even more of this fine-tuning, but this can quickly get very contrived and unnatural.

3.3.3 Deep Weisfeiler Leman

As already said, the detailed introduction of the computation model Deep Weisfeiler Leman (DWL) is deferred to Chapter 10, where it is actually relevant. Here, we just sketch the idea and explain the motivation that led Grohe, Schweitzer, and Wiebking to this alternative presentation of CPT [63]. The main question they were interested in has to do with the graph isomorphism problem, namely: Does the existence of an efficient algorithm for *graph isomorphism* testing (on a given graph class $\mathcal{K}$) also imply the existence of an efficient *canonisation* algorithm? A canonisation algorithm gets as input an (unordered) graph G and outputs an isomorphic graph on a linearly ordered vertex set. This computed graph, called the *canon* of G, should be exactly the same for all input graphs that are isomorphic. If canonisation is possible efficiently, then so is graph isomorphism testing because one can simply check if the canons of two given graphs are identical or not. The other direction is not clear, at least not for all graph classes. Grohe, Schweitzer, and Wiebking were motivated by the observation that so-called *combinatorial* graph isomorphism algorithms can often be turned into canonisation algorithms (whereas, for the other class of graph isomorphism algorithms, the *group-theoretic* ones, this seems to be harder). It is not clearly defined what a combinatorial graph isomorphism test is, but essentially this refers to symmetry-invariant algorithms such as the k-dimensional Weisfeiler Leman algorithm, or more generally, CPT-definable isomorphism tests. The main result from [63] is that if there is a CPT-program that decides graph isomorphism on some class $\mathcal{K}$, then there also exists a polynomial time (but not choiceless) canonisation algorithm for the graphs in $\mathcal{K}$. For the proof of this, the computation model Deep Weisfeiler Leman was developed. It is equivalent to CPT but highlights its graph distinguishing power better. A DWL-algorithm is a Turing machine with choiceless access to the input structure: Basically, the machine only has access to the colouring information that can be computed with the 2-dimensional Weisfeiler Leman algorithm. It can perform the same computations as any standard Turing machine and it can also add new vertices to the input graph and perform contractions, which corresponds to the creation of ordered tuples and unordered sets in CPT/PIL. The operations of DWL are invariant under the Weisfeiler Leman colouring. This also shows that the computation steps of CPT-programs are invariant not only under automorphisms of the input structure (see Lemma 3.0.5), but also under the $\mathcal{C}^k$-types of vertex-pairs (k being related to the variable rank of the program), which is a stronger restriction. Moreover, the work on DWL demonstrates the relevance of CPT also for the graph isomorphism community: It is not just a purely theoretical concept that emerged in the quest for a PTIME-logic, but it also characterises the important class of combinatorial graph isomorphism algorithms. Therefore, exploring the limitations of CPT for defining the isomorphism problem of Cai-Fürer-Immerman graphs can also be understood as the attempt to show that combinatorial algorithms cannot solve the graph isomorphism problem in general. Indeed, CFI-graphs are easy to distinguish with group-theoretic algorithms (which depend on making choices of generating sets), so they might be a suitable example that separates combinatorial from group-theoretic graph isomorphism tests.

3.3.4 Choiceless Polynomial Time with Witnessed Symmetric Choice

The most recent variation of Choiceless Polynomial Time is its extension with an operator for *witnessed symmetric choice* by Moritz Lichter and Pascal Schweitzer [85]. This continues a line of research on fixed-point logics with different choice constructs. The basic idea is always to allow for non-deterministic choices of tuples from some definable relation (the *choice set*) in each step of the fixed-point induction. Generally, such choices could break symmetries, which is why Gire and Hoang [51] and also Dawar and Richerby [39] considered fixed-point logics with *symmetric choice*. This means that non-deterministic choices are only allowed from sets that are orbits of the structure. Such fixed-point operators define branching computation trees in a given structure but since one always branches over a set of isomorphic choices, the computed fixed-point relations are all isomorphic. Therefore, fixed-point logics with symmetric choice are indeed isomorphism-invariant computation models. The problem is, however, that such computations require to decide whether a given choice set is indeed an orbit or not because in the latter case, a choice from this set must be prevented. This has the same complexity as the graph isomorphism problem and so it is not clear, if such logics with symmetric choice are still contained in PTIME. A solution is *witnessed symmetric choice*, which was suggested in [51] in the context of fixed-point logic. A witnessed symmetric choice operator will only perform a non-deterministic choice from a given choice set, if it additionally receives a list of automorphisms that prove the choice set to be an orbit. So, roughly speaking, a witnessed symmetric choice operator can only give a logic additional power on structures where the logic itself can already define automorphisms. What Lichter and Schweitzer have done in [85] is to define a suitable variant of the witnessed symmetric choice operator for CPT. The resulting logic CPT+WSC has the desirable property that if it can decide the isomorphism problem on a given class of structures, then we get canonisation "for free"; and since canons are always linearly ordered, capturing polynomial time is then easy with the Immerman-Vardi theorem.

Theorem 3.3.4 (Theorem 1 in [85])**.** *If* CPT+WSC *defines isomorphism on a class of structures $\mathcal{K}$ (closed under individualisation), then* CPT+WSC *defines a canonisation of $\mathcal{K}$-structures and captures* Ptime *on $\mathcal{K}$.*

The structure class being closed under individualisation means that one can give an arbitrary element of the universe an individual colour, and the resulting coloured structure is still in the class. The proof idea builds on the aforementioned results from [63]: If CPT defines isomorphism on $\mathcal{K}$, then there is also a non-choiceless canonisation algorithm for $\mathcal{K}$. With the help of the witnessed choice operator, this canonisation algorithm can be implemented in CPT+WSC. Very roughly, the idea is to process the orbits one after another, and to individualise the vertices of each orbit one by one, using the choice operator. However, carrying this out with all details requires substantial technical effort. In total, with the logic CPT+WSC, capturing polynomial time reduces to defining the isomorphism problem. In some sense, this shows that the "bottleneck" in the search for a logic for Ptime (at least for the CPT-approach) is the graph isomorphism problem. The main difficulty is not defining a linear order on the input structure because this can be done with the witnessed choice construct, once we can define the automorphisms of the input structure in the logic (which is equivalent to solving the isomorphism problem). This highlights the importance of studying the power and especially the limitations of CPT to decide isomorphism of the typical benchmark instances such as CFI-graphs or multipedes. It is not clear if CPT+WSC is strictly more powerful than CPT. Moreover, one may ask the question whether the lower bound techniques we develop in this work also have any implications for CPT+WSC. For example, if it could be shown that CPT cannot define isomorphism of certain CFI-graphs due to symmetry reasons, does this also rule out CPT+WSC as a logic for Ptime? We suspect that this is not the case and hence, our ideas for approaching the limitations of CPT do not generalise to CPT+WSC. This is because Corollary 58 in [85] says that CPT+WSC can define the CFI-query on all base graphs where CPT can distinguish orbits (and compute a linear order on the set of orbits). We mainly focus on base graphs that only have one orbit and so, distinguishing orbits is trivial on such instances. Perhaps, hard problems for CPT+WSC can be defined on rigid structures, that have no non-trivial automorphisms, because computing an order on the orbits is not easier than ordering the structure itself then. This is beyond the scope of this thesis, though.

3.3.5 Choiceless Polynomial Time as a fragment of an infinite-set based computation model

There is an article by Bojańczyk and Toruńczyk from 2018 [26] in which they study a computation model with *infinite* nested sets as data structures. They call these sets *hereditarily definable*, and the atoms come from an infinite structure such as $(\mathbb{N}, =)$ or $(\mathbb{Q}, <)$. Hereditarily definable means that the sets may be infinite but have to be definable in a certain "set builder" language. For example, the set $\{\{x, y\} \mid x, y \in \mathbb{N}$ such that $x \neq y\}$ is infinite but definable in $(\mathbb{N}, =)$ and hence has a finite representation (namely the one printed here). What Bojańczyk and Toruńczyk are interested in is a programming language that allows to compute functions on such hereditarily definable sets. Despite the sets being infinite, this programming language guarantees that the programmed functions (called *definable while programs*) are actually computable. In particular, the authors also define a notion of polynomial time computation on these infinite data structures. We do not want to go into the details of this computation model; essentially, what makes computations with infinite sets possible is once again symmetry, or more precisely, *orbit finiteness*. A set with atoms is *orbit-finite* if it has finitely many elements, up to renaming of atoms [25]. In other words, the set can be partitioned into finitely many orbits with respect to the automorphism group of the background structure, say $(\mathbb{N}, =)$. The set above clearly satisfies this: It consists of just one orbit because all pairs $\{x, y\}$, for $x \neq y$, are related via permutations of $\mathbb{N}$. If the set also contained, for example, all singletons $\{x\}$, for all $x \in \mathbb{N}$, then it would consist of two orbits, which is still a finite number. Now the set builder expressions that form the basis of definable while programs ensure that the resulting sets are orbit-finite, similarly as the terms of BGS-logic enforce automorphism-invariance of the CPT-definable sets. It can be shown that orbit-finite sets, i.e. hereditarily definable sets, can be treated as "quasi-finite" in the sense that computations on them are possible in finite time. The motivation for studying computations on orbit-finite sets comes from automata over infinite alphabets, where orbit finiteness also plays an important role to keep the automata "finite". Thus, computation with infinite hereditarily definable sets is per se not a topic that is very related to finite model theory.

However, as the authors show in [26], the logic CPT reappears as the finite fragment of the definable while programming language. Namely, CPT corresponds to definable while programs over sets of *dimension* zero. The dimension of a hereditarily definable set refers to the number of variables that are used in its set builder expression, and the sets of dimension zero are precisely the hereditarily *finite* sets. Thus, CPT may be alternatively defined as the finite restriction of this computation model based on infinite sets with atoms from $(\mathbb{N}, =)$. However, this is arguably a much less natural presentation of CPT than, for example, BGS-logic or PIL. Also, we are not aware of any further lower bound tools which would become available when studying CPT from this different perspective, so we do not explore the world of orbit-finite infinite sets in this thesis.

3.4 Questions that are not addressed in this thesis

We end our survey on CPT with a few questions which are natural to ask but which we neglect in this thesis. We would like to give at least superficial explanations why these directions did not seem promising to us.

3.4.1 Separating CPT from NP

We mainly focus on lower bounds against the choiceless definability of the graph isomorphism problem on unordered Cai-Fürer-Immerman graphs, which is a PTIME-problem. It might be that an easier question to study is trying to separate CPT from NP rather than from P. The reason why we neglected this direction is because we simply did not come up with suitable instances of some NP-complete problem. The CFI-instances that we mostly study are based on unordered n-dimensional hypercubes, which are extremely symmetric. Attempting to separate CPT from NP using our automorphism-based methods would be easier than separating it from P only if we could use instances that are even more symmetric than hypercubes. Complete graphs or edgeless graphs come to mind, but they essentially just encode numbers and have no meaningful combinatorial structure. We are not aware of an NP-complete problem with such instances. Perhaps one could use complete graphs as the basis for some kind of combinatorial construction that yields instances of, say, SAT, which are indistinguishable in $\mathcal{C}^k$. But this will probably be something similar to the CFI-construction, and so we might as well stick with CFI-graphs. An example for a construction of 3SAT-instances that are hard (to approximate) for bounded-variable counting logic and hence FPC is given in [12] by Atserias and Dawar. These instances are based on certain bipartite expanders, whose existence is shown with a randomised construction. Therefore, it is hard to say if they are more symmetric than our n-dimensional hypercubes; we suspect that they are not. Anyway, the 3SAT-instances constructed in [12] are actually obtained from 3XOR-SAT instances, i.e. linear equation systems over $\mathbb{F}_2$. Thus, if one were to try and use the same construction for lower bounds against CPT, one might as well consider the linear equation systems instead of the 3SAT-instances because they are in PTIME. So at least in [12], it seems like it is not easier to prove inexpressibility results for NP-complete problems than it is for problems in P. This is because lower bounds in finite model theory are based on the combinatorial structure (and the symmetries) of the problem instances, and from that perspective, it does not make so much of a difference whether one studies the NP-complete 3SAT problem or the 3XOR-SAT problem in P.

3.4.2 A circuit characterisation of CPT

It was asked in [53] whether there is a characterisation of CPT in terms of symmetric circuit families. This is a perfectly reasonable question because, for example, each polynomial time Turing machine corresponds to a sequence $(C_n)_{n\in\mathbb{N}}$ of P-uniform Boolean circuits, one for each input length, such that the circuits perform the same computation as the Turing machine [8]. Since CPT can be viewed as the symmetrised version of

polynomial time Turing machines, one would expect that there also exists such a sequence of circuits for each CPT-program, and that the symmetry-invariance of CPT should somehow become visible as symmetry of the circuits. In fact, this line of thought led Anderson and Dawar to the discovery of symmetric threshold circuits that characterise fixed-point logic with counting [6]. These circuits, however, are much more symmetric than CPT-computations and are therefore too weak to capture them.
In Chapter 8, we prove that certain h.f. sets over CFI-graphs can be translated into symmetric XOR-circuits. However, this is far from a general circuit characterisation for CPT-programs because our XOR-circuits do not actually simulate the computations of a program but rather capture the structural properties of the h.f. sets activated during a computation. Moreover, our circuit view only makes sense on CFI-graphs and not on general structures.

We suspect that a general symmetric circuit characterisation of CPT will be difficult to find and is perhaps simply not the right approach, at least if we assume a similar setting as in [6] for the FPC-circuits. We have a somewhat informal argument sketch, which illustrates the difficulties when trying to adapt the circuit framework from [6] to CPT. It suggests that already very simple CPT-operations will break symmetries in the corresponding circuits. First of all, we have to briefly introduce the circuit model that Anderson and Dawar use for FPC: They show that, for every fixed FPC-sentence ψ, there is a sequence $(C_n)_{n \in \mathbb{N}}$ of symmetric polynomial-size uniform circuits with Boolean and threshold gates such that for any structure $\mathfrak{A}$, the circuit $C_{|A|}$ applied to $\mathfrak{A}$ outputs whether $\mathfrak{A} \models \psi$. Now it is important what it means to apply a circuit to a structure: For every relation symbol R of arity k in the vocabulary of ψ, the circuit C_n has n^k many input gates, one for each tuple in A^k. Each of these input gates is labelled with a tuple of k natural numbers, i.e. a tuple in $[n]^k$ (this is because the circuit is independent of the structure $\mathfrak{A}$, and so it can depend only on the size $|A|$). Thus, feeding the structure $\mathfrak{A}$ to the circuit requires to first number its universe with the numbers in $[n]$. Then, if e.g. the tuple of elements numbered $(1, 2, 3, ..., k)$ is in $R^{\mathfrak{A}}$, the corresponding input gate for the relation R, that is labelled with $(1, 2, 3, ..., k)$, is set to one. If the tuple with this numbering is not in $R^{\mathfrak{A}}$, then the input gate receives the bit zero. The symmetry of the circuits ensures that it is irrelevant in which order we number the elements of A: For any bijection γ between A and $[n]$, the circuit will output the same result when we feed $\mathfrak{A}$ to the circuit using the numbering γ. For FPC, this is the “right” way to connect structures with circuits. Now we would like to try a similar approach for PIL (which is equivalent to CPT). A computation step in PIL corresponds to the application of an FO- or (FO + H)-interpretation to the current structure. When we translate this into a circuit, it seems natural to expect that the application of the step-interpretation $\mathcal{I}$ should be modelled by a circuit which inputs the current structure $\mathfrak{A}$ and outputs $\mathcal{I}(\mathfrak{A})$, in the same format as described above. We now want to argue that already extremely simple computation steps $\mathcal{I}$ cannot be simulated in this way with symmetric circuits.

Consider a family $(\mathfrak{B}_n)_{n \in \mathbb{N}}$ of padded structures, say graphs. So $\mathfrak{B}_n$ is a structure with vocabulary $\{E, U\}$, where E is the edge relation and U a unary relation that marks the elements of the "true structure", i.e. $U^{\mathfrak{B}_n}$ is the universe of the unpadded graph $\mathfrak{A}_n$ in $\mathfrak{B}_n$. We assume that there is as much padding as we need, so $|\mathfrak{B}_n|$ is at least $|U^{\mathfrak{B}_n}|!$ or even more. For simplicity, we also assume that n is equal to $|\mathfrak{B}_n|$. In PIL, it is extremely easy to obtain the structure $\mathfrak{A}_n$, given $\mathfrak{B}_n$. This is just one computation step, which is carried out by an interpretation $\mathcal{I}$ that simply throws away all elements that are not in U. Now let C_n^U be a circuit that realises this step. That means, C_n^U has n input gates for the relation U, and n^2 input gates for E. The output gates are m^2 many, where m denotes the size of the "true graph" $\mathfrak{A}_n$ in $\mathfrak{B}_n$ (actually, this is already a problem: m does not depend on n but on the structure $\mathfrak{B}_n$, which we do not know at circuit construction time; but in the following we assume that $m = U^{\mathfrak{B}_n}$ is somehow fixed beforehand). The values of these output gates will signify where the edges are in $\mathfrak{A}_n = \mathcal{I}(\mathfrak{B}_n)$. We can assume that the relation U is no longer present in $\mathcal{I}(\mathfrak{B}_n)$. For now, we assume that we have such circuits C_n^U, but they are not necessarily symmetric (they must exist, because computing $\mathcal{I}$ can be done by a polynomial time Turing machine). Next, consider any polynomial time Turing machine M. This can be simulated by a uniform family of (asymmetric) polynomial size circuits, say $(C_n^M)_{n\in\mathbb{N}}$. Here, we gloss over the details and assume that the C_n^M can be applied to structures that are presented in the format we are using the whole time. In particular, we want to feed the graphs $\mathfrak{A}_n$ to these circuits, which amounts to the simulation of M on these graphs as input. It should be possible to symmetrise the C_n^M "by brute force", i.e. by closing them under all permutations of the universe A_n, acting on the input gates of C_n^M. Call the resulting symmetric circuit $\widehat{C}_n^M$. This circuit can be much larger than C_n^M but we may assume that its size is still polynomial in $|\mathfrak{B}_n|$, because of the padding. In total, when we glue together, for each n, the circuits C_n^U and $\widehat{C}_n^M$ (i.e. feeding the output of the former as input to the latter), we obtain a polynomial size circuit, whose second component $\widehat{C}_n^M$ is symmetric. Now if it were possible to symmetrise the circuits C_n^U with only a polynomial blow-up, then we would in total get a polynomial size symmetric circuit that takes as input $\mathfrak{B}_n$, throws away the padding, and simulates M on $\mathfrak{A}_n$. But then, by the correspondence between symmetric circuits and FPC shown in [6], this would be possible in FPC, too. This cannot be the case: We know that FPC cannot simulate arbitrary PTIME-computations on padded structures (e.g. consider CFI-graphs). To be more precise, the result in [6] only shows that *uniform* symmetric circuits can be evaluated in FPC, and our circuits here are not necessarily uniform. However, for non-uniform symmetric circuits, one still obtains definability in infinitary counting logic – and this logic cannot express all PTIME-computations on padded structures, either. In summary, this proof sketch shows that the interpretation $\mathcal{I}$, which simply removes the padding, cannot be simulated by polynomial-size symmetric circuits.

Of course, we are not completely sure if this argument would also go through on a formal level. But if so, then it would mean that already very basic PIL computation steps are not realisable by symmetric circuits, at least not in a framework similar to the one from [6]. Actually, this holds even if we consider a weaker kind of symmetry: The circuits from Anderson and Dawar are always symmetric with respect to $\mathbf{Sym}_n$, whereas CPT-computations are only symmetric with respect to the automorphism group of the given input structure, which is in general smaller. But in our example, these two groups are essentially equal because the automorphism group of a structure that consists almost solely of padding is nearly the full symmetric group on the universe.

The reason why applying an interpretation to a structure, especially one that reduces the size of the universe, might be impossible to implement with small symmetric circuits, is because it seems to require looking at all subsets of the universe of $\mathfrak{B}_n$ of size $|U^{\mathfrak{B}_n}|$ (which we have assumed to be known beforehand): In order to find out where the edges are in $\mathfrak{A}_n$, we have to look at all the input gates for the relation U and check which ones of them are set to 1. These correspond to the elements of $\mathfrak{A}_n$. Then we also have to look up the input bits for the relation E for these particular elements. In a symmetric circuit, we would have one such subcircuit for each subset of $\mathfrak{B}_n$ which can potentially be equal to $U^{\mathfrak{B}_n}$. These are simply too many subsets. Or, put differently, the FPC-circuits from Anderson and Dawar are symmetric with respect to all permutations of the entire input universe, throughout the whole circuit. By contrast, our circuit that removes the padding and then simulates M has two parts: One which is symmetric with respect to the large padded universe, and one which only has to respect the symmetries of the much smaller substructure $\mathfrak{A}_n$. The transition between these two probably has a super-polynomial cost if it is realised by a symmetric sub-circuit.

Thus, it seems like PIL computations and symmetric circuits simply do not match very well. It might be that the presentation of CPT as BGS-logic would be better suited but then the question is still how to deal with dynamically changing universe sizes and symmetries during a computation. Perhaps one would need circuits which have different symmetries in different "stages" but we have no reasonable idea how to proceed in such a direction. Anyway, there is also the problem (which we ignored all the time) that the size of the output of an interpretation depends not only on the size of the input, so it is not clear how interpretations should be simulated by circuits that only depend on the input size and not on the input structure itself. This indicates that probably already the "structure input format" for the circuits from [6] is incompatible with CPT computations.

4 Symmetries and Permutation Groups

The main limitations of Choiceless Polynomial Time stem from the fact that the hereditarily finite sets that it can create are always invariant under the automorphisms of the given input structure, and the more symmetry a structure has, the larger are the constructed objects. This fact can be exploited in order to prove non-definability results. Therefore, we need to introduce the necessary language and the tools for reasoning about symmetries of structures. Most of the time, the relevant concept of symmetry will be captured by *automorphism groups*. Only in Chapter 10, we study a computation model called *Deep Weisfeiler Leman* [63], which is equivalent to CPT in expressive power and highlights the fact that actually, CPT computations are invariant not only under automorphisms but also under $\mathcal{C}^k$*-types*. This type-based symmetry is a more fine-grained notion than what is captured by the automorphism group of a structure, but is also more difficult to work with. At the end of this chapter we take a brief look on how the group-theoretic techniques that we are going to present now could be generalised to the "type-symmetry setting" but our main focus is on automorphism-based symmetry.

4.1 Basic notions from group theory

The content of this section can be found in standard textbooks, for example [45]. A *group* is an algebraic structure $(G, \cdot)$ where G is the set of group elements and $\cdot : (G \times G) \longrightarrow G$ is a binary *associative* operation on G. There exists a unique *neutral element* $e \in G$ that satisfies $e \cdot x = x \cdot e = x$ for every $x \in G$. For every $x \in G$, there exists a unique *inverse* element x^{-1} such that $x \cdot x^{-1} = x^{-1} \cdot x = e$. If the operation $\cdot$ is commutative, then the group is called *Abelian*.
The *order* of a group $(G, \cdot)$ is its cardinality $|G|$. We write $H \leq G$ if H is a *subgroup* of G, i.e. H contains the neutral element and is closed under $\cdot$ and under inverses. The *index* of a subgroup H in G is $[G : H] = \frac{|G|}{|H|}$. The index of H in G is equal to the number of *cosets* of H in G. The (left) cosets of H in G are the sets $gH = \{g \cdot h \mid h \in H\}$, for all $g \in G$. Right cosets are defined analogously as Hg. When we have a chain of subgroups $H_1 \leq H_2 \leq G$, the indices multiply along the chain, i.e. it holds $[G : H_1] = [G : H_2] \cdot [H_2 : H_1]$.
It is well-known that indices of subgroups can only decrease under group homomorphisms. This fact will be needed later in this thesis.

Lemma 4.1.1. *Let $H \leq G$ be groups and let $h : G \longrightarrow G'$ be a homomorphism into some other group. Then $[h(G) : h(H)] \leq [G : H]$.*

Proof. The First Isomorphism Theorem for groups says that $h(G) \cong G/\mathbf{Ker}_G(h)$ and $h(H) \cong H/\mathbf{Ker}_H(h)$. Here, $\mathbf{Ker}_G(h)$ and $\mathbf{Ker}_H(h)$ denote the kernels of h applied to

G and H, respectively. The order of the factor groups $G/\mathbf{Ker}_G(h)$ and $H/\mathbf{Ker}_H(h)$ is $[G : \mathbf{Ker}_G(h)]$ and $[H : \mathbf{Ker}_H(h)]$. So we have $[h(G) : h(H)] = \frac{[G:\mathbf{Ker}_G(h)]}{[H:\mathbf{Ker}_H(h)]} = \frac{|G|\cdot|\mathbf{Ker}_H(h)|}{|H|\cdot|\mathbf{Ker}_G(h)|}$. Since $|\mathbf{Ker}_H(h)| \leq |\mathbf{Ker}_G(h)|$, this is at most $\frac{|G|}{|H|} = [G : H]$. □

We are primarily interested in *permutation groups*: The *symmetric group* on the set $[n] = \{1, 2, ..., n\}$ is denoted $\mathbf{Sym}_n$. This is the group that is formed by all permutations of $[n]$ (i.e. all bijections from $[n]$ to itself) together with the function composition operation $\circ$. When we explicitly write permutations, we use their cycle representation. Every permutation $\pi \in \mathbf{Sym}_n$ can be uniquely decomposed into cycles. For example, when we write $\pi = (123)(45)$, this is the permutation that swaps 4 and 5 and maps 1 to 2, 2 to 3, and 3 to 1.
Generally, for a set Ω, we write $\mathbf{Sym}(\Omega)$ for the group of all permutations on Ω. Sometimes, the *alternating group* $\mathbf{Alt}(\Omega)$ also plays an important role. It is the largest proper subgroup of $\mathbf{Sym}(\Omega)$ (its index in $\mathbf{Sym}(\Omega)$ is just 2) and consists of all permutations with even parity. The parity of a permutation $\pi \in \mathbf{Sym}(\Omega)$ is the number of *inversions* of π. An inversion is a pair $x, y \in \Omega$ such that $x < y$ but $\pi(y) < \pi(x)$, for any fixed linear ordering of Ω. The full symmetric group $\mathbf{Sym}(\Omega)$ is generated by the set of all *transpositions* (i.e permutations that swap a pair of elements while leaving all other elements fixed) (xy), for all $x, y \in \Omega$. The alternating group $\mathbf{Alt}(\Omega)$ is generated by the set of all transpositions of two pairs $(uv)(xy)$, for all pairwise distinct $u, v, x, y \in \Omega$.

We say that $\mathbf{Sym}(\Omega)$ and $\mathbf{Alt}(\Omega)$ *act* on the set Ω (and $\mathbf{Sym}_n$ and $\mathbf{Alt}_n$ act on $[n]$). Informally, this means, that every group element is associated with a permutation of the set Ω. In the cases mentioned above, the group elements *are* obviously permutations of the set Ω. More generally, the *action* of a group G on a set Ω is a function from $\Omega \times G$ into Ω that specifies for each group element $\pi \in G$, and each set element $a \in \Omega$, what the image $\pi(a) \in \Omega$ is. The group action must satisfy that, if $\mathrm{id} \in G$ is the neutral element, then $\mathrm{id}(a) = a$ for all $a \in \Omega$, and $\pi(\pi'(a)) = (\pi \circ \pi')(a)$ for all $\pi, \pi' \in G$, $a \in \Omega$. So a group action can be identified with a group homomorphism $h : G \longrightarrow \mathbf{Sym}(\Omega)$. An action is called *faithful* if the kernel of h is trivial, i.e. if the neutral element of G is the only one that is mapped to the identity permutation. This is usually the case in this thesis. Often, the homomorphism h is clear from the context and not explicitly given. For example, every subgroup of $\mathbf{Sym}(\Omega)$ acts on Ω via the identity homomorphism. For a permutation group G acting on some set Ω, the *degree* of G refers to $|\Omega|$, i.e. the size of the permutation domain.

Sometimes we consider groups which are obtained by composing other groups in certain ways, namely by taking *direct* or *semi-direct* products:

Definition 4.1.2 (Products of groups)**.** *Let $(G, \cdot), (H, \cdot)$ be groups. Their* direct *product $G \times H$ is their cartesian product with element-wise group operation, i.e. $(g, h) \cdot (g', h') = (g \cdot h, g' \cdot h')$ for every $(g, h), (g', h') \in G \times H$. The direct product of multiple groups $G_1, ..., G_m$ is written $\Pi_{i=1}^m G_i$.*

The semi-direct product *of G* by *H is $G \rtimes H$. For this to be defined, there must be an action of H on G, i.e. for every $\pi \in H$, the mapping $u \mapsto \pi u$ is an automorphism of G. Then $G \rtimes H$ is the cartesian product of G and H with the group operation defined by:*

$$(u, \pi) \cdot (v, \pi') := (u \cdot \pi^{-1}(v), \pi \cdot \pi').$$

Consequently, we have for the inverses: $(u, \pi)^{-1} = ((\pi(u))^{-1}, \pi^{-1})$.

An example for a semi-direct product is the automorphism group of the n-dimensional hypercube (see Section 5.3) and also the automorphism groups of unordered CFI-graphs (see Section 5.2).

The *automorphism group* of a finite structure is the set of isomorphisms from the structure into itself: Let $\mathfrak{A}$ be a relational τ-structure with universe A. A permutation $\pi \in \mathbf{Sym}(A)$ is an *automorphism* of $\mathfrak{A}$ if for every $R \in \tau$, we have $(a_1, ..., a_k) \in R^{\mathfrak{A}}$ if and only if $(\pi a_1, ..., \pi a_k) \in R^{\mathfrak{A}}$. The automorphisms of any structure $\mathfrak{A}$ form a subgroup of $\mathbf{Sym}(A)$, denoted $\mathbf{Aut}(\mathfrak{A})$. The groups $\mathbf{Sym}(A)$ and $\mathbf{Aut}(\mathfrak{A})$ act naturally not only on A but also on $\mathrm{HF}(A)$, the *hereditarily finite objects* with atoms in A: If $x \in \mathrm{HF}(A)$ is a set, then $\pi x = \{\pi y \mid y \in x\}$.

A concept that is of very high importance when it comes to lower bounds for Choiceless Polynomial Time is that of *orbits* and *stabiliser* subgroups. Whenever a group G acts on some set A, then the binary relation that contains all pairs $(x, y) \in A^2$ such that there exists $\pi \in G$ with $y = \pi x$ is an equivalence relation on A. Its equivalence classes are called *orbits*. For $a \in A$, we usually write

$$\mathbf{Orb}_G(a) := \{\pi a \mid \pi \in G\}$$

to denote the G-orbit of the element a. Thus, the orbits of G's action on A form a partition of A. If this partition has A as its only part, then the action is called *transitive*, and otherwise *intransitive*. Usually, we will consider situations where A is the universe of some structure $\mathfrak{A}$, extended with hereditarily finite sets, and $G = \mathbf{Aut}(\mathfrak{A})$. The *stabiliser* of an element $a \in A$ is the subgroup of G that fixes a:

$$\mathbf{Stab}_G(a) := \{\pi \in G \mid \pi a = a\}.$$

We also consider stabilisers of subsets of A. In this case, two notions have to be distinguished: The *pointwise* and the *setwise* stabiliser of a set $S \subseteq A$. The setwise stabiliser is the subgroup that induces permutations of the set S:

$$\mathbf{Stab}_G(S) := \{\pi \in G \mid \pi(S) = S\},$$

and the pointwise stabiliser is the group that fixes every element in S:

$$\mathbf{Stab}^{\bullet}_G(S) := \{\pi \in G \mid \pi a = a \text{ for every } a \in S\}.$$

In this thesis, we are frequently concerned with bounding the orbit size of certain objects. It is often more convenient to estimate the size of the stabiliser of an object instead of its orbit. The fundamental *Orbit-Stabiliser Theorem* (see e.g. [45]) states that stabiliser and orbit size are indeed related:

Theorem 4.1.3 (Orbit-Stabiliser). *Let G be a group acting on a set A. Let $a \in A$.*

$$|\mathbf{Orb}_G(a)| = [G : \mathbf{Stab}_G(a)] = \frac{|G|}{|\mathbf{Stab}_G(a)|}.$$

In order to apply this theorem later on, we need to introduce some notions and tools that will help us to approximate the size of the relevant stabiliser groups.

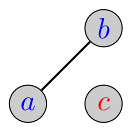
4.2 Supports and supporting partitions

A *support* of a permutation group G acting on some set A is a subset $S \subseteq A$ such that any permutation that fixes every element in S is in G.

Definition 4.2.1 (Support). *Let A be a set and let G be a group acting on A. A* support *of a subgroup $H \leq G$ is a subset $S \subseteq A$ such that $\mathbf{Stab}^\bullet_G(S) \leq H$.*

In our applications, G will usually be the automorphism group of a structure $\mathfrak{A}$, and H will be the stabiliser of a hereditarily finite object a over $\mathfrak{A}$. In that case, when we say that S is a support of a, we mean that S is a support of $\mathbf{Stab}(a)$ in $\mathbf{Aut}(\mathfrak{A})$. The size of the *smallest support* of H is a "group-parameter" that will often be of interest for us. Some groups have the property that every subgroup has a unique minimal support. This is the case if the supports of a subgroup are closed under intersections. For example, $\mathbf{Sym}_n$ has this property, as shown in [21]. Whenever a h.f. set a over a structure $\mathfrak{A}$ has a unique minimal support, we denote it by $\sup(a)$.

Example 4.2.2.
Let $\mathfrak{A} = (V = \{a, b, c\}, E = \{\{a, b\}\})$ be the graph given in the picture. For any h.f. set over $\mathfrak{A}$, the set of atoms occurring in it is a support. For instance, a support of $x := \{\{a\}, c\}$ is the set $\{a, c\}$ because fixing these atoms also fixes x. This support is not of minimum size. A smaller support of x would be $\{a\}$, because c is anyway fixed by every automorphism of $\mathfrak{A}$. This minimum support is not unique. An alternative is $\{b\}$.



Sometimes, when we are dealing with subgroups of the symmetric group, we use an even finer notion, called *supporting partition*. The idea is the same as with supports, only that a more complex object, namely a partition instead of a set, is considered. The pointwise and setwise stabiliser of a partition are as defined above, where we view the partition as the set of its parts. To make it clearer: Let A be a set and $\mathcal{P}$ be a partition of A.

- The *pointwise* stabiliser of $\mathcal{P}$ is $\mathbf{Stab}^\bullet_{\mathbf{Sym}(A)}(\mathcal{P}) := \{\pi \in \mathbf{Sym}(A) \mid \pi(P) = P \text{ for all } P \in \mathcal{P}\}$.

- The *setwise* stabiliser of $\mathcal{P}$ is $\mathbf{Stab}_{\mathbf{Sym}(A)}(\mathcal{P}) := \{\pi \in \mathbf{Sym}(A) \mid \pi(P) \in \mathcal{P} \text{ for all } P \in \mathcal{P}\}$ (these are all $\pi \in \mathbf{Sym}(A)$ that induce a permutation on the parts of $\mathcal{P}$).

The following definitions and lemmas are from [6].

Definition 4.2.3 (Supporting partition). *Let A be a set and $G \leq \mathbf{Sym}(A)$ be a permutation group acting on A. A* supporting partition *of G is a partition $\mathcal{P}$ of A such that $\mathbf{Stab}^{\bullet}_{\mathbf{Sym}(A)}(\mathcal{P}) \leq G$.*

In words: Every permutation that maps each part P of $\mathcal{P}$ to itself (while possibly permuting the elements of P) is in G. Note that if $S = \{s_1, ..., s_k\} \subseteq A$ is a *support* of G, then $\{\{s_1\}, \{s_2\}, ..., \{s_k\}, A \setminus S\}$ is a *supporting partition* of G. In general, supporting partitions can be more informative than supports, especially if we consider the *coarsest* possible one. A partition $\mathcal{P}'$ is *as coarse as* $\mathcal{P}$ (denoted $\mathcal{P}' \sqsupseteq \mathcal{P}$) if every part of $\mathcal{P}$ is contained in a part of $\mathcal{P}'$.

Lemma 4.2.4 (Lemma 1 in [6]). *Each permutation group $G \leq \mathbf{Sym}(A)$ has a unique* coarsest supporting partition*, denoted $\mathbf{SP}(G)$.*

Again, when we write $\mathbf{SP}(a)$ for some object $a \in \mathrm{HF}(A)$, we mean $\mathbf{SP}(\mathbf{Stab}(a))$. The pointwise and setwise stabilisers of the coarsest supporting partition of a permutation group G can be used to approximate G "from below" and "from above":

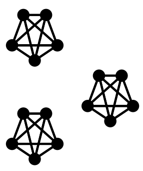
Lemma 4.2.5 (Lemma 4 in [6]). *Let $G \leq \mathbf{Sym}(A)$ Then:*

$$\mathbf{Stab}^{\bullet}(\mathbf{SP}(G)) \leq G \leq \mathbf{Stab}(\mathbf{SP}(G)).$$

Some groups G are sandwiched rather tightly between these two stabilisers. These are the ones whose size is easy to analyse because they can essentially be viewed as stabilisers of partitions. On the other hand, the gap between $\mathbf{Stab}^{\bullet}(\mathbf{SP}(G))$ and $\mathbf{Stab}(\mathbf{SP}(G))$ can also get arbitrarily large: For example, let $G = \mathbf{Alt}_n$ be the alternating group on $[n]$. Then we have $\mathbf{SP}(G) = \{\{1\}, \{2\}, ..., \{n\}\}$, because no transposition is in $\mathbf{Alt}_n$. Whenever the coarsest supporting partition consists only of singletons, we can learn nothing from it because its pointwise stabiliser contains only the identity permutation, and its setwise stabiliser is the whole group $\mathbf{Sym}_n$, so G could be any group in this case (we can only infer that it contains no transpositions).

Example 4.2.6.

Let G be the graph in the picture. Suppose its vertex set is $[n]$, where n is a number divisible by three, and it consists of three cliques of equal size. The coarsest supporting partition of $\mathbf{Aut}(G)$, viewed as a subgroup of $\mathbf{Sym}_n$, is $\{\{1, ..., \frac{n}{3}\}, \{\frac{n}{3}+1, ..., \frac{2n}{3}\}, \{\frac{2n}{3}+1, ..., n\}\}$. In this case, we have $\mathbf{Aut}(G) = \mathbf{Stab}(\mathbf{SP}(\mathbf{Aut}(G)))$, so $\mathbf{Aut}(G)$ is maximal with this supporting partition according to Lemma 4.2.5. If, for example, each clique in the graph had a distinct vertex colour, then $\mathbf{Aut}(G)$ would be equal to $\mathbf{Stab}^{\bullet}(\mathbf{SP}(\mathbf{Aut}(G)))$. This would be the other extreme case of the lemma. If only one clique were coloured and the other two were not, then $\mathbf{Aut}(G)$ would be strictly in the middle between the pointwise and the setwise stabiliser of its supporting partition.



In Chapter 9, the supporting partitions from [6] will no longer suffice for our purposes, and we will use a variant of them, that we call *alternating supporting partitions*.

Definition 4.2.7 (Alternating supporting partition)**.** *Let A be a set and $G \leq \mathbf{Sym}(A)$ be a permutation group acting on A. An* alternating supporting partition *of G is a partition $\mathcal{P}$ of A such that*

$$\prod_{\substack{P \in \mathcal{P} \\ |P| < 5}} \mathbf{Sym}(P) \times \prod_{\substack{P \in \mathcal{P} \\ |P| \geq 5}} \mathbf{Alt}(P) \leq G.$$

The difference to the "standard" supporting partitions is that the odd permutations within the parts of an alternating supporting partition need not be contained in the supported group G. On parts of size < 5, we require the full symmetric group to be in G, because otherwise, the proof of the next lemma is problematic, and in our applications later on, constant-size parts will not play a big role anyway. One of the benefits of alternating supporting partitions is that the undesirable case mentioned above cannot occur: If $G = \mathbf{Alt}_n$, then G has an *alternating* supporting partition with just one part, which will be much more helpful than a supporting partition consisting only of singletons. In particular, as we show soon, the groups that we will study later on all have alternating supporting partitions with at most a sublinear number of singleton parts. Knowing this will be very convenient in some of our proofs.

First of all, we have to verify that alternating supporting partitions work just like the original ones from [6].

Lemma 4.2.8. *Each permutation group $G \leq \mathbf{Sym}(A)$ has a unique* coarsest alternating supporting partition*, denoted $\mathbf{SP}_A(G)$.*

Proof. The proof is similar to the one of Lemma 1 in [6]. We need to prove that for any two alternating supporting partitions $\mathcal{P}, \mathcal{P}'$, the finest partition of which both of them are refinements is still an alternating supporting partition. Then the lemma follows directly. This "finest common coarsification" of $\mathcal{P}$ and $\mathcal{P}'$, denoted $\mathcal{E}(\mathcal{P}, \mathcal{P}')$ is defined like this: Let $\sim \subseteq A^2$ be the transitive closure of the relation "a and b occur together in some part P of $\mathcal{P}$ or $\mathcal{P}'$". The equivalence classes of $\sim$ are the parts of $\mathcal{E}(\mathcal{P}, \mathcal{P}') =: \mathcal{E}$. We want to show that any even permutation within any part Q of $\mathcal{E}$ (that pointwise fixes everything outside of Q) is also in G. We do this by proving: If $P \in \mathcal{P}, P' \in \mathcal{P}'$ have non-empty intersection, then $\mathbf{Alt}(P \cup P') \leq G$. Since $\mathbf{Alt}(P \cup P')$ is generated by pairs of transpositions $(xy)(x'y')$, it suffices to show that all such pairs are in G. So consider $(xy)(x'y')$, for $x, y, x', y' \in P \cup P'$ pairwise distinct. We distinguish the following cases: If x, x', y, y' are all in P (or analogously, in P'), then $(xy)(x'y') \in \mathbf{Alt}(P) \leq G$ (using that $\mathcal{P}$ is an alternating supporting partition of G).
The next case is that $x \in P \setminus P'$, $y \in P' \setminus P$, and x', y' are in the same part, say, both are in P'. In this case, let $z \in P \cap P'$, and $a, b \in P \setminus \{x, z\}$ be two distinct elements (if such a, b do not exist, then $|P| \leq 3$, and so $\mathbf{Sym}(P) \leq G$, which means that the following argument works even without these a, b). It holds $(xy)(x'y') = (xz)(ab)(zy)(x'y')(xz)(ab)$. The number of transpositions within P and within P' is even (we always swap a and b

when we swap x and z, and we swap x' and y' together with (zy)), so this product is in $\mathbf{Alt}(P) \times \mathbf{Alt}(P')$ and therefore in G, because both $\mathcal{P}$ and $\mathcal{P}'$ are supporting partitions of G.
Another case is that $x, x' \in P \setminus P'$ and $y, y' \in P' \setminus P$. Let again $z \in P \cap P'$, and fix some $a, b \in P \setminus \{x, x', z\}$ and $a', b' \in P' \setminus \{y, y', z\}$. Again, if this is not possible, then P and P' are smaller than 5 and so all permutations on them are in G, which makes the next step only easier. Consider $(xz)(ab)(zy)(a'b')(xz)(ab)(x'z)(ab)(zy')(a'b')(x'z)(ab)$. This is equal to $(xy)(x'y')$ and again in $\mathbf{Alt}(P) \times \mathbf{Alt}(P')$ and hence in G.
It remains the case where $x, y \in P$ and $x', y' \in P'$. We can assume that $x, y \in P \setminus P'$ and $x', y' \in P'$ because $\mathbf{Alt}(P) \times \mathbf{Alt}(P') \leq G$, and so we can move the elements that are to be swapped anywhere within P, P', respectively. Then we can simulate the permutation $(xy)(x'y')$ by applying the previous case twice: First, we execute $(xx')(yy')$, and then $(xy')(yx')$. Both are in $\mathbf{Alt}(P) \times \mathbf{Alt}(P')$, as shown above, and hence, in total, we have $(xy)(x'y') \in \mathbf{Alt}(P) \times \mathbf{Alt}(P') \leq G$.

So we have shown that we can take the union of two intersecting parts from $\mathcal{P}$ and $\mathcal{P}'$, and the alternating group on this union will also be in G. Iterating this, we can show that $\mathbf{Alt}(Q) \leq G$, for every $Q \in \mathcal{E}$, because all parts of $\mathcal{E}$ can be obtained by iteratively taking the union of intersecting parts of $\mathcal{P}$ and $\mathcal{P}'$. It remains to show that for all $Q \in \mathcal{E}$ with $|Q| < 5$, $\mathbf{Sym}(Q) \leq G$. But this is clear since such parts $Q \in \mathcal{E}$ can only be the union of small parts $P \in \mathcal{P}$ and $P' \in \mathcal{P}'$. Then we have $\mathbf{Sym}(P) \leq G$ and $\mathbf{Sym}(P') \leq G$. We can easily see that every transposition in $\mathbf{Sym}(P \cup P')$ is also in G by arguing as above, just that we do not need dummy-transpositions anymore in order to keep the sign even. □

Lemma 4.2.9 (variation of Lemma 3 in [6] for alternating supporting partitions)**.** *For any $G \leq \mathbf{Sym}(A)$, and any $\sigma \in \mathbf{Sym}(A), \sigma\mathbf{SP}_A(G) = \mathbf{SP}_A(\sigma G \sigma^{-1})$.*

Proof. As mentioned in [6], it holds $\sigma\mathcal{E}(\mathcal{P}, \mathcal{P}') = \mathcal{E}(\sigma\mathcal{P}, \sigma\mathcal{P}')$ for any $\sigma \in \mathbf{Sym}(A)$. Therefore, it remains to show that for any alternating supporting partition $\mathcal{P}$ of G, and any $\sigma \in \mathbf{Sym}(A)$, $\sigma\mathcal{P}$ is an alternating supporting partition of $\sigma G \sigma^{-1}$. So let

$$\pi \in \prod_{\substack{P \in \sigma\mathcal{P} \\ |P| < 5}} \mathbf{Sym}(P) \times \prod_{\substack{P \in \sigma\mathcal{P} \\ |P| \geq 5}} \mathbf{Alt}(P).$$

Then

$$(\sigma^{-1}\pi\sigma) \in \prod_{\substack{P \in \mathcal{P} \\ |P| < 5}} \mathbf{Sym}(P) \times \prod_{\substack{P \in \mathcal{P} \\ |P| \geq 5}} \mathbf{Alt}(P).$$

Therefore, $(\sigma^{-1}\pi\sigma) \in G$ because $\mathcal{P}$ is an alternating supporting partition of G. Consequently, $\pi \in \sigma G \sigma^{-1}$, and so, $\sigma\mathcal{P}$ is an alternating supporting partition of $\sigma G \sigma^{-1}$. □

Lemma 4.2.10 (Lemma 4 in [6] for alternating supporting partitions)**.** *Let $G \leq \mathbf{Sym}(A)$ Then:*

$$\prod_{\substack{P \in \mathbf{SP}_A \\ |P| < 5}} \mathbf{Sym}(P) \times \prod_{\substack{P \in \mathbf{SP}_A \\ |P| \geq 5}} \mathbf{Alt}(P) \leq G \leq \mathbf{Stab}(\mathbf{SP}_A(G)).$$

Proof. The first part is by definition of alternating supporting partitions. For the second part, let $\sigma \in G$. Then $\sigma G \sigma^{-1} = G$. So $\sigma \mathbf{SP}_A(G) = \mathbf{SP}_A(G)$ by the preceding lemma. $\square$

Thus, every group has a unique coarsest alternating supporting partition and is sandwiched between its pointwise and setwise stabiliser.

4.3 Every large permutation group contains a product of large alternating groups

Now we prove a result that inspired the introduction of alternating supporting partitions. Essentially, this is based on a version of Theorem 5.2A from the Dixon and Mortimer textbook on permutation groups [45]. Informally, that theorem says that any group $G \leq \mathbf{Sym}_n$ with index $[\mathbf{Sym}_n : G] \leq \binom{n}{r}$ acts as the alternating group on a large part of its permutation domain. This has also been used to study the limitations of symmetric circuits (e.g. in [42], [41]). However, in our application in Chapter 9, we are dealing with groups $G \leq \mathbf{Sym}_n$ whose index is bounded by some polynomial in 2^n. This is greater than $\binom{n}{r}$ for any r. Therefore, Theorem 5.2A from [45] is not applicable and we have to "manually" prove a similar result for groups of larger index. This variant of Theorem 5.2A (Lemma 4.3.8) can be used to show the main result of this section:

Theorem 4.3.1. *Let $k \in \mathbb{N}$ be a constant and $(G_n)_{n \in \mathbb{N}}$ be a family of groups such that $G_n \leq \mathbf{Sym}_n$ and for all large enough n, $[\mathbf{Sym}_n : G_n] \leq 2^{nk}$. Then the number of singleton parts in $\mathbf{SP}_A(G_n)$ grows at most sublinearly. In other words: There is no constant $0 < c \leq n$ such that for all large enough n, there exists a $\Delta_n \subseteq [n]$ of size $|\Delta_n| \geq cn$ on which $\mathbf{SP}_A(G_n)$ contains only singleton parts.*

We make use of this result only in Chapter 9. It could also be applied in Chapter 7, which is based on [87], to skip a case in the proof there. However, we found this more general result only much later than the publication of [87], so we keep that chapter in line with the corresponding CSL'21 paper.

For the proof of Theorem 4.3.1, we need some group-theoretic prerequisites. The following can also be found in the Dixon and Mortimer textbook [45], whose notation we adopt. For a group $G \leq \mathbf{Sym}(\Omega)$, and a subset $\Delta \subseteq \Omega$, $G^{(\Delta)}$ denotes the pointwise stabiliser of Δ in G, i.e. $G^{(\Delta)} = \mathbf{Stab}^{\bullet}_G(\Delta)$. If Δ is a union of orbits of G, then we write G^{Δ} to denote the restriction of G to its action on Δ. This is a subgroup of $\mathbf{Sym}(\Delta)$.

As mentioned before, a group $G \leq \mathbf{Sym}(\Omega)$ acts *transitively* on Ω if every element can be mapped to every other element by G, so if Ω is itself an orbit. If G acts transitively on Ω, then a non-empty set $\Delta \subseteq \Omega$ is called a *block* if for each $\pi \in G$, $\pi(\Delta) = \Delta$ or $\pi(\Delta) \cap \Delta = \emptyset$. A *block system*, or *system of imprimitivity*, is a partition of Ω into blocks (of equal size). The group G acts as a permutation group on the set of blocks because it always maps blocks to blocks. Every transitive group has the trivial block systems in which each point forms a singleton block, or the whole point set is one block, respectively. If a transitive group G has other block systems than these two, then G is called *imprimitive*, and otherwise, *primitive*. In particular, primitive groups are always transitive.

A subgroup $N \leq G$ is called *normal* (denoted $N \lhd G$) if its right and left cosets coincide, i.e. if $\gamma N = N\gamma$ for every $\gamma \in G$. An equivalent formulation is that for all $g \in G, h \in N$, we have $ghg^{-1} \in N$. We are interested in normal subgroups because they can be factored out: If $N \lhd G$, then G/N is the group whose elements are the cosets of N, that is: For any two $\gamma N, \gamma' N$, the group operation in the factor group G/N is defined as $(\gamma N) \circ (\gamma' N) = (\gamma \circ \gamma')N$. Thus, the order $|G/N|$ of the factor group is equal to the index $[G : N]$, and so, $|G| = |N| \cdot |G/N|$.

If $G \leq \mathbf{Sym}(\Omega)$ is intransitive and $\Delta \subseteq \Omega$ is an orbit of G, then $G^{(\Delta)}$, the pointwise stabiliser of the orbit, is a normal subgroup of G, as one can easily verify. The factor group $G/G^{(\Delta)}$ is isomorphic to G^{Δ}, the action of G on Δ. Also, if G has a non-trivial block system, then the subgroup of G that fixes every block setwise is normal in G. Factoring out this stabiliser yields a group that is isomorphic to the action of G on the blocks. Thus, if G is intransitive or imprimitive, it has these mentioned "canonical" normal subgroups. These can then be factored out, which is useful in inductive proofs. The only case where it is not clear how to factor out a normal subgroup is if G is primitive. Note that the primitive cases $G = \mathbf{Sym}(\Omega)$ or $G = \mathbf{Alt}(\Omega)$ are not difficult: The symmetric group has the alternating group as a normal subgroup, which leaves $\mathbb{Z}_2$ when it is factored out. The alternating group is *simple*, which means that it only has itself and the trivial group $\{1\}$ as normal subgroups. The other primitive cases are less clear but luckily, the finite primitive groups have been classified completely. For our proof, the following theorem by Babai, which essentially summarises the relevant primitive cases, is sufficient:

Theorem 4.3.2 (Theorem 3.2.1 in [15]). *Let $G \leq \mathbf{Sym}_n$ be a primitive group of order $|G| \geq n^{1+\log n}$ where n is greater than some absolute constant. Then G has a normal subgroup N of index $\leq n$ such that N has a system of imprimitivity on which N acts as a Johnson group $\mathbf{Alt}_k^{(t)}$ with $k \geq \log n$.*

Note that the theorem in [15] has a typo in the order of G, which we have corrected here. The *Johnson group* $\mathbf{Alt}_k^{(t)}$ is isomorphic to $\mathbf{Alt}_k$, the alternating group on k elements, but $\mathbf{Alt}_k^{(t)}$ acts on the set of all t-tuples over $[k]$ (in the natural way). So the above theorem guarantees the existence of a normal subgroup N in any large enough primitive group, and moreover, it tells us that N more or less looks like an alternating group. This will essentially be one of the base cases in the proof of Theorem 4.3.1.
Before we can start with that proof, we need one more concept, namely the *composition series* of a group G. This is a series $1 = H_0 \lhd H_1 \lhd ... \lhd H_n = G$ such that each H_i is a maximal proper normal subgroup of H_{i+1}. The factors H_{i+1}/H_i are called the *composition factors* of G. Every finite group has such a composition series, which is not necessarily unique. But by the Jordan-Hölder theorem, every composition series yields the same composition factors (see for example [94]). Therefore, no matter in which order

we factor out normal subgroups of a given group G, we will eventually encounter the same composition factors (just like in the prime factorisation of a natural number). This holds even if we do not factor out a *maximal* normal subgroup in each step. Therefore, it holds:

Lemma 4.3.3. *Let G be a group and H be a composition factor of G. Let $N \lhd G$ be a normal subgroup. Then $H \cong G/N$, or H is a composition factor of N or of G/N.*

Proof. If N is a maximal normal subgroup in G, then there exists a composition series of G of the form $1 \lhd ... \lhd N \lhd G$. Then either $H = G/N$, or H appears as a composition factor later in the series, which means that it is a composition factor of N. If N is not a maximal normal subgroup in G, then we have $N \lhd N_1 \lhd ... \lhd N_k = G$ for $k \geq 1$ normal subgroups of G containing N. Then either H is a composition factor of N, or if it is not, then it must be equal to N_{i+1}/N_i, for some $i \in [k]$. By the Third Isomorphism Theorem, $1 \lhd N_1/N \lhd ... \lhd N_k/N \lhd G/N$ is a composition series of G/N, and $(N_{i+1}/N)/(N_i/N) \cong H$, so H is a composition factor of G/N in this case. □

What we will also need is that a group which is alternating on one of its orbits A is either still alternating on A when the rest is fixed pointwise, or the action on A is always completely determined by the action outside of A. For the proof idea of this lemma, I thank Daniel Wiebking.

Lemma 4.3.4. *Let $H \leq G \leq \mathbf{Sym}_n$ and let A be an orbit of H such that $\mathbf{Alt}(A) \leq H^A$. Then either, $\mathbf{Alt}(A) \leq (H^{([n]\setminus A)})^A$, or for every $h \in H$, the action of h on $[n] \setminus A$ also determines the action of h on A. The latter means that there are no two distinct $g, h \in H$ which induce the same permutation on $[n] \setminus A$ but distinct permutations on A.*

Proof. It holds that $N := (H^{([n]\setminus A)})^A$ is a subgroup of H^A. Moreover, this subgroup is normal. To see this, let $h \in N, g \in H^A$. We want to show that $ghg^{-1} \in N$. There exist $h' \in H^{([n]\setminus A)}, g' \in H$ such that h', g' are extensions of h and g, i.e. their restriction to A corresponds to h, g, respectively. It is clear that $g'h'g'^{-1} \in H^{([n]\setminus A)}$ because this permutation fixes every point outside of A. Therefore, $ghg^{-1} \in N$, because this is just the action of $g'h'g'^{-1}$ on A. So $N \lhd H^A$. Since $\mathbf{Alt}(A) \leq H^A$, H^A is either $\mathbf{Alt}(A)$ or $\mathbf{Sym}(A)$. If it is $\mathbf{Alt}(A)$, then N is either also $\mathbf{Alt}(A)$ or the trivial group $\{1\}$, because $\mathbf{Alt}(A)$ has no other normal subgroups. If $H^A = \mathbf{Sym}(A)$, then N is trivial, $N = \mathbf{Alt}(A)$, or $N = \mathbf{Sym}(A)$. So if N is not trivial, then $\mathbf{Alt}(A) \leq N = (H^{([n]\setminus A)})^A$. Otherwise, if N is trivial, then every permutation in $H^{([n]\setminus A)}$ also fixes A pointwise. It follows that there do not exist any two distinct $g, h \in H$ such that g, h are equal on $[n] \setminus A$ but different on A. If they existed, then $gh^{-1} \in H^{([n]\setminus A)}$, but gh^{-1} is not the identity on A. □

Now we will prove the main technical result of this section. It essentially says that if a group G has a large alternating group as a *composition factor*, then it also has this large alternating group as a *subgroup* in some sense, or otherwise, the index of G in $\mathbf{Sym}_n$ must be large. The proof is by induction on the compositional structure of G, i.e. in the inductive step, we choose a normal subgroup and factor it out and then continue inductively with the normal subgroup or with the factor group, in the spirit of Lemma

4.3.3. With the next lemma, we can prove Theorem 4.3.1 using a fact from the literature: Every large group must also have a large alternating group as a composition factor. So the key step is the one from composition factor to subgroup. Again, I thank Daniel Wiebking for his help with the proof, especially for solving the primitive case.

Lemma 4.3.5. *Let $0 < c \leq 1$ be a constant. Let $(G_n)_{n \in \mathbb{N}}$ be a family of groups such that for all n, $G_n \leq \mathbf{Sym}_d$, where $c \cdot n \leq d \leq n$ (to be precise: this can be a different d for every n), and such that for all large enough n, G_n has a composition factor isomorphic to $\mathbf{Alt}_m$, for some $m \geq c \cdot n$. Then, for every large enough n, one of following two cases can arise:*

(i) There exists a subgroup $H_n \leq G_n$ and an orbit A of H_n with $|A| \geq c \cdot n$ such that $\mathbf{Alt}(A) \leq H_n^{([d] \setminus A)}$.

(ii) $[\mathbf{Sym}_d : G_n] \geq \frac{1}{n} \cdot \left(\frac{2d \cdot \alpha}{e}\right)^{cn/2} \cdot \alpha^\ell$, where ℓ is the number of occurrences of the transitive imprimitive case in the recursion starting with G and ending with the composition factor $\mathbf{Alt}_m$ or in another non-recursive case. The factor α is $1/(\lfloor c^{-1} \rfloor !)$.

Proof. Fix $n \in \mathbb{N}$ and let d be such that $G_n \leq \mathbf{Sym}_d$ (we suppress the subscript n in the following). We prove the lemma by induction on the compositional structure of G and choose a normal subgroup that we factor out in each step, until we arrive at the composition factor $\mathbf{Alt}_m$, which occurs in G according to the assumption of the lemma. If $G = \mathbf{Alt}_m$, then we are in case (i) and are done. Otherwise, G must have a normal subgroup because else, G would be simple and would not contain the composition factor $\mathbf{Alt}_m$. We choose this subgroup depending on which of the following is the case:

Case 1: G is intransitive.
Let $\Omega \subseteq [d]$ be an arbitrary orbit with $|\Omega| \geq cn$. Such an orbit must exist because otherwise, G cannot have $\mathbf{Alt}_m$ for $m \geq cn$ as a composition factor. To see this, consider a chain of normal subgroups that pointwise fix an orbit, one after the other. The corresponding factor groups are always the restrictions of the next normal subgroup to one orbit, and so they can never contain $\mathbf{Alt}_m$ if all orbits are too small. Eventually, we have fixed every orbit pointwise, which leads to the trivial group, and this cannot contain $\mathbf{Alt}_m$, either. Therefore, at least one large enough orbit Ω must exist.
Let $N := G^{(\Omega)}$ be the pointwise stabiliser of that orbit. It holds $N \lhd G$, and the factor G/N is isomorphic to G^Ω, the action of G on Ω. Let $\Delta := [d] \setminus \Omega$ be the complement of the orbit. Now we apply Lemma 4.3.3. It tells us that the large alternating group which must appear as a composition factor in G is either isomorphic to G/N, or it is a composition factor of N or of G/N.

If $\mathbf{Alt}_m$ is a composition factor of N, then we apply the inductive hypothesis to $N^\Delta \cong N$. It yields in case (i) a subgroup $H \leq N \leq G$ and a set $A \subseteq [d] \setminus \Delta$ with $|A| \geq c \cdot n$ that is an H-orbit and satisfies $\mathbf{Alt}(A) \leq H^{(\Delta \setminus A)}$. Note that in the induction, only the degree d decreases, but $c \cdot n$ remains fixed for each group G_n. Therefore, the size of the set A that we get by induction is indeed $\geq cn$. Since N fixes Ω pointwise, so

does H, and thus, we have $\mathbf{Alt}(A) \leq H^{([d]\setminus A)}$. So we also have case (i) for G.
If case (ii) applies to N, then let $d' := |\Delta|$ be the degree of $N^\Delta \cong N$. The induction hypothesis yields $[\mathbf{Sym}_d^{(\Omega)} : N] = [\mathbf{Sym}(\Delta) : N] \geq \frac{1}{n} \cdot \left(\frac{2d' \cdot \alpha}{e}\right)^{cn/2} \cdot \alpha^\ell$. We have

$$[\mathbf{Sym}_d : N] = [\mathbf{Sym}_d : \mathbf{Sym}_d^{(\Omega)}] \cdot [\mathbf{Sym}_d^{(\Omega)} : N] = (d!/d'!) \cdot [\mathbf{Sym}_d^{(\Omega)} : N].$$

Since $N \leq G \leq \mathbf{Sym}_d$, we also have $[\mathbf{Sym}_d : N] = [\mathbf{Sym}_d : G] \cdot [G : N]$. We know that $[G : N] \leq (d - d')!$ because $[G : N]$ is the order of G/N, whose permutation domain is Ω. In total we get for $[\mathbf{Sym}_d : G]$, using the Stirling approximation $n! \approx \sqrt{2\pi n} \cdot \left(\frac{n}{e}\right)^n$:

$$\begin{aligned}
[\mathbf{Sym}_d : G] = \frac{[\mathbf{Sym}_d : N]}{[G : N]} &\geq \frac{d!}{d'! \cdot (d - d')!} \cdot \frac{1}{n} \cdot \left(\frac{2d' \cdot \alpha}{e}\right)^{cn/2} \cdot \alpha^\ell \\
&\approx \frac{1}{n} \cdot \left(\frac{2d' \cdot \alpha}{e}\right)^{cn/2} \cdot \alpha^\ell \cdot \frac{\sqrt{2\pi d} \cdot (d/e)^d}{d'! \cdot (d - d')!} \\
&= \frac{1}{n} \cdot \left(\frac{2d \cdot \alpha}{e}\right)^{cn/2} \cdot \alpha^\ell \cdot \frac{d^{d-(cn/2)} \cdot d'^{cn/2} \cdot \sqrt{2\pi d}}{d'! \cdot (d - d')! \cdot e^d} \\
&\approx \frac{1}{n} \cdot \left(\frac{2d \cdot \alpha}{e}\right)^{cn/2} \cdot \alpha^\ell \cdot \frac{d^{d-(cn/2)} \cdot d'^{cn/2} \cdot \sqrt{2\pi d}}{d'^{d'} \cdot (d - d')^{d-d'} \cdot 2\pi\sqrt{d'(d - d')}}
\end{aligned}$$

Now since $d' \geq cn$ (otherwise N cannot have $\mathbf{Alt}_m$ as a composition factor), we can use the $d - (cn/2)$ many d-factors in the numerator to dominate all factors $(d - d')$ in the denominator. This yields:

$$\begin{aligned}
&\geq \frac{1}{n} \cdot \left(\frac{2d \cdot \alpha}{e}\right)^{cn/2} \cdot \alpha^\ell \cdot \frac{d^{d'-(cn/2)} \cdot \sqrt{2\pi d}}{d'^{d'-(cn/2)} \cdot 2\pi\sqrt{d'(d - d')}} \\
&\geq \frac{1}{n} \cdot \left(\frac{2d \cdot \alpha}{e}\right)^{cn/2} \cdot \alpha^\ell \cdot \frac{(1 + (c/(1 - c)))^{d'-(cn/2)}}{\sqrt{2\pi d'}}
\end{aligned}$$

In the last step, we used that $d/d' \geq (d' + cn)/d' \geq 1 + \frac{c}{1-c}$. This holds because $d - d' \geq cn$ by the choice of Ω (and therefore, $d' \leq (1 - c)n$). Furthermore, we cancelled $\sqrt{d}$ and $\sqrt{d - d'}$. Now the exponent $d' - (cn/2)$ is at least $d'/2$ (because $d' \geq cn$), and the base is some constant > 1, so it can be checked that the whole fraction in the right factor is ≥ 1, unless d' and hence cn is smaller than some constant depending on c. So, for large enough n, we can remove the factor on the right and are left with:

$$\geq \frac{1}{n} \cdot \left(\frac{2d \cdot \alpha}{e}\right)^{cn/2} \alpha^\ell.$$

So (ii) holds for G. It remains to deal with the case that the large alternating group is a composition factor of $G/N \cong G^\Omega$ or that it is isomorphic to G/N. In the latter case, G acts on Ω as $\mathbf{Alt}_m$, for $m = |\Omega| \geq cn$. If $\mathbf{Alt}(\Omega) \leq G^{(\Delta)}$, then we are done and have case (i) for G. If G does not act as $\mathbf{Alt}(\Omega)$ when it fixes Δ pointwise, then by Lemma 4.3.4, for every $g \in G$, the effect of g on Ω is fully determined by the effect of g on Δ. Thus, $|G| \leq |\Delta|! = d'!$. Then $[\mathbf{Sym}_d : G] \geq \frac{d!}{d'!}$. We have $d' \leq d - cn$ because $|\Omega| \geq cn$. So

$$[\mathbf{Sym}_d : G] \geq \frac{d!}{d'!} \geq \frac{d!}{(d-cn)!} \geq \Big(\frac{d}{e}\Big)^{cn} \cdot \sqrt{d/(d-cn)} \geq \Big(\frac{2d}{e \cdot \lfloor c^{-1} \rfloor !}\Big)^{cn/2} = \Big(\frac{2d \cdot \alpha}{e}\Big)^{cn/2}.$$

The third inequality uses again the Stirling approximation for the factorials, and the last inequality holds because $\frac{d}{e} \geq \frac{2}{c^{-1}}$. Therefore, if this happens, we have case (ii) for G (the additional factors $\frac{1}{n} \cdot \alpha^{\ell}$ in case (ii) only make the expression smaller).

If G/N is not isomorphic to the alternating group, then $\mathbf{Alt}_m$ must be a composition factor of $G/N = G^{\Omega}$. We apply the induction hypothesis. Again, this gives us two cases that can arise for G/N. In case (i), there exists $H \leq G^{\Omega}$ and an orbit $A \subseteq \Omega$ of H with $|A| \geq c \cdot n$ such that $\mathbf{Alt}(A) \leq H^{(\Omega \setminus A)}$. Then let H' be a subgroup of G whose restriction to Ω is H. If $\mathbf{Alt}(A) \leq H'^{([d] \setminus A)}$, then we have case (i) for G, as witnessed by H'. Otherwise, we use again Lemma 4.3.4, which says that the action of H' outside of A determines the action in A. We claim that this also true for G itself, i.e. there are no two $g, g' \in G$ which are different on A and equal on $[d] \setminus A$.
Proof of claim: If the claim is not true, then $G^{([d] \setminus A)}$ is non-trivial. Hence $G^{([d] \setminus A)}$ is not a subgroup of H' because then, the action of H' on A would not be determined by its action outside of A. If $G^{([d] \setminus A)}$ is not a subgroup of H', then $(G^{([d] \setminus A)})^{\Omega}$ is not a subgroup of H, either. But we can always assume that H contains $(G^{([d] \setminus A)})^{\Omega}$: The fact that $\mathbf{Alt}(A) \leq H^{(\Omega \setminus A)}$ will still hold if we add to H all permutations in $G^{([d] \setminus A)}$. This can at most push $H^{(\Omega \setminus A)}$ from $\mathbf{Alt}(A)$ to $\mathbf{Sym}(A)$. Thus, we can assume that H is such that the claim holds and the action of G on A is indeed determined by its action on $[d] \setminus A$.

It follows that $|G| \leq (d - |A|)!$, and since $|A| \geq cn$, we get the same lower bound for $[\mathbf{Sym}_d : G]$ as above in the case where $G/N \cong \mathbf{Alt}_m$. So in this case, case (ii) applies to G.

It remains to check what happens if the induction gives us case (ii) for G/N. Then we have $[\mathbf{Sym}(\Omega) : G/N] \geq \frac{1}{n} \cdot \Big(\frac{2(d-d') \cdot \alpha}{e}\Big)^{cn/2} \cdot \alpha^{\ell}$. It holds

$$|G/N| = |\mathbf{Sym}(\Omega)| / [\mathbf{Sym}(\Omega) : G/N] = \frac{(d-d')!}{[\mathbf{Sym}(\Omega) : G/N]}.$$

We have $|G| = |N| \cdot |G/N|$, and $|N| \leq d'!$. We get a similar chain of inequalities as before:

$$\begin{aligned}
[\mathbf{Sym}_d : G] &= \frac{d!}{|G|} = \frac{d!}{|N| \cdot |G/N|} \\
&\geq \frac{d! \cdot (2(d-d')\alpha)^{cn/2} \cdot \alpha^{\ell}}{d'! \cdot (d-d')! \cdot n \cdot e^{cn/2}} \\
&\approx \frac{1}{n} \cdot \Big(\frac{2d\alpha}{e}\Big)^{cn/2} \cdot \alpha^{\ell} \cdot \frac{d^{d-(cn/2)} \cdot (d-d')^{cn/2} \cdot \sqrt{2\pi d}}{d'^{d'} \cdot (d-d')^{d-d'} \cdot 2\pi \sqrt{d'(d-d')}} \\
&\geq \frac{1}{n} \cdot \Big(\frac{2d\alpha}{e}\Big)^{cn/2} \cdot \alpha^{\ell} \cdot \frac{d^{d'}}{d'^{d'} \cdot \sqrt{2\pi \cdot d'}} \qquad (\star)
\end{aligned}$$

In the last step, we cancelled all factors $(d-d')$ in the denominator with the factors d and $(d-d')$ in the numerator, and also removed $\frac{\sqrt{d}}{\sqrt{d-d'}}$. Now we can almost continue as in the case before, except that we need a case distinction. The important difference is that now, we have no lower bound for d' because this time, the factor $\mathbf{Alt}_m$ is in G^{Ω}, so $[d] \setminus \Omega$ might be arbitrarily small (at least size 1 because it contains at least one orbit). We distinguish the cases whether $d' \geq cn$ or $d' < cn$. Let us start with the latter case. As above, we have $d/d' \geq (d'+cn)/d'$, because this only depends on the fact that $|\Omega| \geq cn$, which still holds by choice of Ω. If $d' < cn$, then this becomes: $d/d' \geq (d'+cn)/d' \geq 2$. So then, the right factor in $(\star)$ is at least $\frac{2^{d'}}{\sqrt{2\pi d'}}$. This is greater than one for all values of $d' \geq 2$. If $d' = 1$, then we can argue differently. Then, the right fraction in $(\star)$ is equal to $d/\sqrt{2\pi}$, which is also greater than one for all large enough n (because $d \geq cn$). In case that $d' \geq cn$, we make the same argument as before, and use the bound $d/d' \geq (d'+cn)/d' \geq 1 + \frac{c}{1-c}$. Then the right factor in $(\star)$ is at least $\frac{(1+\frac{c}{1-c})^{d'}}{\sqrt{2\pi d'}}$. Again, for large enough d', this is ≥ 1. Since $d' \geq cn$, this happens for large enough n. So in all cases, we can remove the right factor and the product only gets smaller. So all in all, we have

$$[\mathbf{Sym}_d : G] \geq \frac{1}{n} \cdot \left(\frac{2d \cdot \alpha}{e}\right)^{cn/2} \alpha^{\ell},$$

as desired. This finishes the case where G is intransitive.

Case 2: G is transitive, but not primitive.
In this case, G has a non-trivial block system. Let $N \lhd G$ be the normal subgroup that stabilises each block setwise. Then G/N is the action of G on the set of blocks. Again, according to Lemma 4.3.3, the large alternating group is either a composition factor of N, of G/N, or it is isomorphic to G/N. In the latter case, (ii) applies to G: If $G/N \cong \mathbf{Alt}_m$ for some $m \geq c \cdot n$, then the block system has $m \leq d/2$ blocks. Each block has size t, with $2 \leq t \leq \lfloor c^{-1} \rfloor$ (note that this case can only happen if $c \leq \frac{1}{2}$). Then we have $|G| \leq (t!)^m \cdot m!$. Thus,

$$\begin{aligned}
[\mathbf{Sym}_d : G] &\geq \frac{d!}{(t!)^m \cdot m!} \geq \frac{d!}{(t!)^m \cdot (d/2)!} \\
&\geq \frac{\sqrt{2} \cdot (2d/e)^{(d/2)}}{(t!)^{(d/2)}} \geq \left(\frac{2d \cdot \alpha}{e}\right)^{d/2} \\
&\geq \left(\frac{2d \cdot \alpha}{e}\right)^{c \cdot n/2} \geq \frac{1}{n} \cdot \left(\frac{2d \cdot \alpha}{e}\right)^{c \cdot n/2} \cdot \alpha^1.
\end{aligned}$$

The last step holds because $\alpha \leq 1$. The next case is that $\mathbf{Alt}_m$ is a composition factor of G/N. Then we do not need the inductive step either and can immediately conclude with the same lower bound as above for $[\mathbf{Sym}_d : G]$. This is because if $\mathbf{Alt}_m$ is a composition factor of G/N, the degree of G/N must be at least $m \geq cn$, and so, the number of blocks must also be greater than cn, and the block size can be at most $\lfloor c^{-1} \rfloor$.

It remains the case that N has $\mathbf{Alt}_m$ as a composition factor. We apply the inductive hypothesis to N. Should case (i) hold for N, then we immediately know that case (i) also applies to G, because then we have some $H \leq N \leq G$ (so in particular, $H \leq G$) and an H-orbit $A \subseteq [d]$ such that $\mathbf{Alt}(A) \leq H^{([d]\setminus A)}$ (this is easier than before because N and G have the same permutation domain now).
If we instead have case (ii) for N, then $[\mathbf{Sym}_d : N] \geq \frac{1}{n} \cdot \left(\frac{2d\cdot\alpha}{e}\right)^{c\cdot n/2} \cdot \alpha^{\ell-1}$.
Each block must be of size at least $c \cdot n$ because N has an alternating group of that degree as a composition factor, and the blocks are the orbits of N. If these orbits were smaller than $c \cdot n$, then N could not have a composition factor of that degree, as we already argued in the intransitive case. Therefore, the number of blocks is at most $\lfloor c^{-1} \rfloor$ and so, $|G/N| = [G : N] \leq \lfloor c^{-1} \rfloor!$. In total, we have

$$\begin{aligned}[\mathbf{Sym}_d : G] = [\mathbf{Sym}_d : N]/[G : N] &\geq \frac{1}{n} \cdot \left(\frac{2d\cdot\alpha}{e}\right)^{c\cdot n/2} \cdot \frac{\alpha^{\ell-1}}{\lfloor c^{-1} \rfloor!} \\ &= \frac{1}{n} \cdot \left(\frac{2d\cdot\alpha}{e}\right)^{c\cdot n/2} \cdot \alpha^{\ell}.\end{aligned}$$

This finishes the transitive and imprimitive case.

Case 3: G is primitive.
It remains the case that G is primitive. Since G has $\mathbf{Alt}_m$ as a composition factor, with $m \geq cn$, we have for the order of G: $|G| \geq (cn)!/2 \approx \frac{1}{2}\left(\frac{cn}{e}\right)^{cn} \cdot \sqrt{2\pi cn} \geq n^{1+\log n} \geq d^{1+\log d}$. Thus, by Theorem 4.3.2, G has a normal subgroup N of index $\leq d$ such that N has a block system on which it acts as a Johnson group $\mathbf{Alt}_k^{(t)}$, where $k \geq \log d$. Here, k is the size of the permutation domain, and t is the length of the tuples over $[k]$ of the Johnson action. Since the index of N in G is $\leq d$, G/N has order at most d, so it is too small to have the composition factor $\mathbf{Alt}_m$. Therefore, N must have $\mathbf{Alt}_m$ as a composition factor. Now N acts as $\mathbf{Alt}_k^{(t)}$ on the blocks of its block system. We have $k \geq \log d$, and every block is identified with a t-tuple over a k-element domain. So the number of blocks is $k^t \geq (\log d)^t$. Thus, the block-size is at most $d/(\log d)^t$. Even for $t = 1$, this is asymptotically less than cn (recall that $d \leq n$). So if n is large enough, then our sought alternating group cannot be a factor of the blockwise stabiliser $N' \lhd N$ (whose orbits are the blocks), but it must be a factor of N/N', which is the action of N on the blocks. This requires the number of blocks to be $\geq cn$. Then the block-size can be at most $\lfloor c^{-1} \rfloor$, as in Case 2 above. If the block-size is ≥ 2, then we obtain $[\mathbf{Sym}_d : N] \geq \left(\frac{2d\alpha}{e}\right)^{cn/2}$ with exactly the same calculation as in Case 2. Then

$$[\mathbf{Sym}_d : G] = [\mathbf{Sym}_d : N]/[G : N] \geq \frac{1}{n} \cdot \left(\frac{2d\cdot\alpha}{e}\right)^{cn/2}.$$

Here, we used that $[G : N] \leq d \leq n$. Thus, we get the desired index-bound for G without having to recurse any further. In the case that the block system of N is trivial and the blocks are singletons, then we must have $t = 1$: If $t \geq 2$, then N is isomorphic to an alternating group on $\leq \sqrt{d} < cn$ points. Alternating groups have no normal subgroups other than the trivial group and itself, so in this case, N cannot have $\mathbf{Alt}_m$ with $m \geq cn$ as a composition factor. So it remains the case that $t = 1$, and $N = \mathbf{Alt}_d$. Since $N \leq G$, we have case (i) for G then. □

We now combine this lemma with the following result from the literature, which guarantees the existence of a large alternating group as a composition factor in any sufficiently large group.

Lemma 4.3.6 (Lemma 2.2 in [14])**.** *Let G be a permutation group of degree n. If G has no composition factor isomorphic to an alternating group of degree $> C$, then $|G| \leq C^{n-1}(C \geq 6)$.*

We get the following consequence for groups of bounded index:

Corollary 4.3.7. *Let $(G_n)_{n \in \mathbb{N}}$ be a family of groups such that for all n, $G_n \leq \mathbf{Sym}_n$. Assume that there exists some constant $k \in \mathbb{N}$ such that asymptotically, $[\mathbf{Sym}_n : G_n] \leq 2^{nk}$.*
Then there is a function $f(n) \in \Theta(n)$ such that for all large enough n, G_n has a composition factor isomorphic to $\mathbf{Alt}_m$, for some $m \geq f(n)$.

Proof. Let $f(n) : \mathbb{N} \longrightarrow \mathbb{N}$ be the smallest upper bound for the maximum degree of an alternating group that appears as a composition factor of G_n. We have to show that $f(n)$ is linear in n. Suppose for a contradiction that $f(n) \in o(n)$. Let $g(n) := f(n) + 1$, for example. Then $g(n) \in o(n)$, and it holds for all large enough n that G_n has no composition factor isomorphic to $\mathbf{Alt}_{g(n)}$. By Lemma 4.3.6, we have $|G_n| \leq g(n)^{n-1}$. Then we get for the index:

$$[\mathbf{Sym}_n : G_n] \geq \frac{n!}{g(n)^{n-1}}.$$

Using the Stirling approximation, we obtain:

$$\frac{n!}{g(n)^{n-1}} \approx \Big(\frac{n}{e \cdot g(n)}\Big)^{n-1} \cdot \Big(\frac{n \cdot \sqrt{2\pi n}}{e}\Big).$$

Since $g(n)$ is sublinear, the fraction $\frac{n}{e \cdot g(n)}$ is not bounded from above by any constant, so we have $[\mathbf{Sym}_n : G_n] > 2^{nk}$, for every constant $k \in \mathbb{N}$. This is a contradiction, so $f(n)$ must be linear. □

Finally, we can put everything together to show that every group of index $\leq 2^{nk}$ has a large alternating subgroup. This lemma is the version of Theorem 5.2A in [45] for groups of index $\leq 2^{nk}$.

Lemma 4.3.8. *Let $(G_n)_{n \in \mathbb{N}}$ be a family of groups such that for all n, $G_n \leq \mathbf{Sym}_n$. Assume that there exists some constant $k \in \mathbb{N}$ such that asymptotically, $[\mathbf{Sym}_n : G_n] \leq 2^{nk}$.*

Then there exists a constant $0 < c \leq 1$ such that for all large enough n, G_n has a subgroup H_n such that $\mathbf{Alt}(A) \leq H_n^{([n] \setminus A)}$ for some H_n-orbit A of size $|A| \geq cn$.

Proof. Corollary 4.3.7 states that all large enough G_n have a composition factor isomorphic to $\mathbf{Alt}_m$, where $m \geq f(n)$, for some function $f(n) \in \Theta(n)$. There is a constant $0 < c \leq 1$ such that $f(n) = c \cdot n$. Hence, Lemma 4.3.5 applies (where $d := n$).
We now want to show that case (ii) from Lemma 4.3.5 cannot occur for any of the G_n. Suppose for a contradiction that $[\mathbf{Sym}_n : G_n] \geq \frac{1}{n} \cdot \left(\frac{2n \cdot \alpha}{e}\right)^{cn/2} \cdot \alpha^\ell$ for some large enough n. First, we bound ℓ. In Lemma 4.3.5, this was defined as the number of times the transitive and imprimitive case occurs in the recursion. By inspection of the proof, we see that this case occurs only if G is transitive and imprimitive, *and* the sought composition factor $\mathbf{Alt}_m$ is in the blockwise stabiliser $N \lhd G$. In this case, the block-size must be $\geq cn$. Whenever this case occurs, we continue with the intransitive case where the orbits are the blocks, which means that we eventually end up with a group whose degree is at most the block size of the block system of G. So whenever we have the transitive imprimitive case and continue the recursion, we at least halve the degree. We may only do this until the degree drops below cn, so we have $\ell \leq \log n$ (actually, the bound could be made even smaller, but $\log n$ is already much tighter than we really need). Thus,

$$\begin{aligned}[\mathbf{Sym}_n : G_n] &\geq \frac{1}{n} \cdot \left(\frac{2n}{e \cdot \lfloor c^{-1} \rfloor !}\right)^{cn/2} \cdot \left(\frac{1}{\lfloor c^{-1} \rfloor !}\right)^{\log n} \\ &\geq \left(\frac{2n}{e \cdot \lfloor c^{-1} \rfloor !^2}\right)^{cn/2}\end{aligned}$$

Now for every constant $k \in \mathbb{N}$ that we could choose, 2^{kn} can be written as a^n for some constant a (that can be arbitrarily large but still constant). Comparing the above expression with a^n, we find:

$$\begin{aligned}\frac{[\mathbf{Sym}_n : G_n]}{a^n} &\geq \left(\frac{2n}{e \cdot (c^{-1})!^2}\right)^{cn/2} \cdot \frac{1}{a^n} \\ &= \left(\frac{2n}{a^{2/c} \cdot e \cdot (c^{-1})!^2}\right)^{cn/2}.\end{aligned}$$

Clearly, for any constant a, this fraction tends to infinity in the limit. So, if $[\mathbf{Sym}_n : G_n]$ is bounded from above by 2^{kn}, then for all large enough n, case (ii) from Lemma 4.3.5 cannot possibly apply to G_n. □

Theorem 4.3.1 now follows from this lemma.

Proof of Theorem 4.3.1. Assume for a contradiction that there was a constant $0 < c \leq 1$ such that for every large enough n, there exists a subset $\Delta_n \subseteq [n]$ of size $|\Delta_n| \geq cn$, which is precisely the set of elements in singleton parts in $\mathbf{SP}_A(G_n)$. In the following, we assume that the size of Δ_n is not only lower-bounded linearly, but that there also exists some other constant $1 > d > c$ such that $|\Delta_n| \leq dn$, for all large enough n. We deal with the other case in the end.
Let $H_n := (G_n^{([n] \setminus \Delta)})^\Delta$ be the subgroup of G_n that fixes every point outside of Δ, restricted to its action on Δ. We now show that $[\mathbf{Sym}(\Delta) : H_n] > 2^{nk}$, for every $k \in \mathbb{N}$.

