TECHNISCHE
UNIVERSITÄT
DARMSTADT

Continuing the Quest for a Logic Capturing Polynomial Time – Potential, Limitations, and Interplay of Current Approaches

Vom Fachbereich Mathematik
der Technischen Universität Darmstadt
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)
genehmigte

Dissertation

von

Moritz Lichter, M.Sc.
aus Zürich

Referent:	Prof. Dr. Pascal Schweitzer
1. Korreferent:	Prof. Dr. Martin Otto
2. Korreferent:	Prof. Dr. Martin Grohe

Tag der Einreichung:	18. April 2023
Tag der mündlichen Prüfung:	23. Juni 2023

Darmstadt, 2023

Continuing the Quest for a Logic Capturing Polynomial Time –
Potential, Limitations, and Interplay of Current Approaches
Doctoral thesis by Moritz Lichter, M.Sc.
Darmstadt, Technische Universität Darmstadt, 2023, D17

Referee:	Prof. Dr. Pascal Schweitzer
1st co-referee:	Prof. Dr. Martin Otto
2nd co-referee:	Prof. Dr. Martin Grohe

Date of hand-in :	April 18, 2023
Date of thesis defense :	June 23, 2023

Published by TUprints in 2023.
Veröffentlicht durch TUprints im Jahr 2023.
URN: urn:nbn:de:tuda-tuprints-242443
URL: http://tuprints.ulb.tu-darmstadt.de/24244

Abstract

The search for a logic capturing PTIME is one central question of finite model theory and descriptive complexity theory: Is there a reasonable logic in which exactly the polynomial-time decidable properties of finite relational structures, e.g., of finite graphs, are definable? This thesis continues the search and investigates all current approaches and candidates for such logics. We examine and compare their expressive power, show limitations, and study the combination and relationship of them. We contribute the following new results:

First, we show that the quantifier depth of 3-variable first-order logic with counting needed to distinguish two n-element structures is in $\mathcal{O}(n \log n)$. Equivalently, the 2-*dimensional Weisfeiler-Leman algorithm* stabilizes in $\mathcal{O}(n \log n)$ iterations. This upper bound matches the known $\Omega(n)$ lower bound up to a logarithmic factor. We use representation-theoretic arguments for matrix algebras closely related to the algorithm.

Second, we consider the logic of *Choiceless Polynomial Time* (CPT). This logic expresses all choice-less polynomial-time computations on relational structures. CPT is a promising candidate for a logic capturing PTIME. Because proving or disproving that CPT captures PTIME is currently out of reach, capturing PTIME with CPT on restricted classes of structures is studied. Capturing PTIME is usually achieved by defining canonization. We show that CPT canonizes structures with bounded color class size for which every color class induces a dihedral automorphism group. The canonization is based on a new normal form, a classification of certain subdirect products of dihedral groups, and the existing canonization for structures with bounded color class size and abelian colors.

Third, we consider the combination of two approaches to capture PTIME. We combine CPT with *witnessed symmetric choice* (CPT+WSC). This restricted choice-mechanism guarantees that the result of a CPT-computation is independent of the choices made. While defining canonization is the usual way to capture PTIME, it is unknown whether canonization has to be definable. For CPT+WSC, we show that a CPT+WSC-definable isomorphism test for a class of structures implies a CPT+WSC-definable canonization. If isomorphism for this class is actually polynomial-time decidable, capturing PTIME with CPT+WSC is equivalent to defining isomorphism in CPT+WSC.

Fourth, we further investigate witnessed symmetric choice. Showing that witnessed symmetric choice makes CPT more expressive would separate CPT from PTIME, which itself is an open problem. We consider *inflationary fixed-point logic with counting* (IFPC) as base logic, for which witnessed symmetric choice strictly increases expressiveness. We separate IFPC+WSC from PTIME and show that the further extension with an operator based on logical interpretations (IFPC+WSC+I) is strictly more expressive. Hence, at least for IFPC, witnessed symmetric choice alone is too weak to capture PTIME. We also make a first step to separate IFPC+WSC+I from PTIME.

Last, we consider extensions of IFPC by operators from linear algebra. We separate *rank logic*, the second promising candidate, from PTIME. Rank logic extends IFPC by an operator to define ranks over finite fields. We show that rank logic fails to define isomorphism for certain structures for which isomorphism is polynomial-time decidable and actually CPT-definable. Hence, rank logic cannot even capture CPT. We also show that the more general *linear-algebraic logic* fails to define this isomorphism problem. This logic encompasses every extension of IFPC by linear-algebraic operators over finite fields. Consequently, linear algebra over finite fields is too weak to capture PTIME.

Zusammenfassung

Die Suche nach einer Logik für Polynomialzeit (Ptime) ist eine zentrale Frage der endlichen Modelltheorie und der deskriptiven Komplexitätstheorie: Gibt es eine vernünftige Logik, die genau die in Polynomialzeit entscheidbaren Eigenschaften endlicher relationaler Strukturen, z. B. endlicher Graphen, definiert? Diese Arbeit setzt die Suche fort und betrachtet alle derzeitigen Ansätze und Kandidaten für solche Logiken. Wir untersuchen und vergleichen deren Ausdrucksstärke, zeigen Limitierungen und studieren deren Kombination und Beziehungen zueinander. Wir tragen folgende neue Ergebnisse bei:

Wir zeigen zuerst, dass die Quantorentiefe, die in Logik erster Stufe mit 3 Variablen und Zählen benötigt wird, um zwei n-elementige Strukturen zu unterscheiden, in $\mathcal{O}(n \log n)$ ist. Dies ist äquivalent dazu, dass der 2-*dimensionale Weisfeiler-Leman Algorithmus* nach $\mathcal{O}(n \log n)$ Iterationen stabilisiert. Diese obere Schranke entspricht der bekannten $\Omega(n)$ unteren Schranke bis auf einen logarithmischen Faktor.

Wir betrachten zweitens die Logik *Choiceless Polynomial Time* (CPT). Diese Logik drückt alle wahlfreien Polynomialzeit-Berechnungen auf relationalen Strukturen aus. CPT ist ein vielversprechender Kandidat für eine Logik für Ptime. Wir untersuchen die logische Charakterisierung von Ptime auf Klassen von Strukturen. Dies zeigt man meistens durch das Definieren einer Kanonisierung. Wir zeigen, dass CPT für Strukturen mit beschränkter Farbklassengröße, für welche jede Farbklasse eine Diedergruppe als Automorphismengruppe induziert, eine Kanonisierung definiert. Diese basiert auf einer neuen Normalform und einer Klassifizierung spezieller subdirekter Produkte von Diedergruppen.

Wir kombinieren drittens zwei verschiedene Ansätze: Wir erweitern CPT um *bezeugte symmetrische Wahl* (CPT+WSC). Dieser eingeschränkte Wahlmechanismus garantiert, dass das Ergebnis einer CPT-Berechnung immer unabhängig von den getroffenen Wahlen ist. Auch wenn Kanonisierung der übliche Weg ist, um Ptime zu charakterisieren, ist es unbekannt, ob Kanonisierung definierbar sein muss. Wir zeigen, dass in CPT+WSC ein definierbarer Isomorphietest eine definierbare Kanonisierung impliziert. Wenn Isomorphie für eine Klasse von Strukturen in Polynomialzeit entscheidbar ist, dann ist in CPT+WSC das Charakterisieren von Ptime äquivalent zum Definieren von Isomorphie.

Viertens untersuchen wir bezeugte symmetrische Wahl genauer. Ein Beweis, dass bezeugte symmetrische Wahl CPT ausdrucksstärker macht, würde CPT von Ptime trennen, was selbst ein offenes Problem ist. Wir betrachten *inflationäre Fixpunktlogik mit Zählen* (IFPC), für welche bezeugte symmetrische Wahl die Ausdrucksstärke strikt erhöht. Wir trennen IFPC+WSC von Ptime und zeigen, dass die Erweiterung mit einem Operator basierend auf logischen Interpretationen (IFPC+WSC+I) strikt ausdrucksstärker ist. Also ist bezeugte symmetrische Wahl zu schwach, um Ptime in IFPC zu charakterisieren.

Zuletzt betrachten wir Erweiterungen von IFPC mit Operatoren aus der linearen Algebra. Wir trennen *Ranglogik*, den zweiten vielversprechenden Kandidaten, von Ptime. Ranglogik erweitert IFPC mit einem Operator, der Ränge über endlichen Körpern definiert. Wir zeigen, dass Ranglogik Isomorphie für eine Klasse von Strukturen nicht definiert, für welche Isomorphie sogar CPT-definierbar ist. Also umfasst Ranglogik nicht einmal CPT. Wir zeigen weiterhin, dass die allgemeinere *linear-algebraische Logik* dieses Isomorphieproblem ebenso nicht definiert. Diese Logik umfasst jede Erweiterung von IFPC mit linear-algebraisches Operatoren über endlichen Körpern. Folglich ist lineare Algebra über endlichen Körpern zu schwach um Ptime charakterisieren.

Acknowledgments

I am thankful towards all people who supported me and my research during the past years. Most importantly, I am grateful to my supervisor Pascal Schweitzer for his support, his advice, all the things I have learned from him, and especially for giving me the freedom to choose the topic to work on and trusting me that after some time I will come up with interesting results. I also thank my colleagues for the great group and the pleasant and fruitful working atmosphere in which I was happy to work.

Special thanks go to my coauthors Anuj Dawar, Erich Grädel, and Illia Ponomarenko. Moreover, I thank Martin Grohe and Martin Otto for reading and evaluating this thesis. Many thanks go to all proofreaders for their time and effort. Thanks to Markus Anders, Oliver Bachtler, Sofia Brenner, Irene Heinrich, Eda Kaja, Benedikt Pago, Giovanna Pirillo, Simon Raßmann, and Thomas Schneider. I also thank the European Research Council for funding me and my work.

Finally, I want to express my thanks to my family and my friends for supporting me, for reaching out to me, and for enduring during the last year that I was even more annoyed by everything than I usually am anyhow.

Contents

Chapter 1

Introduction

A logic in the context of model theory is, roughly speaking, a fixed syntactical way to describe properties of a class of mathematical objects. These objects are often called structures. Model theory considers logics and their relation to structures. For example, one important question in model theory is to investigate which semantic properties of a class of structures are definable in a logic of interest. *Finite model theory* only considers finite structures, for example, finite graphs. Taking into account only finite structures makes a remarkable difference. Many tools and results from classical (or infinite) model theory fail in the restriction to finite structures (see, e.g., [61] for a discussion). Hence, researchers had to develop new tools or to change perspective. Because of the focus on finite structures, finite model theory is naturally relevant for computer science where the objects of interest are almost always finite. In fact, finite model theory has applications, for example, in database theory, artificial intelligence, or descriptive complexity theory. This thesis will focus on the latter. *Descriptive complexity theory* studies the relation between classical computational complexity classes and logics as follows: Computational complexity theory examines the complexity of decision problems. A decision problem is the task to decide whether a problem instance has a certain property or not, for example, to decide whether a graph is connected or not. Complexity of decision problems is studied with respect to time and space bounds of a deciding algorithm. Such a bound defines the class of decision problems decidable within that bound. The two probably most prominent complexity classes are PTIME and NPTIME, the classes of decision problems decidable in polynomial time (in the size of problem instance) by a deterministic, respectively nondeterministic, Turing machine.

From a logical point of view, one studies how difficult it is to describe such a class of problems. Every logic defines the class of decision problems that are definable within this logic, that is, there is a formula of the logic which is exactly satisfied by the positive instances of the problem. In this way, one can study the complexity of decision problems by asking in which logic they are definable or by asking how difficult it is to define them in a certain logic. Such a notion of difficulty can, for example, be the number of nested quantifiers or the number of variables that are required to define the problem. One major goal of descriptive complexity theory is to align these two notions of complexity, i.e., to find logical correspondences for the complexity classes from computational complexity theory based on Turing machines (see [38, 44, 72] for textbook references). Given a complexity class of interest, the goal is to find a logic which *captures* this class. This roughly means that exactly all the decision problems contained in the given complexity class are definable in this logic. The seminal result of descriptive complexity theory is Fagin's result [39] that existential second-order logic captures the complexity class NPTIME. This result is now known as Fagin's Theorem. Similarly, by putting restrictions on the possible quantifier alternations in second-order logic, one can capture every complexity class in the polynomial-time hierarchy. Full second-order logic captures the whole polynomial-time hierarchy. Extensions of second-order logic by appropriate transitive closure operators capture PSPACE [65] and EXPTIME [71]. Various results for complexity classes below NPTIME exist: There are rather natural logics capturing PTIME, LOGSPACE, and NLOGSPACE on *ordered structures*, that is, on relational structures for which one relation is a total order on their universe (e.g., a total order on the vertices of a graph). It is unknown whether these classes can be captured without the restriction to ordered structures.

The question whether logics capturing these three classes on all (unordered) structures exist are important open problems in finite model theory. Research regarding these question often proceeds as follows: Starting with a complexity class C and a logic L that is a candidate for capturing C, one studies whether L defines all properties of C or not. If not, L is separated from C. Such a separation provides insights into the limitations of L-expressiveness. To overcome this lack of expressiveness, L can be extended by some means. The extended logic thereby becomes a new candidate for capturing C. There may be different possibilities of extending L to define the L-undefinable properties yielding multiple candidates. These candidates can be compared to understand the expressiveness of the different extensions: Do the different extensions have the same expressiveness? Is one extension weaker (which, in particular, separates the weaker extension from C)? How expressive are combinations of different extensions? In the case that one is neither able to separate L from C nor to show that L captures C, one can restrict the structures that are considered. By considering larger classes of structures, one can on the one hand try to eventually approach all structures and on the other hand understand the difficulties to prove or disprove that L captures C. These approaches were carried out for different logics and for both LOGSPACE and PTIME, where the latter is arguable the more important class. In this thesis, we focus on capturing PTIME and extend the frontier of current research. We will investigate all current candidates for logics that may capture PTIME. All these candidates are based on different approaches and not much of their relationship is known. We will study the expressiveness of some of these candidates showing new definability results. In particular, we show that one candidate captures PTIME for a larger class of structures. We will combine two of the different, at first sight rather incompatible approaches and find out interesting properties of this combination. Last, we will separate another candidate from PTIME and thus rule it out as a candidate. All these results provide further insights into the logical structure of PTIME.

In the following section, we will firstly go into more details about the quest for a logic capturing PTIME, secondly review the three current candidates for a logic capturing PTIME, and thirdly discuss the contribution of this thesis.

1.1 The Quest for a Logic Capturing PTIME

Without the restriction to ordered structures, the question whether there is a logic capturing PTIME remains wide open. PTIME is, at least from a theoretical point of view, the accepted class of problems decidable by efficient algorithms. Thus, the question for a logic capturing PTIME is not only one of the central open questions in descriptive complexity theory [50], but it is also of relevance, for example, in database theory. Indeed, the question was first posed by Chandra and Harel [22] in the context of database theory: Is there an effective enumeration of all PTIME-properties, or to phrase it differently, *is there a natural and polynomial-time evaluable query language, i.e., a logic, expressing all polynomial-time decidable properties?* Gurevich [61] posed the question for a logic capturing PTIME and, in particular, gave precise requirements on what is now regarded as a reasonable logic capturing PTIME (see [50, 94] for a comparison of [22] and [61]). He conjectured that no such logic exists. On the one hand, proving this result would separate PTIME from NPTIME, which can be seen as another motivation for the quest of a logic for PTIME. On the other hand, a positive answer to the question would provide more insights into the structure of PTIME-properties and polynomial-time algorithms.

While proving the nonexistence of a PTIME-capturing logic should be hard (since it would separate PTIME from NPTIME), showing the existence of such a logic seems to be difficult too. At the core of the 40 years old problem lies a mismatch between algorithms and logics. When designing algorithms, one is used to making arbitrary choices. For example, consider a depth-first search in a graph: the choice of the next vertex to traverse is essentially an arbitrary choice. But how do deterministic algorithms actually make these choices? Conceptually, vertices of a graph cannot always be distinguished by intrinsic properties (e.g., consider a clique). The crucial point here is that the algorithm actually does not get the graph as input, but an ordered version of the graph, that is, a graph with a total order on the vertices. This order is unavoidable because of the need to write the graph on the tape of a Turing machine (or in the memory of a computer). However, this order is not unique. Permuting the vertices of a graph, that is essentially choosing a different order, may result in a different encoding. In general, it is not known whether a unique encoding can be computed in PTIME (which is actually the task of graph canonization and will be discussed later). Hence, the proof that the algorithm correctly decides a property implicitly includes a proof that the algorithm is order-invariant, which means that the algorithm outputs the same result for all possible orders. And, of course, there are algorithms which are not order-invariant. Moreover, it is even undecidable whether a given Turing machine is order-invariant.

For a logic, the situation changes. The formulas of a reasonable logic in the sense of Gurevich are enumerable. In particular, it is decidable whether a given string is a well-formed formula. Reasonable logics are required to be isomorphism-invariant, that is, two isomorphic graphs satisfy exactly the same sentences of the logic: Whether a sentence is satisfied by a graph or not only depends on the structure of the graph but not on the "names" of its vertices. In particular, a logic is invariant under permuting the vertices of a graph, i.e., choosing a different order. This means that logics, unlike algorithms, have to ensure isomorphism-invariance syntactically. As already mentioned, when equipping the input graph with a total order, there is a logic capturing PTIME: This logic is the extension of first-order logic by an inflationary fixed-point operator as shown in the Immerman-Vardi Theorem [69, 114]. Given such a total order, there is no need for arbitrary choices anymore and thus polynomial-time Turing machines can be simulated within this logic.

Fixed-Point Logics. First-order logic itself defines only local properties (see [38]). For example, the class of connected graphs is not definable in first-order logic. To support inductive definitions, extensions of first-order logic by inductive fixed-point operators were studied. Consider a formula Φ defining a k-ary relation $\Phi(R)$, that is, a set of k-tuples of vertices, in terms of a given k-ary relation R. An inductive fixed-point operator considers the series of k-ary relations defined by $R_0 = \emptyset$ and $R_{i+1} = \Phi(R_i)$. This sequence does not necessarily stabilize. In case that it stabilizes, a fixed-point of Φ is reached. There are two important approaches to guarantee that a fixed-point is reached: The first approach is the so-called least fixed-point operator. Using syntactic restrictions to the fixed-point-defining formula Φ, it is ensured that the formula is monotone, that is, for all k-ary relations $R \subseteq R'$, it holds that $\Phi(R) \subseteq \Phi(R')$. In this case, the former sequence is guaranteed to stabilize and the reached fixed-point is indeed the unique least fixed-point of Φ. The other approach is the already mentioned inflationary or inductive fixed-point operator. In this case, consider the series defined via $R_0 = \emptyset$ and $R_{i+1} = R_i \cup \Phi(R_i)$. Because all the R_i are k-ary relations, a fixed-point is reached at the latest when R_i contains all k-tuples of vertices. We call *fixed-point logic* IFP the extension of first-order logic with the inflationary fixed-point operator. First, it was shown that the extension by the inflationary and the one by the least fixed-point operator are equally expressive on finite structures [63]. It was later shown that their expressiveness is actually the same on all structures [81]. For both operators, the sequence of k-ary relations satisfies $R_i \subseteq R_{i+1}$. The fixed-point is reached within a polynomial number of steps because there are only polynomially many k-tuples for a fixed k. Thus, fixed-point logic can be evaluated in polynomial time.

There is also a third variant of defining fixed-points via the partial fixed-point operator. One considers the same series of relations as for least fixed-points but without ensuring that the fixed-point defining formula is monotone. If no fixed-point is reached, the operator evaluates to the empty relation. For partial fixed-points, the sequence of relations may need an exponential number of steps to stabilize (or to guarantee that it will never stabilize). The extension of first-order logic with a partial fixed-point operator can be evaluated in PSPACE, and on ordered structures indeed captures PSPACE [1, 114].

Concerning the quest for a logic capturing PTIME, consider again inflationary fixed-point logic (IFP). It defines a rich class of properties and, by the mentioned Immerman-Vardi Theorem, captures PTIME on ordered structures. Nevertheless, there are very simple non-IFP-definable properties, for example whether a graph has an even number of vertices, which obviously can easily be decided in PTIME. To deal with this lack of natural numbers in IFP, Immerman [70] proposed the *fixed-point logic with counting* (IFPC) extending IFP with numbers and a counting operator. IFPC easily defines whether a graph has an even number of vertices (cf. Figure 1.1). Actually, IFPC was a candidate for a logic capturing PTIME until Cai, Fürer, and Immerman [20] provided a clever graph construction separating IFPC from PTIME. The construction assigns to every base graph two non-isomorphic graphs: one is called even and the other one is called odd. Nowadays, the graphs are known as *CFI graphs*. The *CFI query* is to decide whether a given CFI graph is even. This query is decidable in PTIME, but is undefinable in IFPC. Although IFPC fails to capture PTIME, IFPC captures PTIME on many classes of graphs or structures including planar graphs [49], relational structures of bounded treewidth [56], interval graphs [82], graphs with excluded minors [51,52], which subsumes many previous results, and graphs of bounded rank width [57].

Capturing PTIME via Definable Canonization. The proofs of Fagin's Theorem and the Immerman-Vardi Theorem encode Turing machines into the corresponding logic. However, to encode Turing machines in IFP or IFPC, the availability of an order is crucial. It is not clear how such an encoding can be done without an order. To show that IFPC captures PTIME on a class of unordered graphs, the routinely employed way is the one via definable canonization (e.g., [52, 56, 57, 82]). In general, *canonization* is the task of assigning to a given graph (or relational structure in general) an isomorphic but totally ordered copy, the so-called *canon*, such that the canons of two isomorphic graphs are equal. If canonization for a class of graphs is definable in IFPC (or any other logic which is at least as expressive as IFP), then the Immerman-Vardi Theorem can be exploited to capture PTIME on that class because canons are totally ordered. So the problem of defining canonization is closely related to capturing PTIME. In general, it is not known whether graph canonization is (algorithmically) polynomial-time computable. Closely-linked to canonization is isomorphism testing, i.e., to decide whether two graphs are isomorphic. Graph isomorphism itself is not known to be in PTIME, either. However, graph isomorphism is decidable in quasi-polynomial time due to Babai [9], which makes graph isomorphism one of the rare candidates of NP-intermediate problems [74]. Every polynomial-time canonization algorithm implies a polynomial-time isomorphism test. The converse direction is unknown. However, for many graph classes with a polynomial-time isomorphism test, a polynomial-time canonization algorithm is known (see [109] for an overview). The situation is quite similar in logics: For many graph classes, for which IFPC defines the isomorphism problem, it is also known that IFPC defines canonization (e.g., [52, 57]), but a general reduction is unknown. We will address this issue again for logics that are more expressive than IFPC. For a graph class with a polynomial-time isomorphism test, isomorphism of graphs of the class is, by definition, a PTIME-query. Hence, a logic capturing PTIME has to define isomorphism for these graphs. Since the commonly used approach to capture PTIME is to define canonization, one actually would try to define canonization.

Let us for the moment accept that capturing PTIME is closely related to canonization, which itself is closely related to graph isomorphism. As already mentioned, IFPC fails to capture PTIME as shown by the CFI graphs. Therefore, research investigated logics more expressive than IFPC that potentially capture PTIME. Before discussing these logics, we first discuss some challenges for candidates of logics capturing PTIME. First, it is of particular interest whether the CFI query is definable. The CFI construction and variations of it are essentially the only known way to construct non-IFPC-definable PTIME-properties [20, 45, 64, 66, 68]. From another perspective, one can ask whether a logic captures PTIME on a graph class with a known polynomial-time canonization algorithm for which it is known that IFPC fails to capture PTIME. On these classes, at least graph isomorphism has to be definable. Probably, the simplest such graph class is the class of vertex-colored graphs of *q-bounded color class size*, i.e., at most q many vertices have the same color. In particular, CFI graphs of color class size 4 suffice to separate IFPC from Ptime. Graphs of bounded color class size can be canonized using group-theoretic techniques [8, 10, 41]. These techniques inherently rely on choosing generating sets of groups, so another example of algorithms making arbitrary choices. As a consequence, it is not clear how these group-theoretic techniques can be expressed in an isomorphism-invariant logic. Using algorithmic group theory turned out to be very fruitful in the context of graph isomorphism testing. It led to Luks' famous polynomial-time isomorphism test for graphs of bounded degree [90], which uses a more general and more complicated machinery than needed for bounded color classes. Hence, graphs of bounded color class size are one of the next reasonable classes on which we should try to capture Ptime on.

1.2 Approaches to Capture Ptime

In current research, there are three different approaches [103] to obtain stronger logics which potentially capture Ptime. These approaches come from very different directions and not much about their relation is known. This thesis will discuss all of them.

Choiceless Polynomial Time. The logic of *Choiceless Polynomial Time* (CPT) was introduced by Blass, Gurevich, and Shelah [18] to capture all choice-free computations on relational structures. Originally, CPT was presented as an abstract state machine programmable in a pseudo-code like syntax. Instead of receiving its input as a string (with the already discussed order-related encoding problems), it directly computes on a relational structure and on *hereditarily finite sets* constructed over its universe. The general idea is to eliminate choices by parallel computations for all possible choices. That is, whenever a hereditarily finite set is constructed, the machine can either process all of its contained elements in parallel or none of them. Picking one element arbitrarily out of a set is not possible. The logic CPT is obtained as the polynomial-time fragment of these machines by using explicit polynomial bounds. The absence of choices makes CPT choice-free and, in particular, isomorphism-invariant. Thus, CPT satisfies Gurevich's conditions for a logic that might capture Ptime. Akin to IFPC, CPT possesses a counting or cardinality operator that determines the size of a hereditarily finite set. It was already shown [18] that such an operator is needed to define the parity of the number of vertices of a graph.

It turned out that CPT has multiple equivalent definitions: Rossman [106] and Grädel and Grohe [43] provided more "logical" definitions based on fixed-point iterations on hereditarily finite sets. Later on, Grädel, Pakusa, Schalthöfer, and Kaiser [46] showed that CPT can also be characterized by iteratively applied first-order interpretations. Lastly, Grohe, Schweitzer, and Wiebking [59] proved CPT as expressive as a machine model called *Deep Weisfeiler Leman*, which is based on Turing machines modifying relational structures in an isomorphism-invariant manner. So CPT can be seen as a natural and robust way to capture the concept of choice-free computations.

In some sense, CPT overcomes the restriction of first-order logic and its extensions with fixed-point operators to define relations of fixed arity. For example, CPT is able to define tuples whose lengths are not bounded by a constant as long as at most polynomially many such tuples are constructed. Thus, CPT captures PTIME on padded structures [19], that is, on arbitrary structures extended by large cliques (see also [82] for an improved bound on the needed size of the clique). While adding large cliques does not change the expressiveness of IFPC, in CPT one can consider all possible orders of the small interesting part of the input to exploit the Immerman-Vardi Theorem. Because the clique is sufficiently large, the number of considered orders is still polynomial in the size of the input. In particular, CPT is strictly more expressive than IFPC.

It was shown by Dawar, Richerby, and Rossman [34] that CPT defines the CFI query for CFI graphs obtained from ordered base graphs. Ordered base graphs are precisely the base graphs of color class size one. In particular, this restricted form of the CFI query includes the color class size four CFI graphs, which suffice to separate IFPC from PTIME. The construction uses deeply nested sets to determine the parity of CFI graphs. This technique was generalized by Pakusa, Schalthöfer, and Selman [104] to base graphs of logarithmic color class size and to base graphs whose maximal degree is linear in the number of vertices. However, this technique to define the CFI query can probably not be used to define the unrestricted CFI query [100]. So it remains an open problem whether CPT defines the unrestricted CFI query. Indeed, the unrestricted CFI query is a candidate for separating CPT from PTIME [102]. Moreover, Rossman [106] showed that there is a polynomial-time computable *function* which is not CPT-definable. This however does not separate CPT from PTIME because PTIME only considers polynomial-time decidable *properties*. Another possibility to separate CPT from PTIME was recently established by Pago [102] who shows that CPT has the same power to distinguish graphs as the bounded-degree extended polynomial calculus. However, no sufficient strong bounds are known for that calculus.

Regarding canonization and capturing PTIME in CPT, Abu Zaid, Grädel, Grohe, and Pakusa [118] made a first step towards canonizing relational structures of bounded color class size. The authors only consider structures whose color classes are of bounded size and additionally abelian: For every color class, the automorphism group of the substructure induced by this color class is abelian. This restriction is used to replace the group-theoretic algorithms in the canonization algorithm for structures of bounded color class size by checking a certain class of linear equation systems over finite rings for solvability. The technique to check these systems for solvability is based on the techniques of [34] and [104] used to define the CFI query.

Algebraic Operators. As noted before, certain problems of linear algebra, such as solving linear equation systems for example, became important for capturing PTIME. Solving linear equations over finite fields is polynomial-time computable by Gaussian elimination for example, which however also depends on making choices (choosing the row and column of the coefficient matrix to process next). In particular, the CFI graphs are a graph encoding of a certain class of linear equation systems over the two element field $\mathbb{F}_2$. Indeed, the CFI query reduces to solving a system of linear equation systems over $\mathbb{F}_2$ [27]. Hence, the definability or undefinability of problems of linear algebra are of interest. It was observed by Rossman [17] that the determinant of a matrix over a commutative ring of characteristic 0 is CPT-definable. This was extended by Blass and Gurevich to finite fields [17]. Later, it was shown that the determinant over finite fields is indeed IFPC-definable [27]. The authors also prove that the rank of a matrix over $\mathbb{Q}$ is IFPC-definable. However, the CFI graphs show that this cannot be the case for the rank over $\mathbb{F}_2$. It was shown by Atserias, Bulatov, and Dawar [6] that infinitary finite-variable counting logic, which subsumes IFPC, fails to define the solvability of equation systems over every *fixed* abelian group and, in particular, of linear equation systems over finite fields. This poses the question of descriptive complexity of linear algebra [68, 82, 103].

Hence, considering extensions of IFPC by operators for linear algebra seems natural. There are different possibilities for such extensions, for example, by generalized Lindström quantifiers for the solvability of linear equation system [24, 103]. Of particular interest was *rank logic* introduced by Dawar, Grohe, Holm, and Laubner [27]. Rank logic extends IFPC by a rank operator rk_p for every finite prime field $\mathbb{F}_p$. The rk_p-operator evaluates to the rank of a definable matrix over $\mathbb{F}_p$. The rank operator is quite expressive. It subsumes the counting operator, can be used to express graph connectivity (in the absence of the fixed-point operator), and defines the CFI query. The latter is not surprising because the CFI query reduces to solving linear equations systems over $\mathbb{F}_2$, which reduces to computing ranks over $\mathbb{F}_2$. However, it turned out that using a different rank operator rk_p for every finite prime field is too restrictive. The CFI construction can be generalized from $\mathbb{F}_2$ to every finite prime field $\mathbb{F}_p$ and even to general abelian groups [13, 45, 95]. Using CFI graphs over $\mathbb{F}_p$, Grädel and Pakusa [45] showed that the CFI query over $\mathbb{F}_q$ cannot be defined by only using rk_p-operators for $p \neq q$. So, when considering the class of CFI graphs over all finite prime fields, this version of rank logic fails to define the CFI query because a finite formula can only contain finitely many rk_p-operators. The authors proposed an alternative *uniform* rank operator rk, where the characteristic of the prime field is defined by a term of the logic, so the characteristic depends on the input structure. Uniform rank logic defines the CFI query over all prime fields. Additionally, it was shown by Pakusa [103] that rank logic captures PTIME on relational structures with 2-bounded color classes.

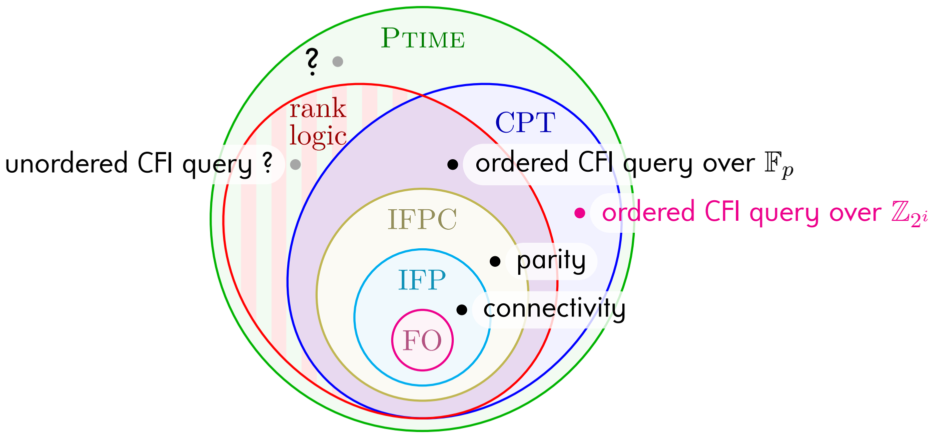
Ehrenfeucht-Fraïssé-like pebble games, also known as back-and-forth games, are a powerful tool for showing that two structures can be distinguished (or cannot be distinguished, respectively) by a logic. Such game-based characterizations exist for finite-variable first-order logic and for its extension with counting (see [98]). This fruitful correspondence can, in some sense, also be established for rank logic. Dawar and Holm [29] provided a game characterization of finite-variable first-order logic extended by a rank quantifier. This game is based on matrix equivalence (which implies equal ranks) and was generalized by the authors to simultaneous similarity of sequences of matrices, which is called the *invertible-map game*. The invertible-map game has the benefit that its induced equivalence relation on finite structures is polynomial-time decidable, which for the version with ranks is unknown. As shown by Dawar, Grädel, and Pakusa [25], the invertible-map game precisely characterizes the so-called *linear-algebraic logic*. This logic is an infinitary logic with a Lindström quantifier for every linear-algebraic operator, that is, for every mapping of a sequence of matrices over a prime field to some number, which is invariant under isomorphisms of the underlying vector space. Hence, linear-algebraic logic is a very expressive logic. It is also non-effective because it is infinitary and not restricted to computable linear-algebraic operators. The relevance of linear-algebraic logic stems from the fact that it encompasses every extension of first-order or fixed-point logics by any linear-algebraic operators over finite fields. It was shown [25] that, akin to rank logic, linear-algebraic operators for every prime field are needed to define the CFI query over all prime fields in linear-algebraic logic. This implies that to capture PTIME with some extension of IFPC by linear-algebraic operators over finite fields, one always has to consider all finite fields.

Symmetric Choice. The last approach to capture PTIME is to directly allow for choices in the logic. In a first step, Arvind and Biswas [5] extended IFP by a nondeterministic choice in the fixed-point operator. Such an inflationary fixed-point operator defines a relation R as follows: At every step, not all tuples satisfying the fixed-point-defining formula are added to R, but a single tuple not already contained in R is nondeterministically chosen (if such a tuple exists) and only this tuple is added to R. However, this unrestricted form of choice breaks isomorphism-invariance. The authors showed that the fragment of the logic evaluating equally for all choices indeed captures PTIME. But this fragment is not a reasonable logic in the sense of Gurevich: It is undecidable whether such a formula defines a choice-invariant property, that is, the logic has no decidable syntax.

Gire and Hoang [42] introduced the concept of *symmetric choice* to syntactically guarantee isomorphism-invariance. The authors studied the logic IFP+SC by extending IFP as follows: An inflationary fixed-point operator with symmetric choices uses two formulas Φ and Ψ to define a fixed-point. The formula Φ defines a fixed-point iteration. At every stage in the iteration, the formula Ψ defines a *choice-set* of tuples, from which one tuple is chosen. This tuple is passed to Φ to define the next stage in the fixed-point computation. The important restriction is that choice-sets always have to be *orbits*, that is, for every pair of tuples a and b in the choice-set, there must be an automorphism of the input structure mapping a to b. With this restriction, the defined fixed-point is *semideterministic* (in the sense of [36]): All possible fixed-points are related by an automorphism. Hence, all fixed-points satisfy the same IFP-formulas (or, in general, formulas of any other isomorphism-invariant logic). This is used to guarantee isomorphism-invariance: Every fixed-point operator with symmetric choice provides a formula, which is evaluated on the defined fixed-point. So a fixed-point operator with symmetric choice actually only defines a truth-value and thus is deterministic (because truth-values are trivially isomorphism-invariant). While IFP+SC is a reasonable logic in the sense of Gurevich, it is unknown whether IFP+SC can be evaluated in PTIME. To evaluate IFP+SC, one needs to compute orbits to check that all choice-sets are indeed orbits. However, computing orbits is polynomial-time equivalent to deciding graph isomorphism for which the complexity status is unknown.

Gire and Hoang [42] resolved this issue by replacing symmetric choice with *witnessed symmetric choice* (which is called specified symmetric choice in [42]) resulting in the logic IFP+WSC. A WSC-fixed-point operator consists of another formula, which has to define witnessing automorphisms proving that the choice-sets are indeed orbits. That is, the obligation to check that choice-sets are orbits is moved from the evaluation to the formulas themselves. In this way, IFP+WSC can be evaluated in polynomial-time. IFP+WSC is strictly more expressive than IFP because it defines the CFI query for ordered base graphs [42]. Defining the CFI query for ordered base graphs in IFP+WSC is comparatively easier than defining it in CPT [34].

Afterwards, IFP+SC was studied by Dawar and Richerby [31]. The authors allowed for nested symmetric choice operators and proved that parameters of choice operators


PTIME
?
rank logic
CPT
unordered CFI query ?
ordered CFI query over $\mathbb{F}_p$
ordered CFI query over $\mathbb{Z}_{2^i}$
IFPC
parity
IFP
connectivity
FO

1.1 Relations between various extensions of fixed-point logic and CPT. The figure shows the inclusions between first-order logic FO, fixed-point logic IFP, fixed-point logic with counting IFPC, rank logic, and Choiceless Polynomial Time CPT. It is unknown whether rank logic is contained in CPT or not. Vertices show queries separating the logics. Gray vertices indicate candidates for separating queries. The pink vertex is a new result of this thesis.

increase the expressiveness. They also showed that nested symmetric choice operators are more expressive than a single one and conjectured that the expressiveness increases further with additional nesting depth. We remark that when dropping the restriction of the choice-sets to be orbits, then the approaches of [5] and [42], albeit different, have the same expressiveness [32].

One severe restriction of symmetric choice is that orbits need to be definable. If orbits are not definable, then symmetric choice is not beneficial. However, it might be the case that the input structure is reducible to an easier structure, for which orbits (or witnessing automorphisms) can be defined. The logical concept of a reduction is an interpretation. Gire and Hoang [42] considered the closure of IFP+WSC under an operator for interpretations. This operator evaluates a formula in the image of an interpretation and this image possibly has different orbits. The resulting logic IFP+WSC+I simulates counting, that is, IFP+WSC+I subsumes IFPC, which IFP+WSC is indicated to fail to do [42]. This leaves IFP+WSC+I as a candidate of a logic capturing PTIME, but Dawar and Richerby conjectured that this is not the case [31].

1.3 Contribution

This thesis provides new results for all the three approaches mentioned before. We will make progress in the questions whether CPT captures PTIME or not, whether PTIME can be captured by extending IFPC with operators from linear algebra, and how these extensions relate to CPT (cf. Figure 1.1). We will study how witnessed symmetric choice relates to CPT and whether witnessed symmetric choice is needed in the presence of counting (so in IFPC and not in IFP) to get a better understanding of its power. However, the first result considers 3-variable counting logic.

Quantifier Depth of 3-Variable Counting Logic. Chapter 3 considers 3-variable counting logic, the 3-variable fragment of first-order logic extended by a counting quantifier. Finite-variable counting logics play an important role in descriptive complexity theory [48]. In particular, IFPC can be embedded into infinitary counting logic [98]. Because the model-theoretic technique of Ehrenfeucht-Fraïssé-like pebbles games can be applied to finite-variable counting logics [20,66], these games can be used to prove that IFPC distinguishes (or fails to distinguish) structures [20, 57, 76]. Apart from that, finite-variable counting logics have a connection to machine learning [53,93,110,117], the Sherali-Adams hierarchy of a natural linear integer programming formulation of graph isomorphism [7,58], and to homomorphism counts from graphs of bounded treewidth [35].

Determining the quantifier depth of a formula of k-variable counting logic that is maximally required to distinguish n-element structures is an open problem. For arbitrary k, no better upper bound than the trivial one of $n^k - 1$ is known. For 3-variable counting logic, the best known upper bound is $\mathcal{O}(n^2/\log n)$ by Kiefer and Schweitzer [79]. We will prove an $\mathcal{O}(n \log n)$ upper bound for 3-variable counting logic, which matches the best known lower bound of $\Omega(n)$ by Fürer [40] up to a logarithmic factor. Recently, the upper bound was generalized in joint work with Martin Grohe and Daniel Neuen to k-variable counting logic yielding an $\mathcal{O}(n^{k-1} \log n)$ upper bound [57].

We will exploit the correspondence between $(k+1)$-variable counting logic and the k-dimensional Weisfeiler-Leman (k-WL) algorithm [20,98]. For the case $k = 2$, the 2-WL algorithm computes a coloring of all vertex pairs of a graph by iteratively applying the 2-WL refinement. The 2-WL refinement colors a vertex pair (u, v) based on the colors of all walks of length 2 from u to v. Either the partition into color classes is properly refined or the process stabilizes. The iteration number of the 2-WL refinement needed to stabilize corresponds to the maximal quantifier depth of a formula of 3-variable counting logic to distinguish vertex pairs in the graph.

To prove the $\mathcal{O}(n \log n)$ upper bound, we take an algebraic point of view. We exploit the one-to-one correspondence between coherent configurations, so in particular, the coloring output by 2-WL, and coherent algebras [67]. We introduce another refinement called the walk-refinement that refines the color of a vertex pair (u, v) by the multiset of colors of all walks (of unrestricted length) from u to v. One iteration of walk-refinement has the same distinguishing power as a certain matrix algebra induced by the partition into color classes. Whenever the walk-refinement strictly refines the colors, we obtain a strictly larger algebra. Using arguments from representation theory, we prove that every proper chain of these algebras has length at most $\mathcal{O}(n)$. Hence, walk-refinement stabilizes after $\mathcal{O}(n)$ iterations. We actually prove that this bound is tight using the same graphs as [40]. We obtain the $\mathcal{O}(n \log n)$ bound on the iteration number of 2-WL by showing that a single step of walk-refinement is subsumed by logarithmically many iterations of 2-WL refinement. In particular, both refinements produce the same stable coloring.

The results in this chapter are joint work with Pascal Schweitzer and Ilia Ponomarenko and appeared at LICS 2019 [87].

Canonizing Structures with Dihedral Colors in CPT. In Chapter 4, we take the next step towards canonizing structures with bounded color class size in CPT. Since there is a CPT-definable canonization for structures with bounded and abelian colors [118], we consider structures with bounded and dihedral or cyclic colors. A color class is dihedral (or cyclic, respectively) if it induces a substructure whose automorphism group is a dihedral (or cyclic, respectively) group. A dihedral group is the automorphism group of a regular n-gon consisting of rotations and reflections. We call the group *odd* if n is odd. In particular, dihedral groups are non-abelian for $n > 2$. We show that the following classes of structures have a CPT-definable canonization:

(a) q-bounded ternary relational structures with odd dihedral or cyclic colors and

(b) q-bounded binary relational structures with dihedral or cyclic colors.

We thereby provide the first CPT-definable canonization for a class of q-bounded structures with non-abelian color classes. In particular, we show that CPT captures PTIME on such classes.

Our approach consists of two steps. As a first step, we propose a normal form for arbitrary finite q-bounded structures. Then, in a second step, we use group-theoretic arguments to canonize structures with dihedral colors given in the aforementioned normal form. Concretely, the first step is a reduction transforming the input structure into a normal form, which ensures that a color class and its adjacent color classes form a “rigid assemblage”. That is, locally the automorphism groups form 2-injective 3-factor subdirect products or they are quotient groups of other color classes. In the case of 2-injective 3-factor subdirect products, the automorphisms of three adjacent color classes are not independent of each other. This means that no nontrivial automorphism of the substructure induced by these three color classes is the identity on two of them. This normal form is not specific to dihedral colors and is possibly of more general interest.

It was not necessary to consider 2-injective groups for abelian colors yet, but it is for non-abelian colors. Towards a reduction step, a purely group-theoretic analysis of 2-injective groups is given by Neuen and Schweitzer [97]. The main insight is basically that such groups decompose naturally into structurally simpler parts which are related via a common abelian normal subgroup. We extend the techniques to canonize abelian color classes and show how they can be combined with the analysis of 2-injective groups to obtain a canonization procedure for said structures with dihedral colors in CPT. That is, we provide new methods to integrate group-theoretic reasoning, which is at the core of canonizing q-bounded structures algorithmically, into logics.

The results in this chapter are based on joint work with Pascal Schweitzer and appeared at CSL 2021 [88].

CPT, Witnessed Symmetric Choice, and Definable Isomorphism. We consider CPT-definable isomorphism tests and how these can be used to capture PTIME. Progress in the quest for a logic capturing PTIME usually comes in one of two flavors, which we have already discussed: Research results either show that some logic captures PTIME for a more extensive class of structures, or a logic is separated from PTIME. In Chapter 5, we take a different point of view. The ultimate goal is to provide a reduction from a definable canonization of a class of structures to a definable isomorphism test for the same class. Such a reduction is not known in general and defining canonization often appears to require considerably more effort than defining isomorphism (e.g., [51, 52, 57]).

We extend CPT with a suitable operator for witnessed symmetric choice (CPT+WSC). For this extended logic, we show that a CPT+WSC-definable isomorphism test implies a CPT+WSC-definable canonization. Thus, we reduced capturing PTIME in CPT+WSC to defining isomorphism. In that sense, our result can be seen as a simplification to capture PTIME in CPT+WSC.

So why should witnessed symmetric choice in CPT suffice to show that isomorphism testing and canonization are equivalent? Here we build on two existing results. The first one [59] shows that a CPT-definable isomorphism test implies a CPT-definable complete invariant, that is, a CPT-definable ordered object which is equal for two input structures if and only if they are isomorphic. The second, more classical result is due to Gurevich [62]. It shows how an algorithm computing complete invariants can be turned into an algorithm computing a canonization. This algorithm requires that the class of graphs is closed under individualization (that is, under coloring individual vertices). Although being closed under individualization is a problem in some contexts [80, Theorem 33], this requirement is usually not a restriction [80, 91]. The canonization algorithm repeatedly uses the complete invariant to compute a canonical orbit, chooses and individualizes one vertex in that orbit, and proceeds until all vertices are individualized. Thereby, a total order on the vertices is defined (which is semideterministic). This algorithm can be expressed in CPT+WSC. Hence, a definable complete invariant can be turned into a definable canonization.

So far, a CPT-definable isomorphism test implies a CPT+WSC-definable canonization. However, we prefer that a definable isomorphism test implies a canonization in the same logic. To show this statement for CPT+WSC, we lift the result that definable isomorphism implies a definable complete invariant from CPT to CPT+WSC. That is, we show that a CPT+WSC-definable isomorphism test implies a CPT+WSC-definable complete invariant and hence a CPT+WSC-definable canonization. This step is based on the characterization of CPT via the Deep Weisfeiler-Leman computation model [59], which we extend with witnessed symmetric choice.

The results in this chapter are based on joint work with Pascal Schweitzer and appeared at LICS 2022 [89].

IFPC, Witnessed Symmetric Choice, and Logical Interpretations. We have already discussed that witnessed symmetric choice increases the expressiveness of IFP and that it enables us in CPT to prove that definable isomorphism is equivalent to definable canonization. In Chapter 6, we want to understand the expressive power of witnessed symmetric choice and its interplay with the operator for logical reductions in general and not specifically for IFP. Most of the existing results showing that (witnessed or unwitnessed) symmetric choice or the interpretation operator increase the expressiveness of IFP in some way are based on simulating counting [31, 42]. However, counting is not the actual reason for using witnessed symmetric choice. Counting can be achieved more naturally in IFPC. Thus, it is unknown whether the interpretation operator increases expressiveness of IFPC+WSC. In CPT, it is not possible to show that witnessed symmetric choice or the interpretation operator increase expressiveness without separating CPT from PTIME. This question itself is still open.

This turns IFPC into a natural base logic for studying the interplay of witnessed symmetric choice and the interpretation operator. On the one hand, separation results based on counting are not applicable in IFPC. On the other hand, there are known IFPC-undefinable PTIME properties, namely the already mentioned CFI query, which can be used to separate extensions of IFPC. We define the logics IFPC+WSC and IFPC+WSC+I extending IFPC with witnessed symmetric choice and the latter additionally with the interpretation operator. We show that the interpretation operator increases expressiveness, that is, we separate IFPC+WSC from IFPC+WSC+I. In particular, we separate IFPC+WSC from PTIME. This separation is established as follows: We show, in some sense similar to CPT+WSC using Gurevich's canonization algorithm, that if IFPC+WSC+I distinguishes orbits, i.e., the orbits can be defined and ordered, then IFPC+WSC+I defines a canonization. We apply this approach to CFI graphs: We show that if IFPC+WSC+I canonizes a class of colored base graphs closed under individualization, then IFPC+WSC+I canonizes the CFI graphs over that class of base graphs. The conclusion is that for IFPC+WSC+I a class of CFI graphs is not more difficult than the corresponding class of base graphs, which is different in many other logics. However, to canonize the CFI graphs, the nesting depth of WSC-fixed-point operators and interpretation operators increases. We prove that this increase is unavoidable: Two nested WSC-fixed-point and interpretation operators are more expressive than a single one.

Finally, to prove that the interpretation operator indeed increases expressiveness, we construct a PTIME-property definable in IFPC+WSC+I but not in IFPC+WSC. This property is a variant of the CFI query that combines CFI graphs with the multipede construction of Gurevich and Shelah [64]. These multipedes are asymmetric structures, i.e., structures without nontrivial automorphisms, for which isomorphism is not IFPC-definable. We show that for a suited combination of CFI graphs with multipedes, we obtain asymmetric structures for which orbits are not IFPC+WSC-definable. Hence, witnessed symmetric choice is not beneficial on these structures, and so IFPC+WSC is not more expressive than IFPC on these structures. In particular, IFPC+WSC does not define their isomorphism problem. However, the isomorphism problem of said structures can be reduced via an interpretation to the CFI query, which is IFPC+WSC+I definable.

In this way, we separate IFPC+WSC from PTIME. Such a result does not follow from existing ones because separating IFP+WSC from PTIME is based on counting in [42]. Moreover, we show that IFPC+WSC is not even closed under FO-interpretations. The same construction shows that IFP+SC is not closed under FO-interpretations. Thereby, we answer an open question of Dawar and Richerby [31].

The results of this chapter appeared at ICALP 2023 [86].

Separating Rank Logic from PTIME. In Chapter 7, we separate rank logic from PTIME. To do so, we consider the CFI query over all rings $\mathbb{Z}_{2^i}$. The result of Grädel and Pakusa [45], which shows that the CFI query over $\mathbb{F}_p$ is not definable using rank operators over $\mathbb{F}_q$ for all $p \neq q$, can be extended to the rings $\mathbb{Z}_{2^i}$. It suffices to consider the rank operator over $\mathbb{F}_2$. We then show that the characteristic 2 invertible-map game fails to distinguish all non-isomorphic CFI graphs over $\mathbb{Z}_{2^i}$. This implies that rank logic fails to define the CFI query over $\mathbb{Z}_{2^i}$. We thereby rule out rank logic as a candidate for a logic capturing PTIME. Because isomorphism of CFI graphs over $\mathbb{Z}_{2^i}$ reduces to solving linear equation systems over $\mathbb{Z}_{2^i}$, this answers the question of solvability of linear equation systems over finite rings in rank logic [6], at least for the case that the ring is part of the input, in the negative. Actually, we even consider CFI graphs over ordered base graphs. From a variant of the canonization for structures with bounded and abelian color classes in CPT [103] it follows that CPT defines the CFI query for ordered base graphs over $\mathbb{Z}_{2^i}$. Hence, we showed that rank logic does not even capture CPT (cf. Figure 1.1).

Recall that the power of the invertible-map game to distinguish graphs is actually captured by the more general linear-algebraic logic [29]. Dawar, Grädel, and Pakusa [25] generalized the result of the characteristic of rank operators and the CFI query from rank logic to linear-algebraic logic: The CFI query over $\mathbb{F}_p$ cannot be defined in linear-algebraic logic using linear-algebraic operator over fields $\mathbb{F}_q$ for all $q \neq p$. This result generalizes to the rings $\mathbb{Z}_{2^i}$: The CFI query can only be defined, if possible at all, using linear-algebraic operators over $\mathbb{F}_2$. Together with our result that the invertible-map game over $\mathbb{F}_2$ fails to distinguish all non-isomorphic CFI graphs over all rings $\mathbb{Z}_{2^i}$, we obtain that linear-algebraic logic fails to define the CFI query over $\mathbb{Z}_{2^i}$. This finally settles the question of capturing PTIME by extending IFPC with operators of linear algebra over finite fields: Linear algebra over finite fields is too weak to capture PTIME.

An extended abstract of the results in this chapter appeared at LICS 2021 [84]. The full results appeared in the Journal of the ACM [85]. Combining the results for rank logic with linear-algebraic logic is joint work with Anuj Dawar and Erich Grädel and appeared in the Journal of Logic and Computation [28].

Structure of this Thesis. This thesis is organized as follows: In the preliminary Chapter 2, we fix our notation and provide the necessary background on finite model theory and, in particular, on descriptive complexity theory. We start with the $\mathcal{O}(n \log n)$ upper bound on the quantifier depth of 3-variable counting logic and the iteration number of 2-WL in Chapter 3. We then turn to CPT in Chapter 4 and show that canonization of structures with dihedral color classes is CPT-definable. Afterwards, Chapter 5 considers the extension of CPT with witnessed symmetric choice and proves definable isomorphism and definable canonization to be equivalent for CPT+WSC. In the next Chapter 6, we investigate the relation of counting, witnessed symmetric choice, and the interpretation operator in IFPC. Finally, rank logic is separated from PTIME in Chapter 7. We draw a conclusion in Chapter 8.

Every chapter provides a short introduction to the topic and discusses further background and related work. To some extent, these introductions repeat some material of this introductory chapter to be more self-contained. Additional more specific preliminaries, if needed, will be discussed in the individual chapters where they are relevant. Every chapter ends with a short discussion of its results.

Chapter 2

Preliminaries

In this chapter, we fix our notation and provide necessary background on finite model theory and descriptive complexity theory, which we need throughout this thesis. We assume that the reader is familiar with basic notions from finite model theory.

- Section 2.1 introduces notation and basic notions for finite relational structures and in particular for graphs.
- Section 2.2 reviews the concept of a reasonable logic by Gurevich and what it means that a logic captures PTIME. It also provides related notions for logics in general.
- Section 2.3 considers extensions of first-order logic: inflationary fixed-point logic with and without counting (IFPC and IFP), finite variable counting logics $\mathcal{C}_k$, and their characterization via the bijective pebble game.
- Section 2.4 reviews the logic of Choiceless Polynomial Time (CPT).
- Section 2.5 introduces logical interpretations and focuses on IFPC-interpretations and CPT-interpretations.
- Section 2.6 uses interpretations to consider the concept of definable canonizations.
- Section 2.7 introduces the 2-dimensional Weisfeiler-Leman refinement and its connection to 3-variable counting logic $\mathcal{C}_3$.
- Section 2.8 discusses the CFI construction and compares different variants of it. Although the CFI construction is in some sense specific, it will play an important role throughout this thesis and hence is presented in this chapter.

For more background material, we refer to textbooks [38,44,52,72,98]. Notation or further background only needed in individual chapters will be introduced at the beginning of the corresponding chapters.

We write $\mathbb{N}$ for the natural numbers, $\mathbb{Z}$ for the integers, and $\mathbb{F}_p$ for the finite prime field of order p. For a positive $j \in \mathbb{N}$, we denote by $\mathbb{Z}_j$ the ring of integers modulo j. Its elements are $\{0, \dots, j-1\}$. We denote by $[k]$ the set $\{1, \dots, k\}$ for every $k \in \mathbb{N}$. The disjoint union of two sets N and M is $N \uplus M$. We denote by $\{\{a_1, \dots, a_k\}\}$ the multiset containing the elements $a_1, \dots, a_k$.

Let N and I be finite sets. The set of I-indexed **tuples** over N is denoted by N^I. For a tuple $\bar{t} \in N^I$, the entry for index $i \in I$ is written as $\bar{t}(i)$. The **restriction** of $\bar{t} \in N^I$

to $J \subseteq I$ is denoted by $\bar{t}|_J \in N^J$ and satisfies $\bar{t}|_J(i) = \bar{t}(i)$ for all $i \in J$. In the case that $J \not\subseteq I$, we write $\bar{t}|_J$ as an abbreviation for $\bar{t}|_{J \cap I}$. We extend the notation to sets of tuples $T \subseteq M^I$:

$$T|_J := \left\{ \bar{t}|_J \;\middle|\; \bar{t} \in T \right\}.$$

For a set of tuples $T \subseteq N^J$, the **extension** of T to I is

$$T|^I := \left\{ \bar{t} \in N^I \;\middle|\; \bar{t}|_J \in T \right\}.$$

For two finite index sets I and J, an $I \times J$ **matrix** M over N is a mapping $M \colon I \times J \to N$. We write $M(i,j)$ for the entry at position (i,j). If $0, 1 \in N$, then $\mathbb{1}_I$ denotes the $I \times I$ identity matrix and $\mathbb{0}_I$ denotes the $I \times I$ zero matrix. We omit the subscript if I is clear from the context. If binary addition $+$ is defined on N, we write $\sum \bar{t} := \sum_{i \in I} \bar{t}(i)$ for a tuple $\bar{t} \in N^I$ and likewise $\sum f := \sum_{i \in I} f(i)$ for a function $f \colon I \to N$.

For $k \in \mathbb{N}$, a tuple $\bar{t} \in N^k = N^{[k]}$, and $i \leq k$, we also write t_i for the i-th entry. The length of $\bar{t}$ is $|\bar{t}| = k$. The **concatenation** of two tuples $\bar{s} \in N^k$ and $\bar{t} \in N^\ell$ is denoted by $\bar{s}\bar{t} \in N^{k+\ell}$. The set of all tuples over N of length at most k is $N^{\leq k}$ and the set of all tuples of finite length is N^*. An $m \times n$ matrix over N is an $[m] \times [n]$ matrix.

2.1 Relational Structures

A **relational signature** is a set of relation symbols $\{R_1, \dots, R_\ell\}$ with associated **arities** $\mathrm{ar}(R_i)$ for every $i \in [\ell]$. We use letters τ and σ for signatures. Let $\tau = \{R_1, \dots, R_\ell\}$ be a signature. A **τ-structure** is a tuple

$$\mathfrak{A} = \left(A, R_1^{\mathfrak{A}}, \dots, R_\ell^{\mathfrak{A}}\right)$$

where $R_i^{\mathfrak{A}} \subseteq A^{\mathrm{ar}(R_i)}$ for every $i \in [\ell]$. The set A is called the **universe** of $\mathfrak{A}$ and its elements **atoms**. We use fraktur letters $\mathfrak{A}$ and $\mathfrak{B}$ for relational structures and denote their universes always by A and B. For atoms, we use the letters u, v, and w. The **order** of $\mathfrak{A}$ is the size $|A|$ of its universe. The **arity** of $\mathfrak{A}$ is the maximal arity of its relations. For $\sigma \subseteq \tau$, the **reduct** $\mathfrak{A} \upharpoonright \sigma$ is the restriction of $\mathfrak{A}$ to the relations contained in σ. For a subset $A' \subseteq A$, we denote by $\mathfrak{A}[A']$ the **induced substructure** of $\mathfrak{A}$ by A', i.e., the τ-structure with universe A' and relations

$$R^{\mathfrak{A}[A']} := R^{\mathfrak{A}} \cap (A')^{\mathrm{ar}(R)}$$

for every $R \in \tau$. The notation is also used for a pair of a structure and atoms. For a tuple $\bar{u} \in A^*$, we set $(\mathfrak{A}, \bar{u})[A'] := (\mathfrak{A}[A'], \bar{u}')$ for the tuple $\bar{u}' \in (A')^*$ obtained from $\bar{u}$ by deleting all entries not in A'. We sometimes view a tuple $\bar{u} \in A^*$ as a set and write $\mathfrak{A}[\bar{u}]$ for $\mathfrak{A}[\{u_i \mid i \in [|\bar{u}|]\}]$.

A relational structure $\mathfrak{A}$ is **ordered** if one of its relations is a total order on A. A **total preorder** $\preceq$ on a set M is a reflexive, transitive, and total binary relation. It induces the equivalence relation $u \sim v$ if and only if $u \preceq v$ and $v \preceq u$ for every $u, v \in M$. A relational structure $\mathfrak{A}$ is **colored** if one of its relations is a total preorder $\preceq^{\mathfrak{A}}$ on A. The $\preceq$-equivalence classes are called the **color classes**. The color classes of $\mathfrak{A}$ are ordered by $\preceq$. The structure $\mathfrak{A}$ has **q-bounded colors** or is **q-bounded** if every color class of $\mathfrak{A}$ has size at most q. In that sense, an ordered structure is a structure with 1-bounded colors.

In this thesis, we almost always consider finite relational structures and thus also just call them structures. We will point out when structures are infinite or not relational.

Graphs. A **graph** is a binary $\{E\}$-structure and a **colored graph** is a binary colored $\{E, \preceq\}$-structure. The atoms of a graph are usually called **vertices**. Let $G = (V^G, E^G)$ be a graph. We usually just write $G = (V, E)$ for graphs. As a binary structure, the edge relation is always directed. We call G **undirected** if, for every $u, v \in V$, we have $(u, v) \in E$ if and only if $(v, u) \in E$ and write $\{u, v\} \in E$. A **simple** graph is an undirected graph without loops. For a set $V' \subseteq V$, we denote by $G - V'$ the graph obtained from G by removing the vertices in V'. For a set of edges $E' \subseteq E$, we denote by $G - E$ the graph obtained from G by removing the edges in E'. For two vertices $u, v \in V$, we denote their **distance** in G by $\mathsf{dist}_G(u, v)$. For two sets $X, Y \subseteq V$ and a vertex $u \in V$, we set

$$\begin{aligned}\mathsf{dist}_G(X, Y) &:= \min_{u \in X, v \in Y} \mathsf{dist}_G(u, v),\\ \mathsf{dist}_G(u, Y) &:= \mathsf{dist}_G(\{u\}, Y).\end{aligned}$$

The set of **neighbors** of a vertex $u \in V$ is denoted by $N_G(u)$. The **k-neighborhood** of u is

$$N_G^k(u) := \left\{ v \in V \;\middle|\; \mathsf{dist}_G(u, v) \leq k \right\}.$$

The subgraph of G **induced by** $W \subseteq V$ is $G[W]$. The graph G is **k-connected** if $|V| > k$ and, for every $V' \subseteq V$ of size at most $k-1$, the graph $G - V'$ is connected. That is, after removing an arbitrary set of $k - 1$ vertices, G is still connected. The **girth** of G is the length of a shortest cycle in G. A graph G is a **minor** of a graph H if G can be obtained from H by deleting vertices, deleting edges, and contracting edges. The **treewidth** of a graph measures how close a graph is to being a tree (see, e.g., [33] for a formal definition). We omit a formal definition here and only use the fact that if a graph G is a minor of H, then the treewidth of G is at most the treewidth of H.

Permutation Groups and Orbits. Let Ω be a finite set. We write $\mathsf{Sym}(\Omega)$ for the **symmetric group** with domain Ω, i.e., the group of all permutations of Ω. Let $\Gamma \leq \mathsf{Sym}(\Omega)$ be a finite permutation group with domain Ω and let p be a prime. The **order** $\mathsf{ord}(\sigma)$ of an element $\sigma \in \Gamma$ is the smallest number $\ell \geq 1$ such that σ^ℓ is the identity. If, for every $\sigma \in \Gamma$, there is an ℓ such that σ is of order p^ℓ, then Γ is called a **p-group**. The **orbit** of $u \in \Omega$ is the set

$$\mathsf{orb}_\Gamma(u) := \left\{ \sigma(u) \;\middle|\; \sigma \in \Gamma \right\}$$

of all elements in the domain, onto which u is mapped by Γ. In this way, Ω is partitioned into orbits. This notation generalizes to k-tuples. A **k-orbit** is a maximal set $O \subseteq \Omega^k$ such that, for every $\bar{u}, \bar{v} \in O$, there is a $\sigma \in \Gamma$ such that $\sigma(\bar{u}) = (\sigma(u_1), \dots, \sigma(u_k)) = \bar{v}$. We write $\mathsf{orb}_k(\Gamma)$ for the set of k-orbits of Γ. The group Γ is **transitive** if $|\mathsf{orb}_1(\Gamma)| = 1$. If additionally $|\Gamma| = |\Omega|$, then Γ is called **regular**.

Isomorphisms, Automorphisms, and Orbits of Structures. Let τ be a relational signature. For two τ-structures $\mathfrak{A}$ and $\mathfrak{B}$, an **isomorphism** $\varphi\colon \mathfrak{A} \to \mathfrak{B}$ is a bijection $A \to B$ such that $\bar{u} \in R^{\mathfrak{A}}$ if and only if $\varphi(\bar{u}) = (\varphi(u_1), \dots, \varphi(u_{\mathrm{ar}(R)})) \in R^{\mathfrak{B}}$ for every $R \in \tau$ and every $\bar{u} \in A^{\mathrm{ar}(R)}$. For $\bar{u} \in A^k$ and $\bar{v} \in B^k$, the structures $(\mathfrak{A}, \bar{u})$ and $(\mathfrak{B}, \bar{v})$ are isomorphic, denoted $(\mathfrak{A}, \bar{u}) \cong (\mathfrak{B}, \bar{v})$, if there is an isomorphism $\varphi\colon \mathfrak{A} \to \mathfrak{B}$ satisfying $\varphi(\bar{u}) = \bar{v}$. An **automorphism** φ of $(\mathfrak{A}, \bar{u})$ is an isomorphism $\varphi\colon (\mathfrak{A}, \bar{u}) \to (\mathfrak{A}, \bar{u})$. We say that φ **stabilizes** $\bar{u}$ and write $\mathsf{Aut}((\mathfrak{A}, \bar{u}))$ for the group of all automorphisms stabilizing $\bar{u}$, i.e., $\mathsf{Aut}((\mathfrak{A}, \bar{u}))$ is a permutation group with domain A. We say that the tuple of atoms $\bar{u}$ is **fixed**. We will use the same notation also for other objects beside tuples, e.g., for automorphisms stabilizing relations. The letters φ and ψ are used for isomorphisms or automorphisms. The set of k**-orbits** of $(\mathfrak{A}, \bar{u})$ is $\mathsf{orb}_k((\mathfrak{A}, \bar{u})) := \mathsf{orb}_k(\mathsf{Aut}((\mathfrak{A}, \bar{u})))$. For $k = 1$, we just write $\mathsf{orb}((\mathfrak{A}, \bar{u})) := \mathsf{orb}_1((\mathfrak{A}, \bar{u}))$.

2.2 Logics and Capturing Polynomial Time

In descriptive complexity theory, one considers isomorphism-closed classes $\mathcal{K}$ of relational τ-structures, that is, if $\mathfrak{A} \in \mathcal{K}$ and $\mathfrak{A} \cong \mathfrak{B}$, then $\mathfrak{B} \in \mathcal{K}$ for all relational τ-structures $\mathfrak{A}$ and $\mathfrak{B}$. An isomorphism-closed class of relational τ-structures is a **class of τ-structures**. A **boolean τ-query** is a class of τ-structures. While a query is just the same as a class, for queries one focuses on deciding them. A **polynomial time** boolean query Q is a class Q such that there is a polynomial time algorithm M deciding Q. That is, there exists a polynomial $p(n)$ such that M decides whether $\mathfrak{A}$ is in Q in time at most $p(|A|)$ for every τ-structure $\mathfrak{A}$. We follow [50] for the notion of a logic capturing PTIME. On an abstract level, a **logic** L provides for every relational signature τ

1. a decidable set $L[\tau]$, whose elements are called $L[\tau]$**-sentences**, and
2. a function that maps every $L[\tau]$-sentence Φ to a class of τ-structures $\mathcal{K}_\Phi$ called the **query defined by** Φ. That is, L is isomorphism-invariant.

A boolean τ-query is **definable in** L if there exists an $L[\tau]$-sentence defining it. A logic L **captures PTIME** if for every signature τ

1. every polynomial time boolean τ-query is L-definable and
2. the logic L can be **evaluated in polynomial-time**, that is, there is a computable function mapping every $\Phi \in L[\tau]$ to a polynomial $p(n)$ and an algorithm M such that M decides whether $\mathfrak{A}$ is in $\mathcal{K}_\Phi$ in time $p(|A|)$ for every τ-structure $\mathfrak{A}$.

In this exact sense, Gurevich [61] asked whether there is a logic capturing PTIME. While it seems to be difficult to answer this question, one can consider restricted classes of structures. Let $\mathcal{J}$ be a class of τ-structures for some signature τ. A logic L **captures PTIME on** $\mathcal{J}$ if

1. for every polynomial time boolean τ-query Q, there is an $L[\tau]$-sentence Φ such that for every $\mathfrak{A} \in \mathcal{J}$ we have $\mathfrak{A} \in \mathcal{K}_\Phi$ if and only if $\mathfrak{A} \in Q$ and

2. there is a computable function mapping every $\Phi \in L[\tau]$ to a polynomial $p(n)$ and an algorithm M such that M decides whether $\mathfrak{A}$ is in $\mathcal{K}_\Phi$ in time $p(|A|)$ for every τ-structure $\mathfrak{A} \in \mathcal{J}$.

Basic Notation for Logics. The sentences of all logics occurring in this thesis are composed of *formulas* and *terms* and use *variables*, which can be bound by quantifiers or operators. An unbound variable is called **free**. A formula or term without free variables is **closed**. A sentence is a closed formula. For a formula or term Φ, we use the usual convention and write $\Phi(x_1, \dots, x_\ell)$ to say that all free variables of Φ are among the x_i. For a logic L and a signature τ, the $L[\tau]$-formulas of a logic are the formulas using (some of) the relation symbols in τ. We use the letters Φ and Ψ for formulas and the letters s, t, and sometimes r for terms. For an $L[\tau]$-formula $\Phi(\bar{x})$ with k many free variables $\bar{x}$ and a τ-structure $\mathfrak{A}$, we denote by $\Phi^{\mathfrak{A}}$ the set of k-tuples $\bar{u}$ satisfying Φ in $\mathfrak{A}$, that is, Φ is satisfied if Φ is evaluated in $\mathfrak{A}$ when interpreting x_i as u_i for every $i \in [k]$. Similarly, for an $L[\tau]$-term $s(\bar{x})$, we denote by $s^{\mathfrak{A}}$ the function mapping tuples $\bar{u}$ for the free variables to whatever object the term defines (e.g., a number). Sometimes it is necessary to extend a relational signature by another relation symbol. For simplicity, we write $L[\tau, R]$ for $L[\tau \cup \{R\}]$ for a signature τ and a relation symbol R. Given two logics L and L', we write $L \leq L'$ if every boolean query definable in L is also definable in L'. This condition only requires that for every $L[\tau]$-sentence there is an equivalent $L'[\tau]$-sentence. However, for many logics and, in particular, all logics considered in this thesis, the expressive power on formulas can be reduced to the expressive power on sentences [98].