Indeed, if there were a $k \in \mathbb{N}$ such that $[\mathbf{Sym}(\Delta) : H_n] \leq 2^{nk}$ for all large enough n, then by Lemma 4.3.8, all the H_n would have a subgroup H'_n containing the alternating group on an orbit A of size $|A| \geq c' \cdot |\Delta|$, where c' is the constant from the lemma. More precisely, $\mathbf{Alt}(A) \leq H_n'^{(\Delta \setminus A)}$. Thus, every even permutation on A (that fixes everything in Δ outside of A) would be contained in H_n, and since H_n is a subgroup of G_n that fixes everything outside of Δ, this means that $\mathbf{Alt}(A) \leq G^{([n] \setminus A)}$. But then, $A \subseteq \Delta$ would be a part of the coarsest alternating supporting partition $\mathbf{SP}_A(G_n)$. This contradicts the fact that $\mathbf{SP}_A(G_n)$ only has singleton parts on Δ. Therefore, we must have that $[\mathbf{Sym}(\Delta_n) : H_n] > 2^{nk}$, for every $k \in \mathbb{N}$.
As a next step, we calculate that this entails a violation of the assumption that $[\mathbf{Sym}_n : G_n]$ can be upper-bounded by 2^{nk}, for some k. Namely, if $[\mathbf{Sym}(\Delta_n) : H_n] > 2^{nk}$ for all $k \in \mathbb{N}$, then $|H_n| < \frac{|\Delta_n|!}{2^{nk}}$, for every choice of k. Then we also have $|G_n| < \frac{|\Delta_n|!}{2^{nk}} \cdot (n - |\Delta_n|)!$. This is because by Lemma 4.2.10, every element of G_n stabilises $\mathbf{SP}_A(G_n)$ setwise and so in particular, maps singleton parts only to singleton parts; therefore, $[n] \setminus \Delta$ is a union of G-orbits and so H_n is normal in G_n. It follows for the index, for every $k \in \mathbb{N}$:

$$[\mathbf{Sym}_n : G_n] = \frac{n!}{|G_n|} > \frac{n! \cdot 2^{nk}}{|\Delta_n|! \cdot (n - |\Delta_n|)!}.$$

Since $cn \leq |\Delta_n| \leq dn$, we have $|\Delta_n| \leq dn$ and $n - |\Delta_n| \leq (1-c)n$. So we can continue:

$$\begin{aligned} \frac{n! \cdot 2^{nk}}{|\Delta_n|! \cdot (n - |\Delta_n|)!} &\geq \frac{n! \cdot 2^{nk}}{(dn)! \cdot ((1-c)n)!} \\ &\approx 2^{nk} \cdot \left(\frac{1}{d}\right)^{dn} \cdot \left(\frac{1}{1-c}\right)^{(1-c)n} \cdot \frac{1}{\sqrt{2\pi \cdot d(1-c) \cdot n}} \end{aligned}$$

Since $d < 1$ and $(1-c) < 1$, the above product is greater than 2^{nk} (for large enough n). Because this lower bound for $[\mathbf{Sym}_n : G_n]$ holds for every $k \in \mathbb{N}$, we can conclude that $[\mathbf{Sym}_n : G_n] > 2^{nk}$, for every k. But the assumption of Theorem 4.3.1 says that there exists a $k \in \mathbb{N}$ such that for all large enough n, $[\mathbf{Sym}_n : G_n] \leq 2^{nk}$. This is a contradiction. This proves the theorem in case that $|\Delta_n|$ can be upper-bounded by some linear function $d \cdot n$, for $d < 1$. The case that $|\Delta_n| > dn$, for all $d < 1$, cannot occur: Lemma 4.3.8 applied to G_n states that G_n contains an alternating group of linear degree (which fixes the rest pointwise). Therefore, for all large enough n, $\mathbf{SP}_A(G_n)$ must have at least one part of linear size. So it is impossible that all but sublinearly many elements of $[n]$ are in singleton parts in $\mathbf{SP}_A(G_n)$. $\square$

4.4 Orbits, supports and Choiceless Polynomial Time

As we already discussed in Chapter 3, no Choiceless Polynomial Time sentence can activate a h.f. set whose orbit with respect to the automorphism group of the input structure has super-polynomial size. Therefore, a way of proving that certain objects are not CPT-definable in a given family of input structures is to show that the size of their orbit grows super-polynomially in the structure size. This is also what we do in Chapter

7.
The ultimate goal is to show that some polynomial time *decision problem* is not CPT-definable. In order to accomplish this with the symmetry approach, one has to prove first that solving the decision problem in CPT requires the construction of hereditarily finite sets with certain properties; as a next step, one shows that these necessary properties always come at the cost of a super-polynomial orbit.

Usually, this necessary property of h.f. objects is the *minimum size of a support*: It turns out that for model-theoretic reasons (a connection between the support size and the number of pebbles needed to distinguish structures in a pebble game), solving certain decision problems in CPT requires the construction of objects whose smallest support is large. Intuitively speaking, the larger the support of an object is, the less symmetric it is: If a h.f. set x over some structure $\mathfrak{A}$ is only fixed by those permutations in $\mathbf{Aut}(\mathfrak{A})$ which also fix, say, half of the elements in A, then one would expect that x is moved by most automorphisms of $\mathfrak{A}$. Indeed, for suitably symmetric structures $\mathfrak{A}$, it can be shown that a large support size also implies super-polynomial orbit size for any h.f. set x over $\mathfrak{A}$.

The first time that this line of argument was employed was already in [21] in 1999, where CPT was introduced. Blass, Gurevich and Shelah showed that CPT *without counting* cannot define the EVEN-query (asking whether the universe of the input structure has even cardinality) on sets without any relations. Their argument is this: Let $\mathfrak{A}$ and $\mathfrak{B}$ be sufficiently large structures with empty vocabulary, and $\mathfrak{A}_k, \mathfrak{B}_k$ be the structures enriched with all h.f. sets over the respective universe that are *transitively k-supported*, i.e. all sets x such that every $y \in \text{tc}(k)$ has a support of size $\leq k$.
It turns out that for any constant m, it holds $\mathfrak{A}_k \equiv_{\mathcal{L}^m_{\infty,\omega}} \mathfrak{B}_k$, provided that $\mathfrak{A}$ and $\mathfrak{B}$ are large enough. In other words, even if a CPT-program (without counting) activated all sets of support $\leq k$ on input $\mathfrak{A}$ or $\mathfrak{B}$, it would not be able to tell any difference between the structures, in particular between their cardinalities. Consequently, solving EVEN in CPT without counting requires the construction of objects with larger support than any constant $k \in \mathbb{N}$.

In a second step, the authors show that any object whose smallest support has super-constant size has a super-polynomial orbit. They conclude that CPT without counting cannot distinguish the cardinalities of arbitrarily large structures. The proof of this second step explicitly constructs exponentially many distinct automorphic images of any given object x with super-constant support. It is quite lengthy and technically involved, but for the sake of illustration, we consider the easier case where $x \subseteq [n]$ is simply a subset of the universe $[n]$:

Example 4.4.1. *Let $x \subseteq [n]$ be a subset of the structure $[n]$. Suppose $f(n) \leq |x| \leq n/2$, for some super-constant function f. Since $\mathbf{Aut}([n]) = \mathbf{Sym}_n$, the smallest support of x is x itself (moving any element from x outside of x also moves x). Thus, the* minimum support size *of x is* super-constant *in n. Its* orbit *is* super-polynomial*: We have $|\mathbf{Orb}(x)| \geq \binom{n}{f(n)} \approx n^{f(n)}$, which is greater than n^k, for any constant k.*

This is a very simple example for the kind of support-orbit connection that is crucial for proving CPT lower bounds. The proof in [21] generalises the argument from the example to all h.f. sets over $[n]$ with super-constant support. Whenever the automorphism group of the structure in question is the full symmetric group on its universe, then the techniques from [21] are sufficient to show that super-constant support implies a super-polynomial orbit. For structures with smaller automorphism groups, though, it is not clear how to prove super-polynomial orbit results in this style. It seems like this is one of the main challenges on the way to a separation of CPT and PTIME because we expect that difficult and interesting problems require inputs with more structure than just naked sets or complete graphs (which are the typical structures with "full" automorphism groups). Intuitively, the more structure and "information" there is in the instances, the less symmetric they are and the more difficult it is to prove super-polynomial orbits theorems for objects with large supports.

Nevertheless, it makes sense to systematically explore to what extent the techniques from [21] are applicable. This was done by Benjamin Rossman in 2010 [95]. He extracted the key properties that the automorphism group of a structure must have such that the statement "super-constant support leads to super-polynomial orbit" can be shown with essentially the same method as in [21].

4.4.1 Groups with the (k,r)-support property

A group has the (k, r)-*support property* if every (k, r)-*constructible* subgroup is k-*supported.* Roughly speaking, this means that every subgroup that can occur as the stabiliser of a CPT-definable hereditarily finite set has a support of size at most k. Here is the formal definition:

Definition 4.4.2 (Definition 4.1 in [95]). *Let A be a set of size n and G be a group acting faithfully on A. A subgroup of G is k-supported if it has a support $S \subseteq A$ with $|S| \leq k$.*
*For all k and r, the set of (k, r)-*constructible subgroups *of G is the minimal family of subgroups such that*

- *every k-supported subgroup is (k, r)-constructible, and*
- *if $H_1, ..., H_r$ are (k, r)-constructible, $H_1 \cap ... \cap H_r \subseteq H$ and $[G : H] \leq n^k$, then H is (k, r)-constructible.*

*The group G has the (k, r)-*support property *if every (k, r)-constructible subgroup of G is k-supported.*

The definition of (k, r)-constructibility reflects the fact that the (setwise) stabiliser of any set $x = \{y_1, ..., y_r\}$ is at least the intersection of the stabilisers $\mathbf{Stab}(y_1) \cap ... \cap \mathbf{Stab}(y_r)$. Potentially, $\mathbf{Stab}(x)$ may be larger than that because permutations of the r elements of x are also in its stabiliser. However, if G is the automorphism group of some input structure $\mathfrak{A}$, and we are considering a CPT-program with resource bound n^k that activates x, then we know that $[\mathbf{Aut}(\mathfrak{A}) : \mathbf{Stab}(x)] \leq n^k$ (because of the Orbit-Stabiliser Theorem and the closure of definable sets under automorphisms). Thus, in structures whose automorphism group has the (k, r)-support property, CPT can only define sets of bounded support. Formally, Rossman shows the following fact. Recall that the *variable rank* of a CPT program is the maximum number of free variables in any of its terms.

Lemma 4.4.3 (Proposition 4.3 / Lemma 4.5 in [95]). *Let* Π *be a program in* $\mathrm{CPT}(n^k)$ *with variable rank* $\leq r$. *Then for every finite structure* $\mathfrak{A}$ *and every* $x \in \mathrm{HF}(A)$ *that is activated in the run of* Π *on* $\mathfrak{A}$, $\mathbf{Stab}_{\mathbf{Aut}(\mathfrak{A})}(x)$ *is a* (k, r)*-constructible subgroup of* $\mathbf{Aut}(\mathfrak{A})$.

Consequently, the power of CPT to define objects with large support is limited on structures whose automorphism groups have the (k, r)-support property. We summarise this as follows.

Theorem 4.4.4. *Let* $(\mathfrak{A}_n)_{n\in\mathbb{N}}$ *be a family of* τ*-structures and let* $k \in \mathbb{N}$ *be fixed. If for every* $r \in \mathbb{N}$, *there exists* $n_0 \in \mathbb{N}$ *such that for all* $n \geq n_0$, $\mathbf{Aut}(\mathfrak{A}_n)$ *has the* (k, r)*-support property, then there is no* CPT*-program that can activate sets whose smallest support has size* $> k$ *on the structures* $\mathfrak{A}_n$, *for arbitrarily large* n.

Proof. Let Π be any CPT-program with vocabulary τ. Let $r \in \mathbb{N}$ be the variable rank of Π. Then for all $\mathfrak{A}_n$ with $n \geq n_0$, by Lemma 4.4.3 it holds that every object that Π activates on input $\mathfrak{A}_n$ has a support of size at most k. Thus, there exists no CPT-program that can activate objects whose smallest support has size $> k$ on structures $\mathfrak{A}_n$, for n arbitrarily large. □

Groups for which the (k, r)-support property is known are the symmetric group and the group of linear automorphisms of finite vector spaces:

Theorem 4.4.5 (Proposition 5.2 in [95]). *For* $n > 2kr$, $\mathbf{Sym}_n$ *has the* (k, r)*-support property.*

Theorem 4.4.6 (Proposition 6.2 in [95]). *If* V *is a finite vector space of dimension* $> r^2k^2$, *then the group* $\mathbf{GL}(V)$ *of linear automorphisms has the* (k, r)*-support property.*

An immediate consequence of this is Theorem 3.2.3, stating that the set of hyperplanes of a given finite vector space is not CPT-definable; the smallest support of such a hyperplane in an n-dimensional space has size $n - 1$, which is greater than k in the theorem above. Therefore, by Theorem 4.4.4, no CPT-program can define such hyperplanes in a finite vector space.

The proofs of Theorems 4.4.5 and 4.4.6 are similar. They roughly work the same way as the original proof by Blass, Gurevich and Shelah [21] for the non-definability of the EVEN problem in CPT^- but are much shorter and simpler. In short, the main properties of $\mathbf{Sym}_n$ (and $\mathbf{GL}(V)$) that Rossman's proof of the (k, r)-support property is based on are:

- Supports (of size $< n/2$) of subgroups of $\mathbf{Sym}_n$ are closed under intersection. Therefore, any $(n/2)$-supported subgroup has a unique minimal support.
- For any set $V \subseteq [n]$ with $|V| < (n/2)$, it holds for the index of the setwise stabiliser: $[\mathbf{Sym}_n : \mathbf{Stab}(V)] = \binom{n}{|V|}$, which is super-polynomial in n if $|V|$ is super-constant.

The second property essentially says that subsets of $[n]$ of super-constant size have a super-polynomial orbit, as we mentioned in Example 4.4.1. Thus, Rossman's analysis shows that for groups where the supports are closed under intersection, it suffices to consider the orbit size of subsets of the universe in order to prove that also any higher-order object with large support has a large orbit. For completeness, we also state an important characterisation of groups with the (k, r)-support property that is used in the proofs of Theorems 4.4.5 and 4.4.6.

Lemma 4.4.7 (Lemma 4.2 in [95])**.** *G has the (k, r)-support property if and only if every kr-supported subgroup with index $\leq n^k$ is k-supported.*

Using this, it is not difficult to prove the (k, r)-support property also for other groups that are similar to $\mathbf{Sym}_n$. For example, the proof of Proposition 5.2 in [95] works also for $\mathbf{Alt}_n$. Essentially, this is because the alternating group also satisfies the two properties mentioned above: The first property can be shown like in the proof of Lemma 5.3 in [95], with the only difference that the supports whose intersection is taken must be of size $< n/2 - 1$ instead of $< n/2$. The second property is satisfied because $\mathbf{Alt}_n$ acts transitively on the subsets of $[n]$ of fixed size. It is also straightforward to generalise the proof to direct products of symmetric or alternating groups. In the next chapter, we introduce the automorphism group of the n-dimensional hypercube, and it also contains $\mathbf{Sym}_n$. However, in that case, $\mathbf{Sym}_n$ is acting on a set of size 2^n instead of n, so the proof of the (k, r)-support property for $\mathbf{Sym}_n$ does not translate to that setting. This is important because the hard CFI-instances we study in this thesis are based on n-dimensional hypercubes. If it were clear that their automorphism groups satisfied the (k, r)-support property, then CPT lower bounds on such structures would follow directly from Rossman's results and we would not have to go to such lengths in the study of their symmetries.

In summary, structures whose automorphism groups have the (k, r)-support property for constant or at least slowly growing values of k are good candidate inputs on which limitations of the power of CPT can be shown – provided that a connection between support size and expressive power can also be established. The disadvantage of this approach is that – as it seems – only extremely symmetric structures do have this property: For example, (disjoint unions of) complete graphs or the like. It is not obvious that there exists a decision problem whose instances are limited to such highly symmetric structures that could potentially separate CPT from Ptime. In fact, as we discuss in Section 5.2, the automorphism groups of Cai-Fürer-Immerman structures do not have the (k, r)-support property. The isomorphism problem of these structures is, however, a promising candidate for the separation of CPT from Ptime.

4.5 Outlook: Symmetries beyond automorphisms

The approach towards CPT lower bounds via the estimation of orbit sizes has its limitations: It is useless on, for example, *rigid* structures, i.e. structures with no non-trivial automorphisms. An important class of such structures are *multipedes* [67]. These are similar to Cai-Fürer-Immerman graphs and have a polynomial time isomorphism problem, but are indistinguishable in bounded-variable counting logic. The difference to CFI-structures is that multipedes are rigid. So, in case that CPT can define isomorphism of unordered CFI-graphs but not of multipedes, then the methods we employ in this thesis are too weak to detect this: We always argue via super-polynomial orbit sizes of h.f. sets, but on rigid structures, all CPT-definable objects trivially have orbit size 1, which is of course not super-polynomial. Nonetheless, it is quite clear that CPT cannot make arbitrary choices in multipedes because in any pair of "feet" of a multipede, the feet are indistinguishable in $\mathcal{C}^k$. The individual computation steps of CPT are not only orbit-invariant but actually even invariant under $\mathcal{C}^k$-types, for a constant k that depends on the CPT program. Therefore, CPT cannot distinguish the feet of a multipede (at least not in a single step), even though they are in singleton orbits. This type-invariance of CPT will become clearer in Chapter 10, when we consider the computation model *Deep Weisfeiler Leman* (DWL). DWL has the same expressive power as CPT, and the steps of a DWL-computation are $\mathcal{C}^3$-type invariant by definition (e.g. the Pair-operation is always executed for all pairs of objects with the same $\mathcal{C}^3$-type). The $\mathcal{C}^3$-type partition of a structure can generally be coarser than its orbit-partition, and so, type-invariance is a stronger limitation than orbit-invariance. The reason why we focus so much on orbit-invariance is that automorphisms of structures are relatively easy to understand. They form groups, and so we have all the group-theoretic techniques available, that we have presented in this chapter. It is less clear how the type-invariance of CPT could be exploited and turned into an argument against the power of CPT. In personal communication, it has been suggested to us by Anuj Dawar that there may be a connection between the type partition of a structure and certain *groupoids* (rather than automorphism groups). A groupoid is essentially a "group" G whose binary operation $\circ : G \times G \longrightarrow G$ is only a partial function. A potential way how groupoids might come into the picture is when we consider *partial* automorphisms ("motions"), as in the proof of the zero-one law for CPT$^-$ [18]. A partial automorphism is a bijection between subsets of the universe with small size. The composition of such partial automorphisms is only defined when co-domain and domain of the two match, so they indeed form a groupoid. As shown in [18], one can even define a suitable notion of supports with respect to partial automorphisms. However, we do not know in how far the Orbit-Stabiliser Theorem generalises to such groupoids, or if the concept of an orbit even makes any sense at all for groupoids. Thus, at least the super-polynomial orbit argument will be difficult to make using only partial automorphisms. Anyway, we feel like the group-theoretic techniques that we mostly employ in this thesis have not yet reached their limitations. Therefore, we decided to explore the group-theoretic route towards CPT lower bounds in depth and to leave the type-based "groupoid method" for the future, in case that the automorphism-based method does not suffice to separate CPT from Ptime.

5 The Cai-Fürer-Immerman graphs and their symmetries

The Cai-Fürer-Immerman construction [29] is a central tool for proving lower bounds in finite model theory. Basically, the construction takes a family of connected base graphs and substitutes the vertices and edges of each graph by certain "CFI-gadgets". These gadgets exist in *twisted* and *non-twisted* versions and depending on how many vertex-gadegts are twisted, two CFI-graphs on the same base graph are either isomorpic or not. The point of the construction is that the "twists" are hard to detect for Spoiler in the bijective k-pebble game and thus, the difference between two non-isomorphic CFI-graphs is hard to express in counting logic/FPC. In Chapter 6 we give a detailed account on what is known about the connection between pebble games and CPT. Indeed, there is strong evidence that distinguishing general CFI-graphs might be hard not only for FPC, but also for CPT (see [40]). A proof that CPT cannot distinguish certain CFI-graphs would separate CPT from PTIME since the isomorphism problem of CFI-graphs reduces easily to a linear equation system over the finite field $\mathbb{F}_2$ [10], which can be solved efficiently by Gaussian elimination. There exist also generalised CFI-constructions which correspond to equation systems over other finite fields or rings (see [55], [82]), but these are more relevant for the study of rank logic, so we only focus on the classical construction over $\mathbb{F}_2$ here.

It is important to note that in the case of CPT, the presence or absence of any kind of ordering on the CFI-graphs makes an enormous difference – this is not the case with weaker logics like FPC. Therefore, the CFI-graphs that we consider in this thesis differ from the usual presentation in the literature in that regard. In other contexts, the base graphs for the CFI-construction are often taken to be linearly ordered or at least preordered with small colour classes. This induces an order on the vertex-gadgets of the corrresponding CFI-structures. Whether or not FPC can distinguish non-isomorphic CFI-graphs does not depend on the presence or absence of the order on the gadgets. However, CPT can distinguish CFI-graphs if there is an order on the gadgets [40], but as the evidence that we collect in this thesis suggests, this may not be possible if the structures are completely unordered.

In this chapter, we present the definition of unordered CFI-graphs and examine their automorphism groups. These are more complex than the automorphisms of ordered CFI-graphs because we not only have the "edge-flip automorphisms" that have already been mentioned in the literature, but also automorphisms of the base graph itself. Furthermore, we present CFI-graphs on unordered hypercubes – as the results in Chapters 7 and 9 suggest, this is a family of CFI-structures which may be particularly hard to distinguish in CPT, due to their symmetries. An even more symmetric family of base graphs are complete graphs but they are not a suitable candidate because CPT can define isomorphism of CFI-structures over complete graphs [91].

5.1 The CFI-construction

Fix an undirected (and unordered) connected graph $G = (V, E)$ as the *base graph* for the CFI-construction. We turn G into a CFI-graph by replacing the edges with certain edge-gadgets and the vertices with vertex-gadgets. There are two types of vertex-gadgets, called odd and even. To construct a concrete CFI-graph over G, we have to fix a set $S \subseteq V$ of vertices which are replaced by the *odd* gadget. The vertices in $V \setminus S$ will be turned into the *even* gadget. Following the notation in [40], we denote the resulting CFI-graph by $\mathfrak{G}^S$. The precise definition is as follows:
Let

$$\widehat{E} := \{e_0, e_1 \mid e \in E\}.$$

These are the vertices that will form the edge-gadgets of $\mathfrak{G}^S$, so there are two vertices per edge-gadget. To define the vertices in vertex-gadgets, we let, for each $v \in V$,

$$v^*_S := \begin{cases} \{v^X \mid X \subseteq E(v), |X| \text{ even } \} & \text{if } v \notin S \\ \{v^X \mid X \subseteq E(v), |X| \text{ odd } \} & \text{if } v \in S \end{cases}$$

Here, $E(v) \subseteq E$ are the edges incident to v in G. The vertices in v^*_S form the vertex-gadget of v. In total, we let

$$\widehat{V}_S := \bigcup_{v \in V} v^*_S.$$

Then the vertex-set of $\mathfrak{G}^S$ is $V(\mathfrak{G}^S) := \widehat{V}_S \cup \widehat{E}$. The edges of the CFI-graph are given by

$$E(\mathfrak{G}^S) := \{\{v^X, e_i\} \mid v^X \in \widehat{V}_S, e_i \in \widehat{E}, |X \cap \{e\}| = i\} \cup \{\{e_0, e_1\} \mid e \in E\}.$$

In other words, for every $v \in V$, we connect each $v^X \in v^*_S$ with the edge-gadgets of all edges $e \in E(v)$ in such a way that v^X is connected with e_0 if $e \notin X$, and otherwise with e_1. Also, we connect e_0 and e_1 to ensure that no automorphism of $\mathfrak{G}^S$ can tear apart the edge-gadgets.

Below are the gadgets v^*_S, w^*_S for two vertices $v, w \in V$, and the gadget for the edge $e \in E$ connecting them. In this example, we have $v \notin S, w \in S$, and $E(v) = \{e, f, g\}, E(w) = \{e, h, i\}$. Only the edge e is drawn. Notice that v^*_S and w^*_S look the same when we only consider their connections to the e-gadget, even though one gadget is even and the other is odd.

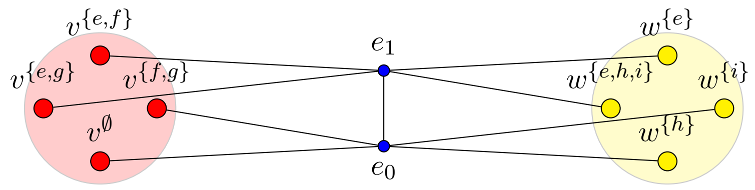


Figure 5.1: Gadgets v^*_S, w^*_S, connected by the gadget for the edge e.

If not stated otherwise, $\mathfrak{G}^S$ is unordered, i.e. its only relation is $E(\mathfrak{G}^S)$. In ordered or preordered versions of the problem, the CFI-structure has an additional relation $\prec^{\mathfrak{G}^S}$. This is usually inherited from a given order or preorder $\prec^G$ on the base graph: Each colour class of $\prec^{\mathfrak{G}^S}$ is the union of the vertex-gadgets v^*_S in the corresponding colour class of $\prec^G$. The colour class of each edge gadget in $\mathfrak{G}^S$ is determined by the colour classes of the endpoints of the edge.

The *CFI-query* asks for the parity of $|S|$, given a CFI-graph $\mathfrak{G}^S$. This is essentially the same question as the graph isomorphism problem for CFI-graphs:

Theorem 5.1.1 ([29] [40])**.** *For two given CFI-graphs over the same base graph, it holds*

$$\mathfrak{G}^S \cong \mathfrak{G}^R \text{ if and only if } |S| \equiv |R| \mod 2.$$

Alternatively, deciding the parity of $|S|$ can be phrased as a linear equation system over $\mathbb{F}_2$ in the variables $\widehat{E}$ (see for instance [10]). Since the reduction to a linear equation system is easily computable from the given CFI-graph $\mathfrak{G}^S$, and linear equation systems can be efficiently solved using, for example, Gaussian elimination, we have the following well-known result:

Theorem 5.1.2. *Let $\mathcal{G}$ be any family of undirected connected graphs. There is a* polynomial time algorithm *that receives as input (a binary encoding of) a CFI-structure $\mathfrak{G}^S$, over a base graph $G \in \mathcal{G}$, and decides the parity of $|S|$.*

Linear-algebraic logics that have a built-in ability to solve linear equation systems over $\mathbb{F}_2$ can therefore also decide the CFI-query quite easily. The reason why this problem is hard for many other logics is that the vertices e_0, e_1 within each edge gadget cannot be easily distinguished by a logic. If we knew for every edge $e \in E$, which of the vertices e_0, e_1 is which, then we could look at every vertex-gadget v^*_S, and determine whether it is odd or even by checking how it is connected to its incident edge-gadgets. Actually, we do not need to know that e_0 is the zero-vertex and e_1 the one-vertex of an edge-gadget: Even if we pretend that it is the other way round, we can still determine the parity of $|S|$ correctly. This is because if we "flip" the edge $e = \{v, w\}$ (i.e. we treat e_0 as e_1 and vice versa), then this corresponds to the isomorphic CFI-graph $\mathfrak{G}^{S \triangle \{v,w\}}$, so this makes no difference. These considerations lead to a polynomial time algorithm for the CFI-query, that is more direct than Gaussian elimination: For each edge $e \in E$, label the vertices e_0, e_1 with 0 and 1 arbitrarily (which is possible on a Turing machine) and then use these labels to determine for each $v \in V$, if v_S^* is the odd or even gadget. Simulating this algorithm in Choiceless Polynomial Time is, however, difficult: We simply do not have the possibility to choose a fixed $\{0,1\}$-assignment to all vertices in $\widehat{E}$. Nonetheless, when the base graph is linearly ordered, there is a trick to still accomplish this in CPT. This was invented in [40], where Dawar, Richerby and Rossman construct so-called *super-symmetric objects*. These deeply nested sets allow in a way to consider all possible $\{0,1\}$-assignments at once without violating polynomial bounds. In Chapter 6, we present this technique in detail. Whether CPT can decide the CFI-query on unordered instances is open, however.

For logics that lack the ability to create higher-order objects, such as bounded-variable counting logic and hence FPC, it is provably impossible to distinguish non-isomorphic CFI-graphs, provided that the treewidth of the base graphs is super-constant:

Theorem 5.1.3 ([29] [10]). *Let $G = (V, E)$ be an undirected connected graph with treewidth t. Then for any two sets $S, S' \subseteq V$, it holds*

$$\mathfrak{G}^S \equiv_{\mathcal{C}^t} \mathfrak{G}^{S'},$$

even if $\mathfrak{G}^S \not\cong \mathfrak{G}^{S'}$.

This holds because Duplicator has a winning strategy in the bijective t-pebble game on $\mathfrak{G}^S$ and $\mathfrak{G}^{S'}$. Intuitively, the difference between $\mathfrak{G}^S$ and $\mathfrak{G}^{S'}$ manifests itself in one single edge whose gadget is twisted, and the aim of Duplicator is to move this twist around in such a way that it is never exposed by the t pebbles. This can be achieved by playing similarly as the robber in the cops and robbers game which witnesses the treewidth to be at least t.

Since for any fixed FPC-sentence ψ, there is a k such that ψ cannot distinguish $\mathcal{C}^k$-equivalent structures, it follows:

Corollary 5.1.4. *Let $(G_n = (V_n, E_n))_{n \in \mathbb{N}}$ be a family of graphs whose treewidth is not bounded from above by a constant. Then there is no sentence $\psi \in$ FPC such that, for all $n \in \mathbb{N}$ and all $S \subseteq V_n$,*

$$\mathfrak{G}_n^S \models \psi \text{ iff } |S| \equiv 0 \mod 2.$$

5.2 Automorphisms of unordered CFI-graphs

For a CFI-graph $\mathfrak{G}^S$ over an *unordered* base graph $G = (V, E)$, two different kinds of automorphisms play a role: Firstly, there are what we call "CFI-automorphisms" or -isomorphisms. These are induced by swapping e_0 and e_1 in some edge-gadgets (this is called "flipping the edge"). Secondly, there are the automorphisms of the underlying graph G itself.

To speak about the CFI-isomorphisms, we use the terminology from [40]: For a given base graph G, we consider not only a concrete CFI-instance with odd and even vertex gadgets, but we can also construct the "full" CFI-graph $\mathfrak{G}$, in which every vertex gadget is both even and odd. Formally, for $v \in V$, let

$$v^* := v^*_\emptyset \cup v^*_{\{v\}} = \{v^X \mid X \subseteq E(v)\},$$

and

$$\widehat{V} := \bigcup_{v \in V} v^*.$$

The vertex-set of $\mathfrak{G}$ is $\widehat{V} \cup \widehat{E}$, and the edge-set is

$$E(\mathfrak{G}) := \{\{v^X, e_i\} \mid v^X \in \widehat{V}, e_i \in \widehat{E}, |X \cap \{e\}| = i\} \cup \{\{e_0, e_1\} \mid e \in E\}.$$

Every CFI-instance $\mathfrak{G}^S$ is an induced subgraph of $\mathfrak{G}$.
For each edge $e = \{v, w\} \in E$, let ρ_e denote the automorphism of $\mathfrak{G}$ induced by flipping the edge e. Formally, $\rho_e(e_0) = e_1, \rho_e(e_1) = e_0$, and $\rho_e(v^X) = v^{X \triangle \{e\}}$, $\rho_e(w^X) = w^{X \triangle \{e\}}$ for all $v^X, w^X \in v^* \cup w^*$. All other vertices in $\widehat{V}$ are fixed by ρ_e. One can check that this is indeed an automorphism of $\mathfrak{G}$; furthermore, ρ_e is an *isomorphism* from any CFI-instance $\mathfrak{G}^S$ to $\mathfrak{G}^{S \triangle \{v,w\}}$ (see also [40]).
It is easy to see that these edge-flip automorphisms commute, so for $F = \{e^1, ..., e^m\} \subseteq E$ we may write ρ_F for $\rho_{e^1} \circ \rho_{e^2} \circ ... \circ \rho_{e^m}$.

So in total, for every $F \subseteq E$, ρ_F is an automorphism of $\mathfrak{G}$. For any edge-set $F \subseteq E$, and $v \in V$ let $\deg_F(v) := |E(v) \cap F|$, i.e. the number of incident edges that are in F. We have $\rho_F(\mathfrak{G}^S) = \mathfrak{G}^{S \triangle T}$, where $T = \{v \in V \mid \deg_F(v) \text{ is odd }\}$. In particular, if every $v \in V$ is incident to an even number of edges in F (so F is the symmetric difference over a set of cycles in G), then ρ_F is also an automorphism of $\mathfrak{G}^S$, not only of $\mathfrak{G}$.

To sum up, we have the following groups of CFI-automorphisms of $\mathfrak{G}$ and $\mathfrak{G}^S$:

$$\mathbf{Aut}_{\text{CFI}}(\mathfrak{G}) := \{\rho_F \mid F \subseteq E\}.$$

This group is isomorphic to the Boolean vector space $\mathbb{F}_2^E$: Each $F \subseteq E$ is identified with its characteristic vector $\chi(F) \in \mathbb{F}_2^E$. It holds $\rho_F \circ \rho_{F'} = \rho_{F \triangle F'}$, and this corresponds to the vector $\chi(F) + \chi(F') \in \mathbb{F}_2^E$. It is sometimes convenient to view the group of edge-flips as this vector space.
As already said, for a CFI-instance $\mathfrak{G}^S$, i.e. an induced subgraph of $\mathfrak{G}$, we have

$$\mathbf{Aut}_{\text{CFI}}(\mathfrak{G}^S) := \{\rho_F \in \mathbf{Aut}_{\text{CFI}}(\mathfrak{G}) \mid \deg_F(v) \text{ is even for every } v \in V\}.$$

This group is isomorphic to a subspace of $\mathbb{F}_2^E$. In addition to the CFI-automorphisms, we also have to consider $\mathbf{Aut}(G) \leq \mathbf{Sym}(V)$, i.e. the automorphism group of the unordered underlying graph; this is different from the typical scenario studied in the literature, where G is ordered and so the automorphisms of $\mathfrak{G}^S$ are just given by the edge-flips.

In total, the automorphism group of the full CFI-graph $\mathfrak{G}$ is isomorphic to the following semi-direct product (recall Definition 4.1.2):

$$\mathbf{Aut}(\mathfrak{G}) \cong \mathbf{Aut}_{\text{CFI}}(\mathfrak{G}) \rtimes \mathbf{Aut}(G) = \{(\rho_F, \pi) \mid \rho_F \in \mathbf{Aut}_{\text{CFI}}(\mathfrak{G}), \pi \in \mathbf{Aut}(G)\}.$$

The action of a pair (ρ_F, π) on $V(\mathfrak{G})$ is determined by its action on $\widehat{E}$: Let $e_i \in \widehat{E}$ with $i \in \{0, 1\}$ and $e = \{v, w\} \in E$. Then $(\rho_F, \pi)(e_i) = f_j$, where $f = \{\pi(v), \pi(w)\}$, and $j = i + |F \cap \{e\}| \mod 2$. This action on $\widehat{E}$ extends to an automorphism of $\mathfrak{G}$ in a unique way. For future reference, we state the following fact about the interplay of the two constituents of $\mathbf{Aut}(\mathfrak{G})$:

Lemma 5.2.1. *For any $\pi \in \mathbf{Aut}(G)$, and any $\rho_F \in \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})$, viewed as functions on $\widehat{E}$, it holds $\rho_{\pi F} = \pi \circ \rho_F \circ \pi^{-1}$.*

Proof. Let $e_i \in \widehat{E}$ be a vertex in an edge gadget. Then $\rho_{\pi F}(e_i) = e_j$, where $j \neq i$ iff $e \in \pi F$. Moreover, $(\rho_F \circ \pi^{-1})(e_i) = (\pi^{-1}e)_k$, where $k \neq i$ iff $\pi^{-1}e \in F$. Then $\pi((\pi^{-1}e)_k) = e_k$. It holds $\pi^{-1}e \in F$ iff $e \in \pi F$. So $k \neq i$ iff $e \in \pi F$. This shows that on $\widehat{E}$, the equality $\rho_{\pi F} = \pi \circ \rho_F \circ \pi^{-1}$ is valid. □

When we consider a CFI-instance $\mathfrak{G}^S$, as opposed to $\mathfrak{G}$, then the automorphism group becomes smaller and somewhat more complicated, because now, the vertex gadgets may have different parities, so we cannot map all vertices to each other without further "corrections". To describe $\mathbf{Aut}(\mathfrak{G}^S)$ formally, we associate with each $\pi \in \mathbf{Aut}(G)$ the set $T(\pi) := \{v \in V \mid |\{v, \pi(v)\} \cap S| = 1\}$.

A *T-join* for a graph G and a vertex-set $T \subseteq V(G)$ is a subset F of the edges such that a vertex v is incident with an odd number of edges in F iff $v \in T$.
With this notion, we can describe the automorphism group of $\mathfrak{G}^S$ as follows:

Lemma 5.2.2.

$$\mathbf{Aut}(\mathfrak{G}^S) \cong \{(\rho_F, \pi) \in \mathbf{Aut}(\mathfrak{G}) \mid F \textit{ is a } T(\pi)\textit{-join in } G\}.$$

Proof. "$\supseteq$:" Let $(\rho_F, \pi) \in \mathbf{Aut}(\mathfrak{G})$ such that F is a $T(\pi)$-join in G. From the above definition of the action of (ρ_F, π), we have that $(\rho_F, \pi)(\mathfrak{G}^S) = \pi(\rho_F(\mathfrak{G}^S))$. That is, we first flip the edges in F, and then we map each edge gadget $\{e_0, e_1\}$ to $\{(\pi(e))_0, (\pi(e))_1\}$ such that e_0 goes to $(\pi(e))_0$, and e_1 to $(\pi(e))_1$. We have that $\rho_F(\mathfrak{G}^S) = \mathfrak{G}^{S \triangle T(\pi)}$. So in this CFI-graph, exactly the vertices in $S \triangle T(\pi)$ have the odd gadget. As already said, we now apply π to $\mathfrak{G}^{S \triangle T(\pi)}$ without flipping edges. So let $v \in V$ and $w := \pi(v)$. For each $v^X \in v^*_{S \triangle T(\pi)}$, we have $\pi(v^X) = w^X \in w^*$. In other words: If v and w have the same membership status in $S \triangle T(\pi)$ (i.e. both have the odd gadget, or both have the even gadget), then $\pi(v^*_{S \triangle T(\pi)}) = w^*_{S \triangle T(\pi)}$. Otherwise, if v and w differ with respect to membership in $S \triangle T(\pi)$, then $\pi(v^*_{S \triangle T(\pi)}) = w^*_{S \triangle T(\pi) \triangle \{w\}}$.
By definition of $T(\pi)$, we have $v \in T(\pi)$ iff $|\{v, w\} \cap S| = 1$. It holds: $|\{v, w\} \cap (S \triangle T(\pi))| = 1$ iff $w \in T(\pi)$. This can be seen by going through all possible combinations of memberships of v, w in S and $T(\pi)$.
Hence, π "flips" w iff $w \in T(\pi)$.
From these considerations and the fact that $\pi \in \mathbf{Aut}(G)$, it follows that $\pi(\mathfrak{G}^{S \triangle T(\pi)}) = \mathfrak{G}^S$.

"$\subseteq$:" It is easy to see that $\mathbf{Aut}(\mathfrak{G}^S)$ embeds into $\mathbf{Aut}(\mathfrak{G})$. So it suffices to take an arbitrary $(\rho_F, \pi) \in \mathbf{Aut}(\mathfrak{G})$ where F is not a $T(\pi)$-join in G, and show that this is not an automorphism of $\mathfrak{G}^S$. If F is not a $T(\pi)$-join, then there is a $v \in T(\pi)$ incident to an even number of edges in F, or a $v \notin T(\pi)$ incident to an odd number of edges in F. We consider the first case. The second case is symmetric.
Since $\deg_F(v)$ is even, either v has the odd gadget in $\mathfrak{G}^S$ and in $\rho_F(\mathfrak{G}^S)$, or it has the even gadget in both graphs. W.l.o.g. assume $v \in S$. Since $v \in T(\pi)$, it holds $\pi(v) \notin S$. Let $R \subseteq V$ be such that $\pi(\rho_F(\mathfrak{G}^S)) = \mathfrak{G}^R$. As already argued above, it follows that $\pi(v) \in R$, so $R \neq S$. Hence, the action of (ρ_F, π) on $\mathfrak{G}$ does not map the subgraph $\mathfrak{G}^S$ to itself, so it is not an automorphism of $\mathfrak{G}^S$. □

Thus, not for every edge-flip $\rho_F \in \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})$, the pair (ρ_F, id_G) is also an automorphism of $\mathfrak{G}^S$.

In Section 4.4.1, we presented a class of groups which – if they occur as automorphism groups of structures – guarantee that CPT cannot define very complex (in terms of support size) objects in these structures. These are the groups with the (k, r)-*support property.* We now show that this technique for proving lower bounds – that is, Theorem 4.4.4 – has its limitations: The automorphism groups of CFI structures fail to have the (k, r)-support property.

Lemma 5.2.3. *Let $(G_n = (V_n, E_n))_{n\in\mathbb{N}}$ be a family of connected base graphs of size $\Theta(n)$. Then for all k and r (that may be functions of n) such that for all large enough n, $2 \leq r \leq \log|\mathfrak{G}_n^S|$ and $k \leq n/r$, the group $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}_n)$ fails to have the (k, r)-support property, for all large enough n.*

Proof. Fix functions k and r such that for all large enough n, $r \leq \log|\mathfrak{G}_n^S|$ and $k \leq n/r$. According to Lemma 4.4.7, $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}_n)$ has the (k, r)-support property if and only if every kr-supported subgroup with index $\leq |\mathfrak{G}_n^S|^k$ is k-supported. We show that for large enough n, this condition is not satisfied. Let $v_1, ..., v_{kr} \in V_n$ be a collection of distinct vertices. Since the base graphs have size $\Theta(n)$, enough vertices exist. Let

$$\Gamma := \{\rho_F \mid F \text{ does not intersect any edge incident to any } v_i, \text{ for } i \in [kr]\} \leq \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}_n)$$

be the subgroup of edge flips that fix $v_1, ..., v_{kr}$. Clearly, this group is kr-supported because we can form a support by fixing one vertex in each vertex gadget v_i^*, for all $i \in [kr]$. The index is $[\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}_n) : \Gamma] = 2^{|E_n|}/2^{|E_n|-kr} = 2^{kr}$. Since $r \leq \log|\mathfrak{G}_n^S|$, this is at most $|\mathfrak{G}_n^S|^k$. But Γ is not k-supported: Since $r \geq 2$, it holds $k < kr$, and it is obvious that any support of Γ must contain at least one vertex of $V(\mathfrak{G}_n^S)$ per v_i, for $i \in [kr]$. So kr is the smallest support size of Γ. Thus, by Lemma 4.4.7, $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}_n)$ does not have the (k, r)-support property. □

Recall Theorem 4.4.4. It says that no CPT-program can activate sets of support size $> k$ on a sequence of structures whose automorphism groups have the (k, r)-support property, for every constant r. The above lemma rules out the (k, r)-support property for all constant values of r, and linear k. Therefore, Theorem 4.4.4 cannot be used to disprove the CPT-definability of sets with linear support in CFI-graphs. As we will see in the next chapter, the non-definability of the CFI-query could be shown by proving the non-definability of sets with linear support (Theorem 6.2.7). Thus, the (k, r)-support property as a group-theoretic criterion by itself is probably not sufficient to prove the non-definability of the CFI-query.

5.3 CFI-structures over hypercubes

A particular family of base graphs that we consider because of their symmetries are n-dimensional hypercubes. Let $\mathcal{H}_n = (\{0,1\}^n, E_n)$ denote the n-dimensional hypercube. For binary words u and v, we denote by $d(u, v)$ their Hamming-distance, i.e. the number of positions at which the words differ. The (undirected) edges of the hypercube connect words with Hamming-distance exactly one, so $E_n := \{\{u, v\} \in \{0,1\}^n \mid d(u, v) = 1\}$.

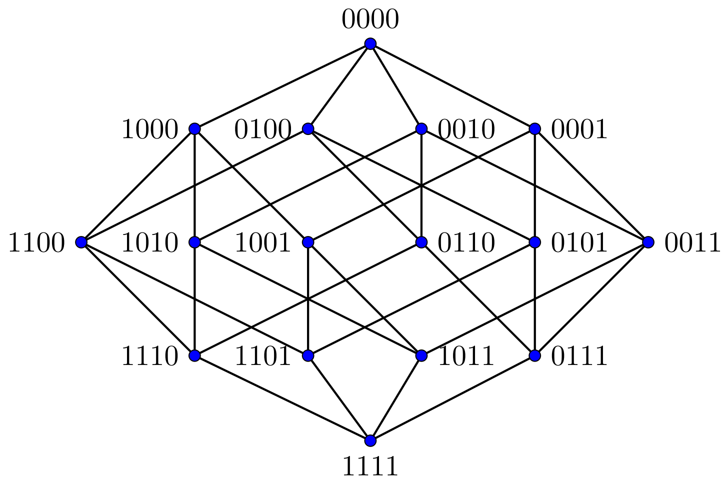


Figure 5.2: The 4-dimensional hypercube.

The automorphisms of $\mathcal{H}_n$ consist in "rotations" of the cube and permutations of the neighbourhood of the vertex 0^n. The rotation is fully specified by saying which vertex v is mapped to 0^n. Permuting the neighbourhood of 0^n according to any permutation in $\mathbf{Sym}_n$ is propagated through the hypercube and fully determines the automorphism. Formally, this group can be written as the following semi-direct product [69]:

$$\mathbf{Aut}(\mathcal{H}_n) = \mathbb{F}_2^n \rtimes \mathbf{Sym}_n.$$

For a pair $(\mathbf{w}, \pi) \in \mathbf{Aut}(\mathcal{H}_n)$, and a vertex $v \in \{0,1\}^n$, $(\mathbf{v}, \pi)(v) = \pi(v + \mathbf{w})$. Here, $v + \mathbf{w}$ is the result of the position-wise XOR of v and $\mathbf{w}$. For any word $v \in \{0,1\}^n$, $\pi(v)$ is obtained from v by permuting its positions according to π, i.e. $\pi(v) = v_{\pi^{-1}(1)} v_{\pi^{-1}(2)} ... v_{\pi^{-1}(n)}$.

We have $|\mathbf{Aut}(\mathcal{H}_n)| = 2^n \cdot n!$. This is super-polynomial (in fact, quasipolynomial) in the size of the hypercube, which is $2^n = |\{0,1\}^n|$. Therefore, it is indeed possible for h.f. objects over hypercubes to have orbits of super-polynomial size – if this were not so, then CFI-structures over hypercubes would not be an interesting candidate for proving lower bounds against CPT. Actually, the super-polynomial size of the automorphism group is due to the symmetric group acting on the positions of the binary strings, and the subgroup $\mathbb{F}_2^n$ does not matter for this size. Therefore, to simplify matters, we will often pretend that the vertex 0^n is fixed, so the automorphism group is just $\mathbf{Sym}_n$. If we can show that certain objects over $\mathcal{H}_n$ have a super-polynomial orbit with respect to $\mathbf{Sym}_n$, then this is also true for the whole group $\mathbf{Aut}(\mathcal{H}_n)$, so this simplification is not problematic.

For $S \subseteq \{0,1\}^n$, we denote the unordered CFI-structure with base graph $\mathcal{H}_n$ as $\mathfrak{H}_n^S$. Its automorphism group is the semi-direct product of $\mathbf{Aut}_{\text{CFI}}(\mathfrak{H}_n^S)$ by $\mathbf{Aut}(\mathcal{H}_n)$, as explained in the previous section. The treewidth of n-dimensional hypercubes is slightly sublinear in their size but still so large that distinguishing non-isomorphic hypercube CFI-structures is difficult (and in particular, impossible in FPC):

Lemma 5.3.1 (Theorem 5 in [102])**.** *The* treewidth *of the n-dimensional hypercube $\mathcal{H}_n$ is a function in $\Theta(2^n/\sqrt{n})$.*

We will get back to hypercube CFI-structures in Chapter 7 and in more detail in Chapter 9; based on our results, we conjecture that the CFI-query on the structures $(\mathfrak{H}_n^S)_{n\in\mathbb{N}}$ is a particularly hard problem for CPT.

6 Prior research on the choiceless (non-)definability of Cai-Fürer-Immerman problems

This chapter is mostly a review of “Choiceless polynomial time, counting and the Cai-Fürer-Immerman graphs” from 2008 by Dawar, Richerby and Rossman [40]. This paper is the main starting point and inspiration for many of our investigations. Its relevance is two-fold: Firstly, it introduces a “design principle” for h.f. sets over CFI graphs which allows to compute the parity of a given ordered CFI instance in CPT. These objects are called *super-symmetric* and the underlying idea of their construction has been picked up in subsequent works ([91], [89]) to tackle the CFI query on different variants of ordered structures (or base graphs with large degree). All known CPT algorithms for the CFI query or for canonisation of certain restricted structure classes can be traced back to this idea of super-symmetric objects. A large part of this thesis is devoted to understanding the limitations of this technique, which is why we review it in detail in this chapter.

A second important contribution of [40] is a fundamental argument that can be used to prove limitations for the power of CPT. Namely, the classical bijective k-pebble game, which characterises the power of counting logic and hence FPC, can be lifted to CPT. In some sense, the distinguishing power of any CPT-program on a given class of (homogeneous) structures depends on the *support size* of the objects the program activates. It is shown that this support size is related to the number of pebbles in the bijective pebble game and thus controls the expressive power of the CPT program. As a consequence, any CPT-program that decides the CFI query must activate a set with large support. We use this fact in Chapters 8 and 9 to develop a circuit-based lower bound approach for the power of CPT on CFI-graphs.

6.1 Positive results: The super-symmetric object technique

6.1.1 Defining the CFI query on linearly ordered base graphs

Recall that the CFI query on linearly ordered base graphs is the following problem: Given a CFI graph $\mathfrak{G}^S$ over some linearly ordered graph G, determine the parity of $|S|$. The linear order on $V(G)$ is translated into a preorder $\prec$ on $V(\mathfrak{G}^S)$, which is part of the vocabulary of $\mathfrak{G}^S$. Any two vertices within the same vertex or edge gadget of $\mathfrak{G}^S$ are incomparable with respect to $\prec$, and vertices in distinct edge or vertex gadgets are compared with respect to their order in G (where the order extends to the edges via the order of their endpoints).

The approach for deciding the CFI query on such instances consists of two phases: In the first phase, a highly nested h.f. set μ is constructed that encodes in its structure the parity of $|S|$. In the second phase, this parity is extracted from μ. The authors of [40] have called μ a *super-symmetric object*, and this is really the point of the construction. Namely, the set μ is stabilised by all automorphisms in $\mathbf{Aut}_{\text{CFI}}(\mathfrak{G})$, which are in general more than the automorphisms of the instance $\mathfrak{G}^S$. As we explained in the previous chapter, $\mathbf{Aut}_{\text{CFI}}(\mathfrak{G})$ consists of *all* edge flips, whereas only the edge flips along cycles in G are automorphisms of the structure $\mathfrak{G}^S$. The other edge flips correspond to isomorphisms from $\mathfrak{G}^S$ into isomorphic CFI graphs. The reason why such a super-symmetric object is algorithmically useful is because it allows us to determine the parity of $|S|$ by trying out all interpretations of the vertices in edge gadgets: For each $e \in E(G)$, we have the vertices e_0 and e_1 in the gadget of e. Once we have fixed for each edge which one is which, it is easy to decide which vertex gadget is an even gadget and which is an odd gadget by just looking at the neighbourhoods of the nodes in each vertex gadget v^*_S and checking if these nodes $v^X \in v^*_S$ are connected to an even or odd number of e_1 nodes. Actually, for the parity of $|S|$, it does not even make a difference which way round we interpret the nodes e_0 and e_1 in each edge gadget, because flipping any edge just leads to an isomorphic CFI graph where the parity of the odd vertex gadgets is the same. Therefore, the problem is really to fix an assignment of e_0 and e_1 to 0 and 1 in every edge in a choiceless fashion. Because e_0 and e_1 are symmetric, we cannot expect this to be possible, and therefore, we have to try out both the assignments $e_0 \mapsto 0, e_1 \mapsto 1$ and $e_0 \mapsto 1, e_1 \mapsto 0$ for each edge. This is why we need the super-symmetric object μ: Because it is symmetric under all possible edge flips, we can replace all atoms e_0, e_1 in its transitive closure with 0 and 1 both ways round, and both these replacements will result in the same object. So once we have μ, we can iterate over the edges in $E(G)$ (using the linear order on G) and replace, for each edge e, e_0 and e_1 with 0 and 1 in both ways, which will always result in the same h.f. object. In the end, we obtain a unique object in which all atoms are 0 or 1. From this, it will be possible to extract the parity of $|S|$ quite easily. If μ were not super-symmetric, then this procedure would not necessarily remain in polynomial bounds because it would produce a greater number of distinct objects.

So the most challenging part is to come up with a h.f. set μ that both contains the relevant information about the parity of $|S|$ and is super-symmetric (and of course, CPT-definable in the structure $\mathfrak{G}^S$). Dawar, Richerby and Rossman solved this by building μ out of sets that represent the vertex gadgets in $\mathfrak{G}^S$. For every $v \in V(G)$, the set τ^S_v is defined as:

$$\tau^S_v := \Big\{ N_\prec(v^X) : X \subseteq E(v) \mid X \text{ is even iff } v \notin S \Big\}$$
$$\tilde{\tau}^S_v := \Big\{ N_\prec(v^X) : X \subseteq E(v) \mid X \text{ is even iff } v \in S \Big\}.$$

Here, $N_\prec(v^X)$ is an ordered tuple (with respect to $\prec$) containing the neighbourhood of

v^X. For example, if the edges incident to v in G are $E(v) = \{e, f, g\}$, and $v \in S$, then

$$\tau_v^S := \Big\{\langle e_1, f_0, g_0\rangle, \langle e_0, f_1, g_0\rangle, \langle e_0, f_0, g_1\rangle, \langle e_1, f_1, g_1\rangle\Big\}$$
$$\tilde{\tau}_v^S := \Big\{\langle e_0, f_0, g_0\rangle, \langle e_1, f_1, g_0\rangle, \langle e_1, f_0, g_1\rangle, \langle e_0, f_1, g_1\rangle\Big\}.$$

Essentially, the set τ_v^S corresponds to the vertex gadget v_S^* in $\mathfrak{G}^S$ (see Section 5.1). Its counterpart $\tilde{\tau}_v^S$ can be thought of as the vertex gadget $v^*_{S\triangle\{v\}}$, so this is exactly the opposite type of gadget than the one we have for v in $\mathfrak{G}^S$. Thus, we define h.f. sets not only for the vertex gadgets that actually exist in $\mathfrak{G}^S$, but also for their flipped versions. This is necessary because we want our final object to be stabilised also by edge flips in $\mathbf{Aut}_{\text{CFI}}(\mathfrak{G})$, and these might flip vertex gadgets as well.

Now in the super-symmetric object that we would like to construct, we aggregate these τ_v^S and $\tilde{\tau}_v^S$ in the order given by $\prec$. Let $v_1, v_2, ..., v_n$ be the ordered vertices of G, and let $\tau_i^S := \tau_{v_i}^S$. We define $\mu_1^S := \tau_1^S$ and $\tilde{\mu}_1^S := \tilde{\tau}_1^S$. Then for $i \in [n-1]$, let

$$\mu_{i+1}^S := \Big\{\langle \mu_i^S, \tau_{i+1}^S\rangle, \langle \tilde{\mu}_i^S, \tilde{\tau}_{i+1}^S\rangle\Big\}$$
$$\tilde{\mu}_{i+1}^S := \Big\{\langle \mu_i^S, \tilde{\tau}_{i+1}^S\rangle, \langle \tilde{\mu}_i^S, \tau_{i+1}^S\rangle\Big\}.$$

In some sense, μ_{i+1}^S adds the vertex v_{i+1} to the "parity count" of μ_i^S: Any edge flip that flips both μ_i^S *and* τ_{i+1}^S at the same time is "hidden" by the set μ_{i+1}^S and has no effect on it. Thus, intuitively, μ_{i+1}^S implements the XOR over μ_i^S and τ_{i+1}^S. This can be made precise and it can be shown inductively (Lemma 12 in [40]) that for all $S, T \subseteq V(G)$, and all $k \in [n]$, it holds:

$$\mu_k^S = \mu_k^T \Leftrightarrow \tilde{\mu}_k^S = \tilde{\mu}_k^S \Leftrightarrow |S \cap \{v_1, ..., v_k\}| \equiv |T \cap \{v_1, ..., v_k\}| \mod 2.$$
$$\mu_k^S = \tilde{\mu}_k^T \Leftrightarrow \tilde{\mu}_k^S = \mu_k^S \Leftrightarrow |S \cap \{v_1, ..., v_k\}| \not\equiv |T \cap \{v_1, ..., v_k\}| \mod 2.$$

So in this sense, μ_k^S tracks how many vertices among the first k ones have an odd gadget, i.e. are in S. If an odd number of vertex gadgets in $\{v_1, v_2, ..., v_k\}$ change their parity, then μ_k^S flips and becomes $\tilde{\mu}_k^S$. This can happen for example if $\rho_e \in \mathbf{Aut}_{\text{CFI}}(\mathfrak{G})$ is applied to μ_k^S, for some edge $e \in E(G)$ that has one endpoint in $\{v_1, ..., v_k\}$ and one endpoint outside. Thus, $\{\mu_k^S, \tilde{\mu}_k^S\}$ forms an $\mathbf{Aut}_{\text{CFI}}(\mathfrak{G})$-orbit, whose two elements – informally speaking – stand for the two parities odd and even. Note that the $\mathbf{Aut}_{\text{CFI}}(\mathfrak{G}^S)$-orbit of μ_k^S is a singleton because $\mathbf{Aut}_{\text{CFI}}(\mathfrak{G}^S)$ stabilises each vertex gadget and hence each τ_i^S. The final object, μ_n^S, is super-symmetric because it is stabilised by all edge flips in $\mathbf{Aut}_{\text{CFI}}(\mathfrak{G})$. This follows directly from the result mentioned above because for every edge flip $\rho_e \in \mathbf{Aut}_{\text{CFI}}(\mathfrak{G})$, we have $\rho_e(\mu_n^S) = \mu_n^{S\triangle\{u,v\}}$, if u and v are the endpoints of e. Since $|S\triangle\{u,v\}| \equiv |S| \mod 2$, it follows that $\rho_e(\mu_n^S) = \mu_n^S$. Moreover, as shown in Lemma 11 in [40], the object μ_n^S can be constructed in CPT, given the input structure $\mathfrak{G}^S$. Doing this in CPT presupposes the linear order on the base graph because constructing μ_{i+1} requires having constructed μ_i first. Actually, it can be seen that the structure of the h.f. set μ_n is that of a linear order on $V(G)$. Each τ_i occurs in μ_n at a distinct nesting depth, so constructing μ_n in a completely unordered CFI instance would be tantamount to computing a linear order on the base graph. This is why this particular approach is limited to the CFI query on totally ordered base graphs.

This finishes the first phase of the algorithm. In the second phase, the atoms in μ_n^S are removed and replaced by constants 0 and 1 (these can for example be encoded as h.f. sets $\emptyset$ and $\{\emptyset\}$). We omit the technical details but the idea is to again iterate over the edges of G according to the linear order on the base graph and to apply, for each edge e, two operations to μ_n^S: First, replace every occurrence of the atom e_0 in μ_n^S with 0, and each atom e_1 with 1. After that (or simultaneously), we perform the same substitutions in μ_n^S but with 0 and 1 swapped. Together, these two replacements are isomorphism-invariant and as explained above, they yield the same object because μ_n^S is super-symmetric. Using the Unique-operator of CPT, we can obtain this unique result of the replacement. After we have done this for every edge in $E(G)$, we are left with an object in $\text{HF}(\{0,1\})$, which is called $B(\mu_n^S)$ in [40].

It remains to extract the parity of $|S|$ from $B(\mu_n^S)$. This is done using a parity function $p : \text{HF}(\{0,1\}) \longrightarrow \{0,1\}$ that is recursively applied to the elements of the transitive closure of $B(\mu_n^S)$. For atoms 0 and 1, we let $p(0) := 0$, $p(1) := 1$. For sets $y = \{x_1, ..., x_k\}$, we let $p(y) := \prod_{i=1}^k p(x_i)$, and for tuples $y = \langle x_1, ..., x_k\rangle$, $p(y) := \sum_{i=1}^k x_i \mod 2$. It is not difficult to implement this function in CPT. One can check that it indeed computes the parity of $|S|$ correctly: For every τ_v^S in $\text{tc}(\mu_n^S)$, every tuple in τ_v^S evaluates to the same value, and this value is 1 if and only if $v \in S$. Similarly, $p(\tilde{\tau}_v^S) = 1$ if and only if $v \notin S$. For μ_{i+1}^S, we can again see that the p-value of both tuples in it is always the same, so the product of these values is 1 if and only if $p(\langle \mu_i^S, \tau_{i+1}^S\rangle) = p(\langle \tilde{\mu}_i^S, \tilde{\tau}_{i+1}^S\rangle) = 1$. By induction, this holds if and only if $|S \cap \{v_1, ..., v_{i+1}\}|$ is odd (because $p(\mu_i^S)$ is the parity of $|S \cap \{v_1, ..., v_i\}|$). Therefore, $p(\mu_n^S) = |S| \mod 2$.

In total, we have:

Theorem 6.1.1 (Theorem 17 in [40]). *There is a* CPT-*algorithm that outputs* $|S| \mod 2$ *on a given CFI structure* $\mathfrak{G}^S = (V(\mathfrak{G}^S), E(\mathfrak{G}^S), \prec)$ *over a linearly ordered base graph* $G = (V(G), E(G), <)$.

It should be noted that this CPT-program can even be implemented without counting, so it is actually in CPT$^-$. Again, we stress that the linear order on the base graph is indispensable for this algorithm. It is needed both for the construction of μ_n^S and for the replacement of the atoms with zero and one: Even though each of the $2^{|E|}$ many different replacements leaves the super-symmetric object invariant, these would still be exponentially many computation steps if they all had to be done simultaneously. As we will see next, the technique can be generalised to an extent such that it also works on less ordered CFI-structures. This requires to modify the definition of the super-symmetric objects a bit and also to perform the edge replacements with a different strategy. However, without any kind of ordering on the input, it is not clear if any of the two phases of the algorithm could be implemented in CPT (unless the CFI-structures contain implicit padding, as in Section 6.1.3).

6.1.2 Defining the CFI query on preordered base graphs

In [91], Wied Pakusa, Svenja Schalthöfer and Erkal Selman generalised the above algorithm. There, it is applied to CFI-graphs $\mathfrak{G}^S$ equipped with a preorder $\prec$ on $V(\mathfrak{G}^S)$ which originates from a *preorder* on the base graph. The preorder on $V(G)$ orders the base graph up to colour classes whose size is bounded by $\log |V(G)|$. Thus, in the corresponding preorder in the CFI graph, up to $\log |V(G)|$ many vertex gadgets are together in a colour class. The edge gadgets are again ordered according to their endpoints, so edges between vertices in the same colour class are incomparable.

The main difference from the algorithm for linearly ordered base graphs is the definition of the μ- and $\tilde{\mu}$-objects. Their basic building blocks, the sets τ_v^S and $\tilde{\tau}_v^S$ are the same as in the other algorithm (with a slight modification that is needed for implementation purposes, which we ignore here). In the linearly ordered setting, we had μ_i and $\tilde{\mu}_i$ for every vertex $v_i \in V(G)$. Now we have one such set for every subset of each colour class. Formally, let $C_1, ..., C_m \subseteq V(G)$ be the ordered collection of colour classes of G. For each $i \in [m]$, let $\mathcal{M}_i$ be the set of all $M \subseteq V(G)$ such that $\bigcup_{j<i} C_j \subseteq M$ and $M \subseteq \bigcup_{j \leq i} C_j$. In other words, $\mathcal{M}_i$ is the set of all subsets of C_i, each taken together with $\bigcup_{j<i} C_j$. Let $\mathcal{M}$ be the set of all $M \subseteq V$ that are in $\mathcal{M}_i$, for some i. For every $M \in \mathcal{M}$, we have objects μ_M^S and $\tilde{\mu}_M^S$. In the base case, M is a set of size 1. Then μ_M is simply the corresponding τ-object. For larger sets M, the object μ_M^S is built from a τ-object for some $v \in M$ and the μ-object for $M \setminus \{v\}$ that we have by induction. To this end, we need intermediate "aggregation objects" $\mu_{M,v}$, which add together the parity information of M and v, similarly as it is done in μ_{i+1} in the previous subsection.

$$\begin{aligned}
\mu_{\{v\}}^S &:= \tau_v^S \\
\tilde{\mu}_{\{v\}}^S &:= \tilde{\tau}_v^S \\
\mu_{M,v}^S &:= \Big\{ \langle \mu_M^S, \tau_v^S \rangle, \langle \tilde{\mu}_M^S, \tau_v^S \rangle \Big\} \\
\tilde{\mu}_{M,v}^S &:= \Big\{ \langle \mu_M^S, \tilde{\tau}_v^S \rangle, \langle \tilde{\mu}_M^S, \tau_v^S \rangle \Big\}.
\end{aligned}$$

With these, we can define μ_M^S for all M of size ≥ 2 by splitting M into a single vertex w and the rest, in all possible ways (because we cannot make a choice for w):

$$\begin{aligned}
\mu_M^S &:= \Big\{ \mu_{N,w}^S \mid N \in \mathcal{M} \text{ and } N \uplus \{w\} = M \Big\} \\
\tilde{\mu}_M^S &:= \Big\{ \tilde{\mu}_{N,w}^S \mid N \in \mathcal{M} \text{ and } N \uplus \{w\} = M \Big\}.
\end{aligned}$$

The final object is then $\mu^S_{G(V)}$. It can be shown that these objects are CPT-definable in $\mathfrak{G}^S$ by iterating over the colour classes according to the preorder. The size of $\text{tc}(\mu^S_{V(G)})$ is polynomial in $|\mathfrak{G}^S|$ because each colour class has $\leq 2^{\log |V(G)|} = |V(G)|$ many subsets. All these subsets appear in the construction of $\mu^S_{V(G)}$, which is why this particular algorithm cannot work for preorders with larger colour classes. Interestingly, as we will see in Chapter 8, this is in some sense best-possible: There, we show that all h.f. sets that are constructed in this parity-aggregating fashion (this can be made precise) correspond to XOR-circuits in a well-defined way. In a certain sense, these XOR-circuits can at most aggregate a logarithmic number of parities in each gate because otherwise, the polynomial bound on the orbit size of the corresponding h.f. sets would be violated. This loosely fits to the fact that the μ-objects here also essentially aggregate parities in log-sized chunks. Note also that the object $\mu^S_{V(G)}$ implicitly defines in its structure the preorder on G because different colour classes appear at different nesting depths in the h.f. set. For this reason, the algorithm cannot work on unordered CFI instances unless they admit a CPT-definable preorder with log-sized colour classes. In the next chapter, we will show that n-dimensional hypercubes are a class of graphs that *do not* admit such CPT-definable preorders.

As for the μ-objects in the previous algorithm, it can be shown inductively (Lemma 5.5 in [91]) that they have the desired behaviour with respect to $|S| \mod 2$: For all $M \in \mathcal{M}$,

$$\mu^S_M = \mu^T_M \Leftrightarrow \tilde{\mu}^S_M = \tilde{\mu}^S_M \Leftrightarrow |S \cap M| \equiv |T \cap M| \mod 2.$$
$$\mu^S_M = \tilde{\mu}^T_M \Leftrightarrow \tilde{\mu}^S_M = \mu^S_M \Leftrightarrow |S \cap M| \not\equiv |T \cap M| \mod 2.$$

So μ^S_M represents the parity of S in M. The second phase of the algorithm is again the replacement of atoms with 0 and 1 in $\mu^S_{V(G)}$. This is now a bit more complicated than in the previous algorithm because we have no linear order on the edges. Therefore, the mapping $e_0 \mapsto 0, e_1 \mapsto 1$ (and vice versa) is not applied edge by edge but we again consider, for each $M \in \mathcal{M}$, all partitions of the form $M = N \uplus \{w\}$. Then we process all edges in the cut between N and w simultaneously, and apply all possible 0-1-assignments to them. This is not too high a number of simultaneous assignments because the CFI gadget w^* contains one node for each subset of incident edges, so $2^{|N_G(w)|}$ is polynomial in $|\mathfrak{G}^S|$. All these replacements of edges in the cut in all possible ways result in a unique object again, by super-symmetry. So we can remove all edges from $\mu^S_{V(G)}$ in the same recursive manner as we constructed μ: Process all cuts of the form $M = N \uplus \{w\}$, for all $M \in \mathcal{M}$, in the same order as in the construction of μ. The details are a bit technical, so we refer to [91] for this. After all atoms have been removed, it is possible to compute the parity of $|S|$ with the same recursive function p as in the previous algorithm: Whenever we have a tuple $\langle x_1, ..., x_k \rangle$, we take the sum over the parities, and for sets, we take the product over its members. Altogether, this shows:

Theorem 6.1.2 (Theorem 5.1 in [91])**.** *Let $\mathcal{K}$ be a class of connected preordered base graphs $G = (V, E, \prec)$ such that the size of each colour class is at most $\log |V|$. There exists a* CPT*-program that computes the parity of $|S|$ in each input structure $\mathfrak{G}^S$ over a base graph from $\mathcal{K}$.*

6.1.3 Defining the CFI query on base graphs of large degree

Finally, there is another variation of the "super-symmetric technique" that was also presented in [91]. This time, the CFI structures are not ordered at all, but the base graphs contain at least one vertex of linear degree. The CFI gadget of such a linear-degree vertex contains exponentially many nodes. So one could say that these CFI structures are implicitly padded and hence, we may use much larger h.f. sets in the algorithm that decides the CFI query.

Theorem 6.1.3 (Theorem 6.1 in [91]). *For every $k \in \mathbb{N}$, the CFI query over unordered base graphs $G = (V, E)$ with maximal degree $\geq \frac{|V|}{k}$ is* CPT*-definable, using only sets of constant rank not depending on k.*

The μ-objects for this algorithm are actually quite simple because as already explained, we now have the resources to consider all $2^{|V|}$ many subsets of the vertices, and so, there is no need to be careful about space.
Let $\mu^S_{\{v\}} = \{\{\tau^S_v\}\}$ and $\tilde{\mu}^S_{\{v\}} = \{\{\tilde{\tau}^S_v\}\}$, and for every M with $|M| \geq 2$, define $\overline{\mu}^S_M := \{\{\overline{\tau}^S_v \mid v \in M\} \mid \overline{\tau}^S_v \in \{\tau^S_v, \tilde{\tau}^S_v\}\}$. Then, for every $M \subseteq V$, let:

$$\mu^S_M := \Big\{ m \in \overline{\mu}^S_M \mid \text{ the number of } \tilde{\tau}^S_v \text{ in } m \text{ is even} \Big\}$$
$$\tilde{\mu}^S_M := \Big\{ m \in \overline{\mu}^S_M \mid \text{ the number of } \tilde{\tau}^S_v \text{ in } m \text{ is odd} \Big\}.$$

It is not hard to see that for every $M \subseteq V$, μ^S_M characterises the parity of $|S \cap M|$ in the same way as before because μ^S_M is stabilised by any even number of "vertex flips" in M.

Removing the edge atoms from μ^S_V is again done by processing all edges in a cut between some set $N \subseteq V$ and a vertex w. Because no such w can be chosen in CPT, this has to be done by iteratively enlarging the sets M, from size 1 to $|V|$. So we do not construct μ^S_V at once (even though it would be possible) but instead we define first, for every $M \subseteq V$, the sets μ^S_M and $\tilde{\mu}^S_M$ by combining μ^S_N and $\tilde{\mu}^S_N$ with τ^S_w, $\tilde{\tau}^S_w$, respectively, for all partitions $M = N \uplus \{w\}$, while removing the edge atoms along the cut between N and w. For all these partitions, we get the same result, namely μ^S_M or $\tilde{\mu}^S_M$, with all edges inside M removed. In the end, we can again apply a recursive parity function to μ^S_V, after all atoms have been replaced by 0 or 1.
The existence of this algorithm is the reason why the CFI-query over complete graphs is not a separating example for P and CPT, despite its high symmetry.

6.2 Negative results: Lower bounds on the support size

As we have seen, it is indeed possible to define the CFI-query in CPT at least for certain classes of base graphs. The algorithms we presented make use of quite complicated h.f. set constructions, which are deeply nested (except in the linear degree case). In [40], Dawar, Richerby and Rossman not only developed the concept of super-symmetric objects but also managed to prove that solving the CFI-query in CPT actually requires the construction of such, let us say, "non-trivial" h.f. sets. Concretely, they proved that a h.f. set with *large support* must be activated by every CPT-program that decides the CFI-query. Moreover, they showed that such sets with large support must have a super-constant nesting depth. We follow the entire line of argument that leads to this result. Some of the more straightforward proofs will be omitted. We stick with the notation from [40], so in particular, structures are usually called I or J, and the same letter is used for their universe.

Let I be a σ-structure and $k \in \mathbb{N}$. The *transitively k-supported* elements of $\text{HF}(I)$ are those sets x such that every $y \in \text{tc}(x)$ has a support of size at most k in I. We denote by $\overline{I}_k$ the set of all transitively k-supported objects in I, and also the corresponding structure with vocabulary $\sigma \cup \{\in, \emptyset\}$. It is not necessary to define this structure here in detail; it is clear that $\text{HF}(I)$ and subsets thereof have such a structural representation.

Simulation of CPT in FPC

The first step towards the support lower bound is a simulation of CPT in fixed-point logic with counting. Since CPT is known to be more powerful than FPC, such a simulation is of course not possible in a general sense. But what is shown in [40] is that if we enrich the input structure I with all h.f. objects that a given CPT-program Π activates on input I, then on that enriched input structure, the evaluation of Π reduces to a fixed-point process and is therefore FPC-definable.

Lemma 6.2.1 (Corollary 21 in [40])**.** *Let Π be a CPT-program and I an input structure. There is a formula $\varphi \in \text{FPC}$ such that $\text{Active}^+(I) \models \varphi$ if and only if Π accepts I.*

Here, $\text{Active}^+(I)$ denotes the structural representation of all objects that Π activates during its run on I. The $+$ stands for the numbers, so formally, this is a subset of $\text{HF}(I \cup N)$, where N is some structure that encodes a suitable ordered initial segment of $\mathbb{N}$. The proof of this result is more or less straightforward. The program Π is viewed as an abstract state machine whose computation stages can be encoded as a relation over $\text{Active}^+(I)$ (specifically because numbers are available in this structure). The state update is definable, so the whole computation is expressible in FPC.

This simulation in FPC will allow us to use the bijective k-pebble game in order to show that CPT cannot distinguish certain structures – at least when we restrict the set of active objects:

Corollary 6.2.2. *Let Π be a CPT-program and I a structure that contains a sufficiently long initial segment of $\mathbb{N}$ to encode the numbers that may occur in the evaluation of Π on I. Let $k \in \mathbb{N}$ such that Π activates only sets with a support of size $\leq k$ on input I. Then there is a formula $\varphi \in \text{FPC}$ such that $\overline{I}_k \models \varphi$ if and only if Π accepts I.*

This follows immediately from Lemma 6.2.1. The extra requirement that I contain an initial segment of $\mathbb{N}$ is needed for the counting. As mentioned on page 46 in [40], we can always assume the presence of such a linear order because it does not change the essential properties of the structure.

At this point, let us already state the main technical theorem, the proof of which we are presenting here. It transfers $\mathcal{C}^{mk}$-equivalence of structures to $\mathcal{C}^{m}$-equivalence of their augmentations with all transitively k-supported objects. Together with the previous corollary, it means that we can indeed use pebble games to bound the power of CPT as long as the CPT-programs only activate sets of bounded support.

Theorem 6.2.3 (Theorem 33 in [40]). *Let $k \geq 0$ and $m > 1$ and let I and J be $\mathcal{C}^{mk}$-homogeneous structures of the same vocabulary. If $I \equiv^{\mathcal{C}^{mk}} J$, then $\overline{I}_k \equiv^{\mathcal{C}^{m}} \overline{J}_k$.*
Remark*: It is sufficient to have $\mathcal{C}^{mk}$-homogeneity for all tuples of length $\leq 2k$.*

The proof requires some preparation. First of all, we introduce the concept of homogeneity because the theorem requires this condition on the structures. However, it seems that homogeneity does not play a central role in the proof and perhaps, this condition just simplifies matters, but could in principle be dispensed with. We will comment on this question later.

Homogeneity, forms and molecules

Definition 6.2.4 (Types and homogeneity, Definition 32 in [40]). *The* $\mathcal{C}^{m}$-type *of a tuple $\overline{a}$ in a structure I is the collection of $\mathcal{C}^{m}$-formulas that are true in $(I, \overline{a})$.*
A structure I is $\mathcal{C}^{m}$-homogeneous if, whenever tuples $\overline{a}$ and $\overline{b}$ of length $\leq m$ have the same $\mathcal{C}^{m}$-type in I, then there is an automorphism of I that maps $\overline{a}$ to $\overline{b}$.

We often use homogeneity in a slightly weaker sense and only require that the condition be satisfied by all tuples of some shorter length than m. In that case, we mention this explicitly (as in the theorem above). Note that homogeneity is "upwards closed" but not "downwards closed". If a structure is $\mathcal{C}^{m}$-homogeneous, it is not necessarily homogeneous for values $< m$, but it is then $\mathcal{C}^{\geq m}$-homogeneous. This is because the partition of m-tuples into $\mathcal{C}^{m'}$-types is equal to or coarser than the orbit partition of I^m, and increasing m' makes the partition finer while decreasing m' makes it coarser.

The reason why homogeneity is needed for Theorem 6.2.3 is that on homogeneous structures I, the elements of $\overline{I}_k$ admit a very useful representation. Every transitively k-supported set can be written as a combination of *forms* and *molecules*. This idea goes back to the paper that introduced CPT [21]. A *form* can be seen as a description or a template of a set, which takes as "input" a *molecule* (this is literally a sequence of atoms). Form and molecule together specify a concrete h.f. set. Interestingly, forms involve types. Therefore, the objects that can be expressed by forms are inherently symmetric with respect to types. This is also a property of CPT-definable objects, which becomes perhaps most clear in Chapter 10, where we take a look at the *Deep Weisfeiler Leman* computation model.

A *molecule* on a structure I is a sequence $\alpha = \alpha_1 ... \alpha_k$ of atoms. Here, k is supposed to be the fixed number from Theorem 6.2.3 because we are going to use supports as molecules.

For the definition of forms, we fix some list $c_1, ..., c_k$ of new symbols. The set of *forms* is the least set containing each of the c_i and every finite set of pairs (φ, τ), where φ is a form and τ is a $\mathcal{C}^{mk}$-type of tuples of length exactly $2k$. In the following, when we write $\text{tp}_I(\alpha, \beta)$ for two molecules α and β, we mean the $\mathcal{C}^{mk}$-type of the tuple $\alpha\beta$ in I.

In a given structure I, a form φ and a molecule α together define a h.f. set in $\text{HF}(I)$, which is called the *denotation* $\varphi \star \alpha$ of φ and α. The inductive definition is $c_i \star \alpha := \alpha_i$ and $\varphi \star \alpha := \{\psi \star \beta \mid (\psi, \text{tp}_I(\alpha, \beta)) \in \varphi\}$, if φ is a set.

Now it is shown in [40] that $\overline{I}_k$ is exactly the set of objects which are representable as the denotation of a form and a molecule (of length k). First of all, it is proven via induction on the nesting depth of a form, that applying an automorphism to a denotation has the same effect as applying it to the molecule:

Lemma 6.2.5 (Lemma 37 in [40])**.** *If $\rho \in \mathbf{Aut}(I)$, then $\rho(\varphi \star \alpha) = \varphi \star \rho(\alpha)$.*

We skip the proof because it is not difficult. It only uses the fact that automorphisms preserve types. Next, we come to the representation result that we are mainly interested in:

Lemma 6.2.6 (Lemma 38 in [40])**.** *Assuming that I is $\mathcal{C}^{mk}$-homogeneous (it suffices for tuples of length $\leq 2k$), it holds: $x \in \overline{I}_k$ if and only if $x = \varphi \star \alpha$ for some I-molecule α and some form φ.*

Proof. Let $x = \varphi \star \alpha$. We want to show that x is transitively k-supported. Let $\rho \in \mathbf{Aut}(I)$ be an automorphism that fixes α. By Lemma 6.2.5, $\rho(\varphi \star \alpha) = \varphi \star \rho(\alpha)$. Therefore, α is a support of $\varphi \star \alpha$. Every $y \in \text{tc}(x)$ is also of the form $y = \psi \star \beta$, so it is also k-supported.

Now let $x \in \overline{I}_k$. We show via induction on the structure of x that it can be written as the denotation of a form and a molecule. If $x = a$ for some atom a, then $x = c_1 \star \alpha$ for any molecule with $\alpha_1 = a$. If $x = \emptyset$, then $x = \emptyset \star \alpha$ for any α. Suppose that $x \in \overline{I}_k$ is a non-empty set. There must be some molecule α supporting x. By the inductive hypothesis, $x = \{\varphi_y \star \alpha_y \mid y \in x\}$. We show that $x = \varphi \star \alpha$, where $\varphi = \{(\varphi_y, \text{tp}_I(\alpha, \alpha_y)) \mid y \in x\}$.

$$\begin{aligned}\varphi \star \alpha &= \{\psi \star \beta \mid (\psi, \text{tp}_I(\alpha, \beta)) \in \varphi\} \\ &= \{\varphi_y \star \beta \mid y \in x \text{ and } \text{tp}_I(\alpha, \beta) = \text{tp}_I(\alpha, \alpha_y)\}.\end{aligned}$$

Then $x \subseteq \varphi \star \alpha$ because for every $y \in x$, $\varphi_y \star \alpha_y$ is a member of the above set. For the converse $\varphi \star \alpha \subseteq x$, we need the homogeneity of I. Suppose that $z \in \varphi \star \alpha$. We must have $z = \psi \star \beta$ with $(\psi, \text{tp}_I(\alpha, \beta)) \in \varphi$. Further, there must be $y \in x$ with $\psi = \varphi_y$ and $\text{tp}_I(\alpha, \beta) = \text{tp}_I(\alpha, \alpha_y)$. Now it follows from the $\mathcal{C}^{mk}$-homogeneity that there exists $\rho \in \mathbf{Aut}(I)$ such that $\rho(\alpha\beta) = \alpha\alpha_y$. Since ρ fixes α pointwise, it also fixes x, so $z \in x$. □

The last step of the above proof is the only place where homogeneity is ever needed in the proof of Theorem 6.2.3. Intuitively, its role is to ensure that we can write any $x \in \overline{I}_k$ as the denotation of a form built from the set of types $\text{tp}_I(\alpha, \alpha_y)$, for all $y \in x$. The point is that this set of types, in combination with the molecule α, might in general yield a superset of x. Roughly speaking, this can happen when the type-partition in I is too coarse, i.e. coarser than the orbit-partition. This is ruled out by requiring homogeneity. But should Theorem 6.2.3 ever be needed for non-homogeneous structures, here is a thought on how to get by without the homogeneity condition: If the type-partition of I is coarser than its orbit-partition, then, presumably, the sets that CPT can define are also more symmetric than the ones which are definable on homogeneous structures. This is because CPT is not only isomorphism-invariant, but also has to satisfy certain symmetries with respect to types. Thus, it could be that on general structures, CPT cannot define all objects in $\overline{I}_k$, but only a subset of them, which are sufficiently "type-symmetric". For these specific h.f. sets in $\overline{I}_k$, it is conceivable that Lemma 6.2.6 goes through even without the homogeneity condition. This would be sufficient for the purposes of CPT lower bounds. In fact, we only need Theorem 6.2.3 for the set of all transitively k-supported *CPT-definable* objects, not for the whole sets $\overline{I}_k$ and $\overline{J}_k$.
We will make use of Theorem 6.2.3 in Chapter 9, where we consider unordered CFI graphs over hypercubes. We show that these structures are sufficiently homogeneous, so the need to strengthen the theorem and drop the homogeneity condition does not arise in this thesis.

Playing pebble games on supports of hereditarily finite sets
For the *proof of Theorem 6.2.3*, we have to lift the winning strategy for Duplicator in the bijective mk-pebble game on I and J to a winning strategy in the m-pebble game on $\overline{I}_k$ and $\overline{J}_k$. This is done in [40] as follows: Let $\overline{x}$ and $\overline{y}$ denote the pebbled elements of $\overline{I}_k$ and $\overline{J}_k$, respectively. Duplicator ensures that, after every move, there are forms $\varphi_1, ..., \varphi_m$, I-molecules $\alpha_1, ..., \alpha_m$ and J-molecules $\beta_1, ..., \beta_m$ such that, for every $i \in \{1, ..., m\}$, $x_i = \varphi_i \star \alpha_i$ and $y_i = \varphi_i \star \beta_i$, and such that $\text{tp}_I(\alpha_1 ... \alpha_m) = \text{tp}_J(\beta_1 ... \beta_m)$. Here, tp refers to the $\mathcal{C}^{mk}$-type of the tuples. First, we show that Duplicator can indeed maintain this invariant, and then, that she wins this way.
Suppose the play is in a position where the condition holds, and w.l.o.g. Spoiler picks up pebbles x_1 and y_1 (or none). We define Duplicator's bijection $f : \overline{I}_k \longrightarrow \overline{J}_k$. Since $I \equiv^{\mathcal{C}^{mk}} J$, there is a bijection $g : I^k \longrightarrow J^k$ such that for all $\alpha \in I^k$, $\text{tp}_I(\alpha\alpha_2 ... \alpha_m) = \text{tp}_J(g(\alpha)\beta_2 ... \beta_m)$. To see this, suppose that such a bijection did not exist. Then there is some $\alpha \in I^k$ such that

$$\begin{aligned} &|\{\alpha' \in I^k \mid \text{tp}_I(\alpha'\alpha_2 ... \alpha_m) = \text{tp}_I(\alpha\alpha_2 ... \alpha_m)\}| \\ \neq &|\{\beta \in J^k \mid \text{tp}_J(\beta\beta_2 ... \beta_m) = \text{tp}_I(\alpha\alpha_2 ... \alpha_m)\}|. \end{aligned}$$

But then, $\text{tp}_I(\alpha_2 ... \alpha_m) \neq \text{tp}_J(\beta_2 ... \beta_m)$ because the sizes of the above sets are definable in $\mathcal{C}^{mk}$ from the parameters $\alpha_2 ... \alpha_m$ and $\beta_2 ... \beta_m$, respectively (since on finite structures, $\mathcal{C}^{mk}$-types are determined by a single formula). Since by the invariant that Duplicator has maintained so far, $\text{tp}_I(\alpha_2 ... \alpha_m) = \text{tp}_J(\beta_2 ... \beta_m)$, we know that a type-preserving

bijection g as claimed must exist.
By Lemma 6.2.6, every $x \in \overline{I}_k$ can be written as $\varphi_x \star \alpha_x$ for an appropriate form and a molecule. We set $f(\varphi_x \star \alpha_x) := \varphi_x \star g(\alpha_x)$. It can be verified that f is indeed a bijection between $\overline{I}_k$ and $\overline{J}_k$. Essentially this follows from Lemma 39 in [40]. This lemma says that for any two denotations $\varphi \star \alpha, \psi \star \beta$, it only depends on φ, ψ and $\text{tp}(\alpha, \beta)$ whether $\varphi \star \alpha = \psi \star \beta$ and whether $\varphi \star \alpha \in \psi \star \beta$. In particular, this is independent of the structure from which the molecules come. Thus, since g preserves $\text{tp}(\alpha\alpha')$, we have $\varphi \star \alpha = \varphi' \star \alpha'$ if and only if $\varphi \star g(\alpha) = \varphi' \star g(\alpha')$. This shows that f is injective. Using a similar argument, one can see that every $\psi \star \beta \in \overline{J}_k$ has an f-preimage $\psi \star g^{-1}(\beta) \in \overline{I}_k$. Hence, f is a bijection.
Now when Spoiler places pebbles on elements $x \in \overline{I}_k$ and $f(x) = y \in \overline{J}_k$, then Duplicator sets $\varphi_1 := \varphi_x$ and $\alpha_1 := \alpha_x, \beta_1 := g(\alpha_x)$. So the invariant is maintained. It remains to show that the pebbled objects in $\overline{I}_k$ and $\overline{J}_k$ induce a local isomorphism. The relations of the structures I and J are preserved because $\text{tp}_I(\alpha_1 ... \alpha_m) = \text{tp}_J(\beta_1 ... \beta_m)$. The fact that the nesting structure (and equality) of the sets is preserved follows again from Lemma 39 in [40]. So we have $x_i = x_j$ iff $y_i = y_j$ and $x_i \in x_j$ iff $y_i \in y_j$, because these relations only depend on the respective forms φ_i, φ_j and on the types $\text{tp}(\alpha_i\alpha_j)$ and $\text{tp}(\beta_i\beta_j) = \text{tp}(g(\alpha_i)g(\alpha_j)) = \text{tp}(\alpha_i\alpha_j)$. This concludes the proof of Theorem 6.2.3.