Distinguishing Structures. Let L be a logic and τ be a signature. An $L[\tau]$-sentence Φ **distinguishes** two τ-structures $\mathfrak{A}$ and $\mathfrak{B}$ if exactly one of $\mathfrak{A}$ and $\mathfrak{B}$ is contained in $\mathcal{K}_\Phi$. The logic L distinguishes $\mathfrak{A}$ and $\mathfrak{B}$ if there is some sentence distinguishing $\mathfrak{A}$ and $\mathfrak{B}$. Otherwise, $\mathfrak{A}$ and $\mathfrak{B}$ are L**-equivalent** which is denoted by $\mathfrak{A} \equiv_L \mathfrak{B}$. The notion of distinguishing structures naturally generalizes to tuples of atoms: Let $\mathfrak{A}$ and $\mathfrak{B}$ be two τ-structures and $\bar{u} \in A^\ell$ and $\bar{v} \in B^\ell$. We say that a logic L **distinguishes** $(\mathfrak{A}, \bar{u})$ from $(\mathfrak{B}, \bar{v})$ if there is an $L[\tau]$-formula Φ with ℓ many free variables such that $\bar{u} \in \Phi^{\mathfrak{A}}$ and $\bar{v} \notin \Phi^{\mathfrak{B}}$. Otherwise, $(\mathfrak{A}, \bar{u})$ and $(\mathfrak{B}, \bar{v})$ are L-equivalent denoted by $(\mathfrak{A}, \bar{u}) \equiv_L (\mathfrak{B}, \bar{v})$.

2.3 Extensions of First-Order Logic

Fixed-Point Logic with Counting. We recall **fixed-point logic with counting** IFPC (proposed in [70], see also [98]). Let τ be a signature and $\mathfrak{A} = (A, R_1^{\mathfrak{A}}, \dots, R_\ell^{\mathfrak{A}})$ be a τ-structure. We extend τ and $\mathfrak{A}$ with counting. We define $\tau^\# := \tau \uplus \{\cdot, +, 0, 1\}$ and $\mathfrak{A}^\# := (A, R_1^{\mathfrak{A}}, \dots, R_\ell^{\mathfrak{A}}, \mathbb{N}, \cdot, +, 0, 1)$ to be the two-sorted $\tau^\#$-structure that is the disjoint union of $\mathfrak{A}$ and $\mathbb{N}$. The structure $\mathfrak{A}^\#$ is not relational, but $\mathfrak{A}$ will always be relational.

IFPC is a two-sorted logic. **Element variables** range over the atoms and **numeric variables** range over the natural numbers. For element variables we use the letters x, y, and z, for numeric variables the Greek letters ν and μ, and for numeric terms the letters s and t. IFPC$[\tau]$-formulas use the signature $\tau^\#$ and extend first-order logic (FO) by a fixed-point operator and a counting term. When quantifying over numeric variables, their range needs to be bounded to ensure PTIME-evaluation: For an IFPC[τ]-formula Φ, a closed numeric IFPC[τ]-term s, a numeric variable ν possibly free in Φ, and a quantifier $Q \in \{\forall, \exists\}$, the formula

$$Q\nu \leq s.\ \Phi$$

is an IFPC[τ]-formula. An **inflationary fixed-point operator** defines a relation R. We allow R to mix atoms with numbers. For an IFPC[τ, R]-formula Φ and variables $\bar{x}\bar{\nu}$ possibly free in Φ, the fixed-point operator

$$[\mathsf{ifp} R\bar{x}\bar{\nu} \leq \bar{s}.\ \Phi]\,\bar{x}\bar{\nu}$$

is an IFPC[τ]-formula. Here $\bar{s}$ is a tuple of $|\bar{\nu}|$ many closed numeric terms which bound the values of $\bar{\nu}$ similar to the case of a quantifier. IFPC-terms are built from the constants 0 and 1, the binary functions $+$ and $\cdot$, and **counting terms**, which are the crucial element of IFPC. Let Φ be an IFPC[τ]-formula with possibly free variables $\bar{x}$ and $\bar{\nu}$ and let $\bar{s}$ be a $|\bar{\nu}|$-tuple of closed numeric IFPC[τ]-terms. Then

$$\#\bar{x}\bar{\nu} \leq \bar{s}.\ \Phi$$

is a numeric IFPC[τ]-term.

IFPC-formulas (or terms) are evaluated over $\mathfrak{A}^{\#}$. For a numeric term $s(\bar{x}\bar{\nu})$, we denote by $s^{\mathfrak{A}} \colon A^{|\bar{x}|} \times \mathbb{N}^{|\bar{\nu}|} \to \mathbb{N}$ the function mapping the values for the free variables of s to the value that s takes in $\mathfrak{A}^{\#}$. Likewise, for a formula $\Phi(\bar{x}\bar{\nu})$, we write $\Phi^{\mathfrak{A}} \subseteq A^{|\bar{x}|} \times \mathbb{N}^{|\bar{\nu}|}$ for the set of values for the free variables satisfying Φ. For an IFPC[τ]-formula $\Phi(\bar{y}\bar{x}\bar{\mu}\bar{\nu})$ and a $|\bar{\nu}|$-tuple $\bar{s}$ of closed numeric terms, the counting term is evaluated as follows:

$$(\#\bar{x}\bar{\nu} \leq \bar{s}.\ \Phi)^{\mathfrak{A}}(\bar{u}\bar{m}) := \left|\left\{\, \bar{w}\bar{n} \in A^{|\bar{x}|} \times \mathbb{N}^{|\bar{\nu}|} \;\middle|\; n_i \leq s_i^{\mathfrak{A}} \text{ for all } i \in [|\bar{\nu}|], \bar{u}\bar{w}\bar{m}\bar{n} \in \Phi^{\mathfrak{A}} \,\right\}\right|.$$

Now let $\Phi(\bar{y}\bar{x}\bar{\mu}\bar{\nu})$ be an IFPC[τ, R]-formula and $\bar{s}$ be a $|\bar{\nu}|$-tuple of closed numeric terms. To evaluate the inflationary fixed-point operator $[\mathsf{ifp} R\bar{x}\bar{\nu} \leq \bar{s}.\ \Phi]\,\bar{x}\bar{\nu}$, let $\bar{u} \in A^{|\bar{y}|}$ and $\bar{m} \in \mathbb{N}^{|\bar{\mu}|}$ and inductively define a series of relations $R_i^{\mathfrak{A}}$ called **stages** via

$$\begin{aligned} R_0^{\mathfrak{A}} &:= \emptyset, \\ R_{i+1}^{\mathfrak{A}} &:= R_i^{\mathfrak{A}} \cup \left\{\, \bar{w}\bar{n} \in A^{|\bar{x}|} \times \mathbb{N}^{|\bar{\nu}|} \;\middle|\; n_i \leq s_i^{\mathfrak{A}} \text{ for all } i \in [|\bar{\nu}|], \bar{u}\bar{w}\bar{m}\bar{n} \in \Phi^{(\mathfrak{A}, R_i^{\mathfrak{A}})} \,\right\}, \end{aligned}$$

where $(\mathfrak{A}, R_i^{\mathfrak{A}})$ denotes the $(\tau \cup \{R\})$-structure obtained from extending $\mathfrak{A}$ with $R_i^{\mathfrak{A}}$. By definition, $R_i^{\mathfrak{A}} \subseteq R_{i+1}^{\mathfrak{A}} \subseteq A^{|\bar{x}|} \times \{0, ..., s_1^{\mathfrak{A}}\} \times \cdots \times \{0, ..., s_{|\bar{\nu}|}^{\mathfrak{A}}\}$ for every $i \in \mathbb{N}$. Because IFPC-terms always evaluate to a number polynomial in the size of the input structure, the series stabilizes after a polynomial number of steps, i.e, $R_\ell^{\mathfrak{A}} = R_{\ell+1}^{\mathfrak{A}} =: R_{\bar{u}\bar{m}}^{\mathfrak{A}}$ for some $\ell \in \mathbb{N}$. The fixed-point operator evaluates as follows:

$$([\mathsf{ifp} R\bar{x}\bar{\nu} \leq \bar{s}.\ \Phi]\,\bar{x}\bar{\nu})^{\mathfrak{A}} := \left\{\, \bar{u}\bar{v}\bar{m}\bar{n} \;\middle|\; \bar{v}\bar{n} \in R_{\bar{u}\bar{m}}^{\mathfrak{A}} \,\right\}.$$

Fixed-Point Logic. Fixed-point logic (without counting) IFP extends first-order logic only with the inflationary fixed-point operator. IFP is the fragment of IFPC not using numeric variables and counting terms.

Theorem 2.1 (Immerman-Vardi Theorem [69, 114]). IFP *captures* PTIME *on the class of ordered structures.*

Finite Variable Counting Logic. The k**-variable first-order logic with counting** $\mathcal{C}_k$ extends the k-variable fragment of first-order logic with **counting quantifiers** $\exists^{\geq n}\bar{x}.\,\Phi$ stating that at least n distinct atoms satisfy the formula Φ (see [98]). Bounded variable logics with counting are a useful tool to prove IFPC-undefinability. Every IFPC-formula using k many variables is equivalent on structures of size up to n to a $\mathcal{C}_{\mathcal{O}(k)}$-formula. For $\mathcal{C}_k$-equivalence, we write $\equiv^k_{\mathcal{C}}$ for $\equiv_{\mathcal{C}_k}$. The logics $\mathcal{C}_k$ are used to prove IFPC-undefinability as follows: If there is a boolean τ-query Q and, for every positive $k \in \mathbb{N}$, there is a pair of τ-structures $\mathfrak{A}_k \equiv^k_{\mathcal{C}} \mathfrak{B}_k$ such that $\mathfrak{A}_k \in Q$ and $\mathfrak{B}_k \notin Q$, then IFPC does not define Q.

The logic $\mathcal{C}_k$ can be characterized by an Ehrenfeucht-Fraïssé-like pebble game – the **bijective** k**-pebble game** [66]. The game is played on two structures $\mathfrak{A}$ and $\mathfrak{B}$ by two players called Spoiler and Duplicator. There are k many pebble pairs (p_i, q_i) for $i \in [k]$. Positions in the game are tuples $(\mathfrak{A}, \bar{u}; \mathfrak{B}, \bar{v})$ for tuples $\bar{u} \in A^{\leq k}$ and $\bar{v} \in B^{\leq k}$ of the same length. In position $(\mathfrak{A}, \bar{u}; \mathfrak{B}, \bar{v})$, a pebble p_j is placed on the atom u_i and the pebble q_j is placed on v_i from some $j \in [k]$. It will not matter which pebble pair (p_j, q_j) is used for the i-th entry of $\bar{u}$ and $\bar{v}$. In the initial position, no pebbles are placed and both $\bar{u}$ and $\bar{v}$ are the empty tuple. The game proceeds as follows: If $|A| \neq |B|$, then Spoiler wins. Otherwise, Spoiler picks up a pair of pebbles (p_i, q_i) (which may or may not be already placed on the structures). Duplicator answers with a bijection $\lambda\colon A \to B$. Spoiler places the pebble p_i on an atom $w \in A$ and q_i on $\lambda(w) \in B$. If in the resulting position $(\mathfrak{A}, \bar{u}; \mathfrak{B}, \bar{v})$ there is no **pebble-respecting** local isomorphism, that is, the map defined via $u_i \mapsto v_i$ is not an isomorphism $(\mathfrak{A}[\bar{u}], \bar{u}) \to (\mathfrak{B}[\bar{v}], \bar{v})$, then Spoiler wins. Otherwise, the game continues with the next round. Duplicator wins if Spoiler never wins. We say that Spoiler (or Duplicator, respectively) has a **winning strategy** in position $(\mathfrak{A}, \bar{u}; \mathfrak{B}, \bar{v})$ if Spoiler (or Duplicator, respectively) can always win the game regardless of the moves of the other player.

Lemma 2.2 ([66])**.** *For all finite τ-structures $\mathfrak{A}$ and $\mathfrak{B}$ and all tuples $\bar{u} \in A^{\leq k}$ and $\bar{v} \in B^{\leq k}$ of the same length, Spoiler has a winning strategy in the bijective k-pebble game in position $(\mathfrak{A}, \bar{u}; \mathfrak{B}, \bar{v})$ if and only if $(\mathfrak{A}, \bar{u}) \not\equiv^k_{\mathcal{C}} (\mathfrak{B}, \bar{v})$.*

2.4 Choiceless Polynomial Time

The logic **Choiceless Polynomial Time** (CPT) was introduced by Blass, Gurevich, and Shelah [18] using abstract state machines. To give a concise definition of CPT, we follow Grädel and Grohe [43] and use ideas of Pakusa [103] to enforce polynomial bounds.

Hereditarily Finite Sets. For a set of atoms A, the set of **hereditarily finite sets over** A, denoted by $\mathsf{HF}(A)$, is the inclusion-wise minimal set such that $A \subseteq \mathsf{HF}(A)$ and $a \in \mathsf{HF}(A)$ for every finite $a \subseteq \mathsf{HF}(A)$. A set $a \in \mathsf{HF}(A)$ is transitive, if $c \in b \in a$ for some b implies $c \in a$. The **transitive closure** $\mathsf{TC}(a)$ of a is the least (with respect to set inclusion) transitive set b such that $a \subseteq b$.

The Logic BGS. Let τ be a relational signature and extend τ by adding set-theoretic function symbols $\tau^{\mathsf{HF}} := \tau \uplus \{\emptyset, \mathsf{Atoms}, \mathsf{Pair}, \mathsf{Union}, \mathsf{Unique}, \mathsf{Card}\}$, where $\emptyset$ and Atoms are constants, Union, Unique, and Card are unary, and Pair is binary. The **hereditarily finite**

expansion $\mathsf{HF}(\mathfrak{A})$ of a τ-structure $\mathfrak{A}$ is the τ^{HF}-structure over the universe $\mathsf{HF}(A)$ defined as follows. All relations in τ are interpreted as they are in $\mathfrak{A}$. The special function symbols have the expected set theoretic interpretation:

- $\emptyset^{\mathsf{HF}(\mathfrak{A})} := \emptyset$ and $\mathsf{Atoms}^{\mathsf{HF}(\mathfrak{A})} := A$,
- $\mathsf{Pair}^{\mathsf{HF}(\mathfrak{A})}(a, b) \quad := \{a, b\}$,
- $\mathsf{Union}^{\mathsf{HF}(\mathfrak{A})}(a) \quad := \{b \mid b \in c \text{ for some } c \in a\}$,
- $\mathsf{Unique}^{\mathsf{HF}(\mathfrak{A})}(a) \quad := \begin{cases} b & \text{if } a = \{b\}, \\ \emptyset & \text{otherwise, and} \end{cases}$
- $\mathsf{Card}^{\mathsf{HF}(\mathfrak{A})}(a) \quad := \begin{cases} |a| & \text{if } a \notin A, \\ \emptyset & \text{otherwise,} \end{cases}$

  where the number $|a|$ is encoded as a von Neumann ordinal.

Note that the Unique function is invariant under automorphisms because it only evaluates nontrivially when applied to singleton sets.

The logic CPT is obtained as the polynomial-time fragment of the logic BGS (named after Blass, Gurevich, and Shelah). BGS-**terms** are composed of variables, function symbols from τ^{HF}, and the two following constructs: If $s(\bar{x}y)$ and $t(\bar{x})$ are BGS[τ]-terms and $\Phi(\bar{x}y)$ is a BGS[τ]-formula, then the **comprehension term**

$$\{\, s \mid y \in t, \Phi \,\}$$

is a BGS[τ]-term, which binds the variable y. If $s(\bar{x}y)$ is a BGS[τ]-term, then the **iteration term**

$$s[y]^*$$

is a BGS[τ]-term that binds the variable y[1]. BGS[τ]**-formulas** are obtained by the usual boolean connectives from relations $R(t_1, \dots, t_k)$ for $R \in \tau$ of arity k and BGS[τ]-terms $t_1, \dots, t_k$ and from equality $t_1 = t_2$ for BGS[τ]-terms t_1 and t_2.

Let $\mathfrak{A}$ be a τ-structure. BGS-terms and formulas are evaluated over $\mathsf{HF}(A)$. For a BGS[τ]-term s with k many free variables $\bar{x}$, we denote by $s^{\mathfrak{A}} \colon \mathsf{HF}(A)^k \to \mathsf{HF}(A)$ the function mapping values for the free variables $\bar{x}$ to the value that s takes in $\mathsf{HF}(A)$. Similarly, for a formula Φ with k many free variables $\bar{x}$, we denote by $\Phi^{\mathfrak{A}}$ the set of all $\bar{a} \in \mathsf{HF}(A)^k$ satisfying Φ in $\mathsf{HF}(A)$. For BGS[τ]-terms $s(\bar{x}y)$ and $t(\bar{x})$ and a BGS[τ]-formula $\Phi(\bar{x}y)$, the comprehension term $\{s \mid y \in t, \Phi\}$ has the following semantics:

$$\{\, s \mid y \in t, \Phi \,\}^{\mathfrak{A}}(\bar{a}) := \left\{\, s^{\mathfrak{A}}(\bar{a}b) \;\middle|\; b \in t^{\mathfrak{A}}(\bar{a}) \text{ and } \bar{a}b \in \Phi^{\mathfrak{A}} \,\right\}$$

for every $\bar{a} \in \mathsf{HF}(A)^{|\bar{x}|}$. An iteration term $s[y]^*$, for a BGS[τ]-term s with free variables $\bar{x}$, and a tuple $\bar{b} \in \mathsf{HF}(A)^{|\bar{x}|}$ of sets for the free variables defines a sequence of sets via $a_0 := \emptyset$ and $a_{i+1} := s^{\mathfrak{A}}(\bar{b}a_i)$. Let $\ell(s[y]^*, \mathfrak{A}, \bar{b})$ be the least number i such that $a_{i+1} = a_i$ if it exists. The iteration term evaluates as follows:

$$(s[y]^*)^{\mathfrak{A}}(\bar{b}) := \begin{cases} a_\ell & \text{if } \ell := \ell(s[y]^*, \mathfrak{A}, \bar{b}) \text{ exists,} \\ \emptyset & \text{otherwise.} \end{cases}$$

[1]Here we differ from the definition in [43], in which s is only allowed to have one free variable y. For CPT, allowing more free variables does not increase expressiveness, but for our extensions later it is useful to allow additional free variables in an iteration term.

Choiceless Polynomial Time. A CPT[τ]-term (or formula, respectively) is a tuple (t, p) (or (Φ, p), respectively) of a BGS[τ]-term (or formula) and a polynomial $p(n)$. The semantics of CPT is derived from BGS by replacing t with (t, p) everywhere (or Φ with (Φ, p), respectively) with the following exception for iteration terms:

$$(s[y]^*, p)^{\mathfrak{A}}(\bar{b}) := \begin{cases} a_\ell & \text{if } \ell := \ell(s[y]^*, \mathfrak{A}, \bar{b}) \text{ exists,} \\ & \qquad \ell \leq p(|A|), \text{ and } |\mathsf{TC}(a_i)| \leq p(|A|) \text{ for every } i \in [\ell], \\ \emptyset & \text{otherwise,} \end{cases}$$

where the sets a_i are defined as above. The size of a_i is measured by $|\mathsf{TC}(a_i)|$ because, by transitivity, $\mathsf{TC}(a_i)$ contains all sets $b_k \in \cdots \in b_1 \in a_i$ occurring somewhere in the structure of a_i. To ensure evaluation of CPT in polynomial time, it suffices to put polynomial bounds on iteration terms because all other terms increase the size of the defined sets only polynomially.

2.5 Logical Interpretations

A logical interpretation is the logical correspondence to an algorithmic reduction. An interpretation transforms a relational structure to another one. For extensions of first-order logic, the notion of an interpretation can be defined rather uniformly. We define IFPC-interpretations and explain afterwards how to alter the definition to obtain, for example, IFP-interpretations or FO-interpretations. The notion of a CPT-interpretation is defined afterwards.

IFPC-Interpretations. An IFPC[τ, σ]-interpretation defines a partial function mapping τ-structures to σ-structures, where the function is defined in terms of IFPC-formulas operating on tuples of the input τ-structure. In the case of IFPC, these tuples not only contain atoms but also numbers. For the sake of readability, in the following we use $\bar{x}$, $\bar{y}$, and $\bar{z}$ for a tuple of both element and numeric variables and $\bar{u}$, $\bar{v}$, and $\bar{w}$ for a tuple of both atoms and numbers. Let $\sigma = \{R_1, \ldots, R_\ell\}$ be a signature. A **d-dimensional IFPC[τ, σ]-interpretation** $\Theta(\bar{z})$ with parameters $\bar{z}$ is a tuple

$$\Theta(\bar{z}) = \Big(\Phi_{\mathrm{dom}}(\bar{z}\bar{x}), \Phi_{\cong}(\bar{z}\bar{x}\bar{y}), \Phi_{R_1}(\bar{z}\bar{x}_1 \ldots \bar{x}_{\mathrm{ar}(R_1)}), \ldots, \Phi_{R_\ell}(\bar{z}\bar{x}_1 \ldots \bar{x}_{\mathrm{ar}(R_\ell)}), \bar{s}\Big)$$

of IFPC[τ]-formulas and a j-tuple $\bar{s}$ of closed numeric IFPC[τ]-terms, where j is the number of numeric variables in $\bar{x}$. The tuples of variables $\bar{x}$, $\bar{y}$, and all the $\bar{x}_i$ are of length d and agree on whether the k-th variable is an element or numeric variable.

Let $\mathfrak{A}$ be a τ-structure and $\bar{u} \in (A \cup \mathbb{N})^{|\bar{z}|}$ match the types of the parameter variables $\bar{z}$ (element or numeric). We now define $\Theta(\mathfrak{A}, \bar{u})$. Assume that, up to reordering, the first j variables in $\bar{x}$ are numeric variables and set $D := \{0, ..., s_1^{\mathfrak{A}}\} \times \cdots \times \{0, ..., s_j^{\mathfrak{A}}\} \times A^{d-j}$. We define the σ-structure $\mathfrak{B} = (B, R_1^{\mathfrak{B}}, \ldots, R_\ell^{\mathfrak{B}})$ via

$$B := \Big\{ \bar{v} \in D \Bigm| \bar{u}\bar{v} \in \Phi_{\mathrm{dom}}^{\mathfrak{A}} \Big\},$$
$$R_i^{\mathfrak{B}} := \Big\{ (\bar{v}_1, \ldots, \bar{v}_{\mathrm{ar}(R_i)}) \in B^{\mathrm{ar}(R_i)} \Bigm| \bar{u}\bar{v}_1 \ldots \bar{v}_{\mathrm{ar}(R_i)} \in \Phi_{R_i}^{\mathfrak{A}} \Big\} \text{ for every } i \in [\ell]$$

and a relation $\sim := \{(\bar{v}, \bar{w}) \in B^2 \mid \bar{u}\bar{v}\bar{w} \in \Phi_{\cong}^{\mathfrak{A}}\}$. The relation $\sim$ is a **congruence relation** if, for every $i \in [\ell]$ and all $(\bar{v}_1, \dots, \bar{v}_{\mathrm{ar}(R_i)}), (\bar{w}_1, \dots, \bar{w}_{\mathrm{ar}(R_i)}) \in B^{\mathrm{ar}(R_i)}$ such that $\bar{v}_k \sim \bar{w}_k$ for every $k \in [\mathrm{ar}(R_i)]$, we have $(\bar{v}_1, \dots, \bar{v}_{\mathrm{ar}(R_i)}) \in R_i^{\mathfrak{B}}$ if and only if $(\bar{w}_1, \dots, \bar{w}_{\mathrm{ar}(R_i)}) \in R_i^{\mathfrak{B}}$. If $\sim$ is a congruence relation, then $\mathfrak{B}/_{\sim}$ denotes the quotient of $\mathfrak{B}$ by $\sim$. We finally define

$$\Theta(\mathfrak{A}, \bar{u}) := \begin{cases} \mathfrak{B}/_{\sim} & \text{if } \sim \text{ is a congruence relation on } \mathfrak{B}, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

An interpretation is called **equivalence-free** if $\Phi_{\cong}(\bar{z}\bar{x}\bar{y})$ is the formula $\bar{x} = \bar{y}$.

If we consider a logic L extending IFPC, then the notion of an $L[\tau, \sigma]$-interpretation is defined exactly in the same way by replacing IFPC-formulas or terms with L-formulas or terms. For logics not possessing numeric variables like FO or IFP, the notion of an interpretation is similar and just omits the numeric part, i.e., there is no numeric term s bounding the range of numeric variables.

CPT-Interpretations. The notion of a CPT-interpretation is formally much simpler. Again, let $\sigma = \{R_1, \dots, R_\ell\}$ be a signature. A **CPT**$[\tau, \sigma]$**-interpretation** $\Theta(\bar{x})$ with parameters $\bar{x}$ is a tuple

$$\Theta(\bar{x}) = \Big(s_{\mathrm{univ}}(\bar{x}), \Phi_{R_1}(\bar{x}\bar{y}_1), \dots, \Phi_{R_\ell}(\bar{x}\bar{y}_\ell)\Big)$$

of a CPT$[\tau]$-term $s_{\mathrm{univ}}(\bar{x})$ and CPT$[\tau]$-formulas $\Phi_{R_\ell}(\bar{x}\bar{y}_i)$ such that $|\bar{y}_i| = \mathrm{ar}(R_i)$ for every $i \in [\ell]$. For every τ-structure $\mathfrak{A}$ and $\bar{a} \in \mathsf{HF}(A)^{|\bar{x}|}$, the image $\Theta(\mathfrak{A}, \bar{a})$ of Θ is the σ-structure $\mathfrak{B} = (B, R_1^{\mathfrak{B}}, \dots, R_\ell^{\mathfrak{B}})$ defined by

$$\begin{aligned} B &:= s_{\mathrm{univ}}^{\mathfrak{A}}(\bar{a}), \\ R_i^{\mathfrak{B}} &:= \left\{ \bar{b} \in B^{\mathrm{ar}(R_i)} \;\middle|\; \bar{a}\bar{b} \in \Phi_{R_i}^{\mathfrak{A}} \right\} \text{ for every } i \in [\ell]. \end{aligned}$$

Note that we also could use CPT-formulas in the setting of the IFPC-interpretation and operate on d-tuples. However, the notion of a dimension is not useful for a CPT-interpretation. Tuples of varying (and in particular unbounded) length can be encoded by the CPT-term s_{univ}. Hence, the restriction to d-tuples of the universe would be an unnecessary restriction and indeed incompatible with the concept of CPT. Similarly, no equivalence relation is needed because the equivalence classes can be defined as a set by s_{univ} directly.

Logical Reductions. Let L be a logic. A boolean τ-query P is L**-reducible** to a boolean σ-query Q if there is a parameter-free $L[\tau, \sigma]$-interpretation Θ such that, for every τ-structure $\mathfrak{A}$, it holds that $\mathfrak{A} \in P$ if and only if $\Theta(\mathfrak{A}) \in Q$. A logic L' is **closed under** L**-interpretations** (or L-reductions) if, for every signature τ, every boolean query P that is L-reducible to an L'-definable boolean query Q is itself L'-definable (cf. [37, 98]). We say that L' is closed under interpretations if L' is closed under L'-interpretations. The logics IFP and IFPC are closed under interpretations [98]. The same holds for CPT.

2.6 Definable Canonization

By the Immerman-Vardi Theorem, a polynomial-time evaluable logic L that is at least as expressive as IFP captures PTIME on ordered structures. That is, on every class of structures for which L defines a total order, L captures PTIME. But there are structures, on which every isomorphism-invariant logic fails to define a unique total order. Consider complete graphs as an example: For all total orders, there is an automorphism mapping one to the other. Hence, L cannot define one of them. To overcome this restriction, one can try to not define a total order on the input structure but a total order on an isomorphic copy. For example for a complete graph of order n, one can, in IFPC, consider the vertex set $[n]$ and then add all possible edges. Using the natural order on $[n]$, one obtains a unique total order on this copy. The mapping of some structure to an isomorphic and ordered copy is naturally defined via an interpretation.

Definition 2.3 (Canonization). Let $\mathcal{K}$ be a class of τ-structures and let L be a logic. An **L-canonization** for $\mathcal{K}$ is an $L[\tau, \tau \uplus \{\leq\}]$-interpretation Θ satisfying the following:

1. $\leq^{\Theta(\mathfrak{A})}$ is a total order on $\Theta(\mathfrak{A})$ for every $\mathfrak{A} \in \mathcal{K}$,
2. $\mathfrak{A} \cong \Theta(\mathfrak{A}) \restriction \tau$ for every $\mathfrak{A} \in \mathcal{K}$, and
3. $\Theta(\mathfrak{A}) \cong \Theta(\mathfrak{B})$ if and only if $\mathfrak{A} \cong \mathfrak{B}$ for every $\mathfrak{A}, \mathfrak{B} \in \mathcal{K}$.

The structure $\Theta(\mathfrak{A})$ is called the **Θ-canon** (or just the canon if unambiguous) of $\mathfrak{A}$. We say that L **canonizes** $\mathcal{K}$ if there is an L-canonization for $\mathcal{K}$.

Condition 1 requires to define an order on $\Theta(\mathfrak{A})$, Condition 2 requires that $\Theta(\mathfrak{A})$ is isomorphic to $\mathfrak{A}$ after removing the order, and Condition 3 requires that the canons of isomorphic structures are isomorphic *as ordered structures*. That is, there is only one function, which is possibly an automorphism, namely mapping the i-th vertex of one structure to the i-th vertex of the other. In an algorithmic context, one usually requires equality instead of isomorphism as ordered structures [109]. For a logic L possessing numbers (as, e.g., IFPC or CPT), Condition 3 can equivalently be stated with equality: There is an interpretation mapping ordered structures to structures whose universes are numbers by mapping the i-th vertex according to the order to the number i. This interpretation maps isomorphic ordered structures to *the same* structure. While Condition 3 is essential for algorithmic canonizations, it is implied by Condition 2 for L-definable canonizations: Let $\mathfrak{A}, \mathfrak{B} \in \mathcal{K}$. If $\mathfrak{A} \cong \mathfrak{B}$, then $\Theta(\mathfrak{A}) \cong \Theta(\mathfrak{B})$ because L is isomorphism-invariant. If $\Theta(\mathfrak{A}) \cong \Theta(\mathfrak{B})$, then by Condition 2,

$$\mathfrak{A} \cong \Theta(\mathfrak{A}) \restriction \tau \cong \Theta(\mathfrak{B}) \restriction \tau \cong \mathfrak{B}.$$

By combining a definable canonization with the Immerman-Vardi Theorem 2.1, we obtain the following:

Lemma 2.4. *Let $\mathcal{K}$ be a class of τ-structures and L be a polynomial-time evaluable logic such that* IFP $\leq L$. *If L defines a canonization for $\mathcal{K}$, then L captures* PTIME *on $\mathcal{K}$.*

When defining canonizations, it will be of particular interest to determine the orbit-partition of a structure by a formula in the following sense.

Definition 2.5 (Distinguishable Orbits). For $k \in \mathbb{N}$, a logic L **distinguishes** k**-orbits** of a class of τ-structures $\mathcal{K}$ if there is an L-formula $\Phi(\bar{x}, \bar{y})$ with $|\bar{x}| = |\bar{y}| = k$ such that, for every $\mathfrak{A} \in \mathcal{K}$,

(a) the formula Φ defines a total preorder $\preceq$ on A^k via $\bar{u} \preceq \bar{v}$ whenever $(\bar{u}, \bar{v}) \in \Phi^{\mathfrak{A}}$ and

(b) the $\preceq$-equivalence classes correspond to the k-orbit partition of $\mathfrak{A}$.

Note that, because Φ defines a total preorder, Φ not only defines the k-orbit partition but also orders the k-orbits.

2.7 The Weisfeiler-Leman Refinement

We review the Weisfeiler-Leman refinement. For every fixed dimension $k \geq 1$, the k-dimensional Weisfeiler-Leman refinement computes an isomorphism-invariant coloring of k-tuples of a relational structure (of arity at most k). For our purpose, it suffices to consider the 2-dimensional Weisfeiler-Leman refinement [116].

Edge-Colored Graphs. An **edge-colored directed graph** is a tuple $G = (V, E, \chi)$, such that (V, E) is a graph and $\chi\colon E \to C$ is a **coloring function** into some set C of colors. We will often consider complete directed graphs (with loops), i.e., the case $E = V^2$. For a tuple of $m > 1$ vertices $(v_1, \dots, v_m) \in V^m$, we set

$$\overline{\chi}(v_1, \dots, v_m) := \big(\chi(v_1, v_2), \chi(v_2, v_3), \dots, \chi(v_{m-1}, v_m)\big)$$

and, for a single vertex $v \in V$, we set $\chi(v) := \chi(v, v)$. A coloring χ induces a partition $\pi(\chi)$ of the vertex pairs. For two colorings χ and χ', we write $\pi(\chi) \preceq \pi(\chi')$ to say that $\pi(\chi)$ is **at least as fine as** $\pi(\chi')$. If not ambiguous, we may just write $\chi \preceq \chi'$. If $\pi(\chi) = \pi(\chi')$, we also write $\chi \equiv \chi'$. We say that χ respects **converse equivalence** if

1. χ assigns different colors to loops and edges, that is, $\chi(u, u) \neq \chi(v, w)$ for every $u, v, w \in V$ with $v \neq w$ and
2. $\chi(u_1, v_1) = \chi(u_2, v_2)$ implies $\chi(v_1, u_1) = \chi(v_2, u_2)$ for all $u_1, u_2, v_1, v_2 \in V$, that is, the color $\chi(u_1, v_1)$ determines the color $\chi(v_1, u_1)$.

An arbitrary coloring χ can be turned isomorphism-invariantly into a converse-equivalence-respecting coloring χ_{init} as follows:

$$\chi_{\text{init}}(u, v) := \big(\chi(u, v), \chi(v, u), \delta_{u,v}\big),$$

where $\delta_{u,v}$ is the Kronecker delta which is 1 if $u = v$ and 0 otherwise. We refer to χ_{init} as the **initial coloring** of G.

A k**-walk** or walk of **length** k from v_1 to v_{k+1} is a tuple $(v_1, \dots, v_{k+1}) \in V^{k+1}$. Its color is $\overline{\chi}(v_1, \dots, v_{k+1})$. We say that tuples in C^k are (potential) k**-walk colors** in χ and omit the coloring if it is clear from the context.

General Refinements. A **refinement** r is a function which for every edge-colored graph $G = (V, E, \chi)$ yields a new coloring χ_r such that $\chi_r \preceq \chi$ and which is isomorphism-invariant: For all isomorphic edge-colored graphs $G = (V, E, \chi) \cong G' = (V', E', \chi')$, that is, there is a color-preserving isomorphism mapping each vertex to a vertex of the same color, we have $(V, E, \chi_r) \cong (V', E', \chi'_r)$. We write χ_r for the application of the refinement r to the coloring χ and χ_r^m for m applications of r, i.e.,

$$\begin{aligned} \chi_r^0 &:= \chi_{\text{init}}, \\ \chi_r^{m+1} &:= (\chi_r^m)_r. \end{aligned}$$

We denote by χ_r^∞ the stable coloring, i.e., the coloring χ_r^m for the smallest m such that $\chi_r^m \equiv \chi_r^{m+1}$. The **stable coloring** χ_r^∞ always exists because the sequence

$$\chi_r^0 \succeq \chi_r^1 \succeq \cdots \succeq \chi_r^m$$

necessarily stabilizes after $|V|^2$ many steps.