Support lower bounds for the CFI query

Finally, we can show the desired lower bound on the support size of h.f. sets that need to be activated in order to distinguish non-isomorphic CFI-structures. The next theorem is implicit in the proof of Theorem 40 in [40].

Theorem 6.2.7. *Let $(G_n)_{n \in \mathbb{N}}$ be a family of base graphs and let $\boldsymbol{tw}_n$ denote the treewidth of G_n. Let $\mathfrak{G}_n^0, \mathfrak{G}_n^1$ denote the even and odd CFI-structures over G_n. Let $f(n) \leq \boldsymbol{tw}_n$ be a function such that $\mathfrak{G}_n^0$ and $\mathfrak{G}_n^1$ are $\mathcal{C}^{\boldsymbol{tw}_n}$-homogeneous for all tuples of length $\leq 2f(n)$. Then any* CPT*-program that distinguishes $\mathfrak{G}_n^0$ and $\mathfrak{G}_n^1$ for all $n \in \mathbb{N}$ must activate on input $\mathfrak{G}_n^i$ a h.f. set x whose smallest support has size at least $\Omega(f(n))$.*

Proof. Let Π be a CPT-program that distinguishes all mentioned CFI-graphs. Suppose for a contradiction that there is a function $f(n) \in o(k_n)$ such that Π activates on input $I := \mathfrak{G}_n^0$ only sets whose support is bounded by $f(n)$. These are objects in $\overline{I}_{f(n)}$. Let $I := \mathfrak{G}_n^1$. As is well-known (Theorem 5.1.3), it holds $I \equiv_{\mathcal{C}^{\mathbf{tw}_n}} J$. Theorem 6.2.3 now entails: $\overline{I}_{f(n)} \equiv_{\mathcal{C}^{\mathbf{tw}_n/f(n)}} \overline{J}_{f(n)}$. Because $f(n) \in o(\mathbf{tw}_n)$, $\mathbf{tw}_n/f(n)$ cannot be upper-bounded by a constant. Therefore, there is no fixed FPC-sentence φ that distinguishes $\overline{I}_{f(n)}$ and $\overline{J}_{f(n)}$ (for all n). But this contradicts Corollary 6.2.2, which says that there is an FPC-sentence φ equivalent to Π on input structures $\overline{I}_{f(n)}$ and $\overline{J}_{f(n)}$. $\square$

Rank lower bound

From Theorem 6.2.7, one can infer that not only the support but also the *rank*, i.e. the nesting depth of one of the activated h.f. sets must be sufficiently large for any CPT-program defining the CFI query. This follows from Lemma 29 in [40], which basically says that the support of a h.f. set x is at most by a factor $\log |\mathbf{Orbit}(x)|$ larger than the supports of its elements. Since the sets that CPT activates have polynomial orbit size in the input structure, the support growth rate per nesting depth is at most logarithmic in the size of the input. One can conclude that constant-rank sets can only have support at most $\log^c(|\mathfrak{G}^S|)$, for a constant c; so together with Theorem 6.2.7 – using base graphs with treewidth $> \log^c(|\mathfrak{G}^S|)$ – we can infer:

Theorem 6.2.8 (Theorem 40 in [40])**.** *The parity query for pre-ordered CFI graphs is not defined by any* CPT*-program that activates sets of rank at most* $o(\frac{\log n}{\log\log n})$.

At first sight, it might seem strange that there does exist a CPT-algorithm that decides the CFI-query on base graphs with linear degree using only sets of *constant* rank (see Theorem 6.1.3). However, this is no contradiction: The CFI-structures in the linear-degree setting have exponential size in the size of the base graph $|G|$. Therefore, also the orbit size of the activated sets may be exponential in $|G|$, and thus, the support growth per nesting depth can be *polynomial* in $|G|$, instead of $\log |G|$. So, it is then possible for a constant-rank set to have a sufficiently large support to decide the CFI-query.

7 The non-definability of preorders with small colour classes in hypercubes

This chapter is based on [87], which was published at CSL 2021. In it, we explore the limitations of the choiceless algorithms for the ordered versions of the CFI-query that we introduced in the previous chapter. Concretely, we prove that there are base graphs for the CFI-construction in which neither a linear order nor a preorder with colour classes of logarithmic size is CPT-definable. Thus, the unordered CFI-query over these base graphs cannot be tackled in CPT with any of the known algorithms. This family of base graphs are the n-dimensional hypercubes. Actually, the non-definability of the required preorders holds also in complete graphs, and is even easier to prove there. However, Theorem 6.1.3 from the last chapter shows that the CFI-query over complete graphs is CPT-definable anyway because of the implicit padding in high-degree CFI-graphs. Thus, the non-definability of preorders in such graphs is not an obstacle for the definability of the CFI-query. However, on CFI-structures over hypercubes, the status of the CFI-query is open and the failure of the preorder-based algorithm on unordered hypercube CFI-graphs can be taken as an indication that these instances are hard for CPT. In particular, hypercubes are a good compromise between the high symmetry of complete graphs and avoiding high degree vertices. All vertices in the n-dimensional hypercube have logarithmic degree, which means that the CFI-construction does not lead to a super-polynomial size blow-up.

In order to prove the non-definability of the said preorders, we show that their orbit-size with respect to the automorphism group of the n-dimensional hypercube grows super-polynomially in the size of the hypercube. Then it follows with Corollary 3.0.6 that no hereditarily finite set that encodes such a preorder is CPT-definable.

As in Section 5.3, we let $(\mathcal{H}_n)_{n\in\mathbb{N}}$ be the family of n-dimensional hypercubes. The vertex-set of $\mathcal{H}_n$ is $\{0,1\}^n$, so n is the logarithm of the size of the hypercube. As explained in Section 5.3, the group $\mathbf{Sym}_n$ acts on $\{0,1\}^n$ by permuting the positions of the binary words. This constitutes a subgroup of $\mathbf{Aut}(\mathcal{H}_n)$.

We view a preorder of $\mathcal{H}_n$ as an ordered partition $(C_1, ..., C_m)$ of $\{0,1\}^n$. Its parts are called *colour classes*. A set $x \in \mathrm{HF}(\{0,1\}^n)$ *encodes* such an ordered partition if there exists a BGS-term t with a polynomial bound which computes the following natural representation of $(C_1, ..., C_m)$, given x: $[\![t(x)]\!]^{\mathcal{H}_n} = \{C_1, \{C_2, \{C_3, \{...\}\}\}\}$. If the ordered partition into colour classes has a super-polynomially large orbit, then the same is true for any encoding as a h.f. set. Therefore, we do not need to specify a concrete one.

Lemma 7.0.1. *For each $n \in \mathbb{N}$, let $\mathcal{P}_n = (C_1, ..., C_{m_n})$ be an ordered partition of $\{0,1\}^n$, and $x_n \in \mathrm{HF}(\{0,1\}^n)$ be a set that encodes $\mathcal{P}_n$. If the orbit of $\mathcal{P}_n$ with respect to the action of $\mathbf{Sym}_n$ on $\{0,1\}^n$ grows super-polynomially in $2^n = |\{0, 1^n\}|$, then the orbit of x_n with respect to the action of $\mathbf{Aut}(\mathcal{H}_n)$ grows super-polynomially in 2^n.*

Proof. Assume for a contradiction that the $\mathbf{Aut}(\mathcal{H}_n)$-orbit of x_n has at most polynomial size in 2^n. Since x_n encodes $\mathcal{P}_n$, the corresponding preorder $\prec$, whose colour classes are the parts of $\mathcal{P}_n$, is CPT-definable from x_n in $\mathcal{H}_n$. Let t be the BGS-term such that $[\![t(x_n)]\!]^{\mathcal{H}_n} = \{C_1, \{C_2, \{C_3, \{...\}\}\}\}$. Since t has just a single free variable and CPT is isomorphism-invariant, the output of t is invariant under all automorphisms that stabilise its input x_n. Thus, $|\mathbf{Orbit}_{\mathbf{Aut}(\mathcal{H}_n)}([\![t(x_n)]\!]^{\mathcal{H}_n})| \leq |\mathbf{Orbit}_{\mathbf{Aut}(\mathcal{H}_n)}(x_n)|$. Hence, the $\mathbf{Aut}(\mathcal{H}_n)$-orbit-size of $\mathcal{P}_n$ is only polynomial in $2^n = |\mathcal{H}_n|$. For the last step, we just have to observe that $\mathbf{Sym}_n$ with its action on $\{0,1\}^n$ is a subgroup of $\mathbf{Aut}(\mathcal{H}_n)$ (see Section 5.3). Therefore, $|\mathbf{Orbit}_{\mathbf{Sym}_n}(\mathcal{P}_n)| \leq |\mathbf{Orbit}_{\mathbf{Aut}(\mathcal{H}_n)}(\mathcal{P}_n)|$, so this orbit-size is also only polynomial in 2^n. This is a contradiction to the assumption of the lemma, which says that $|\mathbf{Orbit}_{\mathbf{Sym}_n}(\mathcal{P}_n)|$ is super-polynomial in 2^n. □

The main result of this chapter reads as follows:

Theorem 7.0.2. *Let $(\mathcal{H}_n)_{n \in \mathbb{N}}$ be the family of n-dimensional hypercubes. For each $n \in \mathbb{N}$, fix an ordered partition $\mathcal{P}_n = (C_1, ..., C_{m_n})$ of $\{0,1\}^n$ such that each part C_i has size at most $\mathcal{O}(n)$.*
Then there exists no CPT*-sentence Π such that for all $n \in \mathbb{N}$, Π activates on input $\mathcal{H}_n$ a h.f. set that encodes $\mathcal{P}_n$.*

Since the algorithm for the preordered CFI-query presented in Section 6.1.2 constructs a h.f. set whose nesting structure implicitly defines a preorder with log-sized colour classes on the base graph, we can immediately conclude:

Corollary 7.0.3. *The CPT-algorithm for preordered CFI-graphs from Theorem 6.1.2 cannot decide the CFI-query over the family of unordered n-dimensional hypercubes, not even with any CPT-definable preprocessing.*

Theorem 7.0.2 follows directly from the technical result below, together with Lemma 7.0.1 and Corollary 3.0.6 (since $|\mathcal{H}_n| = 2^n$).

Theorem 7.0.4. *Let $(\mathcal{H}_n)_{n \in \mathbb{N}}$ be the family of n-dimensional hypercubes. For each $n \in \mathbb{N}$, fix an ordered partition $\mathcal{P}_n = (C_1, ..., C_{m_n})$ of $\{0,1\}^n$ such that each colour class C_i has size at most $\mathcal{O}(n)$.*
Then the orbit-size of $\mathcal{P}_n$ with respect to the action of $\mathbf{Sym}_n$ on $\{0,1\}^n$ grows super-polynomially in 2^n.

For the proof of Theorem 7.0.4, we employ the Orbit-Stabiliser Theorem (Theorem 4.1.3), which tells us that instead of directly proving a lower bound for the orbit size, we can alternatively upper-bound the size of the corresponding stabiliser. For the remainder of this chapter, fix an ordered partition $\mathcal{P}_n = (C_1, ..., C_{m_n})$ of $\{0,1\}^n$ for each $n \in \mathbb{N}$, as in Theorem 7.0.4. Even though $\mathcal{P}_n$ is technically a tuple, we will write $C_i \in \mathcal{P}_n$ for its parts. Our aim is to show a good upper bound on the size of

$$\mathbf{Stab}(\mathcal{P}_n) := \{\pi \in \mathbf{Sym}_n \mid \pi(C_i) = C_i \text{ for each } C_i \in \mathcal{P}_n\} = \bigcap_{i \in [m_n]} \mathbf{Stab}(C_i).$$

Then Theorem 7.0.4 will follow with the Orbit-Stabiliser Theorem.

7.1 Using supporting partitions to estimate the stabiliser sizes

In order to estimate the sizes of the groups $\mathbf{Stab}(\mathcal{P}_n)$ and $\mathbf{Stab}(C_i)$, we look at their *coarsest supporting partitions* (see Definition 4.2.3 in Section 4.2). We write $\mathbf{SP}(\mathcal{P}_n) = \mathbf{SP}(\mathbf{Stab}(\mathcal{P}_n))$ and $\mathbf{SP}(C_i) = \mathbf{SP}(\mathbf{Stab}(C_i))$ to denote the respective coarsest supporting partition. Recall that this is the coarsest partition of $[n]$ such that every $\pi \in \mathbf{Sym}_n$ that fixes every part of $\mathbf{SP}(C_i)$ setwise also fixes the set $C_i \subseteq \{0,1\}^n$ setwise (and likewise for $\mathbf{SP}(\mathcal{P}_n))$). It is also important to recall Lemma 4.2.5, which relates the pointwise and setwise stabilisers of the respective supporting partitions to the stabiliser of the object that is being supported:

$$\mathbf{Stab}^\bullet(\mathbf{SP}(C_i)) \leq \mathbf{Stab}(C_i) \leq \mathbf{Stab}(\mathbf{SP}(C_i)).$$

So in particular, every $\pi \in \mathbf{Stab}(C_i)$ stabilises the partition $\mathbf{SP}(C_i)$ setwise and induces a permutation on its parts. The same holds of course for $\mathbf{Stab}(\mathcal{P}_n)$ and $\mathbf{SP}(\mathcal{P}_n)$.

Example 7.1.1. *Let $n = 6$, and consider the length-n binary string $a := 111000$. Then, $\mathbf{SP}(a) = \mathbf{SP}(\mathbf{Stab}(a)) = \{\{1,2,3\},\{4,5,6\}\}$. As one can see, for single bitstrings a, the stabiliser of a coincides with the* pointwise *stabiliser of its coarsest supporting partition. In contrast, consider now $A = \{111000, 000111\}$.*
Again, it holds $\mathbf{SP}(A) = \{\{1,2,3\},\{4,5,6\}\}$, but now $\mathbf{Stab}(A) = \mathbf{Stab}(\mathbf{SP}(A))$, i.e. the stabiliser of the set A is the setwise *stabiliser of its support. These two examples illustrate the two "extremes" of Lemma 4.2.5. In general, the stabiliser of an object may be located strictly between the pointwise and setwise stabiliser of its coarsest supporting partition and can then only be approximated.*

Some notation regarding partitions to keep in mind: For a partition $\mathcal{P}$ of $[n]$, $|\mathcal{P}|$ denotes its *size*, that is, the number of parts it has. Moreover, for $k \in [n]$, we write $\mathcal{P}(k)$ for the part in $\mathcal{P}$ that contains k. Additionally, we will frequently consider the "intersection" of several supporting partitions. By this, we do not mean the intersection of the sets of parts but rather the coarsest partition that refines the two partitions we are intersecting. We use the symbol $\sqcap$ for this type of partition-intersection.

Definition 7.1.2. *Let $\mathcal{P}, \mathcal{P}'$ be partitions of $[n]$. The* intersection *$\mathcal{P} \sqcap \mathcal{P}'$ is defined like this:*

$$\mathcal{P} \sqcap \mathcal{P}' := \{\mathcal{P}(k) \cap \mathcal{P}'(k) \mid k \in [n]\}.$$

7.2 Proof of the Superpolynomial-Orbit Theorem

In this section, we prove Theorem 7.0.4 by upper-bounding the size of $\mathbf{Stab}(\mathcal{P}_n)$. Our analysis of $|\mathbf{Stab}(\mathcal{P}_n)|$ is divided into two main cases that we treat separately. The distinction is with respect to the maximum size of the coarsest supporting partition of any of the colour classes C_i in $\mathcal{P}_n$, viewed as a function of n: Let $B_n \subseteq \{0,1\}^n$ be the part of $\mathcal{P}_n$ such that $|\mathbf{SP}(B_n)|$ (i.e. its number of parts) is maximal among $\{|\mathbf{SP}(C_i)| \mid C_i \in \mathcal{P}_n\}$. Then the two cases we distinguish are:

(1) The maximal support size of a colour class grows sublinearly: $|\mathbf{SP}(B_n)| \in o(n)$.

(2) The maximal support size of a colour class grows linearly: $|\mathbf{SP}(B_n)| \in \Theta(n)$.

In Case (1), we bound the number of permutations in $\mathbf{Sym}_n$ that setwise stabilize the support of each colour class simultaneously. In other words, we bound $|\bigcap_{C_i \in \mathcal{P}_n} \mathbf{Stab}(C_i)|$. This is equal to $|\mathbf{Stab}(\mathcal{P}_n)|$. Unfortunately, our analysis only leads to a meaningful bound if the maximum support size grows strictly sublinearly in n, which is why we cannot avoid the case distinction. To see the problem, simply assume that the coarsest supporting partition of every $C_i \in \mathcal{P}_n$ consists only of singleton parts (and thus has linear size). Then the whole group $\mathbf{Sym}_n$ stabilises each of those supporting partitions setwise, so we cannot infer any bound on $|\mathbf{Stab}(\mathcal{P}_n)|$.

Therefore, in Case (2), we use some more involved arguments to deduce details about the structure of the set B_n based on the fact that it has a very fine supporting partition. We are able to show that $\mathbf{Stab}(B_n)$ is relatively small, and as a consequence, the orbit of the colour class B_n (and thereby of the whole preorder given by $\mathcal{P}_n$) is large enough to prove the statement of the theorem. The result for this case can also be phrased as: No subset of $\{0,1\}^n$ of size $\mathcal{O}(n)$ and with linearly many parts in its supporting partition has a polynomially-sized orbit in $|\{0,1\}^n| = 2^n$.

We deal with the two cases in the next two subsections. Their results are summarized in Lemma 7.2.1 and Lemma 7.2.7. Together they imply the theorem.

7.2.1 The case of sublinearly bounded supports

The result of this subsection is:

Lemma 7.2.1. *Assume the following three conditions hold:*

1. *Every $v \in \{0,1\}^n$ occurs in at least one of the colour classes C_i in $\mathcal{P}_n$.*
2. *The function $\max_{i \in [m_n]} |C_i|$ is in $\mathcal{O}(n)$.*
3. *$|\mathbf{SP}(B_n)| \in o(n)$.*

Then the orbit size of $\mathcal{P}_n$ w.r.t. the action of $\mathbf{Sym}_n$ on $\{0,1\}^n$ grows faster than any polynomial in 2^n.

It may seem somewhat superfluous to state the first condition explicitly, since we are assuming that $\mathcal{P}_n$ is an ordered partition of $\{0,1\}^n$, so this is always satisfied. However, we would like to stress that our result crucially depends on this fact and thus, the proof will not directly go through if we replace the preorder by, say, a partial one that only partitions a small subset of $\{0,1\}^n$ into colour classes.
The second condition says that the colour classes of the preorder have logarithmic size (compared to the size of the hypercube) and is also required in Theorem 7.0.4. The third condition is specific to the case that we treat in this subsection.

We prove the lemma on the next few pages. By Lemma 4.2.5, we have for the stabiliser of $\mathcal{P}_n$:

$$\mathbf{Stab}(\mathcal{P}_n) \leq \bigcap_{C_i \in \mathcal{P}_n} \mathbf{Stab}(\mathbf{SP}(C_i)).$$

Let us briefly outline how we will use this fact to bound $|\mathbf{Stab}(\mathcal{P}_n)|$. For each colour class $C_i \in \mathcal{P}_n$, $\mathbf{Sym}(\mathbf{SP}(C_i))$ denotes the symmetric group on the parts of the supporting partition $\mathbf{SP}(C_i)$ (in constrast, $\mathbf{Sym}_n$ is the symmetric group on the set $[n]$ that underlies this partition). Every $\pi \in \mathbf{Sym}_n$ that stabilises $\mathbf{SP}(C_i)$ as a set *induces* (or *realises*) a $\sigma \in \mathbf{Sym}(\mathbf{SP}(C_i))$ in the sense that $\sigma(P) = \{\pi(k) \mid k \in P\}$ for all $P \in \mathbf{SP}_i(x_n)$. This can also be extended to a set $J_n \subseteq \mathcal{P}_n$ of several colour classes: Every $\pi \in \bigcap_{C_i \in J_n} \mathbf{Stab}(\mathbf{SP}(C_i))$ induces a $\overline{\sigma} \in \bigtimes_{C_i \in J_n} \mathbf{Sym}(\mathbf{SP}(C_i))$. Here, $\overline{\sigma}$ is the tuple of permutations that π realises simultaneously on the parts of the respective supporting partitions.

Now in order to bound $|\mathbf{Stab}(\mathcal{P}_n)|$, we will choose a subset of colour classes $J_n \subseteq \mathcal{P}_n$ with certain properties that will enable us to bound two quantities: Firstly, there will be an upper bound on the number of $\overline{\sigma} \in \bigtimes_{C_i \in J_n} \mathbf{Sym}(\mathbf{SP}(C_i))$ that can be realised by any $\pi \in \mathbf{Stab}(\mathcal{P}_n)$. Secondly, we observe that each such $\overline{\sigma}$ can only be realised by a small number of distinct $\pi \in \mathbf{Stab}(\mathcal{P}_n)$. The product of these two bounds is then an upper bound for $|\mathbf{Stab}(\mathcal{P}_n)|$.

We start the proof of Lemma 7.2.1 with the second part of the above proof sketch, namely we show how we can generally bound the number of realisations of a given $\overline{\sigma} \in \bigtimes_{i \in [m]} \mathbf{Sym}(\mathbf{SP}(A_i))$, where $A_1, ..., A_m$ are sets of length-n bit-strings (in our case, the colour classes of $\mathcal{P}_n$). We refer back to Definition 7.1.2 for the definition of $\sqcap$, the intersection of partitions. The following lemma shows that any $\pi \in \mathbf{Sym}_n$ that realises the mentioned tuple of permutations $\overline{\sigma}$ of the parts of the supports has to permute the parts of the intersection partition $\bigsqcap_{i=1}^m \mathbf{SP}(A_i)$ in the same way.

Lemma 7.2.2. *Let $A_1, ..., A_m \subseteq \{0,1\}^n$ be a collection of sets of bit-strings. Fix any simultaneous permutation $\overline{\sigma}$ of the parts of the supports of the sets, i.e. $\overline{\sigma} \in \bigtimes_{i=1}^m \mathbf{Sym}(\mathbf{SP}(A_i))$.*

There exists a $\vartheta_{\overline{\sigma}} \in \mathbf{Sym}(\bigsqcap_{i=1}^m \mathbf{SP}(A_i))$ such that every $\pi \in \mathbf{Sym}_n$ that realises $\overline{\sigma}$ also realises $\vartheta_{\overline{\sigma}}$.

Proof. We show the statement via induction on m. For $m = 1$, there is only one set A_1, so $\overline{\sigma} \in \mathbf{Sym}(\mathbf{SP}(A_1))$. Hence, $\vartheta_{\overline{\sigma}} := \overline{\sigma}$ is the desired permutation.
For the inductive step, assume the statement holds for a fixed m. Consider now a collection $A_1, ..., A_{m+1} \subseteq \{0,1\}^n$, and $\overline{\sigma} \in \bigtimes_{i=1}^{m+1} \mathbf{Sym}(\mathbf{SP}(A_i))$. Let $\overline{\sigma}'$ be the m-tuple that is the restriction of $\overline{\sigma}$ to its first m components. By the induction hypothesis there exists a $\vartheta_{\overline{\sigma}'} \in \mathbf{Sym}(\bigsqcap_{i=1}^{m} \mathbf{SP}(A_i))$ that is induced by every $\pi \in \mathbf{Sym}_n$ realising $\overline{\sigma}'$, and therefore also by every π realising $\overline{\sigma}$. Furthermore, every $\pi \in \mathbf{Sym}_n$ that realises $\overline{\sigma}$ has to realise $\overline{\sigma}_{m+1} \in \mathbf{Sym}(\mathbf{SP}(A_{m+1}))$. So we know the following constraints for π: For every $P \in \mathbf{SP}(A_{m+1})$, $\pi(P) = \overline{\sigma}_{m+1}(P)$, and also, for every $P \in \bigsqcap_{i=1}^{m} \mathbf{SP}(A_i)$, $\pi(P) = \vartheta_{\overline{\sigma}'}(P)$. Hence, the desired $\vartheta_{\overline{\sigma}} \in \mathbf{Sym}(\bigsqcap_{i=1}^{m+1} \mathbf{SP}(A_i))$ is defined as follows: Let $P \in \bigsqcap_{i=1}^{m+1} \mathbf{SP}(A_i)$, and call $Q_P \in \bigsqcap_{i=1}^{m} \mathbf{SP}(A_i)$ the part such that $P \subseteq Q_P$, and let $Q'_P \in \mathbf{SP}(A_{m+1})$ be such that $P \subseteq Q'_P$ (the parts Q_P, Q'_P must exist because the partition $\bigsqcap_{i=1}^{m+1} \mathbf{SP}(A_i)$ refines both $\bigsqcap_{i=1}^{m} \mathbf{SP}(A_i)$ and $\mathbf{SP}(A_{m+1})$). Then:

$$\vartheta_{\overline{\sigma}}(P) := \vartheta_{\overline{\sigma}'}(Q_P) \cap \overline{\sigma}_{m+1}(Q'_P).$$

One can easily check that indeed, $\vartheta_{\overline{\sigma}} \in \mathbf{Sym}(\bigsqcap_{i=1}^{m+1} \mathbf{SP}(A_i))$, and, as we argued already, that every $\pi \in \mathbf{Sym}_n$ that realises $\overline{\sigma}$ must realise $\vartheta_{\overline{\sigma}}$ as well. Below is a visualisation for the first inductive step, that adds A_2 to A_1. □

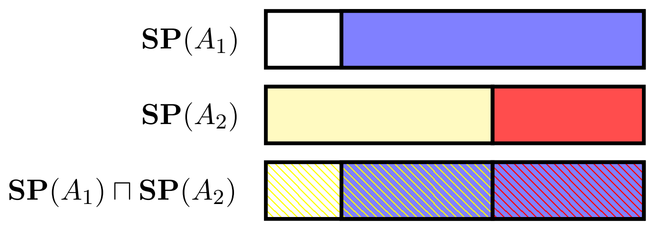


Figure 7.1: Suppose $\overline{\sigma}$ is the identity permutation in both $\mathbf{SP}(A_1)$ and $\mathbf{SP}(A_2)$. Then any $\pi \in \mathbf{Sym}_n$ realising $\overline{\sigma}$ must stabilise each of the three parts of $\mathbf{SP}(A_1) \sqcap \mathbf{SP}(A_2)$ setwise.

So, intuitively speaking, the finer the partition $\bigsqcap_{i=1}^{m} A_i$ is, the fewer realisations are there for any $\overline{\sigma} \in \bigtimes_{i=1}^{m} \mathbf{Sym}(\mathbf{SP}(A_i))$. Therefore, we will aim to select a subset of the colour classes in $\mathcal{P}_n$ such that the intersection over the supports is as fine as possible. More precisely, we would like it to consist of many singleton parts.
For the rest of this subsection we denote by $S_n \subseteq [n]$ the set of positions which are in singleton parts in $\bigsqcap_{C_i \in \mathcal{P}_n} \mathbf{SP}(C_i)$, i.e.

$$S_n := \{k \in [n] \mid \{k\} \in \bigsqcap_{C_i \in \mathcal{P}_n} \mathbf{SP}(C_i)\}.$$

It turns out that there can only be few positions which are *not* in singleton parts in $\bigsqcap_{C_i \in \mathcal{P}_n} \mathbf{SP}(C_i)$; this is a consequence of the assumption that every element of $\{0,1\}^n$ occurs in a colour class of $\mathcal{P}_n$, together with the size bound on the colour classes (the first and second condition of Lemma 7.2.1):

Lemma 7.2.3. *Let c be a constant such that for every colour class $C_i \in \mathcal{P}_n$, for large enough $n \in \mathbb{N}$, it holds: $|C_i| \leq c \cdot n$. Further, assume that every element of $\{0,1\}^n$ occurs in at least one $C_i \in \mathcal{P}_n$. Then, for all large enough n:*

$$|[n] \setminus S_n| < 8 \log n.$$

Proof. Assume for a contradiction: $|[n] \setminus S_n| \geq 8 \log n$. Let

$$\mathbf{P_2} := \{P \in \prod_{C_i \in \mathcal{P}_n} \mathbf{SP}(C_i) \mid |P| \geq 2\}.$$

Choose a string $a \in \{0,1\}^n$ such that its substring at the positions in P, denoted $a[P]$, contains an equal number of zeros and ones (or almost equal if $|P|$ is odd) for each $P \in \mathbf{P_2}$. Let A_n denote the colour class $C_i \in \mathcal{P}_n$ such that $a \in C_i$. This exists by the assumptions of the lemma. For each $P \in \mathbf{P_2}$, a superset of P (or P itself) must occur as a part of $\mathbf{SP}(A_n)$. We conclude that $\mathbf{Stab}(A_n)$ must contain all permutations $\pi \in \mathbf{Sym}_n$ which are the identity on S_n, and arbitrarily permute the elements within each part $P \in \mathbf{P_2}$ (due to Lemma 4.2.5). So let

$$\Gamma_{\mathbf{P_2}} := \{\pi \in \mathbf{Sym}_n \mid \pi(s) = s \text{ for all } s \in S_n \text{ and } \pi(P) = P \text{ for all } P \in \mathbf{P_2}\} \leq \mathbf{Stab}(A_n)$$

be this subgroup of $\mathbf{Sym}_n$.
All images of a under the permutations in $\Gamma_{\mathbf{P_2}}$ must also be in A_n. This entails a violation of the size bound on $|A_n|$, as we show now. It is easy to see (observing that within every $P \in \mathbf{P_2}$, the $|P|/2$ many ones in $a[P]$ can be moved to an arbitrary subset of the positions P) that the orbit of a with respect to $\Gamma_{\mathbf{P_2}}$ has size at least

$$\begin{aligned}\mathbf{Orbit}_{\Gamma_{\mathbf{P_2}}}(a) &\geq \prod_{P \in \mathbf{P_2}} \binom{|P|}{|P|/2} \geq \prod_{P \in \mathbf{P_2}} \delta \cdot \frac{2^{|P|}}{\sqrt{|P|}} \\ &= 2^{|[n]\setminus S_n|} \cdot \prod_{P \in \mathbf{P_2}} \frac{\delta}{\sqrt{|P|}} \qquad (\star)\end{aligned}$$

Here, $0.6 \leq \delta < 1$. The inequality is quite well-known and can be computed with Stirling's approximation. The equality is clear because the parts in $\mathbf{P_2}$ cover exactly the positions $[n] \setminus S_n$.
As for the large product, we can check that its value becomes smallest possible if all parts $P \in \mathbf{P_2}$ are doubleton parts (plus potentially one part with three elements). To see this, take any part $P \in \mathbf{P_2}$ with $|P| \geq 4$ and split off two elements, such that $P = P_1 \dot\cup P_2$, $|P_1| = 2$. Then the contribution of P changes from $\frac{\delta}{\sqrt{|P|}}$ to $\frac{\delta^2}{\sqrt{2(|P|-2)}}$. The latter is strictly smaller because $\delta < 1$. Repeating this argument shows that we get the minimal value of $(\star)$ if we assume that all parts in $\mathbf{P_2}$ are doubletons. In that case, $(\star)$ becomes this:

$$\mathbf{Orbit}_{\Gamma_{\mathbf{P_2}}}(a) \geq \left(\frac{4\delta}{\sqrt{2}}\right)^{|[n]\setminus S_n|/2} \geq \sqrt{2}^{|[n]\setminus S_n|/2}.$$

Now plugging in our initial assumption $|[n] \setminus S_n| \geq 8 \log n$, this expression is $\geq n^2$. This is a contradiction to the fact that $|A_n| \leq cn$, for some constant c. □

Now that we know that the set of singleton positions S_n is almost $[n]$ itself, we proceed to construct our subset of the colour classes $J_n \subseteq \mathcal{P}_n$ such that the intersection of its supports already individualises all positions in S_n. Furthermore, we make sure that there are not too many ways how the supports of the colour classes in J_n can be permuted simultaneously; this will show that $\bigcap_{C_i \in \mathcal{P}_n} \mathbf{Stab}(\mathbf{SP}(C_i))$ cannot be too large, which is precisely our goal.

Lemma 7.2.4. *Let $f(n) \in o(n)$ such that (for large enough n) for all colour classes $P_i \in \mathcal{P}_n$, $|\mathbf{SP}(P_i)| \leq f(n)$.*
For large enough n, there exists a subset $J_n \subseteq \mathcal{P}_n$ of colour classes with the following two properties:

(1) Every position in S_n is in a singleton part of $\bigsqcap_{P_j \in J_n} \mathbf{SP}(P_j)$.

(2) The following bound for the number of realisable simultaneous permutations of the supporting partitions holds:

$$|\{\overline{\sigma} \in \bigtimes_{C_j \in J_n} \mathbf{Sym}(\mathbf{SP}(C_j)) \mid \textit{there is a } \pi \in \mathbf{Sym}_n \textit{ that realises } \overline{\sigma}\}|$$
$$\leq (f(n)!)^{\lceil n/(f(n)-1)\rceil} \cdot 2^n$$

Proof. We construct J_n stepwise, starting with $J_n^0 := \emptyset$ and adding one new colour class $C_{j_i} \in \mathcal{P}_n$ in each step $i \geq 1$ in such a way that

$$\Big| \bigsqcap_{C_j \in J_n^{i-1}} \mathbf{SP}(C_j) \sqcap \mathbf{SP}(C_{j_i}) \Big| > \Big| \bigsqcap_{C_j \in J_n^{i-1}} \mathbf{SP}(C_j) \Big|.$$

Let s be the number of construction steps needed, i.e. $J_n := J_n^s$ is such that property (1) of the lemma holds for this subset of $\mathcal{P}_n$. By definition of S_n, it is clear that such a subset exists because $\mathcal{P}_n$ itself satisfies property (1).
For each construction step i, we let

$$\Gamma_i := \{\overline{\sigma} \in \bigtimes_{C_j \in J_n^i} \mathbf{Sym}(\mathbf{SP}(C_j)) \mid \text{ there is a } \pi \in \mathbf{Sym}_n \text{ that realises } \overline{\sigma}\}.$$

Furthermore, for each step i we let k_i be the increase in the number of parts in the intersection that is achieved in this step:

$$k_i := |\bigsqcap_{C_j \in J_n^i} \mathbf{SP}(C_j)| - |\bigsqcap_{C_j \in J_n^{i-1}} \mathbf{SP}(C_j)|.$$

The main part of the proof consists in showing the following

Claim 7.2.1. *For each step i, the size of $|\Gamma_i|$ is bounded by*

$$|\Gamma_i| \leq \prod_{j=1}^{i} (\min\{(k_j + 1), f(n)\})!$$

Proof of claim. Via induction on i. For $i = 1$, we have $k_1 = |\mathbf{SP}_{j_1}(x_n)| \leq f(n)$, where j_1 is the level chosen in the first step of the construction of J_n. The group Γ_1 is a subgroup of $\mathbf{Sym}(\mathbf{SP}_{j_1}(x_n))$, whose size is bounded by $|\mathbf{SP}_{j_1}(x_n)|!$. Therefore, the claim holds.

For the inductive step, consider the step $i + 1$ of the construction. Let j_{i+1} be the colour class of the preorder that is added in this step. In order to bound the size of Γ_{i+1}, we consider for each $\overline{\sigma} \in \Gamma_i$ the following set:

$$\Gamma^{\overline{\sigma}}_{i+1} := \{\sigma \in \mathbf{Sym}(\mathbf{SP}(C_{j_{i+1}})) \mid \text{there is a } \pi \in \mathbf{Sym}_n \text{ that realises } \sigma \text{ and } \overline{\sigma}\}.$$

We need to show that for each $\overline{\sigma} \in \Gamma_i$, it holds $|\Gamma^{\overline{\sigma}}_{i+1}| \leq (\min\{(k_{i+1} + 1), f(n)\})!$.
Since $|\mathbf{SP}(C_{j_{i+1}})| \leq f(n)$, the bound $|\Gamma^{\overline{\sigma}}_{i+1}| \leq f(n)!$ is clear. It remains to show that for an arbitrary fixed $\overline{\sigma} \in \Gamma_i$, it holds $|\Gamma^{\overline{\sigma}}_{i+1}| \leq (k_{i+1} + 1)!$.

For a part $P \in \mathbf{SP}(C_{j_{i+1}})$, let

$$\mathbf{Q}(P) := \{Q \in \prod_{C_j \in J^i_n} \mathbf{SP}(C_j) \mid Q \cap P \neq \emptyset\}.$$

Then we define an equivalence relation $\sim \subseteq \mathbf{SP}(C_{j_{i+1}})^2$: For parts $P, P' \in \mathbf{SP}(C_{j_{i+1}})$, we let

$$P \sim P' \text{ iff } \mathbf{Q}(P) = \mathbf{Q}(P').$$

Now we show how the equivalence classes of $\sim$ can be used to approximate the images of the parts in $\mathbf{SP}(C_{j_{i+1}})$ under permutations in $\Gamma^{\overline{\sigma}}_{i+1}$: Every $\pi \in \mathbf{Sym}_n$ that realises any $\sigma \in \Gamma^{\overline{\sigma}}_{i+1}$ also realises $\overline{\sigma} \in \Gamma_i$. Hence, by Lemma 7.2.2, all such π induce the same $\vartheta_{\overline{\sigma}} \in \mathbf{Sym}(\prod_{C_j \in J^i_n} \mathbf{SP}(C_j))$. This means that for any $\sigma \in \Gamma^{\overline{\sigma}}_{i+1}$, and every part $P \in \mathbf{SP}(C_{j_{i+1}})$,

$$\sigma(P) \subseteq \bigcup_{Q \in \mathbf{Q}(P)} \vartheta_{\overline{\sigma}}(Q).$$

The situation is visualised in the figure below.

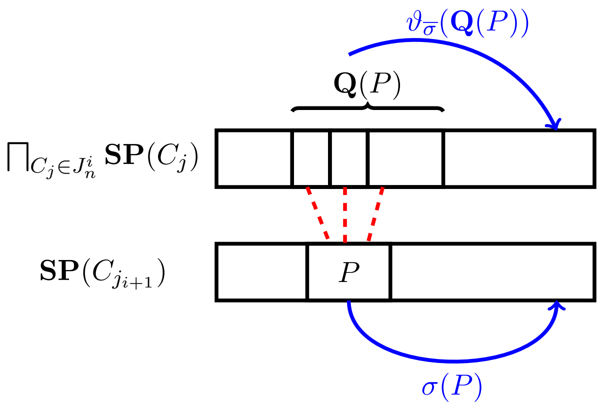

$\vartheta_{\overline{\sigma}}(\mathbf{Q}(P))$
$\mathbf{Q}(P)$
$\prod_{C_j \in J_n^i} \mathbf{SP}(C_j)$
$\mathbf{SP}(C_{j_{i+1}})$
P
$\sigma(P)$

Figure 7.2: A part $P \in \mathbf{SP}(C_{j_{i+1}})$ and the associated parts $\mathbf{Q}(P)$ in $\prod_{C_j \in J_n^i} \mathbf{SP}(C_j)$ that it intersects. Any $\sigma \in \mathbf{Sym}(\mathbf{SP}(C_{j_{i+1}}))$ compatible with $\overline{\sigma}$ must map P to the same position that $\mathbf{Q}(P)$ is mapped to by $\vartheta_{\overline{\sigma}}$.

Therefore, we can fix any $\widehat{\sigma} \in \Gamma_{i+1}^{\overline{\sigma}}$ and obtain:

$$\{\sigma(P) \mid \sigma \in \Gamma_{i+1}^{\overline{\sigma}}\} \subseteq [\widehat{\sigma}(P)]_{\sim}.$$

Note that the class $[\widehat{\sigma}(P)]_{\sim}$ is independent of the choice of $\widehat{\sigma}$. Consequently, we can bound $|\Gamma_{i+1}^{\overline{\sigma}}|$ as follows: Let m be the number of equivalence classes of $\sim$ and let $\ell_1, .., \ell_m$ denote the sizes of the respective classes. Then from our observations so far it follows:

$$|\Gamma_{i+1}^{\overline{\sigma}}| \leq \prod_{t \in [m]} \ell_t! \tag{$\star$}$$

Next, we establish a relationship between the properties of $\sim$ and the number k_{i+1}:

$$\begin{aligned}
k_{i+1} &= \Big|\mathbf{SP}(C_{j_{i+1}}) \sqcap \prod_{C_j \in J_n^i} \mathbf{SP}(C_j)\Big| - \Big|\prod_{C_j \in J_n^i} \mathbf{SP}(C_j)\Big| \\
&= \sum_{Q \in \prod_{C_j \in J_n^i} \mathbf{SP}(C_j)} (|\{P \in \mathbf{SP}(C_{j_{i+1}}) \mid Q \in \mathbf{Q}(P)\}| - 1) \\
&\geq \sum_{\substack{[P]_\sim, \\ P \in \mathbf{SP}(C_{j_{i+1}})}} (|[P]_\sim| - 1) \\
&= |\mathbf{SP}(C_{j_{i+1}})| - m.
\end{aligned}$$

The inequality in this chain requires some explanation: Fix a choice function g that maps each equivalence class $[P]_\sim$ to a part $Q \in \mathbf{Q}(P)$. By definition of $\sim$, we have $g([P]_\sim) \in \mathbf{Q}(P')$ for every $P' \in [P]_\sim$. Hence, for every $Q \in \prod_{C_j \in J_n^i} \mathbf{SP}(C_j)$, it holds: $|\{P \in \mathbf{SP}(C_{j_{i+1}}) \mid Q \in \mathbf{Q}(P)\}| - 1 \geq \sum_{[P]_\sim \in g^{-1}(Q)} (|[P]_\sim| - 1)$.
We can sum up the result of these considerations like this:

$$m \geq |\mathbf{SP}(C_{j_{i+1}})| - k_{i+1}. \tag{$\star\star$}$$

This means that the relation $\sim$ has rather many equivalence classes if k_{i+1} is small. Let us now finish the proof of the claim:

We have already established the upper bound ($\star$) for $|\Gamma_{i+1}^{\overline{\sigma}}|$. Let $p \in [m]$ be such that $\ell_p \geq \ell_t$ for all $t \in [m]$. A consequence of ($\star\star$) is: $\ell_p \leq k_{i+1} + 1$. It can be checked that the values $\ell_1, ..., \ell_m$ that maximise the bound in ($\star$) and satisfy ($\star\star$) are such that $\ell_t = 1$ for all $t \neq p$: If there is some $t \neq p$ with $\ell_t \geq 2$, then decrease ℓ_t by one and increase ℓ_p by one. This does not change the number m of equivalence classes, so it still satisfies ($\star\star$), but the value of $\prod_{t \in [m]} \ell_t!$ is strictly increased.
Therefore, ($\star$) becomes:

$$|\Gamma_{i+1}^{\overline{\sigma}}| \leq \ell_p! \leq (k_{i+1} + 1)!$$

This concludes the proof of the claim. □

Hence, in order to finish the proof of the lemma, we have to bound

$$|\Gamma_s| \leq \prod_{i=1}^{s} (\min\{(k_i + 1), f(n)\})!$$

from above (recall that s is the number of steps needed to construct J_n satisfying property (1)). We know that $\sum_{i=1}^{s} k_i$ is some fixed value $\leq n$. The values of the above product and of the sum solely depend on the sequence $(k_i)_{i \in [s]}$. Now we can make a "redistribute-weight argument" again: Let k_j be such that $k_j + 1 < f(n)$, and such that there is a $k_i \leq k_j$ with $k_i > 1$. We can decrease k_i by one and increase k_j by one. This does not change the value of the sum, and the value of the product of factorials can only get larger (as $k_j + 1$ does not exceed $f(n)$). If we iterate this process, we see that the value of the product is maximised for a sequence $(k_i)_{i \in [s]}$, where every k_i is either 1 or $k_i = f(n) - 1$ (and there may be exactly one k_i with $1 < k_i < f(n) - 1$).
For such a sequence of k_i, the value of the product is at most

$$|\Gamma_s| \leq \prod_{i=1}^{s} (\min\{(k_i + 1), f(n)\})! \leq f(n)!^{n/(f(n)-1)} \cdot 2^n$$

□

Corollary 7.2.5. *Assume the following three conditions hold:*

1. *Every $v \in \{0,1\}^n$ occurs in at least one of the colour classes C_i in $\mathcal{P}_n$.*

2. *The function $\max_{C_i \in \mathcal{P}_n} |C_i|$ is in $\mathcal{O}(n)$.*

3. *$|\mathbf{SP}(B_n)| \in o(n)$.*

Then, for sufficiently large n:

$$|\mathbf{Stab}(\mathcal{P}_n)| \leq (f(n)!)^{n/(f(n)-1)} \cdot 2^n \cdot (8 \log n)!$$

Proof. Consider the set of colour classes $J_n \subseteq \mathcal{P}_n$ that exists by Lemma 7.2.4. Since every $\pi \in \mathbf{Stab}(\mathcal{P}_n)$ fixes every $C_i \in \mathcal{P}_n$, by Lemma 4.2.5, it induces a tuple of permutations on the supporting partitions $\overline{\sigma} \in \bigtimes_{C_i \in \mathcal{P}_n} \mathbf{Sym}(\mathbf{SP}(C_i))$. In particular it also induces a $\overline{\sigma} \in \bigtimes_{C_i \in J_n} \mathbf{Sym}(\mathbf{SP}(C_i))$. By Lemma 7.2.4, there are at most $(f(n)!)^{n/(f(n)-1)} \cdot 2^n$ possibilities for such a $\overline{\sigma}$. Furthermore, each such $\overline{\sigma}$ can be realised by at most $(8\log n)!$ distinct permutations $\pi \in \mathbf{Stab}(\mathcal{P}_n)$: Due to Lemma 7.2.2 and property (1) of J_n (see Lemma 7.2.4), every π realising $\overline{\sigma}$ permutes the positions in S_n in the same way, and according to Lemma 7.2.3, there remain at most $8\log n$ positions which may be permuted arbitrarily by π. □

With this, we prove our final lemma of this subsection, which is just a more precise formulation of Lemma 7.2.1.

Lemma 7.2.6. *Assume again the three conditions from Corollary 7.2.5. Then for any $k \in \mathbb{N}$, the limit*

$$\lim_{n\to\infty} \frac{|\mathbf{Orbit}(\mathcal{P}_n)|}{2^{kn}} = \lim_{n\to\infty} \frac{n!}{|\mathbf{Stab}(\mathcal{P}_n)| \cdot 2^{kn}}$$

does not exist. That is to say, the orbit of $\mathcal{P}_n$ w.r.t. the action of $\mathbf{Sym}_n$ on $\{0,1\}^n$ grows super-polynomially in 2^n.

Proof. Plugging in the stabiliser bound from Corollary 7.2.5, we get for the fraction that we are taking the limit of:

$$\frac{n!}{|\mathbf{Stab}(\mathcal{P}_n)| \cdot 2^{kn}} \geq \frac{n!}{(f(n)!)^{n/(f(n)-1)} \cdot (8\log n)! \cdot 2^{(k+1)n}}$$

According to Stirling's Formula, factorials can be approximated as follows:

$$0.5 \cdot n! \leq \sqrt{2\pi n} \cdot \left(\frac{n}{e}\right)^n \leq n!$$

With this, we obtain the following chain of expressions, where we use the abbreviation $h(n) := \frac{f(n)+0.5}{f(n)-1}$.

$$\begin{aligned}
&\frac{n!}{(f(n)!)^{n/(f(n)-1)} \cdot (8\log n)! \cdot 2^{(k+1)n}} \\
\geq& \frac{1}{2} \cdot \sqrt{\frac{n}{8\log n}} \cdot \frac{1}{(8\pi f(n))^{0.5n/(f(n)-1)}} \cdot \left(\frac{n}{2^{k+1}e}\right)^n \cdot \left(\frac{e}{8\log n}\right)^{8\log n} \cdot \left(\frac{e}{f(n)}\right)^{f(n)\cdot(n/(f(n)-1))} \\
\geq& \frac{1}{2} \cdot \sqrt{\frac{n}{8\log n}} \cdot \left(\frac{n}{(8\pi)^{0.5/(f(n)-1)} \cdot 2^{k+1}e \cdot f(n)^{h(n)}}\right)^n \cdot \left(\frac{e}{8\log n}\right)^{8\log n} \\
=& \frac{1}{2} \cdot \sqrt{\frac{n}{8\log n}} \cdot \left(\frac{n^{(n-8\log\log n)/n}}{(8\pi)^{0.5/(f(n)-1)} \cdot 2^{k+1}e \cdot f(n)^{h(n)}}\right)^n \cdot \left(\frac{\log n \cdot e}{8\log n}\right)^{8\log n} \\
=& \frac{1}{2} \cdot \sqrt{\frac{n}{8\log n}} \cdot \left(\frac{n^{(n-8\log\log n)/n}}{(8\pi)^{0.5/(f(n)-1)} \cdot 2^{k+1}e \cdot f(n)^{h(n)}}\right)^n \cdot \left(\frac{e}{8}\right)^{8\log n}
\end{aligned}$$

In this calculation we used that $n^{8\log\log n} = (\log n)^{8\log n}$. Now we need the following claims:

Claim 7.2.2. $f(n)^{h(n)} \in \Theta(f(n))$.

Proof.

$$\lim_{n\to\infty} \frac{f(n)^{h(n)}}{f(n)} = \lim_{n\to\infty} f(n)^{\frac{1.5}{f(n)-1}} = \lim_{n\to\infty} \exp\Big(\frac{1.5}{f(n)-1} \cdot \ln(f(n))\Big) = 1.$$

□

Claim 7.2.3. $n^{(n-8\log\log n)/n} \in \Theta(n)$.

Proof.

$$\lim_{n\to\infty} \frac{n^{(n-8\log\log n)/n}}{n} = \lim_{n\to\infty} n^{(-8\log\log n)/n} = \lim_{n\to\infty} \exp\Big(\frac{-8\log\log n}{n} \cdot \ln(n)\Big) = 1.$$

□

From these claims and the fact that $f(n) \in o(n)$ it follows that there is no constant c such that

$$\frac{n^{(n-8\log\log n)/n}}{f(n)^{h(n)}} \leq c,$$

for all n. We can conclude that for large enough n,

$$\left(\frac{n^{(n-8\log\log n)/n}}{(8\pi)^{0.5/(f(n)-1)} \cdot 2^{k+1}e \cdot f(n)^{h(n)}}\right)^n \geq 2^n.$$

Furthermore,

$$\left(\frac{e}{8}\right)^{8\log n} \geq \left(\frac{n^8}{n^{24}}\right) = n^{-16}.$$

So, the dominating factor is 2^n, which means that the limit of the whole product does not exist. □

With this, we have proven the super-polynomial orbit theorem for preorders with log-sized colour classes in case that the supporting partitions of the colour classes have at most sublinearly many parts. We now move on to the remaining case.

7.2.2 The case of linearly-sized supports

This subsection is dedicated to proving the result for the case that $|\mathbf{SP}(B_n)| \in \Theta(n)$. Recall that B_n denotes the colour class $C_i \in \mathcal{P}_n$ whose supporting partition has the most parts.

Lemma 7.2.7. *Assume that the following conditions hold for B_n:*

1. $|B_n| \in \mathcal{O}(n)$.
2. $|\mathbf{SP}(B_n)| \in \Theta(n)$

Then the orbit size of B_n (and therefore also of $\mathcal{P}_n$) w.r.t. the action of $\mathbf{Sym}_n$ on $\{0,1\}^n$ grows faster than any polynomial in 2^n.

Proving this lemma requires several steps, and again, a case distinction. The relevant measure here is the number of *singleton* parts in $\mathbf{SP}(B_n)$. Firstly, we show that if the number of singleton parts in $\mathbf{SP}(B_n)$ grows sublinearly in n, while the total number of parts $|\mathbf{SP}(B_n)|$ is linear, the stabiliser of $\mathbf{SP}(B_n)$ is small enough such that Lemma 7.2.7 is true. This is a relatively straightforward calculation.

The difficult part of the proof is the case where the number of singleton parts grows linearly. An extreme example is $\mathbf{SP}(B_n) = \{\{k\} \mid k \in [n]\}$. Then, $\mathbf{Stab}(\mathbf{SP}(B_n)) = \mathbf{Sym}_n$, so nothing can be inferred solely from the supporting partition. Therefore, we will need more involved arguments that also take into account the set B_n itself, rather than only its supporting partition. This is also where we rely on the assumption that $|B_n| \in \mathcal{O}(n)$. Actually, if we used *alternating* supporting partitions, which we defined in Chapter 4, then Theorem 4.3.1 would directly tell us that the number of singleton parts is at most sublinear. However, the present chapter is a faithful presentation of the CSL paper [87], and at that time, we had not yet discovered that the case of linearly many singletons can be ruled out more generally with group-theoretic techniques. Therefore, we solve this case here with a less general combinatorial argument.

Subcase 1: Sublinear number of singleton parts
Let us begin with the easier case, where the number of singleton parts in $\mathbf{SP}(B_n)$ grows sublinearly. We denote by $S_n \subseteq [n]$ the set of positions that are in singleton parts, i.e.

$$S_n := \{k \in [n] \mid \{k\} \in \mathbf{SP}(B_n)\}.$$

Thus, the meaning of S_n is now slightly different as in the previous section. The size of $\mathbf{Stab}(\mathbf{SP}(B_n))$ can be bounded as follows:

Lemma 7.2.8. *Let $s_n := |S_n|$, and $t_n := |\mathbf{SP}(B_n)| - s_n$.*

$$|\mathbf{Stab}(\mathbf{SP}(B_n))| \leq s_n! \cdot t_n! \cdot ((n/t_n)!))^{t_n}.$$

Proof. Recall that $\mathbf{Stab}(\mathbf{SP}(B_n))$ is the setwise stabiliser of the support, so it also involves permutations that map parts to other parts. The factors $s_n!$ and $t_n!$ account for these possible permutations of the parts: All the singleton parts of $\mathbf{SP}(B_n)$ can be mapped to each other, and every non-singleton part can at most be mapped to every other non-singleton part. The factor $((n/t_n)!))^{t_n}$ is an upper bound for the number of possible permutations within all non-singleton parts, as every non-singleton part can have size at most (n/t_n). □

Corollary 7.2.9. *Let $f(n) \in o(n)$ be a function such that $s_n \leq f(n)$ and assume $|\mathbf{SP}(B_n)| \in \Theta(n)$, i.e. there exists a constant $0 < c \leq 1$, such that for all large enough n, $|\mathbf{SP}(B_n)| \geq c \cdot n$. Then, for all large enough n, the following bound holds:*

$$|\mathbf{Stab}(\mathbf{SP}(B_n))| \leq f(n)! \cdot (n/2)! \cdot (d!)^{(n/2)}.$$

Here, d is some positive constant.

Proof. We plug in the right values for s_n and $t_n = |\mathbf{SP}(B_n)| - n$ into Lemma 7.2.8. We have $s_n \leq f(n)$ by assumption. Further, using $|\mathbf{SP}(B_n)| \geq c \cdot n$, and the fact that every non-singleton part consists of at least two elements, we can bound t_n as follows:

$$c \cdot n - f(n) \leq t_n \leq \frac{n}{2}.$$

Now plug in the upper and lower bounds for t_n and s_n in the right places in the bound from Lemma 7.2.8. Note that the expression $\frac{1}{c - f(n)/n}$ that occurs in this bound can be upper-bounded by some $\frac{1}{c-\varepsilon}$ (for large enough n) because $f(n) \in o(n)$. Then set $d := \frac{1}{c-\varepsilon}$. □

Lemma 7.2.10. *Let $f(n) \in o(n)$ be a function such that $s_n \leq f(n)$ and assume $|\mathbf{SP}(B_n)| \in \Theta(n)$. Then for any $k \in \mathbb{N}$, the limit*

$$\lim_{n\to\infty} \frac{|\mathbf{Orbit}(B_n)|}{2^{kn}} \geq \lim_{n\to\infty} \frac{n!}{|\mathbf{Stab}(\mathbf{SP}(B_n))| \cdot 2^{kn}}$$

does not exist. That is to say, the orbit of B_n w.r.t. the action of $\mathbf{Sym}_n$ on $\{0,1\}^n$ grows super-polynomially in 2^n.

Proof. We bound $|\mathbf{Stab}(\mathbf{SP}(B_n))|$ according to the preceding corollary. Factorials can be approximated by the Stirling Formula (in fact, the approximation is much closer to $n!$ than what we state here, but this is sufficient and makes the calculations nicer):

$$0.5 \cdot n! \leq \sqrt{2\pi n} \cdot \left(\frac{n}{e}\right)^n \leq n!$$

Using the upper bound for $|\mathbf{Stab}(\mathbf{SP}(B_n))|$ from Corollary 7.2.9, and the Stirling Formula for the factorials $n!, f(n)!$, and $(n/2)!$, we get:

$$\begin{aligned}\frac{n!}{|\mathbf{Stab}(\mathbf{SP}(B_n))| \cdot 2^{kn}} &\geq \frac{1}{4} \cdot \sqrt{\frac{n}{2\pi \cdot f(n) \cdot (n/2)}} \cdot \left(\frac{n}{e}\right)^n \\ &\quad \cdot \left(\frac{e}{f(n)}\right)^{f(n)} \cdot \left(\frac{e}{n/2}\right)^{(n/2)} \cdot \frac{1}{(d!)^{(n/2)} \cdot 2^{kn+2}} \\ &\geq \frac{1}{4} \cdot \frac{1}{\sqrt{\pi \cdot f(n)}} \cdot \left(\frac{n}{e}\right)^{(n/2)-f(n)} \cdot \frac{1}{(d!)^{(n/2)} \cdot 2^{kn}}.\end{aligned}$$

For the last inequality, we cancelled some of the factors $\frac{n}{e}$ with the other factors, leaving a factor > 1 each time.
The factor $\frac{1}{(d!)^{(n/2)} \cdot 2^{kn}}$ can be written as $\varepsilon^{(n/2)}$, for a small enough constant $\varepsilon > 0$. For large enough n, it holds that $n - 2f(n) \geq n/2$, since $f(n) \in o(n)$. Hence, we get in total:

$$\lim_{n\to\infty} \frac{n!}{|\mathbf{Stab}(\mathbf{SP}(B_n))| \cdot 2^{kn}} \geq \lim_{n\to\infty} \frac{1}{4\sqrt{\pi \cdot f(n)}} \cdot \left(\frac{\varepsilon^2 \cdot n}{e}\right)^{(n/2)-f(n)}$$

Again, observing that $f(n) \in o(n)$, it can be seen that this limit does not exist. $\square$

This proves Lemma 7.2.7 under the assumption that $S_n \in o(n)$. Now we deal with the remaining case.

Subcase 2: Linear number of singleton parts
As already mentioned, this case requires more effort because we cannot solve it by only counting the number of permutations that stabilise $\mathbf{SP}(B_n)$. Instead, we have to relate $\mathbf{SP}(B_n)$ to the set B_n. Let $\mathbf{Sym}(B_n)$ be the group of all permutations of the strings in B_n. For $\pi \in \mathbf{Sym}_n$ and $\sigma \in \mathbf{Sym}(B_n)$, we say that π *realises* or *induces* σ, if $b^\pi = \sigma(b)$ for every $b \in B_n$.
Each $\pi \in \mathbf{Stab}(B_n) \leq \mathbf{Sym}_n$ that permutes the positions of the strings in B_n induces a unique permutation in $\mathbf{Sym}(B_n)$. Conversely, each $\sigma \in \mathbf{Sym}(B_n)$ can be realised by multiple distinct $\pi \in \mathbf{Stab}(B_n)$, but not by too many: Roughly speaking, we will see that the number of distinct realisations of a $\sigma \in \mathbf{Sym}(B_n)$ is related to $|\mathbf{Stab}^\bullet(\mathbf{SP}(B_n))|$ (which is small if there are many singleton parts). Therefore, the aim is to show that only a bounded number of $\sigma \in \mathbf{Sym}(B_n)$ can be realised by a permutation $\pi \in \mathbf{Stab}(B_n)$ at all, and that each such σ only has a small number of realisations. In total, this yields a bound on $|\mathbf{Stab}(B_n)|$.

Now let us go into the details: For a set $A \subseteq \{0,1\}^n$, we write $\bigsqcap A$ for $\bigsqcap_{a\in A} \mathbf{SP}(a)$. Note that $\mathbf{SP}(a)$ is just the partition of $[n]$ into the positions where there are zeros and ones, respectively, in a. It can be seen that $\bigsqcap A$ is a supporting partition for A, which is finer or identical to its coarsest supporting partition.
First, we show: If we partially specify a $\sigma \in \mathbf{Sym}(B_n)$ by only fixing its behaviour on a subset $A \subseteq B_n$, then any $\sigma \in \mathbf{Sym}(B_n)$ compliant with the specification can only be realised by permutations $\pi \in \mathbf{Sym}_n$ which respect the parts of $\bigsqcap A$ in some way. Roughly

speaking, if $\prod A$ consists of many small parts, then each such σ will only have a small number of realisations in $\mathbf{Sym}_n$.
The second step is then to choose a suitable set $A_n \subseteq B_n$ such that fixing the behaviour of $\sigma \in \mathbf{Sym}(B_n)$ on A_n indeed only admits a small number of realisations of σ, and such that A_n is sufficiently small to admit only few possibilities how σ can behave on A_n (as $|B_n| \in \mathcal{O}(n)$, there are $\approx n^{|A_n|}$ such possibilities).

First of all, we show how to bound the number of possible realisations of any $\sigma \in \mathbf{Sym}(B_n)$ if σ is fixed on some subset $A \subseteq B_n$. The next lemma is of a similar flavour as Lemma 7.2.2.

Lemma 7.2.11. *Let $B \subseteq \{0,1\}^n, A \subseteq B$. Let an injective mapping $p : A \longrightarrow B$ be given. Write $\prod A := \prod_{a \in A} \mathbf{SP}(a)$.*

There is an assignment of positions to parts $Q_p : [n] \longrightarrow \prod A$ with the property that $|Q_p^{-1}(P)| = |P|$ for every $P \in \prod A$, and such that:
Every $\pi \in \mathbf{Sym}_n$ realising any $\sigma \in \mathbf{Sym}(B)$ with $\sigma^{-1}(a) = p(a)$ for all $a \in A$ satisfies: $\pi(k) \in Q_p(k)$ for all $k \in [n]$ (it may be that such a π does not exist).

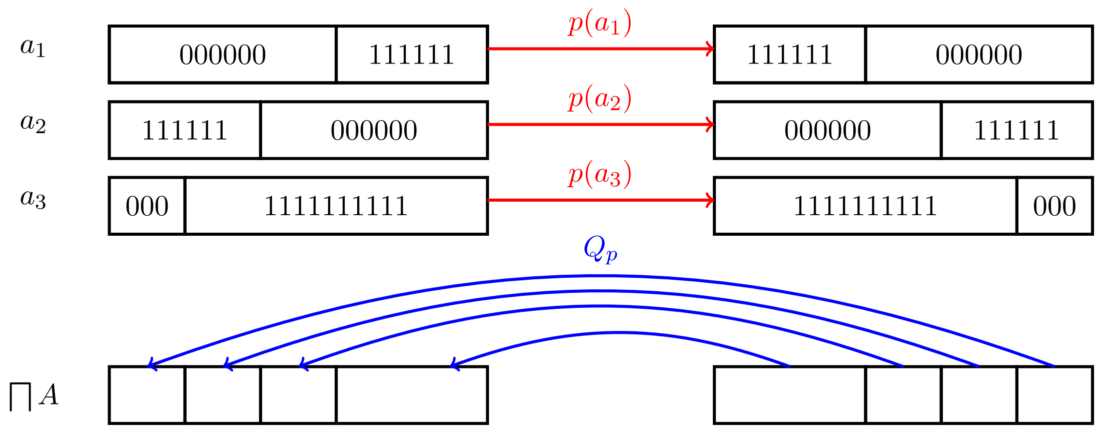


Figure 7.3: An example with $A = \{a_1, a_2, a_3\} \subseteq B$ and a mapping p that specifies an image for each string in A. Every $\sigma \in \mathbf{Sym}(B)$ that acts as the inverse of p on A can only be realised by a $\pi \in \mathbf{Sym}_n$ that complies with Q_p.

Proof. Let $\sigma \in \mathbf{Sym}(B)$ be such that $\sigma(a) = p(a)$ for each $a \in A$ and assume σ is indeed a permutation of the strings in B that can be realised by at least one $\pi \in \mathbf{Sym}_n$. We show the statement via induction on $|A|$.

If $|A| = 1$, then it is determined that $\sigma(p(a)) = a$, for the only string $a \in A$ (and a and $p(a)$ must have the same Hamming-weight). In order to map $p(a)$ to a, the ones and zeros must be mapped correctly. Indeed, this automatically yields the desired assignment Q_p: Let $P_0, P_1 \subseteq [n]$ be the positions where there are 0s and 1s, respectively, in a. Then

$Q_p(k) = P_0$, if and only if $p(a)_k = 0$.

For the inductive step, assume the statement holds for $|A| \leq m$. Consider now A with $|A| = m+1$. Take any m-element subset $A' \subset A$. Any $\pi \in \mathbf{Sym}_n$ realising σ must in particular induce the correct preimages on the elements of A', and therefore, π has to respect the assignment Q'_p given by the induction hypothesis. That is, for each $k \in [n]$, it holds $\pi(k) \in Q'_p(k)$, where $Q'_p(k)$ is a part of $\bigsqcap A'$. Let $a \in A \setminus A'$ be the unique string not contained in A'. Let $\mathbf{0}(a) := \{k \in [n] \mid a_k = 0\}$, and $\mathbf{1}(a) := \{k \in [n] \mid a_k = 1\}$.

We define the desired assignment $Q_p : [n] \longrightarrow \bigsqcap A$ as follows:

$$Q_p(k) := \begin{cases} Q'_p(k) \cap \mathbf{0}(a) & \text{, if } p(a)_k = 0 \\ Q'_p(k) \cap \mathbf{1}(a) & \text{, if } p(a)_k = 1 \end{cases}$$

It is easily seen that the range of Q_p is indeed $\bigsqcap A$, since the range of Q'_p is $\bigsqcap A'$, and we have $\bigsqcap A = \bigsqcap A' \sqcap \mathbf{SP}(a) = \bigsqcap A' \sqcap \{\mathbf{0}(a), \mathbf{1}(a)\}$.
Furthermore, it is clear that any $\pi \in \mathbf{Sym}_n$ realising σ must map every $k \in [n]$ to a position within $Q_p(k)$: By the induction hypothesis, π must map every k to a position within $Q'_p(k)$, and since $\pi(p(a)) = a$, the zeros and ones in $p(a)$ must be moved to zeros and ones in a. $\square$

We will mainly need this lemma for the restriction of the parts in $\bigsqcap A$ to the positions S_n which are singletons in $\mathbf{SP}(B_n)$. Therefore, we state the following important corollary:

Corollary 7.2.12. *Let $A \subseteq B_n$ be arbitrary, and let an injective mapping $p : A \longrightarrow B$ be given. Then every $\pi \in \mathbf{Sym}_n$ that realises a $\sigma \in \mathbf{Sym}(B_n)$ with $\sigma^{-1}(a) = p(a)$ for all $a \in A$ satisfies:*

$$\pi^{-1}(P \cap S_n) = Q_p^{-1}(P) \cap S_n \text{ for all } P \in \bigsqcap A,$$

where $Q_p : [n] \longrightarrow \bigsqcap A$ is the assignment that exists by the preceding lemma.

Proof. Lemma 7.2.11 says that $\pi^{-1}(P) = Q_p^{-1}(P)$. Since π realises a permutation in $\mathbf{Sym}(B_n)$, $\pi \in \mathbf{Stab}(B_n)$. Hence, by Lemma 4.2.5, $\pi \in \mathbf{Stab}(\mathbf{SP}(B_n))$. This means that $\pi(S_n) = S_n$, as singleton parts can only be mapped to singleton parts. Consequently, it must be the case that $\pi^{-1}(P \cap S_n) = Q_p^{-1}(P) \cap S_n$. $\square$

We move on to the more involved step of the proof. Recall that $S_n \subseteq [n]$ is the set of positions that are in singleton parts in $\mathbf{SP}(B_n)$. As the next simple lemma shows, the intersection over all strings in B_n yields a refinement of the coarsest supporting partition $\mathbf{SP}(B_n)$, so in particular, the positions in S_n are also in singleton parts in $\bigsqcap B$.

Lemma 7.2.13. *Let $B \subseteq \{0,1\}^n$. The partition $\bigsqcap B$ is a supporting partition for B.*

Proof. By the definition of the intersection, every string $b \in B_n$ is constant zero or constant one on every part $P \in \bigsqcap B$. Hence, $\mathbf{Stab}^\bullet(\bigsqcap B) \subseteq \mathbf{Stab}(B_n)$. This is the definition of a supporting partition. $\square$

With this in mind, we can now select a subset $A_n \subseteq B_n$ such that the partition $\bigsqcap A_n$ is sufficiently fine on the positions in S_n (such a subset must exist because B_n itself fulfils this condition). If we take a suitable definition of what it means to be sufficiently fine on S_n, we can even guarantee that A_n is considerably smaller than B_n. Later it will become clear that A_n is chosen in such a way that if we partially specify a $\sigma \in \mathbf{Sym}(B_n)$ on A_n (as in Lemma 7.2.11), then there are only few permutations in $\mathbf{Sym}_n$ that realise this σ. The small size of A_n then implies that not too many permutations in $\mathbf{Sym}(B_n)$ can at all be realised by permutations in $\mathbf{Stab}(B_n)$.

Lemma 7.2.14. *There exists a subset $A_n \subseteq B_n$ of size $|A_n| \leq \frac{|S_n|}{2}$ such that for each part $P \in \bigsqcap A_n$, one of the following two statements is true:*

1. *$|P \cap S_n| \leq 2$;* ***or:***
2. *$|P \cap S_n| > 2$ and for every $b \in B_n \setminus A_n$, one of these two conditions holds:*
   - *b is constant on $P \cap S_n$;* ***or***
   - *$b[P \cap S_n]$ is* imbalanced *and, for every $P' \in \bigsqcap A_n$ with $P' \neq P$, $|P' \cap S_n| > 2$, b is* constant *on $P' \cap S_n$.*

By $b[P \cap S_n]$ we mean the substring of b at the positions in $P \cap S_n$, and being imbalanced *means that $b[P \cap S_n]$ contains exactly one 0 and there is a 1 at all other positions, or vice versa (exactly one 1 and the rest 0). Being* constant *means that the string has only zeros or only ones at the respective positions.*

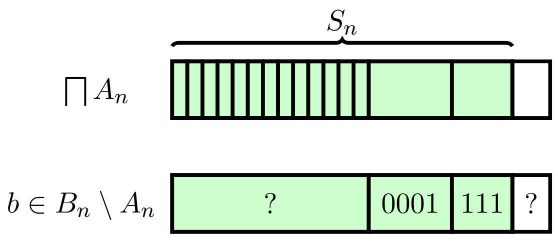


Figure 7.4: The set $A_n \subseteq B_n$ is chosen such that on the large parts of $\bigsqcap A_n$ in S_n (the green area), every $b \in B_n \setminus A_n$ is constant/imbalanced. Elsewhere, b may be arbitrary.

Proof. We construct A_n stepwise, starting with $A_n^0 := \emptyset$, and adding one string $a_i \in B_n$ in step i.
For step $i + 1$ of the construction, assume we have constructed A_n^i. For $k \in [n]$, we write $P_i(k)$ for the part of $\bigsqcap A_n^i$ that k is in. Now we let

$$K_i := \{k \in S_n \mid |P_i(k) \cap S_n| > 2\}.$$

This is the set of positions whose parts need to be refined more. If $K_i = \emptyset$, then the construction is finished because all parts of $\bigsqcap A_n^i$ satisfy condition 1 of the lemma. So assume $K_i \neq \emptyset$.

By Lemma 7.2.13, $\bigsqcap B_n$ is a supporting partition for B_n and therefore at most as coarse as $\mathbf{SP}(B_n)$. Hence, all positions in S_n are in singleton parts of $\bigsqcap B_n$.
We conclude that for all $k \in K_i$, there must be a string $b \in B_n \setminus A_n^i$ that can be added to A_n^i in order to make $P_i(k) \cap S_n$ smaller when it is intersected with $\mathbf{SP}(b)$. In fact, there may be several such strings b that we could choose to add in this step of the construction. So let

$$C_k := \{b \in B_n \setminus A_n^i \mid b \text{ is non-constant on } P_i(k) \cap S_n\}$$

be the non-empty set of such candidate strings. We restrict our candidate set further:

$$\begin{aligned}\widehat{C}_k := \{b \in C_k \mid & \text{ there are two distinct parts } P, P' \in \bigsqcap A_n^i \\ & \text{s.t. } b \text{ is non-constant on } P \cap S_n \text{ and } P' \cap S_n, \text{ and} \\ & |P \cap S_n| > 2 \text{ and } |P' \cap S_n| > 2\} \\ \cup \{b \in C_k \mid & \; b[P_i(k) \cap S_n] \text{ is not imbalanced}\}.\end{aligned}$$

We pick our next string a_{i+1} that is added in this step of the construction from one of the sets $\widehat{C}_k$, where k ranges over all positions in K_i. If $\widehat{C}_k = \emptyset$ for all these k, then A_n^i is already the desired set A_n because it satisfies the conditions of the lemma.
Otherwise, we choose a_{i+1} arbitrarily from one of the $\widehat{C}_k$ and set $A_n^{i+1} := A_n^i \cup \{a_{i+1}\}$. Then we proceed with the construction until $K_i = \emptyset$ or all $\widehat{C}_k$ are empty. In both cases, the constructed set is as required by the lemma.

It remains to show: $|A_n| \leq \frac{|S_n|}{2}$, i.e. that the construction process consists of at most $\frac{|S_n|}{2}$ steps. We do this by defining a potential function Φ that associates with any partition $\mathcal{P}$ of $[n]$ a natural number $\leq n$ that roughly says how many further refinement steps of $\mathcal{P}$ are at most possible. Concretely:

$$\Phi(\mathcal{P}) := \sum_{P \in \mathcal{P}} \max\{(|P \cap S_n| - 2), 0\}.$$

If $\mathcal{P}$ contains as its only part the whole set $[n]$, then $\Phi(\mathcal{P}) = |S_n| - 2$. Now observe that a necessary condition for adding a new string a_{i+1} to A_n is the existence of a part P with $|P \cap S_n| > 2$ in the current partition $\mathcal{P} = \bigsqcap A_n^i$. This is the case if and only if $\Phi(\mathcal{P}) > 0$. Therefore, all that remains to be shown is:

$$\Phi\left(\bigsqcap A_n^i\right) - \Phi\left(\bigsqcap A_n^{i+1}\right) \geq 2 \qquad (\star)$$

for all construction steps i.
To this end, consider step $i+1$: We add a_{i+1} to A_n^i. There are two (not necessarily disjoint) cases:
The first case is: There are two distinct parts $P, P' \in \bigsqcap A_n^i$ such that a_{i+1} is non-constant on $P \cap S_n$ and $P' \cap S_n$, and $|P \cap S_n| > 2$ and $|P' \cap S_n| > 2$. In this case, both $P \cap S_n$ and

$P' \cap S_n$ will be split when a_{i+1} is added, and each of these splits reduces the potential Φ by at least one, so ($\star$) holds.
In the second case, there is a part P such that $a_{i+1}[P \cap S_n]$ is not imbalanced and not constant. Therefore, $P \cap S_n$ will be split into two parts P_1, P_2 with $|P_1 \cap S_n| \geq 2$, $|P_2 \cap S_n| \geq 2$. This means that the contribution of $P \cap S_n$ to $\Phi(\prod A^i_n)$, namely $p := |P \cap S_n| - 2$, is reduced to $|P_1 \cap S_n| - 2 + |P_2 \cap S_n| - 2 = p - 2$ (here it is important that the new parts both have size at least two). So also in this case, ($\star$) holds. $\square$

Now we have almost all pieces that we need to prove a good bound on $|\mathbf{Stab}(B_n)|$: Given that $|B_n| \leq cn$ for some constant c, there are at most $(cn)^{|S_n|/2}$ possibilities to specify a $\sigma \in \mathbf{Sym}(B_n)$ on the elements of the set A_n from the previous lemma. It remains to show that each such σ can not be realised by too many permutations in $\mathbf{Sym}_n$. If for all parts $P \in \prod A_n$, we had $|P \cap S_n| \leq 2$, then Corollary 7.2.12 would already imply our desired bound. However, there may also be parts $P \in \prod A_n$ where $|P \cap S_n|$ is unbounded. The next lemma shows how the properties of these parts that are stated in Lemma 7.2.14 help us to deal with them. Essentially, it says that a permutation in $\mathbf{Stab}(B_n)$ is already fully specified if we only know how it moves the parts $P \in \prod A_n$ with $|P \cap S_n| \leq 2$. In other words, we can count the number of permutations in $\mathbf{Stab}(B_n)$ by just looking at their possible behaviour on the parts where $|P \cap S_n| \leq 2$.

Lemma 7.2.15. *Let $A_n \subseteq B_n$ be the subset that exists by Lemma 7.2.14, and let $p : A_n \longrightarrow B_n$ be an injective function. Let*

$$\Gamma_p := \{\pi \in \mathbf{Stab}(B_n) \mid \pi(p(a)) = a \text{ for all } a \in A_n\}.$$

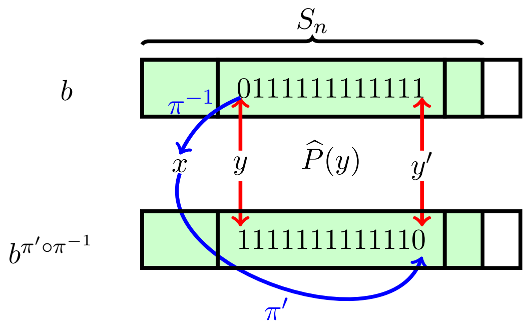
Further, let

$$P_{>2} := \{k \in S_n \mid |P(k) \cap S_n| > 2, \text{ where } P(k) \in \prod A_n \text{ is the part that } k \text{ is in}\}.$$

Then for any $\pi, \pi' \in \Gamma_p$ such that $\pi^{-1}|_{([n]\setminus P_{>2})} = \pi'^{-1}|_{([n]\setminus P_{>2})}$, it also holds $\pi^{-1}|_{P_{>2}} = \pi'^{-1}|_{P_{>2}}$.

Proof. For a contradiction, we assume that there exist $\pi, \pi' \in \Gamma_p$ such that $\pi^{-1}|_{([n]\setminus P_{>2})} = \pi'^{-1}|_{([n]\setminus P_{>2})}$, but $\pi^{-1}|_{P_{>2}} \neq \pi'^{-1}|_{P_{>2}}$. Then there is $x \in [n]$ such that $\pi(x) \in P_{>2}$, and $\pi'(x) \neq \pi(x)$ (i.e. $\pi(x)$ is the point where π^{-1} and π'^{-1} differ). Let $y := \pi(x), y' := \pi'(x)$. Let $P(y) \in \prod A_n$ be the part that y is in, and let $\widehat{P}(y) := P(y) \cap S_n$. We know that $y' \in P(y)$, too, because $\pi, \pi' \in \Gamma_p$, so this follows from Lemma 7.2.11. As $y \in P_{>2}$, in particular, $y \in S_n$. Hence, also $x, y' \in S_n$ because $\pi, \pi' \in \mathbf{Stab}(B_n)$, and by Lemma 4.2.5, singleton parts must be mapped to singleton parts. We conclude that we even have $y' \in \widehat{P}(y)$.
Now, our goal is to show that the transposition $\tau := (y\ y')$ is contained in $\mathbf{Stab}(B_n)$. This is a contradiction because in $\mathbf{SP}(B_n)$, y, y' are both in singleton parts. However, if $(y\ y') \in \mathbf{Stab}(B_n)$, then there is a coarser supporting partition in which $\{y, y'\}$ forms one part. This is a contradiction as $\mathbf{SP}(B_n)$ is the coarsest possible support.
In order to show $\tau \in \mathbf{Stab}(B_n)$, we only need to deal with those strings in B_n which are not constant on the positions $\{y, y'\}$. More precisely, we have to show that every

$b \in B_n$ with $b_y \neq b_{y'}$ has a "swapping partner" $b' \in B_n$ where $b'_y = b_{y'}$ and vice versa, and $b'_i = b_i$ for all other i.
So take any $b \in B_n$ such that w.l.o.g. $b_y = 0, b_{y'} = 1$. Note that $b \notin A_n$, as every string in A_n is constant on $P(y)$ (otherwise, $P(y)$ would not be a single part in $\bigsqcap A_n$). Furthermore, $|\widehat{P}(y)| > 2$, since $y \in P_{>2}$. Therefore, Lemma 7.2.14 implies that the substring $b[\widehat{P}(y)]$ is imbalanced and b is constant on every $P' \cap S_n$, for all $P' \in \bigsqcap A_n$ with $P' \neq P(y)$, $|P' \cap S_n| > 2$. W.l.o.g. let the imbalance of $b[\widehat{P}(y)]$ be such that $b_i = 1$ for every position $i \in \widehat{P}(y), i \neq y$. We claim that $b' := (\pi' \circ \pi^{-1})(b) \in B_n$ is the desired swapping partner of b, i.e. $\tau(b) = b'$ and vice versa. The strings b and $(\pi' \circ \pi^{-1})(b)$ look somewhat like this:



To see that $\tau(b) = (\pi' \circ \pi^{-1})(b)$, consider firstly $\pi^{-1}(b) \in B_n$. Obviously, $(\pi^{-1}(b))_x = 0$. The string $(\pi' \circ \pi^{-1})(b)$ is also in B_n and we have $(\pi' \circ \pi^{-1})(b)_{y'} = 0$. Moreover, the substring $\pi^{-1}(b)[\pi^{-1}(\widehat{P}(y))]$ is imbalanced just like $b[\widehat{P}(y)]$, so $(\pi^{-1}(b))_j = 1$ for all $j \in \pi^{-1}(\widehat{P}(y)) \setminus \{x\}$. As a consequence of Corollary 7.2.12, we have $\pi^{-1}(\widehat{P}(y)) = \pi'^{-1}(\widehat{P}(y))$. Therefore, the substring $(\pi' \circ \pi^{-1})(b)[\widehat{P}(y)]$ is also imbalanced and has a 1 at each position except y'.
This shows that $(\tau(b))[\widehat{P}(y)] = b'[\widehat{P}(y)]$. It remains to show that $b_i = b'_i$ for all $i \in [n] \setminus \widehat{P}(y)$.
We have $(\pi' \circ \pi^{-1})(i) = i$ for $i \in [n] \setminus P_{>2}$, because $\pi^{-1}|_{([n] \setminus P_{>2})} = \pi'^{-1}|_{([n] \setminus P_{>2})}$, so $b_i = b'_i$ for $i \in [n] \setminus P_{>2}$.
For $i \in P_{>2} \setminus \widehat{P}(y)$, let $\widehat{P}(i)$ be the part of $\bigsqcap A_n$ that i is in, intersected with S_n. As already said, we know from Lemma 7.2.14 that b is constant on $\widehat{P}(i)$. Analogously to what we argued already for $\widehat{P}(y)$, we get that $(\pi' \circ \pi^{-1})(\widehat{P}(i)) = \widehat{P}(i)$, so also for $i \in P_{>2} \setminus \widehat{P}(y)$, we have $b_i = b'_i$.
In total, this shows that indeed, $b' = \tau(b)$, and since $b' \in B_n$, we have $\tau \in \mathbf{Stab}(B_n)$. This is a contradiction and finishes the proof of the lemma. $\square$

With this, we can finally compute our upper bound on $|\mathbf{Stab}(B_n)|$.

Lemma 7.2.16. *Let c be a constant such that $|B_n| \leq c \cdot n$ (for large enough n). Then, for large enough n, it holds:*

$$|\mathbf{Stab}(B_n)| \leq (2cn)^{|S_n|/2} \cdot (n - |S_n|)!$$

Proof. Let $A_n \subseteq B_n$ be the subset of B_n whose existence is stated in Lemma 7.2.14. Fix any injective function $p : A_n \longrightarrow B_n$. As in the previous lemma, let

$$\Gamma_p := \{\pi \in \mathbf{Stab}(B_n) \mid \pi(p(a)) = a \text{ for all } a \in A_n\}.$$

We bound $|\Gamma_p|$ by counting the number of possible $\pi \in \Gamma_p$. We know by Lemma 7.2.15 that we only have to count the number of possibilities to choose the preimages of the elements in $[n] \setminus P_{>2}$, where again,

$$P_{>2} := \{k \in S_n \mid |P(k) \cap S_n| > 2, \text{ where } P(k) \in \bigsqcap A_n \text{ is the part that } k \text{ is in}\}.$$

For every part $P \in \bigsqcap A_n$ with $|P \cap S_n| \leq 2$, we know by Corollary 7.2.12 that $\pi^{-1}(P \cap S_n) \subseteq S_n$ is the same fixed set of size ≤ 2 for all $\pi \in \Gamma_p$, so we only have two options how π^{-1} can behave on $P \cap S_n$. The number of such parts P is at most $|S_n|/2$.

For $i \in [n] \setminus S_n$, we can only say that $\pi^{-1}(i) \notin S_n$ (by Lemma 4.2.5). Hence, every $\pi \in \Gamma_p$ can in principle permute the set $[n] \setminus S_n$ arbitrarily. In total, we conclude:

$$|\Gamma_p| \leq 2^{(|S_n|/2)} \cdot (n - |S_n|)!$$

This is for a fixed function p. The number of possible choices for p is bounded by $(cn)^{|S_n|/2}$, since $|A_n| \leq |S_n|/2$ (Lemma 7.2.14) and we are assuming $|B_n| \leq cn$.
Every $\pi \in \mathbf{Stab}(B_n)$ must occur in at least one of the sets Γ_p for some choice of p, so indeed, $(2cn)^{|S_n|/2} \cdot (n - |S_n|)!$ is an upper bound for $|\mathbf{Stab}(B_n)|$. □

As in the case where $|S_n|$ grows sublinearly in n, we conclude this part of the proof by computing the orbit size of B_n based on the stabiliser bound and comparing this to any polynomial in 2^n.

Lemma 7.2.17. *Let c be a constant such that $|B_n| \leq c \cdot n$, and $\delta_s > 0$ be a constant such that $|S_n| \geq \delta_s \cdot n$ (for large enough n). Then for any $k \in \mathbb{N}$, the limit*

$$\lim_{n\to\infty} \frac{|\mathbf{Orbit}(B_n)|}{2^{kn}} = \lim_{n\to\infty} \frac{n!}{|\mathbf{Stab}(B_n)| \cdot 2^{kn}}$$

does not exist. That is to say, the orbit of B_n w.r.t. the action of $\mathbf{Sym}_n$ on $\{0,1\}^n$ grows super-polynomially in 2^n.

Proof. Plugging in $|S_n| \geq \delta_s \cdot n$ into the stabiliser-bound from Lemma 7.2.16 yields:

$$|\mathbf{Stab}(B_n)| \leq (2cn)^{(\delta_s n)/2} \cdot ((1 - \delta_s)n)!$$

This is true because larger values for $|S_n|$ only make the expression $2^{|S_n|/2} \cdot (n - |S_n|)!$ smaller.

Replacing factorials with the Stirling Formula, the fraction that we are taking the limit of becomes at least:

$$\begin{aligned}
&\frac{1}{2\sqrt{1-\delta_s}} \cdot \left(\frac{1}{(1-\delta_s)}\right)^{(1-\delta_s)n} \cdot \left(\frac{n}{e}\right)^{\delta_s n} \cdot \frac{1}{(2cn)^{(\delta_s n)/2}} \cdot \frac{1}{2^{kn}} \\
=&\frac{1}{2\sqrt{1-\delta_s}} \cdot \left(\frac{1}{2^k(1-\delta_s)}\right)^{(1-\delta_s)n} \cdot \left(\frac{n}{2c \cdot 4^k \cdot e^2}\right)^{(\delta_s n)/2} \\
=&\frac{1}{2\sqrt{1-\delta_s}} \cdot \left(\frac{n}{(2^k(1-\delta_s))^{2(1-\delta_s)/\delta_s} \cdot 2c \cdot 4^k \cdot e^2}\right)^{(\delta_s n)/2}
\end{aligned}$$

The denominator is constant, so the limit does not exist. □

This lemma together with Lemma 7.2.10 proves Lemma 7.2.7.
Theorem 7.0.4 now follows directly from Lemmas 7.2.1 and 7.2.7, since these two lemmas cover all cases.

7.3 Conclusion and future research

We have computed an estimate for the orbit-size of any ordered partition of $\{0,1\}^n$ into classes of size $\mathcal{O}(n)$, which is logarithmic in $|\{0,1\}^n|$. The result is that no matter how these partitions are chosen for all values of n, the size of their $\mathbf{Sym}_n$-orbit grows faster than 2^{nk}, for any fixed constant $k \in \mathbb{N}$. This is super-polynomial in the size of the n-dimensional hypercube, and so, hypercubes do not admit CPT-definable preorders with colour classes of at most logarithmic size. One can now ask in how far this result generalises to preorders with larger colour classes. There must be some colour class size at which the orbit-size of the preorders drops from super-polynomial to polynomial in 2^n; namely, orbit-size one can be achieved, at the latest when the colour class size equals the structure size. It would be interesting to know which colour class size marks the boundary between polynomial and super-polynomial orbits. This would rule out even more (hypothetical) preorder-based CPT-algorithms for the CFI-query.
The proof that we have presented in this chapter can probably be generalised to colour classes of size $o(n^{1.9})$ but one can check that the case with linearly many singleton parts in the supporting partition no longer works if the colour classes have size $\Omega(n^2)$. However, with the new technique of alternating supporting partitions and Theorem 4.3.1, we can deal with this case regardless of the colour class size. Then the only remaining bottleneck is Lemma 7.2.3, which shows that a small colour class size implies a large number of singleton parts in the intersection over the supporting partitions of all colour classes. This is necessary to get a good bound on the stabiliser of the whole preorder in Case 1 of our proof. It seems plausible that Lemma 7.2.3 would still yield a sufficient lower bound on the number of singletons in the intersection partition if the colour classes are of polylogarithmic size, i.e. $\mathcal{O}(n^k)$. Then in total, we might be able to prove the non-definability of preorders with *polylogarithmic* colour classes in hypercubes.

However, carrying out this idea is left for future work because firstly, lifting our result to polylogarithmic colour classes would not give any immediate benefit in the current situation, and secondly, we found the crucial Theorem 4.3.1, which opens up this perspective, only towards the end of writing this thesis.

Therefore, we move on to the next chapter, in which we abandon the preorders and study more symmetric objects instead, namely XOR-circuits. These can be seen as the structural backbone of the h.f. sets constructed in general CPT-algorithms for the CFI-query.

8 Lower bounds for Choiceless Polynomial Time via Symmetric Circuits

In the previous chapter we saw that there are unordered CFI-structures which cannot be preordered in CPT such that each colour class has at most logarithmic size. This shows that the CPT-algorithm from [91] for the CFI-query on such preordered structures is not helpful in the unordered case. In this chapter, we go beyond this particular algorithm and aim to understand the limitations of a broader class of CPT-algorithms for the CFI-problem. The overarching idea in the previous chapter and in this one is the following observation: A necessary condition for a CPT-algorithm or a family of CPT-algorithms being able to solve the CFI-query on some class $\mathcal{K}$ of unordered base graphs is always the *existence* of certain combinatorial objects whose symmetries are similar to the symmetries of the graphs in $\mathcal{K}$. In case of the preorder-based algorithm, it is clear that it can only be adapted to CFI-graphs over $\mathcal{K}$ if for each graph in $\mathcal{K}$, there exists a preorder with logarithmic colour classes whose orbit is at most of polynomial size (i.e. a preorder which does not break the symmetry of the graph "too much"). For the more general algorithms that we will study now, it turns out that the appropriate combinatorial objects are not preorders, but Boolean XOR-circuits (which actually subsume preorders): The existence of certain polynomial-size XOR-circuits with the right symmetries is a necessary condition for the CFI-problem on a class $\mathcal{K}$ being solvable by a CPT-algorithm from the family of algorithms we are going to define. This establishes the study of symmetric circuits as a potential route towards separating CPT from P.
In the case of preorders, we were able to show that indeed, there is a graph class (n-dimensional hypercubes) on which no sufficiently symmetric preorder exists. This was a feasible task because preorders with small colour classes are inherently very asymmetric objects. In the next chapter, we attempt a similar analysis for the XOR-circuits from this chapter, again with respect to the symmetries of hypercubes. We do not completely succeed in showing that the required symmetric XOR-circuits do not exist but we come close: There are only few extra restrictions we have to put on the circuits in order to rule out their existence. Thus, the route towards CPT lower bounds via the study of symmetric circuits that we open up in this chapter may indeed have the potential to separate a large class of CPT-algorithms (or even all of CPT) from PTIME.

The first step is to define a class of hereditarily finite objects over unordered CFI-structures which generalises the design principle of the objects that were used in [40] and [91] to decide the CFI-query (see Chapter 6). We call these objects *CFI-symmetric*. In [40], the term *super-symmetric* was also used to describe the relevant objects; this concerns another property of these h.f. sets that is independent of what we call CFI-symmetry. The family of choiceless CFI-algorithms that we mainly consider consists of all algorithms which construct such CFI-symmetric and super-symmetric h.f. objects. More important for our techniques is the CFI-symmetry – the requirement that the sets also be super-symmetric can be dropped but this leads to slightly weaker results. Anyway, the algorithms from [40] and [91] for the CFI-query over linearly ordered base graphs, preordered base graphs, and base graphs with linear degree are both CFI-symmetric *and* super-symmetric. As we will show, the power of all these algorithms and potential generalisations of them depends on whether or not certain symmetric XOR-circuits exist. This is the second step in this chapter. It consists in a construction that transforms CFI-symmetric h.f. sets into XOR-circuits. Under the assumption that the CFI-symmetric set is constructed by a CPT-algorithm which decides the CFI-query on a class $\mathcal{K}$ of base graphs, the resulting XOR-circuit will have certain properties, such as symmetry and a specific fan-in bound on the XOR-gates. Therefore, the existence of XOR-circuits that satisfy these conditions is a necessary condition for the existence of a CPT-algorithm that solves the CFI-query on a graph class $\mathcal{K}$ via the construction of a CFI-symmetric h.f. set. This is formally stated in Theorem 8.5.2. We can summarise it less formally as follows:

Theorem 8.0.1. *Let $(G_n = (V_n, E_n))_{n\in\mathbb{N}}$ be a sequence of base graphs. Let $\mathfrak{G}_n^S$ be a CFI-graph over G_n, and let $\boldsymbol{tw}_n$ denote the treewidth of G_n. If there exists a* CPT*-program* Π *that is* super-symmetric *and* CFI-symmetric *and decides the CFI-query on the instances $\mathfrak{G}_n^S$, for all $n \in \mathbb{N}$, then there also exists a family $(C_n)_{n\in\mathbb{N}}$ of XOR-circuits such that*

1. *The number of gates in C_n is polynomial in $|\mathfrak{G}_n^S|$.*
2. *The $\mathbf{Aut}(G_n)$-orbit of the circuit has size polynomial in $|\mathfrak{G}_n^S|$.*
3. *C_n is* sensitive *to $\Omega(\boldsymbol{tw}_n)$ many input bits.*
4. *The* fan-in dimension *of C_n is $\mathcal{O}(\log|\mathfrak{G}_n^S|)$.*

The input gates of these circuits C_n are labelled with the edges E_n of the base graph. Therefore, $\mathbf{Aut}(G_n)$ acts on these circuits in a natural way. The *fan-in dimension* of the circuits is a generalisation of the more standard *fan-in degree*, which we introduce in Section 8.3.

After the proof of Theorem 8.5.2, in Section 8.6, we attempt to generalise the correspondence between h.f. sets over CFI-graphs and XOR-circuits to *arbitrary* sets. That is, we drop the requirement that the h.f. set be *CFI-symmetric*. This does work out as long as certain Boolean vector spaces that appear as "local stabilisers" within the h.f. set possess a *symmetric basis*. We do not know for which h.f. sets this is the case but we have a counterexample (Lemma 8.6.30), so the class of "h.f. sets with symmetric bases" is a strict generalisation of the CFI-symmetric h.f. sets (see Lemma 8.6.28 and Example 8.6.29), but still does not seem to encompass *all* CPT-definable sets. In order to separate CPT from P via lower bounds against symmetric XOR-circuits, it would be desirable to establish a connection between these circuits and *all possible* CPT-algorithms for the

CFI-query. It remains open whether such a general correspondence can be achieved but at least for the algorithms we call CFI-symmetric, XOR-circuits are a fairly natural description of the relevant h.f. sets.

8.1 Symmetries and supports of hereditarily finite sets over CFI graphs

In Chapter 5, we discussed the automorphisms of unordered CFI-structures $\mathfrak{G}^S$ over a base graph $G = (V, E)$. Recall that the universe of $\mathfrak{G}^S$ is $\widehat{V}_S \cup \widehat{E}$, which denotes the vertices within vertex and edge gadgets, respectively. In this chapter, we are concerned with the action of $\mathbf{Aut}(\mathfrak{G}^S)$ on the objects in $\mathrm{HF}(\widehat{E})$. We only focus on h.f. objects in $\mathrm{HF}(\widehat{E})$ and neglect the vertices $\widehat{V}_S$ in vertex-gadgets. This simplifies matters and it is not really a restriction because any CPT-algorithm that solves the CFI-query by constructing objects in $\mathrm{HF}(\widehat{E} \cup \widehat{V}_S)$ should also be able to work only with objects from $\mathrm{HF}(\widehat{E})$. We do not formally prove this but, informally speaking, any atom from $v^X \in \widehat{V}_S$ that might occur in a h.f. set can alternatively be described by the set $\{e_i \in \widehat{E} \mid \{v^X, e_i\} \in E(\mathfrak{G}^S)\}$, its neighbourhood in the CFI-graph. So for each $v^X \in \widehat{V}_S$, there is a unique definable object in $\mathrm{HF}(\widehat{E})$ that serves as a “name” for v^X. Therefore, we only focus on CPT-algorithms that activate objects in $\mathrm{HF}(\widehat{E})$.

Let $\mathfrak{G}^S$ be a CFI-graph over $G = (V, E)$ and $x \in \mathrm{HF}(\widehat{E})$. The automorphism group $\mathbf{Aut}(\mathfrak{G}^S)$, as well as the edge-flip-group $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})$, and the automorphisms of the base graph $\mathbf{Aut}(G)$ act on $\widehat{E}$ and therefore also on $\mathrm{HF}(\widehat{E})$: For example, let $\pi \in \mathbf{Aut}(G)$, and $x \in \mathrm{HF}(\widehat{E})$. Then $\pi x = \{\pi y \mid y \in x\}$. If x is an atom e_i, with $i \in \{0, 1\}$ and $e \in E$, then $\pi x = \pi(e)_i$.
An automorphism $\pi \in \mathbf{Aut}(G)$ (or more generally, any permutation in $\mathbf{Sym}(V)$) *stabilises* an object $x \in \mathrm{HF}(\widehat{E})$, if $\pi x = x$. More precisely, this means that π, which acts on the atoms of x, extends to some automorphism σ of the DAG-structure $(\mathrm{tc}(x), \in)$, such that for every atom $e_i \in \mathrm{tc}(x)$, $\sigma(e_i) = \pi(e)_i$.
Now we briefly introduce the different orbits and corresponding stabiliser groups of any object $x \in \mathrm{HF}(\widehat{E})$. Recall from Section 5.2 that the automorphism group $\mathbf{Aut}(\mathfrak{G}^S)$ of the CFI-structure is composed of $\mathbf{Aut}(G)$ and $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}^S)$, the automorphisms of the base graph, and the edge flips of the CFI structure. We treat the actions of these two groups separately. In addition, we consider the group $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}) \cong \mathbb{F}_2^E$ of *all* possible edge flips, including ones that are not automorphisms of $\mathfrak{G}^S$. This group is easier to deal with and plays an important role in the context of *super-symmetric* CFI-algorithms. Since the group of edge-flips, $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})$, is isomorphic to the Boolean vector space $\mathbb{F}_2^E$, we often identify an automorphism $\rho_F \in \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})$ with its characteristic vector $\chi(F) \in \mathbb{F}_2^E$. We

will use the following notation:

$$\begin{aligned}
\mathbf{Orb}_E(x) &:= \{\rho_F(x) \mid \rho_F \in \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})\}.\\
\mathbf{Stab}_E(x) &:= \{\chi(F) \mid \rho_F \in \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}), \rho_F(x) = x\} \leq \mathbb{F}_2^E.\\
\mathbf{Orb}_{\mathrm{CFI}}(x) &:= \{\rho_F(x) \mid \rho_F \in \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}^S)\}.\\
\mathbf{Stab}_{\mathrm{CFI}}(x) &:= \{\chi(F) \mid \rho_F \in \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}^S), \rho_F(x) = x\} \leq \mathbb{F}_2^E.\\
\mathbf{Orb}_{\mathfrak{G}^S}(x) &:= \{(\rho_F, \pi)(x) \mid (\rho_F, \pi) \in \mathbf{Aut}(\mathfrak{G}^S)\}.\\
\mathbf{Stab}_{\mathfrak{G}^S}(x) &:= \{(\rho_F, \pi) \in \mathbf{Aut}(\mathfrak{G}^S) \mid (\rho_F, \pi)(x) = x\}.\\
\mathbf{Orb}_G(x) &:= \{(\rho_\emptyset, \pi)(x) \mid \pi \in \mathbf{Aut}(G)\}.\\
\mathbf{Stab}_G(x) &:= \{\pi \in \mathbf{Aut}(G) \mid (\rho_\emptyset, \pi)(x) = x\}.
\end{aligned}$$

It should be emphasised that $\mathbf{Stab}_E(x)$ is always a subspace of $\mathbb{F}_2^E$, so it makes sense to speak about its dimension and to apply linear transformations to it. At this point, we observe for future reference that all objects in the same $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})$-orbit have the same $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})$-stabiliser because the group is Abelian:

Lemma 8.1.1. *Let $x, x' \in \mathrm{HF}(\widehat{E})$ such that $x' = \rho_F(x)$, for some $\rho_F \in \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})$. Then $\mathbf{Stab}_E(x) = \mathbf{Stab}_E(x')$.*

Proof. We have $\mathbf{Stab}_E(x') = \{\chi(F) + \alpha + \chi(F) \mid \alpha \in \mathbf{Stab}_E(x)\}$. Since $\chi(F) = \chi(F)^{-1}$ and $\mathbb{F}_2^E$ is Abelian, we have $\mathbf{Stab}_E(x') = \mathbf{Stab}_E(x)$. □

Ultimately, the main property of CPT-definable objects x that we always exploit is their bounded orbit-size. This applies not only to a set x itself but also to all elements in its transitive closure. Therefore we define:

$$\begin{aligned}
\mathbf{maxOrb}_E(x) &:= \max_{y \in \mathrm{tc}(x)} |\mathbf{Orb}_E(y)|.\\
\mathbf{maxOrb}_{\mathrm{CFI}}(x) &:= \max_{y \in \mathrm{tc}(x)} |\mathbf{Orb}_{\mathrm{CFI}}(y)|.
\end{aligned}$$

Lemma 8.1.2. *Let $x \in \mathrm{HF}(\widehat{E})$. Then $\mathbf{maxOrb}_E(x) \leq |\mathbf{Orb}_E(x)| \cdot |\mathrm{tc}(x)|$ and $\mathbf{maxOrb}_{\mathrm{CFI}}(x) \leq |\mathbf{Orb}_{\mathrm{CFI}}(x)| \cdot |\mathrm{tc}(x)|$.*

Proof. Let $y \in \mathrm{tc}(x)$ be the set where $\mathbf{maxOrb}_E(x)$ is attained, i.e. $|\mathbf{Orb}_E(y)| = \mathbf{maxOrb}_E(x)$. Let $Y := \mathbf{Orb}_E(y) \cap \mathrm{tc}(x)$. Clearly, for any $\rho_F \in \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})$, $\rho_F(Y) \subseteq \mathrm{tc}(\rho_F(x))$, and $\rho_F(x) \in \mathbf{Orb}_E(x)$. Hence:

$$|\mathbf{Orb}_E(y)| \leq \Bigg| \bigcup_{\rho_F \in \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})} \rho_F(Y) \Bigg| \leq |\mathbf{Orb}_E(x)| \cdot |\mathrm{tc}(x)|.$$

Similarly, the statement for $\mathbf{maxOrb}_{\mathrm{CFI}}(x)$ is proven. □

If x is CPT-definable in $\mathfrak{G}^S$, then its $\mathbf{Aut}(\mathfrak{G}^S)$-orbit and therefore $|\mathbf{Orb}_{\mathrm{CFI}}(x)|$ has polynomial size. By the preceding lemma, then also $\mathbf{maxOrb}_{\mathrm{CFI}}(x)$ is polynomially bounded. As indicated earlier, the group $\mathbb{F}_2^E \cong \mathbf{Aut}_{\text{CFI}}(\mathfrak{G})$ is easier to handle than $\mathbf{Aut}_{\text{CFI}}(\mathfrak{G}^S)$, which is why some of our results are phrased in terms of $\mathbf{maxOrb}_E(x)$ rather than $\mathbf{maxOrb}_{\text{CFI}}(x)$. However, the CPT-definability of x only entails a polynomial bound on $\mathbf{maxOrb}_{\text{CFI}}(x)$, not on $\mathbf{maxOrb}_E(x)$. Therefore, it is sometimes convenient to restrict ourselves to *super-symmetric* objects x. In [40], this term has been coined for objects that are stable under the group of all edge flips $\mathbf{Aut}_{\text{CFI}}(\mathfrak{G})$. From now on, we will use it in the following precise sense:

Definition 8.1.3 (Super-symmetric objects)**.** *Fix a family of CFI-graphs $(\mathfrak{G}_n^S)_{n\in\mathbb{N}}$ and a $\mu_n \in \text{HF}(\widehat{E}_n)$ for every n. The objects μ_n are* super-symmetric *if*

$$|\mathbf{Orb}_E(\mu_n)| \leq \text{poly}(|\mathfrak{G}_n^S|).$$

So super-symmetric objects in this sense satisfy the same orbit bound with respect to the bigger symmetry group $\mathbb{F}_2^E \cong \mathbf{Aut}_{\text{CFI}}(\mathfrak{G})$ as all CPT-definable objects naturally do with respect to the automorphism group of the input structure. In [40], super-symmetry is meant in the even stricter sense that a h.f. object has orbit-size 1 under the group $\mathbf{Aut}_{\text{CFI}}(\mathfrak{G})$.

Lemma 8.1.4. *If μ_n is* super-symmetric *and* CPT-definable *in $\mathfrak{G}_n^S$, then $\mathbf{maxOrb}_E(\mu_n)$ is polynomially bounded in $|\mathfrak{G}_n^S|$.*

Proof. By super-symmetry, $|\mathbf{Orb}_E(\mu_n)|$ is polynomially bounded. By CPT-definability, $|\text{tc}(\mu_n)|$ is polynomially bounded. Hence the statement follows with Lemma 8.1.2. □

Supports for CFI automorphisms

Recall from Chapter 4 that a *support* of a permutation group $\Gamma \leq \mathbf{Sym}(A)$ is a subset $S \subseteq A$ such that $\mathbf{Stab}(S) \leq \Gamma$. A support of a h.f. set is a support of its stabiliser group. In this chapter, we will use a slightly modified notion of support, that we call *CFI-support*. The reason why we consider this modification is again because the automorphism group of unordered CFI-structures is composed of two groups that we would like to deal with separately. CFI-supports are essentially supports with respect to the group of edge-flips $\mathbf{Aut}_{\text{CFI}}(\mathfrak{G})$.

Definition 8.1.5 (CFI-support)**.** *A* CFI-support *of an object $x \in \text{HF}(\widehat{E})$ is a subset $S \subseteq E$ such that every $\rho_F \in \mathbf{Aut}_{\text{CFI}}(\mathfrak{G})$ with $F \cap S = \emptyset$ fixes x.*

There is always a unique minimal CFI-support:

Lemma 8.1.6. *Let $\mathfrak{G}^S$ be a CFI-instance and $x \in \text{HF}(\widehat{E})$. Let $A_1, A_2 \subseteq E$ be CFI-supports of x. Then $A_1 \cap A_2$ is also a CFI-support of x.*

Proof. Assume $A_1 \cap A_2$ was not a CFI-support of x. Then there is $F \subseteq E$ disjoint from $A_1 \cap A_2$ such that $\rho_F(x) \neq x$. Let $F_1 := F \cap A_1$ and $F_2 := F \cap A_2$. These sets are both non-empty, because: If F did not intersect A_1, then $\rho_F(x) = x$ because A_1 is a CFI-support for x. Similarly for A_2. Also, by assumption, F_1 and F_2 are disjoint from $A_1 \cap A_2$, and therefore, also $F_1 \cap F_2 = \emptyset$. Furthermore, $F' := F \setminus (F_1 \cup F_2)$ is disjoint from

$A_1 \cup A_2$ and therefore, $\rho_{F'}$ fixes x. It follows that flipping the edges in $F_1 \cup F_2$ moves x, because by assumption, $\rho_F(x) \neq x$. But this is a contradiction because $\rho_{F_1}(x) = x$ (since F_1 is disjoint from the support A_2), and analogously, $\rho_{F_2}(x) = x$. □

This justifies the following definition:

Definition 8.1.7 (Minimal CFI-support)**.** *For $x \in \mathrm{HF}(\widehat{E})$, $\sup_{\mathrm{CFI}}(x) \subseteq E$ denotes the unique minimal subset of E that is a CFI-support of x.*

It should be noted that the minimum CFI-support of a h.f. object x over a CFI-structure $\mathfrak{G}^S$ is not necessarily equal to a smallest $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}^S)$-support of x. With this, we mean a minimum size subset $A \subseteq E$ of edges such that any $\rho_F \in \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}^S)$ with $F \cap A = \emptyset$ stabilises x. The difference is that here, we only consider edge flips along cycles, while our notion of CFI-support takes *all* possible combinations of edge flips into account. The disadvantage of working with $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}^S)$-supports is that smallest supports are not necessarily unique then: Imagine the base graph G is just a cycle, and suppose x is some h.f. object that is changed whenever any edge of G is flipped. Then the minimum CFI-support of x contains the whole cycle, whereas any single edge of it is already a smallest $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}^S)$-support: Once any edge on the cycle is fixed, the whole cycle can no longer be flipped. Thus, there exist as many smallest $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}^S)$-supports as there are edges on the cycle. For this and other reasons, our CFI-supports are defined with respect to $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})$ instead of $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}^S)$.

As explained above, for general CPT-definable objects, the orbit size can only be bounded when the "true" automorphism group $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}^S)$ is considered. Only when the object is additionally super-symmetric, also the $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})$-orbit size is polynomial. However, depending on the structure of the base graph, we can sometimes bound the $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})$-orbit size as well, even if the object is not necessarily super-symmetric:

Lemma 8.1.8. *Let $\mu \in \mathrm{HF}(\widehat{E})$ be a h.f. set over $\mathfrak{G}^S$. Let k be the number of connected components in the graph $G - \sup_{\mathrm{CFI}}(\mu)$ (i.e. the base graph after removing the edges in the minimum CFI-support). Then $|\mathbf{Orb}_E(\mu)| \leq 2^{k^2} \cdot |\mathbf{Orb}_{\mathfrak{G}^S}(\mu)|$.*

Proof. Let $A = \sup_{\mathrm{CFI}}(\mu) \subseteq E$ be the smallest CFI-support of μ. Then for every $\rho_F \in \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})$ with $F \cap A = \emptyset$, it holds that $\rho_F(\mu) = \mu$. Thus, the effect of an edge-flip ρ_F on μ depends only on $F \cap A$. So we have:

$$|\mathbf{Orb}_E(\mu)| \leq 2^{|A|},$$

because there are $2^{|A|}$ ways how any $F \subseteq E$ can intersect the support S, and if $F \cap A = F' \cap A$, then also $\rho_F(\mu) = \rho_{F'}(\mu)$.
Now we compute a lower bound on $|\mathbf{Orb}_{\mathfrak{G}^S}(\mu)|$ by analysing how many subsets of A can occur as the intersection $F \cap A$ for an automorphism $\rho_F \in \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}^S)$. In contrast to the edge-flips in $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})$, these are the edge-flips along cycles in G. Let $X_1, ..., X_k \subseteq V(G)$ denote the vertex-sets of the connected components in the graph $G - A$. We partition the edge-set A into at most k^2 many subsets, according to the components that the edges

connect. So for each pair $(i,j) \in [k]^2$, let $\mathcal{A}_{(i,j)} \subseteq A$ denote those edges in A that run between the components X_i and X_j. Now it can be seen that for every pair $i \neq j$, for every $B \subseteq \mathcal{A}_{(i,j)}$ of even cardinality, there exists some symmetric difference of cycles C_B in G whose intersection with A is exactly B. This is because any two edges $e, e' \in \mathcal{A}_{(i,j)}$ lie on a cycle through the components X_i and X_j. For $\mathcal{A}_{(i,i)} \subseteq A$, *every* subset $B \subseteq \mathcal{A}_{(i,i)}$ can be generated by the symmetric difference of some cycles because the endpoints of every $e \in \mathcal{A}_{i,i}$ are in the same connected component (but for simplicity, we pretend that also in this case, only the even subsets of $\mathcal{A}_{(i,i)}$ can be hit by the symmetric difference of some cycles). Summing up these considerations, we have:

$$\begin{aligned}&|\{B \subseteq A \mid \text{ there exists a } \rho_F \in \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}^S) \text{ such that } F \cap A = B\}| \\ &\geq \prod_{(i,j) \in [k]^2} 2^{|\mathcal{A}_{(i,j)}|-1} = 2^{|A|-k^2}.\end{aligned}$$

Let s denote the number of $B \subseteq A$ in the above set such that flipping B stabilises μ. Then by the Orbit-Stabiliser Theorem, we have $|\mathbf{Orb}_{\mathfrak{G}^S}(\mu)| \geq 2^{|A|-k^2}/s$ and $|\mathbf{Orb}_E(\mu)| \leq 2^{|A|}/s$. Putting these two inequalities together, we get the desired bound $|\mathbf{Orb}_E(\mu)| \leq 2^{k^2} \cdot |\mathbf{Orb}_{\mathfrak{G}^S}(\mu)|$. □

This lemma essentially says that it does not make a difference whether we consider orbit-sizes with respect to the group $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})$ of all edge flips or the group $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}^S)$ of cycle edge flips, as long as the CFI-support of an object separates the base graph only into a small number of components:

Corollary 8.1.9. *Fix a family* $(\mathfrak{G}_n^S)$ *of CFI-structures. Let* $\mu_n \in \mathrm{HF}(\widehat{E}_n)$*. If the number of connected components in* $G_n - \sup_{\mathrm{CFI}}(\mu_n)$ *is at most* $\mathcal{O}(\sqrt{\log|\mathfrak{G}_n^S|})$*, then* $|\mathbf{Orb}_E(\mu_n)| \leq \mathrm{poly}(|\mathfrak{G}_n^S|) \cdot |\mathbf{Orb}_{\mathfrak{G}^S}(\mu)|$.

8.2 CFI-symmetric hereditarily finite sets

In this section, we formalise the specific class of h.f. objects that have a particularly nice correspondence to XOR-circuits. We call these objects *CFI-symmetric* because they consist of "building blocks" that have the same symmetry property as the vertex gadgets in CFI-graphs: They are stabilised whenever an even number of "incident edges" is flipped. This notion is independent of *super-symmetry* defined in the last section. Super-symmetry means that a h.f. object is symmetric under $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})$ whereas *CFI-symmetry* tells us something about the internal structure of the object. Neither of these kinds of symmetry implies the other, and both are satisfied by the objects in all known choiceless CFI-algorithms.

As already said, a set $\mu \in \mathrm{HF}(\widehat{E})$ is *CFI-symmetric* if the objects of which it is composed behave like CFI-gadgets: They are stabilised whenever an even number of their "incident edges" is "flipped", and they themselves are "flipped" whenever an odd number of incident edges is flipped. This description uses the terminology of CFI-graphs; in order to make it precise, we have to say what these words mean in the context of objects in $\mathrm{HF}(\widehat{E})$. By "objects of which μ is composed", we do not mean the elements of $\mathrm{tc}(\mu)$, as it is usually the case when we talk about h.f. sets. We rather mean the *connected components* of these sets, as this concept has been called in [40].

Let us first define the following equivalence relation $\sim_E$ on the elements $x \in \mathrm{tc}(\mu)$: For $x, x' \in \mathrm{tc}(\mu)$, we say $x \sim_E x'$ iff there exists a $\rho_F \in \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})$ such that $x' = \rho_F(x)$. Recall from Chapter 5 that $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}) = \{\rho_F \mid F \subseteq E\}$, so this is the group of all CFI-edge-flips (including ones that are not automorphisms of $\mathfrak{G}^S$ but isomorphisms from $\mathfrak{G}^S$ into some other CFI-graph $\mathfrak{G}^R$ of the same parity). The $\sim_E$-equivalence class in $\mathrm{tc}(\mu)$ of an object $x \in \mathrm{tc}(\mu)$ is denoted $[x]_{\sim_E}$ or $[x]$. The relation $\sim_E$ induces a partition $\mathcal{C}(x)$ on each $x \in \mathrm{tc}(\mu)$, namely $\mathcal{C}(x) := \{([z]_{\sim_E} \cap x) \mid z \in x\}$. In [40], the elements of $\mathcal{C}(x)$ are called the *connected components* of x. Now in a *CFI-symmetric* object, each connected component $\gamma \in \mathcal{C}(x)$, for each $x \in \mathrm{tc}(\mu)$, behaves like a CFI-gadget. That is, the component has exactly two images under $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})$: It can either be flipped or stabilised, and which of these two is the case, depends on the parity of flipped components in the elements of γ. To make all this more clear, let us consider a h.f. object based on the same idea as the ones in [40] and [91].

Example 8.2.1. *Here is an example h.f. set $\mu_{\{e,f,g\}} \in \mathrm{HF}(\widehat{E})$ with $E = \{e, f, g\}$. It is structured similarly as the μ-objects in the known super-symmetric algorithms and tracks the parity of edge-flips for the edges e, f, g. For better readability, the set is printed in a structured form, so the sets $\mu_{\{f,g\}}$ and $\widetilde{\mu}_{\{f,g\}}$ are shown in the level below.*

$$\begin{tikzcd}[column sep=small]
\mu_{\{e,f,g\}} = \Big\{ & \{\mu_{\{f,g\}}, e_0\}, \arrow[d, dash] & \{\widetilde{\mu}_{\{f,g\}}, e_1\} \arrow[d, dash] & \Big\} \\
& \{\{f_0, g_0\}, \{f_1, g_1\}\} & \{\{f_0, g_1\}, \{f_1, g_0\}\} &
\end{tikzcd}$$

Each of the μ-objects has only one connected component that consists of two sets which are related by $\sim_E$. For example, the two elements of $\mu_{\{e,f,g\}}$ are mapped to each other whenever an even number of edges is flipped. These two elements of $\mu_{\{e,f,g\}}$ themselves have two connected components: Clearly, e_0 and $\mu_{\{f,g\}}$ cannot be mapped to each other by any edge-flip. The same goes for example for f_0 and g_0. They form distinct components of the set $\{f_0, g_0\}$, while $\{\{f_0, g_0\}, \{f_1, g_1\}\}$ again only has one component that is stabilised if and only if an even number of edges in $\{f, g\}$ is flipped. This pattern of alternation between sets with two components and sets with one component is typical of the h.f. sets constructed by the known CFI-algorithms.
Now we can observe that the objects which behave analogously to CFI-gadgets are the connected components *inside the sets, not the sets in* $\mathrm{tc}(\mu_{\{e,f,g\}})$ *themselves. For example, the sets $\{f_0, g_0\}$ and $\{\mu_{\{f,g\}}, e_0\}$ cannot be "flipped" between two states, like a CFI-gadget. Their orbit with respect to edge-flips has size four. But whenever these sets occur as elements of another set, they occur together with a counterpart from their orbit, which ensures that its connected component inside the parent set again has the "CFI-property":*

It has orbit-size two and is "flipped" if and only if an even number of elements are flipped. Note that the number of flipped elements is always the same in every member of a connected component. For example, in the component $\{\{f_0, g_0\}, \{f_1, g_1\}\}$, it is clear that f_0 is flipped iff f_1 is flipped and g_0 is flipped iff g_1 is; so $\{f_0, g_0\}$ and $\{f_1, g_1\}$ are always affected by the same number of flips, and the same is true for $\{\mu_{\{f,g\}}, e_0\}$ and $\{\widetilde{\mu}_{\{f,g\}}, e_1\}$. Therefore, it makes sense to view the connected components inside each set as analogues of CFI-vertex-gadgets, and the elements of each/any member of a component as its "incident edges", whose flips affect the "vertex-gadget".

The above example illustrates the design pattern behind all the known CFI-algorithms: The goal is always to construct a h.f. set that represents the parities of all CFI-gadgets of a given instance. Whenever a new gadget is introduced into the h.f. set, this leads to a sub-object with orbit size greater than two, but this is compressed down to two again in later stages of the construction. H.f. sets that obey this design pattern are what we call *CFI-symmetric.* They satisfy the condition that the connected components inside every set in the transitive closure behave like CFI-gadgets, as we saw in the example. Formally:

Definition 8.2.2 (CFI-symmetric components and objects)**.** *Let $\mu \in \text{HF}(\widehat{E})$, $x \in \text{tc}(\mu)$, and $\gamma \subseteq x$ be a connected component of x. Then we say that γ is* CFI-symmetric *if $|\mathbf{Orb}_{\mathbf{Aut}_{\text{CFI}}(\mathfrak{G})}(\gamma)| = 2$, and for each $\rho_F \in \mathbf{Aut}_{\text{CFI}}(\mathfrak{G})$, it holds $\rho_F(\gamma) = \gamma$ iff for each/any $y \in \gamma$, the number of flipped components of y, that is $|\{\gamma' \in \mathcal{C}(y) \mid \rho_F(\gamma') \neq \gamma'\}|$, is even.*

The set μ is CFI-symmetric *if the following two conditions are satisfied:*

1. *For each $\rho_F \in \mathbf{Aut}_{\text{CFI}}(\mathfrak{G})$, it holds $\rho_F(\mu) = \mu$ iff the number of flipped components of μ, that is, $|\{\gamma \in \mathcal{C}(\mu) \mid \rho_F(\gamma) \neq \gamma\}|$, is even.*
2. *For every $x \in \text{tc}(\mu)$, every connected component $\gamma \in \mathcal{C}(x)$ is CFI-symmetric.*

We will never deal with objects $\mu \in \text{HF}(\widehat{E})$ in which only some, but not all connected components of sets in $\text{tc}(\mu)$ are CFI-symmetric. Therefore, when we speak of "flipped components of y" in the above definition, and denote these as $\{\gamma' \in \mathcal{C}(y) \mid \rho_F(\gamma') \neq \gamma'\}$, the component $\rho_F(\gamma') \neq \gamma'$ really is the "flip" of γ', because the orbit of γ' has size exactly two. The condition $|\mathbf{Orb}_{\mathbf{Aut}_{\text{CFI}}(\mathfrak{G})}(\gamma)| = 2$ could actually be dropped because it follows inductively from the fact that γ is fixed if and only if an even number of components in each/any $y \in \gamma$ is fixed.

We still have to show that the formulation "each/any" in Definition 8.2.2 is indeed justified, as we already indicated in the example.

Lemma 8.2.3. *Let $\mu \in \text{HF}(\widehat{E})$, $x \in \text{tc}(\mu)$, and $\gamma \subseteq x$ be a connected component of x. For any two $y, y' \in \gamma$ and every $\rho_F \in \mathbf{Aut}_{\text{CFI}}(\mathfrak{G})$, it holds*

$$|\{\gamma' \in \mathcal{C}(y) \mid \rho_F(\gamma') \neq \gamma'\}| = |\{\gamma' \in \mathcal{C}(y') \mid \rho_F(\gamma') \neq \gamma'\}|.$$

Proof. Fix $\rho_F \in \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})$. Further, let $\rho \in \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})$ be an automorphism such that $\rho(y) = y'$. This exists because $y \sim_E y'$. Thus, ρ induces a bijection from $\mathcal{C}(y)$ to $\mathcal{C}(y')$, as it maps each connected component of y to a connected component of y'. We show that for each component $\gamma' \in \mathcal{C}(y)$ it holds: $\rho_F(\gamma') = \gamma'$ iff $\rho_F(\rho(\gamma')) = \rho(\gamma')$. If $\rho_F(\gamma') = \gamma'$, then we have (because $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})$ is Abelian): $\rho_F(\rho(\gamma')) = \rho(\rho_F(\gamma')) = \rho(\gamma)$. Conversely, if $\rho_F(\rho(\gamma')) = \rho(\gamma')$, then $\gamma' = \rho^{-1}(\rho(\gamma')) = \rho^{-1}(\rho_F(\rho(\gamma'))) = \rho_F(\gamma')$, where the last equality is again by commutativity. The lemma follows because $|\mathcal{C}(y)| = |\mathcal{C}(y')|$. □

Definition 8.2.4 (CFI-symmetric and super-symmetric algorithms)**.** *A* CPT*-program* Π *that decides the CFI-query on a class* $\mathcal{K}$ *of base graphs is called* CFI-symmetric *if it activates a CFI-symmetric h.f. set* $\mu \in \mathrm{HF}(\widehat{E})$ *on every input* $\mathfrak{G}^S$ *over a base graph* $G = (V, E) \in \mathcal{K}$ *(and this set* μ *is necessary for deciding the CFI-query). Similarly,* Π *is called* super-symmetric *if it necessarily activates a super-symmetric set. When* Π *is both CFI-symmetric and super-symmetric, then these two properties apply to the same object constructed by* Π.

The condition that μ is *necessary* to decide the query is not very precise but it makes sense in light of Theorem 6.2.7, which says that the activation of a h.f. set with large support is necessary. So here, we mean that if Π did not activate μ, then it would not be able to define the CFI-query, for example because μ is the only set with sufficient support activated by Π.

Now that we have established the notion of CFI-symmetric sets and algorithms, one may ask how natural this definition is and why we focus on such objects. The answer to the second question is that our construction of XOR-circuits which capture the relevant complexity parameters of sets in $\mathrm{HF}(\widehat{E})$ works best if we restrict our attention to CFI-symmetric sets. Towards the end of the chapter, we will discuss in how far it may be possible to overcome this restriction and transform *all* sets in $\mathrm{HF}(\widehat{E})$ into XOR-circuits. The other motivation for defining CFI-symmetry in this way is the observation that we made in Example 8.2.1, and which is in fact true of any of the various super-symmetric h.f. sets that play the key role in the CFI-algorithms from [40] and [91]: All these objects are built out of sets that alternate between having one connected component and having multiple components. The sets with multiple components always introduce a new vertex or edge gadget of the input graph into the object, while the sets with one component ensure symmetry: They are "flippable" just like CFI-gadgets and their two possible "states" are well-suited to track parities, which is what the CFI-query is all about. This structural pattern, that has been applied successfully in these various CFI-algorithms, is captured by our notion of CFI-symmetry. The class of CPT-algorithms that use CFI-symmetric objects to solve the CFI-query can thus be considered a fairly natural one.

The feature that makes these objects useful for deciding the CFI-query is, however, super-symmetry and not CFI-symmetry. Super-symmetry is needed in all the known algorithms because it allows to arbitrarily substitute the vertices e_0, e_1 from each edge gadget with 0 and 1 in both possible ways without producing two distinct objects (since the object is invariant under any edge flip). The super-symmetric sets in these algorithms happen to be sets of CFI-symmetric objects. Super-symmetry is achieved by constructing an object which behaves like a CFI-vertex-gadget that is incident to *every edge* of the base graph twice. That is, we can imagine the construction process of the desired super-symmetric object as a successive contraction of the input CFI-graph into a single vertex gadget that has one self-loop for each edge of the original base graph. In short, *super-symmetry* is the main algorithmically useful property of a h.f. set, and *CFI-symmetry* is a clever structural recipe for building super-symmetric sets. Informally speaking, it seems that any sensibly constructed super-symmetric object should also be CFI-symmetric: If the CFI-symmetry is dropped, the orbit-sizes just get bigger and super-symmetry seems more difficult to achieve. Actually, CFI-symmetric objects in some sense have the smallest possible "local orbit sizes": If $\mu \in \mathrm{HF}(\widehat{E})$ is CFI-symmetric, then any object $x \in \mathrm{tc}(\mu)$ with orbit size > 2 can only occur as an element of another set $y \in \mathrm{tc}(\mu)$, if it occurs together with a part of its orbit, such that the orbit size is compressed down to two again. This is in a sense smallest-possible because h.f. sets which consist only of objects (or connected components) with orbit-size one are probably rather meaningless. Therefore, for algorithms based on super-symmetric objects (which is the only paradigm we know at the moment), we speculate that CFI-symmetry is no true restriction, i.e. that the class of "super-symmetric CFI-algorithms" is a subclass of the CFI-symmetric ones.

Some properties of our constructed XOR-circuits will be particularly nice when the original object is super-symmetric. In total, our circuit construction will be most powerful for objects that are super-symmetric and CFI-symmetric at the same time. As already mentioned, this holds for all objects constructed in the known algorithms.

8.3 Symmetric XOR-circuits

An XOR-circuit is a connected directed acyclic graph $C = (V_C, E_C)$ with a unique designated root r. Its internal nodes are understood as XOR-gates and its leafs correspond to the input gates of the circuit. If $(g, h) \in E_C$, then the output of gate h is an input of gate g. Every XOR-circuit computes the Boolean XOR-function over a subset of its input bits.
Such circuits are the combinatorial objects that we will use to capture the structure of the CFI-symmetric h.f. sets in $\mathrm{HF}(\widehat{E})$. When we consider these h.f. sets, we always view them as objects over a given CFI-structure $\mathfrak{G}^S$ on some base graph $G = (V, E)$. Defining them in CPT requires to preserve the symmetries of the input structure $\mathfrak{G}^S$, so in particular, the automorphisms $\mathbf{Aut}(G)$ of the base graph. This symmetry will be reflected in the symmetry of the corresponding XOR-circuit. Therefore, we have to formalise how the automorphisms of a graph G act on XOR-circuits:

We say that an XOR-circuit C is a circuit *over a graph* $G = (V, E)$, if the input gates of C are labelled with the edges in E. More precisely, let $L \subseteq V_C$ be the leafs of C. There is an injective labelling function $\ell : L \longrightarrow E$ that relates the input gates with edges of G. To speak about the semantics of the circuit, we introduce a set of formal propositional variables $\mathcal{V}(G) := \{X_e \mid e \in E\}$. For every input gate $g \in L$, the input bit of this gate is the value of the variable $X_{\ell(g)}$.

Since every internal gate is an XOR-gate, the function computed by it is the XOR over a subset of $\mathcal{V}(G)$. For our purposes, this subset is the main interesting property of a gate, and we call it $\mathcal{X}(g)$. Formally, if $g \in L$, then $\mathcal{X}(g) := \{\ell(g)\} \subseteq E$.
If g is an internal gate, then

$$\mathcal{X}(g) := \bigtriangleup_{h \in gE_C} \mathcal{X}(h),$$

that is, the symmetric difference over the $\mathcal{X}(h)$ for all children of g. In other words, $\mathcal{X}(g) \subseteq E$ is precisely the set of edges in E such that g computes the Boolean function $\bigoplus_{e \in \mathcal{X}(g)} X_e$. The function computed by the circuit C is the XOR over $\mathcal{X}(r)$, where r is the root of C. An alternative way to think about this is to say that for any gate g, $\mathcal{X}(g)$ is the set of input bits to which the function computed by g is *sensitive*, that is, flipping a single input bit of the circuit changes the value of g if and only if the flipped edge is in $\mathcal{X}(g)$.

8.3.1 Symmetries of circuits

A circuit C over a graph G is subject to the action of the automorphism group $\mathbf{Aut}(G) \leq \mathbf{Sym}(V)$. Any $\pi \in \mathbf{Aut}(G)$ changes the labels of the input gates in L. So let $g \in L$ with $\ell(g) = e = \{u, v\} \in E$. Then $\pi(g)$ is an input gate with $\ell(\pi(g)) = \pi(e) = \{\pi(u), \pi(v)\} \in E$. This extends to subcircuits of C and to C itself, so $\pi(C)$ is just C with the input labels modified accordingly.

We say that π *extends to an automorphism* of C if there exists a bijection $\sigma : V_C \longrightarrow V_C$ that is an automorphism of the graph (V_C, E_C) and satisfies for each input gate $g \in V_C$: $\ell(\sigma(g)) = \pi(\ell(g))$. We write

$$\mathbf{Stab}_G(C) = \{\pi \in \mathbf{Aut}(G) \mid \pi \text{ extends to an automorphism of } C\} \leq \mathbf{Aut}(G),$$

and:

$$\mathbf{Orb}_G(C) = \{\pi(C) \mid \pi \in \mathbf{Aut}(G)\}.$$

Symmetric circuits have been studied as computation models capturing FPC and rank logic by Dawar together with Anderson and Wilsenach, respectively (see [6] and [42]). Our setting here is a bit different. The symmetric circuits from the literature – roughly speaking – correspond to the case where $\mathbf{Aut}(G) = \mathbf{Sym}(V)$ and $\mathbf{Stab}_G(C) = \mathbf{Sym}(V)$, so they are invariant under *all* possible permutations of V. That is also the reason why their power can be bounded in terms of fixed-point logics, which are weaker than CPT. In contrast, the circuits we consider here need not be symmetric in the sense that *every* $\pi \in \mathbf{Aut}(G)$ induces a circuit automorphism, but rather we will be interested in circuits whose orbit size is at most polynomial in $|G|$, w.r.t. the action of $\mathbf{Aut}(G)$. Also, $\mathbf{Aut}(G)$ may in general be much smaller than $\mathbf{Sym}(V)$.

Other examples for symmetric circuit lower bounds concern symmetric arithmetic circuits for the permanent [41] and determinant [43], AC^0-circuits for the parity function [96], and Boolean circuits for the multiplication of permutation matrices [70]. An interesting aspect about Rossman's lower bound for AC^0-circuits computing parity is the symmetry group he considers: Contrary to the other mentioned results, the symmetry group is in this case not a large permutation group on the input variables but a Boolean vector space which acts on the set of input literals $\{X_1, \overline{X}_1, ..., X_n, \overline{X}_n\}$ by swapping specified literals X_i with their respective negations. Such Boolean vector spaces appear also frequently in our proof as they correspond to the CFI-flips. Unfortunately, Rossman's lower bound does not seem to have a direct bearing on our problem, though, because he shows the hardness of computing parities, whereas our circuits can achieve this with a single XOR-gate.

8.3.2 The parameter fan-in dimension

The circuits that we obtain from h.f. sets over CFI-graphs $\mathfrak{G}^S$ will satisfy a certain non-standard fan-in bound, which generalises the fan-in degree of the gates. This fan-in bound originates from the orbit size of the h.f. set μ that the circuit represents. Recall that the automorphism group of $\mathfrak{G}^S$ can be decomposed into the automorphisms of the underlying graph G and the CFI-edge-flip automorphisms $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}^S)$ (see Lemma 5.2.2). The orbit of μ must be of polynomial size if μ is CPT-definable in $\mathfrak{G}^S$. This orbit size restriction with respect to $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}^S)$ will be reflected in a fan-in bound on the corresponding circuit $C(\mu)$.
This fan-in bound, however, is not actually on the number of incoming wires of each gate (as is more standard), but it is a bound on the "linear algebraic complexity of incoming information", so to say. The subsets of E form a Boolean vector space together with the symmetric difference operation. This space is isomorphic to $\mathbb{F}_2^E$.

Now with each internal gate $g \in V_C$, we can associate a Boolean matrix $M(g) \in \mathbb{F}_2^{gE_C \times E}$, that we call the *gate matrix*: The row at index $h \in gE_C$ is defined as the characteristic vector of $\mathcal{X}(h) \subseteq E$, transposed, i.e. $M(g)_{h-} = \chi(\mathcal{X}(h))^T$. Here and in what follows, we write χ for the bijection from $\mathcal{P}(E)$ to $\mathbb{F}_2^E$ that associates with each subset of E its characteristic Boolean vector. If g is an input gate, then we define $M(g) \in \mathbb{F}_2^{[1] \times E}$ as the one-row matrix whose only row is $\chi(\mathcal{X}(g)))^T = \chi(\{\ell(g)\})^T$.

Definition 8.3.1 (Fan-in dimension). *The* fan-in dimension *of a gate g is the dimension of the row-space of $M(g)$, or equivalently, $\boldsymbol{rk}(M(g))$.*
The fan-in dimension of g, restricted to the space $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}^S)$ (also called the restricted fan-in dimension*) is*

$$\dim(M(g) \cdot \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}^S)) = \dim\{M(g) \cdot \mathbf{v} \mid \mathbf{v} \in \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}^S)\}.$$

The (restricted) fan-in dimension of the circuit C is the maximum (restricted) fan-in dimension of any of the gates in C.

Thus, the fan-in dimension of a gate g is the dimension of the subspace of $\mathbb{F}_2^E$ that is spanned by the characteristic vectors $\chi(\mathcal{X}(h)) \in \mathbb{F}_2^E$, for all children h of g. One interpretation of $\mathbf{rk}(M(g))$ is that it tells us how many different patterns of incoming bits can occur at gate g: When we consider all $2^{|E|}$ possible inputs of the circuit, the number of distinct binary strings in $\{0,1\}^{gE_C}$ that can arise as the values of the children of g is $2^{\mathbf{rk}(M(g))}$.
Sometimes we will also need the restricted fan-in dimension. This describes how many different input patterns of the gate can occur if we only allow circuit input vectors $\mathbf{x} \in \mathbb{F}_2^E$ where the 1-entries in $\mathbf{x}$ form a set of cycles in the base graph G (i.e. input vectors from $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}^S)$).

These notions are unusual but as we will show in the next section, they nicely capture the orbit size of h.f. sets $x \in \mathrm{HF}(\widehat{E})$ with respect to the action of the CFI-automorphisms. Namely, the fan-in dimension of the corresponding XOR-circuit will be logarithmic in the size of the largest $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})$-orbit of any element of $\mathrm{tc}(x)$, and similarly for the restricted fan-in dimension and the size of the $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}^S)$-orbits.

In total, the benefit of the circuit-representation of h.f. objects over CFI-graphs is that this simplifies the effect of the complicated automorphism group $\mathbf{Aut}(\mathfrak{G}^S) \leq \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}) \rtimes \mathbf{Aut}(G)$: In order to show that the CFI-query is not in CPT, we would ultimately like to prove that certain h.f. objects necessarily have super-polynomial orbits w.r.t. $\mathbf{Aut}(\mathfrak{G}^S)$. By translating these objects into circuits, we can express the restrictions imposed by $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})$ in terms of fan-in dimension, and are left with the task of analysing the orbit-size w.r.t. $\mathbf{Aut}(G)$.

8.4 Constructing XOR-circuits from hereditarily finite objects

This is the main theorem that is responsible for the translation from CFI-symmetric h.f. sets to symmetric XOR-circuits:

Theorem 8.4.1. *Fix a family $(G_n)_{n\in\mathbb{N}}$ of base graphs. For every $n \in \mathbb{N}$, let $\mathfrak{G}_n^S$ be a CFI-graph over $G_n = (V_n, E_n)$ and let $\mu_n \in \mathrm{HF}(\widehat{E_n})$ be a* CFI-symmetric *h.f. set that is* CPT*-definable on input $\mathfrak{G}_n^S$ (by the same* CPT*-program for the whole family of graphs). Then for every $n \in \mathbb{N}$, there exists an XOR-circuit $C(\mu_n) = (V_C, E_C)$ over G_n which satisfies:*

1. *The size of the circuit, i.e. $|V_C|$, is polynomial in $|\mathfrak{G}_n^S|$.*
2. *The orbit-size $|\mathbf{Orb}_G(C(\mu_n))|$ of the circuit is polynomial in $|\mathfrak{G}_n^S|$.*
3. *$C(\mu_n)$ is* sensitive *to an edge $e \in E_n$ if and only if $e \in \sup_{\mathrm{CFI}}(\mu_n)$.*

4. *The* fan-in dimension *of $C(\mu)$ is $\mathcal{O}(\log(\mathbf{maxOrb}_E(\mu)))$. The fan-in dimension* restricted to the space $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}^S_n)$ *is $\mathcal{O}(\log(\mathbf{maxOrb}_{\mathrm{CFI}}(\mu)))$.*

We now provide the construction of the circuit and prove several lemmas from which it follows that the circuit has the desired properties. We fix $\mu \in \mathrm{HF}(\widehat{E})$ and denote by $C(\mu) = (V_C, E_C)$ the corresponding XOR-circuit that we are going to define. The gates of the circuit are the $\sim_E$-equivalence classes of the objects in $\mathrm{tc}(\mu)$. Recall that $\sim_E$-equivalence is the same-orbit-relation with respect to the edge-flips $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})$. Whenever we write $[x]$ for an $x \in \mathrm{tc}(\mu)$, we formally mean $[x] = \{\rho_F(x) \mid \rho_F \in \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})$ such that $\rho_F(x) \in \mathrm{tc}(\mu)\}$. The circuit $C(\mu)$ is defined as follows:

- $V_C := \mathrm{tc}(\mu)_{\sim_E} = \{[x] \mid x \in \mathrm{tc}(\mu)\}$.
- $E_C := \{([x],[y]) \mid$ there exists $y' \in [y]$ such that $y' \in x\}$.
- By definition, the leafs of $C(\mu)$ correspond to $\sim_E$-classes of atoms in $\mathrm{tc}(\mu)$. The set of atoms is $\widehat{E}$, so any leaf of C has the form $[e_0]$, for some $e \in E$. We let $\ell([e_0]) := e$.
- The root r of $C(\mu)$ is $[\mu]$.

In other words, the circuit is just the DAG $(\mathrm{tc}(\mu), \in)$, with the $\sim_E$-equivalence factored out.

First of all, we have to check that the set of edges E_C can indeed be defined in this way, i.e. that whether or not there is an E_C-edge between $[x]$ and $[y]$ is independent of the choice of the representative of $[x]$ in the definition. In the following lemma, let $\in^\mu$ denote the element relation on $\mathrm{tc}(\mu)$ within the h.f. set μ.

Lemma 8.4.2. *Let $[x],[y] \subseteq \mathrm{tc}(\mu)$ be two $\sim_E$ classes. If there exists $y' \in [y]$ such that $y' \in^\mu x$, then for every $x' \in [x]$ there is a $y' \in [y]$ such that $y' \in^\mu x'$.*

Proof. Let $y' \in [y]$ such that $y' \in^\mu x$. Now let $x' \in [x]$ be arbitrary, and let $\rho_F \in \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})$ be such that $\rho_F(x) = x'$. Then $\rho_F(y') \in^\mu \rho_F(x)$ because ρ_F is applied element-wise to x. The object $\rho_F(x) \in [x]$ is an element of $\mathrm{tc}(\mu)$ since $[x]$ denotes the $\sim_E$-class of x inside $\mathrm{tc}(\mu)$. Therefore, we also have $\rho_F(y') \in \mathrm{tc}(\mu)$, and thus $\rho_F(y') \in [y]$. This proves the lemma. □

Now we are going to show several properties of $C(\mu)$, which altogether lead to a proof of Theorem 8.4.1.

Property 2 from Theorem 8.4.1 states that the $\mathbf{Aut}(G)$-orbit of $C(\mu)$ is sufficiently small. We prove this by showing that this orbit cannot be larger than the $\mathbf{Aut}(G)$-orbit of the h.f. set μ; and for μ, we know that its orbit is polynomial in $|\mathfrak{G}^S|$, because it is CPT-definable by the assumptions of Theorem 8.4.1.

Lemma 8.4.3. *Every $\pi \in \mathbf{Stab}_G(\mu) \leq \mathbf{Sym}(V)$ extends to an automorphism of the circuit $C(\mu)$, that is:*

$$\mathbf{Stab}_G(\mu) \leq \mathbf{Stab}_G(C(\mu)).$$

Proof. Let $\pi \in \mathbf{Stab}_G(\mu) \leq \mathbf{Aut}(G)$. That is, π extends to an automorphism $\sigma : \mathrm{tc}(\mu) \longrightarrow \mathrm{tc}(\mu)$ of the DAG $(\mathrm{tc}(\mu), \in^\mu)$. We define $\sigma' : V_C \longrightarrow V_C$ by letting $\sigma'([x]) = [\sigma(x)]$. This is well-defined because $x \sim_E x'$ if and only if $\sigma(x) \sim_E \sigma(x')$ (because σ is an automorphism of μ). Now we check that σ' is an automorphism of $C(\mu)$ induced by π. Clearly, σ' is a bijection on V_C, i.e. on the set of $\sim_E$-classes of $\mathrm{tc}(\mu)$: It is surjective because σ is, and then it is already a bijection because it maps V_C to V_C. Let $[e_0] \in V_C$ be an input gate. Then $\ell([e_0]) = e$. We have $\sigma(e_0) = \pi(e)_0$. So $\sigma'([e_0]) = [\pi(e)_0]$. Hence, $\ell(\sigma'([e_0])) = \pi(\ell([e_0]))$, as desired.
Now let $([x],[y]) \in E_C$. Then there exists a $y' \in [y]$ such that $y' \in^\mu x$. Then because σ is an automorphism, it also holds $\sigma(y') \in^\mu \sigma(x)$. Therefore, $([\sigma(x)],[\sigma(y')]) \in E_C$. It holds $([\sigma(x)],[\sigma(y')]) = (\sigma'[x], \sigma'[y']) = (\sigma'[x], \sigma'[y])$, so $(\sigma'[x], \sigma'[y]) \in E_C$. In total, this means that $\pi \in \mathbf{Aut}(G)$ extends to the automorphism σ' of the circuit $C(\mu)$. □

Corollary 8.4.4.
$$|\mathbf{Orb}_G(C(\mu))| \leq |\mathbf{Orb}_G(\mu)|.$$

Proof. Follows from Lemma 8.4.3 together with the Orbit-Stabiliser Theorem, which says that $|\mathbf{Orb}_G(C(\mu))| = |\mathbf{Aut}(G)|/|\mathbf{Stab}_G(C(\mu))|$ and $|\mathbf{Orb}_G(\mu)| = |\mathbf{Aut}(G)|/|\mathbf{Stab}_G(\mu)|$. □

Next, we would like to analyse the *fan-in dimension* of $C(\mu)$, and the connection between $C(\mu)$ and $\sup_{\mathrm{CFI}}(\mu)$. The key for this is to establish a connection between the stabilisers $\mathbf{Stab}_E(x)$, for all $x \in \mathrm{tc}(\mu)$, and the kernels of the corresponding *gate matrices*. For the definition of these matrices, we refer back to Section 8.3.2. Once we have this connection, it is quite clear that the fan-in dimension of the gates essentially captures the $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})$-orbit size of the objects in $\mathrm{tc}(\mu)$.
We start with the following observation that relates the stabilisers of objects in $\mathrm{tc}(\mu)$ with the stabilisers of their elements:

Proposition 8.4.5. *For each $x \in \mathrm{tc}(\mu)$, it holds*

$$\mathbf{Stab}_E(x) = \bigcap_{y \in x} \mathbf{Stab}_E([y] \cap x) = \bigcap_{\gamma \in \mathcal{C}(x)} \mathbf{Stab}_E(\gamma).$$

Proof. Every $\rho_F \in \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})$ fixes x if and only if it fixes every connected component of x setwise (as the connected components are precisely the subsets of $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})$-orbits that are contained in x). □

Lemma 8.4.6. *For every gate $[x] \in V_C$, and its gate matrix $M[x] \in \mathbb{F}_2^{[x]E_C \times E}$, it holds:*

$$\boldsymbol{Ker}(M[x]) = \mathbf{Stab}_E(x) = \mathbf{Stab}_E(x') \text{ for every } x' \in [x].$$

For every row $M[x]_{[y]-}$, for $[y] \in [x]E_C$, it holds:

$$\boldsymbol{Ker}(M[x]_{[y]-}) = \mathbf{Stab}_E([y] \cap x) \qquad (\star)$$

Proof. The fact that $\mathbf{Stab}_E(x) = \mathbf{Stab}_E(x')$, for every $x' \in [x]$, is stated in Lemma 8.1.1. Note that $(\star)$ does not depend on the choice of a representative: It holds $\mathbf{Stab}_E([y] \cap x) = \mathbf{Stab}_E([y] \cap x')$, for every $x' \in [x]$. This is because $([y] \cap x) \sim_E ([y] \cap x')$, and so with Lemma 8.1.1, it follows that $\mathbf{Stab}_E([y] \cap x) = \mathbf{Stab}_E([y] \cap x')$.
From $(\star)$ it immediately follows that $\mathbf{Ker}(M[x]) = \mathbf{Stab}_E(x)$, due to Proposition 8.4.5 and the fact that $\mathbf{Ker}(M[x])$ is the intersection over the kernels of the rows of $M[x]$.

We now prove $(\star)$ via induction from the input gates to the root. If $[x] = [e_0]$ is an input gate, then $M[x]$ has just one row, which is $\chi(e)^T$. The kernel of $\chi(e)^T$ is the set of all vectors in $\mathbb{F}_2^E$ which are zero at index e. This is precisely $\mathbf{Stab}_E(e_0) = \mathbf{Stab}_E(e_1)$, as desired. Now suppose $[x]$ is an internal gate, i.e. x is a non-atomic h.f. set in $\mathrm{tc}(\mu)$. Each row of $M[x] \in \mathbb{F}_2^{[x]E_C \times E}$ is the characteristic vector of $\mathcal{X}[y] \subseteq E$, for a $[y] \in [x]E_C$. We have

$$\mathcal{X}[y] = \bigtriangleup_{[w] \in [y]E_C} \mathcal{X}[w].$$

In matrix-vector notation, we can write this as:

$$M[x]_{[y]-} = \chi(\mathcal{X}[y])^T = \sum_{[w] \in [y]E_C} (M[y]_{[w]-})^T = (1\ 1\ \ldots\ 1) \cdot M[y].$$

The last equality holds because the row index set of $M[y]$ is precisely $[y]E_C$. Let $\gamma \in \mathcal{C}(x)$ be the connected component such that $\gamma = [y] \cap x$. The equation above means that $\mathbf{Ker}(M[x]_{[y]-}) = \mathcal{E}_y$, where $\mathcal{E}_y$ denotes the set of all vectors in $\mathbb{F}_2^E$ whose image under $M[y]$ has even Hamming weight. Thus we have to show that $\mathcal{E}_y = \mathbf{Stab}_E(\gamma)$.
Each row $M[y]_{[w]-}$ corresponds to a connected component $\gamma' \in \mathcal{C}(y)$ with $w \in \gamma'$.
By the induction hypothesis, we have for each row $M[y]_{[w]-}$ and each $\mathbf{v} \in \mathbb{F}_2^E$ that $M[y]_{[w]-} \cdot \mathbf{v} = 1$ iff $\mathbf{v} \notin \mathbf{Stab}_E([w] \cap y)$. So $M[y] \cdot \mathbf{v}$ has even Hamming weight iff $\rho_{\chi^{-1}(\mathbf{v})} \in \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})$ flips an even number of connected components of y. This is true iff $\rho_{\chi^{-1}(\mathbf{v})}$ flips an even number of components in every $y' \in \gamma$ (due to Lemma 8.2.3). By definition of CFI-symmetry (Definition 8.2.2), this is the case iff $\mathbf{v} \in \mathbf{Stab}_E(\gamma)$, because μ is CFI-symmetric, and thus, γ is a CFI-symmetric component. In total, we have shown that $\mathbf{v} \in \mathcal{E}_y$ iff $\mathbf{v} \in \mathbf{Stab}_E(\gamma)$. This proves $(\star)$ for every row of $M[x]$. □

As a consequence of this correspondence between kernels and stabilisers, we can bound the fan-in dimension of $C(\mu)$. This proves **Property 4** from Theorem 8.4.1.

Lemma 8.4.7. *The* fan-in dimension *of* $C(\mu)$ *is* $\log(\mathbf{maxOrb}_E(\mu))$.

Proof. Let $x \in \mathrm{tc}(\mu)$. From the Orbit-Stabiliser Theorem and the fact that $|\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})| = 2^{|E|}$, it follows that

$$\mathbf{Orb}_E(x) = \frac{2^{|E|}}{|\mathbf{Stab}_E(x)|} \leq \mathbf{maxOrb}_E(\mu).$$

This means that

$$\log(\mathbf{maxOrb}_E(\mu)) \geq |E| - \dim\ \mathbf{Stab}_E(x).$$

By Lemma 8.4.6, $\mathbf{Stab}_E(x) = \mathbf{Ker}(M[x])$. With the Rank Theorem we get:

$$\mathbf{rk}(M[x]) = |E| - \dim\ \mathbf{Stab}_E(x) \leq \log(\mathbf{maxOrb}_E(\mu)).$$

Since there is an object $x \in \mathrm{tc}(\mu)$ where $\mathbf{maxOrb}_E(\mu)$ is attained, $\mathbf{rk}(M[x]) = \log(\mathbf{maxOrb}_E(\mu))$ is indeed the maximum rank of any gate matrix of $C(\mu)$. □

Lemma 8.4.8. *The* fan-in dimension *of $C(\mu)$ with respect to the space $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}^S)$ is $\log(\mathbf{maxOrb}_{\mathrm{CFI}}(\mu))$. That is, for every gate $[x]$ in $C(\mu)$, we have*

$$\dim(M[x] \cdot \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}^S)) \leq \log(\mathbf{maxOrb}_{\mathrm{CFI}}(\mu)).$$

Proof. Let $x \in \mathrm{tc}(\mu)$. With the Orbit-Stabiliser Theorem we get

$$\mathbf{Orb}_{\mathrm{CFI}}(x) = \frac{|\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}^S)|}{|\mathbf{Stab}_{\mathrm{CFI}}(x)|} = 2^{\dim \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}^S) - \dim \mathbf{Stab}_{\mathrm{CFI}}(x)} \leq \mathbf{maxOrb}_{\mathrm{CFI}}(\mu).$$

Thus,

$$\log \mathbf{maxOrb}_{\mathrm{CFI}}(\mu) \geq \dim \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}^S) - \dim \mathbf{Stab}_{\mathrm{CFI}}(x).$$

Using Lemma 8.4.6, we get $\mathbf{Stab}_{\mathrm{CFI}}(x) \subseteq \mathbf{Stab}_E(x) \subseteq \mathbf{Ker}(M[x])$. Therefore,

$$\dim(M[x] \cdot \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}^S)) \leq \dim \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}^S) - \dim \mathbf{Stab}_{\mathrm{CFI}}(x) \leq \log \mathbf{maxOrb}_{\mathrm{CFI}}(\mu).$$

□

Let us finally put all these lemmas together to prove that $C(\mu)$ has the desired properties.

Proof of Theorem 8.4.1. First of all, since μ is by assumption CPT-definable in the structure $\mathfrak{G}^S$, the size $|\mathrm{tc}(\mu)|$ and the orbit $|\mathbf{Orb}_{\mathfrak{G}^S}(\mu)|$ are polynomial in $|\mathfrak{G}^S|$. Therefore, **Property 1** from Theorem 8.4.1 clearly holds for $C(\mu)$, because $|V_C| \leq |\mathrm{tc}(\mu)|$. **Property 2** follows from the bound on $|\mathbf{Orb}_{\mathfrak{G}^S}(\mu)|$ together with Corollary 8.4.4, and the fact that $|\mathbf{Orb}_G(\mu)| \leq |\mathbf{Orb}_{\mathfrak{G}^S}(\mu)|$.
Property 4 is proven in Lemmas 8.4.7 and 8.4.8. Finally, **Property 3** can be seen as follows: Suppose $C(\mu)$ is sensitive to an edge $e \in E$. This means that $e \in \mathcal{X}(r)$, for the root $r = [\mu]$ of $C(\mu)$. This is the case iff $e \in \mathcal{X}[y]$ for an odd number of children $[y] \in [\mu]E_C$. This is the same as saying that the column $M[\mu]_{-e}$ has odd Hamming weight. By equation $(\star)$ from Lemma 8.4.6, this holds if and only if $\chi(e) \notin \mathbf{Stab}_E([y] \cap \mu)$ for an odd number of children $[y] \in [\mu]E_C$. Since μ is CFI-symmetric, by Definition 8.2.2 this is the case if and only if $\rho_e(\mu) \neq \mu$. And this holds iff $e \in \mathrm{sup}_{\mathrm{CFI}}(\mu)$ (because $\mathrm{sup}_{\mathrm{CFI}}(\mu)$ is the smallest possible CFI-support of μ). □

8.5 Applying the XOR-circuit construction to CFI-symmetric algorithms

So far, we have a translation of CFI-symmetric h.f. sets in $\mathrm{HF}(\widehat{E})$ into XOR-circuits with the properties mentioned in Theorem 8.4.1. By combining it with the support lower bound from Theorem 6.2.7, we can prove the formal version of Theorem 8.0.1, which shows that the definability of the CFI-query by means of a CFI-symmetric algorithm presupposes the existence of corresponding symmetric circuits. As a reminder, here is the support lower bound from Dawar, Richerby and Rossman once again.

Theorem 6.2.7. *Let $(G_n)_{n\in\mathbb{N}}$ be a family of base graphs and let $\boldsymbol{tw}_n$ denote the treewidth of G_n. Let $\mathfrak{G}_n^0, \mathfrak{G}_n^1$ denote the even and odd CFI-structures over G_n. Let $f(n) \leq \boldsymbol{tw}_n$ be a function such that $\mathfrak{G}_n^0$ and $\mathfrak{G}_n^1$ are $\mathcal{C}^{\boldsymbol{tw}_n}$-homogeneous for all tuples of length $\leq 2f(n)$. Then any* CPT*-program that distinguishes $\mathfrak{G}_n^0$ and $\mathfrak{G}_n^1$ for all $n \in \mathbb{N}$ must activate on input $\mathfrak{G}_n^i$ a h.f. set x whose smallest support has size at least $\Omega(f(n))$.*

Recall that a structure $\mathfrak{G}_n^S$ is $\mathcal{C}^{\mathbf{tw}_n}$*-homogeneous* if whenever two tuples $\overline{a}$ and $\overline{b}$ have the same $\mathcal{C}^{\mathbf{tw}_n}$-type in $\mathfrak{G}_n^S$, then there is an automorphism of $\mathfrak{G}_n^S$ that maps $\overline{a}$ to $\overline{b}$ (see also Definition 6.2.4). In particular, this condition is satisfied for certain ordered CFI-graphs, as stated in [40] and formally proven e.g. in [90]. In the next chapter, we show that this condition also holds for our unordered hypercube CFI-structures.

In the above theorem, the minimum support size of a h.f. set refers to its support with respect to the group $\mathbf{Aut}(\mathfrak{G}_n^S)$. However, in our circuit framework, we are dealing with $\mathrm{sup}_{\mathrm{CFI}}(\mu)$, the minimum *CFI-support* of an object μ. This is the minimum support with respect to the group of edge-flips $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})$. Thus, we have to relate these two different notions of support. The ratio between the minimum support of an object and its minimum CFI-support will in general depend on the base graph. We call this the *CFI-support gap*:

Definition 8.5.1 (CFI-support gap)**.** *Let $G = (V, E)$ be a base graph and $\mathfrak{G}^S$ a CFI-graph over it. Let $\mu \in \mathrm{HF}(\widehat{E})$. Denote by $s(\mu)$ the size of the smallest $\mathbf{Aut}(\mathfrak{G}^S)$-support of μ (while $\mathrm{sup}_{\mathrm{CFI}}(\mu)$ still denotes the smallest CFI-support).*
Then we call the ratio

$$\alpha(\mu) = \frac{s(\mu)}{|\,\mathrm{sup}_{\mathrm{CFI}}(\mu)|}$$

the CFI-support gap *of μ (with respect to $\mathfrak{G}^S$).*

According to Theorem 5.1.2, two CFI-graphs $\mathfrak{G}^S$ and $\mathfrak{G}^R$ over the same base graph G are indistinguishable in $\mathcal{C}^{\mathbf{tw}}$, where $\mathbf{tw}$ denotes the treewidth of G. Putting this and the support lower bound together with Theorem 8.4.1, we obtain the following detailed version of Theorem 8.0.1 that relates the power of CPT with the existence of symmetric XOR-circuits.

Theorem 8.5.2. *Let $(G_n = (V_n, E_n))_{n\in\mathbb{N}}$ be a sequence of base graphs. Let $\mathfrak{G}_n^S$ be a CFI-graph over G_n, let $\boldsymbol{tw}_n$ denote the treewidth of G_n. Let $f(n) \in \mathcal{O}(\boldsymbol{tw}_n)$ be a function* *such that every $\mathfrak{G}_n^S$ is $\mathcal{C}^{tw_n}$-homogeneous, for all tuples of length $\leq 2f(n)$. Let $g(n)$ be a function such that the* CFI-support-gap *for every $\mu \in \mathrm{HF}(\widehat{E}_n)$ with minimum support $s(\mu) \in \Omega(f(n))$ is bounded by $g(n)$.*

If there exists a CFI-symmetric CPT*-program Π that decides the CFI-query on all $\mathfrak{G}_n^S$, then for every $G_n = (V_n, E_n)$, there exists an XOR-circuit C_n over G_n that satisfies the following "instantiated properties" from Theorem 8.4.1:*

1. *The number of gates in C_n is polynomial in $|\mathfrak{G}_n^S|$.*
2. *The orbit-size $|\mathbf{Orb}_{G_n}(C_n)|$ of the circuit is polynomial in $|\mathfrak{G}_n^S|$.*
3. *C_n is* sensitive *to $\Omega(f(n)/g(n))$ many edges in E_n.*
4. *The* fan-in dimension *of C_n,* restricted to the space $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}_n^S)$, *is $\mathcal{O}(\log |\mathfrak{G}_n^S|)$.*
5. *If the program Π is* super-symmetric *in addition to being CFI-symmetric, or if the base graph G_n decomposes into at most $\mathcal{O}(\sqrt{\log |\mathfrak{G}^S|})$ many components when any $f(n)/g(n)$ many edges are removed, then also the (unrestricted)* fan-in dimension *of C_n is $\mathcal{O}(\log |\mathfrak{G}_n^S|)$.*

Proof. Assume such a CPT-program Π exists. Let $\mu_n \in \mathrm{HF}(\widehat{E}_n)$ denote the CFI-symmetric h.f. set with large support that Π activates on input $\mathfrak{G}_n^S$. Then by Theorem 6.2.7, the smallest $\mathbf{Aut}(\mathfrak{G}_n^S)$-support of the object μ_n has size $\Omega(f(n))$. Since the CFI-support gap of μ_n in $\mathfrak{G}_n^S$ is at most $g(n)$, the size of the smallest CFI-support of μ_n is at least: $\sup_{\mathrm{CFI}}(\mu_n) \in \Omega(f(n)/g(n))$. Theorem 8.4.1 applied to μ_n yields the XOR-circuit C_n. **Property 3** from Theorem 8.4.1 in combination with the bound $\sup_{\mathrm{CFI}}(\mu_n) \in \Omega(f(n)/g(n))$ means that C_n is sensitive to $\Omega(f(n)/g(n))$ many edges in E_n. **Property 4** from Theorem 8.4.1 bounds the fan-in dimension and the restricted fan-in dimension in terms of $\log(\mathbf{maxOrb}_E(\mu_n))$ and $\log(\mathbf{maxOrb}_{\mathrm{CFI}}((\mu_n))$, respectively. Lemma 8.1.2 states that $\mathbf{maxOrb}_E(\mu_n) \leq |\mathbf{Orb}_E(\mu_n)| \cdot |\mathrm{tc}(\mu_n)|$ and $\mathbf{maxOrb}_{\mathrm{CFI}}(\mu_n) \leq |\mathbf{Orb}_{\mathrm{CFI}}(\mu_n)| \cdot |\mathrm{tc}(\mu_n)|$. Because μ_n is defined by the CPT-program Π on input $\mathfrak{G}_n^S$, both $|\mathrm{tc}(\mu_n)|$ and $|\mathbf{Orb}_{\mathrm{CFI}}(\mu_n)|$ are polynomially bounded in $|\mathfrak{G}_n^S|$ (the orbit is bounded because $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}^S)$ is a subgroup of $\mathbf{Aut}(\mathfrak{G}_n^S)$). This yields a polynomial bound on $\mathbf{maxOrb}_{\mathrm{CFI}}(\mu_n)$. Together with the $\log(\mathbf{maxOrb}_{\mathrm{CFI}}((\mu_n))$-bound on the restricted fan-in dimension, this gives us **Property 4** from this theorem.
Property 5 follows then with Lemma 8.1.4 if Π is super-symmetric, and with Corollary 8.1.9 in case that the base graph G_n splits into a sufficiently bounded number of components when the edges in $\sup_{\mathrm{CFI}}(\mu_n)$ are removed from it. □

Hence, if we can find a class of base graphs with suitable treewidth, homogeneity, and CFI-support gap, for which we can prove that no circuit family can satisfy all of the above properties simultaneously, then we have essentially separated the class of all CFI-symmetric CPT-programs from PTIME. We suspect that the theorem will be easier to use if we additionally restrict ourselves to super-symmetric algorithms or if our chosen family of base graphs is highly connected. In these cases, the logarithmic fan-in restriction

on the circuits applies to the *fan-in dimension*, which is admittedly not a natural circuit parameter but at least easy to describe in linear algebraic terms – it is just a matrix rank. By contrast, the *fan-in dimension restricted to* $\mathbf{Aut}_{\text{CFI}}(\mathfrak{G}_n^S)$ is the dimension of a certain linear image of a Boolean vector space that somehow depends on the cycles in the base graph – a concept that is hard to get a handle on.
In Chapter 9 we will apply the theorem to the class of hypercubes as base graphs and show that if we sharpen **Property 2** by restricting the orbit size $|\mathbf{Orb}_{G_n}(C_n)|$ down to one and replace the logarithmic bound on the fan-in dimension (**Property 5**) by a logarithmic bound on the "orbit-wise" fan-in and fan-out degrees of the gates, then indeed, such circuits do not exist. Before we move on to that next chapter, though, we present a version of Theorem 8.4.1 that is unfortunately much more complicated to prove. Namely, we would like to get rid of the restriction to *CFI-symmetric* objects and understand *any* h.f. set over a CFI-structure as an XOR-circuit.

8.6 Extending the circuit construction to non-CFI-symmetric sets

So far, we have shown that *CFI-symmetric* h.f. sets over CFI-structures $\mathfrak{G}^S$ can be quite easily transformed into XOR-circuits by factoring out the orbits under the edge-flip-group $\mathbf{Aut}_{\text{CFI}}(\mathfrak{G})$. Importantly, this construction automatically translates the relevant properties of the h.f. set, such as support size and symmetry, into more or less natural circuit-properties. As a consequence, we can – in principle – limit the power of *CFI-symmetric* algorithms for the CFI-query by proving non-existence results for certain families of polynomial size symmetric XOR-circuits. Even though all currently known choiceless algorithms for the CFI-query *are* CFI-symmetric, and it is not clear that non-CFI-symmetric algorithms are really more powerful, it would be much nicer if the circuit-translation were so general that it could be used to separate all of CPT from P, and not only the CFI-symmetric algorithms. In this subsection we explore to what extent the circuit construction can be generalised in that direction. We will present a modification of the circuit construction above, that uses additional gadgets, works without the restriction to CFI-symmetric sets, and has almost all properties from Theorem 8.4.1. By "almost all" we mean that the additional gadgets we have to introduce in the circuit are of unknown size. Hence, we cannot be sure that the constructed circuit is always of polynomial size. However, we can formulate a condition on the h.f. sets which generalises that of CFI-symmetry and guarantees polynomial size of the circuit. This condition concerns Boolean vector spaces with a permutation group acting on the index set. If certain subspaces of $\mathbb{F}_2^E$ that occur as stabilisers of the connected components of a set possess a basis that is (almost) invariant under the permutation group (which will be a subgroup of $\mathbf{Aut}(G)$), then the circuit we are going to construct from μ has polynomial size. Here is the result of this section:

Theorem 8.6.1. *Fix a family $(G_n)_{n\in\mathbb{N}}$ of base graphs. For every $n \in \mathbb{N}$, let $\mathfrak{G}_n^S$ be a CFI-graph over $G_n = (V_n, E_n)$ and let $\mu_n \in \text{HF}(\widehat{E}_n)$ be a h.f. set that is* CPT*-definable* *on input $\mathfrak{G}_n^S$ (by the same* CPT*-program for the whole family of graphs). Then for every $n \in \mathbb{N}$ there exists an XOR-circuit $\widehat{C}(\mu_n) = (V_C, E_C)$ over G_n which satisfies:*

1. *The orbit-size $|\mathbf{Orb}_G(\widehat{C}(\mu_n))|$ of the circuit is polynomial in $|\mathfrak{G}_n^S|$.*
2. *$\widehat{C}(\mu_n)$ is* sensitive *to at least $\frac{|\sup_{\mathrm{CFI}}(\mu)|}{\log(\mathbf{maxOrb}_E(\mu))}$ many edges in E.*
3. *The* fan-in dimension *of $\widehat{C}(\mu)$ is $\mathcal{O}(\log(\mathbf{maxOrb}_E(\mu)))$.*
4. *If for every $x, y \in \mathrm{tc}(\mu)$ such that $[y] \cap x \neq \emptyset$, the space $\mathbf{Stab}_E([y] \cap x)$ has a* symmetric basis *(see Definition 8.6.25), then the size $|V_C|$ is polynomial in $|\mathfrak{G}^S|$.*

This theorem differs from Theorem 8.4.1 for CFI-symmetric objects in two aspects. Firstly, the circuit $\widehat{C}(\mu)$ is not necessarily sensitive to all edges in $\sup_{\mathrm{CFI}}(\mu)$ but only to a logarithmic fraction of them. Secondly, we have no guarantees for the size of the circuit unless all spaces $\mathbf{Stab}_E([y] \cap x)$ admit a symmetric basis; we will introduce this concept formally in Section 8.6.3. It should be noted that this fourth property mentioned in the theorem is – as far as we know – not an "if and only if". It may be that $\widehat{C}(\mu)$ has polynomial size even when the symmetric basis condition is not satisfied for μ.

As a consequence, we have the following version of Theorem 8.5.2 for non-CFI-symmetric CPT-programs that decide the CFI-query. The condition that the h.f. set with large support which is used to decide the CFI-query is CFI-symmetric is weakened to the symmetric basis condition. As we show later, in Lemma 8.6.28, every CFI-symmetric set also has a symmetric basis, and there are also simple examples of non-CFI-symmetric sets with a symmetric basis (see Example 8.6.29). Thus, the symmetric basis condition is indeed a strict generalisation of CFI-symmetry.

Theorem 8.6.2. *Let $(G_n = (V_n, E_n))_{n \in \mathbb{N}}$ be a sequence of base graphs. Let $\mathfrak{G}_n^S$ be a CFI-graph over G_n, let $\boldsymbol{tw}_n$ denote the treewidth of G_n, and let $f(n) \leq \boldsymbol{tw}_n$ be a function such that $\mathfrak{G}_n^S$ is $C^{\boldsymbol{tw}_n}$-homogeneous for all tuples of length $\leq 2f(n)$.*
Let $g(n)$ be a function such that the CFI-support-gap *for every $\mu \in \mathrm{HF}(\widehat{E}_n)$ with minimum support $s(\mu) \in \Omega(f(n))$ is bounded by $g(n)$.*

Let Π be a CPT*-program that decides the CFI-query on all $\mathfrak{G}_n^S$ using a h.f. set $\mu_n \in \mathfrak{G}_n^S$ with sufficient support such that $\mathbf{Stab}_E([y] \cap x)$ has a* symmetric basis *according to Definition 8.6.25, for all $x, y \in \mathrm{tc}(\mu_n)$ with $[y] \cap x \neq \emptyset$.*
Assume additionally that Π is super-symmetric *or that the base graph G_n decomposes into at most $\mathcal{O}(\sqrt{\log |\mathfrak{G}^S|})$ many components when any $f(n)/g(n)$ many edges are removed. Then for every $n \in \mathbb{N}$ there exists an XOR-circuit C_n over the edges E_n that satisfies the following "instantiated properties" from Theorem 8.6.1:*

1. *The number of gates in C_n is polynomial in $|\mathfrak{G}_n^S|$.*
2. *The orbit-size $|\mathbf{Orb}_{G_n}(C_n)|$ of the circuit is polynomial in $|\mathfrak{G}_n^S|$.*
3. *C_n is* sensitive *to $\Omega(f(n)/(g(n) \cdot \log |\mathfrak{G}_n^S|))$ many edges in E_n.*

4. The fan-in dimension *of C_n is $\mathcal{O}(\log |\mathfrak{G}_n^S|)$.*

We omit the proof of this theorem because it follows from Theorem 8.6.1 in the same way as Theorem 8.5.2 follows from Theorem 8.4.1.

Now let us start with the proof of Theorem 8.6.1, which spans the rest of the section. It should be noted that Theorem 8.4.1 is actually a special case of this, so we could have omitted the circuit construction for CFI-symmetric objects; however, the more general construction that we present now is not as natural as the one for CFI-symmetric objects and much harder to describe.
Fix again a base graph $G = (V, E)$, a CFI-graph $\mathfrak{G}^S$ over it, and an object $\mu \in \text{HF}(\widehat{E})$. This time, μ need not be CFI-symmetric. In the previous subsection, we wrote $C(\mu)$ for the circuit obtained by factoring out the $\sim_E$ classes in $\text{tc}(\mu)$. Now, we denote the constructed circuit by $\widehat{C}(\mu)$. Before we explain the construction, let us look at why $C(\mu)$ is not "the circuit we want" if μ is not CFI-symmetric. The only place where CFI-symmetry was required in the previous section is in the proof of Lemma 8.4.6, which relates the kernels of the gate matrices with the vector spaces $\mathbf{Stab}_E(x)$. This relationship is crucial because it leads to the connection between $\text{sup}_{\text{CFI}}(\mu)$ and the sensitivity of $C(\mu)$ to its input bits, and is also necessary to get a bound on the fan-in dimension of $C(\mu)$. Without such a bound, the construction would not be interesting because without fan-in restrictions, there always exist small symmetric XOR-circuits. Hence, we would like to ensure that the statement of Lemma 8.4.6 still holds for $\widehat{C}(\mu)$, even if μ is not CFI-symmetric. Now take a look at the inductive proof of Lemma 8.4.6 again. The key in this induction is that for any gate matrix $M[x]$ and any child $[y]$ of $[x]$, the row $M[x]_{[y]-}$ can be written as the product of another matrix and the child-gate-matrix $M[y]$: $M[x]_{[y]-} = (1\ 1\ ...\ 1) \cdot M[y]$. This equation holds because of the CFI-symmetry of μ. Now in the general case, a similar equation will hold, namely: $M[x]_{[y]-} = N \cdot M[y]$, for some matrix N that has to be chosen depending on $\mathbf{Stab}_E([y] \cap x)$. So our plan for this section is as follows: We will first of all define these N-matrices, that essentially "repair" the proof of Lemma 8.4.6 in the non-CFI-symmetric case. Based on these matrices and on $C(\mu)$, we will construct $\widehat{C}(\mu)$ by introducing gadgets that simulate the effect of the chosen N-matrices. Then this circuit will be exactly such that the proof of Lemma 8.4.6 goes through again, even if μ is not CFI-symmetric. In a sense, we can view the gadgets as corrections for local violations of CFI-symmetry.