Let $G' = (V', E', \chi')$ be an edge-colored complete graph and assume $u, v \in V$ and $u', v' \in V'$. We say that the refinement r **distinguishes** (u, v) from (u', v') in m iterations if $\chi_r^m(u, v) \neq (\chi')_r^m(u', v')$. The refinement r **distinguishes** G and G' in m iterations if the multiset of colors after m iterations is different, that is

$$\left\{\!\!\left\{ \chi_r^m(u, v) \;\middle|\; u, v \in V \right\}\!\!\right\} \neq \left\{\!\!\left\{ (\chi')_r^m(u', v') \;\middle|\; u', v' \in V' \right\}\!\!\right\}.$$

The **iteration number** of the refinement r on the graph G is the number of applications of r needed to obtain the stable coloring.

In the context of refinements, considering edge-colored graphs is more natural than considering relational structures. To relate edge-colored graphs with logics, we turn an edge-colored graph $G = (V, E, \chi)$ for $\chi\colon E \to [\ell]$ into the binary relational structure $\mathfrak{A}_G = (V, R_1^{\mathfrak{A}_G}, \dots, R_\ell^{\mathfrak{A}_G})$ where

$$R_i^{\mathfrak{A}_G} = \left\{ (u, v) \in E \;\middle|\; \chi(u, v) = i \right\}$$

for every $i \in [\ell]$. Likewise, every binary relational structure $\mathfrak{A} = (A, R_1^{\mathfrak{A}}, \dots, R_\ell^{\mathfrak{A}})$ can be turned into the edge-colored graph $G_{\mathfrak{A}} = (A, A^2, \chi)$ where

$$\chi(u, v) = \left\{ i \in [\ell] \;\middle|\; (u, v) \in R_i^{\mathfrak{A}} \right\}.$$

A refinement distinguishes two binary relational structures $\mathfrak{A}$ and $\mathfrak{B}$ if the edge-colored graphs $G_{\mathfrak{A}}$ and $G_{\mathfrak{B}}$ are distinguished by the refinement. The analogous definition applies to pairs of atoms of binary relational structures.

The Weisfeiler-Leman Refinement. Assume $G = (V, E, \chi)$ is an edge-colored graph and assume χ respects converse equivalence. The 2-dimensional **Weisfeiler-Leman refinement** WL is defined as follows:

$$\chi_{\mathsf{WL}}(u, v) := \left\{\!\!\left\{ \overline{\chi}(u, w, v) \;\middle|\; w \in V \right\}\!\!\right\}.$$

Intuitively, WL refines the color of a vertex pair with the colors of all triangles containing this pair. Indeed, this definition gives a refinement: Because loops and non-loops always have distinct colors, the presence of the color $\overline{\chi}(u,u,v)$ (or $\overline{\chi}(u,v,v)$) in the multiset ensures that pairs colored differently remain colored differently after applying WL. In particular, we do not need to include the color $\chi(u,v)$ of the previous iteration in the new color explicitly to ensure that WL is a refinement. There is a close connection between the WL refinement and the 3-variable counting logic $\mathcal{C}_3$.

Lemma 2.6 ([20]). *For all $G = (V, E, \chi)$, $G' = (V', E', \chi')$, $u, v \in V$, and $u', v' \in V'$, the following statements are equivalent:*

(a) There is a $\mathcal{C}_3$-formula of quantifier depth m distinguishing $(\mathfrak{A}_G, uv)$ and $(\mathfrak{A}_{G'}, u'v')$.

(b) The 2-dimensional Weisfeiler-Leman refinement WL distinguishes the pairs (u,v) and (u',v') in m iterations (starting with the initial colorings of G and G').

It follows that the WL refinement distinguishes exactly the same graphs as the logic $\mathcal{C}_3$ and the bijective 3-pebble game by Lemma 2.2.

2.8 The CFI Construction

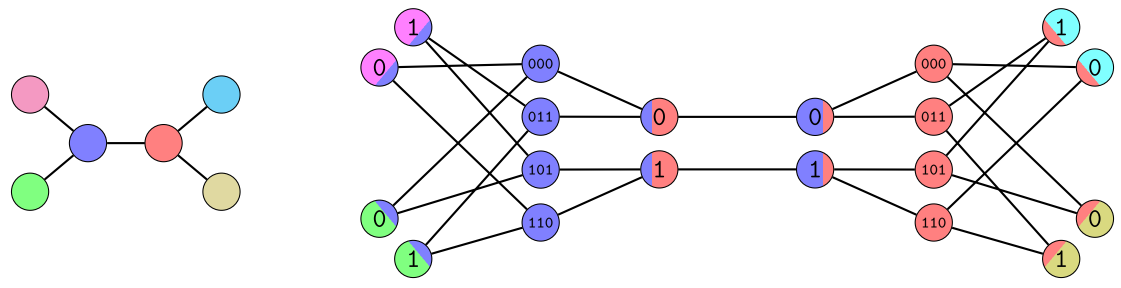
This section presents the so-called CFI graphs introduced by Cai, Fürer, and Immerman [20]. This graph construction produces non-isomorphic graphs, which cannot be distinguished in IFPC. At the core of the construction are the so-called CFI gadgets. In the original work [20], such a gadget consists of gadget and edge vertices, which are also known as inner and outer vertices, respectively. We provide multiple variants of the CFI construction. We begin with the original construction with gadget and edge vertices and later introduce two further variants: one only using gadget vertices and one only using edge vertices. CFI graphs will be used at multiple places in this thesis. Depending on the use, it will be beneficial to work with a different variant of the CFI graphs. The main benefits of each variant are discussed in this section. Most of the following lemmas are well-known. We will give selected proofs or proof sketches for illustration.

2.8.1 The Original CFI Construction

A **base graph** is a simple, connected, and possibly colored graph $G = (V, E, \preceq)$ (uncolored graphs are seen to be colored monochromatically). The vertices of G are called **base vertices** and its edges **base edges**. We use fraktur letters $\mathfrak{u}$, $\mathfrak{v}$, and $\mathfrak{w}$ for base vertices and the letter $\mathfrak{e}$ for a base edge.

A CFI graph is obtained from G by first replacing every base vertex by a gadget. Second, gadgets of adjacent base vertices are connected by adding edges between them. The gadget for the base vertex $\mathfrak{u} \in V$ consists of a set of **gadget vertices** $A^{\mathfrak{u}}_{\mathrm{g}}$, a set of **edge vertices** $A^{\mathfrak{u}}_{\mathrm{e}}$, and a set of edges $E^{\mathfrak{u}}$ between them:

$$\begin{aligned}
A^{\mathfrak{u}}_{\mathrm{g}} &:= \left\{ (\mathfrak{u}, \bar{a}) \;\middle|\; \bar{a} \in \mathbb{F}_2^{N_G(\mathfrak{u})}, \sum \bar{a} = 0 \right\},\\
A^{\mathfrak{u}}_{\mathrm{e}} &:= \left\{ (\mathfrak{u}, \mathfrak{v}, b) \;\middle|\; \mathfrak{v} \in N_G(\mathfrak{u}), b \in \mathbb{F}_2 \right\},\\
A^{\mathfrak{u}}_{\mathrm{ge}} &:= A^{\mathfrak{u}}_{\mathrm{g}} \cup A^{\mathfrak{u}}_{\mathrm{e}},\\
E^{\mathfrak{u}} &:= \left\{ \{(\mathfrak{u}, \bar{a}), (\mathfrak{u}, \mathfrak{v}, b)\} \;\middle|\; (\mathfrak{u}, \bar{a}) \in A^{\mathfrak{u}}_{\mathrm{g}}, (\mathfrak{u}, \mathfrak{v}, b) \in A^{\mathfrak{u}}_{\mathrm{e}}, \bar{a}(\mathfrak{u}) = b \right\}.
\end{aligned}$$



(a) A part of a base graph **(b)** Two connected CFI gadgets

2.1 Construction of a CFI graph with gadget and edge vertices. Figure (a) shows a part of a colored base graph G. Figure (b) shows the two gadgets for the red base vertex $\mathfrak{u}$ and the blue base vertex $\mathfrak{v}$ in the graph $\mathsf{CFI}(G, f)$. The figure assumes that $f(\{\mathfrak{u}, \mathfrak{v}\}) = 0$. Gadget vertices inherit the color from their base vertex. An edge vertex of the gadget of $\mathfrak{u}$ inherits the color from the ordered pair $(\mathfrak{u}, \mathfrak{w})$ of the corresponding neighbor $\mathfrak{w}$ of $\mathfrak{u}$. This color is indicated by asymmetrically coloring a vertex in the figure with two colors.

For every base edge $\mathfrak{e} = \{\mathfrak{u}, \mathfrak{v}\}$ incident to $\mathfrak{u}$, there is a pair of edge vertices $(\mathfrak{u}, \mathfrak{v}, 0)$ and $(\mathfrak{u}, \mathfrak{v}, 1)$ called the **edge-vertex-pair** of $(\mathfrak{u}, \mathfrak{v})$. Note that, for every base edge $\{\mathfrak{u}, \mathfrak{v}\} \in E$, there are two edge-vertex-pairs, namely the one of $(\mathfrak{u}, \mathfrak{v})$ and the one of $(\mathfrak{v}, \mathfrak{u})$. For every collection of one base vertex per edge-vertex-pair $(\mathfrak{u}, \mathfrak{v}_1, b_1), \dots, (\mathfrak{u}, \mathfrak{v}_d, b_d)$, where $N_G(\mathfrak{u}) = \{\mathfrak{v}_1, \dots, \mathfrak{v}_d\}$ such that $b_1 + \dots + b_d = 0$, there is a gadget vertex $(\mathfrak{u}, \bar{a})$ with $\bar{a}(\mathfrak{v}_i) = b_i$ for every $i \in [d]$ adjacent to these edge vertices. Note that $|A^{\mathfrak{u}}_{\mathrm{g}}| = 2^{d-1}$ and that $|A^{\mathfrak{u}}_{\mathrm{e}}| = 2d$. For a gadget vertex $u = (\mathfrak{u}, \bar{a})$ and a base vertex $\mathfrak{v} \in N_G(\mathfrak{u})$, we write $u(\mathfrak{v})$ for $\bar{a}(\mathfrak{v})$.

Definition 2.7 (Origin of CFI Vertices). The **origin** of a gadget vertex $u = (\mathfrak{u}, \bar{a}) \in A^{\mathfrak{u}}_{\mathrm{g}}$ is $\mathsf{orig}(u) = \mathfrak{u}$. The origin of an edge vertex $v = (\mathfrak{u}, \mathfrak{v}, b) \in A^{\mathfrak{u}}_{\mathrm{e}}$ is $\mathsf{orig}(v) = (\mathfrak{u}, \mathfrak{v})$. We extend the notation to tuples. The origin of a k-tuple $\bar{w}$ is $\mathsf{orig}(\bar{w}) = (\mathsf{orig}(w_1), \dots, \mathsf{orig}(w_k))$.

We now compose CFI gadgets to a CFI graph. For a function $f \colon E \to \mathbb{F}_2$, the **CFI graph (with gadget and edge vertices)** $\mathfrak{A}_{\mathrm{ge}} = \mathsf{CFI}_{\mathrm{ge}}(G, f) = (A_{\mathrm{ge}}, E^{\mathfrak{A}_{\mathrm{ge}}}, \preceq^{\mathfrak{A}_{\mathrm{ge}}})$ is defined as follows:

$$A_{\mathrm{ge}} := \bigcup_{\mathfrak{u} \in V} A^{\mathfrak{u}}_{\mathrm{ge}},$$
$$E^{\mathfrak{A}_{\mathrm{ge}}} := \bigcup_{\mathfrak{u} \in V} E^{\mathfrak{u}} \cup \bigcup_{\{\mathfrak{u},\mathfrak{v}\} \in E} \Big\{ \{(\mathfrak{u}, \mathfrak{v}, a), (\mathfrak{v}, \mathfrak{u}, b)\} \;\Big|\; a + b = f(\{\mathfrak{u}, \mathfrak{v}\}) \Big\},$$

that is, we replace every base vertex by a CFI gadget and connect, for every base edge $\{\mathfrak{u}, \mathfrak{v}\} \in E$, the edge vertex pairs of $(\mathfrak{u}, \mathfrak{v})$ and $(\mathfrak{u}, \mathfrak{v})$ by a perfect matching according to f (cf. Figure 2.1). If $f(\{\mathfrak{u}, \mathfrak{v}\}) = 0$, then we add the edges $\{(\mathfrak{u}, \mathfrak{v}, 0), (\mathfrak{v}, \mathfrak{u}, 0)\}$ and $\{(\mathfrak{u}, \mathfrak{v}, 1), (\mathfrak{v}, \mathfrak{u}, 1)\}$. If otherwise $f(\{\mathfrak{u}, \mathfrak{v}\}) = 1$, then we add the edges $(\mathfrak{u}, \mathfrak{v}, 0), (\mathfrak{v}, \mathfrak{u}, 1)$ and $(\mathfrak{u}, \mathfrak{v}, 1), (\mathfrak{v}, \mathfrak{u}, 0)$. The vertices $A_{\mathrm{g}} := \bigcup_{\mathfrak{u} \in V} A^{\mathfrak{u}}_{\mathrm{g}}$ are the gadget vertices of G and likewise A_{e} are the edge vertices of G. The coloring is inherited from the base graph, that is, base and edge vertices always receive different colors and obtain their color from their origin (see Figure 2.1 again). Formally,

$$u \preceq^{\mathfrak{A}_{\mathrm{ge}}} v \Leftrightarrow \begin{cases} u \in A_{\mathrm{g}} \text{ and } v \in A_{\mathrm{e}}, \text{or} \\ u, v \in A_{\mathrm{g}} \text{ and } \mathsf{orig}(u) \preceq \mathsf{orig}(v), \text{or} \\ u, v \in A_{\mathrm{e}} \text{ and } \mathsf{orig}(u) \preceq \mathsf{orig}(v), \end{cases}$$

for every $u, v \in A_{\mathrm{ge}}$, where $\preceq$ is extended to pairs V^2 using the lexicographical order.

Definition 2.8 (Twisted Base Edge). Let $f, g\colon E \to \mathbb{F}_2$ be two arbitrary functions. A base edge $\mathfrak{e} \in E$ is **twisted** by f and g if $f(\mathfrak{e}) \neq g(\mathfrak{e})$.

2.8.2 Automorphisms and Isomorphisms of CFI Graphs

The crucial property of CFI graphs is that they have a rich automorphism structure. Let $G = (V, E, \preceq)$ be a base graph and $f\colon E \to \mathbb{F}_2$. We first consider the automorphisms of a single gadget in $\mathsf{CFI}_{\mathrm{ge}}(G, f)$ for some base vertex $\mathfrak{u} \in V$, i.e., the automorphism of the subgraph induced by $A^{\mathfrak{u}}_{\mathrm{ge}}$. We assume that all neighbors of $\mathfrak{u}$ are colored differently. Then all edge-vertex-pairs in the gadget of $\mathfrak{u}$ are colored differently. By the construction of the gadget vertices, exchanging the two vertices of an even number of edge-vertex-pairs extends to an automorphism of the gadget. All automorphisms of the gadget are obtained in this way because we assumed that all edge-vertex-pairs are colored differently [20, Lemma 6.1].

We now turn to isomorphisms of the CFI graph $\mathsf{CFI}_{\mathrm{ge}}(G, f)$. Assume that $\mathfrak{e}_1, \mathfrak{e}_2 \in E$ are base edges and let $g\colon E \to \mathbb{F}_2$ be another function. Now consider the function $g_{\mathfrak{e}_1,\mathfrak{e}_2}\colon E \to \mathbb{F}_2$ defined via

$$g_{\mathfrak{e}_1,\mathfrak{e}_2}(\mathfrak{e}) := \begin{cases} g(\mathfrak{e}) + 1 & \text{if } \mathfrak{e} \in \{\mathfrak{e}_1, \mathfrak{e}_2\} \text{ and } \mathfrak{e}_1 \neq \mathfrak{e}_2 \\ g(\mathfrak{e}) & \text{otherwise.} \end{cases}$$

If $\mathfrak{e}_1 \neq \mathfrak{e}_2$, then the functions f and $g_{\bar{\mathfrak{u}}}$ twist the edge $\mathfrak{e}_1$ (or the edge $\mathfrak{e}_2$, respectively) if and only if f and g do not twist $\mathfrak{e}_1$ (or $\mathfrak{e}_2$, respectively). The crucial property of CFI graphs is the following:

Lemma 2.9. *For every $g\colon E \to \mathbb{F}_2$ and every path $\bar{\mathfrak{v}} = (\mathfrak{v}_1, \dots, \mathfrak{v}_\ell)$ in G of length $\ell > 2$ (which may contain vertices multiple times), there is an isomorphism*

$$\varphi_{\bar{\mathfrak{v}}}\colon \mathsf{CFI}_{\mathrm{ge}}(G, g) \to \mathsf{CFI}_{\mathrm{ge}}(G, g_{\{\mathfrak{v}_1,\mathfrak{v}_2\},\{\mathfrak{v}_{\ell-1},\mathfrak{v}_\ell\}})$$

which is the identity map on all vertices apart from the gadgets vertices with origin $\mathfrak{v}_i$ for some $1 < i < \ell$ and the edge vertices with origin $(\mathfrak{v}_i, \mathfrak{v}_{i+1})$ or $(\mathfrak{v}_{i+1}, \mathfrak{v}_i)$ for some $1 < i < \ell$ (note that, by definition, all CFI graphs over the same base graph have the same vertex set).

For a formal proof of the lemma in a more general setting we refer to [45, Lemma 3.11]. The **path-isomorphism** $\varphi_{\bar{\mathfrak{v}}}$ is composed of automorphisms of the gadgets of the base vertices $\mathfrak{v}_2, \dots, \mathfrak{v}_{\ell-1}$. For the gadget of $\mathfrak{v}_i$, the automorphisms exchanges the two vertices with origin $(\mathfrak{v}_i, \mathfrak{v}_{i+1})$ and the two vertices with origin $(\mathfrak{v}_i, \mathfrak{v}_{i-1})$. Because the vertices of an even number of edge-vertex-pairs, namely two, are exchanged, we indeed obtain an automorphism. Combining all these automorphisms yields the desired isomorphism. Intuitively, the lemma states that we can move twists along paths in the base graph. Assume that the base edge $\{\mathfrak{v}_1, \mathfrak{v}_2\}$ is twisted by f and g but $\{\mathfrak{v}_{\ell-1}, \mathfrak{v}_\ell\}$ is not. Then, after applying the path-isomorphism $\varphi_{\bar{\mathfrak{v}}}$, the base edge $\{\mathfrak{v}_1, \mathfrak{v}_2\}$ is not twisted by f and $g_{\{\mathfrak{v}_1,\mathfrak{v}_2\},\{\mathfrak{v}_{\ell-1},\mathfrak{v}_\ell\}}$ but the base edge $\{\mathfrak{v}_{\ell-1}, \mathfrak{v}_\ell\}$ is now twisted.

Lemma 2.9 has several consequences. The most important one is that, for a base graph G, there are only two CFI graphs over G up to isomorphism. Multiple twists can all be moved to the same base edge using path-isomorphisms because base graphs are connected. If the number of twists is even, then they cancel:

Lemma 2.10 ([20, Lemma 6.2])**.** $\mathsf{CFI}_{\mathrm{ge}}(G, f) \cong \mathsf{CFI}_{\mathrm{ge}}(G, g)$ *for every* $f, g \colon E \to \mathbb{F}_2$ *twisting an even number of base edges, that is,* $\sum f = \sum g$.

CFI graphs $\mathsf{CFI}_{\mathrm{ge}}(G, f)$ for which $\sum f = 0$ are called **even** and the others are called **odd**. If we are only interested in CFI graphs up to isomorphism, we write $\mathsf{CFI}_{\mathrm{ge}}(G, 0)$ for the even and $\mathsf{CFI}_{\mathrm{ge}}(G, 1)$ for the odd CFI graph over G.

Automorphisms and Orbits. If we consider the special case of paths forming a cycle in Lemma 2.9, we obtain automorphisms of CFI graphs because $g_{\mathfrak{e},\mathfrak{e}} = g$ for every $g \colon E \to \mathbb{F}_2$ and $\mathfrak{e} \in E$. Formally, we need to consider paths $(\mathfrak{v}_1, \dots, \mathfrak{v}_\ell, \mathfrak{v}_1, \mathfrak{v}_2)$ so that the same base edge $\{\mathfrak{v}_1, \mathfrak{v}_2\}$ is twisted twice, that is, the base edge is not twisted. Now assume that $G = (V, E, \leq)$ is a totally ordered base graph. Then G has no nontrivial automorphism and every automorphism of $\mathsf{CFI}_{\mathrm{ge}}(G, g)$ is composed of multiple such **cycle-automorphisms**.

If the base graph itself has nontrivial automorphisms, then the CFI graph inherits these automorphisms. For a (not necessarily ordered) base graph $G = (V, E, \preceq)$, every automorphism of $\mathsf{CFI}_{\mathrm{ge}}(G, g)$ is obtained from an automorphism of the base graph and a "CFI automorphism" where G is assumed to be totally ordered [102]. One consequence is that the two edge vertices of an edge-vertex-pair form a 1-orbit if and only if its origin base edge is contained in a cycle in G:

Lemma 2.11. *Let* $G = (V, E, \preceq)$ *be a base graph,* $\{\mathfrak{u}, \mathfrak{v}\} \in E$ *be a base edge,* $f \colon E \to \mathbb{F}_2$, $\mathfrak{A}_{\mathrm{ge}} = \mathsf{CFI}_{\mathrm{ge}}(G, f)$, $\bar{u} \in A_{\mathrm{g}}^*$, *and* $\bar{v} \in A_{\mathrm{e}}^*$. *Then* $\{w \in A_{\mathrm{ge}} \mid \mathsf{orig}(w) = (\mathfrak{u}, \mathfrak{u})\} \in \mathsf{orb}((\mathfrak{A}, \bar{u}))$ *if and only if there is a cycle containing the edge* $\{\mathfrak{u}, \mathfrak{v}\}$ *in*

$$G - \Big\{ \{\mathfrak{w}, \mathfrak{w}'\} \ \Big|\ (\mathfrak{w}, \mathfrak{w}') = \mathsf{orig}(v_i), i \leq |\bar{v}| \Big\} - \Big\{ \mathsf{orig}(u_i) \ \Big|\ i \leq |\bar{u}| \Big\}.$$

Proof. Set $E' := \{\{\mathfrak{w}, \mathfrak{w}'\} \mid (\mathfrak{w}, \mathfrak{w}') = \mathsf{orig}(v_i), i \leq |\bar{v}|\}$ and $V' := \{\mathsf{orig}(u_i) \mid i \leq |\bar{u}|\}$. Assume that there is a cycle in $G - E' - V'$ containing $\{\mathfrak{u}, \mathfrak{v}\}$. Then we can use a cycle-automorphism for that cycle to exchange the two edge vertex pairs with origin $(\mathfrak{u}, \mathfrak{v})$ by Lemma 2.9. Because this automorphism is the identity on all vertices apart from the ones whose origin is contained in the cycle, it stabilizes $\bar{u}$ and $\bar{v}$.

For the other direction, assume that there is an automorphism φ mapping one edge vertex with origin $(\mathfrak{u}, \mathfrak{v})$ to the other one. Then, in particular, φ has to exchange both. We can assume that φ is base-vertex-respecting, that is, φ maps vertices to vertices of the same origin. If φ was not base-vertex-respecting, then φ induces a nontrivial automorphism of the base graph, whose inverse can be combined with φ to a base-vertex-respecting automorphism. But a base-vertex-respecting automorphism of $(\mathfrak{A}, \bar{u}\bar{v})$ is composed of cycle-automorphisms not using the base edges E' and not using the base vertices V'. So there is a single cycle in $G - E' - V'$ containing $\{\mathfrak{u}, \mathfrak{v}\}$. $\square$

Intuitively, fixing a tuple of edge vertices corresponds to removing the origin base edges from the base graph and fixing a tuple of gadget vertices corresponds to removing the origin base vertices from the base graph. Fixing a gadget vertex is equivalent to fixing its adjacent edge vertices:

Lemma 2.12. *Let $G = (V, E, \preceq)$ be a base graph, $f\colon E \to \mathbb{F}_2$, and $\mathfrak{A}_{\mathrm{ge}} = \mathsf{CFI}_{\mathrm{ge}}(G, f)$. Then, for every gadget vertex $u \in A_{\mathrm{g}}$, we have $\mathsf{Aut}((\mathfrak{A}_{\mathrm{ge}}, u)) = \mathsf{Aut}((\mathfrak{A}_{\mathrm{ge}}, N_{\mathfrak{A}_{\mathrm{ge}}}(u))$.*

In particular, if $\preceq$ is a total order on $N_{\mathfrak{A}_{\mathrm{ge}}}(u)$, then we can equivalently fix a tuple of edge vertices instead of fixing a gadget vertex. In this way, we can also determine the 1-orbits of gadgets vertices from the 1-orbits of adjacent edge vertices. For example, for ordered base graphs, fixing one gadget vertex splits the whole gadget up into singleton 1-orbits.

Lemma 2.13. *Assume $G = (V, E, \leq)$ is an ordered base graph and let $f\colon E \to \mathbb{F}_2$ and $\mathfrak{A}_{\mathrm{ge}} = \mathsf{CFI}_{\mathrm{ge}}(G, f)$. Then, for all gadget vertices $u, v \in A_{\mathrm{g}}$ that have the same origin $\mathfrak{u} = \mathsf{orig}(u) = \mathsf{orig}(v)$, we have $\{v\} \in \mathsf{orb}((\mathfrak{A}_{\mathrm{ge}}, u))$.*

Proof. Let $u, v \in A_{\mathrm{g}}$ have the same origin $\mathfrak{u}$. We show that there is no automorphism of $(\mathfrak{A}_{\mathrm{ge}}, u)$ that maps v to any other vertex. Let $N_{\mathfrak{A}_{\mathrm{ge}}}(u) = \{w_1, \dots, w_d\}$. Because G is totally ordered, we have that $\mathsf{Aut}((\mathfrak{A}_{\mathrm{ge}}, u)) = \mathsf{Aut}((\mathfrak{A}_{\mathrm{ge}}, w_1, \dots, w_d))$ by Lemma 2.12. Let $\varphi \in \mathsf{Aut}((\mathfrak{A}_{\mathrm{ge}}, w_1, \dots, w_d))$ be arbitrary. Then v is adjacent to w_i if and only if $\varphi(v)$ is adjacent to w_i for every $i \in [d]$. Because every gadget vertex is adjacent to exactly one vertex per edge-vertex-pair of its gadget and because there are no gadget vertices with the same neighborhood, it follows that $v = \varphi(v)$. Since φ was arbitrary, the vertex v is contained in a singleton orbit. $\square$

Determining the k-orbit partition for $k \geq 2$ is more complicated. However, in the case of highly connected base graphs, this becomes easier. We sketch the proof of the following lemma to illustrate the requirement of high connectivity on the base graph.

Lemma 2.14 ([45, Lemma 3.14])**.** *Let $G = (V, E, \leq)$ be an ordered and $(k + 2)$-connected base graph and $\mathfrak{A}_{\mathrm{ge}} = \mathsf{CFI}_{\mathrm{ge}}(G, f)$ for some $f\colon E \to \mathbb{F}_2$. Assume $\bar{w} \in A_{\mathrm{ge}}^{\leq k}$ and $\{\mathfrak{u}, \mathfrak{v}\} \in E$ such that no vertex in $\bar{w}$ has origin $\mathfrak{u}$, $\mathfrak{v}$, $(\mathfrak{u}, \mathfrak{v})$, or $(\mathfrak{v}, \mathfrak{u})$. Then the two edge vertices with origin $(\mathfrak{u}, \mathfrak{v})$ form an orbit of $(\mathfrak{A}_{\mathrm{ge}}, \bar{w})$, i.e., $\{u \in A_{\mathrm{ge}} \mid \mathsf{orig}(u) = (\mathfrak{u}, \mathfrak{v})\} \in \mathsf{orb}_k((\mathfrak{A}, \bar{w}))$.*

Proof. We first assume that $\bar{w}$ only consists of gadget vertices. Let $V_{\bar{w}} \subseteq V$ be the set of origins of vertices in $\bar{w}$. Because every vertex in G has degree at least $k + 2$ (G is $(k + 2)$-connected), the vertices $\mathfrak{u}$ and $\mathfrak{v}$ have degree at least 2 in $G - V_{\bar{w}}$. Because G is $(k + 2)$-connected, there is a $\mathfrak{u}$-$\mathfrak{v}$-path in $G - V_{\bar{w}}$ not using the edge $\{\mathfrak{u}, \mathfrak{v}\}$ (removing $\mathfrak{v}$ from $G - V_{\bar{w}}$ removes at most $k + 1$ vertices from G). So there is a cycle in $G - V_{\bar{w}}$ using the edge $\{\mathfrak{u}, \mathfrak{v}\}$ and thus there is an automorphism exchanging the two edge vertices with origin $(\mathfrak{u}, \mathfrak{v})$ by Lemma 2.11.

Now assume that there is an edge vertex with origin $(\mathfrak{u}, \mathfrak{v})$ in $\bar{w}$. Let $\bar{w}'$ be obtained from $\bar{w}$ by replacing this edge vertex with a gadget vertex with origin $\mathfrak{u}$. Every automorphism stabilizing $\bar{w}'$ also stabilizes $\bar{w}$ by Lemma 2.12. $\square$

2.8.3 CFI Graphs and IFPC

CFI graphs provide difficult instances for IFPC. For a class of base graphs $\mathcal{K}$, the class of CFI graphs over $\mathcal{K}$ is

$$\mathsf{CFI}_{\mathrm{ge}}(\mathcal{K}) := \left\{ \mathsf{CFI}_{\mathrm{ge}}(G, f) \;\middle|\; G = (V, E, \preceq) \in \mathcal{K}, f \colon E \to \mathbb{F}_2 \right\}.$$

The CFI query is to decide whether a given CFI graph is even, so formally the **CFI query** for $\mathcal{K}$ is the set

$$\left\{ \mathsf{CFI}_{\mathrm{ge}}(G, f) \;\middle|\; G = (V, E, \preceq) \in \mathcal{K}, f \colon E \to \mathbb{F}_2, \sum f = 0 \right\} \subsetneq \mathsf{CFI}_{\mathrm{ge}}(\mathcal{K})$$

or, abbreviated, $\{\mathsf{CFI}_{\mathrm{ge}}(G, 0) \mid G \in \mathcal{K}\}$. The CFI query is not IFPC-definable. Cai, Fürer, and Immerman [20] showed that if every separator of a base graph G consists of at least $k+1$ many vertices, then the even and the odd CFI graphs over G are $\mathcal{C}_k$-equivalent. Later, Dawar and Richerby showed a stronger result in terms of the treewidth of the base graph.

Lemma 2.15 ([33, Theorem 3]). *If an ordered base graph G is of minimum degree 2 and has treewidth at least k, in particular, if G is k-connected, then $\mathsf{CFI}_{\mathrm{ge}}(G, 0) \equiv_{\mathcal{C}}^{k} \mathsf{CFI}_{\mathrm{ge}}(G, 1)$.*

The important consequence of Lemma 2.15 is that the CFI query for a class of base graphs of unbounded treewidth is not IFPC-definable.

Corollary 2.16. *For every class of base graph of unbounded treewidth $\mathcal{K}$,* IFPC *does not define the CFI query over $\mathcal{K}$.*

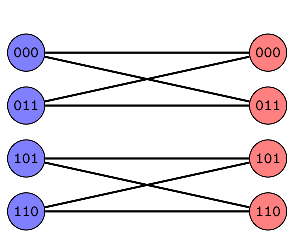
However, the CFI query is polynomial-time decidable, e.g., by reducing the CFI query to solving linear equation systems [27]. We will discuss the connection between CFI graphs and linear equation systems in more detail in Chapter 7.

Corollary 2.17. IFPC $<$ PTIME.

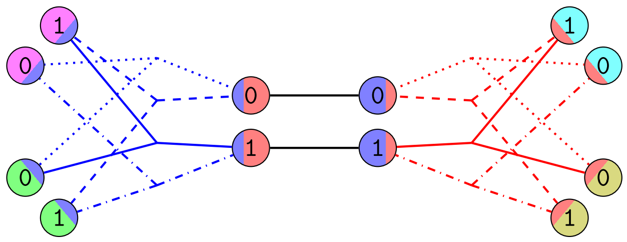
We want to note that while distinguishing CFI graphs is hard, defining the orbits of CFI graphs is very easy in some cases. For example, IFPC does not define the CFI query on a class of base graphs of unbounded connectivity by Lemma 2.15. But for ordered base graphs of connectivity 2 (in which every vertex is contained in a cycle), the 1-orbits are easily definable (and actually distinguishable in the sense of Definition 2.5) in IFPC: By Lemma 2.11, every edge-vertex-pair forms an orbit and thus the gadget vertices of the same gadget form an orbit. This means that the coloring of these CFI graphs coincide with the 1-orbit partition.

2.8.4 Variants of the CFI Construction

We now introduce two variants of the CFI construction, one only using gadget vertices and one only using edge vertices.



(a) Connecting gadget vertices directly



(b) Connecting edge vertices using hyperedges

2.2 CFI graphs with either gadget or edge vertices. This figure shows variants of the CFI construction (compare with Figure 2.1): Figure (a) shows how gadget vertices are connected directly without edge vertices. Every edge vertex is turned into a complete bipartite graph. Figure (b) shows the other alternative: Edge vertices are connected by hyperedges. Every gadget vertex is turned into a hyperedge. The different line styles and colors of the hyperedges are only used to distinguish them visually.

CFI Graphs only Using Gadget Vertices. The edge vertices in the CFI graphs considered so far make it easy to connect gadgets and are helpful for a nice presentation. However, we then have to deal with two types of vertices, which in some proofs cause unnecessary case distinctions. To eliminate these case distinctions, we now consider a variant of the CFI graphs only requiring the gadget vertices. This variant was, e.g., used by Fürer [40].