8.6.1 Definition of the matrices

Actually, we will not only define the said N-matrices, but also, for every $[x] \in \{[x] \mid x \in \text{tc}(\mu)\}$, a Boolean matrix $M[x]$. We keep the notation $M[x]$ from the previous subsection, even though, strictly speaking, $M[x]$ will not be the gate matrix of any gate in $\widehat{C}(\mu)$; rather, it will be a matrix that satisfies $\mathbf{Ker}(M[x]) = \mathbf{Stab}_E(x)$, and it will serve as a kind of construction specification for a gadget in $\widehat{C}(\mu)$. The $M[x]$-matrices that we are going to define depend on the M- and N-matrices of the connected components of x in the object μ. Therefore, the construction of these matrices proceeds inductively from the atoms of μ to the more deeply nested sets. At this point, our main objective is to build the matrices in such a way that their kernels correspond to the stabiliser spaces of the sets they belong to. Furthermore, the matrices should satisfy certain symmetry requirements with respect to the action of $\mathbf{Stab}_G(\mu) \leq \mathbf{Aut}(G)$. Only after the construction of the matrices, we will use them to construct the circuit $\widehat{C}(\mu)$ in such a way that the gate matrices of $\widehat{C}(\mu)$ are as desired. So the procedure in this subsection works the other way round as in the previous one, where the circuit came first, and we then analysed its gate matrices. Now we are specifying the matrices first, and then build the circuit so that the statement of Lemma 8.4.6 holds for $\widehat{C}(\mu)$ by construction. Along with the matrices, we will provide certain group homomorphisms that ensure symmetry. To speak about symmetry of matrices, we introduce notation to express the effect of row- and column-permutations:

Let $M \in \mathbb{F}_2^{I \times J}$ be any Boolean matrix, and $\sigma : I \longrightarrow I', \pi : J \longrightarrow J'$ be bijections. Then $(\sigma, \pi)(M) \in \mathbb{F}_2^{I' \times J'}$ is the matrix with $(\sigma, \pi)(M)_{\sigma i, \pi j} = M_{i,j}$ for each $(i, j) \in I \times J$. In particular, if $\sigma \in \mathbf{Sym}(I), \pi \in \mathbf{Sym}(J)$, then $(\sigma, \pi)(M)$ is the matrix that arises from M when the respective row- and column-permutations are applied.

Proposition 8.6.3. *Let $\mathbf{v}, \mathbf{w} \in \mathbb{F}_2^I$ be two Boolean vectors and $\pi \in \mathbf{Sym}(I)$ a permutation of its entries. Then*

$$\mathbf{v}^T \cdot \mathbf{w} = \pi(\mathbf{v}^T) \cdot \pi(\mathbf{w}).$$

Therefore, if $M \in \mathbb{F}_2^{I \times J}$ is a matrix and $\mathbf{w} \in \mathbb{F}_2^J$ is a vector, then

$$\sigma(M \cdot \mathbf{w}) = (\sigma, \pi)M \cdot \pi(\mathbf{w}),$$

for every $\sigma \in \mathbf{Sym}(I), \pi \in \mathbf{Sym}(J)$.

This proposition follows immediately from the definition of the scalar product of vectors because $\mathbf{v}^T \cdot \mathbf{w}$ and $\pi(\mathbf{v}^T) \cdot \pi(\mathbf{w})$ are the same sum of products, just summed in a different order. The statement about matrix-vector-multiplication then follows because this is just the scalar product of every row vector with $\mathbf{w}$. The proposition will sometimes be used without explicit reference in this section.

In the rest of this section, we will often speak about the following orbits and stabilisers. They differ from the ones from the previous section in so far as they concern the subgroup $\mathbf{Stab}_G(\mu) \leq \mathbf{Aut}(G)$, instead of $\mathbf{Aut}(G)$ itself. Thus, we override the notation from the previous section. Let $x \in \mathrm{tc}(\mu)$.

$$\mathbf{Orb}_G([x]) := \{\pi([x]) \mid \pi \in \mathbf{Stab}_G(\mu) \leq \mathbf{Aut}(G)\}.$$
$$\mathbf{Stab}_G([x]) := \{\pi \in \mathbf{Stab}_G(\mu) \mid \pi([x]) = [x]\}.$$

Similarly, for $x, y \in \mathrm{tc}(\mu)$, $\mathbf{Stab}_G([y] \cap x)$ refers to the stabiliser of the set $[y] \cap x$ in the group $\mathbf{Stab}_G(\mu)$.

Any orbit of a $\sim$-class is a set of $\sim$-equivalence classes:

Lemma 8.6.4. *For any $\sim_E$-class $[x] \subseteq \mathrm{tc}(\mu)$, and any $\pi \in \mathbf{Stab}_G(\mu)$, $\pi([x])$ is also a $\sim_E$-class of* $\mathrm{tc}(\mu)$.

Proof. By Lemma 5.2.1, for any $\pi \in \mathbf{Aut}(G)$, and any $\rho_F \in \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})$ it holds $\rho_{\pi F} = \pi \circ \rho_F \circ \pi^{-1}$. Let $x', x'' \in [x]$ and let $\rho_F \in \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})$ be such that $\rho_F(x') = x''$. Then $\rho_{\pi F}(\pi x') = \pi x''$ by the above equation. Thus, $\pi(x') \sim_E \pi(x'')$. Similarly one can show that if $x' \not\sim_E x''$, then $\pi(x') \not\sim_E \pi(x'')$. Therefore, $\pi([x])$ is again a $\sim_E$-class. The fact that $\pi([x]) \subseteq \mathrm{tc}(\mu)$ follows because π extends to an automorphism of the h.f. set μ. □

Corollary 8.6.5. *Let $x, y \in \mathrm{tc}(\mu)$ such that $[y] \cap x \neq \emptyset$. Then*

$$\mathbf{Stab}_G([y] \cap x) \leq \mathbf{Stab}_G([y]).$$

Proof. The group $\mathbf{Stab}_G([y] \cap x)$ maps the set $[y] \cap x \subseteq [y]$ to itself, so it does not move this subset of the $\sim$-class $[y]$ into another $\sim$-class. Then by Lemma 8.6.4, it must map the whole class $[y]$ to itself because the image of $[y]$ must again be a $\sim$-class. □

Lemma 8.6.6. *For any $x, x', y \in \mathrm{tc}(\mu)$ such that $x \sim_E x'$ and $[y] \cap x \neq \emptyset$, it holds*

$$\mathbf{Stab}_E([y] \cap x) = \mathbf{Stab}_E([y] \cap x').$$

Proof. The sets $[y] \cap x$ and $[y] \cap x'$ are related via an automorphism in $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})$. Therefore, Lemma 8.1.1 applied to the set $[y] \cap x$ yields the desired statement. □

Now we come to the inductive definition of the aforementioned M- and N-matrices. Here is the precise list of objects that we are going to define:

(a) For every $[x] \in \{[x] \mid x \in \mathrm{tc}(\mu)\}$:
   a) An index-set $I_{[x]}$.
   b) A matrix $M[x] \in \mathbb{F}_2^{I_{[x]} \times E}$ with the property that $\mathbf{Ker}(M[x]) = \mathbf{Stab}_E(x)$ (note that $\mathbf{Stab}_E(x) = \mathbf{Stab}_E(x')$ for every $x' \in [x]$ by Lemma 8.1.1).

   Let $\mathcal{C}[x] := \{[y] \mid [y] \cap x \neq \emptyset\}$ (note that this does not depend on the representative of $[x]$ – see Lemma 8.4.2) denote the $\sim$-classes of the connected components of x.

(b) For every $[x] \in \{[x] \mid x \in \mathrm{tc}(\mu)\}$, where x is a set, and every $[y] \in \mathcal{C}[x]$:
   a) An index-set $J_{[x][y]}$.
   b) A matrix $N[x][y] \in \mathbb{F}_2^{J_{[x][y]} \times I_{[y]}}$ with the property that $\mathbf{Ker}(N[x][y] \cdot M[y]) = \mathbf{Stab}_E([y] \cap x)$ (by Lemma 8.6.6, $\mathbf{Stab}_E([y] \cap x)$ is independent of the choice of representative of $[x]$).

(c) For every orbit $\Omega_{[x]} := \mathbf{Orb}_G([x])$, let $I_{\Omega_{[x]}} := \biguplus_{[x'] \in \Omega_{[x]}} I_{[x']}$. For every $\Omega_{[x]}$, we provide a group homomorphism $g_{[x]} : \mathbf{Stab}_G(\mu) \longrightarrow \mathbf{Sym}(I_{\Omega_{[x]}})$ such that for each $[x'] \in \Omega_{[x]}$ and each $\pi \in \mathbf{Stab}_G(\mu)$, it holds $g_{[x]}(\pi)(I_{[x']}) = I_{\pi[x']}$. Furthermore, $M\pi[x'] = (g_{[x]}, \pi)M[x']$ for each $[x'] \in \Omega_{[x]}$ and $\pi \in \mathbf{Stab}_G(\mu)$.

(d) For every orbit $\Omega_{[x]}$, let

$$J_{\Omega_{[x]}} := \biguplus_{\substack{[x'] \in \Omega_{[x]} \\ [y'] \in \mathcal{C}[x']}} J_{[x'][y']}.$$

For every $\Omega_{[x]}$, we provide a group homomorphism $h_{[x]} : \mathbf{Stab}_G(\mu) \longrightarrow \mathbf{Sym}(J_{\Omega_{[x]}})$ such that for each $[x'] \in \Omega_{[x]}$ and each $[y'] \in \mathcal{C}[x']$, it holds $h_{[x]}(\pi)(J_{[x'][y']}) = J_{\pi[x']\pi[y']}$. Furthermore, $N\pi[x']\pi[y'] = (h_{[x]}(\pi), g_{[x]}(\pi))N[x'][y']$ for each $[x'] \in \Omega_{[x]}$, $[y'] \in \mathcal{C}[x']$, and $\pi \in \mathbf{Stab}_G(\mu)$.

The role of the group homomorphisms is to ensure – when we build the circuit from these matrices – that every $\pi \in \mathbf{Stab}_G(\mu)$ that acts on the input gates indeed extends to an automorphism of $\widehat{C}(\mu)$. Before we actually construct anything, we have to verify that it is indeed possible to satisfy the symmetry conditions witnessed by the group homomorphisms and the conditions on the kernels of the matrices simultaneously. In other words, we have to show that the stabiliser spaces, which are supposed to be equal to the respective kernels, are mapped to each other by the permutations in $\mathbf{Stab}_G(\mu)$:

Lemma 8.6.7. *Let $x \in \mathrm{tc}(\mu)$ and $\pi \in \mathbf{Stab}_G(\mu)$. Then*

$$\mathbf{Stab}_E(\pi x) = \pi(\mathbf{Stab}_E(x)) = \{\pi\mathbf{v} \mid \mathbf{v} \in \mathbf{Stab}_E(x)\},$$

where $\pi \in \mathbf{Sym}(E)$ acts on vectors in $\mathbb{F}_2^E$ by permuting their entries. Furthermore, for every $[y] \in \mathcal{C}[x]$, we have

$$\mathbf{Stab}_E(\pi[y] \cap \pi x) = \pi(\mathbf{Stab}_E([y] \cap x)).$$

Proof. By Lemma 5.2.1, for any $\pi \in \mathbf{Aut}(G)$, and any $\rho_F \in \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})$, it holds $\rho_{\pi F} = \pi \circ \rho_F \circ \pi^{-1}$. Thus, $\rho_{\pi F} \in \mathbf{Stab}_E(\pi x)$ if and only if $(\pi \circ \rho_F \circ \pi^{-1})(\pi x) = \pi x$. This holds if and only if $(\pi \circ \rho_F)(x) = \pi x$, which is the case iff $\rho_F(x) = x$. This proves the first part of the lemma since $\chi(\pi F) = \pi(\chi(F))$. The second part can be shown in the same way because $\pi[y] \cap \pi x = \pi([y] \cap x)$. This last equation holds since the action of π on $\mathrm{HF}(\widehat{E})$ is a bijection from $\mathrm{HF}(\widehat{E})$ to itself, so $\pi[y] \cap \pi x \subseteq \pi([y] \cap x)$ (this would not necessarily be true if π were not injective on $\mathrm{HF}(\widehat{E})$). □

Inductive construction

Base case:
Let $x = e_i$, for $e \in E$ and $i \in \{0, 1\}$, be an atom in $\mathrm{tc}(\mu)$. Then we set

$$M[x] := \chi(e)^T.$$

Formally, we define the row index set as $I_{[x]} := \{[x]\}$, but any singleton set that is distinct from all other index sets will do.
Now for every orbit $\Omega_{[x]}$, where x is an atom in $\mathrm{tc}(\mu)$, we define the homomorphism $g_{[x]} : \mathbf{Stab}_G(\mu) \longrightarrow \mathbf{Sym}(I_{\Omega_{[x]}})$ by letting $g_{[x]}(\pi)([x']) := [\pi x']$ for every $\pi \in \mathbf{Stab}_G(\mu)$, $[x'] \in I_{\Omega_{[x]}}$ (note that by definition of the index-sets $I_{[x]}$, $I_{\Omega_{[x]}}$ is equal to the orbit $\Omega_{[x]}$).

Inductive step:
We deal with the items from the above list in the order (b), (d), (a), (c). Let $x \in \mathrm{tc}(\mu)$ be a non-atomic object, that is, a set. Assume that for every $[y] \in \mathcal{C}[x]$ and every $[y] \in \mathcal{C}[x']$, for every $[x'] \in \Omega_{[x]}$, the respective matrix $M[y]$ with index set $I_{[y]}$ has been constructed. Thus we also assume that for any such $[y]$, the homomorphism $g_{[y]}$ corresponding to $\Omega_{[y]}$ has been defined. We fix a y such that $[y] \in \mathcal{C}[x]$. For this fixed pair $([x],[y])$ we will now construct the matrix $N[x][y]$. Then we will close it under the action of $\mathbf{Stab}_G(\mu)$. That is, given this matrix $N[x][y]$, we will symmetrically define $N[x'][y']$ for all $[y'] \in \Omega_{[y]}$, and all $[x'] \in \Omega_{[x]}$ such that $[y'] \in \mathcal{C}[x']$.
After this, there may still exist some components $[y'] \in \mathcal{C}[x]$ for which $N[x][y']$ has not been defined. In that case, we fix such a $[y'] \in \mathcal{C}[x]$, define the corresponding matrix $N[x][y']$ explicitly, and define the matrices for all $\mathbf{Stab}_G(\mu)$-images of $[x]$ and $[y']$ symmetrically, and so on. Hence, we first have to describe how to define the respective initial matrix from which we obtain the other ones by symmetry.

Definition of the N-matrices So let $y \in \mathrm{tc}(\mu)$ be such that $[y] \in \mathcal{C}[x]$. We assume that $M[y]$, $I_{[y]}$ and $g_{[y]}$ have been constructed. The matrix $N[x][y]$ is defined as the smallest Boolean matrix that satisfies the following two conditions:

(i) $\mathbf{Ker}(N[x][y]) \cap \mathbf{Im}(M[y]) = M[y] \cdot \mathbf{Stab}_E([y] \cap x) = \{(M[y] \cdot \mathbf{v}) \mid \mathbf{v} \in \mathbf{Stab}_E([y] \cap x)\}$. Recall that for matrix $M \in \mathbb{F}_2^{I \times J}$, $\mathbf{Im}(M)$ denotes the space that is the image of $\mathbb{F}_2^J$ under M.

(ii) There exists a homomorphism h from $g_{[y]}(\mathbf{Stab}_G([y] \cap x)) \leq \mathbf{Sym}(I_{[y]})$ into the symmetric group on the row index set of $N[x][y]$ such that for every $\sigma \in g_{[y]}(\mathbf{Stab}_G([y] \cap x))$, it holds $(h(\sigma), \sigma)(N[x][y]) = N[x][y]$.

In the second property, we abused notation and wrote $g_{[y]}(\mathbf{Stab}_G([y] \cap x))$ for a subgroup of $\mathbf{Sym}(I_{[y]})$, even though $g_{[y]}(\mathbf{Stab}_G([y] \cap x))$ is formally a subgroup of $\mathbf{Sym}(I_{\Omega_{[y]}})$. However, we know from Corollary 8.6.5 that $\mathbf{Stab}_G([y] \cap x) \leq \mathbf{Stab}_G([y])$, so $g_{[y]}(\mathbf{Stab}_G([y] \cap x))$ indeed maps the row index set of $M[y]$, that is, $I_{[y]}$, to itself (see property (c) of $g_{[y]}$ that holds by the induction hypothesis).

By "smallest" matrix we mean one that satisfies (a) and (b) and has the least number of rows. If there are multiple such matrices with the same minimal number of rows, we choose an arbitrary one of them for $N[x][y]$.

Let m be the number of rows of $N[x][y]$. We define the row index set $J_{[x][y]}$ of $N[x][y]$ as an m-element set that is disjoint from all other index sets constructed so far. Formally, this can be achieved by letting $J_{[x][y]} := \{(i, [x], [y]) \mid i \in [m]\}$.
We have to show that there always exists a matrix that satisfies (a) and (b). A matrix satisfying (a) can be found with methods from linear algebra:

Lemma 8.6.8. *Let $\Gamma \leq \Delta \leq \mathbb{F}_2^I$ be Boolean vector spaces. Let d be the dimension of Γ and $k = \dim \Delta - d$ be the codimension of Γ in Δ. There exists a matrix $N \in \mathbb{F}_2^{[k] \times I}$ such that $\boldsymbol{Ker}(N) \cap \Delta = \Gamma$.*

Proof. Each of the k rows of N can be obtained as the solution to a linear equation system. For $i \in [k]$, $j \in I$, let $n_{ij} := N_{ij}$ denote the sought entry in row i and column j. Fix a basis $\mathcal{B}_\Gamma$ of Γ, and an extension of that basis $\mathcal{B} \supseteq \mathcal{B}_\Gamma$ such that $\mathcal{B} = \mathcal{B}_\Gamma \uplus \{\mathbf{w}_1, ..., \mathbf{w}_k\}$ is a basis of Δ. For each $i \in [k]$, we define an equation system $A_i \cdot \mathbf{x} = \mathbf{b}_i$ whose unique solution vector is the desired row $(n_{i1}, n_{i2}, ..., n_{i|I|})$ of N. The system has $\dim \Delta$ many equations, where each equation is associated with a basis vector in $\mathcal{B}$. For every basis vector $\mathbf{v} \in \mathcal{B} \setminus \{\mathbf{w}_i\}$, we have the equation

$$\sum_{j \in I} \mathbf{v}(j) \cdot \mathbf{x}(j) = 0$$

in the system $A_i \cdot \mathbf{x} = \mathbf{b}_i$. For the basis vector $\mathbf{w}_i \in \mathcal{B}$, we have the equation

$$\sum_{j \in I} \mathbf{w}_i(j) \cdot \mathbf{x}(j) = 1$$

in $A_i \cdot \mathbf{x} = \mathbf{b}_i$. In this way, we define k equation systems, one for each $i \in [k]$. In fact, the coefficient matrix A_i is the same for all of them. Its rows are the vectors in $\mathcal{B}$ (transposed). The vector $\mathbf{b}_i$ has a 1-entry in the row containing $\mathbf{w}_i$, and is zero otherwise. The rank and the number of rows of every A_i is $\dim \Delta$ because $\mathcal{B}$ is a basis of Δ. Hence, each of the equation systems has a unique solution. If we define each entry n_{ij} of N to be the j-th entry of the solution vector to $A_i \cdot \mathbf{x} = \mathbf{b}_i$, then indeed, $\mathbf{Ker}(N) \cap \Delta = \Gamma$, by definition of the equation systems. □

This shows that a matrix satisfying condition (a) always exists. The matrix can be closed under the action of $g_{[y]}(\mathbf{Stab}_G([y] \cap x))$ so that it also satisfies condition (b). This requires that the vector space that we want as the kernel of $N[x][y]$ is invariant under that permutation group:

Lemma 8.6.9. *The space $M[y] \cdot \mathbf{Stab}_E([y] \cap x) \leq \mathbb{F}_2^{I_{[y]}}$ is invariant under the action of the permutation group $g_{[y]}(\mathbf{Stab}_G([y] \cap x)) \leq \mathbf{Sym}(I_{[y]})$ on the entries of its vectors. That is, for every $\mathbf{v} \in M[y] \cdot \mathbf{Stab}_E([y] \cap x)$ and $\pi \in g_{[y]}(\mathbf{Stab}_G([y] \cap x))$, it holds $\pi(\mathbf{v}) \in M[y] \cdot \mathbf{Stab}_E([y] \cap x)$.*

Proof. Let $\mathbf{v} \in M[y] \cdot \mathbf{Stab}_E([y] \cap x)$ and $\pi \in g_{[y]}(\mathbf{Stab}_G([y] \cap x))$. We can write $\mathbf{v} = M[y] \cdot \mathbf{w}$ for some $\mathbf{w} \in \mathbf{Stab}_E([y] \cap x)$. Fix a $\sigma \in g_{[y]}^{-1}(\pi)$, i.e. $\sigma \in \mathbf{Stab}_G([y] \cap x)$. By Proposition 8.6.3 it holds:

$$(\pi, \sigma)(M[y]) \cdot \sigma(\mathbf{w}) = \pi(\mathbf{v}).$$

From the inductive hypothesis we have that $(\pi, \sigma)(M[y]) = M[y]$ since $\sigma \in \mathbf{Stab}_G([y] \cap x) \leq \mathbf{Stab}_G([y])$ (see item (c) in the enumeration above). The fact that $\mathbf{Stab}_G([y] \cap x) \leq$

$\mathbf{Stab}_G([y])$ is shown in Corollary 8.6.5. We conclude: $M[y] \cdot \sigma(\mathbf{w}) = \pi(\mathbf{v})$. If $\sigma(\mathbf{w}) \in \mathbf{Stab}_E([y] \cap x)$, then we are done and have that $\pi(\mathbf{v}) \in M[y] \cdot \mathbf{Stab}_E([y] \cap x)$, as desired. To show that $\sigma(\mathbf{w}) \in \mathbf{Stab}_E([y] \cap x)$, we apply Lemma 8.6.7: Since $\mathbf{w} \in \mathbf{Stab}_E([y] \cap x)$, we have $\sigma\mathbf{w} \in \sigma(\mathbf{Stab}_E([y] \cap x)) = \mathbf{Stab}_E(\sigma[y] \cap \sigma x)$. Finally, as mentioned in the proof of Lemma 8.6.7, we have $\sigma[y] \cap \sigma x = \sigma([y] \cap x)$, and it holds $\sigma([y] \cap x) = [y] \cap x$, because $\sigma \in \mathbf{Stab}_G([y] \cap x)$. □

Knowing this, we can see that it is indeed possible to satisfy both conditions (i) and (ii) at the same time.

Lemma 8.6.10. *Let $[y] \in \mathcal{C}[x]$. There exists a Boolean matrix N that satisfies conditions (i) and (ii) mentioned above, i.e.:*

(i) $\boldsymbol{Ker}(N) \cap \boldsymbol{Im}(M[y]) = M[y] \cdot \mathbf{Stab}_E([y] \cap x)$.

(ii) There exists a homomorphism h from $g_{[y]}(\mathbf{Stab}_G([y] \cap x)) \leq \mathbf{Sym}(I_{[y]})$ into the symmetric group on the row index set of N such that for every $\sigma \in g_{[y]}(\mathbf{Stab}_G([y] \cap x))$, it holds $(h(\sigma), \sigma)(N) = N$.

Proof. Lemma 8.6.8 applied to $\Delta = \mathbf{Im}(M[y])$ and $\Gamma = M[y] \cdot \mathbf{Stab}_E([y] \cap x)$ gives us a matrix $N' \in \mathbb{F}_2^{[k] \times I_{[y]}}$ that satisfies condition (i); here, $k = \dim(\mathbf{Im}\ M[y]) - \dim(M[y] \cdot \mathbf{Stab}_E([y] \cap x))$. We can close N' under the action of $g_{[y]}(\mathbf{Stab}_G([y] \cap x)) \leq \mathbf{Sym}(I_{[y]})$ so that condition (ii) is also satisfied: For each row N'_{i-} of N', let $\mathbf{Orb}(N'_{i-}) := \{\pi(N'_{i-}) \mid \pi \in g_{[y]}(\mathbf{Stab}_G([y] \cap x))\}$. Here, $\pi \in \mathbf{Sym}(I_{[y]})$ permutes the columns, i.e. the entries of the respective row N'_{i-}. Now let N be the Boolean matrix whose set of rows is the disjoint union $\biguplus_{i \in [k]} \mathbf{Orb}(N'_{i-})$. Clearly, there is a homomorphism h from $g_{[y]}(\mathbf{Stab}_G([y] \cap x))$ into the symmetric group on the rows of N (more precisely into $\Pi_{i \in [k]} \mathbf{Sym}(\mathbf{Orb}(N'_{i-}))$). This homomorphism is just the group action of $g_{[y]}(\mathbf{Stab}_G([y] \cap x))$ on the rows of N (separately on the orbits $\mathbf{Orb}(N'_{i-})$).
It remains to show that this symmetry-closed matrix N still satisfies condition (i), i.e. that $\mathbf{Ker}(N) \cap \mathbf{Im}(M[y]) = M[y] \cdot \mathbf{Stab}_E([y] \cap x)$. We have $\mathbf{Ker}(N) \cap \mathbf{Im}(M[y]) \subseteq M[y] \cdot \mathbf{Stab}_E([y] \cap x)$ because for any vector $\mathbf{v} \notin M[y] \cdot \mathbf{Stab}_E([y] \cap x)$, either $\mathbf{v} \notin \mathbf{Im}(M[y])$, or if $\mathbf{v} \in \mathbf{Im}(M[y])$, then $N' \cdot \mathbf{v} \neq \mathbf{0}$ because $\mathbf{Ker}(N') \cap \mathbf{Im}(M[y]) = M[y] \cdot \mathbf{Stab}_E([y] \cap x)$. Since N' is a submatrix of N, we also have $N \cdot \mathbf{v} \neq \mathbf{0}$, so $\mathbf{v} \notin \mathbf{Ker}(N)$. Therefore, $\mathbf{Ker}(N) \cap \mathbf{Im}(M[y]) \subseteq M[y] \cdot \mathbf{Stab}_E([y] \cap x)$. It remains to show: $M[y] \cdot \mathbf{Stab}_E([y] \cap x) \subseteq \mathbf{Ker}(N) \cap \mathbf{Im}(M[y])$. So let $\mathbf{v} \in M[y] \cdot \mathbf{Stab}_E([y] \cap x)$. Then $N' \cdot \mathbf{v} = \mathbf{0}$. We have to prove that for every row N'_{i-} and every $\pi \in g_{[y]}(\mathbf{Stab}_G([y] \cap x))$, we have $\pi(N'_{i-}) \cdot \mathbf{v} = 0$. It holds (see Proposition 8.6.3):

$$\pi(N'_{i-}) \cdot \mathbf{v} = \pi^{-1}(\pi(N'_{i-})) \cdot \pi^{-1}\mathbf{v} = N'_{i-} \cdot \pi^{-1}\mathbf{v} = 0.$$

The final equality holds because $\mathbf{Ker}(N') = M[y] \cdot \mathbf{Stab}_E([y] \cap x)$, and $\pi^{-1}\mathbf{v} \in M[y] \cdot \mathbf{Stab}_E([y] \cap x)$ by Lemma 8.6.9. Since each row of N is of the form $\pi(N'_{i-})$ for some row i of N' and $\pi \in g_{[y]}(\mathbf{Stab}_G([y] \cap x))$, we have shown that $M[y] \cdot \mathbf{Stab}_E([y] \cap x) \subseteq \mathbf{Ker}(N) \cap \mathbf{Im}(M[y])$. □

This lemma shows that there exists a Boolean matrix satisfying conditions (i) and (ii), so it is indeed possible to pick a smallest one for $N[x][y]$. The trouble is that we do not know a priori how small it is. Therefore, the construction of $\widehat{C}(\mu)$ that we are describing does not come with a guaranteed size bound. Later on in Section 8.6.3 we will get back to the choice of $N[x][y]$ and show how we can bound its size in case that $\mathbf{Stab}_E([y] \cap x)$ has a symmetric basis. The idea will be that the size of the closure of N' under the action of $g_{[y]}(\mathbf{Stab}_G([y] \cap x))$ can be bounded then, because the symmetries of the basis of $\mathbf{Stab}_E([y] \cap x)$ "propagate" through the equation systems $A_i \cdot \mathbf{x} = \mathbf{b}_i$ that are used to define the rows of N'.

Since at least one matrix satisfying (i) and (ii) exists, $N[x][y]$ can indeed be defined as the smallest one. Remember that this definition was for a fixed $[y] \in \mathcal{C}[x]$. Now let $[y'] \in \Omega_{[y]}$, and $[x'] \in \Omega_{[x]}$ such that $[y'] \in \mathcal{C}[x']$ and such that there is a $\pi \in \mathbf{Stab}_G(\mu)$ with $\pi([y] \cap x) = [y'] \cap x'$ (it may be that $[x'] = [x]$). Set $\pi_{[x'][y']} := \pi$, so we can refer back to this particular permutation in the future. We set $J_{[x'][y']} := \{(i, [x'], [y']) \mid i \in [m]\}$. Here, m still denotes the number of rows of the previously defined $N[x][y]$. Let $N[x'][y']$ be the Boolean matrix in $\mathbb{F}_2^{J_{[x'][y']} \times I_{[y']}}$ such that $(N[x'][y'])_{(k,[x'],[y']),g_{[y]}(\pi_{[x'][y']})(i)} = (N[x][y])_{(k,[x],[y]),i}$ for every $i \in I_{[y]}$ and $k \in [m]$. We will usually write $N([x'][y'])_{k,-}$ instead of $(N[x'][y'])_{(k,[x'],[y']),-}$ to denote the k-th row of the matrix. In this way, we define the matrices $N[x'][y']$ for all $[y'] \in \Omega_{[y]}$, i.e. their rows are simply permutations of the rows of $N[x][y]$. We will call the $\sim$-class $[y] \in \mathcal{C}[x]$, that we arbitrarily chose as the first one in its orbit to define $N[x][y]$ with Lemma 8.6.10, the *primer* of the orbit $\Omega_{[y]}$. We proceed to pick a new primer $[y] \in \mathcal{C}[x]$ for which $N[x][y]$ has not been defined so far, and repeat the construction for $[y]$ and its orbit $\Omega_{[y]}$. This is done until $N[x][y]$ is defined for every $[y] \in \mathcal{C}[x]$ and for every $[y'] \in \mathcal{C}[x']$, for every $[x'] \in \Omega_{[x]}$.

We show that the defined matrices have the desired properties:

Lemma 8.6.11. *Let $[x'] \in \Omega_{[x]}$ and $[y'] \in \mathcal{C}[x']$. Then*

$$\boldsymbol{Ker}(N[x'][y'] \cdot M[y']) = \mathbf{Stab}_E([y'] \cap x').$$

Proof. Let $[y] \in \mathcal{C}[x]$ be the primer of the orbit $\Omega_{[y']}$ that was used in the matrix construction. Then $[y'] = \pi_{[x'][y']}[y]$ and $[x'] = \pi_{[x'][y']}[x]$. If $[y'] = [y]$, then $\pi_{[x'][y']}$ is the identity permutation in $\mathbf{Sym}(V)$. For ease of notation, we write $\pi := \pi_{[x'][y']}$ in the following. By Lemma 8.6.7, we have

$$\mathbf{Stab}_E([y'] \cap x') = \mathbf{Stab}_E(\pi[y] \cap \pi x) = \pi(\mathbf{Stab}_E([y] \cap x)).$$

By definition of $N[x][y]$, and because $\mathbf{Ker}(M[y]) = \mathbf{Stab}_E(y) \leq \mathbf{Stab}_E([y] \cap x)$, we have $\mathbf{Ker}(N[x][y] \cdot M[y]) = \mathbf{Stab}_E([y] \cap x)$. It holds $M[y'] = (g_{[y]}(\pi), \pi)M[y]$ (item (c) of the inductive hypothesis). So for any vector $\mathbf{v} \in \mathbb{F}_2^E$, we have $M[y'] \cdot \pi(\mathbf{v}) = g_{[y]}(\pi)(M[y] \cdot \mathbf{v})$. Here, $g_{[y]}(\pi)$ acts on a vector $\mathbf{w} \in \mathbb{F}_2^{I_{[y]}}$ by mapping it to a vector $\mathbf{w}' \in \mathbb{F}_2^{I_{[y']}}$ with

$\mathbf{w}'(g_{[y]}(\pi)(i)) = \mathbf{w}(i)$ for every $i \in I_{[y]}$. By definition, we have for the k-th row of $N[x'][y']$: $N([x'][y'])_{k,-} = g_{[y]}(\pi)(N([x][y])_{k,-})$. In total, this means that for every $\mathbf{v} \in \mathbb{F}_2^E$, it holds:

$$N([x'][y'])_{k,-} \cdot M[y'] \cdot \pi(\mathbf{v}) = N([x][y])_{k,-} \cdot M[y] \cdot \mathbf{v}.$$

Therefore, $\mathbf{Ker}(N[x'][y'] \cdot M[y']) = \pi(\mathbf{Ker}(N[x][y] \cdot M[y])) = \pi(\mathbf{Stab}_E([y] \cap x)) = \mathbf{Stab}_E([y'] \cap x')$. $\square$

Lemma 8.6.12. *Let $[x'] \in \Omega_{[x]}$ and $[y'] \in \mathcal{C}[x']$. There exists a homomorphism h' from $g_{[y']}(\mathbf{Stab}_G([y'] \cap x')) \leq \mathbf{Sym}(I_{[y']})$ into the symmetric group on the row index set of $N[x'][y']$ such that for every $\sigma \in g_{[y']}(\mathbf{Stab}_G([y'] \cap x'))$, it holds $(h'(\sigma), \sigma)(N[x'][y']) = N[x'][y']$.*

Proof. Let $[y] \in \mathcal{C}[x]$ be the primer of $\Omega_{[y']}$. Let $\pi_{[x'][y']} \in \mathbf{Stab}_G(\mu)$ be the permutation that was used to define $N[x'][y']$ from $N[x][y]$. For the matrix $N[x][y]$, there exists such a homomorphism $h : g_{[y]}(\mathbf{Stab}_G([y] \cap x)) \longrightarrow \mathbf{Sym}(J_{[x][y]})$ by definition of the matrix. We define the desired homomorphism $h' : g_{[y']}(\mathbf{Stab}_G([y'] \cap x')) \longrightarrow \mathbf{Sym}(J_{[x'][y']})$ as follows. For every $\pi \in \mathbf{Stab}_G([y'] \cap x')$ and every $(i, [x'], [y']) \in J_{[x'][y']}$, let

$$h'(g_{[y']}(\pi))(i, [x'], [y']) := (j, [x'], [y']),$$

where j is the number such that

$$h(g_{[y]}(\pi^{-1}_{[x'][y']} \circ \pi \circ \pi_{[x'][y']}))(i, [x], [y]) = (j, [x], [y]).$$

For every $\pi \in \mathbf{Stab}_G([y'] \cap x')$, $h'(g_{[y']}(\pi))$ is indeed a permutation in $\mathbf{Sym}(J_{[x'][y']})$, because $N[x][y]$ and $N[x'][y']$ have the same number of rows (and the index sets $J_{[x][y]}$ and $J_{[x'][y']}$ differ only with respect to the second and third entry of the index triples), and $(\pi^{-1}_{[x'][y']} \circ \pi \circ \pi_{[x'][y']}) \in \mathbf{Stab}_G([y] \cap x)$, so $h(g_{[y]}(\pi^{-1}_{[x'][y']} \circ \pi \circ \pi_{[x'][y']}))$ is a permutation on the rows of $N[x][y]$.
The fact that h' is a group homomorphism follows directly from the fact that h is one, and because $(\pi^{-1}_{[x'][y']} \circ \pi_1 \circ \pi_{[x'][y']}) \circ (\pi^{-1}_{[x'][y']} \circ \pi_2 \circ \pi_{[x'][y']}) = \pi_1 \circ \pi_2$.
Finally, we have to show that for every $\sigma \in g_{[y']}(\mathbf{Stab}_G([y'] \cap x'))$, it holds $(h'(\sigma), \sigma)(N[x'][y']) = N[x'][y']$. To prove this, we show that $N[x'][y']_{h'(\sigma)(j,[x'],[y']),\sigma(i)} = N[x'][y']_{(j,[x'],[y']),i}$ for every $(j, [x'], [y']) \in J_{[x'][y']}$ and $i \in I_{[y']}$. In the following, we will use that by definition of $N[x'][y']$, we have: $N[x'][y'] = (\mathrm{id}, g_{[y]}(\pi_{[x'][y']}))(N[x][y])$. Let $\sigma = g_{[y']}(\pi) = g_{[y]}(\pi)$ for a $\pi \in \mathbf{Stab}_G([y'] \cap x')$. Then for any $(j, [x'], [y']) \in J_{[x'][y']}, i \in I_{[y']}$, we obtain:

$$\begin{aligned} N[x'][y']_{h'(\sigma)(j,[x'],[y']),\sigma(i)} &= (N[x][y])_{h(g_{[y]}(\pi^{-1}_{[x'][y']} \circ \pi \circ \pi_{[x'][y']}))(j,[x],[y]),(g_{[y]}(\pi_{[x'][y']})^{-1} \circ \sigma)(i)} \\ &= (N[x][y])_{h(g_{[y]}(\pi^{-1}_{[x'][y']} \circ \pi \circ \pi_{[x'][y']}))(j,[x],[y]),g_{[y]}(\pi^{-1}_{[x'][y']} \circ \pi \circ \pi_{[x'][y']})(i')} \\ &= (N[x][y])_{(j,[x],[y]),i'}. \end{aligned}$$

In the second step, we used that $\sigma(i) = g_{[y']}(\pi)(i)$, and we replaced $i \in I_{[y']}$ with $g_{[y]}(\pi_{[x'][y']})(i')$ for some $i' \in I_{[y]}$ (which can be done because of item (c) of the induction hypothesis for $I_{[y]}$ and $I_{[y']} = I_{\pi_{[x'][y']}[y]}$). We also used that $g_{[y]}$ is a group homomorphism. The last step holds because we already know that h satisfies the property that we are trying to prove for h', i.e. $(h(\sigma'), \sigma')N[x][y] = N[x][y]$ for any $\sigma' \in g_{[y]}(\mathbf{Stab}_G([y] \cap x))$ (this is by construction of $N[x][y]$ and h). We can continue the equation, using the definition of $N[x'][y']$ again:

$$\begin{aligned}(N[x][y])_{(j,[x],[y]),i'} &= (N[x'][y'])_{(j,[x'],[y']),g_{[y]}(\pi_{[x'][y']})(i')}\\ &= (N[x'][y'])_{(j,[x'],[y']),i}.\end{aligned}$$

This proves that $(h'(\sigma), \sigma)(N[x'][y']) = N[x'][y']$, as desired. □

Lemmas 8.6.11 and 8.6.12 assert that all the constructed matrices $N[x'][y']$ satisfy the properties (i) and (ii) that $N[x][y]$ has by construction.
Now let

$$J_{\Omega_{[x]}} := \bigcup_{\substack{[x'] \in \Omega_{[x]}\\ [y'] \in \mathcal{C}[x']}} J_{[x'][y']}.$$

We provide a group homomorphism $h_{[x]} : \mathbf{Stab}_G(\mu) \longrightarrow \mathbf{Sym}(J_{\Omega_{[x]}})$ such that for each $[x'] \in \Omega_{[x]}$ and each $[y'] \in \mathcal{C}[x']$, it holds $h_{[x]}(\pi)(J_{[x'][y']}) = J_{\pi[x']\pi[y']}$. Furthermore, we want that $(N\pi[x']\pi[y']) = (h_{[x]}(\pi), g_{[x]}(\pi))N[x'][y']$ for each $[x'] \in \Omega_{[x]}$, $[y'] \in \mathcal{C}[x']$, and $\pi \in \mathbf{Stab}_G(\mu)$. For each triple $(k, [x'], [y']) \in J_{\Omega_{[x]}}$, we set:

$$h_{[x]}(\pi)(k, [x'], [y']) := (\ell, \pi[x'], \pi[y']),$$

where ℓ is defined as follows: Let

$$t := |\{i \in \{1, 2, ..., k-1\} \mid (N[x'][y'])_{i,-} = (N[x'][y'])_{k,-}\}|.$$

That is, for the k-th row of $N[x'][y']$, there are t rows identical to it with a smaller index. Then ℓ is defined such that

$$|\{i \in \{1, 2, ..., \ell-1\} \mid (N\pi[x']\pi[y'])_{i,-} = g_{[y']}(\pi)((N[x'][y'])_{k,-})\}| = t,$$

and such that $(N\pi[x']\pi[y'])_{\ell,-} = g_{[y']}(\pi)((N[x'][y'])_{k,-})$. In other words, ℓ is the $(t+1)$st row of $N\pi[x']\pi[y']$ which is equal to the k-th row of $N[x][y]$, up to a permutation of the columns given by $g_{[y']}(\pi)$.
We have to argue that $h_{[x]}(\pi)(k, [x'], [y'])$ is indeed well-defined:

Lemma 8.6.13. *Let $[x'] \in \Omega_{[x]}, [y'] \in \mathcal{C}[x']$. Let $\pi \in \mathbf{Stab}_G(\mu)$. Then for every $(k, [x'], [y']) \in J_{[x'][y']}$, the number of rows of $N[x'][y']$ which are equal to the k-th row is the same as the number of rows of $N\pi[x']\pi[y']$ that are equal to the k-th row of $N[x'][y']$, up to application of $g_{[y']}$ to the columns. Formally:*

$$\begin{aligned}&|\{(i, [x'], [y']) \in J_{[x'][y']} \mid (N[x'][y'])_{(i,[x'],[y']),-} = (N[x'][y'])_{k,-}\}|\\ =&|\{(i, \pi[x'], \pi[y']) \in J_{\pi[x']\pi[y']} \mid (N\pi[x']\pi[y'])_{(i,\pi[x'],\pi[y']),-} = g_{[y']}((N[x'][y'])_{k,-})\}|\end{aligned}$$

Proof. Let $[y] \in \mathcal{C}[x]$ be the primer of $\Omega_{[y']}$, and write $\sigma := \pi_{[x'][y']}, \sigma' := \pi_{\pi[x']\pi[y']}$. So these are the two permutations in $\mathbf{Stab}_G(\mu)$ that were used to construct $N[x'][y']$ and $N\pi[x']\pi[y']$ from $N[x][y]$. By construction of these matrices, it holds $N[x'][y'] = (\mathrm{id}, g_{[y]}(\sigma))(N[x][y])$ and $N[x'][y'] = (\mathrm{id}, g_{[y]}(\sigma'))(N[x][y])$. Thus, $(N\pi[x']\pi[y'])_{k,-} = (g_{[y]}(\sigma') \circ g_{[y]}(\sigma)^{-1})((N[x'][y'])_{k,-})$. Since $g_{[y]}$ is a group homomorphism, we can also write this as:

$$(N\pi[x']\pi[y'])_{k,-} = g_{[y]}(\sigma' \circ \sigma^{-1})((N[x'][y'])_{k,-}) \tag{$\star$}$$

If $\sigma' \circ \sigma^{-1}$ were equal to π, then this would suffice to prove the lemma. However, we only know that $(\sigma' \circ \sigma^{-1})([y'] \cap x') = \pi([y'] \cap x')$. We use this to show the following
Claim: Let m be the number of rows of $N[x'][y']$ and $N\pi[x']\pi[y']$. There exists a permutation $\vartheta \in \mathbf{Sym}_m$ such that $(N\pi[x']\pi[y'])_{\vartheta(k),-} = g_{[y]}(\pi)((N[x'][y'])_{k,-})$, for every $k \in [m]$.
Proof. It holds that $(\pi \circ \sigma \circ (\sigma')^{-1}) \in \mathbf{Stab}_G(\pi[y'] \cap \pi x')$. By Lemma 8.6.12, there exists a $\vartheta \in \mathbf{Sym}_m$ such that $(\vartheta, g_{[y]}(\pi \circ \sigma \circ (\sigma')^{-1})))(N\pi[x']\pi[y']) = N\pi[x']\pi[y']$. It holds $\sigma \circ (\sigma')^{-1} = (\sigma' \circ \sigma^{-1})^{-1}$. Thus, by $(\star)$ we have: $(N[x'][y'])_{k,-} = g_{[y]}(\sigma \circ (\sigma')^{-1})((N\pi[x']\pi[y'])_{k,-})$. It follows that $(N\pi[x']\pi[y'])_{\vartheta(k),-} = g_{[y]}(\pi)(N[x'][y'])_{k,-}$ for every $k \in [m]$. This proves the claim.
The claim entails the lemma because $g_{[y']} = g_{[y]}$ (as $\Omega_{[y]} = \Omega_{[y']}$), and so we know that the rows of $N\pi[x']\pi[y']$ are the rows of $N[x'][y']$, with an application of $g_{[y']}(\pi)$ to the columns, and a potential reordering of the rows. □

This lemma shows that we can indeed define $h_{[x]}$ as we did. Now we have to verify that $h_{[x]}$ is a group homomorphism with the desired properties:

Lemma 8.6.14. *The mapping $h_{[x]} : \mathbf{Stab}_G(\mu) \longrightarrow \mathbf{Sym}(J_{\Omega_{[x]}})$ is a group homomorphism. For every $[x'] \in \Omega_{[x]}$, $[y'] \in \mathcal{C}[x']$ and each $\pi \in \mathbf{Stab}_G(\mu)$, it holds $h_{[x]}(\pi)(J_{[x'][y']}) = J_{\pi[x']\pi[y']}$. Furthermore, $N\pi[x']\pi[y'] = (h_{[x]}(\pi), g_{[y']}(\pi))N[x'][y']$ for each $[x'] \in \Omega_{[x]}$, $[y'] \in \mathcal{C}[x']$, and $\pi \in \mathbf{Stab}_G(\mu)$.*

Proof. First, we show that $h_{[x]}$ is a group homomorphism. It can be seen directly from the definition that the identity permutation in $\mathbf{Stab}_G(\mu)$ is mapped to the identity permutation in $\mathbf{Sym}(J_{\Omega_{[x]}})$. Now let $\pi, \pi' \in \mathbf{Stab}_G(\mu)$. Let $(k, [x'], [y']) \in J_{\Omega_{[x]}}$. Then $h_{[x]}(\pi' \circ \pi)(k, [x'], [y']) = (\ell, (\pi' \circ \pi)[x'], (\pi' \circ \pi)[y'])$, where ℓ is such that

$$(N(\pi' \circ \pi)[x'](\pi' \circ \pi)[y'])_{\ell,-} = g_{[y]}(\pi' \circ \pi)(N[x'][y']_{k,-}),$$

and there are exactly t rows identical to it with an index $< \ell$ in $N(\pi' \circ \pi)[x'](\pi' \circ \pi)[y']$. As in the definition, t denotes the number of rows in $N[x'][y']$ with an index $< k$ that are identical to $(N[x'][y'])_{k,-}$. Now we want to argue that $(h_{[x]}(\pi') \circ h_{[x]}(\pi))(k, [x'], [y']) = (\ell, (\pi' \circ \pi)[x'], (\pi' \circ \pi)[y']) = h_{[x]}(\pi' \circ \pi)(k, [x'], [y'])$.
We have $h_{[x]}(\pi)(k, [x'], [y']) = (\ell_1, \pi[x], \pi[y])$, where ℓ_1 is the row index in $N\pi[x']\pi[y']$ such that

$$(N\pi[x']\pi[y'])_{\ell_1,-} = g_{[y]}(\pi)(N[x'][y']_{k,-}),$$

and there are again exactly t identical rows with index $< \ell_1$ in $N\pi[x']\pi[y']$. Thus we have $h_{[x]}(\pi')(\ell_1, \pi[x'], \pi[y']) = (\ell_2, (\pi' \circ \pi)[x'], (\pi' \circ \pi)[y'])$, where ℓ_2 is the row index in

$N(\pi' \circ \pi)[x'](\pi' \circ \pi)[y']$ such that

$$(N(\pi' \circ \pi)[x'](\pi' \circ \pi)[y'])_{\ell_2,-} = g_{[y]}(\pi' \circ \pi)(N[x'][y']_{k,-}),$$

and such that there are exactly t identical rows with index $< \ell_2$ in $N(\pi' \circ \pi)[x'](\pi' \circ \pi)[y']$. In the equation above we used that $g_{[y]}(\pi') \circ g_{[y]}(\pi) = g_{[y]}(\pi' \circ \pi)$, because $g_{[y]}$ is a group homomorphism. Now we see that $\ell_2 = \ell$ because the row $(N[x'][y'])_{k-}$ is always mapped to the respective $(t+1)$-st row of the same kind in each matrix. This is what we had to show. Similarly one can see that $h^{-1}_{[x]}(\pi) = h_{[x]}(\pi^{-1})$. This shows that $h_{[x]}$ is a group homomorphism.

It remains to prove: For every $[x'] \in \Omega_{[x]}$ and each $[y'] \in \mathcal{C}[x']$, it holds $h_{[x]}(\pi)(J_{[x'][y']}) = J_{\pi[x']\pi[y']}$. This is true by the definition of $h_{[x]}$ and Lemma 8.6.13. The fact that $(N\pi[x']\pi[y'])_{h_{[x]}(\pi)(j),g_{[x]}(\pi)(i)} = N[x'][y']_{j,i}$ for each $[x'] \in \Omega_{[x]}$, $[y'] \in \mathcal{C}[x']$, $\pi \in \mathbf{Stab}_G(\mu)$, $i \in I_{[x']}$, and $j \in J_{[x'][y']}$ is also true by definition. □

This finishes the construction and correctness proof of the matrices $N[x'][y']$ for all $[x'] \in \Omega_{[x]}$ and $[y'] \in \mathcal{C}[x']$ and of the associated homomorphism $h_{[x]} : \mathbf{Stab}_G(\mu) \longrightarrow J_{\Omega_{[x]}}$. Items (b) and (d) of the inductive step are thus covered. To complete the inductive step we still have to define the matrices $M[x']$ for all $[x'] \in \Omega_{[x]}$.

Definition of the M-matrices Let $[x'] \in \Omega_{[x]}$. Informally, $M[x']$ is obtained by collecting all the rows of the matrices $(N[x'][y'] \cdot M[y'])$, for all $[y'] \in \mathcal{C}[x']$, and putting them together as the rows of $M[x']$. Formally, let

$$I_{[x']} := \biguplus_{[y'] \in \mathcal{C}[x']} J_{[x'][y']}.$$

Note that by construction, the sets $J_{[x'][y']}, J_{[x'][y'']}$ are pairwise disjoint if $[y'] \neq [y'']$. Then the rows of $M[x'] \in \mathbb{F}_2^{I_{[x']} \times E}$ are defined as follows: For $[y'] \in \mathcal{C}[x']$ and $(i, [x'], [y']) \in J_{[x'][y']}$, we let

$$M[x']_{(i,[x'],[y']),-} := (N[x'][y'] \cdot M[y'])_{(i,[x'],[y']),-}.$$

This matrix has the desired kernel:

Lemma 8.6.15. *For every $[x'] \in \Omega_{[x]}$, the matrix $M[x']$ defined as above satisfies:*

$$\boldsymbol{Ker}(M[x']) = \mathbf{Stab}_E(x').$$

Proof. By definition of $M[x']$, a vector $\mathbf{v} \in \mathbb{F}_2^E$ is in $\mathbf{Ker}(M[x'])$ if and only if $\mathbf{v} \in \mathbf{Ker}(N[x'][y'] \cdot M[y'])$ for all $[y'] \in \mathcal{C}[x']$. By Lemma 8.6.11, this is the case iff $\mathbf{v} \in \mathbf{Stab}_E([y'] \cap x')$, for all $[y'] \in \mathcal{C}[x']$. That is to say,

$$\mathbf{v} \in \bigcap_{[y'] \in \mathcal{C}[x']} \mathbf{Stab}_E([y'] \cap x').$$

This is equivalent to $\mathbf{v} \in \mathbf{Stab}_E(x')$ because $x' = \bigcup_{[y'] \in \mathcal{C}[x']}([y'] \cap x')$ (see Proposition 8.4.5). □

Finally, we have to provide the homomorphism $g_{[x]} : \mathbf{Stab}_G(\mu) \longrightarrow \mathbf{Sym}(I_{\Omega_{[x]}})$, where $I_{\Omega_{[x]}} = \biguplus_{[x'] \in \Omega_{[x]}} I_{[x']}$. Note that $I_{\Omega_{[x]}} = J_{\Omega_{[x]}}$ by definition of the index sets $I_{[x']}$. Therefore we can simply set $g_{[x]} := h_{[x]}$. This homomorphism indeed satisfies the desired properties:

Lemma 8.6.16. *For each $[x'] \in \Omega_{[x]}$ and each $\pi \in \mathbf{Stab}_G(\mu)$, it holds $g_{[x]}(\pi)(I_{[x']}) = I_{\pi[x']}$. Furthermore, $M\pi[x'] = (g_{[x]}(\pi), \pi)M[x']$.*

Proof. Let $\pi \in \mathbf{Stab}_G(\mu)$ and $[x'] \in \Omega_{[x]}$. Due to Lemma 8.6.4, $\pi[x']$ is again a $\sim$-class in $\mathrm{tc}(\mu)$. Because π extends to an automorphism of μ, it also holds

$$\pi(\mathcal{C}[x']) = \{\pi[y'] \mid [y'] \in \mathcal{C}[x']\} = \mathcal{C}(\pi[x']).$$

Hence, by Lemma 8.6.14, $h_{[x]}(\pi)$ maps the set $I_{[x']} = \biguplus_{[y'] \in \mathcal{C}[x']} J_{[x'][y']}$ to the set $I_{\pi[x']} = \biguplus_{\pi[y'] \in \mathcal{C}\pi[x']} J_{\pi[x']\pi[y']}$. It remains to show $(M\pi[x'])_{g_{[x]}(\pi)(i),\pi(e)} = M[x']_{i,e}$ for each $i = (j, [x'], [y']) \in I_{[x']}$, and $e \in E$. By definition of the M-matrices, we have:

$$\begin{aligned}(M\pi[x'])_{g_{[x]}(\pi)(i),\pi(e)} = (M\pi[x'])_{(j',\pi[x'],\pi[y']),\pi(e)} &= (N\pi[x']\pi[y'] \cdot M\pi[y'])_{(j',\pi[x'],\pi[y']),\pi(e)} \\ &= (N[x'][y'] \cdot M[y'])_{(j,[x'],[y']),e} = M[x']_{i,e}.\end{aligned}$$

Here, j' is such that $(j', \pi[x'], \pi[y']) = g_{[x]}(\pi)(j, [x'], [y']) = h_{[x]}(\pi)(j, [x'], [y'])$. The final equality holds for the following reason: $(N\pi[x']\pi[y'] \cdot M\pi[y'])_{(j',\pi[x'],\pi[y']),\pi(e)}$ is the product of row j' of $N\pi[x']\pi[y']$ with column $\pi(e)$ of $M\pi[y']$. The lemma we are currently proving already holds for $M[y']$ by induction hypothesis, so $(M\pi[y'])_{-\pi(e)} = g_{[y']}(\pi)(M[y']_{-e})$. Also, we have $(N\pi[x']\pi[y'])_{j',-} = g_{[y']}(\pi)((N[x'][y'])_{j,-})$ by Lemma 8.6.14. Then Proposition 8.6.3 tells us that $(N\pi[x']\pi[y'])_{j',-} \cdot (M\pi[y'])_{-,\pi(e)} = (N[x'][y'])_{j,-} \cdot (M[y'])_{-e}$ because these vector products are really the same sums of products, where the summands are just reordered by $g_{[y']}(\pi)$. □

8.6.2 Construction of the circuit

We construct the circuit $\widehat{C}(\mu) = (V_C, E_C)$ from the matrices that we have defined in the previous section. The set of gates is

$$V_C := \biguplus_{\substack{x \in \mathrm{tc}(\mu) \\ (i,[x],[y]) \in I_{[x]}}} \mathbf{g}_{i,[x],[y]}.$$

So every row of any of the $M[x]$-matrices with index $(i, [x], [y])$ will correspond to a gate $\mathbf{g}_{i,[x],[y]}$. The construction will ensure that

$$\chi(\mathcal{X}(\mathbf{g}_{i,[x],[y]})) = M[x]^T_{(i,[x],[y]),-}.$$

Thus, the gate will compute the XOR over precisely the input gates labelled with edges that have a 1-entry in the row $M[x]_{(i,[x],[y]),-}$.

For every atom $x \in \mathrm{tc}(\mu)$, we have $I_{[x]} = \{[x]\}$. We define the corresponding gate $\mathbf{g}_{[x]}$ as an input gate of $\widehat{C}(\mu)$ with $\ell(\mathbf{g}_{[x]}) = e$, where $e \in E$ is the edge such that $[x] \subseteq \{e_0, e_1\}$.

If $x \in \mathrm{tc}(\mu)$ is not an atom, then for every $(i, [x], [y]) \in I_{[x]}$, $\mathbf{g}_{i,[x],[y]}$ is an internal gate that has incoming wires from exactly those gates $\mathbf{g}_{j,[y],[z]}$ such that the matrix $N[x][y]$ has a 1-entry in row $(i, [x], [y])$ and column $(j, [y], [z])$. Note that $(j, [y], [z]) \in I_{[y]}$, so $[z] \in \mathcal{C}[y]$. Formally, the set of children of $\mathbf{g}_{i,[x],[y]}$ is:

$$\mathbf{g}_{i,[x],[y]} E_C := \{\mathbf{g}_{j,[y],[z]} \mid (j, [y], [z]) \in I_{[y]}, (N[x][y])_{(i,[x],[y]),(j,[y],[z])} = 1\}.$$

It remains to specify the root of $\widehat{C}(\mu)$. Consider the matrix $M[\mu]$. Let $(i, [\mu], [y])$ be a row of $M[\mu]$ with a maximum number of one-entries. We define the root r to be the gate $\mathbf{g}_{i,[\mu],[y]}$.

The figure below shows the matrices and XOR-gates for an object x consisting of two connected components $[y] \cap x$ and $[y'] \cap x$. The matrices $N[x][y]$ and $N[x][y']$ have just one row each in this example, and we are assuming that $\mathcal{C}[y] = \{[z]\}$ and $\mathcal{C}[y'] = \{[z']\}$, and that $M[y] = N[y][z] \cdot M[z]$ has three rows and $M[y'] = N[y'][z'] \cdot M[z']$ has two rows.

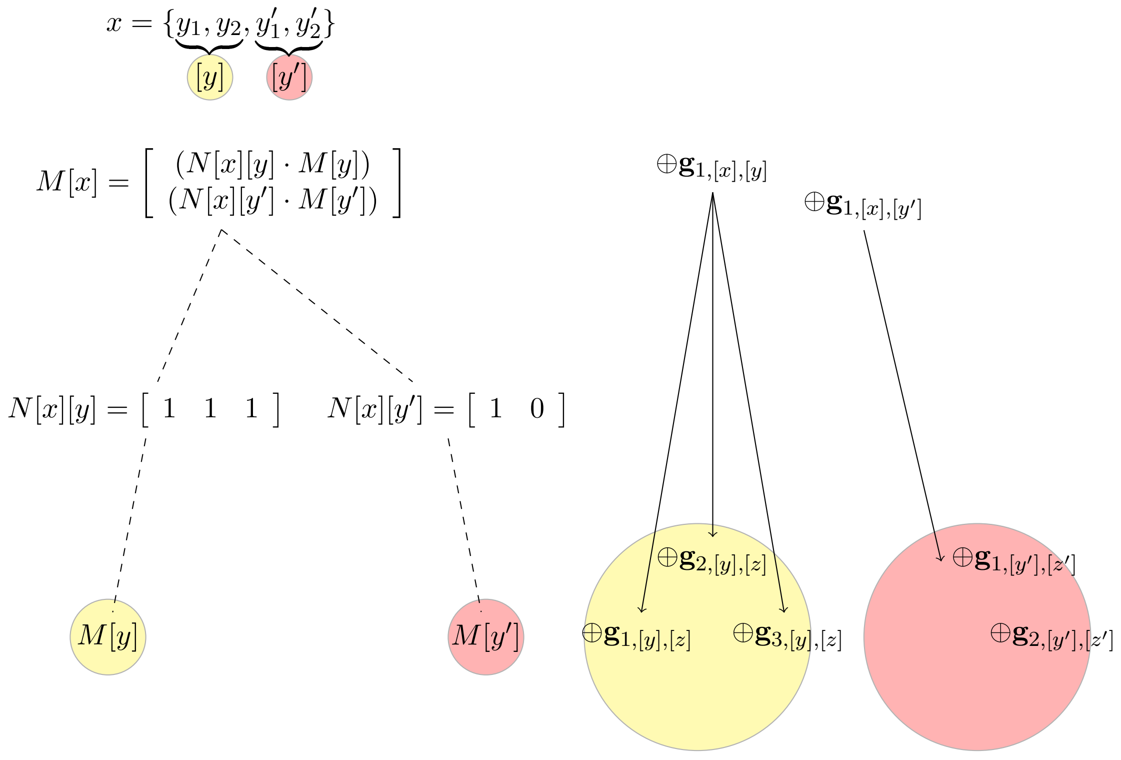

$x = \{\underbrace{y_1, y_2}_{[y]}, \underbrace{y'_1, y'_2}_{[y']}\}$
$M[x] = \begin{bmatrix} (N[x][y] \cdot M[y]) \\ (N[x][y'] \cdot M[y']) \end{bmatrix}$
$N[x][y] = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$
$N[x][y'] = \begin{bmatrix} 1 & 0 \end{bmatrix}$
$M[y]$
$M[y']$
$\oplus\mathbf{g}_{1,[x],[y]}$
$\oplus\mathbf{g}_{1,[x],[y']}$
$\oplus\mathbf{g}_{2,[y],[z]}$
$\oplus\mathbf{g}_{1,[y],[z]}$
$\oplus\mathbf{g}_{3,[y],[z]}$
$\oplus\mathbf{g}_{1,[y'],[z']}$
$\oplus\mathbf{g}_{2,[y'],[z']}$

Figure 8.1: Example showing how the XOR-gates are connected according to the N-matrices.

Now we prove that $\widehat{C}(\mu)$ has the desired properties.

Lemma 8.6.17. *Every $\pi \in \mathbf{Stab}_G(\mu)$ extends to an automorphism of the circuit $\widehat{C}(\mu)$, that is:*

$$\mathbf{Stab}_G(\mu) \leq \mathbf{Stab}_G(\widehat{C}(\mu)).$$

Proof. Let $\pi \in \mathbf{Stab}_G(\mu)$. We claim that the following mapping $\sigma : V_C \longrightarrow V_C$ is an automorphism of $\widehat{C}(\mu)$ that π extends to. We let

$$\sigma(\mathbf{g}_{i,[x],[y]}) := \mathbf{g}_{g_{[x]}(\pi)(i,[x],[y])},$$

where $g_{[x]} : \mathbf{Stab}_G(\mu) \longrightarrow \mathbf{Sym}(I_{\Omega_{[x]}})$ is the group homomorphism for $\Omega_{[x]}$ from the construction of the matrices in the previous subsection. We have to show three things about σ. Firstly, that $\sigma \in \mathbf{Sym}(V_C)$. Secondly, that σ maps wires to wires and non-wires to non-wires of $\widehat{C}(\mu)$. Finally, that for every input gate $\mathbf{g}_{[x]}$ it holds: $\ell(\sigma(\mathbf{g}_{[x]})) = \pi(\ell(\mathbf{g}_{[x]}))$. This last statement actually follows directly from the definition of $g_{[x]}$ for atoms $x \in \text{tc}(\mu)$: If $[x] \subseteq \{e_0, e_1\}$, then $\pi(\ell(\mathbf{g}_{[x]})) = \pi(e)$, and $g_{[x]}(\pi)([x]) = \pi[x] \subseteq \{\pi(e_0), \pi(e_1)\}$, so $\ell(\sigma(\mathbf{g}_{[x]})) = \pi(e)$.
The fact that $\sigma \in \mathbf{Sym}(V_C)$ follows because for each orbit $\Omega_{[x]}$, we have that $g_{[x]}(\pi) \in \mathbf{Sym}(I_{\Omega_{[x]}})$, and the set of gates V_C can be partitioned into these orbits so that each part has the form $\{\mathbf{g}_{i,[x'],[y]} \mid (i, [x'], [y]) \in I_{\Omega_{[x]}}\}$, for some orbit $\Omega_{[x]}$.
It remains to prove that σ preserves the wire structure of the circuit. For any two gates $\mathbf{g}_{i,[x],[y]}, \mathbf{g}_{j,[x'],[y']}$, we have

$$(\mathbf{g}_{i,[x],[y]}, \mathbf{g}_{j,[x'],[y']}) \in E_C \text{ if and only if } [y] = [x'] \text{ and } (N[x][y])_{(i,[x],[y]),(j,[y],[y'])} = 1.$$

The latter equation holds if and only if

$$(N\pi[x]\pi[y])_{h_{[x]}(\pi)(i,[x],[y]),g_{[y]}(\pi)(j,[y],[y'])} = 1.$$

This is true by Lemma 8.6.14. If $[x'] = [y]$, then we have $\sigma(\mathbf{g}_{j,[x'],[y']}) = \mathbf{g}_{g_{[y]}(\pi)(j,[y],[y'])}$. Furthermore, it holds $\sigma(\mathbf{g}_{i,[x],[y]}) = \mathbf{g}_{g_{[x]}(\pi)(i,[x],[y])}$. By the definition in the previous section, $g_{[x]} = h_{[x]}$, and it holds that $g_{[x]}(\pi)(i, [x], [y])$ is of the form $(\ell, \pi[x], \pi[y])$, and $g_{[y]}(\pi)(j, [y], [y'])$ is of the form $(\ell', \pi[y], \pi[y'])$, so these are indeed row- and column-indices of the matrix $N\pi[x]\pi[y]$. So altogether, the above equation is equivalent to

$$(\sigma\mathbf{g}_{i,[x],[y]}, \sigma\mathbf{g}_{j,[x'],[y']}) \in E_C.$$

□

As in the case of the circuit $C(\mu)$, this lemma implies that the orbit-size of $\widehat{C}(\mu)$ cannot be greater than the orbit-size of μ (here, $\mathbf{Orb}_G(\mu)$ again refers to the $\mathbf{Aut}(G)$-orbit and not to the $\mathbf{Stab}_G(\mu)$-orbit, which would be pointless).

Corollary 8.6.18.

$$|\mathbf{Orb}_G(\widehat{C}(\mu))| \leq |\mathbf{Orb}_G(\mu)|.$$

Proof. Analogous to the proof of Corollary 8.4.4. □

Lemma 8.6.19. *For every internal gate* $\mathbf{g} := \mathbf{g}_{i,[x],[y]} \in V_C$ *it holds:*

$$\mathcal{X}(\mathbf{g}) = \chi^{-1}(M[x]^T_{(i,[x],[y]),-}).$$

Proof. By induction. For each input gate $\mathbf{g}_{[x]}$ with $[x] \subseteq \{e_0, e_1\}$ we have $\mathcal{X}(\mathbf{g}_{[x]}) = \{e\}$. Now let $\mathbf{g} = \mathbf{g}_{i,[x],[y]}$ be an internal gate. By definition of $\widehat{C}(\mu)$, it holds

$$\mathbf{g}E_C = \{\mathbf{g}_{j,[y],[z]} \mid (j,[y],[z]) \in I_{[y]}, (N[x][y])_{(i,[x],[y]),(j,[y],[z])} = 1\}.$$

Thus,

$$\begin{aligned}\mathcal{X}(\mathbf{g}) &= \mathop{\triangle}_{\substack{(j,[y],[z]) \in I_{[y]},\\ (N[x][y])_{(i,[x],[y]),(j,[y],[z])}=1}} \mathcal{X}(\mathbf{g}_{j,[y],[z]}) \\ &= \chi^{-1}\Big(\sum_{\substack{(j,[y],[z]) \in I_{[y]},\\ (N[x][y])_{(i,[x],[y]),(j,[y],[z])}=1}} \chi(\mathcal{X}(\mathbf{g}_{j,[y],[z]})) \mod 2\Big) \\ &= \chi^{-1}(((N[x][y])_{(i,[x],[y]),-} \cdot M[y])^T).\end{aligned}$$

The last step uses the induction hypothesis and the fact that $I_{[y]}$ is the row-index-set of $M[y]$. By the definition of the matrix $M[x]$ in the previous subsection, we have $(N[x][y])_{(i,[x],[y]),-} \cdot M[y] = M[x]_{(i,[x],[y]),-}$. This finishes the proof. □

Lemma 8.6.20. *The* fan-in dimension *of* $\widehat{C}(\mu)$ *is at most* $\log(\mathbf{maxOrb}_E(\mu))$.

Proof. Consider any gate $\mathbf{g} := \mathbf{g}_{i,[x],[y]} \in V_C$. By definition of $\widehat{C}(\mu)$, the children of $\mathbf{g}$ are

$$\mathbf{g}E_C = \{\mathbf{g}_{j,[y],[z]} \mid (j,[y],[z]) \in I_{[y]}, (N[x][y])_{(i,[x],[y]),(j,[y],[z])} = 1\}.$$

By Lemma 8.6.19, we have for each of these children:

$$\mathcal{X}(\mathbf{g}_{j,[y],[z]}) = \chi^{-1}(M[y]_{(j,[y],[z]),-}).$$

Thus, the gate matrix $M\mathbf{g} \in \mathbb{F}_2^{\mathbf{g}E_C \times E}$ is a submatrix of $M[y]$: It consists precisely of those rows $(j,[y],[z])$ of $M[y]$ such that $(N[x][y])_{(i,[x],[y]),(j,[y],[z])} = 1$. Therefore,

$$\mathbf{rk}(M\mathbf{g}) \leq \mathbf{rk}(M[y]).$$

Now we can argue as in the proof of Lemma 8.4.7 in order to bound $\mathbf{rk}(M[y])$. Using the Orbit-Stabiliser Theorem, we obtain again:

$$\log(\mathbf{maxOrb}_E(\mu)) \geq |E| - \dim \mathbf{Stab}_E(y).$$

By Lemma 8.6.15, we have $\mathbf{Ker}(M[y]) = \mathbf{Stab}_E(y)$. Therefore, with the Rank Theorem we get:

$$\mathbf{rk}(M[y]) = |E| - \dim\ \mathbf{Stab}_E(y) \leq \log(\mathbf{maxOrb}_E(\mu)).$$

In total, we have $\mathbf{rk}(M\mathbf{g}) \leq \log(\mathbf{maxOrb}_E(\mu))$. Since $\mathbf{g} \in V_C$ was arbitrary, this is a bound on the fan-in dimension of $\widehat{C}(\mu)$. □

Lemma 8.6.21. *For the root r of $\widehat{C}(\mu)$, it holds*

$$|\mathcal{X}(r)| \geq \frac{|\sup_{\text{CFI}}(\mu)|}{\log(\mathbf{maxOrb}_E(\mu))}.$$

Proof. By definition of $\widehat{C}(\mu)$, we have $r = \mathbf{g}_{i,[\mu],[y]}$ for some $(i, [\mu], [y]) \in I_{[\mu]}$ such that $M[\mu]_{(i,[\mu],[y]),-}$ is a row with a maximum number of 1-entries. We have

$$\mathcal{X}(r) = \chi^{-1}(M[\mu]^T_{(i,[\mu],[y]),-})$$

according to Lemma 8.6.19. Therefore, we have to show that $M[\mu]$ has a row with at least $\frac{|\sup_{\text{CFI}}(\mu)|}{\log(\mathbf{maxOrb}_E(\mu))}$ many 1-entries.

Claim: For every $e \in \sup_{\text{CFI}}(\mu)$ there is a row of $M[\mu]$ which is non-zero in column e.
Proof of claim: Since $\sup_{\text{CFI}}(\mu)$ is the minimal CFI-support of μ, $\chi(e) \notin \mathbf{Stab}_E(\mu)$. Otherwise there is a smaller support not containing e. Suppose for a contradiction that all rows of $M[\mu]$ are zero in column e. Then $\chi(e) \in \mathbf{Ker}(M[\mu])$. But this contradicts the fact that $\mathbf{Ker}(M[\mu]) = \mathbf{Stab}_E(\mu)$ (Lemma 8.6.15). This proves the claim.
Now take a subset $\mathcal{B}$ of the rows of $M[\mu]$ that forms a basis of the row space of $M[\mu]$. By Lemma 8.6.20, we have $|\mathcal{B}| \leq \log(\mathbf{maxOrb}_E(\mu))$. For every $e \in E$ such that some row of $M[\mu]$ is non-zero in column e, there must also be a row in $\mathcal{B}$ that is non-zero in column e (else $\mathcal{B}$ does not generate the whole row space). So by the claim and by the size bound on $\mathcal{B}$, there is a row in $\mathcal{B}$ with at least $\frac{|\sup_{\text{CFI}}(\mu)|}{\log(\mathbf{maxOrb}_E(\mu))}$ many 1-entries. $\square$

So far we have shown **Properties 1**, **2** and **3** from Theorem 8.6.1. **Property 1** is proved by Corollary 8.6.18, **Property 2** by Lemma 8.6.21, and **Property 3** by Lemma 8.6.20. It remains to estimate the size of the circuit under the assumption that every space $\mathbf{Stab}_E([y] \cap x)$ has a symmetric basis.

8.6.3 Bounding the size of the circuit

Lemma 8.6.22. *Let $Atoms(\mu) \subseteq \text{tc}(\mu)$ denote the set of atoms in $\text{tc}(\mu)$, and $Sets(\mu) := \text{tc}(\mu) \setminus Atoms(\mu)$. The size of $\widehat{C}(\mu)$ is*

$$|V_C| = |\{e \in E \mid e_0 \text{ or } e_1 \in Atoms(\mu)\}| + \sum_{\substack{([x],[y]), x \in Sets(\mu), \\ [y] \in \mathcal{C}[x]}} |J_{[x][y]}|.$$

In other words: The size of $\widehat{C}(\mu)$ is determined by the total number of rows of all $N[x][y]$-matrices.

Proof. By definition of $\widehat{C}(\mu)$, $|V_C| = \sum_{[x], x \in \text{tc}(\mu)} |I_{[x]}|$. For each $x \in \text{tc}(\mu) \cap \text{Sets}(\mu)$, we have $|I_{[x]}| = \sum_{[y] \in \mathcal{C}[x]} |J_{[x][y]}|$. Every set $J_{[x][y]}$ is exclusively associated with the pair $([x], [y])$, so

$$\sum_{[x], x \in \text{tc}(\mu)} |I_{[x]}| = \sum_{\substack{([x],[y]), x \in \text{Sets}(\mu), \\ [y] \in \mathcal{C}[x]}} |J_{[x][y]}|.$$

For each atomic $x \in \mathrm{tc}(\mu)$, $|I_{[x]}| = 1$. $\square$

Hence, in order to bound $|V_C|$, we have to bound the number of rows of each of the $N[x][y]$-matrices. To do so, we revisit the proofs of Lemmas 8.6.8 and 8.6.10. We will see that the matrix that is constructed in these lemmas can be chosen to have polynomial size if a symmetric basis for $\mathbf{Stab}_E([y] \cap x)$ exists. The first step in the construction of $N[x][y]$ in Lemma 8.6.8 is to solve a family of linear equation systems. We now show that the symmetries of such systems correspond to symmetries of their solutions.

Lemma 8.6.23. *Let I, J be abstract index sets and $A, A' \in \mathbb{F}_2^{I \times J}, \mathbf{b}, \mathbf{b}' \in \mathbb{F}_2^I$ such that the linear equation systems $A \cdot \mathbf{x} = \mathbf{b}$ and $A' \cdot \mathbf{x} = \mathbf{b}'$ each have a unique solution $\mathbf{s}, \mathbf{s}'$, respectively. Let $\mathbf{G} \leq \mathbf{Sym}(J)$ be a permutation group acting on the columns of A. Let $\pi \in \mathbf{G}$ a permutation that maps the row-set of the extended coefficient matrix $(A|\mathbf{b})$*

$$R = \{(A_{i-}, \mathbf{b}_i) \mid i \in I\}$$

to the row-set R' of $(A'|\mathbf{b}')$. Here, the action of π on R is $\pi(A_{i-}, \mathbf{b}_i) := (\pi(A_{i-}), \mathbf{b}_i)$. Then $\pi(\mathbf{s}) = \mathbf{s}'$.

Proof. We only have to show that $\pi(\mathbf{s})$ is a solution of $A' \cdot \mathbf{x} = \mathbf{b}'$. Then we have $\pi(\mathbf{s}) = \mathbf{s}'$ by uniqueness of the solution. If $\pi \in \mathbf{G}$ maps R to R', then there is a permutation $\sigma \in \mathbf{Sym}(I)$ that is induced by the action of π, i.e. $\pi(A_{i-}, \mathbf{b}_i) = (A'_{\sigma i,-}, \mathbf{b}'_{\sigma i})$ for all $i \in I$. For every $i \in I$, it holds $A_{i-} \cdot \mathbf{s} = \mathbf{b}_i$, because $\mathbf{s}$ is a solution to the equation system. Since (by Proposition 8.6.3) $A_{i-} \cdot \mathbf{s} = \pi(A_{i-}) \cdot \pi(\mathbf{s}) = A'_{\sigma i-} \cdot \pi(\mathbf{s})$, and $\mathbf{b}_i = \mathbf{b}'_{\sigma i}$, $\pi(\mathbf{s})$ is a solution to the equation $A'_{\sigma i-} \cdot \mathbf{x} = \mathbf{b}'_{\sigma i}$. Since every row of $(A'|\mathbf{b}')$ has such a preimage under π, it follows that $\pi(\mathbf{s})$ is a solution for every equation in $A' \cdot \mathbf{x} = \mathbf{b}'$. By assumption, the equation system has a unique solution. Therefore, $\pi(\mathbf{s}) = \mathbf{s}'$. $\square$

The next lemma will become interesting once we are dealing with symmetric bases of vector spaces. It shows that the permutation invariance of a set of vectors (for example a symmetric basis) is preserved under linear maps and appropriate group homomorphisms.

Lemma 8.6.24. *Let $M \in \mathbb{F}_2^{I \times J}$ and let $\mathbf{G} \leq \mathbf{Sym}(J)$ be a permutation group acting on the columns of M. Let $B \subseteq \mathbb{F}_2^J$ be a set of vectors and $S := \mathbf{Stab}_{\mathbf{G}}(B)$. Let $g : \mathbf{G} \longrightarrow \mathbf{Sym}(I)$ be a group homomorphism such that for all $\pi \in \mathbf{G}$, $(g(\pi), \pi)M = M$. Then $g(S)$ stabilises the set $M \cdot B = \{M \cdot \mathbf{v} \mid \mathbf{v} \in B\}$.*

Proof. Let $\pi \in S$ and $\mathbf{v} \in B$. Let $\mathbf{w} = M \cdot \mathbf{v}$. We show that $(g(\pi))^{-1}(\mathbf{w}) = M \cdot \pi(\mathbf{v})$. For each $i \in I$, we have

$$M_{i,-} \cdot \pi(\mathbf{v}) = \pi(M_{(g(\pi))(i),-}) \cdot \pi(\mathbf{v}) = M_{(g(\pi))(i),-} \cdot \mathbf{v}.$$

The first equality holds because $(g(\pi), \pi)M = M$ by assumption and the second one is due to Proposition 8.6.3. It follows that $g(\pi)(M \cdot \pi(\mathbf{v})) = M \cdot \mathbf{v} = \mathbf{w}$. So $(g(\pi))^{-1}(\mathbf{w}) = M \cdot \pi(\mathbf{v})$. We have $\pi(\mathbf{v}) \in B$ because $\pi \in S$. Therefore, $(g(\pi))^{-1}(\mathbf{w}) \in M \cdot B$. Because $\mathbf{w} = M \cdot \mathbf{v}$ was arbitrary, we know that $(g(\pi))^{-1} = g(\pi^{-1}) \in g(S)$ stabilises the set $M \cdot B$. Since $\pi \in S$ was also arbitrary, $g(S)$ stabilises $M \cdot B$. $\square$

Finally, we provide the exact definition of what we mean by a *symmetric basis*. This definition is tailored to the spaces $\mathbf{Stab}_E([y] \cap x)$ that occur for the objects in $\mathrm{tc}(\mu)$. When we say "symmetric basis", we actually mean two bases: We require that both the basis of $\mathbf{Stab}_E([y] \cap x)$ as well as its extension to a basis of the ambient space be symmetric. Symmetry is meant in the sense that the orbit must have polynomial size.

Definition 8.6.25. *Let $x \in \mathrm{tc}(\mu)$ and $[y] \in \mathcal{C}[x]$. We say that the vector space $\Gamma := \mathbf{Stab}_E([y] \cap x) \leq \mathbb{F}_2^E$ has a* symmetric basis *if there exist two bases*

$$\mathcal{B}_\Gamma \subseteq \mathcal{B}$$

such that $\mathcal{B}_\Gamma$ is a basis of $\Gamma = \mathbf{Stab}_E([y] \cap x)$ and $\mathcal{B}$ is a basis of $\mathbb{F}_2^E$, and such that: The group

$$\mathbf{Stab}_{\mathbf{Stab}_G([y]\cap x)}(\mathcal{B}_\Gamma) \cap \mathbf{Stab}_{\mathbf{Stab}_G([y]\cap x)}(\mathcal{B}) = \\ \{\pi \in \mathbf{Stab}_G([y] \cap x) \mid \pi(\mathcal{B}_\Gamma) = \mathcal{B}_\Gamma \text{ and } \pi(\mathcal{B}) = \mathcal{B}\} \leq \mathbf{Stab}_G([y] \cap x)$$

has index $\leq \mathrm{poly}(|\mathfrak{G}^S|)$ in $\mathbf{Stab}_G([y] \cap x)$.

In the above definition, the polynomial $\mathrm{poly}(\cdot)$ is of course meant to be fixed for the whole family of CFI-instances that we are considering in Theorem 8.6.1. Now let us continue with the main lemma that bounds the size of $N[x][y]$ assuming the existence of a symmetric basis.

Lemma 8.6.26. *Let $x \in \mathrm{tc}(\mu)$ and $[y] \in \mathcal{C}[x]$. If $\mathbf{Stab}_E([y] \cap x)$ has a* symmetric basis*, then the number of rows of $N[x][y]$ is polynomial in $|\mathfrak{G}^S|$.*

Proof. We show that there is a Boolean matrix N with a polynomial number of rows which satisfies conditions (i) and (ii) from Lemma 8.6.10. This proves the lemma because $N[x][y]$ is defined as the smallest such matrix.
Consider the proof of Lemma 8.6.8. When applied to $\Gamma = M[y] \cdot \mathbf{Stab}_E([y] \cap x)$ and $\Delta = M[y] \cdot \mathbb{F}_2^E = \mathbf{Im}(M[y])$, the proof shows that there is a matrix $N \in \mathbb{F}_2^{[k] \times I_{[y]}}$, for $k = \dim(\mathbf{Im}\ M[y]) - \dim(M[y] \cdot \mathbf{Stab}_E([y] \cap x))$, such that $\mathbf{Ker}(N) \cap \mathbf{Im}(M[y]) = M[y] \cdot \mathbf{Stab}_E([y] \cap x)$. In the proof, k linear equation systems $A_i \cdot \mathbf{x} = \mathbf{b}_i$ are defined, each one with a unique solution. Then, for $i \in [k]$, the i-th row of N is defined as the unique solution to the equation system $A_i \cdot \mathbf{x} = \mathbf{b}_i$. The matrix A_i is the same for every $i \in [k]$. It depends on the choice of a basis for $M[y] \cdot \mathbf{Stab}_E([y] \cap x)$ and an extension to a basis of $\mathbf{Im}(M[y])$. More precisely, let $\{\mathbf{w}_1, ..., \mathbf{w}_k\}$ be the vectors that extend the basis of $M[y] \cdot \mathbf{Stab}_E([y] \cap x)$ to a basis of $\mathbf{Im}(M[y])$. The rows of A_i are the basis vectors of $\mathbf{Im}(M[y])$, and the vector $\mathbf{b}_i$ has exactly one 1-entry in the row corresponding to $\mathbf{w}_i$. In this way, $\mathbf{b}_i$ is defined for every $i \in [k]$.
One can check that this proof of Lemma 8.6.8 still goes through if one uses a *generating set* for the space $M[y] \cdot \mathbf{Stab}_E([y] \cap x)$ instead of a basis for the rows of A_i – as long as the extension $\{\mathbf{w}_1, ..., \mathbf{w}_k\}$ to a basis of the full space $\mathbf{Im}(M[y])$ is a linearly independent set of vectors. This changes nothing and in particular, each equation system $A_i \cdot \mathbf{x} = \mathbf{b}_i$

will still have a unique solution because we have just added some redundant equations.

We choose appropriate bases now. Since $\mathbf{Stab}_E([y] \cap x)$ has a symmetric basis by assumption, there are bases $\mathcal{B}_\Gamma \subseteq \mathcal{B}$ of $\mathbf{Stab}_E([y] \cap x)$ and $\mathbb{F}_2^E$, respectively, such that $\mathbf{Stab}_{\mathbf{Stab}_G([y]\cap x)}(\mathcal{B}_\Gamma) \cap \mathbf{Stab}_{\mathbf{Stab}_G([y]\cap x)}(\mathcal{B})$ has small index in $\mathbf{Stab}_G([y] \cap x)$. Let $\mathcal{B}'_\Gamma := M[y] \cdot \mathcal{B}_\Gamma \setminus \{\mathbf{0}\}$ and $\mathcal{B}' := M[y] \cdot \mathcal{B} \setminus \{\mathbf{0}\}$. Then $\mathcal{B}'_\Gamma$ is a generating set for $M[y] \cdot \mathbf{Stab}_E([y] \cap x)$, and $\mathcal{B}' \setminus \mathcal{B}'_\Gamma$ extends this generating set to a basis of $\mathbf{Im}(M[y])$. Importantly, $\mathcal{B}' \setminus \mathcal{B}'_\Gamma$ is a linearly independent set of vectors (while $\mathcal{B}'_\Gamma$ may be linearly dependent). This is because $\mathbf{Ker}(M[y]) = \mathbf{Stab}_E(y)$ (Lemma 8.6.15), and $\mathbf{Stab}_E(y) \leq \mathbf{Stab}_E([y] \cap x)$. Thus, if there were a subset $K \subseteq \mathcal{B}' \setminus \mathcal{B}'_\Gamma$ such that $\sum K = 0$, then the sum of the $M[y]$-preimages of the vectors in K would be in $\mathbf{Ker}(M[y]) = \mathbf{Stab}_E(y)$. This cannot be the case because $\mathcal{B}_\Gamma$ is a basis for $\mathbf{Stab}_E([y] \cap x) \geq \mathbf{Stab}_E(y)$, so no linear combination of vectors in $\mathcal{B} \setminus \mathcal{B}_\Gamma$ can be in $\mathbf{Stab}_E(y)$.

Now apply Lemma 8.6.24 to the matrix $M = M[y] \in \mathbb{F}_2^{I_{[y]} \times E}$, $\mathbf{G} = \mathbf{Stab}_G([y] \cap x), B = \mathcal{B}_\Gamma$ and the homomorphism g defined like this: $g : \mathbf{Stab}_G([y] \cap x) \longrightarrow \mathbf{Sym}(I_{[y]})$ maps any $\pi \in \mathbf{Stab}_G([y] \cap x)$ to the $I_{[y]}$-restriction of the permutation $g_{[y]}(\pi) \in \mathbf{Sym}(I_{\Omega_{[y]}})$. This is well-defined and $g(\pi) \in \mathbf{Sym}(I_{[y]})$ because $\mathbf{Stab}_G([y] \cap x) \leq \mathbf{Stab}_G([y])$ by Corollary 8.6.5, and $g_{[y]}(\mathbf{Stab}_G([y]))$ maps $I_{[y]}$ to itself by Lemma 8.6.16.
Now it follows with Lemma 8.6.24 that $g(\mathbf{Stab}_{\mathbf{Stab}_G([y]\cap x)}(\mathcal{B}_\Gamma)) \leq \mathbf{Sym}(I_{[y]})$ stabilises the set $M[y] \cdot \mathcal{B}_\Gamma \subseteq \mathbb{F}_2^{I_{[y]}}$. It holds $\mathcal{B}'_\Gamma = M[y] \cdot \mathcal{B}_\Gamma \setminus \{\mathbf{0}\}$, and the zero-vector forms a singleton orbit with respect to permutations of the entries, so also $\mathcal{B}'_\Gamma$ is stabilised by $g(\mathbf{Stab}_{\mathbf{Stab}_G([y]\cap x)}(\mathcal{B}_\Gamma))$. Similarly, by applying Lemma 8.6.24 to $B = \mathcal{B}$, we get that $g(\mathbf{Stab}_{\mathbf{Stab}_G([y]\cap x)}(\mathcal{B}))$ stabilises $\mathcal{B}'$.
Every permutation $\pi \in \mathbf{Sym}(I_{[y]})$ that stabilises the sets $\mathcal{B}'_\Gamma$ and $\mathcal{B}'$ induces a permutation $\sigma \in \mathbf{Sym}(\mathcal{B}')$, whose restriction to $\mathcal{B}'_\Gamma$ is a permutation in $\mathbf{Sym}(\mathcal{B}'_\Gamma)$. Since the rows of each coefficient matrix A_i are the vectors in $\mathcal{B}'$, every $\pi \in \mathbf{Sym}(I_{[y]})$ that fixes $\mathcal{B}'$ setwise reorders the rows of A_i. The right hand side $\mathbf{b}_i$ has only one 1-entry in the row corresponding to $\mathbf{w}_i \in \mathcal{B}' \setminus \mathcal{B}'_\Gamma$. So if π also stabilises $\mathcal{B}'_\Gamma$, the action of π on $(A_i|\mathbf{b}_i)$ moves the row vector $\pi^{-1}(\mathbf{w}_i) \in \mathcal{B}' \setminus \mathcal{B}'_\Gamma$ to the row where $\mathbf{b}_i$ has its 1-entry. Up to a reordering of rows, this yields one of the other linear equation systems $\{(A_i|\mathbf{b}_i) \mid i \in [k]\}$, because in some linear equation system, the equation with coefficient vector $\pi^{-1}(\mathbf{w}_i)$ has a 1 on the right hand side. So any π that stabilises both $\mathcal{B}'_\Gamma$ and $\mathcal{B}'$ induces a permutation on the set of linear equation systems $\{(A_i|\mathbf{b}_i) \mid i \in [k]\}$ in the sense of Lemma 8.6.23 (with the action of column permutations on the row set of an equation system as defined there).
Let

$$\mathcal{S} := g(\mathbf{Stab}_{\mathbf{Stab}_G([y]\cap x)}(\mathcal{B}_\Gamma)) \cap g(\mathbf{Stab}_{\mathbf{Stab}_G([y]\cap x)}(\mathcal{B})).$$

As we argued above, this group fixes both $\mathcal{B}'$ and $\mathcal{B}'_\Gamma$. Therefore, it induces a permutation on the equation systems and so Lemma 8.6.23 tells us that $\mathcal{S}$ also induces a corresponding permutation on the set

$$\{\mathbf{s}_i \in \mathbb{F}_2^{I_{[y]}} \mid \mathbf{s}_i \text{ is the unique solution to } A_i \cdot \mathbf{x} = \mathbf{b}_i\}$$

(where $\mathcal{S}$ acts on these vectors by permuting the entries). These solution vectors form exactly the rows of the matrix $N \in \mathbb{F}_2^{[k]\times I_{[y]}}$ that is being constructed in the proof of Lemma 8.6.8. Therefore, the group $\mathcal{S}$ acting on the columns of N induces corresponding permutations on the rows of N. In other words, for every $\pi \in \mathcal{S}$ there is a $\sigma \in \mathbf{Sym}_k$ such that $(\sigma, \pi)N = N$. Now we close the rows of N under the action of $g(\mathbf{Stab}_G([y] \cap x))$, exactly like in the proof of Lemma 8.6.10. This yields a matrix satisfying the desired conditions (a) and (b). We now argue that for each row of N, only $\text{poly}(k + |\mathfrak{G}^S|)$ many rows are added to form the closure under $g_{[y]}(\mathbf{Stab}_G([y] \cap x))$.

To show this, we have to bound the size of $\mathbf{Orb}(N_{i-}) = \{\pi(N_{i-}) \mid \pi \in g(\mathbf{Stab}_G([y] \cap x))\}$. Let $\mathbf{Orb}(N)$ denote the $g(\mathbf{Stab}_G([y] \cap x))$-orbit of the set of rows of N, and $\mathbf{Stab}(N) \leq g(\mathbf{Stab}_G([y] \cap x))$ the setwise stabiliser of the set of rows. It holds $|\mathbf{Orb}(N_{i-})| \leq k \cdot |\mathbf{Orb}(N)|$ because any image of the row N_{i-} is an element of at least one of the k-element row sets in $\mathbf{Orb}(N)$. Together with the Orbit-Stabiliser Theorem, we get

$$|\mathbf{Orb}(N_{i-})| \leq k \cdot |\mathbf{Orb}(N)| = k \cdot \frac{|g(\mathbf{Stab}_G([y] \cap x))|}{|\mathbf{Stab}(N)|}.$$

By what we argued above, we have $\mathcal{S} \leq \mathbf{Stab}(N)$ and thus

$$\begin{aligned}\frac{|g(\mathbf{Stab}_G([y] \cap x))|}{|\mathbf{Stab}(N)|} &\leq \frac{|g(\mathbf{Stab}_G([y] \cap x))|}{|\mathcal{S}|} \\ &\leq \frac{|g(\mathbf{Stab}_G([y] \cap x))|}{|g(\mathbf{Stab}_{\mathbf{Stab}_G([y]\cap x)}(\mathcal{B}_\Gamma) \cap \mathbf{Stab}_{\mathbf{Stab}_G([y]\cap x)}(\mathcal{B}))|}.\end{aligned}$$

The last inequality holds because $g(\mathbf{Stab}_{\mathbf{Stab}_G([y]\cap x)}(\mathcal{B}_\Gamma) \cap \mathbf{Stab}_{\mathbf{Stab}_G([y]\cap x)}(\mathcal{B})) \leq \mathcal{S}$. It is known (see Lemma 4.1.1) that the application of a group homomorphism can only decrease the index of H in G, i.e. $[h(G) : h(H)] \leq [G : H]$. According to Definition 8.6.25, the index of $\mathbf{Stab}_{\mathbf{Stab}_G([y]\cap x)}(\mathcal{B}_\Gamma) \cap \mathbf{Stab}_{\mathbf{Stab}_G([y]\cap x)}(\mathcal{B})$ in $\mathbf{Stab}_G([y] \cap x)$ is polynomially bounded in$|\mathfrak{G}^S|$. As g is a group homomorphism, this is also a bound for the index of the image under g, which is equal to the fraction above. In total, we have shown:

$$|\mathbf{Orb}(N_{i-})| \leq k \cdot \text{poly}(|\mathfrak{G}^S|) \leq \text{poly}(k + |\mathfrak{G}^S|).$$

Now this orbit bound applies to each of the k rows of N, so when closing the rows of N under the action of $g(\mathbf{Stab}_G([y]\cap x))$, we add at most $\text{poly}(k+|\mathfrak{G}^S|)$ many new rows to the matrix. Because $k = \dim(\mathbf{Im}\ M[y]) - \dim(M[y] \cdot \mathbf{Stab}_E([y] \cap x)) \leq \dim(\mathbf{Im}\ M[y]) \leq |E|$ (this holds because $M[y]$ is a linear map defined on the $|E|$-dimensional space $\mathbb{F}_2^E$), $\text{poly}(k + |\mathfrak{G}^S|) = \text{poly}(|\mathfrak{G}^S|)$. The resulting matrix is a candidate for $N[x][y]$, so this shows that $N[x][y]$ has at most $\text{poly}(|\mathfrak{G}^S|)$ many rows. □

Lemma 8.6.27. *If for all $x \in \text{tc}(\mu)$ and all $[y] \in \mathcal{C}[x]$, $\mathbf{Stab}_E([y] \cap x)$ has a symmetric basis, then $|V_C|$ has size polynomial in $|\mathfrak{G}^S|$.*

Proof. This follows directly from Lemma 8.6.22 and Lemma 8.6.26 (remember that $|J_{[x][y]}|$ is the number of rows of $N[x][y]$), and from the fact that $|\text{tc}(\mu)|$ is polynomial in $|\mathfrak{G}^S|$ because μ is CPT-definable in $\mathfrak{G}^S$. □

This proves **Property 4** from Theorem 8.6.1.

8.6.4 Which vector spaces have symmetric bases?

We have shown that the size of $\widehat{C}(\mu)$ can be polynomially bounded if for all $x \in \mathrm{tc}(\mu)$ and all $[y] \in \mathcal{C}[x]$, the stabiliser space $\mathbf{Stab}_E([y] \cap x)$ has a *symmetric basis*. If this is not the case, then we do not know anything about the size of $\widehat{C}(\mu)$. There may be other ways to bound it but a priori we have to assume that it is super-polynomial then. This makes these symmetric XOR-circuits less useful for deriving lower bounds against CPT. We would have to show that for a certain family of base graphs, all families of symmetric circuits with the properties mentioned in Theorem 8.6.1 are necessarily larger than $\widehat{C}(\mu)$, which is difficult if we do not even know precisely how large $\widehat{C}(\mu)$ is. Also, if $\widehat{C}(\mu)$ is very large, then it is less likely that the size contradicts the symmetry requirement of the circuit. Therefore, Theorem 8.6.1 is most interesting when the size of $\widehat{C}(\mu)$ is polynomially bounded in the size of the respective CFI-instance because then, super-polynomial lower bounds on the size of the required symmetric circuit families would imply that CPT cannot decide the CFI-query using the object μ. The question is: Which objects μ satisfy the symmetric basis condition for all $\mathbf{Stab}_E([y] \cap x)$? As we remarked earlier, the *CFI-symmetric* objects from the previous section are a special case where the symmetric basis condition holds. We prove this now to justify that our Definition 8.6.25 is "right".

In the following, whenever we have a Boolean vector space $\mathbb{F}_2^I$ over some index-set I, then $\widetilde{\mathbb{F}}_2^I$ denotes the *even subspace* of $\mathbb{F}_2^I$, i.e. the subspace consisting of all vectors with even Hamming weight. Moreover, $\mathbb{F}_2^I \oplus \mathbb{F}_2^J$ denotes the direct sum of the two vector spaces. If I and J are disjoint, then $\mathbb{F}_2^I \oplus \mathbb{F}_2^J$ is simply equal to $\mathbb{F}_2^{I \cup J}$.

Lemma 8.6.28. *Let $\mu \in \mathrm{HF}(\widehat{E})$ be CFI-symmetric. Then μ satisfies the symmetric basis condition from Definition 8.6.25.*

Proof. Let $x \in \mathrm{tc}(\mu)$ and $[y] \in \mathcal{C}[x]$. Let $\Gamma := \mathbf{Stab}_E([y] \cap x) \leq \mathbb{F}_2^E$. We have to define two bases $\mathcal{B}_\Gamma \subseteq \mathcal{B}$ of Γ and of $\mathbb{F}_2^E$, respectively, such that the group $\mathbf{Stab}_{\mathbf{Stab}_G([y] \cap x)}(\mathcal{B}_\Gamma) \cap \mathbf{Stab}_{\mathbf{Stab}_G([y] \cap x)}(\mathcal{B})$ has polynomial index in $\mathbf{Stab}_G([y] \cap x)$. Since μ is CFI-symmetric, by Definition 8.2.2, the $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})$-orbit of $[y] \cap x$ has size exactly two. We have $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}) \cong \mathbb{F}_2^E$, so by the Orbit-Stabiliser Theorem, $\mathbf{Stab}_E([y] \cap x)$ is a subspace of $\mathbb{F}_2^E$ with co-dimension one. We use this to analyse the structure of the space $\mathbf{Stab}_E([y] \cap x)$. The group $\mathbf{Stab}_G([y] \cap x)$ is a subgroup of $\mathbf{Aut}(G) \leq \mathbf{Sym}(V)$ and therefore also acts on the edge set E. Thus, we can partition E into its $\mathbf{Stab}_G([y] \cap x)$-orbits. Let $\mathcal{P} = \{P_1, ..., P_m\}$ denote this orbit partition of E.

Claim: There is a partition $E = A \uplus B$ such that $\mathbf{Stab}_E([y] \cap x) = \mathbb{F}_2^A \oplus \widetilde{\mathbb{F}}_2^B$ and $B \neq \emptyset$. Moreover, A and B are unions of $\mathbf{Stab}_G([y] \cap x)$-orbits.
Proof of claim. Let $\mathcal{P}' \subseteq \mathcal{P}$ denote the set of orbits P_i such that for any $e \in P_i$, the unit vector $\chi(e)$ is in $\mathbf{Stab}_E([y] \cap x)$. Note that whenever $\chi(e) \in \mathbf{Stab}_E([y] \cap x)$, then $\chi(e') \in \mathbf{Stab}_E([y] \cap x)$ for every e' in the orbit of E because $\mathbf{Stab}_G([y] \cap x)$ is transitive on each orbit and the space $\mathbf{Stab}_E([y] \cap x)$ is invariant under the action of $\mathbf{Stab}_G([y] \cap x)$ on the coordinates (Lemma 8.6.7). We let $A := \bigcup \mathcal{P}'$ and $B := E \setminus A$. It remains to show that $\widetilde{\mathbb{F}}_2^B$ is a subspace of $\mathbf{Stab}_E([y] \cap x)$. Assume for a contradiction that there is some

vector $\mathbf{v} \in \mathbb{F}_2^E$ with even Hamming weight on B and zero on A which is not contained in $\mathbf{Stab}_E([y] \cap x)$. Since $\mathbf{Stab}_E([y] \cap x)$ has co-dimension exactly one in $\mathbb{F}_2^E$, and since moreover, by definition of $\mathcal{P}'$, no unit vector $\chi(e)$ with $e \in B$ is in $\mathbf{Stab}_E([y] \cap x)$, we know that for any such unit vector $\chi(e)$ with $e \in B$, there exists some $\mathbf{w}_e \in \mathbf{Stab}_E([y] \cap x)$ such that $\mathbf{v} = \mathbf{w}_e + \chi(e)$. But then, every vector with Hamming-weight exactly two on B is in $\mathbf{Stab}_E([y] \cap x)$. Namely, for any two $e, e' \in B$, it then holds that $\mathbf{w}_e + \mathbf{w}_{e'} = \chi(e) + \chi(e')$, and we have $\mathbf{w}_e, \mathbf{w}_{e'} \in \mathbf{Stab}_E([y] \cap x)$. So then, $\mathbf{Stab}_E([y] \cap x)$ does contain $\widetilde{\mathbb{F}}_2^B$. In total, this proves the claim (it holds $B \neq \emptyset$ because otherwise, the co-dimension would be zero).

Now it is not hard to define a symmetric basis for $\mathbf{Stab}_E([y] \cap x)$. Fix an arbitrary edge $f \in B$. We define

$$\mathcal{B}_\Gamma := \{\chi(e) \mid e \in A\} \cup \{\chi(\{e, f\}) \mid e \in B \setminus \{f\}\}.$$

One can check that this is indeed a basis of $\mathbb{F}_2^A \oplus \widetilde{\mathbb{F}}_2^B$. The basis $\mathcal{B}$ of $\mathbb{F}_2^E$ is then simply defined as $\mathcal{B} := \mathcal{B}_\Gamma \cup \{\chi(f)\}$.
Now the group $\mathbf{Stab}_{\mathbf{Stab}_G([y] \cap x)}(\mathcal{B}_\Gamma) \cap \mathbf{Stab}_{\mathbf{Stab}_G([y] \cap x)}(\mathcal{B})$ contains all permutations in $\mathbf{Stab}_G([y] \cap x)$ that fix the edge f and fix the sets A and B (setwise). By the Claim, A and B are unions of $\mathbf{Stab}_G([y] \cap x)$-orbits, so the latter condition is fulfilled by all permutations in $\mathbf{Stab}_G([y] \cap x)$. Therefore, $\mathbf{Stab}_{\mathbf{Stab}_G([y] \cap x)}(\mathcal{B}_\Gamma) \cap \mathbf{Stab}_{\mathbf{Stab}_G([y] \cap x)}(\mathcal{B})$ is simply the pointwise stabiliser of f in $\mathbf{Stab}_G([y] \cap x) \leq \mathbf{Sym}(E)$, and this has index at most $|E|$. This is polynomial in $|\mathfrak{G}^S|$. □

Thus, we have shown that all CFI-symmetric h.f. sets satisfy the symmetric basis property from Definition 8.6.25. But are there any other objects that have symmetric bases? The answer is affirmative. To keep things simple, we do not give a fully specified example but only sketch how a family of *non-CFI-symmetric* h.f. sets *with symmetric bases* may look like.

Example 8.6.29. *For $n \in \mathbb{N}$, let E_n denote an n-element set of base edges. We do not fix a specific family $(G_n = (V_n, E_n))_{n \in \mathbb{N}}$ of base graphs. Let $A_n \uplus B_n \uplus \{e\} = E_n$ be an arbitrary partition of the edge set such that one part is a singleton. Consider again Example 8.2.1. There, we defined the CFI-symmetric object $\mu_{\{e,f,g\}} = \{\{\mu_{\{f,g\}}, e_0\}, \{\widetilde{\mu}_{\{f,g\}}, e_1\}\}$. In this construction, $\mu_{\{f,g\}}$ and its automorphic image $\widetilde{\mu}_{\{f,g\}}$ are sets that are stabilised by every $\rho \in \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G})$ that flips an even number of edges in $\{f, g\}$. Such an object can indeed be defined, for example as described in Section 6.1. So let more generally $\mu_{B_n}, \widetilde{\mu}_{B_n}$ denote two sets that form an $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}_n)$-orbit and are stabilised by any $\rho \in \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}_n)$ flipping an even number of edges in B_n. In other words, $\mathbf{Stab}_E(\mu_{B_n}) = \mathbb{F}_2^{A_n \cup \{e\}} \oplus \widetilde{\mathbb{F}}_2^{B_n}$. Now for every $n \in \mathbb{N}$, let*

$$\mu_n := \{\{\mu_{B_n}, e_0\}\} \in \mathrm{HF}(\widehat{E}_n).$$

This object is similar to the one from Example 8.2.1, with the difference that the connected component of $\{\mu_{B_n}, e_0\}$ contains just this set itself, and therefore, the $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}_n)$-orbit of this component has size four instead of two. Thus, μ_n is not *CFI-symmetric. However,* *it does satisfy the symmetric basis property (assuming that μ_{B_n} does – which is possible since μ_{B_n} could e.g. be CFI-symmetric). Let $y = \{\mu_{B_n}, e_0\}$ and $x = \mu_n$. Then $[y] \cap x = y$, and hence $\textbf{Stab}_E([y] \cap x) = \textbf{Stab}_E(y) = \textbf{Stab}_E(\{\mu_{B_n}, e_0\})$. It is not hard to construct a symmetric basis for this space. By the properties of μ_{B_n}, we have*

$$\textbf{Stab}_E(y) \cong \mathbb{F}_2^{A_n} \oplus \widetilde{\mathbb{F}}_2^{B_n}.$$

In other words, this space contains every vector that has even Hamming weight on B_n and a zero entry at coordinate e. A basis for this can be defined as in the proof of Lemma 8.6.28: Fix some $f \in B_n$. Then include in the basis $\mathcal{B}_\Gamma$ every unit vector $\chi(g)$ for $g \in A_n$ and the vector $\chi(\{f, g\})$ for each $g \in B_n \setminus \{f\}$. Let $\mathcal{B} := \mathcal{B}_\Gamma \cup \{\chi(e), \chi(f)\}$. We have not specified the base graphs exactly, so we have not made any assumptions on $\textbf{Aut}(G)$. Suppose now that A_n and B_n are not part of the same orbit of $\textbf{Stab}_G([y] \cap x)$. This makes sense because otherwise, μ_{B_n} would not necessarily be stabilised. Then $\textbf{Stab}_{\textbf{Stab}_G([y]\cap x)}(\mathcal{B}_\Gamma) \cap \textbf{Stab}_{\textbf{Stab}_G([y]\cap x)}(\mathcal{B})$ is the pointwise stabiliser of $\{e, f\}$ in $\textbf{Stab}_G([y] \cap x)$. This has index $\leq n^2$, which is polynomial.

Objects with symmetric bases are therefore indeed a strict generalisation of CFI-symmetric objects. Nonetheless, there currently exists no choiceless algorithm for the CFI-query that requires the construction of objects which go beyond the CFI-symmetric ones, and it is not clear whether such non-CFI-symmetric objects are actually algorithmically more useful than CFI-symmetric ones.

Finally, the most important question is whether there also exist objects that are neither CFI-symmetric nor have symmetric bases. Ideally, we would like the answer to be that *every* CPT-definable object μ satisfies the symmetric basis condition. Then, super-polynomial size lower bounds against suitable circuits would actually separate CPT from PTIME because they would rule out *any* CPT-algorithm for the CFI-query, not just special algorithms like the CFI-symmetric ones. We do not know if this ideal situation is in fact reality. However, we have an example that suggests it is not.

The line of thought is this: An obvious way to show that every CPT-definable object μ has the "symmetric basis property" would be to try and exploit the fact that for every $x \in \text{tc}(\mu)$ and every $[y] \in \mathcal{C}[x]$, $\textbf{Stab}_E([y] \cap x)$ must have a polynomial index in $\textbf{Aut}_{\text{CFI}}(\mathfrak{G}^S) \leq \mathbb{F}_2^E$ (otherwise, the orbit of $[y] \cap x \subseteq \text{tc}(\mu)$ would be super-polynomial, so μ would not be CPT-definable). This is perhaps the most obvious consequence that follows from the CPT-definability of μ. To simplify things a bit, let us assume that $\textbf{Aut}_{\text{CFI}}(\mathfrak{G}^S) = \mathbb{F}_2^E$. Then in terms of vector spaces, $[\mathbb{F}_2^E : \textbf{Stab}_E([y] \cap x)]$ being polynomial means that the codimension of $\textbf{Stab}_E([y] \cap x)$ in $\mathbb{F}_2^E$, i.e. $|E| - \dim\ \textbf{Stab}_E([y] \cap x)$, is logarithmic. What we also know by Lemma 8.6.7 is that the space $\textbf{Stab}_E([y] \cap x)$ is invariant under the action of the permutation group $\textbf{Stab}_G([y] \cap x)$. This leads to the question if these two restrictions on $\textbf{Stab}_E([y] \cap x)$ are sufficient to show that $\textbf{Stab}_E([y] \cap x)$ necessarily has a symmetric basis in the sense of Definition 8.6.25? Unfortunately, the answer is no. There is a family of Boolean vector spaces together with permutation groups on their index sets such that the spaces are invariant under the permutations, have at most logarithmic codimension in the ambient space, and do not admit a symmetric basis. We construct such an example in Lemma 8.6.30 below. From this it does not follow directly that there are actually families of CFI-graphs and CPT-definable h.f. sets over them which do not have the symmetric basis property. It just means that we cannot show the symmetric basis property for general CPT-definable sets with arguments that are only based on the obvious properties of vector spaces which can occur as $\mathbf{Stab}_E([y] \cap x)$ in CPT-definable objects.

Lemma 8.6.30. *There exists a family of Boolean vector spaces $(\Gamma_n)_{n \in \mathbb{N}}$, a function $t(n) \in \Theta(n)$ with $\Gamma_n \leq \mathbb{F}_2^{t(n)}$, and a family of permutation groups $(\mathbf{G}_n)_{n \in \mathbb{N}}$ with $\mathbf{G}_n \leq \mathbf{Sym}_{t(n)}$ such that*

1. *Γ_n is $\mathbf{G}_n$-invariant.*
2. *The codimension of Γ_n in $\mathbb{F}_2^{t(n)}$ is $\mathcal{O}(\log n)$.*
3. *For any pair of bases $\mathcal{B}_\Gamma \subseteq \mathcal{B}$ such that $\mathcal{B}_\Gamma$ is a basis of Γ_n and $\mathcal{B}$ is a basis of $\mathbb{F}_2^{t(n)}$, $[\mathbf{G}_n : \mathbf{Stab}_{\mathbf{G}_n}(\mathcal{B})] \geq \left(\frac{n}{(\log n)^2}\right)^{\log n}$, which is super-polynomial in n.*

Proof. Define $t(n)$ as the next even natural number $\geq n$. We now construct Γ_n and $\mathbf{G}_n$. Let $\mathcal{P}_n$ be a partition of $[t(n)]$ into $\approx \log n$ many parts such that each part is roughly of the same size, namely $\approx \frac{n}{\log n}$. Importantly, every part must be of even size; such a partition exists because $t(n)$ is even. For a part $P \in \mathcal{P}_n$, let $\widetilde{\mathbb{F}}_2^P \leq \mathbb{F}_2^{t(n)}$ denote the Boolean vector space that contains all vectors whose projection to P has even Hamming weight and which are zero outside of P. Then we define

$$\Gamma_n := \bigoplus_{P \in \mathcal{P}_n} \widetilde{\mathbb{F}}_2^P.$$

In other words, Γ_n contains exactly those vectors that have even Hamming weight on each of the parts in $\mathcal{P}_n$ (but not all vectors with even Hamming weight in $\mathbb{F}_2^{t(n)}$, namely not the vectors which are odd on an even number of parts). The permutation group $\mathbf{G}_n \leq \mathbf{Sym}_{t(n)}$ is defined as the largest group that setwise stabilises the partition $\mathcal{P}_n$. So $\mathbf{G}_n$ contains the direct product $\mathbf{H_n} := \prod_{P \in \mathcal{P}_n} \mathbf{Sym}(P)$ and all permutations that map each part of $\mathcal{P}_n$ to another part.

It is clear that Γ_n is invariant under $\mathbf{G}_n$. Furthermore, the codimension of Γ_n is logarithmic in $t(n) \approx n$: Suppose $\mathcal{B}_\Gamma$ is any basis of Γ_n. Then it can be extended to a basis of $\mathbb{F}_2^{t(n)}$ by adding one unit vector e_P for each part $P \in \mathcal{P}_n$, such that e_P has a 1-entry in P and is zero otherwise. The number of parts is logarithmic, so the same holds for the codimension. Finally, we have to prove the third condition.
Let $\mathcal{B}_\Gamma \subseteq \mathcal{B}$ be arbitrary bases for Γ and $\mathbb{F}_2^{t(n)}$, respectively. Observe that $\mathbf{Stab}_{\mathbf{G}_n}(\mathcal{B}) \leq \mathbf{Stab}_{\mathbf{G}_n}(\mathcal{B} \setminus \mathcal{B}_\Gamma)$ because Γ_n is $\mathbf{G}_n$-invariant and so, the vectors in $\mathcal{B} \setminus \mathcal{B}_\Gamma$ cannot be moved into Γ. Therefore: $[\mathbf{G}_n : \mathbf{Stab}(\mathcal{B} \setminus \mathcal{B}_\Gamma)] \leq [\mathbf{G}_n : \mathbf{Stab}(\mathcal{B})]$. Thus, it suffices to show the desired lower bound for $[\mathbf{G}_n : \mathbf{Stab}(\mathcal{B} \setminus \mathcal{B}_\Gamma)]$. Let $\mathbf{w}_1, ..., \mathbf{w}_{\log n}$ be an enumeration of

$\mathcal{B} \setminus \mathcal{B}_\Gamma$. For each $i \leq \log n$, let $\mathcal{Q}_i \subseteq \mathcal{P}_n$ denote the set of parts $P \in \mathcal{P}_n$ such that $\mathbf{w}_i$ has odd Hamming weight on P. We know that for each i, $\mathcal{Q}_i \neq \emptyset$ because otherwise, $\mathbf{w}_i$ would be in Γ. Moreover, each $P \in \mathcal{P}_n$ is in at least one of the $\mathcal{Q}_i$ because otherwise, $\mathcal{B}$ would not generate the whole space $\mathbb{F}_2^{t(n)}$.
Now in order to estimate $|\mathbf{Stab}(\mathcal{B} \setminus \mathcal{B}_\Gamma)|$, we first estimate the size of the pointwise stabiliser of $\mathcal{B} \setminus \mathcal{B}_\Gamma$ in $\mathbf{H}_n$, $\mathbf{Stab}^\bullet_{\mathbf{H}_n}(\mathcal{B} \setminus \mathcal{B}_\Gamma)$. This is the subgroup of $\mathbf{H}_n$ that stabilises each $\mathbf{w}_i \in \mathcal{B} \setminus \mathcal{B}_\Gamma$, so it consists of all permutations that fix each part $P \in \mathcal{P}_n$ and each $\mathbf{w}_i$. We can bound this stabiliser as follows:

$$|\mathbf{Stab}^\bullet_{\mathbf{H}_n}(\mathcal{B} \setminus \mathcal{B}_\Gamma)| \leq \prod_{P \in \mathcal{P}_n} (|P|-1)!$$

This holds because for each part $P \in \mathcal{P}_n$, there is a vector $\mathbf{w}_i \in \mathcal{B} \setminus \mathcal{B}_\Gamma$ which has odd weight on P. Since each part P has even size, $\mathbf{w}_i$ is not the all-1-vector on P (nor the all-zero vector, of course); therefore, the vector $\mathbf{w}_i$ is not fixed by all permutations in $\mathbf{Sym}(P)$ but at most by $(|P|-1)!$ many of them (more precisely by $k! \cdot (|P|-k)!$ many, if k is the number of 1-entries in P – but this is at most $(|P|-1)!$). Now because $[\mathbf{G}_n : \mathbf{H}_n] = (\log n)!$, we have

$$|\mathbf{Stab}^\bullet_{\mathbf{G}_n}(\mathcal{B} \setminus \mathcal{B}_\Gamma)| \leq (\log n)! \cdot |\mathbf{Stab}^\bullet_{\mathbf{H}_n}(\mathcal{B} \setminus \mathcal{B}_\Gamma)|.$$

Furthermore, $[\mathbf{Stab}_{\mathbf{G}_n}(\mathcal{B} \setminus \mathcal{B}_\Gamma) : \mathbf{Stab}^\bullet_{\mathbf{G}_n}(\mathcal{B} \setminus \mathcal{B}_\Gamma)] \leq (\log n)!$. So in total, we get:

$$\mathbf{Stab}_{\mathbf{G}_n}(\mathcal{B} \setminus \mathcal{B}_\Gamma) \leq (\log n)!^2 \cdot \prod_{P \in \mathcal{P}_n} (|P|-1)!$$

Since $|\mathbf{G}_n| = (\log n)! \cdot \prod_{P \in \mathcal{P}_n} |P|!$, we get for the index:

$$[\mathbf{G}_n : \mathbf{Stab}_{\mathbf{G}_n}(\mathcal{B})] \geq [\mathbf{G}_n : \mathbf{Stab}_{\mathbf{G}_n}(\mathcal{B} \setminus \mathcal{B}_\Gamma)] \geq 1/(\log n)! \cdot \prod_{P \in \mathcal{P}_n} |P| \geq \left(\frac{n}{(\log n)^2}\right)^{\log n}.$$

The last inequality follows because $\mathcal{P}_n$ consists of $\log n$ many parts of size $\frac{n}{\log n}$ each, and because $(\log n)! \leq (\log n)^{\log n}$. □

We do not know if there actually exist CFI-graphs $(\mathfrak{G}^S_n)_{n \in \mathbb{N}}$ and h.f. sets $(\mu_n)_{n \in \mathbb{N}}$ over them in which $\Gamma_n \cong \mathbf{Stab}_E([y] \cap x)$ for some $x, y \in \mathrm{tc}(\mu_n)$, and $\mathbf{G}_n \cong \mathbf{Stab}_G([y] \cap x)$. This could a priori be the case. Anyway, we can conclude that the question whether a CPT-definable object μ over some CFI-instance $\mathfrak{G}^S$ admits symmetric bases for all relevant spaces $\mathbf{Stab}_E([y] \cap x)$ cannot be answered without using further information about $\mathfrak{G}^S$ and μ: It seems that CPT-definability of the objects is not sufficient to infer the existence of the required symmetric bases (or this requires more sophisticated techniques than just using the logarithmic bound on the codimension). Because we are already very "far down the tree" at this point, we will not study these questions further and just note that in general, our construction of $\widehat{C}(\mu)$ may have super-polynomial size if we cannot prove that the object μ satisfies the symmetric basis condition.

8.7 Conclusion and future research

In this chapter, we have established a connection between the CPT-definability of the CFI-query on a given family of base graphs and the existence of certain families of symmetric XOR-circuits. The CFI-query can only be CPT-definable by CFI-symmetric algorithms – in particular by all the known ones – if the corresponding symmetric circuits with the properties mentioned in Theorem 8.5.2 exist. In short, these circuits must have polynomial size and orbit size, with respect to the automorphism group of the base graphs, be sensitive to a large number of input bits and satisfy a certain fan-in restriction on the gates. This *fan-in dimension* is a generalisation of the more standard fan-in degree. Property 5 in Theorem 8.5.2 makes a stronger statement about this than Property 4, but Property 5 only takes effect when we additionally restrict our attention to *super-symmetric* CFI-symmetric algorithms (which still includes all the known ones) or if we choose a family of base graphs with sufficiently high connectivity.

Lower bound results for other kinds of symmetric circuits [41, 43, 96, 70] suggest that often, symmetry requirements for circuits necessarily entail large size. Therefore, we believe that showing the non-existence of the required circuit families may be a viable approach to rule out the existence of *CFI-symmetric* CPT-algorithms for the CFI-query on suitable unordered base graphs. It is quite clear that we really need all the circuit properties from Theorem 8.5.2 for this approach to have any chance of success. If any of these restrictions on the circuits were missing, then without any doubt, the required circuit families would always exist, so no non-definability result for CPT could be inferred from this. For example, without any fan-in restriction, a circuit that connects every input bit to a single output gate would fulfil all requirements. Only with all properties taken together, it is a priori not clear that such circuit families exist for all choices of base graphs. In the next chapter, we will look at CFI-graphs over hypercubes (once again) and show an almost good enough lower bound against them. This is perhaps a first step towards a non-definability result based on Theorem 8.5.2.

Before we move on to that next chapter, we should not forget to recapitulate also the generalisation of Theorem 8.5.2 that allows us to drop the restriction to CFI-symmetric sets. This result is summarised in Theorem 8.6.2. It turned out that the connection between these h.f. objects and XOR-circuits is not just a consequence of the somewhat arbitrary-seeming notion of CFI-symmetry, but in fact hinges on the interplay between the "local stabiliser spaces" occurring in the h.f. set and the automorphism group of the base graph. This led to the notion of *symmetric bases* (Definition 8.6.25) that the relevant vector spaces must admit if they are to be encoded as gadgets of a symmetric XOR-circuit. Theorem 8.6.2 allows us to argue in principle against *all* CPT-algorithms for the CFI-query by disproving the existence of the respective families of XOR-families. The caveat is of course that it is unclear which h.f. sets satisfy the symmetric basis condition. If it were satisfied by all CPT-definable objects, then Theorem 8.6.2 could indeed be used against all CPT-programs. Unfortunately, Lemma 8.6.30 makes this seem unlikely.

To sum up, here is a plan how the results from this chapter could be used concretely in order to generate non-definability results for the CFI-query in CPT.

1. Choose a family $(G_n)_{n\in\mathbb{N}}$ of base graphs with super-constant treewidth $\mathbf{tw}_n$ such that every CFI-instance $\mathfrak{G}_n^S$ is $\mathcal{C}^{\mathbf{tw}_n}$-homogeneous and where the *CFI-support gap* (see Definition 8.5.1) is small compared to $\mathbf{tw}_n$. Moreover, G_n should be as symmetric possible to make the next step easier:

2. Try to prove that there exists no family of symmetric XOR-circuits $(C_n)_{n\in\mathbb{N}}$ over the graphs $(G_n)_{n\in\mathbb{N}}$ that satisfies all properties from Theorem 8.5.2 together. The more symmetric the graphs G_n are, the more likely it is that such a non-existence proof for the circuits can indeed be found (because intuitively, symmetric XOR-circuits with bounded "degree" that are sensitive to a large proportion of the input bits should be big).

3. Once the non-existence of the relevant circuits is established, it follows that no *CFI-symmetric* CPT-algorithm decides the CFI-query on the base graphs $(G_n)_{n\in\mathbb{N}}$ (Theorem 8.5.2). The next goal would be to lift this statement to *all* CPT-algorithms, which would separate P from CPT. Here, we see three options how to proceed:

   (i) Try to prove that for every CPT-algorithm that decides the CFI-query on the base graphs $(G_n)_{n\in\mathbb{N}}$, there is an equivalent CFI-symmetric algorithm or one that activates only objects with symmetric bases.

   (ii) Try to prove that on this particular family of base graphs, every CPT-definable object admits symmetric bases.

   (iii) Try to find another proof to show that the circuit $\widehat{C}(\mu)$ constructed in Theorem 8.6.1 has polynomial size without using the symmetric basis property but perhaps instead using certain specific properties of the chosen base graphs $(G_n)_{n\in\mathbb{N}}$.

Another alternative could of course also be to try and lift potential future lower bound results for the relevant families of symmetric XOR-circuits directly to the h.f. objects. This way, the detour via circuits could potentially be avoided. This should then yield lower bounds against all possible choiceless algorithms for the CFI-query, irrespective of whether they are CFI-symmetric or not, because restrictions such as CFI-symmetry or the symmetric basis condition are only needed when the h.f. sets are translated into circuits.

9 Lower bounds for families of symmetric XOR-circuits over hypercubes

In the last chapter, we developed a circuit-approach towards CPT lower bounds, that works at least against the class of algorithms we called *CFI-symmetric*, with the potential for generalisation. As a next step, we put this approach to the test. We choose a family of CFI-graphs and try to prove that the corresponding families of XOR-circuits mentioned in Theorem 8.5.2 cannot exist. The results from Chapter 7 suggest that CFI-graphs over hypercubes may be suitably hard for CPT, due to their high symmetry. This is reflected in the symmetry group that will act on the input gates of the XOR-circuits. The "closer to the full symmetric group" it is, the less likely it is – intuitively – that the symmetric circuits with the required properties are sufficiently small. Therefore, we study in this chapter again the CFI-query over hypercubes.
If we could successfully show the non-existence of the circuits from Theorem 8.5.2, then this would imply that no CFI-symmetric choiceless algorithm can solve the hypercube CFI-problem. Unfortunately, we only manage this to a certain extent. We impose stronger constraints on the circuits than the ones mentioned in Theorem 8.5.2 and then show that such circuit families over hypercubes indeed cannot exist. Concretely, we strengthen the symmetry condition on the circuits and assume that they are stabilised by *all* automorphisms of the base graphs (i.e. n-dimensional hypercubes), so their orbit-size is one. Theorem 8.5.2 states only that the orbit-size of the circuits has to be polynomial. Moreover, we impose the condition that the (orbit-wise) number of children and parents of every gate has to be logarithmically bounded. This may be related to the logarithmic bound on the fan-in dimension that we get from Theorem 8.5.2, but it is probably a stronger restriction. For circuits with these properties over the n-dimensional hypercubes, we show that they are not sensitive to enough input gates and hence violate Property 3 from Theorem 8.5.2. The precise formulation of our negative result is Theorem 9.2.1. But before we come to that, we have to verify that hypercubes are indeed suitable base graphs and satisfy the prerequisites of Theorem 8.5.2.

9.1 Properties of hypercubes

Before we begin with our analysis of symmetric XOR-circuits over hypercubes, we would like to take a look at the relevant properties of hypercubes, that are mentioned in Theorem 8.5.2. For convenience, here is the theorem once more:

Theorem 8.5.2. *Let $(G_n = (V_n, E_n))_{n\in\mathbb{N}}$ be a sequence of base graphs. Let $\mathfrak{G}_n^S$ be a CFI-graph over G_n, let $\boldsymbol{tw}_n$ denote the treewidth of G_n. Let $f(n) \in \mathcal{O}(\boldsymbol{tw}_n)$ be a function* *such that every $\mathfrak{G}_n^S$ is $\mathcal{C}^{\mathbf{tw}_n}$-homogeneous, for all tuples of length $\leq 2f(n)$. Let $g(n)$ be a function such that the* CFI-support-gap *for every $\mu \in \mathrm{HF}(\widehat{E}_n)$ with minimum support $s(\mu) \in \Omega(f(n))$ is bounded by $g(n)$.*

If there exists a CFI-symmetric CPT*-program Π that decides the CFI-query on all $\mathfrak{G}_n^S$, then for every $G_n = (V_n, E_n)$, there exists an XOR-circuit C_n over G_n that satisfies the following "instantiated properties" from Theorem 8.4.1:*

1. *The number of gates in C_n is polynomial in $|\mathfrak{G}_n^S|$.*
2. *The orbit-size $|\mathbf{Orb}_{G_n}(C_n)|$ of the circuit is polynomial in $|\mathfrak{G}_n^S|$.*
3. *C_n is* sensitive *to $\Omega(f(n)/g(n))$ many edges in E_n.*
4. *The* fan-in dimension *of C_n,* restricted to the space $\mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{G}_n^S)$*, is $\mathcal{O}(\log|\mathfrak{G}_n^S|)$.*
5. *If the program Π is* super-symmetric *in addition to being CFI-symmetric, or if the base graph G_n decomposes into at most $\mathcal{O}(\sqrt{\log|\mathfrak{G}^S|})$ many components when any $f(n)/g(n)$ many edges are removed, then also the (unrestricted)* fan-in dimension *of C_n is $\mathcal{O}(\log|\mathfrak{G}_n^S|)$.*

The theorem mainly depends on three parameters of the base graphs: The *treewidth* of the graph, the *CFI-support gap* of the h.f. sets, and the fact that the CFI-graphs over the base graphs are $\mathcal{C}^{\mathbf{tw}_n}$-homogeneous.
Recall that the n-dimensional hypercube $\mathcal{H}_n$ is the undirected graph with universe $\{0,1\}^n$ in which there is an edge between any two words with Hamming distance exactly one. Its automorphism group is the semi-direct product $\mathbb{F}_2^n \rtimes \mathbf{Sym}_n$, where $\mathbf{Sym}_n$ acts on the positions of the binary words (see Section 5.3) but as in Chapter 7, we pretend it is just $\mathbf{Sym}_n$. This makes life easier and besides, if $\mathbf{Sym}_n$-symmetric XOR-circuits with the necessary properties do not exist, then this is "even more true" for the larger symmetry group $\mathbb{F}_2^n \rtimes \mathbf{Sym}_n$.
In the following, when we speak about CFI-structures over hypercubes, we do not distinguish between isomorphic ones, so we only consider *the* even and *the* odd CFI-structure over $\mathcal{H}_n$ and denote them $\mathfrak{H}_n^0$ and $\mathfrak{H}_n^1$, respectively. The size $|\mathfrak{H}_n^i|$ of these CFI-structures is polynomial in $2^n = |\mathcal{H}_n|$, because the CFI-construction increases the size of the graph exponentially in the maximum degree. This maximum degree in $\mathcal{H}_n$ is n, so the size increase by a factor of 2^n is still polynomial in $|\mathcal{H}_n|$. Now let us check the relevant properties of the hypercubes and their CFI-structures.

Treewidth of hypercubes As mentioned earlier in Lemma 5.3.1, the treewidth of n-dimensional hypercubes is a function in $\Theta(2^n/\sqrt{n})$, which can be looked up in [102]. This is close to being linear in $2^n = |\mathcal{H}_n|$, so it is sufficiently large to translate into a meaningful lower bound on the input sensitivity of the resulting circuits.

Homogeneity of hypercubes

As in [40], the $\mathcal{C}^k$*-type* of a tuple $\overline{a}$ in a given structure $\mathfrak{A}$ is the collection of all $\mathcal{C}^k$-formulas that are true in $(\mathfrak{A}, \overline{a})$. Recall that a structure $\mathfrak{A}$ is $\mathcal{C}^k$*-homogeneous* for tuples of length ℓ, for some $\ell \leq k$, if for any two tuples $\overline{\alpha}, \overline{\alpha}'$ in $\mathfrak{A}$ of length $\leq \ell$ that have the same $\mathcal{C}^k$-type, there is an automorphism of $\mathfrak{A}$ that moves $\overline{\alpha}$ to $\overline{\alpha}'$.

We say that a tuple $\overline{\alpha}$ in $V(\mathfrak{H}_n^i)$ *contains a star* if there is a centre $c \in V(\mathcal{H}_n)$ such that for each incident edge $e \in E_n(c)$, there is an entry of α in the edge gadget e^*.

The next lemma is a technical ingredient that we need for the homogeneity result for hypercube CFI-structures. It shows the homogeneity condition for specific tuples.

Lemma 9.1.1. *Let $\boldsymbol{tw}_n \in \Theta(2^n/\sqrt{n})$ denote the treewidth of $\mathcal{H}_n$. Let $\overline{\alpha}$ be a tuple in $V(\mathfrak{H}_n^i)$ that contains a star and has length at most $(\boldsymbol{tw}_n/n) - 2$. Let $\gamma, \gamma' \in V(\mathfrak{H}_n^i)$ and let tp$(\overline{\alpha}\gamma)$ denote the $\mathcal{C}^{\boldsymbol{tw}_n}$-type of this extended tuple. If tp$(\overline{\alpha}\gamma) =$ tp$(\overline{\alpha}\gamma')$, then there is an automorphism $\rho \in \mathbf{Aut}(\mathfrak{H}_n^i)$ such that $\rho(\gamma) = \gamma'$ and $\rho(\overline{\alpha}) = \overline{\alpha}$.*

Proof. We assume that $\text{tp}(\overline{\alpha}\gamma) = \text{tp}(\overline{\alpha}\gamma')$. The first half of the proof consists in establishing that then, γ, γ' are in the same edge or vertex gadget. Assume first that $\gamma, \gamma' \in \widehat{E}$. Now suppose for a contradiction that the edge gadget of γ and γ' is not the same. Let $\{u, v\} \in E_n$ be the edge in whose gadget γ is, and let $\{u', v'\} \in E_n$ be the corresponding edge for γ'. W.l.o.g. we may assume $u \neq u'$ and $u \neq v'$. Now let $s_1, ..., s_n \in E_n$ be the edges that form the star which is covered by $\overline{\alpha}$ according to the assumption. We may assume that the centre of the star is the string 0^n, because the automorphism group of the hypercube is transitive and so we can always move the centre of the star to 0^n. Let s_i denote the edge $\{0^n, 0^{i-1}10^{n-i}\}$, i.e. the edge along which the position i is flipped.
Let $U \subseteq [n]$ be the positions at which the string u is 1. We construct a $\mathcal{C}^5$-formula $\varphi_u(x)$ that defines the gadget u^* in $\mathfrak{H}_n^i$ using the star $s_1, ..., s_n$ as parameters. More precisely, let $s_1', ..., s_n'$ be the respective vertices in the edge gadgets that occur in $\overline{\alpha}$.
Our formula uses some auxiliary formulas: $\psi_{\text{dist}=\ell}(x, y) \in \mathcal{C}^3$ which asserts that there is a path of length ℓ from x to y. This can be expressed with only three variables by requantifying variables in an alternating way (see e.g. Proposition 3.2 in [76]). Also, we use a formula $\psi_{\approx}(x, y)$ which asserts that x and y are in the same vertex-gadget. This can be expressed by saying that both of them have exactly n neighbours, and: For every neighbour z of x, z is either also a neighbour of y or adjacent to a neighbour of y (in the same edge-gadget). The same must hold for every neighbour z of y. Expressing this requires not more than five variables in total. Now we define:

$$\varphi_u(x) := \bigwedge_{i \in U} \exists z (\psi_{\text{dist}=1+2(|U|-1)}(s_i', z) \wedge \psi_{\approx}(z, x)) \wedge$$
$$\bigwedge_{i \in [n] \setminus U} \neg \exists z (\psi_{\text{dist}=1+2(|U|-1)}(s_i', z) \wedge \psi_{\approx}(z, x)).$$

Claim: $\mathfrak{H}_n^i \models \varphi_u(a)$ iff $a \in u^*$.
Proof of claim: We are assuming that s_i is the edge between 0^n and the string with a 1

at position i. Now we show that $\mathfrak{H}_n^i \models \varphi_u(a)$ if $a \in u^*$: We have to check that for every $i \in [n]$, the respective conjunct of the formula is satisfied. If $i \in U$, then for any $s_i' \in s_i^*$, there exists $z \in u^*$ such that there is a path from s_i' to a of length exactly $1 + 2(|U| - 1)$ in $\mathfrak{H}_n^i$: The path goes one step from s_i' into the vertex gadget for $0^{i-1}10^{n-i}$, and from there, the path follows a shortest path of length $|U| - 1$ in $\mathcal{H}_n$ that goes from $0^{i-1}10^{n-i}$ to the vertex u and flips the remaining $|U| - 1$ zeros on the way. That path in the CFI-structure $\mathfrak{H}_n^i$ is twice as long because every edge is subdivided by a gadget. The path will end in some node in the vertex-gadget u^*. If $i \notin U$, then there is no path from s_i' of length $1 + 2(|U| - 1)$ that ends in a node in u^*: The shortest path from s_i' into the gadget u^* requires $1 + 2|U|$ steps. Hence, $\mathfrak{H}_n^i \models \varphi_u(a)$.
If $a \notin u^*$, then $\mathfrak{H}_n^i \not\models \varphi_u(a)$, because $\varphi_u(a)$ can only be satisfied if the required paths exist (do not exist, respectively) in $\mathcal{H}_n$, and the above arguments also show that these conditions are only satisfiable if a is in the gadget of u. This proves the claim.

Therefore, we have

$$\mathfrak{H}_n^i \models \exists x(Ex\gamma \wedge \varphi_u(x)),$$

but

$$\mathfrak{H}_n^i \not\models \exists x(Ex\gamma' \wedge \varphi_u(x)).$$

This is a contradiction to the assumption that $\overline{\alpha}\gamma$ and $\overline{\alpha}\gamma'$ have the same $\mathcal{C}^{\mathbf{tw}_n}$-type, because φ_u uses only $\mathcal{O}(n)$ many variables. Thus we have shown that there is one edge $g \in E_n$ such that $\gamma, \gamma' \in g^*$.
In case that γ and γ' are both in vertex gadgets, then the same argument shows that they must be in the same gadget u^* because we can define this gadget with the above formula. Similarly, we can argue if γ is in a vertex gadget and γ' is in an edge gadget: Then we define the vertex gadget of γ with the above formula, and γ' will not satisfy it. So the types of the tuples being equal entails that γ and γ' must be in the same gadget, be it of an edge or vertex.

In the second half of the proof we show that there is an automorphism $\rho \in \mathbf{Aut}(\mathfrak{H}_n^i)$ that maps $\overline{\alpha}\gamma$ to $\overline{\alpha}\gamma'$, again under the assumption that $\text{tp}(\overline{\alpha}\gamma) = \text{tp}(\overline{\alpha}\gamma')$. We first deal with the case that $\gamma, \gamma' \in g^*$ for some edge gadget g^*. It is not necessary to permute the hypercube, so it suffices to find an edge-flip automorphism, i.e. $\rho \in \mathbf{Aut}_{\text{CFI}}(\mathfrak{H}_n^i)$.
In case that $\gamma = \gamma'$, there is nothing to show. So let us assume that w.l.o.g. $\gamma = g_0$ and $\gamma' = g_1$. We need to find $\rho \in \mathbf{Aut}(\mathfrak{H}_n^i)$ such that $\rho(g_0) = g_1$, and such that ρ fixes $\overline{\alpha}$. Now we call an edge $e = \{u, v\} \in E_n$ *fixed* if e_0 or e_1 occurs in $\overline{\alpha}$. We call a vertex $v \in V_n$ *fixed* if some node inv^* occurs in $\overline{\alpha}$. We know that g is not fixed because if it is, then $\overline{\alpha}g_0$ and $\overline{\alpha}g_1$ have different types. A cycle in $\mathcal{H}_n$ is called *fixed* if at least one edge or one vertex on it is fixed. Else, the cycle is *free*. If there exists a free cycle in $\mathcal{H}_n$ on which g lies, then the desired automorphism is $\rho_F \in \mathbf{Aut}_{\text{CFI}}(\mathfrak{H}_n^i)$, where F is the edge-set of the free cycle.
Otherwise, every cycle on which g lies is fixed. We want to show that in this case, $\overline{\alpha}\gamma$ and $\overline{\alpha}\gamma'$ do not have the same $\mathcal{C}^{\mathbf{tw}_n}$-type and so, this situation cannot occur.

Let $u, v \in \{0,1\}^n$ be the endpoints of g. Let $X_u \subseteq \{0,1\}^n$ be the set of vertices in $\mathcal{H}_n$ that are reachable from u via paths using only free vertices and free edges in $E_n \setminus \{g\}$. The set X_u is meant to include the fixed vertices which are reachable in this way. Similarly, we define X_v as the set of reachable vertices from v via such free paths (also including fixed vertices). Because no free cycle exists, the sets X_u and X_v must be disjoint (except for potential shared fixed vertices). We restrict the graph to the smaller of these two sets, w.l.o.g. this is X_u. So let $X := X_u$. We now consider the graphs G and G', which are induced subgraphs of $\mathfrak{H}^i_n$ on the universe

$$\begin{aligned}\mathcal{X} := \{v^Y \mid v^Y \in v^*, v \in X\} \cup \{e_j \mid j \in \{0,1\}, e \in E(\mathcal{H}_n[X])\} \\ \cup \{a \in V(\mathfrak{H}^i_n) \mid a \text{ an entry in } \overline{\alpha}\gamma\} \cup \{\gamma'\}.\end{aligned}$$

Now $G := (\mathfrak{H}^i_n[\mathcal{X}], \overline{\alpha}\gamma)$ and $G' := (\mathfrak{H}^i_n[\mathcal{X}], \overline{\alpha}\gamma')$, that is, they are both the same induced subgraph of the CFI-graph, expanded with the respective tuples of constants (strictly speaking, the constant symbol for γ and γ' should be the same, but with the interpretation γ and γ', respectively).
As for the size of $X \subseteq V(\mathcal{H}_n)$, we have: $|X| < |\overline{\alpha}\gamma| \cdot n = \mathbf{tw}_n - n$. To see this, let $\delta X \subseteq E_n$ be the cut of X in $\mathcal{H}_n$, i.e. the set of edges between X and its complement. The Cheeger number of $\mathcal{H}_n$, which denotes the minimum of $\frac{|\delta A|}{|A|}$ over all $A \subseteq V(\mathcal{H}_n)$ with $|A| \leq |V(\mathcal{H}_n)|/2$, is between 1 and $2\sqrt{n}$. This can be seen from the Cheeger inequalities (see e.g. in [46]) together with the fact that the smallest non-zero Eigenvalue of the Laplacian of any Hypercube is 2 [50] (the fastest way to look this up is actually Wikipedia). This means that $|\delta X| \geq |X|$. From this it follows that $|\overline{\alpha}\gamma| \cdot n \geq |X|$ because the edges in the cut δX are exactly the fixed edges, and each entry of $\overline{\alpha}\gamma$ fixes at most n edges. Moreover, we can say that the inequality must actually be strict, so $|\overline{\alpha}\gamma| \cdot n > |X|$. This is because not all entries in $\overline{\alpha}$ are used to fix the edges in δX; some entries of $\overline{\alpha}$ must also be in $\delta X_v \setminus \delta X_u$. Hence, we have $|X| < |\overline{\alpha}\gamma| \cdot n = \mathbf{tw}_n - n$.

Claim: Spoiler wins the bijective $\mathbf{tw}_n$-pebble game on G and G'.
Proof of claim: Trivially, the treewidth of $\mathcal{H}_n[X]$ is strictly less than $\mathbf{tw}_n - n$ because $|X| < \mathbf{tw}_n - n$. Hence, the Cops win the Cops and Robber game on $\mathcal{H}_n[X]$ with $\mathbf{tw}_n - n$ many cops. Spoiler's goal is to pebble in both structures G and G' the vertex e_1 in every edge in $E_n(u) \setminus \{g\}$, i.e. every edge incident to u, except g. If he achieves that, then he wins in the next round: Assume w.l.o.g. that the gadget u^* is even. Then in G', an even number of neighbours of every vertex in u^* is pebbled or equal to the constant $\gamma' = g_1$. In G, this number is odd for every vertex in u^*, because $\gamma = g_0$. Therefore, Spoiler can then place an additional pebble on an arbitrary vertex in u^* and wins because Duplicator's bijection must map the gadget u^* in G to u^* in G'.
Now Spoiler can achieve this goal by a standard argument, as for example given in [10]: Initially, the "target vertex" for Spoiler is u. This means that he has to pebble the e_1-vertices in all its incident edges in both graphs. Duplicator's bijections can flip edges, which changes the set of target vertices for Spoiler. Suppose $F \subseteq E(\mathcal{H}_n[X])$ is the set of edges flipped by Duplicator in a given round. This has the effect that every vertex in $\mathcal{H}_n[X]$ whose F-degree is odd changes its role from target- to non-target vertex and vice

versa. However, Duplicator cannot flip edges which are pebbled by Spoiler. If Spoiler places his pebbles on edges (or their endpoints) according to the Cops' winning strategy (while Duplicator "moves the robber" by flipping paths), then he can eventually pin down a target vertex that Duplicator cannot move anywhere else. From such a position, he can enforce a situation as described above and wins the game. This argument is well-known; the only additional difficulty in our setting is that the base graph is not ordered, so we have to argue that Duplicator cannot win by playing bijections other than edge-flips. This can be enforced by Spoiler, using at most n extra pebbles: The important observation is that Duplicator's bijection must respect distances to all pebbles on the board and to the parameters $\overline{\alpha}\gamma$. So Duplicator can only map a gadget w^* to some other gadget $\pi(w^*)$ if the vertices w and $\pi(w)$ have the same distance in $\mathcal{H}_n[X]$ to every pebbled vertex and parameter. This holds because if Duplicator disrespects distances between such marked elements, then Spoiler can easily win using three pebbles, that he moves along the shortest paths.
Now Spoiler can simply start by pebbling some star in $\mathcal{H}_n[X]$ with n pebbles (this is possible because we are playing with $\mathbf{tw}_n$ pebbles, but only $\mathbf{tw}_n - n$ many are needed to simulate the Cops' winning strategy). Once a star is pebbled in G and G', it follows with the argument used earlier in the proof of this lemma that any $w \in V(\mathcal{H}_n[X])$ has a unique set of distances to the edges of the star, and therefore, Duplicator is then forced to map every w^* to a unique vertex gadget $\pi(w^*)$. This entails that there is also a unique edge gadget $\pi(e^*)$ that she has to map each e^* to. So, from that moment on, Duplicator is indeed limited to playing only edge-flips, and then, Spoiler wins in the aforementioned way using the Cops' strategy. This proves the claim.

The claim directly entails that $\overline{\alpha}\gamma$ and $\overline{\alpha}\gamma'$ do not have the same $\mathcal{C}^{\mathbf{tw}_n}$-type if g does not lie on a free cycle. This finishes the case where γ and γ' are in an edge gadget. The other case is that $\gamma = u^*_Y$ and $\gamma' = u^*_{Y'}$ are both in some vertex gadget u^*. Then there is an even-sized set of edges $F = Y \triangle Y'$ incident with u in $\mathcal{H}_n$ such that we have to flip the edges in F (and no other edges in $E(u)$) in order to map γ to γ'. This is possible if we can pair up the edges in F in such a way that each pair $\{f_1, f_2\} \subseteq F$ lies on a free cycle (that avoids all other edges in $E(u)$). Suppose for a contradiction that there is some pair $\{f_1, f_2\} \subseteq F$ which is not on a free cycle. Then let $g := f_1$, and make the same argument as above in the case where we wanted to flip the edge g (with the difference that we now remove both f_1 and f_2 in order to get the two sets X_u and X_v that are not connected by any free path). Then the above proof shows that the two nodes in the gadget f_1^* are distinguishable in $\mathcal{C}^{\mathbf{tw}_n}$ using the parameters $\overline{\alpha}$. But then also γ and γ' in u^* are distinguishable because one of them is adjacent to the 0-node in f_1^*, and the other is adjacent to the 1-node in f_1^*. So again, the required free cycles must exist because otherwise, $\overline{\alpha}\gamma$ and $\overline{\alpha}\gamma'$ have distinct $\mathcal{C}^{\mathbf{tw}_n}$-types. □

Lemma 9.1.2. *The structures $\mathfrak{H}_n^0$ and $\mathfrak{H}_n^1$ are homogeneous in the following sense: Let $\boldsymbol{tw}_n$ denote the treewidth of $\mathcal{H}_n$. For any tuples $\overline{\alpha}, \overline{\alpha}'$ in $V(\mathfrak{H}_n^i)$ of length $|\overline{\alpha}| = |\overline{\alpha}'| \leq \boldsymbol{tw}_n/n - n - 1$ it holds: If $\overline{\alpha}$ and $\overline{\alpha}'$ have the same $\mathcal{C}^{\boldsymbol{tw}_n}$-type in $\mathfrak{H}_n^i$, then there is an automorphism of $\mathfrak{H}_n^i$ that maps $\overline{\alpha}$ to $\overline{\alpha}'$.*

Proof. First of all, we show the following statement via induction on $|\overline{\alpha}|$.
Claim 1: Let $\overline{s}$ be a tuple of length n that contains a star. If $\mathrm{tp}(\overline{s\alpha}) = \mathrm{tp}(\overline{s\alpha'})$, then there is an automorphism of $\mathfrak{H}_n^i$ that maps the tuple $\overline{s\alpha}$ to $\overline{s\alpha'}$.
Proof of claim: In the base case, $|\overline{\alpha}| = 0$, there is nothing to show because the identity permutation is the desired automorphism then.
For the inductive step, let $\overline{\alpha} = \overline{\beta}\gamma$ where $|\overline{\beta}| = |\overline{\alpha}| - 1$, and similarly, write $\overline{\alpha}' = \overline{\beta}'\gamma'$. Since $\mathrm{tp}(\overline{s\alpha}) = \mathrm{tp}(\overline{s\alpha'})$, we also have $\mathrm{tp}(\overline{s\beta}) = \mathrm{tp}(\overline{s\beta}')$. Therefore, the induction hypothesis gives us an automorphism $\pi \in \mathbf{Aut}(\mathfrak{H}_n^i)$ such that $\pi(\overline{s\beta}) = \overline{s\beta}'$. Since automorphisms preserve types, we have $\mathrm{tp}(\overline{s\beta}\gamma) = \mathrm{tp}(\pi(\overline{s\beta}\gamma)) = \mathrm{tp}(\overline{s\beta}'\pi(\gamma)) = \mathrm{tp}(\overline{s\beta}'\gamma')$. The length of the tuples $\overline{s\beta}'\pi(\gamma)$ and $\overline{s\beta}'\gamma'$ is at most $\mathbf{tw}_n/n - 1$, so we can apply Lemma 9.1.1 to them. This gives us another automorphism σ such that $\sigma(\overline{s\beta}'\pi(\gamma)) = \overline{s\beta}'\gamma'$. In total, $\sigma \circ \pi \in \mathbf{Aut}(\mathfrak{H}_n^i)$ is the desired automorphism that maps $\overline{s\alpha}$ to $\overline{s\alpha'}$.

Now we will use Claim 1 to prove the lemma. Let $\overline{\alpha}$ and $\overline{\alpha}'$ be as in the lemma. Let $\overline{s}$ be a tuple of length n that contains a star (and only a star). By Lemma 34 in [40], there exists a tuple $\overline{s}'$ such that $\mathrm{tp}(\overline{s\alpha}) = \mathrm{tp}(\overline{s}'\overline{\alpha}')$. This holds because $(\mathfrak{H}_n^i, \overline{\alpha}) \equiv_{\mathcal{C}^{\mathbf{tw}_n}} (\mathfrak{H}_n^i, \overline{\alpha}')$ (Theorem 5.1.3), and so for any extension of the tuple $\overline{\alpha}$ (which has length at most $\mathbf{tw}_n - n$) by only n elements, there is an extension of $\overline{\alpha}'$ that preserves the type.

Claim 2: There exists an automorphism $\sigma \in \mathbf{Aut}(\mathfrak{H}_n^i)$ such that $\sigma(\overline{s}) = \overline{s}'$.
Proof of claim: Let $s_1, ..., s_n \in E_n$ be the edges of the star that is covered by $\overline{s}$ and $s_1', ..., s_n' \in E_n$ be the edges of the star of $\overline{s}'$. There is an automorphism $\sigma' \in \mathbf{Aut}(\mathcal{H}_n)$ such that $s_i' = \sigma'(s_i)$, for every i. This is easy to see because we can map the centre c of one star to the centre c' of the other, and apply the right permutation to its incident edges. In $\mathfrak{H}_n^i$, the gadgets c^* and c'^* are either both even or both odd, in relation to the tuple $\overline{s}$. That is, if we pretend that the vertices $\overline{s}_i$ are the 1-vertices in their respective edge-gadgets, then the vertex-gadgets c^* and c'^* have the same parity: This is because we can express in, say, $\mathcal{C}^5$ that every vertex in c^* is connected with an even number of vertices in $\overline{s}$ (and $\overline{s}'$, respectively).
We say that there is a mismatch between $\sigma'(\overline{s})$ and $\overline{s}'$ at position i if $\sigma'(\overline{s})_i$ is the 0-vertex in its edge gadget, and $\overline{s}_i'$ the 1-vertex, or vice versa. By what we just argued, the number of mismatches between $\sigma'(\overline{s})$ and $\overline{s}'$ is even. This can be corrected with an automorphism $\rho_F \in \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{H}_n^i)$ that flips edges along ℓ disjoint cycles originating in c', where ℓ is half the number of mismatches. Then $\sigma = \rho_F \circ \sigma'$ is an automorphism that takes $\overline{s}$ to $\overline{s}'$. This proves the claim.

With Claim 2, we get that $\mathrm{tp}(\overline{s}'\sigma(\overline{\alpha})) = \mathrm{tp}(\overline{s}'\overline{\alpha}')$ because automorphisms preserve types. The fact that $\overline{s}$ contains a star is easily definable in counting logic, so $\overline{s}'$ also contains a star. Therefore, the lemma now follows from Claim 1, which gives us an automorphism π that maps $\sigma(\overline{\alpha})$ to $\overline{\alpha}'$. Then $\pi \circ \sigma$ is the automorphism whose existence is claimed in the lemma. □

In total, the hypercube CFI-structures satisfy the homogeneity condition required by Theorem 8.5.2 if we take $f(n)$ (the tuple-length) to be a function in $\Theta(\mathbf{tw}_n/n) = \Theta(2^n/n^{1.5})$. This will be the lower bound for the sensitivity of the circuits.

CFI-support gap of hypercube CFI-structures

Recall from Definition 8.5.1 that the CFI-support gap of a h.f. set μ is $\alpha(\mu) = \frac{s(\mu)}{|\sup_{\mathrm{CFI}}(\mu)|}$ where $s(\mu)$ denotes the size of the smallest support; this depends on the structure. Let again $\mathfrak{H}_n^i$ denote the odd/even CFI-structure over the n-dimensional hypercube, and let E_n be the edge set of that hypercube. Let $\mathfrak{H}_n$ denote the full CFI-graph over $\mathcal{H}_n$ (see Chapter 5). We would like to prove an upper bound on the ratio $\alpha(\mu)$ over all h.f. sets $\mu \in \mathrm{HF}(\widehat{E}_n)$. Here, $s(\mu)$ denotes the size of the smallest $\mathbf{Aut}(\mathfrak{H}_n^i)$-support of μ, and $\sup_{\mathrm{CFI}}(\mu)$ the size of the smallest CFI-support of μ. Recall from Definition 8.1.5 that a CFI-support of μ is a set of edges $S \subseteq E$ such that any edge-flip $\rho_F \in \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{H}_n)$ that fixes all edges in S also fixes μ. The edge-flips considered here are not necessarily automorphisms of $\mathfrak{H}_n^i$, so they include all combinations of flipped edges and not only cycles.

Lemma 9.1.3. *Let $n \in \mathbb{N}$ and $\mu \in \mathrm{HF}(\widehat{E}_n)$. Let $S \subseteq E$ be a smallest CFI-support of μ. Then there exists an $\mathbf{Aut}(\mathfrak{H}_n^i)$-support of μ of size at most $|S| + n$.*

Proof. Every $\rho_F \in \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{H}_n)$ such that $F \cap S = \emptyset$ fixes μ. This holds in particular for every such $\rho_F \in \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{H}_n^i) \leq \mathbf{Aut}_{\mathrm{CFI}}(\mathfrak{H}_n)$. Thus, let $\overline{\alpha}$ be an arbitrary tuple in $\widehat{E}_n$ that contains exactly one vertex e_i from every edge $e \in S$. So $|\overline{\alpha}| = |S|$. Then any automorphism of the form $(\rho_F, \mathrm{id}) \in \mathbf{Aut}(\mathfrak{H}_n^i)$ that fixes $\overline{\alpha}$ also fixes μ. Now extend $\overline{\alpha}$ to a tuple $\overline{\beta}$ that contains a star. This is always possible such that $|\overline{\beta}| \leq |\overline{\alpha}| + n$. Now any automorphism in $\mathbf{Aut}(\mathfrak{H}_n^i)$ that fixes $\overline{\beta}$ must have id as its second component, because any $(\rho_F, \pi) \in \mathbf{Aut}(\mathfrak{H}_n^i)$ with $\pi \neq \mathrm{id}$ moves every star in the hypercube. So in total, every automorphism that fixes $\overline{\beta}$ fixes μ, and the length of $\overline{\beta}$ is at most $|S| + n$. □

Let $\mathbf{tw}_n \in \Theta(2^n/\sqrt{n})$ denote the treewidth of the n-dimensional hypercube. Let $\alpha(n) := \max_{\substack{\mu \in \mathrm{HF}(\widehat{E}_n),\\ s(\mu) \in \Omega(\mathbf{tw}_n/n)}} \alpha(\mu)$ be the maximum CFI-support gap that can occur for any object in $\mathrm{HF}(\widehat{E}_n)$ with minimum support size at least $\Omega(\mathbf{tw}_n/n)$.

Corollary 9.1.4. *There is a function $g(n) \in \mathcal{O}(1)$ which is an upper bound for $\alpha(n)$.*

Proof. Let $\mu \in \mathrm{HF}(\widehat{E}_n)$ be an object whose minimum $\mathbf{Aut}(\mathfrak{H}_n^i)$-support size $s(\mu)$ is at least $\Omega(\mathbf{tw}_n/n)$. By Lemma 9.1.3, its smallest CFI-support $\sup_{\mathrm{CFI}}(\mu)$ must have size at least $s(\mu) - n$. Thus, the CFI-support gap $\frac{s(\mu)}{|\sup_{\mathrm{CFI}}(\mu)|}$ is at most $\frac{s(\mu)}{s(\mu)-n} = 1 + \frac{n}{s(\mu)-n}$. Since $s(\mu) \in \Theta(2^n/n^{1.5})$, this expression is asymptotically equal to 1. □

Consequently, in the setting of Theorem 8.5.2, we can take a constant function $g(n)$ for the upper bound of the CFI support gap. This is convenient because it means that Theorem 8.5.2 yields XOR-circuits that are sensitive to as many edges as possible, namely $\Omega(\mathbf{tw}_n/n)$ many. We are aiming to prove that the symmetric circuits given by the theorem cannot exist, so it is good that the support gap does not loosen the constraints on the circuits here.

Connectivity of hypercubes
Property 5 from Theorem 8.5.2 requires either a restriction to super-symmetric objects or that the hypercube splits into at most $\mathcal{O}(\sqrt{\log |\mathfrak{H}_n^i|}) = \mathcal{O}(\sqrt{n})$ many components when $\Theta(2^n/n^{1.5})$ many edges are removed. This connectivity condition is unfortunately not satisfied by hypercubes: Since every vertex in $\mathcal{H}_n$ has degree n, it is easily possible to isolate $\Theta(2^n/n^{2.5})$ many vertices by removing $\Theta(2^n/n^{1.5})$ many edges. This yields a much higher number of connected components than $\mathcal{O}(\sqrt{n})$. Therefore, if we want to use the fan-in dimension bound from Property 5 in the context of hypercube CFI-structures, we have to restrict ourselves to *super-symmetric* algorithms.

In total, we can summarise the instantiated version of Theorem 8.5.2 for hypercube CFI structures as follows:

Theorem 9.1.5. *If there exists a* super-symmetric *and* CFI-symmetric CPT-*program* Π *that decides the CFI-query on the family of all hypercube CFI-structures* $(\mathfrak{H}_n^i)_{n\in\mathbb{N}}$*, then for every* $\mathcal{H}_n = (V_n, E_n)$*, there exists an XOR-circuit* C_n *over* $\mathcal{H}_n$ *that satisfies:*

1. *The number of gates in* C_n *is polynomial in* 2^n*.*
2. *The orbit-size* $|\mathbf{Orb}_{\mathcal{H}_n}(C_n)|$ *of the circuit is polynomial in* 2^n*.*
3. C_n *is* sensitive *to* $\Omega(2^n/n^{1.5})$ *many edges in* E_n*.*
4. *The (unrestricted)* fan-in dimension *of* C_n *is* $\mathcal{O}(n)$*.*

9.2 Symmetric XOR-circuits over hypercubes

Now we come to the non-existence result for certain symmetric circuit families over hypercubes. The circuits that we show to be non-existent have slightly stricter constraints than the ones in the theorem above, namely with respect to Property 2 and 4. Our investigation is inspired by an "almost right" construction of circuits satisfying the properties from Theorem 9.1.5. The most difficult part about constructing such circuits seems to be the condition that they should have polynomial orbit size with respect to the action of $\mathbf{Sym}_n$ on $\{0,1\}^n$. A first idea would be to use some tree with logarithmic degree whose leafs are labelled with the elements of $\{0,1\}^n$. This would satisfy all properties except (maybe) the orbit size. Actually, we do not have a proof that tree-like circuits with the required orbit size do not exist, but we suspect that trees are not symmetric enough: Namely, with tree-like circuits, it would suffice to show that they must contain one path of super-polynomial orbit size. Then already the orbit of the whole tree would be super-polynomial because a polynomial-size tree can only contain polynomially many paths. Intuitively, paths in trees seem to be structurally not so much different from preorders. Hence, it may well be that the results from Chapter 7 can be generalised to such paths, which would rule out tree-like circuits.

However, we do not have any formal results in that direction and therefore leave the tree-like circuits aside for now. Instead we try a different construction idea, that could be considered the opposite of trees: In order to build a circuit that is guaranteed to be symmetric under the hypercube automorphisms, we can simply use the hypercube itself: Cut the hypercube in the middle, and use one half of the hypercube as the circuit. The output gate will then be, for example, the string 0^n, and the input gates are labelled with the strings of Hamming weight $(n/2)$, which are located in the "middle slice" of the hypercube. This construction is visualised below.

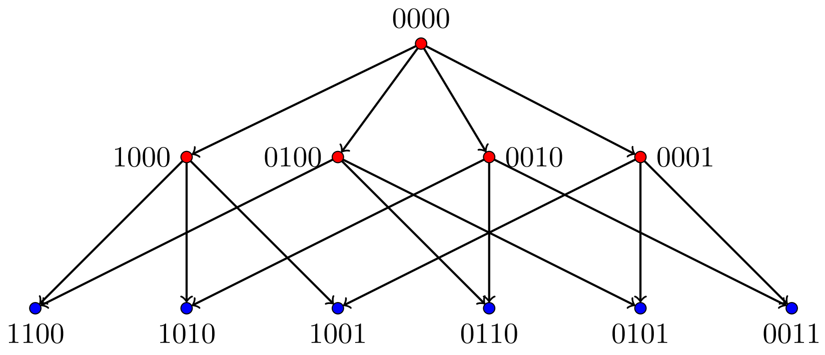


Figure 9.1: The 4-dimensional hypercube cut in half. Red nodes are XOR-gates, blue nodes input gates.

The circuit satisfies the correct size and orbit size bounds (namely, the orbit size of the circuit is one by construction), and also, it has degree n, which is logarithmic in $|\mathfrak{H}_n^i|$. Since the degree is an upper-bound for the fan-in dimension, the logarithmic fan-in bound is satisfied as well. However, we can observe that the circuit is actually not sensitive to *any* input bit at all. Already in the second layer, counted from the input layer, all inputs cancel out: Each input bit arrives in the root an even number of times. So we learn from this example that it is possible to build highly symmetric circuits by making them very DAG-like, but the high number of distinct paths that any given input bit can take through the circuit can easily lead to the input bit cancelling itself at some point in the circuit. This happens whenever this number of paths is even. So roughly speaking, tree-like circuits and these "halved-hypercube circuits" are two extremes: With trees, the sensitivity condition (Property 3 in Theorem 9.1.5) is clearly satisfied, but the symmetry condition is not obvious. Conversely, in the halved hypercubes, the symmetry is satisfied by construction but the circuits are not sensitive to any input bit. The question is: What is in between these two extremes? We will now show that the input-cancellation effect from the halved-hypercube circuits actually occurs in a larger class of symmetric circuits. In some sense, this rules out all circuits that are too similar to the halved hypercube. Roughly speaking, these include all circuits over the n-dimensional hypercube that are stabilised by all permutations in $\mathbf{Sym}_n$ and have "orbit-wise" logarithmic in- and out-degree.

Fix a family $(C_n)_{n\in\mathbb{N}}$ of XOR-circuits such that the input gates of C_n are labelled with edges of the n-dimensional hypercube $\mathcal{H}_n$. As in the previous chapter, the circuits are connected DAGs with a designated unique output gate, the root. We define

$$\begin{aligned}\mathbf{Aut}(C_n) :=\{\sigma \in \mathbf{Sym}(V_{C_n}) \mid \sigma &\text{ is an automorphism of the rooted DAG } (V_C, E_C, r)\\ &\text{and there exists a } \pi \in \mathbf{Sym}_n \text{ s.t. } \ell(\sigma(g)) = \pi(\ell(g)) \text{ for every input gate } g\}.\end{aligned}$$

Note that in particular, every automorphism must fix the root of the circuit and must permute the leafs in a way that complies with a permutation in $\mathbf{Sym}_n$ acting on the labels of leafs. We say that a permutation $\pi \in \mathbf{Sym}_n$ *extends* to an automorphism $\sigma \in \mathbf{Aut}(C_n)$ if σ maps the input gates g such that $\ell(\sigma(g)) = \pi(\ell(g))$ is satisfied. It may be that the identity permutation in $\mathbf{Sym}_n$ extends to non-trivial circuit automorphisms in C_n. In this case, the circuit is not rigid and every permutation in $\mathbf{Sym}_n$ has multiple circuit automorphisms that it extends to.

For a gate g in C_n and a parent h of g, we let

$$\mathbf{Orbit}_{(g)}(h) := \{\sigma(h) \mid \sigma \in \mathbf{Aut}(C_n), \sigma(g) = g\}.$$

Similarly,

$$\mathbf{Orbit}_{(h)}(g) := \{\sigma(g) \mid \sigma \in \mathbf{Aut}(C_n), \sigma(h) = h\}.$$

So these are the orbits of g, h, respectively, with respect to those circuit automorphisms that fix the child g, or the parent h, respectively. Note that $\mathbf{Orbit}_{(g)}(h) \subseteq E_C g$, and $\mathbf{Orbit}_{(h)}(g) \subseteq hE_C$ because circuit automorphisms preserve the wires and if one endpoint of a wire is fixed, then the image of the other endpoint must still be connected with the fixed gate. In the rest of this chapter, we prove:

Theorem 9.2.1. *Let $(C_n)_{n\in\mathbb{N}}$ be a family of XOR-circuits over the n-dimensional hypercubes such that for all $n \in \mathbb{N}$ it holds:*

1. *The size $|V_{C_n}|$ is polynomial in 2^n (and thus polynomial in $|\mathfrak{H}_n^i|$).*
2. Every *permutation $\pi \in \mathbf{Sym}_n$ acting on $\{0,1\}^n$ extends to a circuit-automorphism of C_n. Thus, the $\mathbf{Sym}_n$-orbit of C_n has size one.*
3. *There exists a function $f(n) \in \mathcal{O}(n)$ such that for all large enough n, for every gate g and every parent h of g in C_n, both $|\mathbf{Orbit}_{(g)}(h)|$ and $|\mathbf{Orbit}_{(h)}(g)|$ are at most $f(n)$.*

Then for any constant $\varepsilon > 0$, it holds for all large enough n: The circuit C_n can only be sensitive to an input gate g if $\ell(g) = \{u, v\}$ is an edge such that the zero-one-split in the binary strings $u, v \in \{0,1\}^n$ is more imbalanced than εn vs. $(1-\varepsilon)n$. In other words, the number of 1s or the number of 0s in u and v must be $< \varepsilon n$, and otherwise, this input gate does not contribute to the output of C_n.

In particular, these circuits do not satisfy Property 3 from Theorem 9.1.5 and are therefore ruled out:

Corollary 9.2.2. *Let $(C_n)_{n \in \mathbb{N}}$ be a circuit family as in Theorem 9.2.1. Then for all large enough n, the circuit C_n is sensitive to strictly less than $o(2^n/n^{1.5})$ many inputs.*

Proof. According to Exercise 9.42 in [57], it holds for any $\alpha \leq \frac{1}{2}$:

$$\sum_{k \leq \alpha n} \binom{n}{k} = 2^{nH(\alpha) - \frac{1}{2}\log n + \mathcal{O}(1)},$$

where $H(\alpha) = \alpha \log(\frac{1}{\alpha}) + (1-\alpha)\log(\frac{1}{1-\alpha})$. According to Theorem 9.2.1, the only edges of $\mathcal{H}_n$ that C_n can be sensitive to are between binary strings with less than εn many one- or zero-entries, for any $\varepsilon > 0$. The number of potential endpoints of such edges is twice the above sum, for $\alpha = \varepsilon$. The degree of $\mathcal{H}_n$ is n, so in total, C_n is sensitive to at most $2n \cdot \sum_{k \leq \varepsilon n} \binom{n}{k}$ many edges of $\mathcal{H}_n$. We can calculate that this is in $o(2^n/n^{1.5})$, for any $0 < \varepsilon < \frac{1}{2}$:

$$\lim_{n\to\infty} \frac{2^n}{n^{1.5} \cdot 2n \cdot 2^{nH(\varepsilon) - \frac{1}{2}\log n + \mathcal{O}(1)}} =$$
$$\lim_{n\to\infty} \frac{2^{(1-H(\varepsilon))n - \frac{1}{2}\log n + \mathcal{O}(1)}}{2n \cdot n^{1.5}} = \infty$$

In the last step, we used that $H(\varepsilon) < 1$, which holds as long as ε is chosen to be strictly less than $\frac{1}{2}$. □

This result does not yet completely rule out the existence of a symmetric circuit family as required by Theorem 9.1.5: Firstly, we assume the circuits here to be *fully symmetric*, i.e. they are stabilised by *every* permutation in $\mathbf{Sym}_n$; in Theorem 9.1.5, the circuits need only have a polynomial orbit with respect to the automorphisms of the base graph, so they are stabilised by many, but not necessarily by *all* these automorphisms. Secondly, in Theorem 9.1.5, we only have a logarithmic bound on the fan-in dimension, but it is not clear that this also entails a bound on the orbit-wise number of children and parents of each gate as in Property 4 above. Nevertheless, we hope that this negative result for the existence of fully symmetric bounded-degree XOR-circuits is a useful starting point to rule out further circuit classes over hypercubes, and eventually defeat all circuits from Theorem 9.1.5. This would then show that the CFI-query over hypercubes is not definable by any super- and CFI-symmetric algorithm.

For the proof of the theorem, we again make use of supporting partitions, that already served us well in Chapter 7. However, this time, we employ *alternating supporting partitions*. Recall Definition 4.2.7 from Section 4: An alternating supporting partition of a group $G \leq \mathbf{Sym}_n$ is essentially a partition of $[n]$ such that the alternating group within each part is a subgroup of G. Every group G has a unique coarsest alternating supporting partition (Lemma 4.2.8), denoted $\mathbf{SP}_A(G)$. For a gate $g \in V_{C_n}$, we denote by $\mathbf{SP}_A(g)$ the coarsest alternating supporting partition of the group

$$\mathbf{Stab}_n(g) := \{\pi \in \mathbf{Sym}_n \mid \pi \text{ extends to an automorphism of } C_n \text{ that fixes } g\}.$$

Here, we mean that at least one of the automorphisms that π extends to fixes g. It can be seen that $\mathbf{Stab}_n(g)$ is indeed a subgroup of $\mathbf{Sym}_n$ because it contains the identity permutation, and: If $\pi, \pi' \in \mathbf{Stab}_n(g)$, then there exist circuit automorphisms σ, σ' that π, π' extend to such that $\sigma(g) = \sigma'(g) = g$. Thus, $\sigma \circ \sigma'$ fixes g and is a circuit automorphism that $\pi \circ \pi'$ extends to. Thus, $\mathbf{Stab}_n(g) \leq \mathbf{Sym}_n$.