Again, let $G = (V, E, \preceq)$ be a colored base graph and $f\colon E \to \mathbb{F}_2$. We define the **CFI graph (with gadget vertices)** $\mathfrak{A}_\mathrm{g} = \mathsf{CFI}_\mathrm{g}(G, f) = (A_\mathrm{g}, E^{\mathfrak{A}_\mathrm{g}}, \preceq^{\mathfrak{A}_\mathrm{g}})$ as follows. For a base vertex $\mathfrak{u} \in G$, the set $A_\mathrm{g}^\mathfrak{u}$ is defined as before. We define (cf. Figure 2.2a)

$$\begin{aligned} A_\mathrm{g} &:= \bigcup_{\mathfrak{u}\in V} A_\mathrm{g}^\mathfrak{u}, \\ E^{\mathfrak{A}_\mathrm{g}} &:= \bigcup_{\{\mathfrak{u},\mathfrak{v}\}\in E} \Big\{ \{(\mathfrak{u},\bar{a}),(\mathfrak{v},\bar{b})\} \;\Big|\; (\mathfrak{u},\bar{a}) \in A_\mathrm{g}^\mathfrak{u}, (\mathfrak{v},\bar{b}) \in A_\mathrm{g}^\mathfrak{v}, \bar{a}(\mathfrak{v}) = \bar{b}(\mathfrak{u}) + f(\{\mathfrak{u},\mathfrak{v}\}) \Big\}. \end{aligned}$$

The total preorder $\preceq^{\mathfrak{A}_\mathrm{g}}$ is defined as before: $u \preceq^{\mathfrak{A}_\mathrm{g}} v$ if and only if $\mathsf{orig}(u) \preceq \mathsf{orig}(v)$ for every $u, v \in A_\mathrm{g}$. Alternatively, one can think of obtaining $\mathsf{CFI}_\mathrm{g}(G, f)$ from $\mathsf{CFI}_\mathrm{ge}(G, f)$ by adding an edge between all gadget vertices u and v, for which there is a path (u, w, w', v) for two edge vertices w and w', and then deleting all edge vertices (compare Figures 2.1b and 2.2a). In the CFI graph $\mathfrak{A}_\mathrm{g}$, two gadgets for adjacent base vertices $\mathfrak{u}$ and $\mathfrak{v}$ are connected by two complete bipartite graphs between the sets $M_a^\mathfrak{u} = \{u \in A_\mathrm{g}^\mathfrak{u} \mid u(\mathfrak{v}) = a\}$ and $M_b^\mathfrak{v} = \{v \in A_\mathrm{g}^\mathfrak{u} \mid v(\mathfrak{u}) = b\}$ for $a, b \in \mathbb{F}_2$. If $f(\{\mathfrak{u},\mathfrak{v}\}) = 0$, then the sets $M_0^\mathfrak{u}$ and $M_0^\mathfrak{v}$ and the sets $M_1^\mathfrak{u}$ and $M_1^\mathfrak{v}$ are connected by a complete bipartite graph. In the other case that $f(\{\mathfrak{u},\mathfrak{v}\}) = 1$, then the sets $M_0^\mathfrak{u}$ and $M_1^\mathfrak{v}$ and the sets $M_1^\mathfrak{u}$ and $M_0^\mathfrak{v}$ are connected. Figure 3.2 in Section 3.5 shows a more complex example than Figure 2.2a.

CFI Graphs only Using Edge Vertices. While CFI graphs only using gadget vertices have many nice properties, they have one drawback: Fixing one vertex of a CFI graph fixes the whole gadget of the vertex as seen in Lemma 2.13. This is not the case for the other alternative: CFI graphs that only use edge vertices (see, e.g., [66]). Here we replace gadget vertices by hyperedges. We view the resulting hypergraph as a relational structure (in the same way as we view graphs as relational structures). However, we call these hypergraphs or structures just CFI graphs. Let $G = (V, E, \preceq)$ be a colored base graph of maximal degree d and $f\colon E \to \mathbb{F}_2$. The **CFI graph (using <u>e</u>dge vertices)** $\mathfrak{A}_\mathrm{e} = \mathsf{CFI}_\mathrm{e}(G, f) = (A_\mathrm{e}, E^{\mathfrak{A}_\mathrm{e}}, R^{\mathfrak{A}_\mathrm{e}}, \preceq^{\mathfrak{A}_\mathrm{e}})$ is defined as follows (cf. Figure 2.2b):

$$\begin{aligned}
A_\mathrm{e} &:= \bigcup_{\mathfrak{u} \in V} A_\mathrm{e}^{\mathfrak{u}},\\
E^{\mathfrak{A}_\mathrm{e}} &:= \bigcup_{\{\mathfrak{u},\mathfrak{v}\} \in E} \Big\{ \{(\mathfrak{u}, \mathfrak{v}, a), (\mathfrak{v}, \mathfrak{u}, b)\} \;\Big|\; a + b = f(\{\mathfrak{u}, \mathfrak{v}\}) \Big\},\\
R^{\mathfrak{A}_\mathrm{e}} &:= \bigcup_{\mathfrak{u} \in V} \Big\{ \{(\mathfrak{u}, \mathfrak{v}_1, b_1), \dots, (\mathfrak{u}, \mathfrak{v}_{d'}, b_{d'})\} \;\Big|\; N_G(\mathfrak{u}) = \{\mathfrak{v}_1, \dots, \mathfrak{v}_{d'}\}, b_1 + \dots + b_{d'} = 0 \Big\}.
\end{aligned}$$

The total preorder $\preceq^{\mathfrak{A}_\mathrm{e}}$ is again defined as before: $u \preceq^{\mathfrak{A}_\mathrm{e}} v$ if and only if $\mathsf{orig}(u) \preceq \mathsf{orig}(v)$ for every $u, v \in A_\mathrm{e}$. Because G is of maximal degree d, every hyperedge contains at most d many vertices and $\mathfrak{A}_\mathrm{e}$ is a relational structure of arity at most d (and can be turned into a structure of arity exactly d by repeating the last entry in the tuples encoding the hyperedges). This is the major drawback of the construction only using edge vertices: If our aim is to construct a class of structures of the same signature, then the degree of the base graphs must be bounded by a constant. Hence, this construction cannot be combined with a graph class of unbounded connectivity, so in particular, of unbounded degree, for which defining k-orbits is easy (as we will see in Chapter 7).

Chapter 3

The Iteration Number of the 2-Dimensional Weisfeiler-Leman Algorithm

This chapter considers the classical 2-dimensional Weisfeiler-Leman algorithm [116] and shows that the algorithm stabilizes after at most $\mathcal{O}(n \log n)$ many iterations of the 2-dimensional Weisfeiler-Leman refinement on n-vertex graphs. By the close connection to 3-variable counting logic $\mathcal{C}_3$ (Lemma 2.6), there is, for every $\mathcal{C}_3$-formula distinguishing two n-vertex graphs, a $\mathcal{C}_3$-formula of quantifier depth at most $\mathcal{O}(n \log n)$ distinguishing the same graphs (we will actually consider binary structures in this chapter). We thereby improve the known $\mathcal{O}(n^2/\log n)$ upper bound of Kiefer and Schweitzer [79]. To prove this bound, the authors use combinatorial techniques and a case distinction into small and large vertex-color classes. In this chapter, we use an algebraic approach to show the $\mathcal{O}(n \log n)$ bound. We exploit a one-to-one correspondence between coherent configurations and coherent algebras [67]. The WL algorithm produces the former as output, whereas the latter are semisimple matrix algebras closed with respect to the Hadamard multiplication. Our upper bound matches, up to a logarithmic factor, the best known lower bound of $\Omega(n)$ by Fürer [40].

Related Work. Deep Stabilization (see [115]), developed by Weisfeiler and Leman, is a generalization of the classic 2-dimensional WL algorithm. It can in turn be seen as a restricted form of the k-dimensional WL algorithm (for a suitable k) in the sense of Babai (see [20]), which was independently introduced by Immerman and Lander [9, 73]. For each $k \in \mathbb{N}$, both generalizations give a polynomial-time algorithm that, as k increases, can distinguish more and more non-isomorphic graphs. The k-dimensional algorithm iteratively refines a coloring of all k-tuples of vertices of the graph until this process stabilizes.

While at first it was unclear whether the algorithm (for some k) solves the graph isomorphism in polynomial time, Cai, Fürer, and Immerman [20] answered this question in the negative. The close correspondence between 2-WL and $\mathcal{C}_3$ generalizes to arbitrary k, that is, k-WL and $\mathcal{C}_{k+1}$ distinguish the same graphs (or structures). So the CFI graphs, providing two non-isomorphic graphs indistinguishable by $\mathcal{C}_k$ for every k, also prove that k-WL does not solve the graph isomorphism problem for a fixed k. However, the Weisfeiler-Leman algorithm (or, more precisely, its 1-dimensional version also known as color refinement) is continuously and repeatedly applied in all state-of-the-art graph isomorphism solvers [3, 92]. Babai employs the k-dimensional WL algorithm, with k logarithmic in the input, as a subroutine in his quasi-polynomial time algorithm for graph isomorphism testing [9]. The techniques of the WL algorithm were also applied in the context of group-CSP [13] (constrained satisfaction problems in which the constraints are cosets of a group).

Regarding bounds, Berkholz and Nordström [14] proved a lower bound on the number of iterations of the k-dimensional WL algorithm for finite structures. Specifically, they show for sufficiently large k the existence of n-element relational structures distinguished by the k-dimensional WL algorithm but for which $n^{\Omega(k/\log k)}$ iterations do not suffice. The bound was recently improved to $n^{\Omega(k)}$ [55]. For a different logic, namely the 3-variable existential negation-free fragment of first-order logic, Berkholz also developed techniques to prove tight bounds [12]. In contrast to these bounds, Fürer's lower bound [40] of $\Omega(n)$ mentioned above is applicable to graphs and in fact also applies to all fixed dimensions k.

For the 1-dimensional version, the trivial lower bound of $n-1$ iterations on n-vertex graphs is tight [75]. There are more results for restricted graph classes. The k-dimensional Weisfeiler-Leman algorithm distinguishes all graphs of treewidth k [76] but needs possibly linearly many iterations. But the $(4k+3)$-dimensional algorithm needs at most logarithmically many iterations to distinguish treewidth k graphs. A similar result was shown for planar graphs: 3-dimensional Weisfeiler-Leman distinguishes all planar graphs [78] (and it is open whether the 2-dimensional algorithm suffices [77]). However, there is a constant k such that the k-dimensional algorithm distinguishes all planar graphs after logarithmically many iterations [54].

Overview of this Chapter. The 2-dimensional Weisfeiler-Leman refinement refines the color of a vertex pair (u, v) by the multiset of colors of the vertex pairs (u, w) and (w, v) for every other vertex w. That is, the WL refinement considers all walks of length 2 from u to v. Generalizing the idea of considering walks of length 2, we consider a new type of refinement, which we call the walk refinement, in Section 3.1. In one iteration it distinguishes pairs of vertices not only according to the multiset of 2-walks between them, but rather considers the multiset of all walks of arbitrary length between the two vertices. Naturally, considering all walks cannot be weaker than considering 2-walks. However, using arguments from linear algebra, it can be proved that it suffices to consider walks of bounded length. This in turn can be used to argue that the walk refinement is subsumed by a logarithmic number of traditional 2-walk refinements. In particular, the two kinds of refinement yield the same stabilization and their iteration numbers differ by at most a logarithmic factor.

The cornerstone of our argument is then to show that the number of iterations of the walk refinement is at most linear in the number of vertices in Section 3.2. This is done by observing that the result of the walk refinement corresponds to a semisimple matrix algebra. Multiple iterations of the walk refinement must therefore correspond to an increasing chain of semisimple subalgebras of a full matrix algebra. We can show that the length of such a chain is at most $\mathcal{O}(n)$, which gives a linear upper bound for the iteration number of repeated walk refinement.

Since our upper bound on the iteration number of the WL refinement is tight up to a logarithmic factor, the question arises whether the factor of $\log n$ can be removed. We show that the walk refinement requires $\Theta(n)$ iterations on the same graphs, for which

Fürer [40] showed that the WL refinement requires $\Theta(n)$ iterations. This leaves the problem open whether our method can be pushed further.

To show that the walk refinement requires $\Theta(n)$ many iterations, we follow an approach similar to the one used to show that 2-WL requires $\Theta(n)$ many iterations. We associate with the walk refinement a variant of a counting logic in Section 3.3 and a special Ehrenfeucht-Fraïssé-like Duplicator-Spoiler game in Section 3.5. We call them the bijective walk pebble game and the walk counting logic, respectively. They are suitable adaptations of the bijective 2-pebble game and the logic $\mathcal{C}_3$ that are associated with the classic Weisfeiler-Leman (2-walk) refinement. The close correspondences between aspects of refinement algorithm, game, and logic translate to our scenario (Theorem 3.21). In particular, we prove tight bounds on the length of shortest winning strategies and optimal quantifier depth, respectively, of $\Theta(n)$ in Section 3.6 (Theorem 3.29, Corollaries 3.30 and 3.31). That is, for walk-refinement the linear lower bound is tight. We end with a discussion in Section 3.6.

3.1 The Walk Refinement

We introduce a new refinement called the walk refinement (cf. Section 2.7 for the necessary preliminaries in this chapter, in particular for the formal definition of a refinement). Assume $G = (V, E, \chi)$ is a complete and colored graph, that is, $E = V^2$ and $\chi\colon E \to C$ is a function into some set C of colors, and $v_1, \ldots, v_m \in V$. Recall that

$$\overline{\chi}(v_1, \ldots, v_m) = \big(\chi(v_1, v_2), \chi(v_2, v_3), \ldots, \chi(v_{m-1}, v_m)\big)$$

from Section 2.7.

Definition 3.1 (k-Walk Refinement)**.** For $k \geq 2$, the **k-walk refinement** is the function that for a colored complete graph $G = (V, E, \chi)$ yields the new coloring $\chi_{\mathcal{W}[k]}$ defined by

$$\chi_{\mathcal{W}[k]}(u, v) := \left\{\!\!\left\{ \overline{\chi}(u, w_1, \ldots, w_{k-1}, v) \;\middle|\; w_1, \ldots, w_{k-1} \in V \right\}\!\!\right\}.$$

Intuitively, the k-walk refinement refines the color of a vertex pair (u, v) with the color sequence of the traversed vertex pairs along walks, taken over all possible walks of length k from u to v (collected in a multiset). Note that, if χ respects converse equivalence, then χ assigns different colors to loops and the refinement implicitly also refines with respect to walks of shorter lengths $k' < k$ (indeed, the information is contained in the walks whose first $k-k'$ steps are of color $\chi(u)$, i.e., which are stationary at u). So the k-walk refinement is indeed a refinement, i.e., $\chi_{\mathcal{W}[k]} \preceq \chi$, because walks of length 1 are just the old colors. It is easy to see that the k-walk refinement is isomorphism-invariant and preserves converse equivalence.

From what we just argued, obviously $\chi_{\mathcal{W}[k]} \preceq \chi_{\mathcal{W}[j]}$ if $k \geq j$. Also note that the 2-walk refinement is exactly the 2-dimensional Weisfeiler-Leman refinement. Thus, for $k \geq 2$,

$$\chi_{\mathcal{W}[k]} \preceq \chi_{\mathsf{WL}} \preceq \chi.$$

We argue next that the k-walk refinement can be simulated with a logarithmic number of Weisfeiler-Leman refinements. In this chapter, we will always use "the Weisfeiler-Leman refinement" to refer to the classic 2-dimensional Weisfeiler-Leman refinement (WL) defined in Section 2.7.

Lemma 3.2. *If $k \geq 2$, then $\chi_{\mathsf{WL}}^{\lceil \log k \rceil} \preceq \chi_{\mathcal{W}[k]}$.*

Proof. Let C be the set of colors of $\chi \colon V^2 \to C$ and likewise let C_i be the set of colors of $\chi_{\mathsf{WL}}^i \colon V^2 \to C_i$. We show by induction that after $i \geq 1$ iterations of the Weisfeiler-Leman refinement, for each color $d \in C_i$, there is a function $\hat{d} \colon C^{(2^i)} \to \mathbb{N}$ with the following property: For all $u, v \in V$ of color $\chi_{\mathsf{WL}}^i(u,v) = d$ and for all 2^i-walk colors $(c_1, \dots c_{2^i}) \in C^{(2^i)}$ in χ, there are exactly $\hat{d}(c_1, \dots, c_{2^i})$ many $(c_1, \dots, c_{2^i})$-colored walks between u and v in χ. In particular, the number of $(c_1, \dots, c_{2^i})$-colored walks is the same for all such u and v. This implies $\chi_{\mathsf{WL}}^i \preceq \chi_{\mathcal{W}[2^i]}$.

For $i = 1$, the Weisfeiler-Leman refinement assigns colors such that every color just contains the possible 2-walk colors. So assume $i > 1$. Let $u, v \in V$ be vertices, $d_1, e_1, \dots, d_n, e_n \in C_i$ be colors of χ_{WL}^i,

$$d = \{\!\!\{(d_1, e_1), \dots, (d_n, e_n)\}\!\!\} \in C_{i+1}$$

be a color of χ_{WL}^{i+1}, and $(c_1, \dots, c_{2^{i+1}}) \in C^{(2^{i+1})}$ be a $2^{(i+1)}$-walk color. We set

$$\hat{d}(c_1, \dots, c_{2^{i+1}}) := \sum_{j \in [n]} \hat{d}_j(c_1, \dots, c_{2^i}) \cdot \hat{e}_j(c_{2^i+1}, \dots, c_{2^{i+1}}).$$

By induction hypothesis, $\hat{d}_i$ and $\hat{e}_i$ yield the correct number of 2^i-walks and so $\hat{d}$ yields the correct number of 2^{i+1}-walks. $\square$

This lemma corresponds to the known fact that in $\mathcal{C}_3$ walks of length k can be defined by a formula of quantifier depth $\lceil \log k \rceil$. These walks can be counted using counting quantifiers similarly. Considering paths, we see that the k-walk refinement cannot be simulated with fewer than a logarithmic number of Weisfeiler-Leman refinements, and in that sense the bound in the lemma is tight. On the other hand the relation can be strict, that is, $\chi_{\mathsf{WL}}^{\lceil \log k \rceil} \prec \chi_{\mathcal{W}[k]}$. However, the Weisfeiler-Leman and the k-walk refinement produce the same stable partition because finitely many steps of one subsume a single step of the other.

Lemma 3.3. *If $k \geq 2$, then $\chi_{\mathsf{WL}}^{\infty} \equiv \chi_{\mathcal{W}[k]}^{\infty}$.*

Proof. Assume that $\chi_{\mathcal{W}[k]}^{\infty} \equiv \chi_{\mathcal{W}[k]}^{j}$ and $\chi_{\mathsf{WL}}^{\infty} \equiv \chi_{\mathsf{WL}}^{j}$ for some suitable j. Then

$$\chi_{\mathcal{W}[k]}^{j} \preceq \chi_{\mathsf{WL}}^{j} \equiv \chi_{\mathsf{WL}}^{j \cdot \lceil \log k \rceil} \preceq \chi_{\mathcal{W}[k]}^{j}$$

by Lemma 3.2, and hence $\chi_{\mathsf{WL}}^{\infty} \equiv \chi_{\mathcal{W}[k]}^{\infty}$. $\square$

We remark that it is possible that the partitions produced by the Weisfeiler-Leman refinement and the partitions produced by the k-walk refinement all disagree except for the stable partitions in the end (for example, this is the case for the graphs $\mathsf{CFI}_{\mathrm{g}}(G_n^2, 0)$ for $2 \leq n \leq 10$ used in Section 3.5 as shown by computer calculations with $k = n$).

We define the **walk refinement** $\chi_{\mathcal{W}}$ as the finest k-walk refinement. More precisely, we define it as $\chi_{\mathcal{W}[k]}$ for the smallest k, for which $\chi_{\mathcal{W}[k]}$ induces the finest partition over all choices of k. We will prove in Section 3.2 that the n^2-walk refinement always produces this finest partition, and thus $k \leq n^2$. From that we will conclude that $\chi_{\mathcal{W}} \equiv \chi_{\mathcal{W}[n^2]}$ and $\chi_{\mathsf{WL}}^{\mathcal{O}(\log n)} \preceq \chi_{\mathcal{W}}$. This will allow us to bound the iteration number of the Weisfeiler-Leman refinement by bounding the iteration number of the walk refinement.

3.2 Iteration Number of the Walk Refinement

In this section we show that the walk refinement stabilizes after $\mathcal{O}(n)$ iterations. We interpret the partitions produced by the walk refinement as matrix algebras. If the walk refinement strictly refines the partition, then the algebra is strictly enlarged. We obtain the linear bound by observing that these algebras can be nested at most a linear number of times. Throughout this section, let $G = (V, E, \chi)$ be a complete and colored graph with $V = [n]$ and assume that $\chi\colon E \to C$ respects converse equivalence.

3.2.1 Background on Matrix Algebras

In this section we make use of standard material from representation theory, see for example [119]. Let S be a set of $n \times n$ matrices over the complex numbers $\mathbb{C}$. We denote by $\mathbb{C}S$ the $\mathbb{C}$-linear span of S and with

$$S^{\leq k} := \left\{ M_1 \cdot \ldots \cdot M_j \;\middle|\; j \leq k, M_i \in S \text{ for all } i \in [j] \right\}$$

the set of all products of matrices in S with at most k factors. Clearly, $S^{\leq k} \subseteq S^{\leq k+1}$ for every $k \in \mathbb{N}$. We write $\widehat{S}$ for the union of all $S^{\leq k}$. For a color $c \in C$, we denote by M_c the $n \times n$ **color c adjacency matrix**, that is,

$$M_c(i,j) := \begin{cases} 1 & \text{if } \chi(i,j) = c, \\ 0 & \text{otherwise.} \end{cases}$$

The set of all color adjacency matrices is denoted by $M_\chi := \{M_c \mid c \in C\}$. The coloring χ thereby induces an $n \times n$ matrix algebra $\langle \chi \rangle$ over the complex numbers:

$$\langle \chi \rangle := \mathbb{C}\widehat{M}_\chi.$$

The algebra $\langle \chi \rangle$ is closed under (conjugate) transposition because χ respects converse equivalence.

We write $\mathsf{M}_k(\mathbb{C})$ for the (full) matrix algebra of all $k \times k$ matrices over the complex numbers. It is a well-known fact that a matrix algebra $\mathcal{A} \subseteq \mathsf{M}_n(\mathbb{C})$ closed under conjugate transposition is always semisimple. Indeed, if M is in the Jacobson radical of $\mathcal{A}$, so is M^*M. But M^*M is diagonalizable (because it is Hermitian) and nilpotent (because the radical is nilpotent, see Lemma 1.6.6 in [119]) and hence $M^*M = 0$ and so $M = 0$. Then the radical itself is 0, which is one characterization of semisimplicity. By the theorem of Wedderburn (Corollary 1.4.17 in [119]), a semisimple matrix algebra $\mathcal{A} \subseteq \mathsf{M}_n(\mathbb{C})$ is always isomorphic to a direct sum of full matrix algebras, that is,

$$\mathcal{A} \cong \mathsf{M}_{a_1}(\mathbb{C}) \oplus \cdots \oplus \mathsf{M}_{a_k}(\mathbb{C}),$$

for some positive integers k and $a_1, \ldots, a_k$. The direct sum decomposition is unique up to reordering. We will prove a bound on the length of proper chains $\mathcal{A}_1 \subsetneq \cdots \subsetneq \mathcal{A}_m \subseteq \mathsf{M}_n(\mathbb{C})$ of semisimple matrix algebras. This is the essential theorem to bound the iteration number of the walk refinement:

Theorem 3.4. *Let $\mathcal{A}_1 \subsetneq \cdots \subsetneq \mathcal{A}_m \subseteq \mathsf{M}_n(\mathbb{C})$ be a chain of semisimple strict subalgebras. Then $m \leq 2n$.*

To prove the theorem, we need several auxiliary lemmas, which may be self-evident for a reader familiar with the theory of semisimple algebras. They show that such chains behave well with respect to the direct sum decompositions of the $\mathcal{A}_i$.

Lemma 3.5. *If there is an algebra monomorphism*

$$\mathsf{M}_{a_1}(\mathbb{C}) \oplus \cdots \oplus \mathsf{M}_{a_k}(\mathbb{C}) \to \mathsf{M}_m(\mathbb{C}),$$

then $\sum_{i=1}^k a_i \leq m$ for all $a_1, \ldots, a_k, m \in \mathbb{N}$.

Proof. For each $i \in [k]$, there are exactly a_i many diagonal matrices in $\mathsf{M}_{a_i}(\mathbb{C})$ with one entry 1 and all other 0. These matrices are nonzero, idempotent, and pairwise orthogonal (i.e., the product of any two of them is 0). It follows that the direct sum, and hence the monomorphism image, contains a set of $d = \sum_{i=1}^k a_i$ many nonzero, idempotent, and pairwise orthogonal elements. Let $M_1, \ldots, M_d \in \mathsf{M}_m(\mathbb{C})$ be the matrices of this set. Since all M_i are nonzero, each has rank at least 1. Assume $i \neq j \in [d]$. Then $M_i + M_j$ is idempotent, orthogonal to all other M_ℓ, and satisfies

$$\operatorname{rank}(M_i + M_j) = \operatorname{rank}(M_i) + \operatorname{rank}(M_j)$$

(see e.g., Theorem IV.12 in [2]). Because the maximal rank of an $m \times m$ matrix is m, it follows by induction that $d = \sum_{i=1}^k a_i \leq \operatorname{rank}(\sum_{i=1}^k M_i) \leq m$. □

Lemma 3.6. *Assume $a_1, \ldots, a_k, b_1, \ldots, b_\ell \in \mathbb{N}$ and let*

$$\varphi \colon \bigoplus_{i=1}^{k} \mathsf{M}_{a_i}(\mathbb{C}) \to \bigoplus_{j=1}^{\ell} \mathsf{M}_{b_j}(\mathbb{C})$$

be an algebra monomorphism. Then for every $i \in [k]$, there is a $j \in [\ell]$ such that $\pi_j \circ \varphi$ maps $\mathsf{M}_{a_i}(\mathbb{C})$ injectively into $\mathsf{M}_{b_j}(\mathbb{C})$, where π_j is the projection onto the j-th component $\mathsf{M}_{b_j}(\mathbb{C})$.

Proof. Full matrix algebras are simple, i.e., contain no proper nontrivial two-sided ideals. For a simple algebra $\mathcal{A}$, every homomorphism $\psi \colon \mathcal{A} \to \mathcal{B}$ into some algebra $\mathcal{B}$ is either injective or zero (otherwise, $\ker(\psi)$ is a proper nontrivial two-sided ideal of $\mathcal{A}$). Assume $i \in [k]$. The map $\pi_j \circ \varphi_i \colon \mathsf{M}_{a_i}(\mathbb{C}) \to \mathsf{M}_{b_j}(\mathbb{C})$ is an algebra homomorphism for all $j \in [\ell]$, where φ_i is the restriction of φ to the i-th component $\mathsf{M}_{a_i}(\mathbb{C})$. For every $j \in [\ell]$, the homomorphism $\pi_j \circ \varphi_i$ is either injective or zero. Now, $\pi_j \circ \varphi_i$ must be injective for some j because if all $\pi_j \circ \varphi_i$ were zero, φ was not injective. □

Lemma 3.7. *Let $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathsf{M}_n(\mathbb{C})$ be two semisimple matrix algebras with direct sum decompositions*

$$\mathcal{A} \cong \mathsf{M}_{a_1}(\mathbb{C}) \oplus \cdots \oplus \mathsf{M}_{a_k}(\mathbb{C}) \text{ and}$$
$$\mathcal{B} \cong \mathsf{M}_{b_1}(\mathbb{C}) \oplus \cdots \oplus \mathsf{M}_{b_\ell}(\mathbb{C}).$$

Then $2(\sum_{i=1}^k a_i) - k \leq 2(\sum_{j=1}^\ell b_j) - \ell$ with equality exactly if $\mathcal{A} \cong \mathcal{B}$.

Proof. First, we pick an algebra monomorphism

$$\varphi\colon \bigoplus_{i=1}^{k} \mathsf{M}_{a_i}(\mathbb{C}) \to \bigoplus_{j=1}^{\ell} \mathsf{M}_{b_j}(\mathbb{C}).$$

Second, for each $i \in [k]$, we choose an $f(i) = j$ such that $\pi_j \circ \varphi$ maps $\mathsf{M}_{a_i}(\mathbb{C})$ injectively into $\mathsf{M}_{b_j}(\mathbb{C})$. Such choices exist by Lemma 3.6. For each $j \in [\ell]$, we obtain a monomorphism

$$\bigoplus_{i \in f^{-1}(j)} \mathsf{M}_{a_i}(\mathbb{C}) \to \mathsf{M}_{b_j}(\mathbb{C})$$

by restricting $\pi_j \circ \varphi$. From Lemma 3.5 it now follows that $b_j \geq \sum_{i \in f^{-1}(j)} a_i$. Then, to show that $2(\sum_{i=1}^{k} a_i) - k \leq 2(\sum_{j=1}^{\ell} b_j) - \ell$, simply observe that, for each $j \in [\ell]$, we have

$$2b_j - 1 \geq 2\Bigg(\sum_{i \in f^{-1}(j)} a_i\Bigg) - |f^{-1}(j)|$$

(since $b_j > 0$) and that summing up over all j yields the desired equation. Finally, consider the case that $\mathcal{A} \cong \mathcal{B}$ and let $j \in [\ell]$. Then

$$2b_j - 1 = 2\Bigg(\sum_{i \in f^{-1}(j)} a_i\Bigg) - |f^{-1}(j)|$$

and because $b_j \geq \sum_{i \in f^{-1}(j)} a_i$, it follows that $|f^{-1}(j)| = 1$ and $b_j = a_{f^{-1}(j)}$. Thus, f is a bijection satisfying $a_i = b_{f(i)}$ for all $i \in [k]$ and φ is an isomorphism. □

We now conclude the proof of Theorem 3.4.

Proof of Theorem 3.4. For all $i \in [m]$, assume that

$$\mathcal{A}_i \cong \bigoplus_{j=1}^{n_i} \mathsf{M}_{a_{ij}}(\mathbb{C})$$

and define $s_i := 2(\sum_{j=1}^{n_i} a_{ij}) - n_i$. Then $s_m \leq 2n - 1$ by Lemma 3.5. Moreover, for all $i \in [m-1]$, we have $s_i \leq s_{i+1}$ by Lemma 3.7, and in fact even $s_i < s_{i+1}$ since equality would imply $\mathcal{A}_i = \mathcal{A}_{i+1}$. Thus, $m \leq 2n$. □

3.2.2 Matrix Algebras and the Walk Refinement

A matrix $M \in \mathsf{M}_n(\mathbb{C})$ **distinguishes** (u_1, v_1) from (u_2, v_2) if $M(u_1, v_1) \neq M(u_2, v_2)$ and a set $S \subseteq \mathsf{M}_n(\mathbb{C})$ distinguishes (u_1, v_1) from (u_2, v_2) if S contains a matrix distinguishing them. We now show that with one iteration of the walk refinement we can distinguish the same vertex pairs as with the induced algebra $\langle \chi \rangle$. Let $c_1, c_2 \in C$ be colors. Then $(M_{c_1} \cdot M_{c_2})(u, v)$ is the number of (c_1, c_2)-colored walks from u to v. In general, let $c_1, \dots, c_k$ be colors. Then $(M_{c_1} \cdot \dots \cdot M_{c_k})(u, v)$ is the number of $(c_1, \dots, c_k)$-colored walks from u to v. Because the walk refinement and the induced algebra essentially count colored walks, they distinguish the same vertex pairs:

Lemma 3.8. *Let $u_1, v_1, u_2, v_2 \in V$. The walk refinement $\chi_{\mathcal{W}}$ distinguishes (u_1, v_1) from (u_2, v_2) if and only if the induced algebra $\langle\chi\rangle$ distinguishes them.*

Proof. On the one hand, assume that the walk refinement distinguishes the vertices, i.e., $\chi_{\mathcal{W}}(u_1, v_1) \neq \chi_{\mathcal{W}}(u_2, v_2)$. Then there is a sequence of colors $c_1, \dots, c_k$ such that the number of $(c_1, \dots, c_k)$-colored walks between u_1 and v_1 is different from the number of such walks between u_2 and v_2. Hence, $M_{c_1} \cdot \ldots \cdot M_{c_k}$ distinguishes the two vertex pairs.

On the other hand, let $M \in \langle\chi\rangle$ distinguish (u_1, v_1) and (u_2, v_2). The matrix M is a linear combination of products of color adjacency matrices:

$$M = \sum_{i=1}^{m} z_i \cdot \prod_{j=1}^{k_i} M_{c_{ij}},$$

where $z_i \in \mathbb{C}$ and $c_{ij} \in C$ for all $i \in [m]$ and $j \in [k_i]$. There must be an i such that $\prod_{j=1}^{k_i} M_{c_{ij}}$ distinguishes (u_1, v_1) and (u_2, v_2) because M distinguishes them. Hence, the number of $(c_{i1}, \dots, c_{ik_i})$-colored walks between u_1 and v_1 is different from the number of such walks between u_2 and v_2 and the pairs are distinguished by the walk refinement. $\square$

Corollary 3.9. *Either $\langle\chi\rangle \subsetneq \langle\chi_{\mathcal{W}}\rangle$ or $\chi \equiv \chi_{\mathcal{W}}$.*

The induced algebra gets strictly larger if the partition induced by the walk refinement gets strictly finer. We obtain the bound on the walk refinement iterations because the algebras can only be nested $2n$ many times.

Theorem 3.10. *The walk refinement stabilizes in $2n$ iterations.*

Proof. Assume that m is the smallest number such that $\chi_{\mathcal{W}}^m \equiv \chi_{\mathcal{W}}^\infty$. Because the walk refinement preserves converse equivalence, the algebras $\langle\chi\rangle, \langle\chi_{\mathcal{W}}\rangle, \dots, \langle\chi_{\mathcal{W}}^m\rangle$ are semisimple. Then from Corollary 3.9 it follows that

$$\langle\chi\rangle \subsetneq \langle\chi_{\mathcal{W}}\rangle \subsetneq \cdots \subsetneq \langle\chi_{\mathcal{W}}^m\rangle$$

and from Theorem 3.4 that $m \leq 2n$. $\square$

To obtain a bound on the iteration number of the Weisfeiler-Leman refinement, it remains to relate the Weisfeiler-Leman refinement and the walk refinement.

Lemma 3.11. *$\chi_{\mathcal{W}} \equiv \chi_{\mathcal{W}[n^2]}$ and $\chi_{\mathsf{WL}}^{\mathcal{O}(\log n)} \preceq \chi_{\mathcal{W}}$.*

Proof. We first show $\chi_{\mathcal{W}} \equiv \chi_{\mathcal{W}[n^2]}$. A close inspection of the proof of Lemma 3.8 shows that $\mathbb{C}M_\chi^{\leq k}$ distinguishes the same vertex pairs as the k-walk refinement. It suffices to show that $\mathbb{C}\widehat{M}_\chi = \mathbb{C}M_\chi^{\leq n^2}$, which implies $\langle\chi\rangle = \mathbb{C}M_\chi^{\leq n^2}$ and $\chi_{\mathcal{W}} \equiv \chi_{\mathcal{W}[n^2]}$. The argument is well-known: Let S be a set of $n \times n$ matrices. Clearly,

$$\dim \mathbb{C}S^{\leq k} \leq \dim \mathbb{C}S^{\leq k+1}.$$

If $\dim \mathbb{C}S^{\leq k} = \dim \mathbb{C}S^{\leq k+1}$, then $\mathbb{C}S^{\leq k} = \mathbb{C}S^{\leq j}$ for all $j \geq k$. Hence, $\mathbb{C}\widehat{S} = \mathbb{C}S^{\leq n^2}$ because the dimension can be at most n^2. Now $\chi_{\mathsf{WL}}^{\mathcal{O}(\log n)} \preceq \chi_{\mathcal{W}}$ follows by Lemma 3.2. $\square$

Theorem 3.12. *The 2-dimensional Weisfeiler-Leman algorithm stabilizes after $\mathcal{O}(n \log n)$ iterations on n-vertex graphs.*

Proof. Combining Theorem 3.10 with Lemmas 3.3 and 3.11 proves the claim. □

Corollary 3.13. *If two n-vertex graphs can be distinguished by a $\mathcal{C}_3$-sentence, then there is also a $\mathcal{C}_3$-sentence of quantifier depth at most $\mathcal{O}(n \log n)$ that distinguishes the two graphs.*

We argued that the length of the involved matrix algebras (the smallest number k such that $\mathbb{C}S^{\leq k} = \mathbb{C}\widehat{S}$) is at most n^2. We remark that there is even an $\mathcal{O}(n \log n)$ bound [111] for the length of matrix algebras. But this bound does not improve our bound on Weisfeiler-Leman iterations asymptotically.