Importantly, $\mathbf{Stab}_n(g) \leq \mathbf{Stab}(\mathbf{SP}_A(g))$ (Lemma 4.2.10), so every permutation in $\mathbf{Stab}_n(g)$ acts as a permutation on the parts of $\mathbf{SP}_A(g)$. The supporting partition of an automorphic image of a gate can be obtained by applying a corresponding permutation in $\mathbf{Sym}_n$ to the supporting partition:

Lemma 9.2.3. *Let $g, \sigma g$ be two gates in C_n, for a $\sigma \in \mathbf{Aut}(C_n)$. Let $\pi \in \mathbf{Sym}_n$ be a permutation that extends to the circuit automorphism σ. Then*

$$\mathbf{SP}_A(\sigma g) = \pi(\mathbf{SP}_A(g)).$$

Proof. By Lemma 4.2.9, the partition $\pi\mathbf{SP}_A(g)$ is the coarsest alternating supporting partition of the group $\pi\mathbf{Stab}_n(g)\pi^{-1}$. It holds $\pi\mathbf{Stab}_n(g)\pi^{-1} = \mathbf{Stab}_n(\sigma g)$. □

Moreover, our group-theoretic Theorem 4.3.1 tells us that the supporting partitions $\mathbf{SP}_A(g_n)$, for every gate g_n in C_n, have at most $o(n)$ many singleton parts. This will become important later, when we analyse the supporting partitions of the gates. First of all, we would like to formalise what it means that a given input gate g cancels itself out in the circuit:

Lemma 9.2.4. *Let g be an input gate of an XOR-circuit C. The circuit C is sensitive to the input gate g if and only if the number of distinct paths from the root to g is odd.*

Proof. Via induction on the number of gates in C. In a circuit where g is the root, there is only one path and the input determines the output. The smallest possible case where the number of paths from the root to g is even is if C consists of a root with two children h_1, h_2, and one input gate g that is the child of both h_1 and h_2. Clearly, the input bit is canceled in the root.
For the inductive step, let $h_1, ..., h_m$ be the children of the root r. Let p_i be the number of distinct paths from h_i to the input g. By the induction hypothesis, the output of h_i depends on g iff p_i is odd (use the statement for the smaller subcircuit rooted at h_i). The number of distinct paths from r to g is $\sum p_i$. That number is even iff an even number of the p_i is odd. Then the input g has no influence on the value computed at r because only the h_i with p_i odd are sensitive to g, and these effects cancel at r because it is an even number. If an odd number of the p_i is odd, then r is sensitive to g. □

Thus, our goal is to prove that the number of paths between the root and each input gate labelled with a "too balanced edge" of $\mathcal{H}_n$ is even. Now the technical theorem that we want to prove in the next step reads as follows. From it, Theorem 9.2.1 follows with Lemma 9.2.4.

Theorem 9.2.5. *Let $(C_n)_{n\in\mathbb{N}}$ be a family of XOR-circuits with the properties mentioned in Theorem 9.2.1. Let a gate g_n in every C_n and a constant $\varepsilon > 0$ be fixed such that $\mathbf{SP}_A(g_n)$ contains at least two parts of size $\geq \varepsilon \cdot n$.*
Then for all large enough n, the number of distinct paths from the root of C_n to g_n is even.

The proof idea is vaguely similar to a technique known as "bottleneck counting", that has been used in proof complexity to establish lower bounds for resolution. Roughly speaking, we associate with every gate in a circuit a certain quantity of which we know that it must be high in the root and much lower in the input gates. Furthermore, we will prove that this quantity can only change by a small amount as we move from a gate to its parents. In other words: The quantity cannot "jump" from the low value at the leafs to the high value at the root, but it has to pass through many intermediate values in the middle of the circuit. We will then show that certain intermediate values, which must necessarily occur at some gates, entail that the number of paths from the root to the gate is even.

First, here is an observation about supporting partitions (or partitions in general). In most cases, their orbit has at least quadratic size, unless the partition has a very particular shape.

Lemma 9.2.6. *Let $\mathcal{P}$ be some partition of $[n]$, for some $n \geq 4$, and let $\mathbf{Stab}_n(\mathcal{P}) \leq \mathbf{Alt}_n$ denote the setwise stabiliser of the partition in the alternating group. Then the orbit size of $\mathcal{P}$, that is, $(1/2)n!/|\mathbf{Stab}_n(\mathcal{P})|$, is at least $\Omega(n^2)$ unless $\mathcal{P}$ has one of the following forms:*

- $\mathcal{P} = \{[n]\}$.
- $\mathcal{P} = \{\{s\}, [n] \setminus \{s\}\}$, *for some* $s \in [n]$.
- $\mathcal{P} = \{\{s\} \mid s \in [n]\}$.

Proof. It is easy to see that in each of the three cases above, the $\mathbf{Alt}_n$-orbit of $\mathcal{P}$ has size one or n (since we are assuming n to be large enough such that $\mathbf{Alt}_n$ acts transitively on $[n]$). It remains to show that the orbit size is at least quadratic if $\mathcal{P}$ has any other form. In that case, $\mathcal{P}$ must contain some part P of size $|P| \geq 2$, whose complement in $[n]$ is also of size ≥ 2. Since $\mathbf{Alt}_n$ is transitive on the subsets of $[n]$ (for each fixed subset-size), the part P has $\binom{n}{|P|}$ many $\mathbf{Alt}_n$-images, which is in $\Omega(n^2)$. If $|P| > n/2$, then we are done because any permutation that does not map P to itself is not in $\mathbf{Stab}_n(\mathcal{P})$ then, and so the $\mathbf{Alt}_n$-orbit of $\mathcal{P}_n$ is as large as claimed.
Otherwise, if $|P| \leq n/2$, we estimate $|\mathbf{Stab}_n(\mathcal{P})|$ as follows: Let k be the number of parts in $\mathcal{P}$ of size $|P|$. Then

$$|\mathbf{Stab}_n(\mathcal{P})| \leq (1/2) \cdot k! \cdot (|P|!)^k \cdot (n - k|P|)!$$

This is maximised for $k = n/|P|$ or $k = 1$ because if $k < \frac{n}{|P|}$, it can be calculated that decreasing k by one makes the expression larger, so if k is not largest possible, then the

bound for $\mathbf{Stab}_n(\mathcal{P})$ is maximised by making k as small as possible.
We have already dealt with the case that $k = 1$ above (where we assumed $|P|$ to be so large that no other part of this size can exist). Therefore, assume now that $k = n/|P|$. Then we get for the orbit size of $\mathcal{P}$:

$$\frac{n!}{(n/|P|)! \cdot (|P|!)^{n/|P|}} \approx \frac{(n/e)^n \cdot \sqrt{2\pi n}}{(n/|P|)! \cdot (|P|/e)^n \cdot \sqrt{2\pi|P|}^{(n/|P|)}}$$

$$\approx \Big(\frac{n}{|P|}\Big)^n \cdot \frac{\sqrt{2\pi n}}{\Big(\frac{n\sqrt{2\pi|P|}}{|P|\cdot e}\Big)^{n/|P|}}$$

$$\geq \Big(\frac{n}{|P|}\Big)^n \cdot \Big(\frac{\sqrt{|P|}\cdot e}{n\sqrt{2\pi}}\Big)^{n/|P|}$$

$$= \Big(\frac{n^{1-1/|P|} \cdot e^{1/|P|}}{|P|^{1-1/(2|P|)} \cdot \sqrt{2\pi}^{1/|P|}}\Big)^n$$

We have $2 \leq |P| \leq n/2$, so the fraction is at least 2, which makes the whole term certainly greater than n. □

In combination with our assumption that orbits of parents and children have size $\mathcal{O}(n)$, this lemma will help us to get a handle on the interplay of the supporting partitions of parent and child gates. We now define the quantity that we associate with each gate, as described above. This quantity is actually rather a vector, that we call the *size profile* (of the supporting partition). It is invariant under symmetries, so we define this measure not for individual gates but for their entire orbits. For a gate $g \in V_{C_n}$, we denote by $[g]$ its $\mathbf{Aut}(C_n)$-orbit in V_{C_n}. A size profile is a mapping $\zeta : \mathbb{N} \longrightarrow \mathbb{N}$. For an orbit $[g]$, we define $\zeta[g] : \mathbb{N} \longrightarrow \mathbb{N}$ as follows:

$$\zeta[g](i) := |\{P \in \mathbf{SP}_A(g) \mid |P| = i\}|.$$

Note that only for finitely many i, $\zeta(i) \neq 0$. Due to Lemma 9.2.3, this definition is indeed independent of the choice of the representative g of the orbit. Note that, as we promised earlier, the measure ζ differs considerably between the root and the input gates of a circuit. For the root r, $\zeta[r](n) = 1$, and $\zeta[r](i) = 0$ for all $i < n$. This is because our circuits are invariant under all permutations in $\mathbf{Sym}_n$ by assumption, so the root is stabilised by all permutations, and hence its coarsest alternating supporting partition contains just one large part. For the input gates, by contrast, we know that their supporting partition always has two parts: If an input gate is labelled with a hypercube edge, say, $\{0^a1^{n-a}, 0^{a-1}1^{n-a+1}\}$, then its supporting partition is $\{\{1, ..., a-1\}, \{a\}, \{a+1, ..., n\}\}$. Unless either a or b are very small, this partition has two large parts. We will prove that, with each layer in the circuit, the size profile of the gates cannot change very much, so for example, the largest part will grow by one, and another part will shrink by one, as we go one layer up in the circuit. As a consequence, in the middle of the circuit, we must encounter several different part sizes in the supporting partitions until we can reach the supporting partition $\{[n]\}$ at the root. In particular, we will encounter even part sizes in some gates, and when that happens, this more or less leads to an even number of paths. The next lemma is the key in our proof. It tells us precisely how the size-profiles of the gates can differ between children and parents. Essentially, the size of large parts can only change by at most one.

Lemma 9.2.7. *Let $0 < \varepsilon < 1$ be any constant. Let g a gate in C_n, h a parent of g. Assume that $|\mathbf{Orbit}_{(g)}(h)| \in \mathcal{O}(n)$ and $|\mathbf{Orbit}_{(h)}(g)| \in \mathcal{O}(n)$. Let $\Delta : \{m \in \mathbb{N} \mid m \geq \varepsilon \cdot n\} \longrightarrow \mathbb{N}$ be the function defined as $\Delta(s) := \zeta[h](s) - \zeta[g](s)$. For all large enough $n \in \mathbb{N}$ and every $s \geq \varepsilon \cdot n$ it holds:*

- $|\Delta(s)| \leq 2$.
- *If $\Delta(s) = 2$, then $\Delta(s+1) = -1, \Delta(s-1) = -1$. If additionally s is odd, then $|\mathbf{Orbit}_{(g)}(h)|$ is even.*
- *If $\Delta(s) = -2$, then $\Delta(s+1) = 1, \Delta(s-1) = 1$. If additionally s is even, then $|\mathbf{Orbit}_{(g)}(h)|$ is even.*
- *If no value of Δ is 2 or -2, then one of the following is possible:*
  - *If $\Delta(s) = 1$, then $\Delta(s-1) = -1$ or $\Delta(s+1) = -1$. In case that s is odd and $\Delta(s+1) = -1$, then $|\mathbf{Orbit}_{(g)}(h)|$ is even.*
  - *If $\Delta(s) = -1$, then $\Delta(s-1) = 1$ or $\Delta(s+1) = 1$. In case that s is even and $\Delta(s-1) = 1$, then $|\mathbf{Orbit}_{(g)}(h)|$ is even.*
- *If $\Delta(s) \neq 0$, then for all other s' except the ones mentioned in the cases above, it holds $\Delta(s') = 0$.*

Proof. Let $\mathbf{SP}^*_A(g) \subseteq \mathbf{SP}_A(g)$ be the set of parts of size $\geq \varepsilon \cdot n$ and $\mathbf{SP}^*_A(h) \subseteq \mathbf{SP}_A(h)$ the parts of size $\geq \varepsilon \cdot n - 1$ in $\mathbf{SP}_A(h)$.
Claim: There is a bijection $\gamma : \mathbf{SP}^*_A(g) \longrightarrow \mathbf{SP}^*_A(h)$ such that:

(a) For every $Q \in \mathbf{SP}^*_A(g)$, it holds $|\gamma(Q) \cap Q| \geq |Q| - 1$.

(b) For every $Q \in \mathbf{SP}^*_A(g)$, it holds $|\gamma(Q) \setminus Q| \leq 1$.

(c) There is at most one part $Q \in \mathbf{SP}^*_A(g)$ such that $|\gamma(Q) \cap Q| = |Q| - 1$.

(d) There is at most one part $Q \in \mathbf{SP}^*_A(g)$ such that $|\gamma(Q) \setminus Q| = 1$.

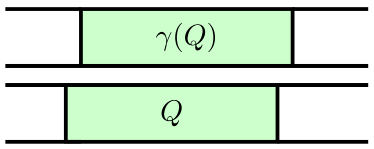


Figure 9.2: This is the most extreme way how Q and $\gamma(Q)$ may differ: Each has at most one element that is not shared with the other part.

Proof of claim: Construct γ by defining $\gamma(Q) \in \mathbf{SP}^*_A(h)$ as the part whose intersection with Q is largest possible. This is well-defined because there are only two cases how $\mathbf{SP}_A(h)|_Q$ can look like: It either consists of one part or we have $\mathbf{SP}_A(h)|_Q = \{\{s\}, Q \setminus \{s\}\}$, for some $s \in Q$. Everything else is ruled out by Lemma 9.2.6 and Theorem 4.3.1. This is because by assumption, only $\mathcal{O}(n)$ many parents of g are in $\mathbf{Orbit}_{(g)}(h)$. Therefore, the restriction of $\mathbf{SP}_A(h)$ to Q, denoted $\mathbf{SP}_A(h)|_Q$, can have at most $\mathcal{O}(n)$ many images under $\mathbf{Alt}(Q)$: Namely, every permutation in $\mathbf{Alt}(Q)$ extends to a circuit automorphism that fixes g because Q is a part in $\mathbf{SP}_A(g)$. Moreover, by Lemma 9.2.3, any two distinct $\mathbf{Alt}(Q)$-images of $\mathbf{SP}_A(h)|_Q$ must be the Q-restrictions of supporting partitions of distinct parents of g. So indeed, $\mathbf{Orbit}_{(g)}(h)$ contains at least as many gates as the size of the $\mathbf{Alt}(Q)$-orbit of $\mathbf{SP}_A(h)|_Q$. This size can only be in $\mathcal{O}(n)$ if $\mathbf{SP}_A(h)|_Q$ consists of singletons only or has just one big part or if it as a one-vs-rest split (because of Lemma 9.2.6, where we also use that $|Q| \in \Theta(n)$). The case that $\mathbf{SP}_A(h)|_Q$ consists only of singletons cannot happen because Q has linear size and by Theorem 4.3.1, $\mathbf{SP}_A(h)$ has at most $o(n)$ many singletons. Therefore, $\mathbf{SP}_A(h)|_Q$ indeed either consists of one part or we have $\mathbf{SP}_A(h)|_Q = \{\{s\}, Q \setminus \{s\}\}$, for some $s \in Q$. Thus, $\gamma(Q)$ is well-defined (and indeed, every part $\gamma(Q) \in \mathbf{SP}_A(h)$ has size $\geq \varepsilon \cdot n - 1$).

The above reasoning also directly proves statement (a). Statement (b) follows in a similar way because if it were not true, then $\mathbf{SP}_A(g)|_{\gamma(Q)}$ would have $\Omega(n^2)$ many images under $\mathbf{Alt}(\gamma(Q))$ (again by combining Theorem 4.3.1 and Lemma 9.2.6), resulting in $\mathbf{Orbit}_{(h)}(g)$ being too large.

We show that γ is injective: If it were not, then there would be some $P \in \mathbf{SP}^*_A(h)$ and $Q_1, Q_2 \in \mathbf{SP}^*_A(g)$ such that $|Q_1 \cap P| \geq |Q_1| - 1$ and $|Q_2 \cap P| \geq |Q_2| - 1$. This is impossible because then, by Lemma 9.2.6, $\mathbf{SP}_A(g)|_P$ has $\Omega(n^2)$ many automorphic images under $\mathbf{Alt}(P)$, but $\mathbf{Alt}(P)$ fixes h, so h has more than $\mathcal{O}(n)$ many children in $\mathbf{Orbit}_{(h)}(g)$, which is a contradiction.
Also, γ is surjective: Suppose there were a part $P \in \mathbf{SP}_A(h)$ of size $\geq g(n) - 1$ that has no preimage. Then $\mathbf{SP}_A(g)|_P$ must consist of parts smaller than $\varepsilon \cdot n$. Then again, $\mathbf{SP}_A(g)|_P$ has $\Omega(n^2)$ many images under $\mathbf{Alt}(P)$ by Lemma 9.2.6 (using also that the number of singleton parts in $\mathbf{SP}_A(g)$ is sublinear and hence less than $|P|$ by Theorem 4.3.1).

If statement (c) were not true, then there would be two parts $Q_1, Q_2 \in \mathbf{SP}^*_A(g)$ such that $|\gamma(Q_i) \cap Q_i| = |Q_i| - 1$. Then $\mathbf{Alt}(Q_1) \times \mathbf{Alt}(Q_2)$ fixes g but generates $\Omega(n^2)$ many distinct automorphic images of $\mathbf{SP}_A(h)$. Then again, $|\mathbf{Orbit}_{(g)}(h)|$ is greater than $\mathcal{O}(n)$, which contradicts the assumptions of the lemma.
Similarly, statement (d) is shown: If it were not true, then h would have too many children in $\mathbf{Orbit}_{(h)}(g)$. This proves the claim.

Now with the claim we see that there are five possible cases:

1. $\gamma(Q) = Q$ for all $Q \in \mathbf{SP}^*_A(g)$.

2. There is one part $Q \in \mathbf{SP}^*_A(g)$ such that $|\gamma(Q) \cap Q| = |Q| - 1$ and $\gamma(Q) \subseteq Q$, and for all other parts Q', $\gamma(Q') = Q'$.

3. There is one part Q with $|\gamma(Q) \setminus Q| = 1$ and $\gamma(Q) \supseteq Q$, and for all other parts Q', $\gamma(Q') = Q'$.

4. There is one part Q that satisfies $|\gamma(Q) \cap Q| = |Q| - 1$ *and* $|\gamma(Q) \setminus Q| = 1$. For all other parts Q', $\gamma(Q') = Q'$.

5. There is one part Q_1 that satisfies $|\gamma(Q_1) \cap Q_1| = |Q_1| - 1$ (and $\gamma(Q_1) \subseteq Q_1$), and another part Q_2 that satisfies $|\gamma(Q_2) \setminus Q_2| = 1$ (and $Q_2 \subseteq \gamma(Q_2)$), and for all other parts Q', $\gamma(Q') = Q'$.

In Case 1, the Δ-vector is zero.
In Case 2, we have $\Delta(|Q|) = -1$ and $\Delta(|Q| - 1) = 1$, and all other entries of Δ are zero.
In Case 3, we have $\Delta(|Q|) = -1$ and $\Delta(|Q| + 1) = 1$, and all other entries of Δ are zero.
In Case 4, the Δ-vector is zero.
In Case 5, we have to distinguish several cases. If $|Q_1| = |Q_2|$, then $\Delta(|Q_1|) = -2$ and $\Delta(|Q_1| - 1) = 1, \Delta(|Q_1| + 1) = 1$. If $|Q_1| \neq |Q_2|$ and $|Q_1| - 1 \neq |Q_2| + 1$, then $\Delta(|Q_1|) = -1, \Delta(|Q_2|) = -1, \Delta(|Q_1| - 1) = 1, \Delta(|Q_2| + 1) = 1$. If $|Q_1| = |Q_2| + 2$, then $\Delta(|Q_1|) = -1, \Delta(|Q_2|) = -1, \Delta(|Q_1| - 1) = 2$.

In Case 2, assume that $|Q|$ is even. Then $\mathbf{SP}_A(h)|_Q$ has an even number of images under $\mathbf{Alt}(Q)$ (because $\mathbf{SP}_A(h)|_Q$ has one singleton part and the rest, and this singleton can be mapped to all $|Q|$ positions by $\mathbf{Alt}(Q)$). We now want to argue that therefore, $|\mathbf{Orbit}_{(g)}(h)|$ must be even. Let $H(Q) \subseteq \mathbf{Orbit}_{(g)}(h)$ be the set of parents h' such that $\mathbf{SP}_A(h')|_Q$ consists of one singleton part and the rest. It holds that $|H(Q)|$ is even: Every $\pi \in \mathbf{Alt}(Q)$ extends to a $\sigma \in \mathbf{Aut}(C_n)$ that fixes g. By Lemma 9.2.3, this σ maps the parent h of g to another parent of g with $\mathbf{SP}_A(\sigma h)|_Q = \pi(\mathbf{SP}_A(h))|_Q$. So the $\mathbf{Alt}(Q)$-orbit of every element of $H(Q)$ is even; hence, H can be partitioned into $\mathbf{Alt}(Q)$-orbits, each of which is even, and so $|H(Q)|$ is even.

Now if $H(Q)$ is equal to the whole set $\mathbf{Orbit}_{(g)}(h)$, then we are done. Otherwise, $\mathbf{Orbit}_{(g)}(h)$ contains gates whose supporting partition on Q does not split into a singleton and the rest. Let $h' \in \mathbf{Orbit}_{(g)}(h)$ be such a gate. There must exist a permutation $\pi \in \mathbf{Sym}_n$ that extends to a circuit automorphism σ which maps h to h' and fixes g. So the corresponding π must stabilise the partition $\mathbf{SP}_A(g)$ setwise, and it will map $\mathbf{SP}_A(h)$ to $\mathbf{SP}_A(h')$. Therefore, $\pi(Q) \in \mathbf{SP}_A(g)$ is a part for which we will again have Case 2 when we apply the above reasoning to g and $h' = \sigma(h)$. Then we can define $H(\pi Q) \subseteq \mathbf{Orbit}_{(g)}(h)$ as the set of all parents whose supporting partition splits into singleton and rest on πQ, and we get that $|H(\pi Q)|$ is even. In total, with this reasoning we see that $\mathbf{Orbit}_{(g)}(h)$ is partitioned into even-size sets $H(\pi Q)$, for all $\pi \in \mathbf{Sym}_n$ which extend to circuit automorphisms that fix g and permute its parents. So in total, $|\mathbf{Orbit}_{(g)}(h)|$ is even.

Similarly, assume in Case 5 that $|Q_1|$ is even. Then the same argument shows that $|\mathbf{Orbit}_{(g)}(h)|$ is even. The lemma follows directly from these considerations. □

Corollary 9.2.8. *Let $0 < \varepsilon < 1$ be any constant. Let g be a gate in C_n (for large enough n), h a parent of g. Assume that $|\mathbf{Orbit}_{(g)}(h)| \in \mathcal{O}(n)$ and $|\mathbf{Orbit}_{(h)}(g)| \in \mathcal{O}(n)$, and that $|\mathbf{Orbit}_{(g)}(h)|$ is odd.*
Let s with $\varepsilon \cdot n \leq s < n$ be an even natural number. Then

$$\sum_{i \geq s} \zeta[h](i) \geq \sum_{i \geq s} \zeta[g](i)$$

Proof. According to Lemma 9.2.7, for any $i > s$, whenever $\zeta[h](i) < \zeta[g](i)$, then this is compensated by other values of $\zeta[h](j)$ in the sum $\sum_{i \geq s} \zeta[h](i)$.
It only remains to consider the case $\zeta[h](s) < \zeta[g](s)$. Since we are assuming that $|\mathbf{Orbit}_{(g)}(h)|$ is odd, and s is even, Lemma 9.2.7 implies that $\zeta[h](s) = \zeta[g](s) - 1$, and $\zeta[h](s+1) = \zeta[g](s+1) + 1$. Therefore, the sum $\sum_{i \geq s} \zeta[h](i)$ cannot be strictly less than $\sum_{i \geq s} \zeta[g](i)$. □

Intuitively speaking, this means that if along some path from the root to a gate g, the orbit size of the next parent gate in the stabiliser group of its child is always odd, then the number of large parts in the supporting partitions can only *increase* along the path towards the root. This will allow us to show that an even orbit must occur along each path. And this means that the path together with its automorphic images cancels itself out in the XOR computation.

When we look at a path $P = (r, h_1, h_2, ..., g)$ from the root r of a circuit to a certain gate g, then we can associate with P its *orbit-profile* $\Omega(P)$. This orbit profile says for every gate h_i on the path, which orbit its predecessor h_{i-1} belongs to. By orbit, we mean again $\mathbf{Orbit}_{(h_i)}(h_{i-1})$, so we refer to the partition of the parents of h_i into the orbits with respect to the subgroup of $\mathbf{Sym}_n$ that fixes h_i. The orbit profile of a path is not supposed to describe that path uniquely but we rather want that several paths share the same orbit profile – in a sense, we want the orbit profile to describe the "path" that we get when we factor out the respective orbits $\mathbf{Orbit}_{(h_i)}(h_{i-1})$. We have to show that this indeed makes sense:

Lemma 9.2.9. *Let g be a gate and h a parent of g. Let g' be another gate such that there is a $\sigma \in \mathbf{Aut}(C)$ with $g' = \sigma(g)$. In the partition of $E_C g'$ into orbits $\mathbf{Orbit}_{(g')}(h')$, for $h' \in E_C g'$, there is a unique orbit $\mathbf{Orbit}_{(g')}(h')$ to which $\mathbf{Orbit}_{(g)}(h)$ can be mapped by $\mathbf{Aut}(C)$.*

Proof. Firstly, it is clear that every $\sigma \in \mathbf{Aut}(C)$ that takes g to g' must map $\mathbf{Orbit}_{(g)}(h)$ to some orbit $\mathbf{Orbit}_{(g')}(h')$, for a $h' \in E_C g'$. We now show that there cannot be two distinct $\mathbf{Orbit}_{(g')}(h'), \mathbf{Orbit}_{(g')}(h'')$ that $\mathbf{Orbit}_{(g)}(h)$ can be mapped to. Suppose for a contradiction that there were $\sigma, \sigma' \in \mathbf{Aut}(C)$ with $\sigma(g) = \sigma'(g) = g'$ and $\sigma(\mathbf{Orbit}_{(g)}(h)) = \mathbf{Orbit}_{(g')}(h')$ and $\sigma'(\mathbf{Orbit}_{(g)}(h)) = \mathbf{Orbit}_{(g')}(h'')$. Then $\sigma' \circ \sigma^{-1}$ maps $\mathbf{Orbit}_{(g')}(h')$ to $\mathbf{Orbit}_{(g')}(h'')$ while fixing g'. Thus, $\mathbf{Orbit}_{(g')}(h') = \mathbf{Orbit}_{(g')}(h'')$, which is a contradiction because these orbits are distinct. □

Thus, for any gate g in C, and h a parent of g, we can define

$$\mathbf{Orbit}(\mathbf{Orbit}_{(g)}(h)) := \{\sigma(\mathbf{Orbit}_{(g)}(h)) \mid \sigma \in \mathbf{Aut}(C)\},$$

and this orbit of orbits contains exactly one $\mathbf{Orbit}_{(g')}(h')$ for every $g' \in \mathbf{Orbit}(g) = \{\sigma(g) \mid \sigma \in \mathbf{Aut}(C)\}$. The orbit profile $\Omega(P)$ of a path $P = (r = h_1, h_2, ..., h_\ell = g)$ is defined as

$$\Omega(P) := (\mathbf{Orbit}(\mathbf{Orbit}_{(h_\ell)}(h_{\ell-1})), \mathbf{Orbit}(\mathbf{Orbit}_{(h_{\ell-1})}(h_{\ell-2})), ..., \mathbf{Orbit}(\mathbf{Orbit}_{(h_2)}(h_1))).$$

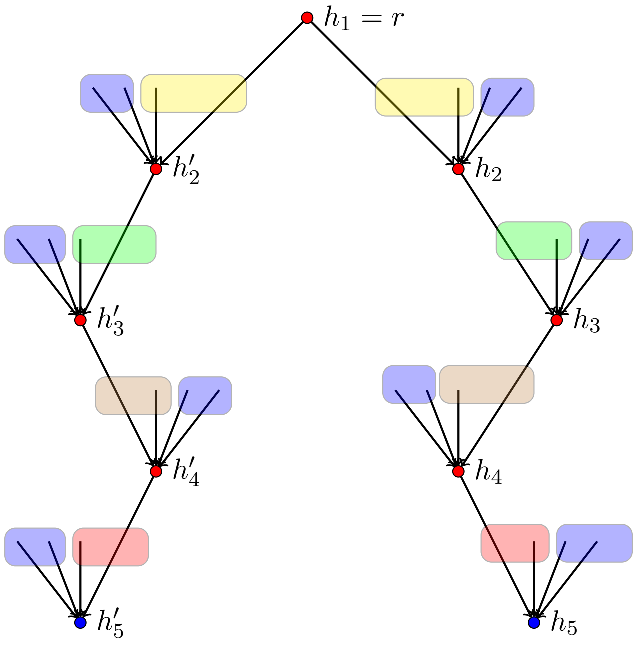


Figure 9.3: The colours indicate the partition of each set of parents into orbits. Orbits with the same colour belong to the same orbit of orbits. The orbit profile of the two paths in the picture is thus "red, brown, green, yellow". Both paths are related by a circuit automorphism.

Lemma 9.2.10. *Let $P = (r = h_1, ..., h_\ell = g)$ be a path from r to g in C_n. The number of paths in C_n from r to g with orbit-profile $\Omega(P)$ is exactly*

$$\prod_{2 \leq i \leq \ell} |\mathbf{Orbit}_{(h_i)}(h_{i-1})|$$

Proof. We go backwards from $g = h_\ell$ to r and count how many ways there are to construct a path with orbit-profile $\Omega(P)$. In the beginning, there are $|\mathbf{Orbit}_{(h_\ell)}(h_{\ell-1})|$ many options to choose a predecessor of g that is in the orbit required by $\Omega(P)$. Let h be the predecessor of g that we choose. From there, we have $|\mathbf{Orbit}_{(h_{\ell-1})}(h_{\ell-2})|$ predecessors that we could continue with in a way that respects $\Omega(P)$. To see this, we use Lemma 9.2.9: No matter which gate we chose for h, it is in $\mathbf{Orbit}(h_{\ell-1})$. Therefore, by Lemma 9.2.9, there exists a unique $\mathbf{Orbit}_{(h)}(h')$ in $\{\mathbf{Orbit}_{(h)}(h') \mid h' \in E_C h\}$ that is also a member of $\mathbf{Orbit}(\mathbf{Orbit}_{(h_{\ell-1})}(h_{\ell-2}))$. From this $\mathbf{Orbit}_{(h)}(h')$, we can choose the next gate on our path, and this orbit has the same size as $\mathbf{Orbit}_{(h_{\ell-1})}(h_{\ell-2})$. Hence, we have so far $|\mathbf{Orbit}_{(h_\ell)}(h_{\ell-1})| \cdot |\mathbf{Orbit}_{(h_{\ell-1})}(h_{\ell-2})|$ possibilities to go two steps from g towards r in a way that complies with the orbit-profile $\Omega(P)$. In the same fashion, we continue counting until we reach the root, and obtain the number of paths that is stated in the lemma. □

Lemma 9.2.11. *Let $0 < \varepsilon \leq 1$. For each n, fix a gate g_n in C_n such that $\mathbf{SP}_A(g_n)$ has at least two parts of size $\geq \varepsilon n$. For every possible orbit-profile $\Omega(P)$ that any path P from the root of C_n to g_n can have, there exists an even number of distinct paths from the root to g_n with exactly that orbit-profile.*

Proof. Fix a path P from the root to g_n in C_n and the corresponding orbit-profile $\Omega(P)$. We are going to show that there exists an even number of distinct paths from the root to g_n with orbit-profile $\Omega(P)$.
By the assumption on g_n, it holds $\zeta[g_n](s_1) \geq 1$ and $\zeta[g_n](s_2) \geq 1$ for $s_1, s_2 \geq \varepsilon \cdot n$. For the root r_n it holds $\zeta[r_n](n) = 1$ and $\zeta[r_n](s) = 0$ for every $s \neq n$ (because the root is fixed by all permutations in $\mathbf{Sym}_n$). Therefore, the size profiles ζ must change along the path P from g to r. Let s be an even natural number such that $\varepsilon \cdot n \leq s \leq \min\{s_1, s_2\}$. This always exists because otherwise we can just make ε a bit smaller such that $\min\{s_1, s_2\} - 1 \geq \varepsilon \cdot n$. Assume for a contradiction that for every gate $h \in P$, for its predecessor h' on the path P it holds: $|\mathbf{Orbit}_{(h)}(h')|$ is odd. Then applying Corollary 9.2.8 inductively along the path P shows that $\sum_{i \geq s} \zeta[h](i) \geq \sum_{i \geq s} \zeta[g_n](i) \geq 2$, for every $h \in P$. This is a contradiction to the fact that $\sum_{i \geq s} \zeta[r_n](i) = 1$.
This shows that there must be some h on the path P such that the predecessor h' of h satisfies: $|\mathbf{Orbit}_{(h)}(h')|$ is even. Then the total number of paths from r_n to g_n with profile $\Omega(P)$ is even because by Lemma 9.2.10, this number is a product containing the even number $|\mathbf{Orbit}_{(h)}(h')|$. □

From this, our main technical theorem follows, which states that not only the number of paths with a given orbit profile, but the total number of paths from r to g is even:

Proof of Theorem 9.2.5: Every path from r_n to g_n has exactly one orbit-profile. Hence, the number of paths from r_n to g_n is just

$$\sum_{\Omega \text{ an orbit profile of a path from } r_n \text{ to } g_n} \#(\Omega),$$

where $\#(\Omega)$ denotes the number of paths with orbit-profile Ω that end in g_n. By Lemma 9.2.11, all summands in this sum are even. □

Finally, let us summarise why Theorem 9.2.1 (“fully symmetric XOR-circuits are insensitive to all inputs except those labelled with very imbalanced binary strings”) follows from Theorem 9.2.5.

Proof of Theorem 9.2.1:
Fix any $\varepsilon > 0$. Let g_n be an input gate of C_n labelled with a hypercube-edge $\ell(g_n) = \{0^a 1^{n-a}, 0^{a-1}1^{n-a+1}\}$ such that $a \geq \varepsilon n$ and $n - a \geq \varepsilon n$. It is easy to see that $\mathbf{SP}_A(g_n) = \{\{1, ..., a-1\}, \{a\}, \{a+1, ..., n\}\}$. This contains two parts of size $\geq \varepsilon n$, so Theorem 9.2.5 applies and the number of paths from the root to g_n is even. By Lemma 9.2.4, the circuit C_n is not sensitive to the input gate g_n. $\square$

9.3 Conclusion and future research

We have made progress towards showing that no CFI-symmetric (and super-symmetric) CPT-algorithm defines the CFI-query over unordered hypercubes: According to the results from Chapter 8, the existence of such an algorithm would entail the existence of a family of symmetric XOR-circuits whose sizes and orbit sizes are polynomial in the size of the n-dimensional hypercube (namely 2^n), which compute the XOR over $\Omega(2^n/n^{1.5})$ many input bits, and whose fan-in dimension is bounded by $\mathcal{O}(n)$ (i.e. logarithmic in the hypercube size). Our aim was to show that such circuit families do not exist. We have not fully accomplished this but at least we have identified interesting further restrictions on the circuits which altogether are unsatisfiable: If the orbit size of each circuit is assumed to be exactly 1 instead just polynomial, and instead of the fan-in dimension bound of $\mathcal{O}(n)$, we impose an $\mathcal{O}(n)$-bound on the number of children and parents of each gate (per orbit in the stabiliser of the gate), then these circuits cannot compute the XOR over $\Omega(2^n/n^{1.5})$ many input bits. It follows that if nonetheless there does exist a CFI-symmetric algorithm for the hypercube CFI-query, then the corresponding circuit families either have orbit size > 1 or must violate the orbit-wise bound on the number of child or parent gates. Thus, the next step should be to try and lift our techniques developed in this chapter to a more general setting. It seems plausible that this can be done but there are technical challenges involved:

The first problem is how to argue for circuits whose orbit size is not exactly 1, but bounded by some polynomial in 2^n. Then the supporting partition of the root does not necessarily consist of only one part, but it can be many more (although if it is too many, then the orbit size will be greater than 2^{nk}, which is forbidden). Our argument exploited the fact that the number of linear-size parts in the supporting partition can never decrease along a path from an input gate to the root unless the number of parents is even at some point. But if the supporting partition of the root can now have multiple linear-size parts, then this no longer leads to a contradiction. It might be that with a much more careful analysis of the circuits, our argument could still be recovered in this case, though. Our key technical lemma (Lemma 9.2.7) is actually stronger than what we needed in our proof because it gives us several cases in which the number of parents of a gate must be even. Thus, if even parent numbers are forbidden, then the ways in which the size profiles of the supporting partitions can change along a path are very limited.

But surely we can expect that not all gates have the same size profile, so changes will occur somewhere, and then again, this will lead to even parent numbers. It is just not clear at this moment how to turn this into a formal argument.

The second problem concerns the relationship between the logarithmic bound on the *fan-in dimension*, that we get from Theorem 9.1.5, and on the logarithmic orbit-wise fan-in and fan-out bounds that we imposed in this chapter. Currently, we do not know if one of these bounds implies the other. Probably, the bound on the parent number is not directly related to fan-in dimension but the bound on the children might be. It would be nice if logarithmic fan-in dimension implied a logarithmic number of children per orbit. Then we would have this covered with the result of this chapter. In case that the gates in $\mathbf{Orbit}_{(h)}(g) \subseteq hE_C$ all have distinct sensitivity sets $\mathcal{X}(g')$, which are also linearly independent as vectors in $\mathbb{F}_2^E$, then $|\mathbf{Orbit}_{(h)}(g)|$ is indeed at most the fan-in dimension. But it is unclear how to reason about the properties of these sets $\mathcal{X}(g')$, for all $g' \in \mathbf{Orbit}_{(h)}(g)$.
For removing the $\mathcal{O}(n)$-bound on the orbit-wise parent number of the gates, we have a rough idea. Namely, because our circuits are single-rooted, their levels should get narrower closer to the root. Therefore, it seems plausible that close enough to the root, each gate indeed only has a bounded number of parents because otherwise, the circuit would get wider. The good thing about our even-paths theorem (Theorem 9.2.5) is that it can be applied to any gate in the circuit, not only input gates. So we could potentially focus on the top-most part of the circuit, where its levels only get narrower, and could show that in this top part, all paths cancel each other out. This would suffice to show that the circuit is not sensitive to enough input bits.

All in all, it feels like our even-paths technique has more potential and might also work for less restricted circuit classes, perhaps even for *all* circuits satisfying the necessary properties for the existence of a CFI-symmetric algorithm for the hypercube CFI-problem. In particular, it might also be possible to improve our Theorem 4.3.1 from the group-theory chapter, which says that the alternating supporting partitions can not have linearly many singleton parts. This was what we needed in our proof in this chapter but actually, the lemmas we used in the proof of Theorem 4.3.1 can probably lead to stronger statements. For example, the key group-theoretic Lemma 4.3.8 says that every alternating supporting partition has a part of linear size. Perhaps, an iterated application of that lemma could lead to a good upper bound on the number of parts in the alternating supporting partitions here. This would give us more information about the relevant stabiliser groups, which could potentially be helpful. So we seem to be in the situation where we probably have not yet reached the limitations of our technique, but nonetheless, making further progress might be technically very challenging.

Finally, let us briefly return to the question about tree-like circuits, that we mentioned earlier in this chapter. Are tree-like circuits with logarithmic fan-in dimension (or perhaps, simply logarithmic fan-in degree) a sufficiently symmetric candidate or is there a way to rule them out? The results from this chapter indicate that tree-like circuits with logarithmic fan-in degree cannot have orbit size one: Then they would satisfy the preconditions of Theorem 9.2.1 because in a tree, every gate has just one parent. Hence, such trees would not be sensitive to enough input bits, even though a treelike XOR-circuit *is* of course sensitive to all its inputs because none cancel out. Therefore, we can conclude that trees with $\Theta(2^n/n^{1.5})$ many input gates can simply not be symmetric with respect to the action of $\mathbf{Sym}_n$ on $\{0,1\}^n$; at least, not in the sense that *all* permutations in $\mathbf{Sym}_n$ extend to circuit automorphisms. We can take this as a hint that perhaps, the orbit size of such trees is generally not even polynomial in 2^n, which would rule them out as candidates. We currently have no proof for this but it seems like a question that can realistically be solved.

Another direction that may be interesting to investigate is in how far the new lower bound technique against symmetric XOR-circuits that we developed here can be applied to other scenarios as well. For example, studying lower bounds for symmetric circuits also seems to be a promising approach towards separating the algebraic complexity classes VNP and VP. There exist lower bounds against symmetric arithmetic circuits for computing the determinant and permanent polynomials by Dawar and Wilsenach [43, 41]. They raise the question in how far these lower bounds can be improved to weaker symmetry groups, and perhaps our technique can be adapted to that end. Of course, the even-paths theorem is probably only useful for circuits which purely consist of XOR-gates; but the statement that the alternating supporting partitions of the gates cannot change much between the layers could lead to new insights. A novelty of our technique in comparison with [6, 42, 43, 41] is that it does not use any “support theorem”. Support theorems are a key ingredient in all these previous works, and they usually state that any gate in a highly symmetric circuit is supported by a constant number of elements of the permutation domain. For poly-size circuits with hypercube-symmetries – which is a weaker kind of symmetry than what is studied in the aforementioned articles – we believe that a support theorem in that strong form does not hold. Thus, our approach via alternating supporting partitions might perhaps open up a perspective to study such weaker symmetry groups as well.

10 Choiceless Polynomial Time and Propositional Proof Complexity

In this chapter, we take a different perspective on Choiceless Polynomial Time and relate its power to that of a propositional/algebraic proof system. The main consequence is that super-polynomial lower bounds for deciding graph non-isomorphism of CFI-graphs or multipedes in this proof system, the *degree-*3 *extended polynomial calculus*, would separate CPT from PTIME. We show this by simulating the runs of CPT-programs that distinguish non-isomorphic graphs in the extended polynomial calculus. This approach extends prior research on the connections between finite model theory and *proof complexity* ([90] and [17]). Because proof complexity has not appeared elsewhere in this thesis, we start with a brief introduction to the subject. For a more thorough survey, see for example [100] or [16] or the textbook [79].

Propositional proof complexity A *propositional proof system* can be informally understood as a collection of syntactical inference rules that are used to derive new "proof *lines*" from already derived ones or from the input. The format of these lines depends on the proof system. For example, in the well-known resolution calculus, the lines are propositional clauses. In the *polynomial calculus* (PC), that we consider here, the lines are polynomials over $\mathbb{Q}$, that are understood as polynomial equations of the form $p(X_1, ..., X_n) = 0$. Typically, the instances that one wishes to solve with a proof system are propositional formulas, potentially converted into, say, CNF or a set of polynomial equations over $\{0, 1\}$-variables. The question is whether the given instance is unsatisfiable. A proof of unsatisfiability, also called a *refutation*, is usually a derivation of a certain obviously unsatisfiable proof line from the input. In the case of resolution, this is the empty clause, whereas in the polynomial calculus, it is the equation $1 = 0$. There are also proof systems that work the other way round, i.e. they can be used to prove that a given formula is a tautology by deriving it from a fixed set of axioms, that are tautological. More abstractly, Cook and Rechow defined the notion of a propositional proof system as a polynomial time computable function that takes as input a propositional formula and a certificate of it being a tautology (or unsatisfiable) and correctly checks the certificate. Proof systems are usually sound and complete, so a certificate that is accepted by this verifier exists if and only if the input formula is a tautology/unsatisfiable. The *complexity* of such a proof system is then the smallest upper bound that can be put on the length of the certificates in terms of the length of the instance. Thus, the existence of a proof system with polynomial complexity is equivalent to $\overline{\text{SAT}}$ being in NP. So if a proof system that admits short proofs for all tautologies is found, this would prove that coNP = NP. This – and the fact that proof systems such as resolution or the polynomial calculus form the basis for many real-world SAT-solvers – explains why so much effort is put into proving lower bounds in proof complexity. Potentially, one day, super-polynomial lower bounds for *all* propositional proof systems can be shown, which would separate coNP from NP, or this line of research might lead to the discovery of a proof system with polynomial complexity, which would mean that coNP = NP.

For resolution and also for the polynomial calculus, super-polynomial lower bounds are known (e.g. [68] and [77]). An important family of proof systems, for which such bounds have still not been found, are so-called *Frege* systems. This is a general term for all "classical " propositional proof systems in the style of, for example, Gentzen's sequent calculus. Such systems all polynomially simulate each other, so the precise set of inference rules does not matter too much. The main difference between Frege and, for instance, resolution, is that the lines of a Frege proof are not limited to clauses, but may consist of arbitrary propositional formulas. There also exists *extended* Frege, which refers to the standard Frege proof systems with the additional power to introduce new variables as abbreviations for complex formulas in a proof. The power and limitations of this proof system are even less clear than for Frege itself. In fact, not even the relationship between Frege and extended Frege is known. It is conjectured that extended Frege is super-quasipolynomially stronger than Frege but a separating example has not been found yet [28].

What we consider in this chapter is the *polynomial calculus with extension axioms*, where – analogously to extended Frege – the standard polynomial calculus is augmented with axioms for the introduction of new variables. Formally, these axioms read $\frac{}{X_f - f}$, where f is any polynomial that does not contain the variable X_f. The axiom is understood as the equation $X_f - f = 0$, so it simply says that the new variable X_f must have the same value as the polynomial f. This proof system is still relatively unexplored, compared to the polynomial calculus itself but there does exist a fairly recent super-polynomial lower bound by Yaroslav Alekseev [5] for the so-called bit-value principle $1 + x_1 + 2x_2 + ... + 2^{n-1}x_n = 0$. Unfortunately, this seems to be unrelated to the graph isomorphism problem that we consider here, and so, Alekseev's techniques are probably not applicable to separate CPT from P in combination with our Theorem 10.0.3. In particular, his proof uses a coefficient blow-up argument, and it seems unlikely that such a thing could happen in refutations of the graph isomorphism polynomials. We provide more details on the extended polynomial calculus in Section 10.1.

Connections to finite model theory Many lower bounds in the history of proof complexity were obtained using specific combinatorial arguments tailored to the proof systems and instances at hand, but in fact, some of them can alternatively be proven with more general finite-model-theoretic tools such as the bijective k-pebble game. Conversely, lower bounds for typical logics from finite model theory, such as fixed-point logic or fixed-point logic with counting, can in principle also be deduced from proof-theoretic lower bounds. This connection between finite model theory and proof complexity is explored in detail

in [90]. In that article, restricted variants of resolution and the polynomial calculus are studied: If one restricts the width (that is, the number of literals) of the clauses that may be used in a resolution refutation to some constant k, then the existence of a width-k refutation of a given input CNF can be checked in polynomial time. This holds simply because the number of possible width-k clauses is polynomially bounded in the number of input variables. Similarly, if the polynomials in a polynomial calculus refutation are restricted to a constant degree k, then proofs can be efficiently computed with the Gröbner basis algorithm [32]. Thus, these restrictions form natural fragments of the proof systems that enable efficient deterministic proof search, while sacrificing completeness. As shown in [90], the existence of width-k resolution refutations or degree-k polynomial calculus refutations can even be defined in fixed-point logic, or fixed-point logic with counting, respectively (using $\mathcal{O}(k)$ variables). A similar statement holds for the definability of sums-of-squares (SOS) proofs in FPC [13].
Therefore, if the structural representations of two given input CNF-formulas or polynomial equation systems are indistinguishable in k-variable FO/k-variable counting logic, then also width-k resolution/degree-k polynomial calculus cannot distinguish them: Either, both are refutable or none of them are. Hence, these proof systems fail on families of unsatisfiable formulas which possess a respective $\mathcal{C}^k$-equivalent (with respect to the structural representation) counterpart that *is* satisfiable. In this way, finite-model-theoretic methods can be used to show that, for example, graph non-isomorphism for CFI-graphs, or the pigeonhole principle, are not refutable in width-k resolution/degree-k PC. The other direction holds as well: The evaluation of fixed-point formulas on finite structures can be simulated in width-k resolution/degree-k polynomial calculus (the latter is needed for fixed-point logic with counting). More precisely, for any fixed FPC-sentence ψ, there exists an FO-interpretation that maps any structure $\mathfrak{A}$ (with a number sort) of matching signature to the structural representation of a polynomial equation system that is refutable in degree-k PC if and only if $\mathfrak{A} \models \psi$. That means, up to a rather weak FO-definable precomputation, the polynomial calculus can simulate fixed-point iterations of FO-formulas with counting [90].

Results of this chapter A similar result was obtained before in [17], where Berkholz and Grohe focused on the *graph isomorphism problem*. A proof system, such as the polynomial calculus, is said to distinguish two graphs G and H, if it can refute the statement "the graphs G and H are isomorphic", formally encoded as a polynomial equation system $P_{\text{iso}}(G, H)$ – see Definition 10.2.1. The main result of Berkholz and Grohe is that the degree-k polynomial calculus (or actually a certain restriction thereof) can distinguish G and H if and only if G and H can be distinguished by the $(k-1)$-dimensional Weisfeiler Leman algorithm. Since the power of the $(k-1)$-dimensional Weisfeiler Leman method is characterized by that of $\mathcal{C}^k$, this means that the degree-k PC cannot distinguish more graphs than the logic $\mathcal{C}^k$. In this chapter, we lift one direction of this result to CPT and the *extended degree-3 polynomial calculus* over $\mathbb{Q}$ (denoted EPC$_3$): If CPT distinguishes all graphs in a given class $\mathcal{K}$, then so does EPC$_3$. Moreover, the corresponding EPC_3-refutations have polynomial size and use extension axioms only for expressions of a certain limited form. The main consequence is that super-polynomial EPC_3 lower bounds for graph isomorphism on some graph class $\mathcal{K}$ would imply that CPT cannot distinguish these graphs, either. If $\mathcal{K}$ is some class of CFI-graphs or multipedes [67], for which isomorphism is in polynomial time, then this would separate CPT from P. This is the main result of this chapter:

Theorem 10.0.1. *Let $\mathcal{K}$ be a class of binary structures such that* CPT *distinguishes all graphs in $\mathcal{K}$. Then the degree-*3 *extended polynomial calculus over $\mathbb{Q}$ (denoted* EPC_3*) distinguishes all graphs in $\mathcal{K}$ with refutations of polynomial size.*
Moreover, the EPC_3*-refutation uses only extension axioms $\overline{X_f - f}$ for polynomials of the form $f = X \cdot Y$ or $f = \frac{1}{n} \cdot \left(\sum_{i=1}^{n^2} X_i \right)$. That is, only monomials and certain "averaged sums" are replaced with new variables.*

When we say that CPT distinguishes all structures in a class $\mathcal{K}$, we mean that for any two non-isomorphic graphs in that class, there exists a distinguishing CPT-sentence:

Definition 10.0.2 (Distinguishing relational structures in CPT)**.** *Let $\mathcal{K}$ be a class of τ-structures. We say that* CPT *distinguishes all structures in $\mathcal{K}$ if there exists a polynomial $p(n)$ and a constant $k \in \mathbb{N}$ such that for any two structures $G, H \in \mathcal{K}$ which are non-isomorphic, there exists a sentence $\Pi \in \mathrm{CPT}(p(n))$ with $\leq k$ variables such that $G \models \Pi$ and $H \not\models \Pi$.*

This definition is perhaps a bit unexpected because the distinguishing sentence does not have to be the same one for all pairs of graphs in $\mathcal{K}$. This is so because when we refute graph isomorphism in the polynomial calculus, then the refutation is also a different one for each pair of non-isomorphic graphs; the distinguishing CPT-sentences will play the role of the refutations. We have to require that all these distinguishing sentences must obey the same polynomial resource bound and use only a constant number of variables. Otherwise, we could distinguish any pair of non-isomorphic finite graphs already in FO, by explicitly defining the graphs up to isomorphism.

So by this definition, if CPT distinguishes all graphs in $\mathcal{K}$, it does not necessarily mean that there also exists a *single* CPT-program that decides the graph isomorphism problem on that class (because it might have to guess the right distinguishing sentence). But if CPT does not distinguish the graphs in $\mathcal{K}$, then it cannot possibly decide isomorphism for $\mathcal{K}$ with a single program. Therefore, we directly get the desired consequence:

Theorem 10.0.3. *If there is a class $\mathcal{K}$ of graphs on which the isomorphism problem is in* Ptime*, but which cannot be distinguished in* EPC_3 *with polynomial-size refutations using only extension axioms of the form mentioned in Theorem 10.0.1, then* CPT $\neq$ Ptime*.*

A second, though perhaps less interesting, consequence of Theorem 10.0.1 can be obtained by applying the theorem the other way round, from finite model theory to proof complexity: Our restricted version of EPC_3 that we consider here is strictly more powerful than the degree-k PC without extension axioms, for every constant k:

Theorem 10.0.4. *There exists a sequence of pairs of non-isomorphic graphs $(G_n, H_n)_{n \in \mathbb{N}}$ such that $P_{iso}(G_n, H_n)$ has a polynomial-size refutation in the degree-3 extended polynomial calculus (using only extension axioms of the aforementioned form) but there is no $k \in \mathbb{N}$ such that the degree-k polynomial calculus can refute $P_{iso}(G_n, H_n)$ for all n.*

We show in Section 10.3 how this theorem follows from Theorem 10.0.1 together with the definability of the CFI-query over ordered base graphs (see Theorem 6.1.1 in Chapter 6) and a polynomial calculus lower bound stated in [17].
To our knowledge, the separation of these two bounded-degree proof systems has not explicitly been stated before – specifically for the graph isomorphism problem. In the *unbounded-degree* setting, an exponential separation between PC and EPC is known, even if one only allows extension axioms of the form $\overline{X} = 1 - X$, as in *polynomial calculus resolution* [44]. Nevertheless, this is another nice example of how results from finite model theory can lead to new insights in proof complexity. Similar examples are the alternative finite-model-theoretic proofs for previously known proof complexity lower bounds in [90].

It should be noted that the connection between CPT and EPC_3 that we have discovered here is weaker than the results in [17] or [90]. While in [17], the correspondence goes in both ways – graph distinguishability in the degree-k PC also implies distinguishability in $\mathcal{C}^k$ – we believe that this is not true for EPC_3 and CPT. This has two reasons: Firstly, EPC_3 is a priori not symmetry-invariant, whereas CPT is, and secondly, we do not know of any efficient deterministic procedure for computing EPC_3-refutations. For the bounded-degree polynomial calculus without extension axioms, this is possible, but for EPC_3, we would probably have to guess the right set of extension axioms to be used and then apply the Gröbner basis algorithm to deterministically compute the refutation. Hence, the existence of EPC_3 proofs for graph non-isomorphism does not seem to be CPT-definable, unless CPT were extended with some kind of non-determinism and EPC_3 were restricted to a symmetric fragment. Potentially, a suitable non-deterministic extension for this could be NPIL, which we sketched in Section 3.3. This logic allows to guess which symmetry-invariant operation is to be executed in each step. As a side remark, we also suspect that Definition 10.0.2 could be phrased quite naturally in terms of NPIL: While we now allow to choose a different CPT-program for every pair of graphs in $\mathcal{K}$ that is to be distinguished, we could probably use a *single* NPIL-sentence that distinguishes every pair of graphs: This sentence would simply guess and execute the suitable distinguishing deterministic program for the two input graphs.

Another difference between the results here and those in [90] is that the mutual simulation of degree-k PC and FPC from [90] is not limited to the graph isomorphism problem. It works for *every* FPC-sentence and *every* polynomial equation system. We do not see a way how Theorem 10.0.1 could be lifted to this effect. That is, we do not know if *all* CPT-definable properties of finite structures can also be decided by EPC_3 on some definable polynomial axiom system. If this were possible, then we would obtain more candidate polynomial axiom systems, for which EPC_3 lower bounds imply CPT $\neq$ PTIME. However, the most promising candidate problems for separating CPT from P seem to be instances of graph isomorphism anyway (e.g. the CFI query). Also, the aforementioned bit-value principle, for which an EPC lower bound is known, does not seem like it can be used to express any meaningful CPT-computations in EPC. So in any case, there is not much hope to use that particular lower bound by Alekseev [5] against CPT, even if Theorem 10.0.1 could be extended to other problems than graph isomorphism.

Actually, a realistic chance to make Theorem 10.0.1 more powerful would be to state that the resulting EPC_3-refutations for graph isomorphism are *symmetry-invariant* in a certain sense. This can be seen by inspection of our proof. Basically, it then follows that already a super-polynomial lower bound for graph isomorphism in some *symmetric fragment* of EPC_3 would suffice to separate CPT from P; the problem is that our results do not lead to a clear and general definition of what *symmetric* EPC_3 should be. We discuss this topic in more detail in the last section of this chapter and explain one potential notion of symmetric refutations that fits to our setting here, but seems hard to generalise to other problems than graph isomorphism.

For the proof of Theorem 10.0.1, we have to take a detour via the relatively new computation model *Deep Weisfeiler Leman* (DWL). It has recently been introduced by Grohe, Schweitzer and Wiebking [63] and was shown to have the same expressive power as CPT. Roughly speaking, a DWL algorithm is a polynomial time Turing machine that operates on a given finite input structure. It has read and write access to that structure, but only in an isomorphism-invariant way. Essentially, the Turing machine only "sees" the 2-dimensional Weisfeiler Leman colouring of the structure. For our purposes, this connection to the Weisfeiler Leman algorithm is extremely useful because it allows us to apply the machinery from [17] that relates Weisfeiler Leman with the polynomial calculus. Furthermore, in the DWL computation model, it becomes much more apparent what it really means to distinguish graphs in CPT. Since DWL is technically quite involved, a large part of this chapter (Section 10.4) consists in its presentation and proofs of several of its properties. Only after that, we show how to actually construct an EPC_3-refutation for graph isomorphism, given a CPT- and thus a DWL-program that distinguishes the given graphs. In short, the key idea behind the EPC_3-refutation is to represent the higher-order objects constructed by the distinguishing CPT sentence with extension variables that stand for suitable polynomials. Ordered tuples of elements will be represented by degree-2 monomials, i.e. products of the elements. Unordered sets correspond to averaged sums of their elements. This is also the reason why only extension axioms as mentioned in Theorem 10.0.1 are needed. Using this encoding of higher-order objects, we can derive from a given polynomial equation system $P_{\text{iso}}(G, H)$ (see Definition 10.2.1) the equation system $P_{\text{iso}}(G', H')$, where G' and H' are the input graphs augmented with new CPT-definable h.f. sets. Thus, refuting isomorphism of G and H reduces in EPC_3 to the isomorphism problem on any CPT-definable higher-order objects over these graphs. This problem can then be solved in the degree-3 polynomial calculus using the results from [17].

In this chapter, relations are always binary. Whenever unary relations are needed, for example to encode vertex-colours, we represent them with self-loops of binary relations. For the Deep Weisfeiler Leman computation model, we closely follow the presentation in [63] and also adopt the notation, even though it sometimes deviates from the conventions in the rest of the thesis. For example, in this section we denote structures as A rather than $\mathfrak{A}$. For a relation symbol R, $R(A)$ denotes the corresponding relation in the structure A. The universe of A is denoted $V(A)$. The induced substructure of A with universe $W \subseteq V(A)$ is denoted $A[W]$.

We now begin by formally defining the extended polynomial calculus and reviewing the relevant material from [17] about its application to the graph isomorphism problem.

10.1 The (extended) polynomial calculus

The polynomial calculus was introduced in [32]. It is applicable to the following problem: Given a set P of multivariate polynomials over a fixed field (in our case, $\mathbb{Q}$), decide if the polynomials in P have a common zero with respect to $\{0,1\}$-assignments. The polynomial 1 is derivable from P if and only if the polynomials in P have no common zero over $\{0,1\}$. A derivation of the 1-polynomial is formally a sequence $p_1, p_2, ..., p_n = 1$ of polynomials such that each p_i is either in P or an axiom of the polynomial calculus or is obtained from one or multiple p_j, for $j < i$, with the application of a derivation rule. A derivation of the 1-polynomial from P is called a *refutation* of P (or a *proof* that P is not satisfiable).
A restricted variant of the polynomial calculus, the monomial calculus, has been introduced in [17]. The derivation rules of the polynomial/monomial calculus are the following:

Definition 10.1.1 (Inference rules of the (extended) polynomial calculus)**.** *Let P be the set of input polynomials/axioms, $a, b \in \mathbb{Q}$, X a variable, and f, g polynomials with rational coefficients.*

$$\frac{p \in P}{p}\ \textit{(Axioms)} \qquad \frac{}{X^2 - X}\ \textit{(Boolean axioms)}$$

$$\frac{f}{Xf}\ \textit{(Multiplication rule)} \qquad \frac{g \quad f}{ag + bf}\ \textit{(Linear combination rule)}$$

In the extended polynomial calculus *(*EPC*),* extension axioms *of the form* $\overline{X_f - f}$ *may be used, subject to the following condition: For any* EPC*-refutation, there must exist a linear order $<$ on the occurring extension variables such that for every extension axiom $\overline{X_f - f}$ used in the refutation, f contains only extension variables $X_g < X_f$. The Boolean axioms do not apply to extension variables.*

The *monomial calculus* (MC) is a restriction of PC that permits the use of the multiplication rule only in the cases where f is either a monomial or the product of a monomial and an axiom. The Boolean axioms are part of the polynomial calculus because the domain of the variables is always assumed to be $\{0,1\}$.

The *size* of a refutation $p_1, p_2, ..., p_n = 1$ is the total number of occurrences of monomials in all its polynomials. Its *bit-complexity* is the maximum number of bits required to represent any of the occurring coefficients, where values in $\mathbb{Q}$ are stored as a fraction of two binary numbers.

In general, derivations of the 1-polynomial may have super-polynomial size. This is for example the case in refutations of the pigeonhole principle [77]. Hence, one often considers the degree-k versions of the proof systems, which we denote by $\mathrm{MC}_k, \mathrm{PC}_k$ and EPC_k. In these proof systems, only polynomials of degree $\leq k$ may occur in a refutation. As already said, proofs in $\mathrm{MC}_k, \mathrm{PC}_k$ can be efficiently found using the Gröbner basis algorithm, provided that the representations of the occurring coefficients do not require a super-polynomial number of bits. Of course, when we bound the degree by a constant, we lose completeness, because e.g. the pigeonhole principle does not admit proofs of sublinear degree [92]. For the extended polynomial calculus, however, it is – to the best of our knowledge – not known whether the bounded-degree fragment is strictly weaker. It may be the case that for some fixed k, EPC_k simulates EPC (perhaps even with only a polynomial increase in the refutation size).
Intuitively, the effect of the extension axioms in the bounded-degree proof system EPC_3 is that the degree-bound of 3 may be "locally" violated: Monomials like $A \cdot B \cdot C \cdot D$ can be written as $X_{AB} \cdot X_{CD}$, where X_{AB} and X_{CD} are fresh extension variables such that $X_{AB} = A \cdot B$, and $X_{CD} = C \cdot D$. Thus, with the help of extension axioms, we can implicitly use monomials of larger degree than allowed. If we restrict ourselves to refutations of polynomial size, then we can think of this proof system as a variant of the degree-3 polynomial calculus where the degree bound can be violated a limited number of times.

10.2 Expressing graph isomorphism as a polynomial equation system

Let G and H be fixed graphs, potentially with a colouring of the vertices or with multiple edge relations. We consider the following polynomial axiom system $P_{\text{iso}}(G, H)$ that expresses the existence of a (colour-preserving) isomorphism between G and H. A refutation of $P_{\text{iso}}(G, H)$ in any variant of the polynomial calculus then witnesses that G and H are non-isomorphic. The system $P_{\text{iso}}(G, H)$ is similar as in [17]. As shown there, the particular definition of $P_{\text{iso}}(G, H)$ is not decisive: Certain alternative natural formulations of graph isomorphism as polynomial equation systems can be derived from $P_{\text{iso}}(G, H)$ in the polynomial calculus.

Definition 10.2.1 ($P_{iso}(G,H)$, [17])**.** *Let G and H be two graphs (potentially vertex-coloured). Let $\sim \subseteq V(G) \times V(H)$ be the relation "vertex $v \in V(G)$ and $w \in V(H)$ have the same colour".*
The system $P_{iso}(G,H)$ consists of the following polynomials in the variables $\{X_{vw} \mid v \in V(G), w \in V(H), v \sim w\}$.

$$\sum_{\substack{v \in V(G) \\ v \sim w}} X_{vw} - 1 \qquad \textit{for all } w \in V(H) \tag{10.1}$$

$$\sum_{\substack{w \in V(H) \\ v \sim w}} X_{vw} - 1 \qquad \textit{for all } v \in V(G) \tag{10.2}$$

$$X_{vw}X_{v'w'} \qquad \begin{array}{l} \textit{for all } v, v' \in V(G), w, w' \in V(H) \\ \textit{with } v \sim w \textit{ and } v' \sim w' \\ \textit{such that } \{(v,w),(v',w')\} \textit{ is not} \\ \textit{a local isomorphism.} \end{array} \tag{10.3}$$

The intended meaning of the variable X_{vw} being set to one is "v is mapped to w". When we say that a certain variant of the polynomial calculus *distinguishes* two graphs G, H, we mean that the polynomial equation system $P_{\text{iso}}(G,H)$ has a refutation in that proof system. The main result from [17] links graph distinguishability in this sense to graph distinguishability by the k-dimensional Weisfeiler Leman algorithm.

Theorem 10.2.2 (Theorem 4.4 in [17])**.** *Let $k \in \mathbb{N}$ and let G and H be graphs. The axiom system $P_{iso}(G,H)$ has a refutation in the degree-k monomial calculus iff the $(k-1)$-dimensional Weisfeiler Leman algorithm distinguishes G and H.*

In [17], this is not stated for vertex- or edge-coloured graphs, but it can be checked that the proof still goes through in these cases. For our result, we need some of the technical ingredients from the proof of Theorem 10.2.2: What is shown in [17] is that Spoiler's winning positions in the bijective k-pebble game on G and H are derivable in MC_k from $P_{\text{iso}}(G,H)$.
A position in the game is a set of pebble pairs $\pi \subseteq V(G) \times V(H)$ of size $|\pi| \leq k$. The position π corresponds to a monomial in the variables from $P_{\text{iso}}(G,H)$. We denote this monomial as $X_\pi := \prod_{(v,w)\in\pi} X_{vw}$ (so the X_{vw} are the variables, whereas X_π is shorthand for a *product* of variables). We will use the following central technical result as a blackbox:

Lemma 10.2.3 (Lemma 4.2 in [17])**.** *Let $k \geq 2$ and G, H be graphs (such that for every vertex-colour Q, there are exactly as many vertices of colour Q in G as in H). If Spoiler has a winning strategy for the bijective k-pebble game on G, H with initial position π, then there is an MC_k-derivation of the monomial X_π from $P_{iso}(G,H)$.*

This lemma accounts for one direction of Theorem 10.2.2 because if $G \not\equiv_{\mathcal{C}^k} H$, then Spoiler wins the bijective k-pebble game from the initial position $\emptyset$, and we have $X_\emptyset = 1$.

10.3 Separating the extended polynomial calculus from its non-extended version

Intuitively speaking, adding extension axioms to a proof system should increase the power of the formalism since proofs can often be represented more concisely with extension variables. For *Frege* proof systems, it is conjectured that their respective *extended* versions cannot be polynomially simulated by non-extended Frege [28]. For the bounded-degree polyomial calculus (which is not a Frege system, though), we can confirm this conjecture using our newly established connection to finite model theory: Theorem 10.0.4 says that EPC_3, even with the limited types of extension axioms that we use here, is stronger than PC_k, for every k because there are families of graphs that cannot be distinguished in PC_k, but in EPC_3.

Proof of Theorem 10.0.4: According to Theorem 6.2 in [17], for every $n \in \mathbb{N}$, there exist pairs $G_n, \widetilde{G}_n$ of non-isomorphic CFI-graphs of size $\mathcal{O}(n)$ such that $P_{\text{iso}}(G_n, \widetilde{G}_n)$ has no degree-n polynomial calculus refutation (over $\mathbb{Q}$). A closer examination of the construction in [17] reveals that the axioms (10.1) and (10.2) in $P_{\text{iso}}(G_n, \widetilde{G}_n)$ are for coloured versions of the respective CFI-graphs, i.e. each vertex-gadget and each edge-gadget of G_n and $\widetilde{G}_n$, respectively, forms a distinct colour class and the axioms restrict possible isomorphisms to colour-preserving ones. Therefore, we have $P_{\text{iso}}(G_n, \widetilde{G}_n) = P_{\text{iso}}((G_n, \preceq), (\widetilde{G}_n, \preceq))$ for any preorder $\preceq$ on $V(G_n)$ that is obtained from a linear order on the respective base graph. Note that our definition of P_{iso} also works for graphs with multiple edge relations, and we can simply view the binary relation $\preceq$ as another type of edge relation. Now the system $P_{\text{iso}}((G_n, \preceq), (\widetilde{G}_n, \preceq))$ does have a polynomial-size refutation in EPC_3, for any choice of the ordering, because CFI-graphs over linearly ordered base graphs can be distinguished in CPT (see Theorem 6.1.1) and thus, a refutation exists by our Theorem 10.0.1.

10.4 Deep Weisfeiler Leman

Deep Weisfeiler Leman (DWL) is a model of choiceless computation equivalent to CPT. It separates the algorithmic aspects of a computation from the choiceless access to the input structure. A DWL-algorithm is a deterministic Turing machine operating on what is called the *algebraic sketch* of the input structure. This algebraic sketch contains – roughly speaking – the kind of information that can be computed with the 2-dimensional Weisfeiler Leman algorithm. In other words: Information about the $\mathcal{C}^3$-types of pairs that are realised in the structure, and how they "intersect". Importantly, the algebraic sketch of a structure is isomorphism-invariant, which ensures that the DWL-algorithm is choiceless. In addition to reading the algebraic sketch of the input structure, the machine may also add new elements to the input structure in a way that respects $\mathcal{C}^3$-types. This corresponds to the creation of higher-order objects in Choiceless Polynomial Time.
Before we can introduce the Deep Weisfeiler Leman framework in detail, we have to say precisely what the algebraic sketch of a structure is. It is a representation of its *coarsest coherent configuration* (or "coherent colouring"), that is defined below. The coarsest

coherent configuration of a structure is also known as its stable 2-dimensional Weisfeiler Leman colouring (equivalent to the partition of all pairs into their $\mathcal{C}^3$-types). This is where the name "Deep Weisfeiler Leman" comes from; in this thesis, however, we will not deal with the Weisfeiler Leman algorithm itself. It suffices to know that this method computes in polynomial time a canonical coarsest coherent configuration of any given graph structure. For details on the Weisfeiler Leman graph isomorphism test and its power, see e.g. [78].
The idea of Weisfeiler Leman is inspired by a similar computation model capturing the logic FPC, which has been outlined by Martin Otto [86]. In his framework, there is also a Turing machine that "communicates" with the input structure in an isomorphism-invariant way.

10.4.1 Coherent configurations and algebraic sketches

Definition 10.4.1 (Notions concerning binary relations, [63]).

- *The* domain *of a relation R is the set* $\mathrm{dom}(R) := \{u \mid \exists v : (u,v) \in R\}$.
- *The* codomain *of a relation R is the set* $\mathrm{codom}(R) := \{v \mid \exists u : (u,v) \in R\}$.
- *The* converse *of a relation R is the relation* $R^{-1} := \{(v,u) \mid (u,v) \in R\}$.
- *For a set V, the* diagonal *of V is the relation* $\mathrm{diag}(V) := \{(v,v) \mid v \in V\}$. *For a relation R we let $R^{\mathrm{diag}} := R \cap \mathrm{diag}(\mathrm{dom}(R))$ be the diagonal elements in R. We call R a* diagonal relation *if $R = R^{\mathrm{diag}} = \mathrm{diag}(\mathrm{dom}(R))$.*
- *The* strongly connected components *of a relation R are defined in the usual way as inclusionwise maximal sets $S \subseteq \mathrm{dom}(R) \cap \mathrm{codom}(R)$ such that for all $u,v \in S$ there is an R-path of length at least 1 from u to v. (In particular, a singleton set $\{u\}$ can be a strongly connected component only if $(u,u) \in R$.) We write $\mathrm{scc}(R)$ to denote the set of strongly connected components of R. Moreover, we let $R^{\mathrm{scc}} := \bigcup_{S \in \mathrm{scc}(R)} S^2$ be the relation describing whether two elements are in the same strongly connected component.*

Definition 10.4.2 (Coherent configurations, [63]). *Let σ be a vocabulary. A* coherent σ-configuration *C is a σ-structure C with the following properties.*

- *$\{R(C) \mid R \in \sigma\}$ is a partition of $V(C)^2$.*
- *For each $R \in \sigma$ the relation $R(C)$ is either a subset of or disjoint from the diagonal* $\mathrm{diag}(V(C))$.
- *For each $R \in \sigma$ there is an $R^{-1} \in \sigma$ such that $R^{-1}(C) = (R(C))^{-1}$.*
- *For all $R_1, R_2, R_3 \in \sigma$ there is a number $q = q(R_1,R_2,R_3) \in \mathbb{N}$ such that for all $(u,v) \in R_1(C)$ there are exactly q elements $w \in V(C)$ such that $(u,w) \in R_2(C)$ and $(w,v) \in R_3(C)$.*

The numbers $q(R_1, R_2, R_3)$ *are called the* intersection numbers *of* C *and the function* $q : \sigma^3 \to \mathbb{N}$ *is called the* intersection function.

A coherent σ-configuration C is at least as *fine* as, or *refines*, a τ-structure A (we write $C \sqsubseteq A$) if $V(A) = V(C)$, and for each $R \in \sigma$ and each $E \in \tau$ it holds that $R(C) \subseteq E(A)$ or that $R(C) \subseteq A^2 \setminus E(A)$. We say that a coherent configuration C is a *coarsest coherent configuration refining* a structure A if $C \sqsubseteq A$ and $C' \sqsubseteq C$ for every coherent configuration C' satisfying $C' \sqsubseteq A$. If for two configurations C, C' it holds $C \sqsubseteq C'$ and $C' \sqsubseteq C$, we write $C \equiv C'$. In this case, C and C' are equal up to a renaming of relation symbols.
In the following, we will usually write τ for the vocabulary of a given structure and σ for the vocabulary of the corresponding coherent configuration, without further specifying σ.

Every binary structure A has a coarsest coherent configuration, which can be computed in time $\mathcal{O}(n^3 \log n)$ with the 2-dimensional Weisfeiler Leman algorithm (Theorem 2.1 in [63]). This configuration is unique up to the renaming of relation symbols. We write $C(A)$ for the coarsest coherent configuration of A with canonical names of the relation symbols, as for example produced by a fixed implementation of 2-WL. We call the relation symbols in σ *colours* to distinguish them from the relation symbols in τ. In the following, we often identify the symbols in τ and σ with binary strings, because this is how they are represented in a Turing machine.
The *algebraic sketch* of a structure contains information about the colours appearing in its coarsest coherent configuration, which relations of the structure they refine, and the intersection function q. Formally, the algebraic sketch of a structure A is the tuple

$$D(A) = (\tau, \sigma, \subseteq_{\sigma,\tau}, q).$$

The relation $\subseteq_{\sigma,\tau}$ relates the colours in σ with the relations in τ they refine: $\subseteq_{\sigma,\tau} := \{(R, E) \in \sigma \times \tau \mid R(C(A)) \subseteq E(A)\}$.
In order to feed $D(A)$ to a Turing machine, we have to agree on some encoding in binary. If the binary string encodings of the relation symbols are fixed, then there is a canonical encoding of $D(A)$, based on the lexicographic ordering of the relation names and ordering of the intersection numbers $q(R_1, R_2, R_3)$. The string that encodes $D(A)$ is the initial tape content in a DWL-computation on the structure A.

10.4.2 The Deep Weisfeiler Leman computation model

A DWL-algorithm is a two-tape Turing machine M with an additional storage device that the authors of [63] have named "the *cloud*". It contains a coherently coloured structure $(A, C(A))$ to which the machine only has limited access. The storage that the machine itself can use is a work tape and an *interaction tape*, which allows for interaction with the cloud.

The input of a DWL-program M is a binary τ-structure A. Initially, the cloud contains the coherently coloured structure $(A, C(A))$, and on the interaction tape the algebraic sketch $D(A)$ is written (canonically encoded as a binary string). The work tape is empty. The Turing machine works as a standard Turing machine with three special transitions that can modify the structure in the cloud. To execute these, the machine writes a binary string $s \in \{0,1\}^*$ on the interaction tape and enters one of the four distinguished states $q_{\texttt{addPair}}, q_{\texttt{contract}}, q_{\texttt{create}}, q_{\texttt{forget}}$. The string s must be the name of some relation symbol $X \in \tau \cup \sigma$ or encode a set of colours $\pi \subseteq \sigma$.

For example, if $X = R$ for some colour $R \in \sigma$, then the operation `addPair`(R) adds a new vertex for every pair (u,v) of vertices with colour R, and connects the new pair-vertex with the elements u and v. This can be thought of as the creation of ordered pairs $(u,v) = \{u, \{v\}\}$ in CPT. The point of executing this operation is to gain more refined information about the structure in the cloud: The algebraic sketch $D(A)$ that the Turing machine knows initially is based on the 2-dimensional Weisfeiler Leman colouring of A. Now suppose we add a new vertex (u,v) for every pair $(u,v) \in V(A)^2$. It can be shown that the 2-WL-colouring of the new structure A' is equivalent to the 4-WL-colouring of the original structure A: For any four vertices $u, v, u', v' \in V(A)$, the 2-WL-colour of the pair $((u,v),(u',v')) \in V(A')^2$ carries as much information as the 4-WL-colour of the tuple $(u,v,u',v') \in V(A)^4$ (see also [86]). So in a sense, DWL allows to "locally" simulate higher-dimensional Weisfeiler Leman by introducing new pair-vertices for some of the colours in σ.

The second important operation is `contract`(R). It contracts all SCCs of the colour R (see Definition 10.4.1) into single vertices. In CPT, this corresponds to the creation of an unordered set for each SCC. This contraction does not lead to a finer and more informative coherent configuration but it can be useful in order to keep the structure in the cloud small. According to personal communication with Daniel Wiebking, DWL without the contraction operation is equivalent to the tuple-based fragment of CPT, which is $\approx$-free PIL; this is strictly weaker than CPT itself (see Section 3.3).

The operations `create` and `forget` add or remove relations from the structure in the cloud. They are convenient for programming in DWL but as we explain in the next section, they can be dispensed with without loss of expressive power. Formally, the effects of the four DWL-operations are as follows:

- `addPair`(X): If $X = E \in \tau$ is a relation symbol, let $P := E(A)$; else, if $X = R \in \sigma$ is a colour, then let $P := R(C(A))$. The machine adds a fresh vertex for each pair in P to A, i.e. it updates $V(A)$ to $V(A) \uplus P$. These pairs are then connected with their elements in A. Therefore, τ is updated to $\tau \cup \{E_{\text{left}}, E_{\text{right}}\} \uplus \{D_X\}$. The new relation D_X identifies the newly added vertices, i.e. $D_X(A) := \text{diag}(P)$. The symbol D_X is chosen as the lexicographically smallest binary string that is not yet used as a relation symbol.
  Furthermore, $E_{\text{left}}(A)$ is updated to $E_{\text{left}}(A) \cup \{(u,(u,v)) \in V(A)^2 \mid (u,v) \in P\}$, and $E_{\text{right}}(A)$ is set to $E_{\text{right}}(A) \cup \{(v,(u,v)) \in V(A)^2 \mid (u,v) \in P\}$ (where initially, $E_{\text{left}}(A) = E_{\text{right}}(A) = \emptyset$).

- `contract`(X): If $X = E \in \tau$, then let $\mathcal{S} := \mathrm{scc}(E(A))$; if $X = R \in \sigma$, let $\mathcal{S} = \mathrm{scc}(R(C(A)))$, i.e. $\mathcal{S}$ is the set of strongly connected components of the relation named X. Let $U := V(A) \setminus \bigcup \mathcal{S}$ be the set of vertices that are not in one of the strongly connected components. The components in $\mathcal{S}$ are contracted. That means we update $V(A)$ to $U \uplus \mathcal{S}$, and τ to $\tau \uplus \{D_X\}$. Again, $D_X(A) := \mathrm{diag}(\mathcal{S})$. For each relation $E \in \tau$, we update $E(A)$ to $(E(A) \cap U^2) \cup \{(u, S) \mid \exists v \in S \in \mathcal{S} : (u, v) \in E(A)\} \cup \{(S, v) \mid \exists u \in S \in \mathcal{S} : (u, v) \in E(A)\} \cup \{(S_1, S_2) \mid \exists u \in S_1 \in \mathcal{S}, \exists v \in S_2 \in \mathcal{S} : (u, v) \in E(A)\}$.
- `create`(π): In this case, the string on the interaction tape must be the encoding of a subset $\pi \subseteq \sigma$ of colours. Let E_π be the lexicographically first string that is not already in τ, and update τ to $\tau \uplus \{E_\pi\}$ where $E_\pi(A) := \bigcup_{R \in \pi} R(C(A))$.
- `forget`(X): In this case, X must be some relation $E \in \tau$. The machine updates τ to $\tau \setminus \{E\}$.

Each of these special transitions modifies the structure A in the cloud in an isomorphism-invariant way. After that, the cloud storage device computes $C(A)$ (for example, with the 2-WL algorithm) and stores the coherently coloured structure $(A, C(A))$. The algebraic sketch $D(A)$ of the new structure is written on the interaction tape.
A DWL-algorithm decides a class $\mathcal{K}$ of τ-structures in the usual sense, i.e. the program halts with output 1 on input A if $A \in \mathcal{K}$, and else, it halts with output 0. We say that a DWL-algorithm *runs in polynomial time* if the number of computation steps of the Turing machine and the size of the structure in the cloud is bounded by a polynomial in the size of the input structure.

10.4.3 Simplifications of DWL

The "full" DWL computation model described above is inconvenient for the translation into the extended polynomial calculus. Therefore, we define a simplified version, which has the same expressive power as DWL. A first simplification that is also introduced in [63] is called *pure* DWL. A DWL-algorithm is *pure* if it executes `addPair`(R) and `contract`(R) only for colours $R \in \sigma$ and not for relation symbols $E \in \tau$. Theorem 7 in [63] states that any polynomial time DWL-algorithm can be simulated by a pure one in polynomial time. We need a further restriction here. The following definition is new and not from [63].

Definition 10.4.3 (DWL with reduced instruction set). *A DWL-algorithm has* reduced instruction set *if it is pure and never executes* `create` *or* `forget` *on any input.*

For the purposes of distinguishing binary structures (see next section), we may always assume that DWL-algorithms have a reduced instruction set, as the next lemma shows. Namely, for a fixed input structure, any DWL-algorithm can be polynomially simulated by one with reduced instruction set, in the sense that the coarsest coherent configuration of the produced structure in the cloud can only get finer in the simulation. In other words, the graph distinguishing power of our reduced DWL is not less than that of original DWL.

Lemma 10.4.4. *Let A be a τ-structure and let M be a DWL-algorithm that terminates on input A with the structure B in the cloud. There exists a DWL-algorithm M'* with reduced instruction set *which terminates on input A with a structure B' in the cloud such that $C(B') \sqsubseteq C(B)$. Moreover, the time and space consumption of M' on input A is polynomially bounded in the time and space consumption of M on A.*

Proof sketch. Let t be the number of cloud-interaction-operations that M performs on input A and let $(\mathrm{op}_i(s_i))_{1 \le i \le t}$ be this sequence of operations, with $\mathrm{op}_i \in \{\texttt{addPair}, \texttt{contract}, \texttt{create}, \texttt{forget}\}$. Since Lemma 10.4.4 only claims the existence of a simulation for the fixed input A, we can design M' such that it executes a predefined hardcoded sequence of cloud-interaction-operations. We now explain what this instruction sequence looks like. As a first step, we remove all occurrences of `forget` from $(\mathrm{op}_i(s_i))_{1 \le i \le t}$. When the resulting operation-sequence is applied to A, then after the first removed `forget`-operation, the structure in the cloud may have a finer coarsest coherent configuration than in the original run of M on A at that point. This is simply because when the `forget`-operation is no longer executed, then the structure in the cloud has more relations, so its 2-dimensional Weisfeiler Leman colouring may be finer. Hence, whenever `addPair`(R) is executed after a removed `forget`-operation, then we may have to replace `addPair`(R) with some sequence `addPair`$(R_1), ...,$ `addPair`(R_m), where the R_i are the colours whose union is the relation R in the original run of M on A. To simulate the effect of `contract`(R) for a colour R that is refined in our new run, we use `create` to introduce a new relation E_R that is interpreted as the union over all colours appearing between vertices that are in the same R-SCC (such a set of colours exists by Lemma 10.4.15). After that, we call `contract`(E_R) to get the same effect as with the call of `contract`(R) in the original run. In this way, we can remove all occurrences of `forget`-operations and repair subsequent calls of `addPair` and `contract` such that they are applied to exactly the same pairs and SCCs as in the original run. This whole procedure is very similar to the one described in the proof of Theorem 7 in [63], so we refer there for the technical details. Let $(\mathrm{op}'_i(s_i))_{1 \le i \le t'}$ be the operation-sequence that we have constructed now. As in the proof of Theorem 7 in [63], its length t' can be polynomially bounded in terms of $|A|$ and the original length t because whenever a `addPair`(R) is replaced by `addPair`$(R_1), ...,$ `addPair`(R_m), then the number m of colours is polynomially bounded in $|A|$.

Now the sequence $(\mathrm{op}'_i(s_i))_{1 \le i \le t'}$ contains no more `forget`-operations. Before we also remove the `create`-operations, we invoke Theorem 7 from [63] to turn $(\mathrm{op}'_i(s_i))_{1 \le i \le t'}$ into a sequence of pure operations, which again increases the length and the required cloud space only polynomially. One can check that this Theorem 7 does not re-introduce `forget`-operations (more precisely, its proof can be implemented in such a way).
After we have turned the instruction-sequence into a pure one, we can simply remove all occurrences of `create`. This is not harmful because it can be shown that the coarsest coherent configurations are equally fine as before after all `create`-operations are removed. This is because the relations that are created by `create` are always unions of colours of the 2-WL colouring, and adding such relations to a structure can never lead to a finer

2-WL colouring. Moreover, since our instruction sequence was pure, the relations created by `create` never appear as arguments in any call of `addPair` or `contract`. Thus, we have to repair nothing after removing all create-operations.
In total, we have constructed a sequence of pure instructions in $\{\texttt{addPair}, \texttt{contract}\}$ that is at most polynomially longer than the run of M on A, and uses at most polynomially more space in the cloud. We let M' be the DWL-algorithm that simply executes this instruction sequence (it is hard-coded and the behaviour of M' does not actually depend on the input). The final structure B' in the run of M' on A is such that $C(B') \sqsubseteq C(B)$. The coarsest coherent configuration may be finer than $C(B)$ because in the run of M on A, relations may have been deleted with `forget`, which are now present in B'. □

10.4.4 Distinguishing graphs in Deep Weisfeiler Leman

In [63], the authors say that a DWL-algorithm *decides isomorphism* on a structure class $\mathcal{K}$ if it gets as input the disjoint union $A := G \uplus H$ of two connected binary structures and correctly decides whether $G \cong H$. A crucial technical result in [63] shows that one can always assume that at any stage of the computation, the structure in the cloud is the disjoint union of two connected structures: It is never necessary for the algorithm to produce connections between the two components, i.e. `addPair` and `contract` are only executed for colours R with $R(C(A)) \subseteq V(G)^2 \cup V(H)^2$. A DWL-algorithm that maintains this invariant is called *normalised* in [63].

Definition 10.4.5 (Distinguishing structures in DWL)**.** *The computation model DWL* distinguishes *all structures in a class $\mathcal{K}$ (of connected τ-structures) in* polynomial time *if: There is a polynomial $p(n)$ such that for any two non-isomorphic structures $G, H \in \mathcal{K}$, there exists a* normalised *DWL-algorithm M which, given $G \uplus H$ as input, terminates with a structure $G' \uplus H'$ in the cloud such that $D(G') \neq D(H')$, and takes time and space at most $p(|G| + |H|)$.*

This is simply the DWL-version of Definition 10.0.2 for distinguishing structures in CPT. The main difference to the CPT-setting is that here, the constant bound on the number of variables is already implicit in the definition of DWL (because DWL only accesses a structure via its coherent configuration, and two structures with the same configuration are $\mathcal{C}^3$-equivalent).

Let us elaborate on what is meant precisely by $D(G') \neq D(H')$. Whenever $A = A_1 \uplus A_2$ is a τ-structure consisting of two separate connected components, then we write $D(A)[A_1]$ and $D(A)[A_2]$ for the restrictions of the algebraic sketch $D(A)$ to the respective substructures. Formally: For $i \in \{1, 2\}$, let

$$D(A)[A_i] := (\tau, \sigma_i, \subseteq_{\sigma_i,\tau}, q_i),$$

where $\sigma_i := \{R \in \sigma \mid R(C(A)) \cap V(A_i)^2 \neq \emptyset\}$ and q_i is the restriction of q to domain σ_i^3. The function q_i is indeed a well-defined intersection function:

Lemma 10.4.6. *Let $A = A_1 \uplus A_2$ be the disjoint union of two connected τ-structures. For $i \in \{1, 2\}$, $D(A)[A_i]$ is an algebraic sketch and equal to $D(A_i)$, up to a renaming of the colours in σ_i.*

Proof. By Lemma 8 in [63], the colours of the coarsest coherent σ-configuration $C(A)$ can be partitioned into $\sigma_1, \sigma_2, \sigma_{\text{cross}}$, where $\sigma_i = \{R \in \sigma \mid R(C(A)) \cap V(A_i)^2 \neq \emptyset\}$, and for every $R \in \sigma_{\text{cross}}$, $R(C(A)) \subseteq (V(A_1) \times V(A_2)) \cup (V(A_2) \times V(A_1))$. Furthermore, also by Lemma 8 in [63], $C(A)[V(A_1)]$, i.e. the substructure of the configuration with universe $V(A_1)$, is equivalent to the coarsest coherent configuration $C(A_1)$, and similarly, $C(A)[V(A_2)] \equiv C(A_2)$. We can conclude that for $i \in \{1, 2\}$, q_i as defined above is indeed equivalent to the intersection function of $C(A_i)$ (up to a renaming of the colours) because for any $R_1, R_2, R_3 \in \sigma_i$ we know that in $C(A)$, only pairs in $V(A_1)^2 \cup V(A_2)^2$ can have these colours. Let q be the intersection function of $C(A)$. Then $q(R_1, R_2, R_3)$ is the same value as that of the intersection function of $C(A_i)$ for the corresponding three colours in $C(A_i)$. □

Thus, when we write $D(G') \neq D(H')$, we are formally referring to the respective restrictions of $D(G' \uplus H')$, but these are equivalent to $D(G')$ and $D(H')$, respectively. The algebraic sketches being distinct means that Spoiler has a winning strategy in the bijective 3-pebble game on the two structures. This is true because Spoiler can win from any position $\{(v, w), (v', w')\}$ where (v, v') and (w, w') receive distinct colours in the stable 2-WL-colouring. This connection between the 2-WL-colouring and the bijective pebble game is a standard result (see for example Theorem 2.2 in [78]). However, the setting in the literature always concerns the execution of the 2-dimensional Weisfeiler Leman algorithm separately on the two graphs G and H. Here, we consider the colouring of the *disjoint union* $G \uplus H$. It is perhaps not surprising that in this setting, the correspondence between Weisfeiler-Leman colourings and pebble games also exists, but we are not aware of a formal proof for this statement for $G \uplus H$ in the literature. Therefore, we explicitly prove the next lemma here, even though it is just a variation of a standard result and the proof is unfortunately quite long. Essentially it works as expected, with Spoiler's strategy being determined by the refinements made in the iterations of the 2-WL-algorithm. Additionally, we have to combine this with a technical insight from [63] for handling disjoint unions of connected binary structures.

Lemma 10.4.7. *Let G, H be two connected τ-structures and $A := G \uplus H$ with its coarsest coherent σ-configuration $C(A)$. Let $(v, v') \in V(G)^2, (w, w') \in V(H)^2$ such that there is no $R \in \sigma$ with $(v, v') \in R(C(A))$ and $(w, w') \in R(C(A))$.*
Then Spoiler has a winning strategy for the bijective 3-pebble game on G and H with initial position $\{(v, w), (v', w')\}$.

Proof. We show the statement by induction on the number of iterations that 2-dimensional Weisfeiler Leman needs to distinguish (v, v') and (w, w') in the structure A. Let us make precise how the 2-WL algorithm computes $C(A)$ by iteratively refining colourings of $V(A)^2$. In the initial colouring C_0, there is only one diagonal colour R_{diag} with $R_{\text{diag}}(C_0) := V(A)^2$. One colour R_{cross} is reserved for all crossing pairs, i.e.

$R_{\text{cross}}(C_0) = (V(G) \times V(H)) \cup (V(H) \times V(G))$. The remaining pairs are coloured according to their atomic types, so there is one colour for each atomic type of pairs in $V(G)^2 \cup V(H)^2$ that is realised in A. The atomic type of a pair (v, w) is the set of relations $R \in \tau$ such that $(v, w) \in R(A)$. Note that this initial colouring is not necessarily a coherent configuration: It satisfies all properties from Definition 10.4.2 except the last one about intersection numbers. In fact, this is the case for all colourings that are computed throughout the iteration, except for the final one, which is stable and a coarsest coherent configuration of A. To argue why this resulting configuration is indeed equivalent to $C(A)$, it is important that A is the disjoint union of two connected structures, and therefore, by Lemma 8 in [63], its crossing colours are distinct from its non-crossing colours. Therefore, the choice of our initial colouring C_0 will not lead to a stable colouring that is different from $C(A)$.

The colouring C_{i+1} is defined from the σ_i-colouring C_i as follows: Each colour class $R(C)$ is split along the intersection numbers of its pairs with other colour classes. That means $R(C)$ is split into the coarsest possible partition $\{P_1, ..., P_m\}$ such that for each P_i it holds: For all pairs $(u, v) \in P_i$, and all $S_1, S_2 \in \sigma_i$, the number of all $x \in V(A)$ such that $(u, x) \in S_1(C_i)$ and $(x, v) \in S_2(C_i)$ is the same (i.e. independent of the chosen pair in P_i).
Refining every colour in σ_i in this way yields the colouring C_{i+1}. We simply enumerate the colours in σ_{i+1} and call them $R_1, R_2, ...$ and so on, because we do not care about their actual names. This refinement process stops when the colouring is stable and cannot be refined further – the resulting colouring is equivalent to $C(A)$, the canonical coarsest coherent configuration, as it is the coarsest possible colouring that also satisfies the last condition of Definition 10.4.2.

We show the following four statements via induction on the number i of iterations of the refinement procedure:

(a) For every colour $R \in \sigma_i$, either $R(C_i) \subseteq (V(G) \times V(H)) \cup (V(H) \times V(G))$ or $R(C_i)$ is disjoint from $(V(G) \times V(H)) \cup (V(H) \times V(G))$. In the former case we say that R is *crossing.*

(b) For every colour $R \in \sigma_i$, there exist diagonal colours $D_1, D_2 \in \sigma_i$ such that for every pair $(u, v) \in R(C_i)$, it holds $(u, u) \in D_1(C_i)$ and $(v, v) \in D_2(C_i)$.

(c) Let $v, v' \in V(G), w, w' \in V(H)$ and let $D_1, D_2 \in \sigma_i$ be diagonal colours such that $(v, v), (w, w) \in D_1(C_i)$ and $(v', v'), (w', w') \in D_2(C_i)$. Then (v, w) and (v', w') have the same (crossing) colour.

(d) Let $(v, v') \in V(G)^2, (w, w') \in V(H)^2$ such that (v, v') and (w, w') do not have the same colour in C_i. Then Spoiler has a winning strategy for the bijective 3-pebble game on G and H with initial position $\{(v, w), (v', w')\}$.

Proof: For $i = 0$, (a), (b) and (c) are clear by definition of C_0, and (d) is also clear since in the initial colouring, distinct colours mean distinct atomic types. In that case,

$\{(v,w),(v',w')\}$ is not a local isomorphism and Spoiler wins immediately.

Now consider iteration $i+1$. Statement (a) follows from the inductive hypothesis because the colouring is refined in every step and thus, each crossing colour is always partitioned into crossing colours again, and the same holds for non-crossing colours.

Next, we show statement (b). Fix a colour $R \in \sigma_i$ and diagonal colours $D_1, D_2 \in \sigma_i$ such that for every pair $(u,v) \in R(C_i)$, it holds $(u,u) \in D_1(C_i)$ and $(v,v) \in D_2(C_i)$. We have to show: If any of the diagonal colours $D_1(C_i), D_2(C_i)$ are split, then the colour R is split in such a way that statement (b) still holds after iteration $i+1$. Assume w.l.o.g. that $D_1(C_i)$ is split: Let $(u,u),(u',u') \in D_1(C_i)$ and let v, v' be such that $(u,v),(u',v') \in R(C_i)$. Further, let $S_1, S_2 \in \sigma_i$ such that

$$\begin{aligned} |X| &:= |\{x \in V(A) \mid (u,x) \in S_1(C_i) \text{ and } (x,u) \in S_2(C_i)\}| \\ &\neq |\{x \in V(A) \mid (u',x) \in S_1(C_i) \text{ and } (x,u') \in S_2(C_i)\}| =: |X'|. \end{aligned}$$

Note that we have $S_2 = S_1^{-1}$. Now consider any two pairs $(u,v),(u',v') \in R(C_i)$. Partition X according to the colours of its elements paired with v, i.e. for any colour $T \in \sigma_i$, let $X_T := \{x \in X \mid (x,v) \in T\}$. Then the non-empty X_T form a partition of X. Similarly, define $X'_T := \{x \in X' \mid (x,v') \in T\}$. Since $|X| \neq |X'|$, there must exist a colour $T \in \sigma$ such that $|X_T| \neq |X'_T|$. Then for this colour, we have

$$\begin{aligned} &|\{x \in V(A) \mid (u,x) \in S_1(C_i) \text{ and } (x,v) \in T(C_i)\}| = |X_T| \\ \neq |X'_T| = &|\{x \in V(A) \mid (u',x) \in S_1(C_i) \text{ and } (x,v') \in T(C_i)\}|. \end{aligned}$$

Thus, the pairs $(u,v),(u',v')$ are in distinct colours after iteration $i+1$, as witnessed by the intersection numbers with the colours S and T. Hence, the invariant (b) still holds.

We know that statement (c) holds after iteration i. In order to show that it still holds after iteration $i+1$, we need to prove that whenever a crossing colour is refined, then at least one of its endpoint-colours is also refined: Fix a crossing colour $R \in \sigma_i$ and two pairs $(u_1,u_2),(u_1',u_2') \in R(C_i)$. By statement (b), we know that there are diagonal colours D_1, D_2 such that $(u_1,u_1),(u_1',u_1') \in D_1(C_i)$ and $(u_2,u_2),(u_2',u_2') \in D_2(C_i)$. Now suppose that in iteration $i+1$, the pairs (u_1,u_2) and (u_1',u_2') are separated. Our goal is to show that also u_1 and u_1' or u_2 and u_2' get distinct diagonal colours because the only way how (c) can fail to be true is if (u_1,u_2) and (u_1',u_2') get distinct colours but their respective first and second entries keep the same diagonal colour as before. So let $S_1, S_2 \in \sigma_i$ be colours that witness the separation of (u_1,u_2) and (u_1',u_2'):

$$\begin{aligned} |X| &:= |\{x \in V(A) \mid (u_1,x) \in S_1(C_i) \text{ and } (x,u_2) \in S_2(C_i)\}| \\ &\neq |\{x \in V(A) \mid (u_1',x) \in S_1(C_i) \text{ and } (x,u_2') \in S_2(C_i)\}| =: |X'|. \end{aligned}$$

Assume w.l.o.g. that $|X| > 0$. Exactly one of the colours S_1, S_2 is crossing, and the other is non-crossing. Assume w.l.o.g. that S_1 is non-crossing and S_2 is crossing. Then $X, X' \subseteq V(G)$.

Claim: $X = \{x \in V(A) \mid (u_1, x) \in S_1(C_i) \text{ and } (u_1, x) \in S_1^{-1}(C_i)\}$.
Proof of claim: The inclusion $\subseteq$ is clear. For the inclusion $\supseteq$, we have to show that for every $x \in V(A)$ with $(u_1, x) \in S_1(C_i)$ it holds $(x, u_2) \in S_2(C_i)$. This is true because: The diagonal colour of (x, x) is the same as that of every vertex in X, according to statement (b) with respect to S_1. Then statement (c) from the induction hypothesis implies that $(x, u_2) \in S_2(C_i)$. This proves the claim.

Similarly, we can prove $X' = \{x \in V(A) \mid (u_1', x) \in S_1(C_i) \text{ and } (u_1', x) \in S_1^{-1}(C_i)\}$. Hence we have

$$\begin{aligned}
&|\{x \in V(A) \mid (u_1, x) \in S_1(C_i) \text{ and } (x, u_1) \in S_1^{-1}(C_i)\}| \\
\neq &|\{x \in V(A) \mid (u_1', x) \in S_1(C_i) \text{ and } (x, u_1') \in S_1^{-1}(C_i)\}|.
\end{aligned}$$

Therefore, $(u_1, u_1), (u_1', u_1') \in D_1(C_i)$ will get distinct diagonal colours after iteration $i + 1$, as witnessed by the intersection numbers with S_1 and S_1^{-1}. If S_2 is non-crossing and S_1 is crossing, then it is the colour D_2 that is refined. This is what we wanted to show, so statement (c) is still true after iteration $i + 1$.

Finally, we can use this to prove statement (d). Assume that $(v, v') \in V(G)^2$ and $(w, w') \in V(H)^2$ have the same colour in C_i and get distinct colours in C_{i+1}. Then there exist colours $R_1, R_2 \in \sigma_i$ such that

$$\begin{aligned}
|X_{vv'}| :&= |\{x \in V(A) \mid (v, x) \in R_1(C_i) \text{ and } (x, v') \in R_2(C_i)\}| \\
&\neq |\{x \in V(A) \mid (w, x) \in R_1(C_i)) \text{ and } (x, w') \in R_2(C_i)\}| =: |X_{ww'}|.
\end{aligned}$$

We distinguish two cases:

Case 1: $|X_{vv'} \cap V(G)| \neq |X_{ww'} \cap V(H)|$. In this case, Spoiler can play as follows from position $\{(v, w), (v', w')\}$: Let $f : V(G) \longrightarrow V(H)$ be the bijection chosen by Duplicator. If $|X_{vv'} \cap V(G)| > |X_{ww'} \cap V(H)|$, then Spoiler chooses some $x \in X_{vv'} \cap V(G)$ such that $f(x) \in V(H) \setminus X_{ww'}$, and if $|X_{vv'} \cap V(G)| < |X_{ww'} \cap V(H)|$, then he chooses $x \in V(G) \setminus X_{vv'}$ such that $f(x) \in X_{ww'} \cap V(H)$. In both cases, the resulting position $\{(v, w), (v', w'), (x, f(x)))\}$ is a winning position for Spoiler by the inductive hypothesis because either in $\{(v, w), (x, f(x))\}$ or in $\{(v', w'), (x, f(x))\}$, the pebble pairs have distinct colours in C_i.

Case 2: $|X_{vv'} \cap V(G)| = |X_{ww'} \cap V(H)|$. In this case, we have $|X_{vv'} \cap V(H)| \neq |X_{ww'} \cap V(G)|$. W.l.o.g. assume that $|X_{vv'} \cap V(H)| > 0$. It can be seen that R_1 and R_2 are crossing colours. By statement (b), there is a diagonal colour $D \in \sigma_i$ such that all vertices in $(X_{vv'} \cap V(H)) \cup (X_{ww'} \cap V(G))$ have the diagonal colour D, because these are the second entries of pairs in $R_1(C_i)$. Statement (c) says even more: For every vertex $x \in V(H)$ with $(x, x) \in D$, we have $(v, x) \in R_1(C_i)$ and $(x, v') \in R_2(C_i)$, and for every $x \in V(G)$ with $(x, x) \in D$, we have $(w, x) \in R_1(C_i)$ and $(x, w) \in R_2(C_i)$. Summarising these considerations, we get $X_{vv'} \cap V(H) = \mathrm{dom}(D) \cap V(H)$ and $X_{ww'} \cap V(G) = \mathrm{dom}(D) \cap V(G)$.

Thus, we have $|\operatorname{dom}(D) \cap V(H)| \neq |\operatorname{dom}(D) \cap V(G)|$. Then Spoiler wins the game on G and H from any starting position: He can enforce a position $\{(y, z)\}$ with $(y, y) \in D(C_i)$ and $(z, z) \notin D(C_i)$ (or vice versa). From there, he wins by the induction hypothesis. This finishes the inductive proof of (a) – (d). Lemma 10.4.7 now follows from statement (d). □

Corollary 10.4.8. *Let G, H be connected τ-structures. It holds $D(G) \neq D(H)$ if and only if Spoiler has a winning strategy for the bijective 3-pebble game on G and H.*

Proof. To start with, we observe that if $D(G) = D(H)$, then Duplicator can always maintain the invariant that the pebbles induce a local isomorphism between $C(G)$ and $C(H)$, i.e. the colours are preserved. This is possible because the intersection functions of $C(G)$ and $C(H)$ are equal then.
Now assume $D(G) \neq D(H)$. Let $A := G \uplus H$. By Lemma 10.4.6, $D(G) = D(A)[G], D(H) = D(A)[H]$ (up to a renaming of the colours). Let σ_1, σ_2 be as in the proof of Lemma 10.4.6, i.e. these are the sets of colours of $C(A)$ that appear for pairs inside G and H, respectively. We now show: If $D(G) \neq D(H)$, then $\sigma_1 \neq \sigma_2$. Suppose for a contradiction that $\sigma_1 = \sigma_2$. Then it must also hold $q_1 = q_2$ for the intersection functions of $D(A)[G]$ and $D(A)[H]$, because q_1 and q_2 are the restrictions of the intersection function of $D(A)$ to the colours in σ_1, σ_2, respectively. But then, we have $D(A)[G] = D(A)[H]$, so by Lemma 10.4.6, it also holds $D(G) = D(H)$. Therefore, $D(G) \neq D(H)$ entails $\sigma_1 \neq \sigma_2$. Now let $R \in \sigma_1 \triangle \sigma_2$. Spoiler can pebble a pair that has colour R, and Duplicator cannot respond in a colour-preserving way. Spoiler wins from that position by Lemma 10.4.7. □

Remark. The above corollary also follows directly from the fact that if $D(G) \neq D(H)$, then the 2-dimensional Weisfeiler Leman algorithm distinguishes G and H. It is well-known (see e.g. [78]) that this is equivalent to the existence of a winning strategy for Spoiler in the bijective 3-pebble game on G and H.

Next, we show that if CPT distinguishes all structures in a class $\mathcal{K}$, then also DWL distinguishes all structures in $\mathcal{K}$ in polynomial time. This result is based on the simulation of CPT by DWL given in [63]. Hence, in our proof of Theorem 10.0.1, we can indeed start with the assumption that DWL polynomially distinguishes all graphs in $\mathcal{K}$.

Lemma 10.4.9. *Let $\mathcal{K}$ be a class of connected structures that are distinguished by* CPT *in the sense of Definition 10.0.2. Then DWL distinguishes all structures in $\mathcal{K}$ in polynomial time in the sense of Definition 10.4.5.*

Proof. Let $p(n)$ be the resource bound for the distinguishing CPT-programs for the class $\mathcal{K}$ that exists by Definition 10.0.2. Fix two τ-structures $G, H \in \mathcal{K}$ such that $G \not\cong H$. Let $\Pi \in \text{CPT}(p(n))$ be a distinguishing sentence. By Theorem 21 in [63], there exists a polynomial time DWL-algorithm M which simulates Π (and the polynomial resource bound of M depends only on $p(n)$, not on G and H). That means M w.l.o.g. accepts G and rejects H. Let $\rho_1 := (G = G_0, G_1, G_2, ..., G_{t_1})$ be the sequence of structures in the cloud that corresponds to the run of M on G, and let $\rho_2 := (H = H_0, H_1, H_2, ..., H_{t_2})$ be

the corresponding sequence for the run on H. Let $j \in \mathbb{N}$ be the smallest index such that $D(G_j) \neq D(H_j)$. This must exist because otherwise, the two runs would be identical, as the behaviour of M is only determined by the algebraic sketch of the structure in the cloud.
Our goal is to construct a normalised DWL-algorithm M' that produces on input $A := G \uplus H$ a structure $G' \uplus H'$ with $D(G') \neq D(H')$. This algorithm will simply consist in a hard-coded sequence of cloud-interaction-operations. These operations will be equivalent to the operations executed by M on G and H, respectively. We will make use of the fact that for any disjoint union of connected structures $A = A_1 \uplus A_2$, $C(A)[V(A_i)] \equiv C(A_i)$, for $i \in \{1, 2\}$ (Lemma 8 in [63]). In other words, the coarsest coherent configuration of the disjoint union, restricted to one of the substructures A_i, is equal to the coarsest coherent configuration of A_i by itself (up to a renaming of the colours). Therefore, M' can simulate the runs of M on G and H, respectively, simultaneously on $G \uplus H$: Let $\mathrm{op}_i(s_i)$ be the cloud-interaction-operation that M executes to get from G_i to G_{i+1} in the run ρ_1. We know that in ρ_2, the operation executed in this step is exactly the same (if $i < j$) because M is deterministic, and for $i < j$, we have $D(G_i) = D(H_i)$. We may assume that $\mathrm{op}_i \in \{\texttt{addPair}, \texttt{contract}\}$, and that s_i encodes a colour (and not a relation symbol in τ): If this does not hold, simply apply the proof of Lemma 10.4.4 to ρ_1 and ρ_2 (up to the point where they differ); this yields a reduced instruction sequence that still produces distinct algebraic sketches on the inputs G and H because the coarsest coherent configurations can only get finer than with the original instruction sequence.
We now wish to simulate the effect of $\mathrm{op}_i(s_i)$ on G_i and H_i in the run of M' on A. By induction, we can assume that the current structure in the cloud is $A_i := G_i \uplus H_i$. Let R be the colour that is encoded by s_i. Now there are two cases:
The first case is that there exists a colour R' such that $R'(C(A_i)) = R(C(G_i)) \cup R(C(H_i))$. Then we simulate $\mathrm{op}_i(s_i)$ by executing $\mathrm{op}_i(R')$.
The second case is that there are two distinct colours R'_1, R'_2 such that $R'_1(C(A_i)) = R(C(G_i)$ and $R'_2(C(A_i)) = R(C(H_i))$. Then we simulate $\mathrm{op}_i(s_i)$ by executing $\mathrm{op}_i(R'_1)$ and then $\mathrm{op}_i(R'_2)$.
There can be no more cases because the colourings $C(G_i)$ and $C(H_i)$ are equally fine as the respective substructures of $C(G_i \uplus H_i)$. Furthermore, it is important to note that in $C(A_i)$, every colour $R(C(A_i))$ either consists only of pairs inside the same connected component or of "crossing pairs" only (see Lemma 8 in [63]). Therefore, it cannot happen that the colour R above corresponds to some colour R' in $C(A_i)$ which contains undesired crossing pairs.
In both cases above, the new structure in the cloud after execution of the simulating operations is $G_{i+1} \uplus H_{i+1}$.
In this way, M' can simulate every step of the runs ρ_1 and ρ_2, such that M' on input A halts with the structure $G_j \uplus H_j$ in the cloud, and we have $D(G_j) \neq D(H_j)$. It is easy to see that M' is normalised (we never call a cloud-operation for a crossing colour) and runs in polynomial time and space (the latter holds because M is polynomially bounded). $\square$

10.4.5 Properties of coherent configurations

Here is a small collection of lemmas concerning coherent configurations. We will need them in our construction of the EPC_3-refutation in the next section. I would like to thank Daniel Wiebking for his extensive answers to all my questions regarding these properties and an anonymous referee of [88] for some simplifications of the proofs.

Lemma 10.4.10. *Let A be a τ-structure and $C(A)$ its coarsest coherent σ-colouring. Let $R \in \sigma$. There are diagonal colours $D_1, D_2 \in \sigma$ such that for all pairs $(u,v) \in R(C(A))$ we have $(u,u) \in D_1(C(A))$, and $(v,v) \in D_2(C(A))$.*

Proof. This is Corollary 2.1.7 in [31] (the term “fibers” there means the same as our “diagonal colours”). □

Corollary 10.4.11. *Let A be a τ-structure and $C(A)$ its coarsest coherent σ-colouring. Let $R \in \sigma$. If for any $(u,v) \in R(C(A))$, (u,u) or (v,v) is in some relation $E(A)$, for $E \in \tau$, then for all other $(u',v') \in R(C(A))$, it also holds $(u',u') \in E(A)$, or $(v',v') \in E(A)$, respectively.*

Proof. Follows from the previous lemma and the fact that the coarsest coherent configuration of A is a refinement of the relations of A. □

Lemma 10.4.12. *Let A be a τ-structure and let $C(A)$ be its coarsest coherent σ-configuration. Let $R \in \sigma$ and let $\mathcal{S} = \text{SCC}(R)$ be the set of R-SCCs in $C(A)$. There is a diagonal relation $P \in \sigma$ such that $\text{diag}(\bigcup \mathcal{S}) = P(C(A))$.*

Proof. First, we show that $\text{diag}(\bigcup \mathcal{S}) \subseteq P(C(A))$. Any vertex $v \in \bigcup \mathcal{S}$ has some outgoing R-neighbour w that is in the same SCC (possibly, $w = v$). That is, we have $(v,w) \in R(C(A))$. By Lemma 10.4.10, there are specific diagonal relations D_1, D_2 such that $(v,v) \in D_1(C(A)), (w,w) \in D_2(C(A))$, and all endpoints of R-edges have these diagonal colours. But since w is itself the left entry in some other R-edge (w,w'), we must have $D_1 = D_2 =: P$.
It remains to prove $P(C(A)) \subseteq \text{diag}(\bigcup \mathcal{S})$. For any $v \in \text{diag}(\bigcup \mathcal{S})$, there exists an R-path of length ≥ 1 from v to itself. We have already argued that $(v,v) \in P(C(A))$. It follows that any other vertex w with $(w,w) \in P(C(A))$ also has an R-path to itself and is thus in $\bigcup \mathcal{S}$. To see this, recall that $C(A)$ corresponds to the stable 2-WL-colouring [63], which in turn partitions A^2 into $\mathcal{C}^3$-types [78]. Hence, all vertices with the same diagonal colour satisfy exactly the same $\mathcal{C}^3$-formulas. The existence of an R-path from a vertex to itself (in the fixed structure A) is expressible in $\mathcal{C}^3$ using standard techniques: Namely, for every fixed number $d \leq |A|$, we can write a $\mathcal{C}^3$ formula $\varphi_d(x,y)$ that asserts the existence of a path of length d from x to y. Only 3 variables are needed because one can alternately requantify used variables (see e.g. Proposition 3.2 in [76]). In the formula, we have access to the relation R because it is itself $\mathcal{C}^3$-definable: Essentially, R is a $\mathcal{C}^3$-type of vertex-pairs in A, and it is known that on finite structures, such a type is definable with a single formula. □

Corollary 10.4.13. *Let A be a τ-structure such that for every $v \in V(A)$, (v, v) is in exactly one diagonal relation $P(A)$, for $P \in \tau$ (i.e. A is a graph with vertex colours). Let $C(A)$ be the coarsest coherent σ-configuration of A. Let $R \in \sigma$ and let $\mathcal{S} = \mathrm{SCC}(R)$. There is a colour (i.e. a diagonal relation) $P \in \tau$ such that for every SCC $S \in \mathcal{S}$, $\mathrm{diag}(S) \subseteq P(A)$.*

Proof. The coarsest coherent configuration $C(A)$ is a refinement of A. Therefore, the diagonal relations in $C(A)$ are subsets of the diagonal relations in A. Now the statement follows directly from Lemma 10.4.12. □

Lemma 10.4.14. *Let $A, C(A), R \in \sigma$, and $\mathcal{S}$ be as above. All R-SCCs in $\mathcal{S}$ are of equal size.*

Proof. For any number $k \in \mathbb{N}$, we can write a $\mathcal{C}^3$-formula $\varphi_k(x)$ asserting that the size of the R-SCC of x is exactly k. To do this, we can just use a counting quantifier and the fact that the existence of an R-path between two vertices (and back) is $\mathcal{C}^3$-definable (see proof of Lemma 10.4.12). Now since all vertices in R-SCCs have the same diagonal colour (Lemma 10.4.12), and colours coincide with $\mathcal{C}^3$-types, they all satisfy the same φ_k and thus, all SCCs have equal size. □

Lemma 10.4.15. *Let $A, C(A), R \in \sigma$ and $\mathcal{S}$ be as above. There is a collection of colours $R_1, ..., R_t \in \sigma$ such that $R^{\mathrm{scc}} = \bigcup_{S \in \mathcal{S}} S^2 = \bigcup_{i=1}^{t} R_i(C(A))$.*

Proof. We let $R_1, ..., R_t$ be the smallest collection of colours such that every $(u, v) \in \bigcup_{S \in \mathcal{S}} S^2$ occurs in one of them. To see that this has the desired property, let $(u, v) \in V(A)^2$ be such that u and v are not in the same SCC. Let $T \in \sigma$ be the colour such that $(u, v) \in T(C(A)))$. There is no R-path from u to v and back. As already argued in the proof of Lemma 10.4.12, this fact is expressible in $\mathcal{C}^3$. Since there does exist an R-path in both directions between any two vertices inside each SCC, and colours coincide with $\mathcal{C}^3$-types, T cannot be among the R_i. □

The next lemma tells us that the colour of a pair (v', z) between some vertex v' and any other vertex z inside a given SCC contains the information whether or not there exists an edge from v' into the SCC. In particular, the colour "between v' and the SCC" is independent of the choice of the vertex z in the SCC. This also explains why contracting SCCs is possible without loss of information.

Lemma 10.4.16. *Let $A, C(A), R \in \sigma$ and $\mathcal{S}$ be as above. Fix any relation symbol $E \in \tau$. Let $V, W \in \mathcal{S}$ and $v', w' \in V(A)$ be such that:*
There is a $v \in V$ such that $(v', v) \in E(A)$, and for all $w \in W$ it holds $(w', w) \notin E(A)$.
Let $z \in V$ be arbitrary and $T \in \sigma$ such that $(v', z) \in T(C(A))$. Then $T(C(A)) \cap (\{w'\} \times W) = \emptyset$.

Proof. There is a $\mathcal{C}^3$-formula $\varphi(x, y)$ that asserts: There exists some vertex y' in the same R-SCC as y such that $(x, y') \in E(A)$. This formula can be constructed as described in the proof of Lemma 10.4.12. Since $\varphi(x, y)$ is satisfied in A for $x \mapsto v'$ and $y \mapsto z$, for any $z \in V$, but not for $x \mapsto w'$ and $y \mapsto w$, for any $w \in W$, the lemma follows again from the fact that pairs with distinct $\mathcal{C}^3$-types receive distinct colours in $C(A)$. □

The next lemma is of a similar kind. It states that the colours of pairs between different SCCs contain the information whether or not there exist edges between the two SCCs.

Lemma 10.4.17. *Let $(A, C(A))$, $R \in \sigma$ and $\mathcal{S}$ be as above. Fix any relation symbol $E \in \tau$. Let $V, W, V', W' \in \mathcal{S}$ be such that: There exists $v' \in V', v \in V$ such that $(v', v) \in E(A)$, and for all $w' \in W'$, all $w \in W$, $(w', w) \notin E(A)$.*
Let $z \in V, z' \in V'$ be arbitrary, and let $T \in \sigma$ such that $(z', z) \in T(C(A))$. Then $T(C(A)) \cap (W' \times W) = \emptyset$.

Proof. Analogous to the proof of Lemma 10.4.16. Here, we use a $\mathcal{C}^3$-formula $\varphi(x, y)$ that asserts: There exists x' in the same SCC as x, and y' in the same SCC as y such that $(x', y') \in E(A)$. $\square$

10.5 Refuting graph isomorphism in the extended polynomial calculus

Let $\mathcal{K}$ be a class of connected binary structures such that CPT distinguishes all structures in $\mathcal{K}$. By Lemma 10.4.9, then also DWL distinguishes all structures in $\mathcal{K}$ in polynomial time. Now Theorem 10.0.1 follows from Lemma 10.5.2 below that establishes the link between DWL-distinguishability and the extended polynomial calculus. Before we can prove Lemma 10.5.2, we have to state the key technical result that it depends on:

Lemma 10.5.1. *Let G, H be two connected binary τ-structures, which are potentially vertex-coloured in such a way that for every vertex-colour Q, there are as many vertices with colour Q in G as in H. Let $\mathrm{op} \in \{\texttt{addPair}, \texttt{contract}\}$ and let $R \in \sigma$, where σ is the vocabulary of the coarsest coherent configuration $C := C(G \uplus H)$. Assume that $R(C) \subseteq V(G)^2 \cup V(H)^2$.*
Let $G' \uplus H'$ be the result of executing $\mathrm{op}(R)$ on $G \uplus H$.

Then the polynomial axiom system $P_{iso}(G', H')$ is derivable from $P_{iso}(G, H)$ in EPC_3, up to a renaming of variables. The number of extension variables used in the derivation is at most $|V(G')|^2$, and the derivation has polynomial size and uses only coefficients with polynomial bit-complexity. Moreover, for every extension axiom $\frac{}{X_f - f}$ used in the derivation, f is of the form $f = X \cdot Y$ or $f = \frac{1}{n} \cdot \left(\sum_{i=1}^{n^2} X_i \right)$.

The proof is quite lengthy and would interrupt the proof of Theorem 10.0.1 at this point; therefore, we first present the lemma and proof that explains how Theorem 10.0.1 follows from Lemma 10.5.1. Afterwards, we provide the actual polynomial calculus derivations whose existence is claimed in Lemma 10.5.1.

Lemma 10.5.2. *Let G, H be two connected binary τ-structures. Let $p(n)$ be a polynomial and M be a normalised DWL-algorithm which produces on input $G \uplus H$ a structure $G' \uplus H'$ with $D(G') \neq D(H')$, such that the length of the run and the size of the structure in the cloud is bounded by $p(|G| + |H|)$ at any time.*

Then the system $P_{iso}(G, H)$ has an EPC_3*-refutation that uses at most* $p(|G| + |H|)^3$ *many extension variables, has polynomial size and polynomial bit-complexity. Moreover, for every extension axiom* $\frac{}{X_f - f}$ *used in the derivation,* f *is of the form* $f = X \cdot Y$ *or* $f = \frac{1}{n} \cdot \left(\sum_{i=1}^{n^2} X_i \right)$.

Proof. Follows from Lemma 10.5.1 together with Corollary 10.4.8 and Lemma 10.2.3. In detail: Let $(\mathrm{op}_i(R_i))_{i \leq t}$ with $\mathrm{op}_i \in \{\mathtt{addPair}, \mathtt{contract}\}$ be the sequence of cloud-interaction-operations in the run of M on $G \uplus H$. This sequence of operations produces a sequence of structures $(G \uplus H, G_1 \uplus H_1, G_2 \uplus H_2, ..., G_t \uplus H_t)$ such that $D(G_t) \neq D(H_t)$. For each i, R_i is a colour in the coarsest coherent configuration of the current structure $A_i := G_i \uplus H_i$ in the cloud. Since M is normalised, $R(C(A_i)) \subseteq V(G_i)^2 \cup V(H_i)^2$. Thus, we can inductively apply Lemma 10.5.1 to derive in EPC_3 polynomial axiom systems $P_{\mathrm{iso}}(G_i, H_i)$ for every $i \in [t]$. The induction requires that for every vertex-colour (i.e. diagonal relation), the colour classes always have the same size in G_i and H_i (this is a prerequisite of Lemma 10.5.1). This is satisfied because $D(G_i) = D(H_i)$, for $i < t$. Since $D(G_t) \neq D(H_t)$, it follows from Corollary 10.4.8 and Lemma 10.2.3 that the 1-polynomial is derivable from $P_{\mathrm{iso}}(G_t, H_t)$ in the degree-3 monomial calculus, so in total, it is derivable from $P_{\mathrm{iso}}(G, H)$ in EPC_3. In order to apply Lemma 10.2.3 to G_t and H_t, we need to argue that the vertex-colour-classes are of equal size in both graphs, even though $D(G_t) \neq D(H_t)$. If step t is $\mathtt{addPair}(R)$, then we introduce equally many new pair-vertices in both graphs because $D(G_{t-1}) = D(H_{t-1})$, so the numbers of R-pairs are equal. If step t is $\mathtt{contract}(R)$, we also produce the same number of new vertices. Namely, the number of R-SCCs in $C(G_{t-1})$ and $C(H_{t-1})$ is equal, because by Lemma 10.4.14, all R-SCCs have equal size, and by Lemma 10.4.12, vertices in R-SCCs receive the same diagonal colour P distinct from all diagonal colours outside SCCs (and $|P(C(G_{t-1}))| = |P(C(H_{t-1}))|$).

Finally, we bound the number of extension variables used in the derivation of $P_{\mathrm{iso}}(G_t, H_t)$: As stated in Lemma 10.5.1, for every i, the derivation of $P_{\mathrm{iso}}(G_i, H_i)$ from $P_{\mathrm{iso}}(G_{i-1}, H_{i-1})$ uses at most $|V(G_i)|^2$ many new extension variables. Therefore, the total number of extension variables that are used in the derivation of $P_{\mathrm{iso}}(G_t, H_t)$ is at most $\sum_{i \in [t]} |V(G_i)|^2$. Since $t \leq p(|G| + |H|)$ and $|V(G_i)| \leq p(|G| + |H|)$, for every $i \in [t]$, this sum is at most $p(|G| + |H|)^3$. Similarly we can bound the size and bit-complexity of the derivation: Each time we invoke Lemma 10.5.1, we only incur a polynomial cost in size, and this happens polynomially many times. The occurring coefficients can be encoded with polynomially many bits as the lemma asserts. Also, Lemma 10.5.1 uses only extension axioms of the required form. Finally, we bound the number of extension variables used in the derivation of $P_{\mathrm{iso}}(G_t, H_t)$: As stated in Lemma 10.5.1, for every i, the derivation of $P_{\mathrm{iso}}(G_i, H_i)$ from $P_{\mathrm{iso}}(G_{i-1}, H_{i-1})$ uses at most $|V(G_i)|^2$ many new extension variables. Therefore, the total number of extension variables that are used in the derivation of $P_{\mathrm{iso}}(G_t, H_t)$ is at most $\sum_{i \in [t]} |V(G_i)|^2$. Since $t \leq p(|G| + |H|)$ and $|V(G_i)| \leq p(|G| + |H|)$, for every $i \in [t]$, this sum is at most $p(|G| + |H|)^3$. Similarly we can bound the size and bit-complexity of the derivation: Each time we invoke Lemma 10.5.1, we only incur a polynomial cost

in size, and this happens polynomially many times. The occurring coefficients can be encoded with polynomially many bits as the lemma asserts. Also, Lemma 10.5.1 uses only extension axioms of the required form. □

Proof of Lemma 10.5.1: First, we have to explain how the variables of the new system $P_{\text{iso}}(G', H')$ are encoded as polynomials in the old variables. Recall that the variable set of $P_{\text{iso}}(G, H)$ is

$$\mathcal{V}(G, H) := \{X_{vw} \mid v \in V(G), w \in V(H), v \text{ and } w \text{ have the same vertex-colour}\}.$$

The intended meaning of X_{vw} is "v is mapped to w". The graphs G', H' contain new vertices, which either represent contracted R-SCCs or pairs of colour R. The set of vertex-pairs (v, w) for which we need new variables is $\mathbf{NewPairs} := (V(G') \setminus V(G)) \times (V(H') \setminus V(H))$.
We would like to map each $(v, w) \in \mathbf{NewPairs}$ to a polynomial $f(vw)$ such that we can represent variables X_{vw} for $(v, w) \in \mathbf{NewPairs}$ as extension variables $X_{f(vw)}$, which we can introduce with the extension axiom $X_{f(vw)} = f(vw)$. If op $=$ `addPair` and v is a new pair-vertex, then we let $\text{pair}(v)$ be the vertex-pair that v corresponds to. If op $=$ `contract` and v is a new SCC-vertex, then we let $\text{scc}(v)$ denote the set of vertices in the SCC that is contracted into v. We define f as the following injective mapping $f : \mathbf{NewPairs} \to \mathbb{Q}[\mathcal{V}(G, H)]$.

$$f(vw) := \begin{cases} X_{v_1w_1}X_{v_2w_2} & \text{, if op} = \texttt{addPair} \text{ and} \\ & (v_1, v_2) = \text{pair}(v), (w_1, w_2) = \text{pair}(w). \\ \frac{1}{|\text{scc}(v)|} \cdot \sum\limits_{(v',w') \in \text{scc}(v) \times \text{scc}(w)} X_{v'w'} & \text{, if op} = \texttt{contract}. \end{cases}$$

Note that $f(vw)$ is indeed always a polynomial in variables $\mathcal{V}(G, H)$: To see this, we have to check that in the pair-case, the vertex-colours of v_1, w_1 and of v_2, w_2, respectively, are equal, and in the SCC-case, the vertex-colours of all elements of $\text{scc}(v)$ and $\text{scc}(w)$ are equal. In the pair-case, this follows from Corollary 10.4.11, and in the SCC-case from Corollary 10.4.13.

If v and w are newly introduced pair-vertices with $\text{pair}(v) = (v_1, v_2)$ and $\text{pair}(w) = (w_1, w_2)$, then the variable X_{vw} will be the extension variable for the monomial $X_{v_1w_1}X_{v_2w_2}$. This makes sense because if X_{vw} is set to 1, then the bijection encoded by the assignment maps v to w; but then it also has to map v_1 to w_1 and v_2 to w_2. Similarly, if v and w are new SCC-vertices, then any bijection that takes v to w must also map the elements of $\text{scc}(v)$ to the elements of $\text{scc}(w)$ in any possible way. This is reflected in our representation of X_{vw} as the "average" over all possible mappings from $\text{scc}(v)$ to $\text{scc}(w)$.

Here are the new polynomial axioms that we have to derive in order to go from $P_{\text{iso}}(G, H)$ to $P_{\text{iso}}(G', H')$:

$$\sum_{v \in V(G') \setminus V(G)} X_{f(vw)} - 1 \qquad \text{for all } w \in V(H') \setminus V(H). \tag{10.4}$$

$$\sum_{w \in V(H') \setminus V(H)} X_{f(vw)} - 1 \qquad \text{for all } v \in V(G') \setminus V(G). \tag{10.5}$$

$$X_{f(vw)} X_{v'w'} \qquad \begin{array}{l} \text{for all } v, v' \in V(G'), w, w' \in V(H') \\ \text{such that } (v, w) \in \mathbf{NewPairs} \\ \text{and } v', w' \in V(G) \cup V(H), v' \sim w' \\ \text{and } \{(v, w), (v', w')\} \text{ is not} \\ \text{a local isomorphism.} \end{array} \tag{10.6}$$

$$X_{f(vw)} X_{f(v'w')} \qquad \begin{array}{l} \text{for all } v, v' \in V(G'), w, w' \in V(H') \\ \text{such that } (v, w) \in \mathbf{NewPairs} \\ \text{and } (v', w') \in \mathbf{NewPairs} \\ \text{and } \{(v, w), (v', w')\} \text{ is not} \\ \text{a local isomorphism.} \end{array} \tag{10.7}$$

The relation $\sim$ is the same-colour-relation, as in Definition 10.2.1. Note that Axioms (10.4) and (10.5) only sum over vertices of the same colour as w and v, respectively (as Axioms (10.1) and (10.2) do), because $V(G') \setminus V(G)$ and $V(H') \setminus V(H)$ are the sets of newly added vertices. These vertices receive a new colour distinct from all other vertex colours in G and H (see definition of the DWL-operations in Section 10.4.2). The next step is to verify that $P_{\text{iso}}(G', H')$ is indeed derivable from $P_{\text{iso}}(G, H)$.

Derivation of Axioms (10.4) **and** (10.5)**:**
Fix $w \in V(H') \setminus V(H)$. We show how to derive Axiom (10.4) for w. Two cases have to be distinguished, namely whether op = `addPair` or op = `contract`.

Case 1: op = `addPair`**:** Let $(w_1, w_2) = \text{pair}(w)$. Using the multiplication rule and linear combinations, we derive from Axiom (10.1) for w_2 in $P_{\text{iso}}(G, H)$:

$$\left(\sum_{\substack{v_1 \in V(G), \\ v_1 \sim w_1}} X_{v_1 w_1} \right) \cdot \left(\sum_{\substack{v_2 \in V(G), \\ v_2 \sim w_2}} X_{v_2 w_2} - 1 \right)$$

$$= \sum_{\substack{v_1, v_2 \in V(G) \\ v_1 \sim w_1, \\ v_2 \sim w_2}} X_{v_1 w_1} X_{v_2 w_2} - \sum_{\substack{v_1 \in V(G), \\ v_1 \sim w_1}} X_{v_1 w_1}$$

Recall from the statement of Lemma 10.5.1 that $R \in \sigma$ is the colour such that $\mathrm{op}(R)$ is executed to obtain $G' \uplus H'$ from $G \uplus H$. Further, let $C = C(G \uplus H)$. Since $(w_1, w_2) \in R(C)$, we can use Lemma 10.4.7 and Lemma 10.2.3 to derive from $P_{\mathrm{iso}}(G,H)$ all monomials $X_{v_1 w_1} X_{v_2 w_2}$ where $(v_1, v_2) \notin R(C)$. Hence, we may cancel these monomials from the above sum with the linear combination rule. This yields:

$$\sum_{\substack{v_1, v_2 \in V(G) \\ (v_1, v_2) \in R(C)}} X_{v_1 w_1} X_{v_2 w_2} - \sum_{\substack{v_1 \in V(G) \\ v_1 \sim w_1}} X_{v_1 w_1}.$$

Here, we used that for all pairs $(v_1, v_2) \in V(G)^2 \cap R(C(A))$, it holds that $v_1 \sim w_1$ and $v_2 \sim w_2$. This follows from Corollary 10.4.11 and the fact that vertex-colours are represented by diagonal relations. Now we are almost done: We add Axiom (10.1) for w_1 to the above expression and replace each remaining monomial $X_{v_1 w_1} X_{v_2 w_2}$ with the new extension variable $X_{f(vw)}$, where $v \in V(G')$ is the respective new pair-vertex with $\mathrm{pair}(v) = (v_1, v_2)$. One can see that

$$V(G') \setminus V(G) = \{v \in V(G') \mid \mathrm{pair}(v) \in R(C) \cap V(G)^2\}.$$

Thus, we have indeed derived Axiom (10.4) for w.
Similarly, we get Axiom (10.5) for a vertex $v \in V(G') \setminus V(G)$ if we perform the same derivations from the Axioms (10.2) instead of (10.1).

Case 2: op $=$ `contract`: We derive Axiom (10.4) for a fixed vertex $w \in V(H') \setminus V(H)$. Now w is a vertex that represents a contracted R-SCC $\mathrm{scc}(w) \subseteq V(H)$.

For every vertex $w' \in \mathrm{scc}(w)$, we have Axiom (10.1) for w' in $P_{\mathrm{iso}}(G,H)$:

$$\sum_{\substack{v' \in V(G), \\ v' \sim w'}} X_{v'w'} - 1.$$

Now from this, we may cancel all $X_{v'w'}$ where (v', v') and (w', w') are in distinct diagonal relations in C. This is done again by deriving the respective variables $X_{v'w'}$ with Lemma 10.4.7 and Lemma 10.2.3. After that step, we have for each $w' \in \mathrm{scc}(w)$:

$$\sum_{\substack{v' \in V(G), \\ \text{there is } v \in V(G') \setminus V(G) \text{ with } v' \in \mathrm{scc}(v)}} X_{v'w'} - 1. \tag{$\star$}$$

This holds because the vertex $v' \in V(G)$ has the same diagonal colour as $w' \in \mathrm{scc}(w)$ in the coherent configuration C if and only if it is also contained in some R-SCC (Lemma 10.4.12).

Next, we use the variable introduction rule and introduce the variables $X_{f(vw)}$ for every $v \in V(G') \setminus V(G)$. That means, we obtain the following polynomials:

$$\frac{1}{|\mathrm{scc}(v)|} \sum_{(v',w') \in \mathrm{scc}(v) \times \mathrm{scc}(w)} X_{v'w'} - X_{f(vw)} \qquad \text{for each } v \in V(G') \setminus V(G).$$

Now take the sum of all polynomials ($\star$) for all $w' \in \operatorname{scc}(w)$, multiplied by $\frac{1}{|\operatorname{scc}(w)|}$. From this, subtract the above polynomials for all $v \in V(G') \setminus V(G)$. This yields Axiom (10.4) for the vertex w because we have $\frac{1}{|\operatorname{scc}(v)|} = \frac{1}{|\operatorname{scc}(w)|}$ for all $v \in V(G') \setminus V(G)$, since all R-SCCs have equal size (Lemma 10.4.14). In a similar way we can derive Axiom (10.5) for an SCC-vertex $v \in V(G') \setminus V(G)$.

Derivation of Axioms (10.6)**:**
Let $v, v' \in V(G'), w, w' \in V(H')$ such that $(v, w) \in \textbf{NewPairs}$ and $v', w' \in V(G) \cup V(H)$, and $v' \sim w'$. Furthermore, assume that $\{(v, w), (v', w')\}$ is not a local isomorphism. Since v and w are newly introduced vertices and v', w' are old ones, it holds $v' \neq v$ and $w' \neq w$. Thus, if $\{(v, w), (v', w')\}$ is not a local isomorphism, there must be a relation symbol $E \in \tau$ such that $(v', v) \in E(G)$ and $(w', w) \notin E(H)$, or $(v, v') \in E(G)$ and $(w, w') \notin E(H)$, or vice versa. Again, we have to distinguish two cases:

Case 1: op = `addPair`**:** In this case, v and w are new pair-vertices representing pairs (v_1, v_2) and (w_1, w_2), respectively. Therefore, the only non-diagonal relations in which they occur are E_{left} and E_{right}. Suppose $(v', v) \in E_{\text{left}}(G')$ and $(w', w) \notin E_{\text{left}}(H')$. That means $v' = v_1$ and $w' \neq w_1$. We take the extension axiom for $X_{f(vw)}$ and multiply it by $X_{v'w'}$ to obtain $X_{v'w'}(X_{v_1w_1}X_{v_2w_2} - X_{f(vw)})$. Since $v' = v_1$ and $w' \neq w_1$, the monomial $X_{v'w'}X_{v_1w_1}$ represents a pebble position that is not a local isomorphism and is therefore an axiom in $P_{\text{iso}}(G, H)$. We can thus derive $X_{v'w'}X_{v_1w_1}X_{v_2w_2}$ and cancel it from the polynomial above. Then we multiply by (-1) and are left with Axiom (10.6), as desired. Similarly, we can derive the axiom in the case that $(v, v') \in E_{\text{right}}(G')$ and $(w, w') \notin E_{\text{right}}(H')$. The symmetric cases in which (w, w') or (w', w) is in the respective relation, and (v, v') or (v', v) is not, are analogous.

Case 2: op = `contract`**:** In this case, v and w are contracted R-SCCs of G and H. Let $E \in \tau$ be a relation symbol such that $(v', v) \in E(G')$ and $(w', w) \notin E(H')$. Then by definition of $E(G' \uplus H')$ (see Section 10.4.2), there exists a $v_1 \in \operatorname{scc}(v)$ such that $(v', v_1) \in E(G)$, and there is no $w_1 \in \operatorname{scc}(w)$ such that $(w', w_1) \in E(H)$. In order to derive $X_{f(vw)}X_{v'w'}$, we multiply the extension axiom

$$\frac{1}{|\operatorname{scc}(v)|} \sum_{(v'', w'') \in \operatorname{scc}(v) \times \operatorname{scc}(w)} X_{v''w''} - X_{f(vw)}$$

with $X_{v'w'}$. From the resulting sum, we can cancel all monomials of the form $X_{v'w'}X_{v''w''}$, for all $v'' \in \operatorname{scc}(v), w'' \in \operatorname{scc}(w)$, because (v', v'') and (w', w'') have distinct colours (using again Lemma 10.4.7 and Lemma 10.2.3). The colours are distinct because there exists an E-edge from v' into $\operatorname{scc}(v)$, but none from w' into $\operatorname{scc}(w)$ (see Lemma 10.4.16). After cancelling these monomials, we are left with $X_{f(vw)}X_{v'w'}$. Again, the symmetric cases work analogously.

Derivation of Axioms (10.7)**:**
Let $v, v' \in V(G'), w, w' \in V(H')$ such that $(v, w) \in$ **NewPairs** and $(v', w') \in$ **NewPairs**. Furthermore, assume that $\{(v, w), (v', w')\}$ is not a local isomorphism. Again, we distinguish between the two operation types:

Case 1: op = `addPair`**:** If all four vertices v, v', w, w' are newly created pair-vertices, then (v, v') and (w, w') are not in any relation. Therefore, the only way how $\{(v, w), (v', w')\}$ can fail to be a local isomorphism is if $v = v'$ and $w \neq w'$ (or vice versa).
So let $v = v'$ and $\text{pair}(v) = \text{pair}(v') = (v_1, v_2)$. Further, let $\text{pair}(w) = (w_1, w_2)$ and $\text{pair}(w') = (w'_1, w'_2)$, where $\text{pair}(w) \neq \text{pair}(w')$. Suppose that $w_1 \neq w'_1$ (if $w_2 \neq w'_2$, the derivation is analogous). We multiply the extension axiom for $X_{f(vw)}$ with $X_{v_1 w'_1}$ and obtain:

$$X_{v_1 w'_1}(X_{v_1 w_1} X_{v_2 w_2} - X_{f(vw)}).$$

Since $w'_1 \neq w_1$, the monomial $X_{v_1 w'_1} X_{v_1 w_1}$ encodes a pebble position which is not a local isomorphism and therefore, it is in $P_{\text{iso}}(G, H)$. Thus, we can derive $X_{v_1 w'_1} X_{v_1 w_1} X_{v_2 w_2}$ and cancel it from the above polynomial, yielding $-X_{v_1 w'_1} X_{f(vw)}$. Now multiply this by $X_{v_2 w'_2}$ and add the result to the lifted extension axiom $X_{f(vw)}(X_{v_1 w'_1} X_{v_2 w'_2} - X_{f(v'w')})$ (recall that $\text{pair}(v') = (v_1, v_2)$). The result, multiplied by (-1), is Axiom (10.7), namely $X_{f(vw)} X_{f(v'w')}$.
Again, the symmetric cases are analogous.

Case 2: op = `contract`**:** In this case, two subcases must be considered because there are two ways in which $\{(v, w), (v', w')\}$ can fail to be a local isomorphism.

Case 2.1: Mismatch of equality types. Like in the previous case, let $v = v'$ and $w \neq w'$. We multiply the extension axiom for $X_{f(vw)}$ with a weighted sum of variables (using the multiplication and the linear combination rule) to obtain:

$$\left(\frac{1}{|\text{scc}(v')|} \sum_{v'_1 \in \text{scc}(v')} \sum_{w_2 \in \text{scc}(w')} X_{v'_1 w_2} \right) \cdot \left(\frac{1}{|\text{scc}(v)|} \sum_{(v_1, w_1) \in \text{scc}(v) \times \text{scc}(w)} X_{v_1 w_1} - X_{f(vw)} \right).$$

Because $\text{scc}(w) \cap \text{scc}(w') = \emptyset$, and $\text{scc}(v') = \text{scc}(v)$, w_1 and w_2 in the above sum are always in distinct SCCs, while v_1 and v'_1 are in the same SCC. Lemma 10.4.15 states that the colours of pairs in the same SCC are distinct from colours of pairs which do not lie in the same SCC. Hence, all monomials of the form $X_{v'_1 w_2} X_{v_1 w_1}$ are derivable from $P_{\text{iso}}(G, H)$ using Lemma 10.4.7 and Lemma 10.2.3. Cancelling these monomials from the above sum yields:

$$\left(\frac{1}{|\text{scc}(v')|} \sum_{v'_1 \in \text{scc}(v')} \sum_{w_2 \in \text{scc}(w')} X_{v'_1 w_2} \right) \cdot \left(-X_{f(vw)} \right).$$

With the help of the extension axiom for $X_{f(v'w')}$, we can replace the sum in this expression by $X_{f(v'w')}$ and are done. Again, symmetric cases work analogously.

Case 2.2: Mismatch of relations. In this case, the reason why $\{(v,w),(v',w')\}$ is not a local isomorphism is that there is a relation $E \in \tau$ such that $(v',v) \in E(G')$ and $(w',w) \notin E(H')$ (again, we skip the symmetric cases because they are analogous). Then by definition of $E(G' \uplus H')$, there exist $v_1' \in \mathrm{scc}(v')$ and $v_1 \in \mathrm{scc}(v)$ such that $(v_1',v_1) \in E(G)$, but for every pair $(w_1',w_1) \in \mathrm{scc}(w') \times \mathrm{scc}(w)$, it holds $(w_1',w_1) \notin E(H)$.

We take the extension axiom for $X_{f(vw)}$ and multiply it with $X_{v_2w_2}$, for all $v_2 \in \mathrm{scc}(v'), w_2 \in \mathrm{scc}(w')$. This yields polynomials of the form (where we now write v'' for the vertices in $\mathrm{scc}(v)$ to avoid confusion with the vertex v'):

$$\frac{1}{|\mathrm{scc}(v)|} \sum_{(v'',w'') \in \mathrm{scc}(v) \times \mathrm{scc}(w)} X_{v''w''}X_{v_2w_2} - X_{f(vw)}X_{v_2w_2}.$$

We obtain such a polynomial for every $v_2 \in \mathrm{scc}(v'), w_2 \in \mathrm{scc}(w')$.
By Lemma 10.4.17, the pairs (v'',v_2) and (w'',w_2) have distinct colours in the coarsest coherent configuration C, for every $v'' \in \mathrm{scc}(v), w'' \in \mathrm{scc}(w)$, because there is an E-edge between $\mathrm{scc}(v')$ and $\mathrm{scc}(v)$, but none between $\mathrm{scc}(w')$ and $\mathrm{scc}(w)$. Therefore, each monomial $X_{v''w''}X_{v_2w_2}$ is derivable from $P_{\mathrm{iso}}(G,H)$ and can be cancelled from the above sums.
So in total, we can derive:

$$-X_{f(vw)}X_{v_2w_2} \text{ , for all } v_2 \in \mathrm{scc}(v'), w_2 \in \mathrm{scc}(w').$$

We use these monomials to cancel all the summands in the product of the extension axiom for $X_{f(v'w')}$ with the variable $X_{f(vw)}$, which is the following expression:

$$\frac{1}{|\mathrm{scc}(v')|} \sum_{(v_2,w_2) \in \mathrm{scc}(v') \times \mathrm{scc}(w')} X_{f(vw)}X_{f(v_2w_2)} - X_{f(vw)}X_{f(v'w')}.$$

Cancelling out the summands as described yields the desired Axiom (10.7):
$X_{f(vw)}X_{f(v'w')}$.

In total, we can derive $P_{\mathrm{iso}}(G',H')$ from $P_{\mathrm{iso}}(G,H)$. The number of new variables is clearly bounded by $|V(G')|^2$. It is also not difficult to see that only polynomially many monomials occur in the derivation, and the binary encoding of the coefficients occurring in them has complexity at most $\mathcal{O}(\log|V(G)|)$. The used extension axioms are all for polynomials that are averaged sums or degree-2 monomials, as mentioned in Lemma 10.5.1. The derivations obtained with Lemma 10.2.3 also have polynomial complexity because they can be carried out in MC_3. □

10.6 Conclusion and future research

We have shown that the degree-3 extended polynomial calculus can simulate the pair- and contract-operations of Deep Weisfeiler Leman in the sense that the axiom system

$P_{\text{iso}}(G', H')$ is derivable from $P_{\text{iso}}(G, H)$ if there is a sequence of DWL-operations that transforms $G \uplus H$ into $G' \uplus H'$. Together with the simulation of k-dimensional Weisfeiler Leman in the degree-k monomial calculus given in [17], this shows that EPC_3 can distinguish two graphs G and H if they can be distinguished in DWL, and the EPC_3-refutation has the same complexity as the DWL-algorithm. Since DWL-algorithms and CPT-programs mutually simulate each other, this result upper-bounds the graph distinguishing power of CPT by that of EPC_3.

This raises the question whether a super-polynomial lower bound for graph isomorphism can be established for EPC_3, preferably for graph classes such as unordered CFI-graphs or multipedes, whose isomorphism problem reduces to a linear equation system and is thus in PTIME. If such a lower bound is found, then by Theorem 10.0.3, we would also have that CPT $\neq$ PTIME.

Unfortunately, we do not know how strong the system EPC_3 is, and in particular, if the degree-restriction is a true limitation. It may even be the case that EPC_3 polynomially simulates the *unbounded-degree* extended polynomial calculus. Then it would be as strong as extended Frege because in EPC, the extension variables can encode arbitrary polynomials and thereby arbitrary Boolean circuits. This would make it less useful for proving lower bounds against CPT, as extended Frege lower bounds seem to be out of reach at the moment.
However, Theorem 10.0.1 also asserts that the simulation of CPT is possible using only extension axioms of a limited form, namely for degree-2 monomials and averaged sums. In this restricted version of EPC_3, the obvious representation of Boolean circuits as polynomials is no longer possible: The Boolean functions $X \wedge Y, X \vee Y$, and $\neg X$ can naturally be represented as the polynomials $X \cdot Y, X + Y - X \cdot Y$, and $1 - X$. When we represent Boolean circuits using extension variables, then each extension variable corresponds to a gate in the circuit. If the only allowed extension axioms are $\frac{}{X_f - f}$ for $f = X \cdot Y$ or $f = \frac{1}{n}\sum_{i=1}^{n^2} X_i$, then the only gates that we can naturally express are AND-gates (with extension axioms of the first type). Neither NOT-gates nor OR-gates can be simulated (directly) by such extension axioms because this requires sums which are not of the form $\frac{1}{n}\sum X_i$. In particular, these extension axioms cannot be applied to polynomials where variables occur with a negative coefficient. Hence, the corresponding circuits are in some sense *monotone*. This is of course no formal proof that EPC_3 with restricted extension axioms is strictly weaker than extended Frege but at least it rules out the natural simulation of Boolean circuits in EPC_3. In total, the success chances of our suggested approach for CPT lower bounds via proof complexity depend highly on the true power of EPC_3 (with restricted extension axioms), and its relation to unrestricted EPC. Investigating this remains a problem for future work.

Symmetry-invariance of the refutations Actually, our Theorem 10.0.1 could be strengthened more: A simulation of CPT in EPC_3 is even possible in a certain *symmetry-invariant*

fragment of EPC$_3$. However, it seems tricky to give a precise definition of "*symmetric* EPC$_3$" that is both natural and fits the kind of symmetry we encounter in our CPT-simulation. A neat way to put it would be to say that the set of extension axioms used in a derivation has to be closed under symmetries. With the right definition of "symmetries", this is indeed true for the refutation constructed in Lemma 10.5.1. Namely, whenever an extension variable $X_{f(vw)}$ is introduced, where v and w are new pair- or SCC-vertices, then we introduce it for *all* $(v,w) \in V(G') \times V(H')$ that are new. The corresponding polynomials $f(vw)$ consist of variables that refer to the vertices in the respective pairs or SCCs of v and w. The automorphisms of the graphs G and H preserve the colours of all vertex-pairs in the coarsest coherent configuration. Therefore, the set of extension axioms that we introduce in each step of the refutation is closed under the automorphisms of G and H. Actually, it is even closed under $\mathcal{C}^3$-types in some sense and can thus even be meaningfully symmetric when the graphs are rigid. However, it is quite awkward to formalise this symmetry condition in the EPC$_3$-context. Therefore, let us look at the easier symmetry notion with respect to automorphisms: The action of the automorphism groups $\mathbf{Aut}(G)$ and $\mathbf{Aut}(H)$ on the set of variables of $P_{\text{iso}}(G,H)$ is the natural one, i.e. if π is an automorphism of G and σ an automorphism of H, then they take X_{vw} to $X_{\pi(v)\sigma(w)}$. This extends naturally to the extension axioms, so for example, if v and w are pair-vertices with $\text{pair}(v) = (v_1, v_2), \text{pair}(w) = (w_1, w_2)$, then the extension axiom $X_{f(vw)} - X_{v_1w_1}X_{v_2w_2}$ is mapped to $X_{f(v'w')} - X_{\pi(v_1)\sigma(w_1)}X_{\pi(v_2)\sigma(w_2)}$, where v', w' are the newly introduced pair-vertices for $(\pi(v_1), \pi(v_2))$ and $(\sigma(w_1), \sigma(w_2))$ (such v', w' must exist because DWL is isomorphism-invariant and introduces new vertices for all pairs with the same colour). So in this sense, the extension axioms used in Lemma 10.5.1 are closed under all automorphism-pairs $(\pi, \sigma) \in \mathbf{Aut}(G) \times \mathbf{Aut}(H)$.
Unfortunately, this does not lead to a *general* definition of symmetric EPC$_3$ because it depends on the automorphisms of G and H, the graphs which are implicitly encoded in $P_{\text{iso}}(G,H)$. When EPC$_3$ is applied to other polynomial axiom systems, then there might be no graphs "in the background". So for a general set of input polynomials $\mathcal{P}$, it would be natural to require that the set of extension axioms in a refutation be closed under the automorphisms of $\mathcal{P}$ – these are the permutations of the variables that extend to permutations of the polynomials in $\mathcal{P}$. However, this would no longer fit to our derivation from Lemma 10.5.1: The system $P_{\text{iso}}(G,H)$ in general has more automorphisms than $\mathbf{Aut}(G) \times \mathbf{Aut}(H)$. Namely, $P_{\text{iso}}(G,H)$ contains no information about where the edges and non-edges in G and H actually are; it just relates pairs $(v,v') \in V(G)^2$ with pairs $(w,w') \in V(H)^2$ where (v,v') is an edge and (w,w') is not, or vice versa (Axiom (10.3)). Therefore, an automorphism of $P_{\text{iso}}(G,H)$ may swap all edges with non-edges, as long as it does so in both G and H (such examples can be constructed). But the automorphisms of the graphs must preserve edges and non-edges, so such an automorphism of $P_{\text{iso}}(G,H)$ does not correspond to one from $\mathbf{Aut}(G) \times \mathbf{Aut}(H)$. Our constructed refutation is only symmetric with respect to the latter. Thus, our simulation of CPT in EPC$_3$ is possible in a way that respects *specific* symmetries of $P_{\text{iso}}(G,H)$, but we do not know if this kind of symmetry-invariance can be formulated independently of the graph isomorphism problem as a general restriction to the proof system EPC$_3$.

It is an intriguing direction for future research to formulate a generic way of symmetrising known proof systems such as the extended polynomial calculus or variants of resolution and Frege. This would not only yield the right framework for expressing our symmetric graph isomorphism refutations that we have obtained from CPT but it would also open the field of proof complexity to symmetry-based lower bound methods: It is conceivable that lower bounds for proof systems such as Frege or even extended Frege, which have been notoriously difficult in the asymmetric setting, may be provable for suitable *symmetric* versions of these proof systems. An approach towards symmetric proof systems that seems particularly promising to us is to consider the so-called *Ideal Proof System* (IPS) by Grochow and Pitassi [58]. Proofs in this system are algebraic circuits – these are predestined for putting a symmetry restriction on them. Moreover, IPS allows to naturally simulate all standard proof systems such as (extended) Frege or the (extended) polynomial calculus, so making IPS symmetric should symmetrise all other proof systems, too. Lower bounds for symmetric IPS could then, hopefully, be obtained with known techniques from the study of symmetric circuits. Coincidentally, this would also be the ideal way to unify the two major directions we have explored in this thesis, namely symmetric circuits and proof complexity.

Bibliography

[1] Serge Abiteboul and Victor Vianu. Datalog extensions for database queries and updates. *Journal of Computer and System Sciences*, 43(1):62–124, 1991.

[2] Serge Abiteboul and Victor Vianu. Computing with first-order logic. *Journal of Computer and System Sciences*, 50(2):309–335, 1995.

[3] Faried Abu Zaid, Anuj Dawar, Erich Grädel, and Wied Pakusa. Definability of Summation Problems for Abelian Groups and Semigroups. In *Proceedings of 32th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, 2017. URL: `http://www.logic.rwth-aachen.de/pub/graedel/AbuzaidDawGraPak17.pdf`.

[4] Faried Abu Zaid, Erich Grädel, Martin Grohe, and Wied Pakusa. Choiceless Polynomial Time on structures with small Abelian colour classes. In *Mathematical Foundations of Computer Science 2014*, volume 8634 of *Lecture Notes in Computer Science*, pages 50–62. Springer, 2014. URL: `http://logic.rwth-aachen.de/pub/pakusa/cptcan.pdf`.

[5] Yaroslav Alekseev. A Lower Bound for Polynomial Calculus with Extension Rule. In *36th Computational Complexity Conference (CCC 2021)*, volume 200 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 21:1–21:18, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: `https://drops.dagstuhl.de/opus/volltexte/2021/14295`, `doi:10.4230/LIPIcs.CCC.2021.21`.

[6] Matthew Anderson and Anuj Dawar. On symmetric circuits and fixed-point logics. *Theory of Computing Systems*, 60(3):521–551, 2017.

[7] Matthew Anderson, Anuj Dawar, and Bjarki Holm. Maximum matching and linear programming in fixed-point logic with counting. In *2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 173–182. IEEE, 2013.

[8] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach.* Cambridge University Press, 2009.

[9] Albert Atserias. On sufficient conditions for unsatisfiability of random formulas. *Journal of the ACM (JACM)*, 51(2):281–311, 2004.

[10] Albert Atserias, Andrei Bulatov, and Anuj Dawar. Affine systems of equations and counting infinitary logic. *Theoretical Computer Science*, 410(18):1666–1683, 2009.

[11] Albert Atserias and Víctor Dalmau. A combinatorial characterization of resolution width. *Journal of Computer and System Sciences*, 74(3):323–334, 2008.

[12] Albert Atserias and Anuj Dawar. Definable inapproximability: new challenges for duplicator. *Journal of Logic and Computation*, 29(8):1185–1210, 2019.

[13] Albert Atserias and Joanna Ochremiak. Definable ellipsoid method, sums-of-squares proofs, and the isomorphism problem. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 66–75, 2018.

[14] László Babai, Peter Cameron, and Péter Pálfy. On the orders of primitive groups with restricted nonabelian composition factors. *Journal of Algebra*, 79(1):161–168, 1982.

[15] László Babai. Graph isomorphism in quasipolynomial time. 2015. URL: `https://arxiv.org/abs/1512.03547`, `doi:10.48550/ARXIV.1512.03547`.

[16] Paul Beame and Toniann Pitassi. Propositional proof complexity: Past, present, and future. *Current Trends in Theoretical Computer Science Entering the 21st Century*, pages 42–70, 2001.

[17] Christoph Berkholz and Martin Grohe. Limitations of algebraic approaches to graph isomorphism testing. In *International Colloquium on Automata, Languages, and Programming*, pages 155–166. Springer, 2015.

[18] Andreas Blass and Yuri Gurevich. Strong extension axioms and Shelah's zero-one law for choiceless polynomial time. *The Journal of Symbolic Logic*, 68(1):65–131, 2003.

[19] Andreas Blass and Yuri Gurevich. A new zero-one law and strong extension axioms. In *Current Trends in Theoretical Computer Science: The Challenge of the New Century Vol 1: Algorithms and Complexity Vol 2: Formal Models and Semantics*, pages 99–118. World Scientific, 2004.

[20] Andreas Blass and Yuri Gurevich. A quick update on the open problems in Blass-Gurevich-Shelah's article "On polynomial time computations over unordered structures". 2005. URL: `https://www.microsoft.com/en-us/research/wp-content/uploads/2017/01/150a.pdf`.

[21] Andreas Blass, Yuri Gurevich, and Saharon Shelah. Choiceless polynomial time. *Annals of Pure and Applied Logic*, 100(1-3):141–187, 1999.

[22] Andreas Blass, Yuri Gurevich, and Saharon Shelah. On polynomial time computation over unordered structures. *The Journal of Symbolic Logic*, 67(3):1093–1125, 2002.

[23] Andreas Blass, Yuri Gurevich, and Jan Van den Bussche. Abstract state machines and computationally complete query languages. *Information and Computation*, 174(1):20–36, 2002.

[24] Hans Bodlaender. Treewidth: Structure and algorithms. In *International Colloquium on Structural Information and Communication Complexity*, pages 11–25. Springer, 2007.

[25] Mikołaj Bojańczyk. *Slightly infinite sets.* 2019. URL: `https://www.mimuw.edu.pl/~bojan/upload/main-10.pdf`.

[26] Mikołaj Bojańczyk and Szymon Toruńczyk. On computability and tractability for infinite sets. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 145–154, 2018.

[27] Andrei Bulatov. A dichotomy theorem for nonuniform CSPs. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 319–330. IEEE, 2017.

[28] Sam Buss. Propositional proofs in Frege and Extended Frege systems. In *International Computer Science Symposium in Russia*, pages 1–6. Springer, 2015.

[29] J. Cai, M. Fürer, and N. Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12:389–410, 1992.

[30] Ashok Chandra and David Harel. Structure and complexity of relational queries. In *21st Annual Symposium on Foundations of Computer Science (sfcs 1980)*, pages 333–347. IEEE, 1980. `doi:10.1109/SFCS.1980.41`.

[31] Gang Chen and Ilia Ponomarenko. Lectures on coherent configurations. *Lecture notes*, 2019. URL: `http://www.pdmi.ras.ru/~inp/ccNOTES.pdf`.

[32] Matthew Clegg, Jeffery Edmonds, and Russell Impagliazzo. Using the Groebner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 174–183, 1996.

[33] Laszlo Csanky. Fast parallel matrix inversion algorithms. In *16th Annual Symposium on Foundations of Computer Science (sfcs 1975)*, pages 11–12. IEEE, 1975.

[34] Anuj Dawar. The nature and power of fixed-point logic with counting. *ACM SIGLOG News*, 2(1):8–21, 2015.

[35] Anuj Dawar, Erich Grädel, and Wied Pakusa. Approximations of Isomorphism and Logics with Linear-Algebraic Operators. In *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 112:1–112:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: `http://drops.dagstuhl.de/opus/volltexte/2019/10688`, `doi:10.4230/LIPIcs.ICALP.2019.112`.

[36] Anuj Dawar, Martin Grohe, Bjarki Holm, and Bastian Laubner. Logics with rank operators. In *2009 24th Annual IEEE Symposium on Logic In Computer Science*, pages 113–122. IEEE, 2009.

[37] Anuj Dawar, Erich Grädel, and Moritz Lichter. Limitations of the invertible-map equivalences. *Journal of Logic and Computation*, 09 2022. URL: `https://academic.oup.com/logcom/advance-article/doi/10.1093/logcom/exac058/6687793?guestAccessKey=7349c158-be02-4116-815a-840ac7880a03`, `doi:10.1093/logcom/exac058`.

[38] Anuj Dawar and Bjarki Holm. Pebble games with algebraic rules. In *International Colloquium on Automata, Languages, and Programming*, pages 251–262. Springer, 2012.

[39] Anuj Dawar and David Richerby. A fixed-point logic with symmetric choice. In *International Workshop on Computer Science Logic*, pages 169–182. Springer, 2003.

[40] Anuj Dawar, David Richerby, and Benjamin Rossman. Choiceless Polynomial Time, Counting and the Cai–Fürer–Immerman graphs. *Annals of Pure and Applied Logic*, 152(1-3):31–50, 2008.

[41] Anuj Dawar and Gregory Wilsenach. Symmetric Arithmetic Circuits. In *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*, volume 168 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 36:1–36:18, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. URL: `https://drops.dagstuhl.de/opus/volltexte/2020/12443`, `doi:10.4230/LIPIcs.ICALP.2020.36`.

[42] Anuj Dawar and Gregory Wilsenach. Symmetric circuits for rank logic. *ACM Transactions on Computational Logic (TOCL)*, 23(1):1–35, 2021.

[43] Anuj Dawar and Gregory Wilsenach. Lower Bounds for Symmetric Circuits for the Determinant. In *13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*, volume 215 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 52:1–52:22, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: `https://drops.dagstuhl.de/opus/volltexte/2022/15648`, `doi:10.4230/LIPIcs.ITCS.2022.52`.

[44] Susanna de Rezende, Massimo Lauria, Jakob Nordström, and Dmitry Sokolov. The power of negative reasoning. In *36th Computational Complexity Conference (CCC 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

[45] John Dixon and Brian Mortimer. *Permutation Groups*. Springer, New York, 1996.

[46] Luca Donetti, Franco Neri, and Miguel Muñoz. Optimal network topologies: expanders, cages, Ramanujan graphs, entangled networks and all that. *Journal of Statistical Mechanics: Theory and Experiment*, 2006(08):P08007–P08007, aug 2006. URL: `https://doi.org/10.1088%2F1742-5468%2F2006%2F08%2Fp08007`, `doi:10.1088/1742-5468/2006/08/p08007`.

[47] Heinz-Dieter Ebbinghaus and Jörg Flum. *Finite Model Theory*. Springer, 1999.

[48] Kousha Etessami and Neil Immerman. Tree canonization and transitive closure. In *Proceedings of Tenth Annual IEEE Symposium on Logic in Computer Science*, pages 331–341. IEEE, 1995.

[49] Ronald Fagin. Generalized first-order spectra and polynomial-time recognizable sets. *Complexity of computation*, 7:43–73, 1974.

[50] Stanley Florkowski III. Spectral graph theory of the hypercube. Technical report, NAVAL POSTGRADUATE SCHOOL MONTEREY CA, 2008.

[51] F. Gire and H.K. Hoang. An extension of fixpoint logic with a symmetry-based choice construct. *Information and Computation*, 144(1):40–65, 1998. `doi:https://doi.org/10.1006/inco.1998.2712`.

[52] E. Grädel, W. Pakusa, S. Schalthöfer, and L. Kaiser. Characterising Choiceless Polynomial Time with First-Order Interpretations. In *Proceedings of the 30th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 677–688, 2015.

[53] Erich Grädel and Martin Grohe. Is Polynomial Time Choiceless? In *Fields of Logic and Computation II*, pages 193–209. Springer, 2015.

[54] Erich Grädel and Martin Otto. Inductive definability with counting on finite structures. In *Computer Science Logic: 6th Workshop, CSL'92 San Miniato, Italy, September 28–October 2, 1992 Selected Papers 6*, pages 231–247. Springer, 1993.

[55] Erich Grädel and Wied Pakusa. Rank logic is dead, long live rank logic! *The Journal of Symbolic Logic*, 84(1), March 2019. URL: `https://www.cambridge.org/core/journals/journal-of-symbolic-logic/article/rank-logic-is-dead-long-live-rank-logic/F161E336281F2B67E8B3E4CDA5614933`.

[56] Erich Grädel and Svenja Schalthöfer. Choiceless Logarithmic Space. In *44th International Symposium on Mathematical Foundations of Computer Science (MFCS 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[57] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics*. Addison-Wesley, 2 edition, 1994.

[58] Joshua Grochow and Toniann Pitassi. Circuit complexity, proof complexity, and polynomial identity testing: The ideal proof system. *Journal of the ACM (JACM)*, 65(6):1–59, 2018.

[59] Martin Grohe. The quest for a logic capturing PTIME. In *2008 23rd Annual IEEE Symposium on Logic in Computer Science*, pages 267–271. IEEE, 2008. `doi:10.1109/LICS.2008.11`.

[60] Martin Grohe. Fixed-point definability and polynomial time on graphs with excluded minors. *Journal of the ACM (JACM)*, 59(5):1–64, 2012.

[61] Martin Grohe. *Descriptive complexity, canonisation, and definable graph structure theory.* Cambridge University Press, 2017.

[62] Martin Grohe, Berit Grußien, André Hernich, and Bastian Laubner. L-Recursion and a new Logic for Logarithmic Space. In *Computer Science Logic (CSL'11) - 25th International Workshop/20th Annual Conference of the EACSL*, volume 12 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 277–291, Dagstuhl, Germany, 2011. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: `http://drops.dagstuhl.de/opus/volltexte/2011/3237`, `doi:10.4230/LIPIcs.CSL.2011.277`.

[63] Martin Grohe, Pascal Schweitzer, and Daniel Wiebking. Deep Weisfeiler Leman, 2020. `arXiv:2003.10935`.

[64] Berit Grußien. *Capturing Polynomial Time and Logarithmic Space using Modular Decompositions and Limited Recursion.* PhD thesis, Humboldt-Universität zu Berlin, 2016.

[65] Erich Grädel et al. *Finite Model Theory and its Applications.* Springer, 2007.

[66] Yuri Gurevich. Logic and the Challenge of Computer Science. In *Current Trends in Theoretical Computer Science.* Computer Science Press, 1988.

[67] Yuri Gurevich and Saharon Shelah. On finite rigid structures. *The Journal of Symbolic Logic*, 61(2):549–562, 1996.

[68] Armin Haken. The intractability of resolution. *Theoretical computer science*, 39:297–308, 1985.

[69] Frank Harary. The automorphism group of a hypercube. *J. Univers. Comput. Sci.*, 6(1):136–138, 2000.

[70] William He and Benjamin Rossman. Symmetric formulas for products of permutations, 2022. URL: `https://arxiv.org/abs/2211.15520`.

[71] Lauri Hella. Logical hierarchies in PTIME. *Information and Computation*, 129(1):1–19, 1996.

[72] Lauri Hella, Phokion Kolaitis, and Kerkko Luosto. Almost everywhere equivalence of logics in finite model theory. *Bulletin of Symbolic Logic*, 2(4):422–443, 1996.

[73] Bjarki Holm. *Descriptive Complexity of Linear Algebra.* PhD thesis, University of Cambridge, 2010.

[74] Neil Immerman. Relational queries computable in polynomial time. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 147–152, 1982.

[75] Neil Immerman. *Descriptive Complexity.* Springer Science, 2012.

[76] Neil Immerman and Eric Lander. Describing graphs: A first-order approach to graph canonization. In *Complexity theory retrospective*, pages 59–81. Springer, 1990.

[77] Russell Impagliazzo, Pavel Pudlák, and Jiri Sgall. Lower bounds for the polynomial calculus and the Gröbner basis algorithm. *Computational Complexity*, 8(2):127–144, 1999.

[78] Sandra Kiefer. The Weisfeiler-Leman algorithm: an exploration of its power. *ACM SIGLOG News*, 7(3):5–27, 2020.

[79] Jan Krajíček. *Proof Complexity*, volume 170. Cambridge University Press, 2019.

[80] Bastian Laubner. *The Structure of Graphs and New Logics for the Characterization of Polynomial Time*. PhD thesis, Humboldt-Universität zu Berlin, 2011.

[81] Leonid Libkin. *Elements of Finite Model Theory*. Springer, 2004.

[82] Moritz Lichter. Separating Rank Logic from Polynomial Time. *J. ACM*, nov 2022. URL: `https://doi.org/10.1145/3572918`.

[83] Moritz Lichter. Witnessed Symmetric Choice and Interpretations in Fixed-Point Logic with Counting, 2022. URL: `https://arxiv.org/abs/2210.07869`.

[84] Moritz Lichter and Pascal Schweitzer. Canonization for Bounded and Dihedral Color Classes in Choiceless Polynomial Time. In *29th EACSL Annual Conference on Computer Science Logic (CSL 2021)*, volume 183 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 31:1–31:18, Dagstuhl, Germany, 2021. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. URL: `https://drops.dagstuhl.de/opus/volltexte/2021/13465`, `doi:10.4230/LIPIcs.CSL.2021.31`.

[85] Moritz Lichter and Pascal Schweitzer. Choiceless Polynomial Time with Witnessed Symmetric Choice. LICS '22. Association for Computing Machinery, 2022. URL: `https://doi.org/10.1145/3531130.3533348`.

[86] Martin Otto. *Bounded Variable Logics and Counting*, volume 9 of *Lecture Notes in Logic*. Springer, 1997.

[87] Benedikt Pago. Choiceless Computation and Symmetry: Limitations of Definability. In *29th EACSL Annual Conference on Computer Science Logic (CSL 2021)*, volume 183 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 33:1–33:21, Dagstuhl, Germany, 2021. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. URL: `https://drops.dagstuhl.de/opus/volltexte/2021/13467`, `doi:10.4230/LIPIcs.CSL.2021.33`.

[88] Benedikt Pago. Finite Model Theory and Proof Complexity Revisited: Distinguishing Graphs in Choiceless Polynomial Time and the Extended Polynomial Calculus. In *31st EACSL Annual Conference on Computer Science Logic (CSL*

2023), volume 252 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 31:1–31:19, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: `https://drops.dagstuhl.de/opus/volltexte/2023/17492`, `doi:10.4230/LIPIcs.CSL.2023.31`.

[89] Wied Pakusa. *Linear Equation Systems and the Search for a Logical Characterisation of Polynomial Time.* PhD thesis, RWTH Aachen, 2015.

[90] Wied Pakusa, Benedikt Pago, Martin Grohe, and Erich Grädel. A Finite-Model-Theoretic View on Propositional Proof Complexity. *Logical Methods in Computer Science*, 15, 2019.

[91] Wied Pakusa, Svenja Schalthöfer, and Erkal Selman. Definability of Cai-Fürer-Immerman problems in Choiceless Polynomial Time. *ACM Transactions on Computational Logic (TOCL)*, 19(2):1–27, 2018. `doi:10.1145/3154456`.

[92] Alexander Razborov. Lower bounds for the polynomial calculus. *Computational Complexity*, 7(4):291–324, 1998.

[93] David Richerby. *Fixed-point logics with choice.* PhD thesis, University of Cambridge, 2004.

[94] Harvey Rose. Series, Jordan–Hölder Theorem and the Extension Problem. In *A Course on Finite Groups*, pages 187–207. Springer London, London, 2009. `doi:10.1007/978-1-84882-889-6_9`.

[95] Benjamin Rossman. Choiceless Computation and Symmetry. In *Fields of Logic and Computation*, pages 565–580. Springer, 2010.

[96] Benjamin Rossman. Subspace-Invariant AC^0 Formulas. *Logical Methods in Computer Science*, 15, 2019.

[97] Svenja Schalthöfer. *Choiceless Computation and Logic.* PhD thesis, RWTH Aachen, 2020.

[98] Svenja Schalthöfer. Computing on Abstract Structures with Logical Interpretations. Master's thesis, RWTH Aachen University, 2013.

[99] Uwe Schöning. Graph Isomorphism is in the Low Hierarchy. *Journal of Computer and System Sciences*, 37(3):312–323, 1988.

[100] Nathan Segerlind. The Complexity of Propositional Proofs. *Bulletin of symbolic Logic*, 13(4):417–481, 2007.

[101] Saharon Shelah. Choiceless Polynomial Time Logic: Inability to Express. In *International Workshop on Computer Science Logic*, pages 72–125. Springer, 2000.

[102] L. Sunil Chandran and T. Kavitha. The treewidth and pathwidth of hypercubes. *Discrete Mathematics*, 306(3):359–365, 2006. URL: `https://www.sciencedirect.com/science/article/pii/S0012365X05006102`, `doi:https://doi.org/10.1016/j.disc.2005.12.011`.

[103] Moshe Vardi. The complexity of relational query languages. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 137–146, 1982.

[104] Daniel Wiebking. *A Decomposition-Compatible Canonization Framework for the Graph Isomorphism Problem.* PhD thesis, RWTH Aachen University, 2021.

[105] Dmitriy Zhuk. A Proof of the CSP Dichotomy Conjecture. *Journal of the ACM (JACM)*, 67(5):1–78, 2020.