3.3 Walk Counting Logic

The Weisfeiler-Leman refinement can distinguish the same graphs as the counting logic $\mathcal{C}_3$. Moreover, the number of Weisfeiler-Leman iterations needed to distinguish two vertex pairs equals the minimum quantifier depth of a formula distinguishing them. As we have already seen, for $k \geq 2$, the k-walk refinement distinguishes the same vertex pairs, too. But the required iterations of the walk refinement do not correspond to the quantifier depth of $\mathcal{C}_3$. We now introduce **k-walk counting logic** $\mathcal{W}_k$ for which such a correspondence holds. The logic is defined for binary structures. To relate this logic to the walk refinement, we need to switch between binary structures and their representation as edge-colored graphs (cf. Section 2.7). For the sake of readability, we identify a binary relational structure $\mathfrak{A}$ with its induced edge-colored graph $G_{\mathfrak{A}}$. For example, we refer with the initial coloring of $\mathfrak{A}$ to the initial coloring of $G_{\mathfrak{A}}$ and say that the walk refinement distinguishes two binary structures $\mathfrak{A}$ and $\mathfrak{B}$ if the walk refinement distinguishes the graphs $G_{\mathfrak{A}}$ and $G_{\mathfrak{B}}$.

A $\mathcal{W}_k$-formula can use multiple variables but has at most two free variables, which is indicated using the notation $\Phi(x, y)$. $\mathcal{W}_k$-formulas are formed from atomic 2-variable first-order formulas (so of $R(x, y)$ and $x = y$), usual boolean connectives such that the number of free variables remains two (e.g., $\Phi(x, y) \wedge \Psi(x, y)$), and the following **$k$-walk quantifier**. If $\Phi_1(x_1, x_2), \Phi_2(x_2, x_3), \dots, \Phi_k(x_k, x_{k+1})$ are $\mathcal{W}_k$-formulas such that the variables x_i are pairwise distinct, then

$$\Phi(x_1, x_{k+1}) = \exists^j (x_2, \dots, x_k). \bigwedge_{i \in [k]} \Phi_i(x_i, x_{i+1})$$

is a $\mathcal{W}_k$-formula for every $j \in \mathbb{N}$. Note that by renaming the variables in the Φ_i, we can assume that every $\mathcal{W}_k$-formula uses at most $k+1$ many variables. If the free variables of Φ are $\bar{x}$ (and hence $|\bar{x}| \leq 2$), then for a binary structure $\mathfrak{A}$, the walk quantifier Φ is satisfied by a tuple $\bar{u} \in A^{|\bar{x}|}$ if there are j many distinct tuples $(v_2, \dots, v_k) \in A^{k-1}$ satisfying the conjunction $\bigwedge_{i \in [k]} \Phi_i(x_i, x_{i+1})$ when interpreting x_i by v_i for every $2 \leq i \leq k$. We write $\Phi^{\mathfrak{A}} \subseteq A^{|\bar{x}|}$ for the set of tuples $\bar{u}$ satisfying Φ in $\mathfrak{A}$.

Note that $\mathcal{W}_k$-sentences can, for example, be obtained by setting $\Phi_1(x_1, x_2)$ to be the formula $x_2 = x_2$ and setting $\Phi_k(x_k, x_{k+1})$ to be $x_k = x_k$. This restricts the top most walk quantifier to quantify over walks of length $k-2$. The restriction could be relaxed, but that would complicate the definition and is not needed for our purpose. Syntactically, $\mathcal{W}_k$ is not a subset of $\mathcal{W}_{k+1}$, but obviously for every $\mathcal{W}_k$-formula there is an equivalent $\mathcal{W}_{k+1}$-formula. We call the union of the $\mathcal{W}_k$-logics for all $k \in \mathbb{N}$ the **walk counting logic**. Consequently, the number of variables in walk counting logic is unbounded.

Let $\mathfrak{A}$ be a binary relational τ-structure and let $\chi\colon A^2 \to C$ be a coloring of $\mathfrak{A}$. A $\mathcal{W}_k[\tau]$-formula $\Phi(x,y)$ **identifies** a color $c \in C$ in χ if $(u,v) \in \Phi^{\mathfrak{A}}$ if and only if $\chi(u,v) = c$ for all $u, v \in V$. We now show that with $\mathcal{W}_k$-formulas of quantifier depth m one can distinguish at least as many atom pairs as with m iterations of the k-walk refinement.

Lemma 3.14. *Let $\mathfrak{A}$ be a binary relational τ-structure, χ the initial coloring for $\mathfrak{A}$, and c a color produced by $\chi^m_{\mathcal{W}[k]}$. Then there is a $\mathcal{W}_k[\tau]$-formula $\Phi(x,y)$ of quantifier depth m identifying c in $\chi^m_{\mathcal{W}[k]}$. Moreover, Φ only depends on τ, $|A|$, and c (but not on $\mathfrak{A}$).*

Proof. The proof is by induction on m. If $m = 0$, then there is a set $\sigma \subseteq \tau$ such exactly the pairs (u,v) receive color c that are exactly contained in all σ-relations and for all of which it holds that either $u = v$ or $u \neq v$. This is identified by the formula

$$\bigwedge_{R\in\sigma} R(x,y) \wedge \bigwedge_{R\in\tau\setminus\sigma} \neg R(x,y) \wedge \Phi,$$

where $\Phi(x,y)$ is $x = y$ or $x \neq y$ depending on c.

Let c be a color in the $(m+1)$-th iteration. Hence, c is a multiset of k-walk colors in $\chi^m_{\mathcal{W}[k]}$. Let $(c_1,\dots,c_k)$ occur with multiplicity j in c. Then there are $\mathcal{W}_k$-formulas Φ_{c_i} of quantifier depth m identifying c_i in $\chi^m_{\mathcal{W}[k]}$ for every $i \in [m]$ by the induction hypothesis. The $\mathcal{W}_k$-formula

$$\Phi_{(c_1,\dots,c_k)}(x_1, x_{k+1}) := \exists^j x_2, \dots, x_k.\ \bigwedge_{i\in[k]} \varphi_{c_i}(x_i, x_{i+1})$$

is satisfied by atoms u and v assigned to x_1 and x_{k+1}, respectively, if and only if there are at least j many $(c_1,\dots,c_k)$-colored walks from u to v in $\chi^m_{\mathcal{W}[k]}$. Then the conjunction

$$\bigwedge_{(c_1,\dots,c_k)\in c} \Phi_{(c_1,\dots,c_k)}$$

identifies c in $\chi^{m+1}_{\mathcal{W}[k]}$ and is of quantifier depth $m+1$. □

Similar to counting logics and the WL-refinement, when one is interested in distinguishing structures rather than distinguishing atom pairs, one (sometimes) needs an additional quantifier. This is the case because a refinement distinguishes two graphs after m applications if the multisets of colors of both graphs are different. Hence, there is a hidden quantifier saying that there is a color, that occurs with different multiplicity in both graphs.

Lemma 3.15. *If m iterations of the k-walk refinement distinguish two binary relational structures $\mathfrak{A}$ and $\mathfrak{B}$, then a $\mathcal{W}_k$-sentence of quantifier depth $m+1$ (or $m+2$ if $k=2$) distinguishes $\mathfrak{A}$ and $\mathfrak{B}$.*

Proof. Assume m iterations of the k-walk refinement distinguish $\mathfrak{A}$ and $\mathfrak{B}$ and let $\chi_{\mathfrak{A}}$ and $\chi_{\mathfrak{B}}$ be the coloring obtained for $\mathfrak{A}$ and $\mathfrak{B}$, respectively. Then there is a color c occurring for a different number of atom pairs, say n_1 and n_2 with $n_1 > n_2$, in $\chi_{\mathfrak{A}}$ and $\chi_{\mathfrak{B}}$, respectively. Let $\Phi(x, y)$ be the $\mathcal{W}_k$-formula from Lemma 3.14 of quantifier depth m that identifies atom pairs of color c in both colorings. Now, for $k > 2$, the $\mathcal{W}_k$-formula $\exists^{n_1} x, y.\, \Phi(x, y)$ is of quantifier depth $m + 1$ and distinguishes the structures.

Now assume that $k = 2$ (and hence the prior formula is not a valid $\mathcal{W}_2$-formula). Let D be the multiset of c-outdegrees of all atoms of $\mathfrak{A}$, that is, for an atom u, the number of atoms v such that $\chi_{\mathfrak{A}}(u, v) = c$. Then the sum of all c-outdegrees (respecting the multiplicity) is n_1. Let $\mathcal{D}$ be the set of all possible c-outdegree multisets with sum n_1. Then the $\mathcal{W}_2$-formula

$$\bigvee_{D \in \mathcal{D}} \bigwedge_{(i,d) \in D} \exists^i x.\ \exists^d y.\ \Phi(x, y)$$

distinguishes $\mathfrak{A}$ and $\mathfrak{B}$, where $(i, d) \in D$ says that d occurs with multiplicity i in D. □

3.4 The Bijective Walk Pebble Game

We now describe a game called the **bijective k-walk pebble game**, which corresponds to the k-walk refinement and k-walk counting logic. It is an adaptation of the bijective 3-pebble game to agree with the k-walk refinement.

There are two players, Spoiler and Duplicator. The game is played on two binary relational τ-structures $\mathfrak{A}$ and $\mathfrak{B}$. Spoiler obtains $k+1$ pairs of pebbles $(p_1, q_1), \dots, (p_{k+1}, q_{k+1})$ labeled with numbers 1 to $k + 1$. We say that the pebble pairs (p_i, q_i) and (p_{i+1}, q_{i+1}) for all $i \in [k]$ and the pairs (p_{k+1}, q_{k+1}) and (p_1, q_1) are **consecutive**. Given a pebble pair (p_i, q_i), we will for simplicity write (p_{i+1}, q_{i+1}) for the next consecutive pebble pair, in particular, in the case $i = k + 1$, where (p_1, q_1) is meant. A position in the game is a pair $(\mathfrak{A}, \bar{u}; \mathfrak{B}, \bar{v})$ for $\bar{u} \in A^{\leq k+1}$, $\bar{v} \in B^{\leq k+1}$, and $|\bar{u}| = |\bar{v}|$. The pebble p_i is placed on u_i and the pebble q_i is placed on v_i for every $i \in [|\bar{u}|]$. In the initial positions, the tuples $\bar{u}$ and $\bar{v}$ are empty. If $|A| \neq |B|$, Spoiler wins immediately. One round consists of the following three moves:

1. If there are pebbles already placed on the structures, Spoiler can choose a pair of pebbles (p_i, q_i), replace it with (p_1, q_1) and then must also replace the next consecutive pair (p_{i+1}, q_{i+1}), if it is placed on the graph, with (p_{k+1}, q_{k+1}). In either case, Spoiler then picks up all pebble pairs apart from the first and last.

2. Duplicator chooses a bijection $\lambda \colon A^{k-1} \to B^{k-1}$.

3. Spoiler places the pebbles p_i for $2 \leq i \leq k$ onto atoms of $\mathfrak{A}$, where Spoiler may place multiple pebbles on the same atom. Assume pebble p_i is placed onto atom u_i and $\lambda(u_2, \dots, u_k) = (v_2, \dots, v_k)$. Then, Spoiler also places the pebbles q_i onto v_i for all $2 \leq i \leq k$.

Thus, as opposed to a bijection between atoms in the classic game, Duplicator chooses a bijection from the k-walks in $\mathfrak{A}$ from p_1 to p_{k+1} to the k-walks from q_1 to q_{k+1} in $\mathfrak{B}$ (hence the name walk pebble game).

We say that Spoiler wins the game after the m-th round, if there are consecutive pebble pairs (p_i, q_i) and (p_{i+1}, q_{i+1}) placed on atoms $(u, v) \in A^2$ and $(u', v') \in B^2$, such that the induced substructures $(\mathfrak{A}[u, v], uv)$ and $(B[u'v'], u'v')$ are not isomorphic. Duplicator wins the game if Spoiler never wins the game. Spoiler can **force a win after the m-th round** or has **a winning strategy in m rounds** if Spoiler can always win the game after the m-th round for all possibles moves of Duplicator. With **bijective walk pebble game** we refer to the game in which Spoiler is allowed to choose the number k in the beginning.

Note that, similar to the toplevel quantifier of a $\mathcal{W}_k$-sentence, in the first round only $k - 1$ many pebbles are placed on the graph and hence they describe a $(k - 2)$-walk. This also ensures in the case $k = 2$ that the game becomes the bijective 3-pebble game (besides some irrelevant replacements of pebble pairs).

In the following, let $\mathfrak{A}$ and $\mathfrak{B}$ be two arbitrary but fixed binary τ-structures. Furthermore, let $u, v \in A$, and $u', v' \in B$. We say that the bijective k-walk pebble game **distinguishes** (u, v) from (u', v') in m rounds if Spoiler has a winning strategy in m rounds in the game in position $(\mathfrak{A}, uv; \mathfrak{B}, u'v')$. If uv and $u'v'$ do not induce a local isomorphism, then they are distinguished in 0 rounds.

Lemma 3.16. *Let $u, v \in A$ and $u', v' \in B$. If there is a $\mathcal{W}_k$-formula $\Phi(x, y)$ of quantifier depth m that distinguishes (u, v) from (u', v'), then they are also distinguished by the bijective k-walk pebble game in m rounds.*

Proof. Assume that $\Phi(x, y)$ distinguishes (u, v) from (u', v') and that the pebble pairs (p_1, q_1) and (p_{k+1}, q_{k+1}) are placed on these atoms. The proof proceeds by induction on m. If $m = 0$, $\Phi(x, y)$ is quantifier free, hence (u, v) is contained in a relation in which (u', v') is not contained (or vice versa), hence the two induced substructures are not isomorphic and Spoiler wins the game immediately.

Assume $\Phi(x, y)$ has quantifier depth $m + 1$. If $\Phi = \neg\Psi$, then Ψ distinguishes (u, v) from (u', v'), too. If $\Phi = \Psi_1 \wedge \Psi_2$, one formula of Ψ_1 and Ψ_2 distinguishes (u, v) from (u', v'). Hence, we can assume that Φ is a walk quantifier:

$$\Phi(x_1, x_{k+1}) = \exists^j x_2, \dots, x_k. \bigwedge_{i\in[k]} \Phi_i(x_i, x_{i+1}).$$

Assume w.l.o.g. that $(u, v) \in \Phi^{\mathfrak{A}}$ and $(u', v') \notin \Phi^{\mathfrak{B}}$. Duplicator chooses a bijection $\lambda\colon A^{k-1} \to B^{k-1}$. There must be a tuple $(w_2, \dots, w_k) \in A^{k-1}$ serving as witness of the quantifier in $\mathfrak{A}$ but $\lambda(w_2, \dots, w_k) = (w'_2, \dots, w'_k)$ does not serve as witness for $\mathfrak{B}$ because otherwise $(u', v') \in \Phi^{\mathfrak{B}}$. Then Spoiler places for $2 \le i \le k$ the pebble p_i on w_i and the pebble q_i on w'_i. There must be an $i \in [k]$ such that $(w_i, w_{i+1}) \in \Phi_i^{\mathfrak{A}}$ and $(w'_i, w'_{i+1}) \notin \Phi_i^{\mathfrak{B}}$ because otherwise $(w'_2, \dots, w'_k)$ is a witness. The formula Φ_i is of quantifier depth m and the i-th and $(i + 1)$-th pebble pairs are placed on the correct atoms. Thus, Spoiler removes all other pebbles and wins the game in m additional rounds by the induction hypothesis. □

Lemma 3.17. *If there is a $\mathcal{W}_k$-sentence Φ of quantifier depth m distinguishing $\mathfrak{A}$ and $\mathfrak{B}$, then Spoiler has a winning strategy in m rounds in the bijective k-walk pebble game played on $\mathfrak{A}$ and $\mathfrak{B}$.*

Proof. Assume that Φ is a sentence of quantifier depth $m > 0$ which distinguishes the structures $\mathfrak{A}$ and $\mathfrak{B}$. For the same reasons as in Lemma 3.16, we can assume that Φ is a walk-quantifier:

$$\Phi = \exists^j x_1, \dots, x_{k-1}. \bigwedge_{i \in [k-2]} \Phi_i(x_i, x_{i+1}).$$

Again as in Lemma 3.16, for each bijection $\lambda\colon A^{k-1} \to B^{k-1}$, there exists a witness $(w_1, \dots, w_{k-1}) \in A^{k-1}$ for $\mathfrak{A}$ such that $\lambda(w_1, \dots, w_{k-1}) = (w'_1, \dots, w'_{k-1})$ is not a witness for $\mathfrak{B}$. Again, there is an $i \in [k-2]$ such that Φ_i distinguishes (w_i, w_{i+1}) and (w'_i, w'_{i+1}). When Spoiler places the pebbles p_i on the w_i and the pebbles q_i on the w'_i, then Spoiler can force a win in $m-1$ additional rounds by Lemma 3.16. So overall Spoiler has a winning strategy in m rounds. □

Lemma 3.18. *Let $u, v \in A$ and $u', v' \in B$. If the bijective k-walk pebble game distinguishes (u, v) from (u', v') in m rounds, then m iterations of the k-walk refinement distinguish them in $\mathfrak{A}$ and $\mathfrak{B}$, respectively.*

Proof. Assume that the pebble pairs (p_1, q_1) and (p_{k+1}, q_{k+1}) are placed on (u, v) and (u', v') and Spoiler has a winning strategy in m additional rounds. Let $\chi_{\mathfrak{A}}$ and $\chi_{\mathfrak{B}}$ be the initial colorings of $\mathfrak{A}$ and $\mathfrak{B}$. The proof proceeds by induction on m. If $m = 0$, then the pair (u, v) induces a substructure not isomorphic to the one induced by (u', v'). Hence, $\chi(u, v) \neq \chi'(u', v')$.

Assume Spoiler can force a win in the game in $m + 1$ additional rounds. Whatever bijection Duplicator chooses, Spoiler can place the pebbles forcing a win in m additional rounds. This means, by the induction hypothesis, that for every bijective mapping between the k-walks from u to v in $(\chi_{\mathfrak{A}})^m_{\mathcal{W}[k]}$ and the k-walks from u' to v' in $(\chi_{\mathfrak{B}})^m_{\mathcal{W}[k]}$, there is a walk that is mapped to a walk of a different color. Hence, there is a k-walk color that occurs with different multiplicity from u to v in $(\chi_{\mathfrak{A}})^m_{\mathcal{W}[k]}$ than from u' to v' in $(\chi_{\mathfrak{B}})^m_{\mathcal{W}[k]}$. This just says that the atom pairs obtain different colors in $(\chi_{\mathfrak{A}})^m_{\mathcal{W}[k]}$ or $(\chi_{\mathfrak{B}})^m_{\mathcal{W}[k]}$, respectively. □

Lemma 3.19. *If Spoiler can force a win in the bijective k-walk pebble game played on $\mathfrak{A}$ and $\mathfrak{B}$ in m rounds, then the k-walk refinement distinguishes $\mathfrak{A}$ and $\mathfrak{B}$ after $m - 1$ iterations.*

Proof. At the beginning of the game, Duplicator chooses a bijection. For every such bijection, Spoiler can place the pebbles forcing a win in $m - 1$ additional rounds. As in and by Lemma 3.18, there is no bijective mapping between the walks in $\mathfrak{A}$ and those in $\mathfrak{B}$ such that assigned walks have the same color after $m - 1$ iterations of the k-walk refinement. This implies that the multiset of colors after $m - 1$ iterations of the k-walk refinement are different, that is, the k-walk refinement distinguishes $\mathfrak{A}$ and $\mathfrak{B}$ after $m - 1$ iterations. □

Theorem 3.20. *For all binary τ-structures $\mathfrak{A}$ and $\mathfrak{B}$, $u_1, u_2 \in A$, and $v_1, v_2 \in B$, the following are equivalent:*

1. *(u_1, u_2) and (v_1, v_2) are distinguished by m iterations of the k-walk refinement on $\mathfrak{A}$ and $\mathfrak{B}$.*
2. *$(\mathfrak{A}, u_1u_2)$ and $(\mathfrak{B}, v_1v_2)$ are distinguished by a k-walk counting logic $\mathcal{W}_k[\tau]$-formula of quantifier depth m.*
3. *Spoiler has a winning strategy in the m-round bijective k-walk pebble game in position $(\mathfrak{A}, u_1u_2; \mathfrak{B}, v_1v_2)$.*

Proof. The claim follows immediately from Lemmas 3.14, 3.16, and 3.18. □

To distinguish structures (instead of atoms of structures), one needs one additional quantifier or round compared to the walk-refinement. This is similar to the case of $\mathcal{C}_3$ and the Weisfeiler-Leman algorithm (but note that $\mathcal{C}_3$ needs up to 2 additional quantifiers).

Theorem 3.21. *For all binary τ-structures $\mathfrak{A}$ and $\mathfrak{B}$, the following are equivalent:*

1. *$\mathfrak{A}$ and $\mathfrak{B}$ are distinguished by m iterations of the k-walk refinement.*
2. *$\mathfrak{A}$ and $\mathfrak{B}$ are distinguished by a k-walk counting logic $\mathcal{W}_k[\tau]$-sentence of quantifier depth $m+1$.*
3. *Spoiler has a winning strategy in the $(m+1)$-round bijective k-walk pebble played on $\mathfrak{A}$ and $\mathfrak{B}$.*

Proof. The claim follows immediately from Lemmas 3.15, 3.17, and 3.19. □

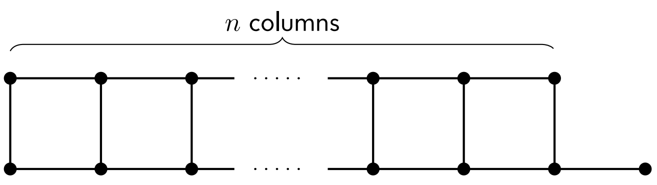
Corollary 3.22. *For all binary τ-structures $\mathfrak{A}$ and $\mathfrak{B}$, the following are equivalent:*

1. *$\mathfrak{A}$ and $\mathfrak{B}$ are distinguished by the walk refinement,*
2. *$\mathfrak{A}$ and $\mathfrak{B}$ are distinguished by walk counting logic, and*
3. *Spoiler has a winning strategy in the bijective walk pebble game played on $\mathfrak{A}$ and $\mathfrak{B}$.*

In particular, these equivalences imply that the upper bound for the walk refinement (Theorem 3.10) translates to the game and logic scenarios as follows.

Corollary 3.23. *For all binary τ-structures $\mathfrak{A}$ and $\mathfrak{B}$ of order $|A| = |B| = n$, if Spoiler has a winning strategy in the bijective walk-pebble game on the two structures $\mathfrak{A}$ and $\mathfrak{B}$, then Spoiler has a winning strategy requiring $\mathcal{O}(n)$ rounds.*

Corollary 3.24. *For all binary τ-structures $\mathfrak{A}$ and $\mathfrak{B}$ of order $|A| = |B| = n$, if the structures $\mathfrak{A}$ and $\mathfrak{B}$ are distinguished by walk counting logic, then they can be distinguished by a walk counting logic sentence of quantifier depth $\mathcal{O}(n)$.*



3.1 The base graph G_n^2. Every vertex in the graph is identified by its distance to the degree 1 vertex at the right and the two degree 2 vertices on the left.

3.5 A Linear Lower Bound for the Walk Refinement

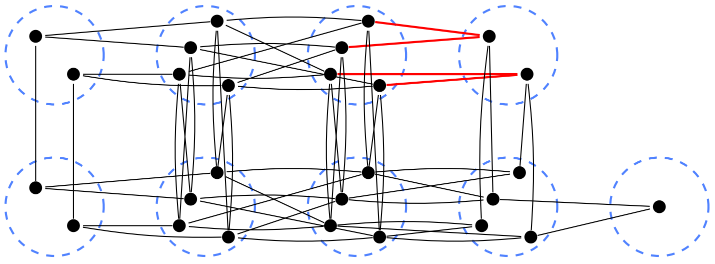
In this section we show that there are graphs on which the walk refinement stabilizes only after $\Omega(n)$ iterations. Specifically, we show this for the same graphs for which Fürer already showed that the WL refinement requires $\Omega(n)$ iterations [40]. We do this by demonstrating that Duplicator has a strategy in the bijective walk pebble game played on these graphs that delays the win of Spoiler for at least $\Omega(n)$ rounds. The graph class used by Fürer are CFI graphs over grids of height 2.

3.5.1 Lower Bound for the Weisfeiler-Leman Refinement

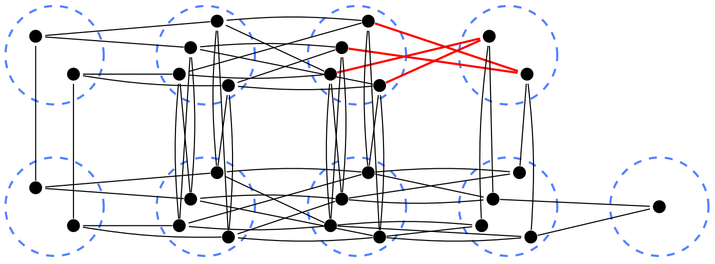
We recall the necessary parts of Fürer's lower bound on the iteration number of the d-dimensional WL refinement. We only deal with the 2-dimensional case. Let $n \geq 3$ be arbitrary but fixed and let G_n^2 be the $2 \times n$ grid with an additional vertex attached to one corner (depicted in Figure 3.1). In this graph all vertices can be uniquely identified by their distance to the unique vertex of degree 1 as well as the distance to the two adjacent vertices of degree 2 (because $n \geq 3$). Indeed, all vertices of G_n^2 are $\mathcal{C}_3$-distinguishable. So we actually can think of G_n^2 as being totally ordered.

We consider the CFI construction only using gadget vertices (cf. Section 2.8.4). The graph $\mathsf{CFI}_{\mathrm{g}}(G_4^2, 0)$ is shown in Figure 3.2. The graphs $\mathsf{CFI}_{\mathrm{g}}(G_n^2, 0)$ and $\mathsf{CFI}_{\mathrm{g}}(G_n^2, 1)$ are not isomorphic and can be distinguished by the (2-dimensional) Weisfeiler-Leman refinement. Hence, Spoiler has a winning strategy in the bijective 3-pebble game and consequently also in the bijective walk pebble game. Note that, since vertices of G_n^2 have degree at most 3, every gadget has size at most 4. Thus, $\mathsf{CFI}_{\mathrm{g}}(G_n^2, 0)$ and $\mathsf{CFI}_{\mathrm{g}}(G_n^2, 1)$ have $\Theta(n)$ many vertices.

We recall some facts for the bijective 3-pebble game played on CFI graphs over these grids [40]. Intuitively, since Duplicator can move the twist to different base edges using isomorphisms (Lemma 2.9), Spoiler needs to "catch" the twist with the pebbles. If a pebble p_i is placed on a vertex u with origin $\mathfrak{u}$ in $\mathsf{CFI}_{\mathrm{g}}(G_n^2, 0)$, then all pebble-respecting automorphisms fix all vertices with origin $\mathfrak{u}$ (by Lemma 2.13). If q_i is placed on a vertex u' with the same origin $\mathfrak{u}$ in $\mathsf{CFI}_{\mathrm{g}}(G_n^2, 1)$, then every pebble-respecting isomorphism has to map u to u'. But also, the image of every vertex with origin $\mathfrak{u}$ is determined by mapping u to u' (also as a consequence of Lemma 2.13). Hence, it does not matter on which vertex originating from $\mathfrak{v}$ Spoiler places a pebble and we can simply say that Spoiler places a pebble on $\mathfrak{v}$.



(a) The even CFI graph $\mathsf{CFI}(G_4^2, f)$ where f assigns 0 to every base edge



(b) The odd CFI graph $\mathsf{CFI}(G_4^2, g)$ where g assigns 1 to the highlighted base edge

3.2 CFI graphs over grids. The figure shows two CFI graphs over the base graph G_4^2. A dashed cycle indicates a base vertex and hence a CFI gadget. Figure (a) shows an even CFI graph and (b) an odd one. Between them, the highlighted base edge is twisted.

The general strategy of Spoiler is to maintain a pebbled separator in the base graph. Through such a separator, Duplicator cannot move the twist because these gadgets are fixed as explained.

Definition 3.25 (Wall)**.** A set of vertices of $\mathsf{CFI}_{\mathrm{g}}(G_n^2, a)$ (for $a \in \mathbb{F}_2$) is called a **wall** if its origin is a separator of G_n^2, that is, a set of vertices whose removal separates the graph into at least two connected components.

We say that Spoiler **builds a wall** if the vertices covered by the pebbles form a wall. To avoid a quick win for Spoiler, Duplicator picks the bijection so that it is origin-respecting. That is, Duplicator maps a vertex u to a vertex u' with the same origin. Our strategy for Duplicator in the bijective walk-pebble game described below has this property, too. Consequently, when asking whether the pebbles form a wall or not, it does not matter whether we consider the pebbles on $\mathsf{CFI}_{\mathrm{g}}(G_n^2, 0)$ or $\mathsf{CFI}_{\mathrm{g}}(G_n^2, 1)$. Since we will only consider origin-respecting strategies, we will often simply think of the base vertices as being pebbled.

Let V' be a set of base vertices of G_n^2. A **component** of the graph G_n^2, with respect to V', is an inclusion-wise maximal and nonempty set of base edges C satisfying the following property: For all base edges $\mathfrak{e}_1, \mathfrak{e}_2 \in C$, there is a path $(\mathfrak{v}_1, \dots, \mathfrak{v}_j)$ only using base edges $\{\mathfrak{v}_i, \mathfrak{v}_{i+1}\} \in C$ for all $i \in [j-1]$ such that $\mathfrak{e}_1 = \{\mathfrak{v}_1, \mathfrak{v}_2\}$, $\mathfrak{e}_2 = \{\mathfrak{v}_{j-1}, \mathfrak{v}_j\}$, and $\mathfrak{v}_2, \dots, \mathfrak{v}_{j-1} \notin V'$. A component C **contains** a base vertex $\mathfrak{v}$ if all base edges incident to $\mathfrak{v}$ are contained in C. The **size** of the component is the number of vertices it contains. A component is **nontrivial** if its size is nonzero.

We are interested in components with respect to the pebbled vertices. Intuitively, one can think of components as the parts of the graph G_n^2 obtained by deleting only the vertices covered by pebbles, but not the edges incident to these vertices. This results in "dangling" edges in nontrivial components (edges, which were incident to only one vertex covered by a pebble) and edges not incident to any vertex forming the trivial components (edges, whose both endpoints are covered by a pebble). By Lemma 2.9, the component C containing a base edge $\mathfrak{e}$ contains exactly all the base edges to which a twist from $\mathfrak{e}$ can be moved using pebble-respecting isomorphisms. Hence, the parity of twists in a component is invariant under pebble-respecting isomorphisms.

Definition 3.26 (Twisted Component)**.** Let $f, g\colon E \to \mathbb{F}_2$ and V' be a set of base vertices of G_n^2. We call a component C **twisted** with respect to f, g, and V' if C is a component with respect to V' and contains an odd number of base edges twisted by f and g.

With 3 pebbles, Spoiler can build at most one wall and hence in the bijective 3-pebble game there are at most two nontrivial components. Hence, there is exactly one twisted component. When a trivial component is twisted, Spoiler wins the game. To delay the win of Spoiler, Duplicator maintains a single twisted component, whose size only decreases by a constant per round.

3.5.2 Lower Bound for the Walk Refinement

The situation changes in the bijective walk pebble game, since the game does not have a bound on the number of pebbles that are used. In the situation where more than two pebbles are placed on the graph, there are possibly many components (Spoiler may cover every vertex and every edge). But, once Spoiler has to remove all but two pebbles, there can be at most one wall again. Let $n \geq 3$ and $k \geq 2$ be arbitrary but fixed, $G_n^2 = (V, E)$, and $\mathfrak{A}_f := \mathsf{CFI}_{\mathrm{g}}(G_n^2, f)$ for every $f\colon E \to \mathbb{F}_2$. Recall that CFI graphs over the same base graph always have the same vertex set A independent of f. We describe a strategy of Duplicator in the bijective k-walk pebble game played on $\mathsf{CFI}_{\mathrm{g}}(G_n^2, 0)$ and $\mathsf{CFI}_{\mathrm{g}}(G_n^2, 1)$ with the following properties:

1. If the size of the twisted component is at most $n-2$, its size reduces by at most 2 after one round.

2. If the size of the twisted component is greater than $n-2$ (e.g., in the beginning of the game), the size of the twisted component is at least $n-4$ after one round.

Combining these properties, Spoiler needs at least $\Omega(n)$ many rounds to win. Intuitively, the existence of such a strategy comes from the fact that in the bijective walk pebble game Duplicator needs to preserve adjacency and/or equality only for consecutive pebble pairs. This allows Duplicator to fix the twist locally, possibly introducing global inconsistencies but never introducing inconsistencies between consecutive pebble pairs. We describe a strategy how Duplicator can introduce these inconsistencies only on edges incident to a single chosen base vertex.

Let $\mathfrak{v} \in V$ be a base vertex of degree 3 and let $\mathfrak{e}_1$ and $\mathfrak{e}_2$ be two distinct base edges incident to $\mathfrak{v}$. Furthermore, let $u_1, u_2 \in A$. We define a function $\lambda_{\mathfrak{e}_1\mathfrak{e}_2}^{u_1u_2}\colon A^{k-1} \to A^{k-1}$ as follows (note that $\mathfrak{e}_1$ and $\mathfrak{e}_2$ determine their common incident base vertex $\mathfrak{v}$ uniquely): Let $(w_2, \dots, w_k) \in A^{k-1}$ and set $w_1 := u_1$ and $w_{k+1} := u_2$. We define

$$\lambda_{\mathfrak{e}_1\mathfrak{e}_2}^{u_1u_2}(w_2, \dots, w_k) := (w'_2, \dots, w'_k)$$

entry-wise for $2 \leq i \leq k$ by the following case distinction. Set $\mathfrak{w}_i := \mathsf{orig}(w_i)$ for every $i \in [k+1]$.

1. If $\mathfrak{w}_i \neq \mathfrak{v}$, we set $w'_i := w_i$.
2. If $\mathfrak{w}_i = \mathfrak{v}$, let $j < i$ and $\ell > i$ be the unique indices such that

   $$\mathfrak{w}_j \neq \mathfrak{w}_{j+1} = \dots = \mathfrak{w}_i = \dots = \mathfrak{w}_{\ell-1} \neq \mathfrak{w}_\ell.$$

   We pick a base edge $\mathfrak{e}$ incident to $\mathfrak{v}$ by a second case distinction:

   (a) If $\mathfrak{e}'_1 := \{\mathfrak{w}_j, \mathfrak{w}_i\}$ and $\mathfrak{e}'_2 := \{\mathfrak{w}_i, \mathfrak{w}_\ell\}$ are distinct base edges, let $\mathfrak{e} \notin \{\mathfrak{e}'_1, \mathfrak{e}'_2\}$ be the third base edge incident to $\mathfrak{v}$.

   (b) Otherwise, $\{\mathfrak{w}_j, \mathfrak{w}_i\} = \{\mathfrak{w}_\ell, \mathfrak{w}_i\}$ or at least one of $\{\mathfrak{w}_j, \mathfrak{w}_i\}$ and $\{\mathfrak{w}_i, \mathfrak{w}_\ell\}$ is not a base edge. Hence, when passing through $\mathfrak{v}$ at position i, the walk $(\mathfrak{w}_1, \dots, \mathfrak{w}_{k+1})$ uses at most one base edge $\mathfrak{e}'$ incident to $\mathfrak{v}$. Let the edge $\mathfrak{e} \in \{\mathfrak{e}_1, \mathfrak{e}_2\} \setminus \{\mathfrak{e}'\}$ be the smallest one according to the $\mathcal{C}_3$-definable order of base vertices (and thus edges) incident to $\mathfrak{v}$ (the actual order does not matter as long as we always use the same one).

   Finally, let ψ be the unique isomorphism $\psi\colon \mathsf{CFI}_{\mathrm{g}}(G_n^2, g) \to \mathsf{CFI}_{\mathrm{g}}(G_n^2, g')$ for some $g, g'\colon E \to \mathbb{F}_2$ such that g and g' twist exactly the edges $\mathfrak{e}_1$ and $\mathfrak{e}$ and ψ is the identity on all vertices apart from the ones with origin $\mathfrak{v} = \mathfrak{w}_i$ (which exists by Lemma 2.9). Note that ψ is equal for all such g and g'. We set $w'_i := \psi(w_i)$.

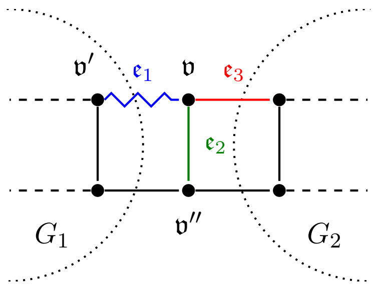
Lemma 3.27. *The function $\lambda_{\mathfrak{e}_1\mathfrak{e}_2}^{u_1u_2}$ is a bijection for all $u_1, u_2 \in A$ and all distinct base edges $\mathfrak{e}_1$ and $\mathfrak{e}_2$ incident to the same degree 3 base vertex $\mathfrak{v}$.*

Proof. The function $\lambda_{\mathfrak{e}_1\mathfrak{e}_2}^{u_1u_2}$ maps a k-walk to a k-walk with the same origin. For two k-walks with the same origin, the construction of $\lambda_{\mathfrak{e}_1\mathfrak{e}_2}^{u_1u_2}$ uses the same isomorphism ψ. Since all chosen ψ are bijections, the map $\lambda_{\mathfrak{e}_1\mathfrak{e}_2}^{u_1u_2}$ is a bijection. □

Lemma 3.28. *Let $\mathfrak{v} \in V$ be a base vertex of degree 3, let $\mathfrak{e}_1, \mathfrak{e}_2 \in E$ be distinct base edges incident to $\mathfrak{v}$, let $f, g\colon E \to \mathbb{F}_2$ such that $\mathfrak{e}_1$ is the only base edge twisted by f and g, and let $u_1, u_2 \in A$ such that $\mathfrak{v}$ is in the twisted component (with respect to f, g, u_1, and u_2). Consider the bijective k-walk pebble game in position $(\mathfrak{A}_f, u_1u_2; \mathfrak{A}_g, u_1u_2)$ when it is Duplicator's turn to choose a bijection (in particular, we assume that the pebbles induce a local isomorphism). If Duplicator chooses $\lambda_{\mathfrak{e}_1\mathfrak{e}_2}^{u_1u_2}$ as bijection, then for all $(w_2, \dots, w_k) \in A^{k-1}$ and $\lambda_{\mathfrak{e}_1\mathfrak{e}_2}^{u_1u_2}(w_2, \dots, w_k) = (w'_2, \dots, w'_k)$ picked and pebbled by Spoiler, the following holds:*

(a) Spoiler does not win in this round.

(b) For every $i \in [k]$, there is an isomorphism $\varphi_i\colon \mathfrak{A}_g \to \mathfrak{A}_{g'_i}$ satisfying $\varphi_i(w'_j) = w_j$ for every $j \in \{i, i+1\}$.



3.3 The situation in Lemma 3.28(c). Dashed edges may exist but do not have to and the two subgraphs G_1 and G_2 are indicated by dotted lines. The base edge $\mathfrak{e}_1$ is twisted.

(c) *Assume $\mathfrak{e}_1 = \{\mathfrak{v}, \mathfrak{v}'\}$ and $\mathfrak{e}_2 = \{\mathfrak{v}, \mathfrak{v}''\}$ such that $\{\mathfrak{v}, \mathfrak{v}''\}$ is a separator of G_n^2 separating G_n^2 into two subgraphs G_1 and G_2, $\mathfrak{v}'$ is contained in G_1, and G_2 has ℓ many vertices. Then for every $i \in [k]$, the twisted component with respect to f, g_i', w_i, and w_{i+1} has size at least* $\min\{\ell, 2n - \ell - 2\}$.

Proof. Note that $\mathfrak{v} \notin \{\mathsf{orig}(u_1), \mathsf{orig}(u_2)\}$ because $\mathfrak{v}$ is in the twisted component. We first prove Part (a). Assume that Spoiler picks the walk $(w_2, \dots, w_k) \in A^{k-1}$ and assume that $\lambda^{u_1 u_2}_{\mathfrak{e}_1 \mathfrak{e}_2}(w_2, \dots, w_k) = (w_2', \dots, w_k')$. We set $w_1 := u_1$, $w_1' := u_1$, and likewise $w_{k+1} := u_2$ and $w_{k+1}' := u_2$. Also, set $\mathfrak{w}_i := \mathsf{orig}(w_i) = \mathsf{orig}(w_i')$ for every $i \in [k+1]$. Then the pebble p_i is placed on w_i and the pebble q_i is placed on w_i' for every $i \in [k+1]$. To show that Spoiler does not win in this round, we show that the i-th and $(i+1)$-th pebble pair induce a local isomorphism for every $i \in [k]$. So let $i \in [k]$ be arbitrary but fixed.

(i) Assume $\mathfrak{w}_i \neq \mathfrak{v}$ and $\mathfrak{w}_{i+1} \neq \mathfrak{v}$. In this case we have $w_i' = w_i$, $w_{i+1}' = w_{i+1}$, and $\mathfrak{e}_1 \neq \{\mathfrak{w}_i, \mathfrak{w}_{i+1}\}$. Thus, the pebbles define a local isomorphism.

(ii) Assume $\mathfrak{w}_i = \mathfrak{w}_{i+1} = \mathfrak{v}$. In this case $w_i' = \psi(w_i)$ and $w_{i+1}' = \psi(w_{i+1})$ for some isomorphism ψ. Because ψ is a bijection and inside a gadget there are no edges, the pebbles define a local isomorphism.

(iii) Lastly, assume $\mathfrak{w}_i = \mathfrak{v}$ and $\mathfrak{w}_{i+1} \neq \mathfrak{v}$. In this case $w_i' = \psi(w_i)$ and $w_{i+1}' = w_{i+1}$ for another isomorphism $\psi\colon \mathfrak{A}_g \to \mathfrak{A}_{g'}$ such that g and g' twist $\mathfrak{e}_1$ and another edge $\mathfrak{e}$ incident to $\mathfrak{v}$. By construction, $\mathfrak{e} \neq \{\mathfrak{w}_i, \mathfrak{w}_{i+1}\}$. Since f and g only twist $\mathfrak{e}_1$, f and g' twist exactly the edge $\mathfrak{e}$. Recall that ψ was chosen such that it is the identity on all vertices that do not originate from $\mathfrak{v}$. In particular, $\psi(w_{i+1}) = w_{i+1}$.

Assume that $\{\mathfrak{w}_i, \mathfrak{w}_{i+1}\} = \mathfrak{e}_1$. Then

$$\begin{aligned} \{w_i, w_{i+1}\} &\in E^{\mathfrak{A}_f} && \text{if and only if} \\ \{w_i, w_{i+1}\} &\notin E^{\mathfrak{A}_g} && \text{if and only if} \\ \{w_i', w_{i+1}'\} = \{\psi(w_i), \psi(w_{i+1})\} &\notin E^{\mathfrak{A}_{g'}} && \text{if and only if} \\ \{w_i', w_{i+1}'\} &\in E^{\mathfrak{A}_g} \end{aligned}$$

because f and g twist the base edge $\mathfrak{e}_1$ and g and g' twist the base edge $\mathfrak{e}_1$, too. Otherwise, $\{\mathfrak{w}_i, \mathfrak{w}_{i+1}\} \neq \mathfrak{e}_1$ and the reasoning is similar using that f and g (or g

and g', respectively) do no twist $\{\mathfrak{w}_i, \mathfrak{w}_{i+1}\}$ (by replacing "$\notin$" with "$\in$" in the prior equation). So the pebbles define a local isomorphism.

The case where w_{i+1} originates from $\mathfrak{v}$ but w_i does not is symmetric.

We now show Part (b). Let $i \in [k]$ and assume that Spoiler picks up all pebbles apart from the i-th and $(i+1)$-th pebble pair. Consider again the same situation as in Part (a): In Case (i), the identity map is the desired isomorphism. In Cases (ii) and (iii), the isomorphism $\psi = \psi^{-1}$ has the desired property (path-isomorphisms are always self-inverse because twisting an edge twice is the identity map).

Finally, we show Part (c). The situation is shown in Figure 3.3. If G_2 contains ℓ many vertices, then G_1 contains $\ell' := 2n - \ell - 1$ many vertices. Assume again that Spoiler picks $(w_2, \dots, w_k) \in A^{k-1}$ and $\lambda^{u_1,u_2}_{\mathfrak{e}_1\mathfrak{e}_2}(w_2, \dots, w_k) = (w'_2, \dots, w'_k)$. As before, we set $w_1 := u_1$, $w'_1 := u_1$, $w_{k+1} := u_2$, and $w'_{k+1} := u_2$. For every $i \in [k+1]$, the pebble p_i is placed on w_i and the pebble q_i is placed on w'_i. Assume that Spoiler chooses the i-th and $(i+1)$-th pebble pair to remain and picks up all other pebbles.

We first note that if w_i and w_{i+1} do not form a wall, then the twisted component has size $2n - 1$. In the case of a wall, we make the following case distinction. Set $\mathfrak{w}_1 := \mathsf{orig}(w_i) = \mathsf{orig}(w'_i)$ and $\mathfrak{w}_2 := \mathsf{orig}(w_{i+1}) = \mathsf{orig}(w'_{i+1})$.

- Assume $\mathfrak{v} \notin \{\mathfrak{w}_1, \mathfrak{w}_2\}$. In this case $\mathfrak{v}$ is in the twisted component because $\mathfrak{e}_1$ is incident to $\mathfrak{v}$. The twisted component has size at least $\min\{\ell, \ell'\} + 1$.

- Assume $\mathfrak{v} = \mathfrak{w}_1$ and $\mathfrak{w} := \mathfrak{w}_2$ is adjacent to $\mathfrak{v}$. If $\{\mathfrak{v}, \mathfrak{w}\} \in \{\mathfrak{e}_1, \mathfrak{e}_3\}$, where $\mathfrak{e}_3$ is the third edge incident to $\mathfrak{v}$, then there is no wall (cf. Figure 3.3). So $\mathfrak{w} = \mathfrak{v}'$ and thus the twist can be moved to $\mathfrak{e}_1$ or $\mathfrak{e}_3$ by the choice of ψ in the construction of $\lambda^{u_1u_2}_{\mathfrak{e}_1\mathfrak{e}_2}$. Then the twisted component has size $\min\{\ell, \ell'\}$. The case where $\mathfrak{w} = \mathfrak{w}_2$ and $\mathfrak{w}_1$ is adjacent to $\mathfrak{v}$ is analogous.

- Assume $\mathfrak{v} = \mathfrak{w}_1$ and $\mathfrak{w} := \mathfrak{w}_2$ is not adjacent to $\mathfrak{v}$. In the construction of $\lambda^{u_1u_2}_{\mathfrak{e}_1\mathfrak{e}_2}$, the isomorphism ψ is chosen such that the twist is moved to $\mathfrak{e}_1$ or $\mathfrak{e}_2$. If $\mathfrak{w}$ is in G_1, the twisted component has size $\ell + 1$ if the twist was moved to $\mathfrak{e}_2$ and size $\ell' - 1$ if the twist was moved to $\mathfrak{e}_1$. Otherwise, $\mathfrak{w}$ is in G_2. Since the twist can be moved to $\mathfrak{e}_1$ or $\mathfrak{e}_2$, the twisted component has size $\ell + 1$. Again, the case where $\mathfrak{w}_1$ and $\mathfrak{w}_2$ are swapped is analogous. □

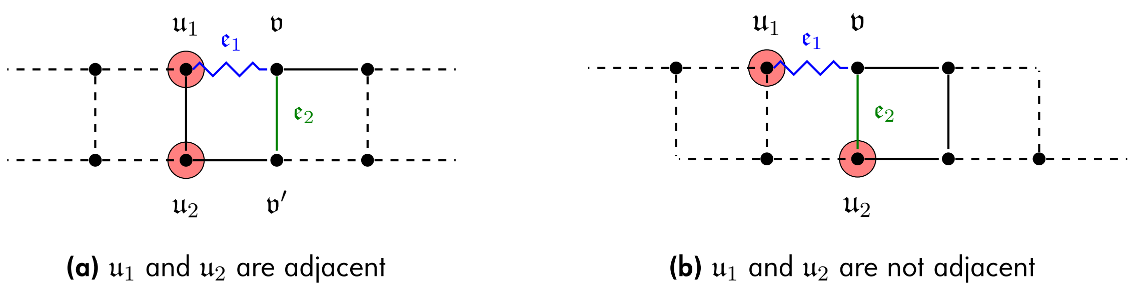
Theorem 3.29. *For all $k \geq 2$ and $n \geq 3$, Duplicator has a strategy in the bijective k-walk pebble game played on the graph $\mathsf{CFI}_{\mathrm{g}}(G^2_n, 0)$ and $\mathsf{CFI}_{\mathrm{g}}(G^2_n, 1)$ such that Spoiler wins in $\Omega(n)$ rounds at the earliest.*

Proof. Let $f, g \colon E \to \mathbb{F}_2$ such that $\sum f = 0$ and $\sum g = 1$. We show that Duplicator has a strategy such that after m rounds played on $\mathsf{CFI}_{\mathrm{g}}(G^2_n, f)$ and $\mathsf{CFI}_{\mathrm{g}}(G^2_n, g)$

1. there exists a pebble-respecting isomorphism

$$\varphi \colon \mathsf{CFI}_{\mathrm{g}}(G^2_n, g) \to \mathsf{CFI}_{\mathrm{g}}(G^2_n, g')$$

such that the pebbles are placed on the same vertices u_1 and u_2 on $\mathsf{CFI}_{\mathrm{g}}(G^2_n, f)$ and $\mathsf{CFI}_{\mathrm{g}}(G^2_n, g')$ and



3.4 The two possible situations in Theorem 3.29. The base vertices $\mathfrak{u}_1$ and $\mathfrak{u}_2$ are covered by pebbles (indicated by circles) and form a wall. Dashed edges may or may not exist. The other edges have to exist because the twisted component has size at least 3. The twisted component is to the right of the wall, where the base edge $\mathfrak{e}_1$ is twisted.

2. the twisted component with respect to f, g', u_1, and u_2 has size at least $n-2(m+1)$ and Spoiler can win only if its size is at most 2.

Clearly, in the beginning of the game there are no pebbles placed and there is only one twisted component of size $2n+1$.

Assume inductively that after m rounds the pebbles are placed on u_1 and u_2 in $\mathsf{CFI}_{\mathrm{g}}(G_n^2, f)$ and $\mathsf{CFI}_{\mathrm{g}}(G_n^2, g')$, that the twisted component has size at least $n-2(m+1) > 2$, and its Duplicator's turn to pick a bijection. (If less than 2 pebbles are placed, we just assume that the remaining ones are all placed at the same vertex at a harmless place, e.g., on the unique vertex originating from the unique base vertex of degree 1). There are two cases:

(a) The twisted component has size at most $n-2$. Then, in particular, Spoiler builds a wall. Let the pebbles be placed on base vertices $\mathfrak{u}_1 := \mathsf{orig}(u_1)$ and $\mathfrak{u}_2 := \mathsf{orig}(u_2)$. Duplicator picks $\mathfrak{v}$ as depicted in Figure 3.4:

- If $\mathfrak{u}_1$ and $\mathfrak{u}_2$ are adjacent, we possibly exchange names of $\mathfrak{u}_1$ and $\mathfrak{u}_2$ such that $\mathfrak{v}$ is the neighbor of $\mathfrak{u}_1$ in the twisted component of degree 3 and $\mathfrak{v}'$ is the common neighbor of $\mathfrak{v}$ and $\mathfrak{u}_2$. Up to isomorphism (Lemma 2.9), we w.l.o.g. assume that $\mathfrak{e}_1 := \{\mathfrak{u}_1, \mathfrak{v}\}$ is the only edge twisted by f and g'. Set $\mathfrak{e}_2 := \{\mathfrak{v}, \mathfrak{v}'\}$. Duplicator chooses the bijection $\lambda^{u_1 u_2}_{\mathfrak{e}_1 \mathfrak{e}_2}$ (Lemma 3.27).
- If otherwise $\mathfrak{u}_1$ and $\mathfrak{u}_2$ are not adjacent, $\mathfrak{v}$ is the neighbor of both $\mathfrak{u}_1$ and $\mathfrak{u}_2$ in the twisted component. We possibly exchange names of $\mathfrak{u}_1$ and $\mathfrak{u}_2$ so that $\mathfrak{e}_2 := \{\mathfrak{u}_2, \mathfrak{v}\}$ is a separator of the graph and set $\mathfrak{e}_1 := \{\mathfrak{u}_1, \mathfrak{v}\}$. Then Duplicator chooses the bijection $\lambda^{u_1 u_2}_{\mathfrak{e}_1 \mathfrak{e}_2}$.

In both cases, by Lemma 3.28(a), Spoiler does not win in the current round, by Lemma 3.28(b), there is the desired pebble-respecting isomorphism when Spoiler picks up all pebbles apart from a consecutive pebble pair, and by Lemma 3.28(c), the size of the new twisted component is at least $n-2(m+2)$.

(b) The twisted component has size greater than $n-2$. Let $\{\mathfrak{u}_1, \mathfrak{u}_2\}$ be a base edge that forms a separator of G_n^2 separating G_n^2 into two subgraphs containing at least

$n-2$ vertices such that $\mathfrak{u}_1$ has a neighbor of degree 3 in the twisted component. Such a separator exists because the size of the twisted component is greater than $n-2$. Duplicator proceeds with this choice of $\mathfrak{u}_1$, $\mathfrak{u}_2$, and $\mathfrak{v}$ as in the case before. After the current round, the twisted component has size at least $n-4$. □

Together with Theorem 3.21, we obtain the following corollaries:

Corollary 3.30. *Every walk counting logic formula which distinguishes* $\mathsf{CFI}_{\mathrm{g}}(G_n^2, 0)$ *and* $\mathsf{CFI}_{\mathrm{g}}(G_n^2, 1)$ *has quantifier depth* $\Omega(n)$.

Corollary 3.31. *The walk refinement distinguishes* $\mathsf{CFI}_{\mathrm{g}}(G_n^2, 0)$ *and* $\mathsf{CFI}_{\mathrm{g}}(G_n^2, 1)$ *in* $\Omega(n)$ *iterations. In particular, the walk refinement stabilizes on* $\mathsf{CFI}_{\mathrm{g}}(G_n^2, 0)$ *as well as* $\mathsf{CFI}_{\mathrm{g}}(G_n^2, 1)$ *in* $\Omega(n)$ *iterations.*

Recall that $\mathsf{CFI}_{\mathrm{g}}(G_n^2, 0)$ and $\mathsf{CFI}_{\mathrm{g}}(G_n^2, 1)$ have $\Theta(n)$ vertices, so both corollaries give bounds that are linear in the number of vertices.

We want to remark that 4 pebble pairs already suffice for Spoiler to reduce the size of the twisted component by 2 in each round: Spoiler places the pebbles always on four vertices originating from a 4-cycle starting in the middle of the graph. Then Spoiler moves two of them so that together the 4 pebbles cover a 4-cycle that shares an edge with the 4-cycle of the previous round. Overall, this strategy requires $n/2+1$ rounds if n is even and $(n+1)/2+1$ rounds otherwise. In particular, the 4-walk refinement has the same iteration number as walk-refinement on these graphs. Nevertheless, after only one iteration, the n-walk refinement distinguishes vertices of different gadgets, but the 4-walk refinement does not. With only 3 pebble pairs the size reduces by at most one per iteration (as already shown in [40]).

3.6 Discussion

We showed that the 2-dimensional Weisfeiler-Leman refinement stabilizes an n-vertex graph after $\mathcal{O}(n \log n)$ iterations. Hence, in the counting logic $\mathcal{C}_3$ we only require a quantifier depth of $\mathcal{O}(n \log n)$. This matches the best known lower bound of the form $\Omega(n)$ up to a logarithmic factor. Thus, the question remains what the precise bound is, and whether the iteration number can be superlinear. At least for the walk refinement we now have matching linear lower and upper bounds.

Recently, our algebraic approach was generalized from the 2-dimensional WL algorithm to the k-dimensional WL algorithm to establish the first nontrivial general upper bound of $O(n^{k-1} \log n)$ on its iteration number [55]. The approach is similar: First, regard the coloring of k-tuple as a matrix algebra, whereby the crucial point is to provide a suitable embedding of k-tuples of a graph into $V^{k-1} \times V^{k-1}$ matrices, where the vertex-set of the graph is V. This is not as straightforward as in the 2-dimensional case. Second, show that these algebras are semisimple, and lastly apply our bound on the length of chains of semisimple matrix algebras (Theorem 3.4).

It also remains an open problem whether our techniques can be applied to finite-variable first order logic *without* counting. The combinatorial techniques from [79] showing the upper bound of $\mathcal{O}(n^2/\log n)$ also translate to the setting without counting. Our techniques strongly rely on counting, since only the counting itself ensures the correspondence to matrix algebras that we exploit.

A combinatorial argument for the $\mathcal{O}(n)$ bound for the walk refinement or for the $\mathcal{O}(n \log n)$ bound for the 2-dimensional Weisfeiler-Leman algorithm could help to understand these questions.

Chapter 4

Canonization of Structures with Bounded Dihedral Colors in CPT

This chapter turns to capturing PTIME on a more extensive class of structures with the logic Choiceless Polynomial Time (CPT). One next reasonable major goal is to capture PTIME on the class of structures with q-bounded colors for each $q \in \mathbb{N}$, i.e., on colored structures in which every color class has size at most q. Structures with bounded color class size can be canonized in polynomial time (see [8, 10, 41]) using group-theoretic techniques. Canonizing structures with q-bounded colors requires easy group-theoretic techniques compared to Luks's polynomial-time isomorphism algorithm for graphs of bounded degree [90] or Babai's quasipolynomial-time isomorphism algorithm for arbitrary graphs [9]. Because these group-theoretic techniques inherently rely on choosing generating sets, it is not clear how this approach can be used in a choiceless logic. Indeed, we still do not know how to transfer these techniques into logics.

A first result towards the canonization of structures with bounded color class size in CPT was the canonization of structures with bounded and abelian colors due to Abu Zaid, Grädel, Grohe, and Pakusa [118]. These structures have the additional restriction that every color class induces a substructure with an abelian automorphism group. The authors use a certain class of linear equation systems to encode the group-theoretic structure of abelian color classes and solve these systems in CPT. Considering dihedral groups is a next natural step because dihedral groups are extensions of abelian groups by abelian groups or, more precisely, extensions of a cyclic group by another cyclic group and, in particular, the automorphism groups of regular n-gons. Dihedral groups for n odd are also called odd and the others are called even. As such, dihedral groups are, in some sense, the easiest non-abelian groups. We consider structures with q-bounded and dihedral or cyclic colors, that is, the automorphism group of every color class is a dihedral or cyclic group. We show that, for every $q \in \mathbb{N}$, CPT canonizes

(a) q-bounded ternary relational structures with odd dihedral or cyclic color classes and

(b) q-bounded binary relational structures with dihedral or cyclic color classes.

Thereby, CPT captures PTIME on these structures. The general strategy of our canonization is to reduce the dihedral groups in some way to abelian groups and then exploit the canonization procedure for them. However, we will see that this turns out to require effort.

Related Work. There already exist various results for CPT regarding structures with bounded color class size in addition to the ones mentioned above. As seen in Section 2.8, CFI graphs of totally ordered base graphs of bounded degree are graphs with bounded color class size. Indeed, 3-regular base graphs, so CFI graphs of color class size 4, suffice to show that IFPC does not define the CFI query and thus to separate IFPC from PTIME [20]. Dawar, Richerby, and Rossman [34] showed that the isomorphism problem of CFI graphs can be solved in CPT for ordered base graphs (of arbitrary degree), that is, for base graphs of color class size 1. This result was strengthened by Pakusa, Schalthöfer, and Selman [104] to base graphs with logarithmic color class size. The techniques of [34] and [104] are generalized in [118] to solve the equation systems mentioned before.

Recently, limitations of CPT were studied by Pago [100–102], but a separation of CPT from PTIME still seems to be out of reach. An approach similar to CPT was made to try to capture LOGSPACE via a logic called Choiceless Logarithmic Space by Grädel and Schalthöfer [47]. Choiceless Logarithmic Space is contained in both PTIME and CPT and is strictly more expressive than all logics for LOGSPACE that have been known so far. Nevertheless, the authors proved that Choiceless Logarithmic Space is strictly contained in LOGSPACE, i.e., it fails to capture LOGSPACE.

Overview of this Chapter. We first review necessary preliminaries for groups in Section 4.1. Next, we consider dihedral groups and 2-injective 3-factor subdirect products of dihedral and cyclic groups in Section 4.2. Such a product is a subgroup of the direct product of three groups and has the crucial property that no element is the identity in exactly two of the factors. We classify all 2-injective 3-factor subdirect products of dihedral and cyclic groups. Dihedral groups, which are the automorphism groups of regular n-gons, have a natural notion of reflections and rotations. They key insight is that most of these 2-injective 3-factor subdirect products are rotate-or-reflect groups, that is, every element is either composed of a rotation in every factor or of a reflection in every factor.

The next step in Section 4.3 is to introduce a normal form for structures with bounded color class size. In this normal form, there are two types of connections between color classes: Either, the automorphism group of three adjacent color classes form a 2-injective subdirect product or the automorphism group of one color class is a quotient of the automorphism group of an adjacent one. We prove that arbitrary structures with q-bounded colors can be reduced to said normal form by a CPT-interpretation in a canonization-preserving way: Given a CPT-definable canonization for the resulting normal form, we obtain a CPT-definable canonization for the original structures. If the class of automorphism groups of the color classes in the original structures is closed under sections, i.e., under subgroups and quotients, then the reduction preserves the class of automorphism groups. This is the reason for considering combinations of dihedral and cyclic groups.

Towards canonizing structures with dihedral colors, we start by a preprocessing step in Section 4.4. The preprocessing makes the dihedral automorphism group explicit in the structure by essentially defining a canonical n-gon in every color class. Next, we consider tree-like cyclic linear equation systems (TCES) in Section 4.5. TCESs are a special class of linear equations systems and generalize the equation systems used in [118] to canonize structures with bounded and abelian colors. We show that solvability of a certain subclass of TCESs is CPT-definable.

Finally, we show in Section 4.6 that the color classes of dihedral structures in said normal form decompose into reflection components. In a reflection component every automorphism either is a reflection or a rotation in all color classes contained in the reflection component. To prove this, we exploit the characterization of 2-injective subdirect products of dihedral and cyclic groups. However, reflection components are not really independent but can have "global" dependencies. Here, odd dihedral groups turn out to be easier. For these, reflection components can only be connected via abelian color classes. For even dihedral groups, there is a single non-abelian exception that can connect reflection components, which complicates matters. This exception restricts us to binary structures when allowing all dihedral groups. In each reflection component, the reflections can be forbidden in two canonical ways (essentially, by the two possibilities to turn an undirected cycle, i.e., an n-gon, into a directed cycle). For each such orientation, the reflection component has abelian colors since all reflections are removed. We then use the CPT-canonization for abelian color classes to canonize reflection components. To combine the canons of all reflection components, we encode the automorphisms of each reflection component via a TCES. We use the CPT-definable solvability test of TCESs to obtain canonizations of each reflection component, which are compatible with each other. That is, we encode the "global" dependencies by TCESs.

Towards generalization, it unfortunately becomes cumbersome to exploit the group structure theory in CPT, which is heavily required to execute the approach. Extending the treatment of linear equation systems, which is a subroutine in [118], to dihedral groups requires significant work already. We still follow the strategy of [118] and use equation systems, TCESs in our case, to encode the global dependencies. But because we use the more general notion of a TCES, we have to adapt all operations used on these equations systems to work in the more general setting (e.g. the check for solvability). This becomes even more technically involved than the techniques of [118] already are. We end with a discussion in Section 4.7.

4.1 Preliminaries

Groups. All groups considered in this chapter will be finite. We make use of standard group notation: Let Γ be a group. The identity (or trivial) element of Γ is written as 1. The group operation is written as multiplication, i.e., the product of $g, h \in \Gamma$ is gh. Recall that we denote the order of an element $g \in \Gamma$ by $\mathsf{ord}(g)$. The number of elements $|\Gamma|$ of Γ is called the **order of** Γ. For elements $g_1, \dots, g_i \in \Gamma$, we denote by $\langle g_1, \dots, g_i \rangle$ the **group generated** by $\{g_1, \dots, g_i\}$. We additionally use the notation for subgroups $\Delta_1, \dots, \Delta_i \leq \Gamma$, where $\langle \Delta_1, \dots, \Delta_i \rangle$ is the group generated by all elements contained in the subgroups. The **index** $[\Gamma : \Delta]$ of a subgroup $\Delta \leq \Gamma$ in Γ is $[\Gamma : \Delta] = |\Gamma|/|\Delta|$. A **(right) coset** of Γ is a set $\Gamma g := \{hg \mid h \in \Gamma\}$. A **normal subgroup** $N \triangleleft \Gamma$ is a subgroup $N \leq \Gamma$ satisfying $gN = Ng$ for all $g \in \Gamma$. Let $N \triangleleft \Gamma$. The quotient group Γ/N is the group of cosets gN for all $g \in \Gamma$. Two groups Γ_1 and Γ_2 are isomorphic, denoted $\Gamma_1 \cong \Gamma_2$, if there is a group isomorphism from Γ_1 to Γ_2.

Of special interest in this chapter are permutation groups. Let $\Gamma \leq \mathsf{Sym}(\Omega)$ be a finite permutation group with domain Ω. Recall that the **orbit** of $u \in \Omega$ in Γ is the set of points onto which u can be mapped, i.e.,

$$\mathsf{orb}_\Gamma(u) = \left\{ \sigma(v) \;\middle|\; \sigma \in \Gamma \right\} = \left\{ v \in \Omega \;\middle|\; \sigma(u) = v \text{ for some } \sigma \in \Gamma \right\}.$$

If the group is clear from the context, we just write $\mathsf{orb}(u)$. The set of orbits of Γ is $\mathsf{orb}(\Gamma) := \{\mathsf{orb}(u) \mid u \in \Omega\}$ and defines a partition of Ω. The group Γ is called **transitive** if Γ has only one orbit, i.e., $\mathsf{orb}(\Gamma) = \{\Omega\}$. An **action** of a group Γ on a set Ω' is a homomorphism from Γ to $\mathsf{Sym}(\Omega')$. By considering the image of group elements under the homomorphism, we can speak about orbits of group elements on Ω'. In that sense, the k**-orbits** of Γ are the orbits of Γ acting on Ω^k component-wise. Lastly, the finite permutation group Γ is **regular** if Γ is transitive and $|\Gamma| = |\Omega|$. Let $\Omega' \subseteq \Omega$. We denote by $\mathsf{Stab}_\Gamma(\Omega') := \{\sigma \in \Gamma \mid \sigma(\Omega') = \Omega'\}$ the **setwise stabilizer** of Ω' in Γ. For $u \in \Omega$, we write $\mathsf{Stab}_\Gamma(u)$ for $\mathsf{Stab}_\Gamma(\{u\})$. We define

$$\Gamma|_{\Omega'} := \left\{ \sigma|_{\Omega'} \;\middle|\; \sigma \in \mathsf{Stab}_\Gamma(\Omega') \right\}$$

to be the stabilizer of Ω' restricted to Ω'.

Let $\Gamma \leq \bigotimes_{i\in[r]} G_i$ be a subgroup of the direct product of the groups G_i. We denote by $\pi_j^\Gamma \colon \Gamma \to G_j$ the natural projection onto the j-th factor and with $\overline{\pi}_j^\Gamma \colon \Gamma \to \bigotimes_{i\in[r]\setminus\{j\}} G_i$ the projection onto all factors other than the j-th one. If, for every $i \in [r]$, the groups G_i are permutation groups acting on pairwise disjoint sets Ω_i, we also write $\pi_{\Omega_j}^\Gamma$ and $\overline{\pi}_{\Omega_j}^\Gamma$. We omit the group and only write π_j and $\overline{\pi}_j$ if Γ is clear from the context. In this chapter, we diverge from our usual notation and use the letters G and H for factors of direct products of groups.

Canonical labelings. Given a τ-structure $\mathfrak{A}$ and a canon $\mathfrak{B}$ of $\mathfrak{A}$, that is, an ordered $\tau \uplus \{\leq\}$-structure with $\mathfrak{B} \upharpoonright \tau \cong \mathfrak{A}$, the set of **canonical labelings** of $\mathfrak{A}$ to $\mathfrak{B}$ is the set of isomorphisms $\mathsf{Iso}(\mathfrak{A}, \mathfrak{B} \upharpoonright \tau)$. For readability, we also just write $\mathsf{Iso}(\mathfrak{A}, \mathfrak{B})$ in the case that $\mathfrak{B}$ is a canon of $\mathfrak{A}$ (and hence there are no isomorphisms $\mathfrak{A} \to \mathfrak{B}$ but only $\mathfrak{A} \to \mathfrak{B} \upharpoonright \tau$). Slightly abusing terminology, the canonical labelings form a coset, that is, $\mathsf{Iso}(\mathfrak{A}, \mathfrak{B}) = \mathsf{Aut}(\mathfrak{A})\varphi$ for every $\varphi \in \mathsf{Iso}(\mathfrak{A}, \mathfrak{B})$.

4.2 Classification of 2-Injective Subdirect Products of Dihedral Groups

In this section we focus on dihedral groups and classify all 2-injective 3-factor subdirect products of dihedral and cyclic groups. We first recall some existing definitions and introduce new ones, which are then used to state our classification concisely. The rest of the section is used to prove this classification.

2-Injective Subdirect Products. A group $\Gamma \leq G_1 \times G_2 \times G_3$ is called a **(3-factor) subdirect product** if the projection to each factor is surjective, that is, $\pi_i^\Gamma(\Gamma) = G_i$ for all $i \in [3]$. It is called **2-injective** if $\mathsf{ker}(\overline{\pi}_i^\Gamma) = \{1\}$ for all $i \in [3]$, that is, each projection onto two components is an injective map. Another way of looking at this is that two components of an element of Γ determine the third one uniquely. Simple examples of 2-injective subdirect products are diagonal subgroups: For a group G, the group $\{(g, g, g) \mid g \in G\} \leq G^3$ is called the **diagonal subgroup**.

Dihedral Groups. For $n \geq 3$, the **dihedral group** D_n of order $2n$ is the automorphism group of a regular n-gon in the plane. It consists of n rotations and n reflections and acts naturally on the set of n vertices of the polygon. In the degenerate cases D_1 and D_2 of orders 2 and 4, respectively, the dihedral group is abelian. For $n > 2$ it is non-abelian. We write C_n for the cyclic group of order n. It holds that $\mathsf{D}_1 \cong \mathsf{C}_2$ and $\mathsf{D}_2 \cong \mathsf{C}_2^2$. For $n > 2$, the notion of rotations and reflections of dihedral groups can be defined independently of the action of the group, but it coincides with the intuition of the action on regular n-gons.

Definition 4.1 (Rotation and Reflection). Let $n > 2$. We call an element $r \in \mathsf{D}_n$ a **rotation** if $\mathsf{ord}(r) \geq 3$ or r commutes with all elements of D_n which have order at least 3. An element $\alpha \in \mathsf{D}_n$ is called a **reflection** if it is not a rotation. We extend this notion to tuples $g = (g_1, \dots, g_k) \in \mathsf{D}_{n_1} \times \dots \times \mathsf{D}_{n_k}$: If g_i is a rotation (or reflection, respectively) for all $i \in [k]$, then g is called a rotation (or reflection, respectively).

Note that we regard the identity 1 as a rotation. In this section, we use the letters r and s for rotations and the letters α and β for reflections. In particular, if we for example write $(r, \alpha) \in \mathsf{D}_4 \times \mathsf{D}_6$, then we implicitly require r to be a rotation and α to be a reflection. Assume for $n > 2$ that $r, s \in \mathsf{D}_n$ are rotations and $\alpha, \beta \in \mathsf{D}_n$ are reflections. Then $rs = sr$ and $\alpha\beta$ are rotations and $r\alpha = \alpha r^{-1}$ is a reflection. For $k \geq 1$ and $n_i > 2$ for every $i \in [k]$, the direct product $\mathsf{D}_{n_1} \times \dots \times \mathsf{D}_{n_k}$ contains mixed elements that are neither a reflection nor a rotation. Subgroups of such a group may or may not contain such mixed elements.

Definition 4.2 (Rotate-or-Reflect Group). Let $\Gamma \leq \mathsf{D}_{n_1} \times \dots \times \mathsf{D}_{n_k}$ and $n_i > 2$ for all $i \in [k]$. We call Γ a **rotate-or-reflect** group if every $g \in \Gamma$ is a rotation or a reflection.

Definition 4.3 (Rotation Subgroup). We define the rotation subgroup of D_n for $n > 2$ as

$$\mathsf{Rot}(\mathsf{D}_n) := \{\, g \in \mathsf{D}_n \mid g \text{ is a rotation} \,\} \cong \mathsf{C}_n.$$

For a group $\Gamma \leq \mathsf{D}_{n_1} \times \ldots \times \mathsf{D}_{n_k}$ with $n_i > 2$ for all $i \in [k]$, we define

$$\mathsf{Rot}(\Gamma) := \Gamma \cap (\mathsf{Rot}(\mathsf{D}_{n_1}) \times \dots \times \mathsf{Rot}(\mathsf{D}_{n_k}))\,.$$

Corollary 4.4. *If $\Gamma \leq \mathsf{D}_{n_1} \times \dots \times \mathsf{D}_{n_k}$ is a rotate-or-reflect group, then*

$$\mathsf{Rot}(\Gamma) = \Gamma \cap (\mathsf{Rot}(\mathsf{D}_{n_1}) \times \mathsf{D}_{n_2} \times \dots \times \mathsf{D}_{n_k})\,.$$

By symmetry, restricting any other factor to its rotation subgroup yields the same group.

Now, we are prepared to state the classification of 2-injective subdirect products of dihedral groups and cyclic groups. For this we analyze how reflections and rotations of dihedral groups in 2-injective subdirect products can be combined. We will see that if no factor is isomorphic to D_1, D_2, or D_4 we always obtain a rotate-or-reflect group. Specifically, the goal of this section is to prove the following two theorems (the mentioned double CFI group is defined in Section 4.2.2, see Figure 4.1):

Theorem 4.5. *Let $\Gamma \leq \mathsf{D}_{n_1} \times \mathsf{D}_{n_2} \times \mathsf{D}_{n_3}$ be a 2-injective subdirect product. Then exactly one of following holds:*

1. *$n_1, n_2, n_3 > 2$ and Γ is a rotate-or-reflect group.*
2. *$n_i \leq 2$, $n_j, n_k > 2$ for $\{i,j,k\} = [3]$, and $\overline{\pi}_i(\Gamma)$ is a rotate-or-reflect group.*
3. *$n_1 = n_2 = n_3 = 4$ and Γ is isomorphic to the double CFI group Γ_{2CFI}.*
4. *$n_1, n_2, n_3 \leq 2$ and Γ is abelian.*

We also have to consider the case that some factors are cyclic groups.

Theorem 4.6. *Let $\Gamma \leq \mathsf{C}_{n_1} \times \mathsf{D}_{n_2} \times \mathsf{D}_{n_3}$ be a 2-injective subdirect product. Then exactly one of the following holds:*

1. *$n_1 \leq 2$, $n_2, n_3 > 2$, and $\overline{\pi}_1(\Gamma)$ is a rotate-or-reflect group.*
2. *$n_1 = n_2 = n_3 = 4$ and $\Gamma \cong \Gamma_{\mathsf{2CFI}} \cap (\mathsf{Rot}(\mathsf{D}_4) \times \mathsf{D}_4 \times \mathsf{D}_4)$.*
3. *$n_1, n_2, n_3 \leq 2$ and Γ is abelian.*

Furthermore, there are no 2-injective subdirect products of $\mathsf{D}_n \times G_2 \times G_3$ for $n > 2$ if G_2 and G_3 are abelian groups.

We also checked the classification with a computer program written in the computer algebra system GAP [60] up to $n_i = 20$.

4.2.1 Dihedral Groups not of Order 2, 4, or 8

We first consider the more general case where all factors are dihedral groups D_i for $i \notin \{1, 2, 4\}$. For this, we collect some basic facts about 2-injective subdirect products.

Lemma 4.7. *Let $\Gamma \leq G_1 \times G_2 \times G_3$ be a 2-injective group and $g_i \in G_i$ for every $i \in [3]$. If $(g_1, g_2, g_3) \in \Gamma$, then $\mathsf{ord}(g_i)$ divides the least common multiple $\text{lcm}\{\mathsf{ord}(g_{i+1}), \mathsf{ord}(g_{i+2})\}$ (addition on indices is meant to be wrapping around). In particular:*

(a) If $g_1 \neq 1$, then $(g_1, 1, 1) \notin \Gamma$.

(b) If $\mathsf{ord}(g_1) > \mathsf{ord}((g_2, g_3))$, then $(g_1, g_2, g_3) \notin \Gamma$.

Proof. By symmetry, we only consider $i = 1$. Suppose for $(g_1, g_2, g_3) \in \Gamma$ that $\mathsf{ord}(g_1)$ does not divide $\ell := \text{lcm}\{\mathsf{ord}(g_2), \mathsf{ord}(g_3)\}$. Then $(g_1^\ell, g_2^\ell, g_3^\ell) = (g_1^\ell, 1, 1) \in \Gamma \setminus \{1\}$, which contradicts 2-injectivity. Items (a) and (b) follow immediately. $\square$

For a 2-injective subdirect product Γ, we define

$$H_i^\Gamma := \mathsf{ker}(\pi_i^\Gamma) = \left\{ (g_1, g_2, g_3) \;\middle|\; g_i = 1 \right\}.$$

We set $H^\Gamma := \langle H_1^\Gamma, H_2^\Gamma, H_3^\Gamma \rangle$. In the following we just write H_i (or H, respectively) for H_i^Γ (or H^Γ, respectively) if the group Γ is clear from the context. Note that H_i^Γ defines an isomorphism between $\pi_{i+1}(H_i^\Gamma)$ and $\pi_{i+2}(H_i^\Gamma)$ (indices again wrapping around) since the entries for g_{i+1} and g_{i+2} are in one-to-one correspondence when $g_i = 1$.

Lemma 4.8 ([97]). *Let $\Gamma \leq G_1 \times G_2 \times G_3$ be a 2-injective subdirect product and $i \in [3]$. Then the map defined via $g_{i+1} \mapsto g_{i+2}$ whenever $(g_1, g_2, g_3) \in H_i^\Gamma$ is an isomorphism $\varphi\colon \pi_{i+1}(H_i^\Gamma) \to \pi_{i+2}(H_i^\Gamma)$ (indices again wrapping around).*

Proof. Assume by symmetry that $i = 1$. We first show that φ is bijective (and in particular a well-defined map). Suppose that $(1, g_2, g_3), (1, h_2, g_3) \in H_1^\Gamma$ for $g_2 \neq h_2$ and thus $\varphi(g_2) = g_3 = \varphi(h_2)$. We obtain a contradiction to Lemma 4.7(a) because $1 \neq (1, g_2, g_3)(1, h_2, g_3)^{-1} = (1, g_2 h_2^{-1}, 1) \in H_1^\Gamma \subseteq \Gamma$.

Now, let $(1, g_2, g_3), (1, h_2, h_3) \in H_1^\Gamma$. By definition $\varphi(g_2) = g_3$ and $\varphi(h_2) = h_3$. Because $(1, g_2, g_3)(1, h_2, h_3) = (1, g_2 h_2, g_3 h_3) \in H_1^\Gamma$, it holds that $\varphi(g_2 h_2) = g_3 h_3 = \varphi(g_2)\varphi(h_2)$. Thus, φ is a group isomorphism. □

Theorem 4.9 ([97]). *Let $\Gamma \leq G_1 \times G_2 \times G_3$ be a 2-injective subdirect product. Then $[G_i : \pi_i(H)] = [\Gamma : H]$.*

Now, we turn to dihedral groups.

Lemma 4.10. *Let $\Gamma \leq \mathrm{D}_{n_1} \times \mathrm{D}_{n_2} \times \mathrm{D}_{n_3}$ be a 2-injective subdirect product and $n_i \notin \{1, 2\}$ for all $i \in [3]$. If $(\alpha_1, r_2, r_3) \in \Gamma$ for a reflection α_1 and rotations r_2 and r_3, then there is an element $(r_1', \alpha_2', \alpha_3') \in \Gamma$ such that r_1' is a rotation and α_2' and α_3' are reflections.*

Proof. For each $i \in \{2, 3\}$, we pick a reflection α_i in D_{n_i}. Because Γ is a subdirect product, there is, for every $i \in \{2, 3\}$, an element $h_i = (g_{i,1}, g_{i,2}, g_{i,3}) \in \Gamma$ such that $g_{i,i} = \alpha_i$. We make the following case distinction:

1. One of the h_i already has the desired form of having a reflection in the first component and rotations elsewhere. In this case we are done.

2. If one $h_i = (\beta_1, \beta_2, \beta_3)$ only consists of reflections, then

$$h_i(\alpha_1, r_2, r_3) = (\beta_1 \alpha_1, \beta_2 r_2, \beta_3 r_3) = (r_1', \alpha_2', \alpha_3') \in \Gamma$$

   has the desired form.

3. If $h_2 = (s_1, \alpha_2, s_3)$, we make another case distinction on h_3. If $h_3 = (s_1', s_2', \alpha_3)$, then

$$h_2 h_3 = (s_1 s_1', \alpha_2 s_2', s_3 \alpha_3) = (r_1', \alpha_2', \alpha_3') \in \Gamma$$

   has the desired form. Otherwise, $h_3 = (\beta_1', s_2', \alpha_3)$ and

$$h_2 h_3 = (s_1 \beta_1', \alpha_2 s_2', s_3 \alpha_3)$$

   only consists of reflections. Thus, we reduced to Case 2.

4. Otherwise, $h_2 = (\beta_1, \alpha_2, s_3)$ and we perform the same case distinction on h_3. If $h_3 = (s_1', s_2', \alpha_3)$, then $h_2 h_3$ only consists of reflections and we reduced to Case 2. If otherwise $h_3 = (\beta_1', s_2', \alpha_3)$, then $h_2 h_3$ has the desired form similar to Case 3. □

Lemma 4.11. *Let $\Gamma \leq \mathrm{D}_{n_1} \times \mathrm{D}_{n_2} \times \mathrm{D}_{n_3}$ be a 2-injective subdirect product and $n_i \notin \{1, 2\}$ for all $i \in [3]$. If, for every $i \in [3]$, the group $\overline{\pi}_i(H_i)$ contains no reflections, then Γ is a rotate-or-reflect group.*

Proof. If there is a violating element in Γ, then it consists of one rotation and two reflections or of two rotations and one reflection. By Lemma 4.10, we can assume that the element consists of one rotation and two reflections and that up to reordering of the factors $(r_1, \alpha_2, \alpha_3) \in \Gamma$. In particular, $\mathsf{ord}(r_1) \leq 2$ by Lemma 4.7(b). If $r_1 = 1$, then $(\alpha_2, \alpha_3) \in \overline{\pi}_1(H_1)$, which contradicts that there is no reflection in $\overline{\pi}_1(H_1)$. Otherwise, $\mathsf{ord}(r_1) = 2$ and in particular $n_1 > 3$ (D_3 contains no rotation of order 2). Consider a k-th root s_1 of r_1 such that $s_1^k = r_1$ for some $k > 1$ (s_1 exists because $n_1 > 3$). Because Γ is a subdirect product, there is an element $(s_1, g_2, g_3) \in \Gamma$. Now, $\mathsf{ord}(s_1) > 2$ and thus one of g_2 and g_3 is a rotation of order greater than 2 by Lemma 4.7, say up to reordering $g_2 = s_2$. Then $(r_1, \alpha_2, \alpha_2)(s_1, s_2, g_3)^k = (1, \alpha_2 s_2^k, \alpha_3 g_3^k)$. Clearly, $\alpha_2 s_2^k$ is a reflection, hence $\alpha_3 g_3^k$ is a reflection (Lemma 4.8), and $(\alpha_2 s_2^k, \alpha_3 g_3^k) \in \overline{\pi}_1(H_1)$, which again contradicts that $\overline{\pi}_1(H_1)$ contains no reflection. We conclude that Γ is a rotate-or-reflect group. □

Lemma 4.12. *Let $\Gamma \leq \mathsf{D}_{n_1} \times \mathsf{D}_{n_2} \times \mathsf{D}_{n_3}$ be a 2-injective subdirect product and $n_i \notin \{1, 2, 4\}$ for some $i \in [3]$ and $n_j > 2$ for all $j \in [3]$. Then H_i contains only rotations.*

Proof. By symmetry, we fix w.l.o.g. $i = 1$. For the sake of contradiction, suppose that $(1, g_2, g_3) \in H_1$ is an element which is not a rotation. Then g_2 or g_3 is not a rotation. Because H_1 defines an isomorphism between $\pi_2(H_1)$ and $\pi_3(H_i)$ (Lemma 4.8), $g_2 = \alpha_2$ and $g_3 = \alpha_3$ must both be reflections.

Let $r_1 \in \mathsf{D}_{n_1}$ be a rotation such that $\mathsf{ord}(r_1) > 2$ and $\mathsf{ord}(r_1^2) > 2$, which is possible because $n_1 \notin \{1, 2, 4\}$. Then, by being subdirect, there is an element $(r_1, h_2, h_3) \in \Gamma$. The elements h_2 and h_3 cannot be both of order at most 2 by Lemma 4.7(b). So assume w.l.o.g. that $\mathsf{ord}(h_2) > 2$ and hence $h_2 = r_2$ is a rotation. If $h_3 = r_3$ is a rotation, too, then

$$(r_1, r_2, r_3)(1, \alpha_2, \alpha_3) = (r_1, r_2\alpha_2, r_3\alpha_3) \in \Gamma$$

yields a contradiction to Lemma 4.7(b) because $\mathsf{ord}(r_1) > 2$ and $r_i\alpha_i$ are reflections of order 2 (for $i \in \{2, 3\}$).

So, finally $h_3 = \beta_3$ must be a reflection. But then consider

$$(r_1, r_2, \beta_3)^2(1, \alpha_2, \alpha_3) = ({r_1}^2, {r_2}^2\alpha_2, \alpha_3) \in \Gamma$$

and note the last two components are reflections of order 2 but the first component has order greater than 2. This again contradicts Lemma 4.7(b). □

Corollary 4.13. *Let $\Gamma \leq \mathsf{D}_{n_1} \times \mathsf{D}_{n_2} \times \mathsf{D}_{n_3}$ be a 2-injective subdirect product and $n_i \notin \{1, 2, 4\}$ for all $i \in [3]$. Then Γ is a rotate-or-reflect group.*

Proof. This follows from Lemmas 4.11 and 4.12. □

The previous corollary classifies all 2-injective subdirect products of dihedral groups as rotate-or-reflect groups if D_1, D_2, and D_4 are not involved as one of the factors. We now analyze how rotations and reflections of the single factors can be combined if these groups are involved.

4.2.2 The CFI Group and the Double CFI Group

The following exception where rotations can be combined with reflections will be of particular interest.

Definition 4.14 (CFI Groups). We call the following group $\Gamma_{\mathsf{CFI}} < \mathsf{D}_1^3$ the **CFI group**:

$$\Gamma_{\mathsf{CFI}} := \Big\{ (g_1, g_2, g_3) \in \mathsf{D}_1^3 \;\Big|\; g_1 g_2 g_3 = 1 \Big\},$$

that is, Γ_{CFI} is the group consisting of those triples that contain an even number of the nontrivial elements of D_1. We call the wreath product $\Gamma_{\mathsf{2CFI}} := \Gamma_{\mathsf{CFI}} \wr \mathsf{C}_2$ the **double CFI group**.

The CFI group Γ_{CFI} is the automorphism group of the degree 3 CFI gadgets where each edge-vertex-pair is colored uniquely (cf. Section 2.8), or more precisely, the induced automorphism group on the edge vertices. The double CFI group Γ_{2CFI} is the automorphism group of two such CFI gadgets, which has the additional automorphism exchanging the two gadgets (cf. Figure 4.1 for two colored graphs whose automorphism groups are isomorphic to Γ_{CFI} and Γ_{2CFI}, respectively). It is straightforward to verify that the CFI group is a 2-injective subdirect product. We now show how the double CFI group can be realized as a 2-injective subdirect product $\Gamma_{\mathsf{2CFI}} < \mathsf{D}_4^3$ (cf. Figure 4.1b): Let $r \in \mathsf{D}_4$ be a rotation of order 4 and $\alpha, \alpha' \in \mathsf{D}_4$ be reflections such that $\alpha\alpha' = r^2$ (i.e., $\{\alpha, \alpha'\}$ is a conjugacy class, that is, α and α' are the two reflections at the diagonal of the square), and $\beta \in \mathsf{D}_4$ be a reflection with $\beta \notin \{\alpha, \alpha'\}$. Then

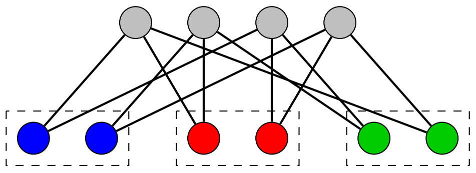
$$\begin{aligned}\Gamma_{\mathsf{2CFI}} &\cong \Big\langle (1,\alpha,\alpha), (\alpha,1,\alpha), (1,\alpha',\alpha'), (\alpha',1,\alpha'), (\beta,\beta,\beta) \Big\rangle \\ &= \Big\langle (1,\alpha,\alpha), (\alpha,1,\alpha), (r,r,\beta) \Big\rangle.\end{aligned}$$

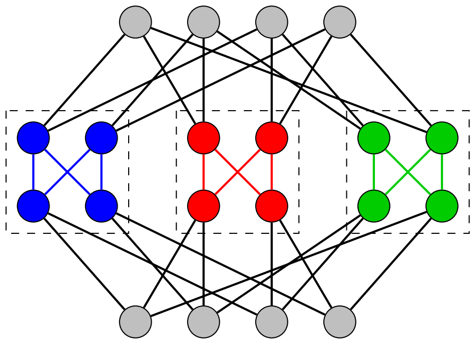
The elements with the two reflections α and α' generate the two independent CFI groups, respectively, and (β,β,β) exchanges the two CFI groups (swaps top with bottom in Figure 4.1b).

Lemma 4.15. *There is (up to isomorphism) exactly one 2-injective subdirect product of* D_4^3 *that is not a rotate-or-reflect group, namely the double CFI group.*

Proof. The group D_4 has the following normal subgroups: $\mathsf{D}_4, \mathsf{D}_2, \mathsf{C}_4, \mathsf{C}_2, \mathsf{C}_1$. Note that the normal subgroup C_2 of D_4 is generated by the 180 degree rotation (and not by a reflection because reflections do not generate normal subgroups). The H_i must be isomorphic to one of these normal subgroups because H_2 defines an isomorphism between $\pi_1(H_2)$ and $\pi_3(H_2)$ (Lemma 4.8) and thus $\pi_1(H_2) \cong \pi_3(H_2) \cong H_2$. Furthermore, $|\pi_i(H)| = |\pi_j(H)|$ for all $i, j \in [3]$ by Theorem 4.9. We now show that if Γ is not a rotate-or-reflect group, then $H_i \cong \mathsf{D}_2$ for all $i \in [3]$. First, if all H_i are cyclic groups, then Γ is a rotate-or-reflect group by Lemma 4.11.

Second, we show that (up to reordering) $H_1 \cong \mathsf{D}_4$ is contradictory. Let $r \in \mathsf{D}_4$ be a rotation of order 4. There is a rotation s of order 4 such that $g = (1, r, s) \in H_1 \subseteq \Gamma$ because H_1 defines an isomorphism between its projections to the second and third component (Lemma 4.8). Since $\pi_1(H) = \mathsf{D}_4$, there is an $h = (\alpha, h_2, h_3) \in H_2 \cup H_3 \subseteq \Gamma$. W.l.o.g. assume $h = (\alpha, \beta, 1) \in H_2$. Then $gh = (\alpha, \beta r, s) \in \Gamma$ which is impossible by Lemma 4.7(b) because α and βr are reflections of order 2 and s is of order 4.


(a) The CFI group Γ_{CFI}


(b) The double CFI group $\Gamma_{2\mathsf{CFI}}$

4.1 The CFI group and the double CFI group. The figure shows two vertex-colored graphs whose automorphism groups are the CFI group (Figure (a)) and the double CFI group (Figure (b)). The edge colors in Figure (b) are only for illustration: Each color forms an undirected cycle of length 4 and induces the automorphism group D_4 of its color class.

Third, assume that (up to reordering) $H_1 \cong \mathsf{D}_2$, $H_2 \ncong \mathsf{D}_2$, and $H_3 \ncong \mathsf{D}_2$. Because the case of one $H_i \cong \mathsf{D}_4$ was proved contradictory, it follows that H_2 and H_3 are isomorphic to C_4, C_2, or C_1. Suppose $H_2 \cong \mathsf{C}_j$ for some $j \in [2]$. Then $\pi_1(\langle H_1, H_2\rangle) \cong \mathsf{C}_j$ because the first entry of elements in H_1 equal 1. So $\mathsf{D}_2 \leq \pi_i(H)$ because $H_1 \cong \mathsf{D}_2$ and the rotations in C_j do not generate an element of order 4. In particular, $|\pi_i(H)| \geq 4$ for every $i \in \{2,3\}$. Then $|\pi_1(H)| \geq 4$ by Theorem 4.9 and so $H_3 \cong \mathsf{C}_4$ ($H_3 \ncong \mathsf{D}_2$ by assumption and $H_3 \cong \mathsf{D}_4$ was already proved inconsistent). Hence, $|\pi_1(H)| = 4$ but also $|\pi_2(H)| = |\mathsf{D}_4| = 8$ contradicting Theorem 4.9. The case $H_3 \cong \mathsf{C}_j$ for some $j \in [2]$ is symmetric. So the case $H_2 \cong H_3 \cong \mathsf{C}_4$ remains. Then $\pi_1(H) \cong \mathsf{C}_4$ but $\pi_2(H) \cong \mathsf{D}_4$, which contradicts Theorem 4.9 again.

Lastly, we assume that (up to reordering) $H_1 \cong H_2 \cong \mathsf{D}_2$. Then $\pi_3(\langle H_1, H_2\rangle)$ is either D_2 or D_4 (depending on whether H_1 and H_2 contain reflections from the same conjugacy class or not). In the case that $\pi_3(H_1) \neq \pi_3(H_2)$, H_1 and H_2 use reflections from different conjugacy classes of D_4. Let α and α' be these reflections such that $(1,\beta,\alpha) \in H_1$ and $(\beta',1,\alpha') \in H_2$. Then $(1,\beta,\alpha)(\beta',1,\alpha') = (\beta',\beta,r) \in \Gamma$ where r is the rotation of order 4 and thus contradicts Lemma 4.7(b). So $\pi_3(H_1) = \pi_3(H_2)$, that is, both kernels use the same reflections of D_4. Let $\alpha \in \mathsf{D}_4$ be one of these reflections. Then $(1,\beta,\alpha) \in H_1$, $(\beta',1,\alpha) \in H_2$, and $(1,\beta,\alpha)(\beta',1,\alpha) = (\beta',\beta,1) \in H_3$. So H_3 contains reflections and cannot be isomorphic to D_4 by the prior reasoning, i.e., $H_3 \cong \mathsf{D}_2$.

So we have shown that, unless Γ is a rotate-or-reflect group, we have $H_i \cong \mathsf{D}_2$ and $\pi_i(H) = \mathsf{D}_2$ for all $i \in [3]$. That is, all H_i use reflections of the same conjugacy class of D_4 for each component (there are two embeddings of D_2 in D_4). We apply an isomorphism to Γ, such that the embedding is the same for each component. Let $\alpha, \alpha' \in \mathsf{D}_4$ be the two reflections of this embedding of D_2 into D_4 and $r \in \mathsf{D}_4$ be a rotation of order 4. We assume that $(1,\alpha,\alpha), (1,\alpha',\alpha') \in H_1$ and $(\alpha,1,\alpha), (\alpha',1,\alpha') \in H_2$. In the case that α gets combined with α' we apply isomorphisms to the first and/or second factor exchanging α and α'. Then H_3 also combines α with α.

Recall that r is a rotation of order 4. Now $(r,g,h) \in \Gamma$, one of g and h has to equal r or r^{-1}, too, by Lemma 4.7(b) and the other one cannot equal 1 because $H_2, H_3 \ncong \mathsf{D}_4$.

Assume, w.l.o.g., that $g = r$ (if needed, we reorder the factors, and if $g = r^{-1}$, we apply another isomorphism to the second factor exchanging r and r^{-1} but which is constant on the reflections). Let β and β' be the reflections not in the conjugacy class of α. Then $\alpha\beta, \alpha\beta', \alpha'\beta, \alpha'\beta' \in \{r, r^{-1}\}$. Suppose first that $h = s$ is a rotation. But then $(r, r, s)(1, \alpha, \alpha) = (r, r\alpha, s\alpha) \in \Gamma$ which is a contradiction to Lemma 4.7(b).

So h is a reflection. If $h \in \{\alpha, \alpha'\}$, then $((r, r, h)(1, h, h))^2 = (r, rh, 1)^2 = (r^2, 1, 1) \neq 1$ because h and rh are reflections and r is of order 4. This contradicts Lemma 4.7(a). Hence, $h \in \{\beta, \beta'\}$, say up to isomorphism $h = \beta$, and finally $\Delta := \langle (r, r, \beta), (\alpha, 1, \alpha), (1, \alpha, \alpha)\rangle$ is isomorphic to the double CFI group. Note that $|H| = |\langle H_1, H_2, H_3\rangle| = 16$, that $|\Delta| = 32$, and that $[\Gamma : H] = [\mathsf{D}_4 : \pi_1(H)] = [\mathsf{D}_4 : \mathsf{D}_2] = 2 = [\Delta : H]$ by Theorem 4.9. Hence $|\Delta| = |\Gamma|$ and $\Gamma \cong \Delta$ is isomorphic to the double CFI group. □

4.2.3 Products Involving D_4, D_2, or D_1

We complete the classifications of 2-injective subdirect products of dihedral groups in the case that D_4, D_2, or D_1 occur as factors.

Lemma 4.16. *Let $\Gamma \leq \mathsf{D}_i \times \mathsf{D}_j \times \mathsf{D}_k$ with $i \in [2]$ and $j, k > 2$ be a 2-injective subdirect product. Then $\overline{\pi}_1(\Gamma)$ is a rotate-or-reflect group.*

Proof. For the sake of a contradiction, suppose $(g, r, \alpha) \in \Gamma$. By Lemma 4.7(b), r is of order at most 2 because g and α are of order at most 2. Let $s \in \mathsf{D}_j$ be a rotation of order greater 2 and, because Γ is a subdirect product, let $(h, s, s') \in \Gamma$ be an element whose second component is s (the last component must be a rotation of order larger than 2). Then $(gh, rs, \alpha s') \in \Gamma$, gh and $\alpha s'$ are of order at most 2, and rs is a rotation with $\mathsf{ord}(rs) > 2$. This is a contradiction to Lemma 4.7(b). □

Lemma 4.17. *There are no 2-injective subdirect products of $\mathsf{D}_i \times G_2 \times G_3$ if $i > 2$ and both G_2 and G_3 are abelian.*

Proof. Suppose $\Gamma \leq \mathsf{D}_i \times G_2 \times G_3$ is a 2-injective subdirect product and $\alpha, r \in \mathsf{D}_i$ such that $\alpha r \neq r\alpha$. Two such elements exist because $i > 2$ and D_i is non-abelian. Consider two elements $g = (\alpha, g_2, g_3), h = (r, h_2, h_3) \in \Gamma$ for some $g_2, h_2 \in G_2$ and $g_3, h_3 \in G_3$. Now $gh = (\alpha r, g_2h_2, g_3h_3)$ and $hg = (r\alpha, g_2h_2, g_3h_3) \neq gh$ contradicting 2-injectivity of Γ. □

Lemma 4.18. *There are no 2-injective subdirect products of $\mathsf{D}_4 \times \mathsf{D}_4 \times \mathsf{D}_i$ if $i \notin \{1, 2, 4\}$.*

Proof. Suppose $\Gamma \leq \mathsf{D}_4 \times \mathsf{D}_4 \times \mathsf{D}_i$ is a 2-injective subdirect product and $i \notin \{1, 2, 4\}$. Let $r \in \mathsf{D}_i$ be a rotation with $\mathsf{ord}(r) = i$ and $(g, h, r) \in \Gamma$ for some $g, h \in \mathsf{D}_4$. For all $g, h \in \mathsf{D}_4$, it holds that $\mathsf{ord}(g), \mathsf{ord}(h) \in \{1, 2, 4\}$ contradicting Lemma 4.7. □

Lemma 4.19. *Let $\Gamma \leq \mathsf{D}_4 \times \mathsf{D}_{n_2} \times \mathsf{D}_{n_3}$ be a 2-injective subdirect product. If $n_2, n_3 \notin \{1, 2, 4\}$, then Γ is a rotate-or-reflect group.*

Proof. We assume that $n_2, n_3 \notin \{1, 2, 4\}$ and apply Lemma 4.11. We have to show that $\overline{\pi}_i(H_i)$ contains no reflection for every $i \in [3]$. By Lemma 4.12, this is the case for H_2 and H_3. Assume that $(1, \alpha_2, \alpha_3) \in H_1$ and note that $\pi_i(H_1) \lhd \mathsf{D}_{n_i}$ for every $i \in \{2, 3\}$. So $\pi_i(H_1) \in \{\mathsf{D}_{n_i}, \mathsf{D}_{n_i/2}\}$ for every $i \in \{2, 3\}$ (if n_i is odd, $\mathsf{D}_{n_i/2}$ of course does not exist).

Recall that by Theorem 4.9, we have $[\mathsf{D}_{n_i} : \pi_i(H)] = [\Gamma : H]$. If $\pi_2(H_1) = \mathsf{D}_{n_2}$, then $\pi_2(H) = \mathsf{D}_{n_2}$ and

$$1 = [\mathsf{D}_{n_2} : \pi_2(H)] = [\mathsf{D}_{n_1} : \pi_1(H)]$$

and so $\pi_1(H) = \mathsf{D}_4$. So one of H_2 and H_3 contains reflections because the first entry of all elements in H equals 1, which is a contradiction.

Otherwise, $\pi_2(H) = \pi_2(H_1) = \mathsf{D}_{n_2/2}$, by symmetry $\pi_3(H) = \pi_3(H_1) = \mathsf{D}_{n_3/2}$, and $[\mathsf{D}_{n_1} : \pi_1(H)] = 2$. So $\pi_1(H) = \langle \pi_1(H_2), \pi_1(H_3)\rangle \in \{\mathsf{D}_2, \mathsf{C}_4\}$. If $\pi_1(H) = \mathsf{D}_2$, then H_2 or H_3 again contain reflections. Thus, $\pi_1(H) = \mathsf{C}_4$. Let $r \in \mathsf{C}_4 < \mathsf{D}_4$ be a rotation of order 4. There is an element $(r, r', 1) \in H_3$. Hence, $(r, r', 1)(1, \alpha_2, \alpha_3) = (r, r'\alpha_2, \alpha_3) \in \Gamma$, which is a contradiction to Lemma 4.7(b). □

Proof of Theorem 4.5. Let $\Gamma \leq \mathsf{D}_{n_1} \times \mathsf{D}_{n_2} \times \mathsf{D}_{n_3}$ be a 2-injective subdirect product. If $n_i \leq 2$ for all $i \in [3]$, then Γ is abelian. If $n_i \leq 2$ for exactly one $i \in [3]$, then $\overline{\pi}_i(\Gamma)$ is a rotate-or-reflect group by Lemma 4.16. The case that $n_i \leq 2$ for exactly two i is a contradiction to Lemma 4.17 because D_1 and D_2 are the only abelian dihedral groups.

Lastly, consider the case that $n_i > 2$ for all $i \in [3]$. If $n_i = 4$ for all $i \in [3]$, then Γ is a rotate-or-reflect group or the double CFI group by Lemma 4.15. The case $n_i = n_j = 4$ and $n_k \notin \{1, 2, 4\}$ for $\{i, j, k\} = [3]$ is impossible due to Lemma 4.18. If $n_i = 4$ for exactly one $i \in [3]$, then $n_j \notin \{1, 2, 4\}$ for every $j \neq i$. Consequently, Γ is a rotate-or-reflect group by Lemma 4.19. If $n_i \neq 4$ for all $i \in [3]$, then Γ is a rotate-or-reflect group by Corollary 4.13. □

4.2.4 Combinations with Cyclic Groups

Finally, we consider 2-injective subdirect products of a mixture of dihedral and cyclic groups.

Lemma 4.20. *There are no 2-injective subdirect products of $\mathsf{C}_i \times \mathsf{D}_j \times \mathsf{D}_k$ if $i \notin \{1, 2, 4\}$.*

Proof. The proof is essentially a simpler version of the proof of Lemma 4.12. We show that the reflections in the dihedral group cannot be combined with the rotations in the cyclic groups. Suppose there is a 2-injective subdirect product $\Gamma \leq \mathsf{C}_i \times \mathsf{D}_j \times \mathsf{D}_k$. We argue first that there is an element $(r, \alpha_2, \alpha_3) \in \Gamma$. Let $\alpha \in \mathsf{D}_j$ be a reflection and $(r, \alpha, g) \in \Gamma$ for some $r \in \mathsf{C}_i$ and $g \in \mathsf{D}_k$. If g is a reflection, we are done. Otherwise, $g = r_3$. Let $\alpha' \in \mathsf{D}_k$ and $(r', g', \alpha') \in \Gamma$ be arbitrary (the latter exists because Γ is subdirect). Again, if g' is a reflection, we are done. Otherwise, $g' = r_2$ and $(r, \alpha, r_3)(r', r_2, \alpha') = (rr', \alpha r_2, r_3\alpha')$ is the desired group element. Then $\mathsf{ord}(r) \leq 2$ by Lemma 4.7(b).

Let $s \in \mathsf{C}_i$ be a rotation with $\mathsf{ord}(s) \notin \{1, 2, 4\}$ and $(s, g, h) \in \Gamma$ be some group element. By Lemma 4.7(b), one of g and h must have order greater 2, say w.l.o.g. $g = r_2$. If $h = r_3$ is a rotation, too, then $(r, \alpha_2, \alpha_3)(s, r_2, r_3) = (rs, \alpha_2 r_2, \alpha_3 r_3) \in \Gamma$. This contradicts Lemma 4.7(b) because $\mathsf{ord}(rs) > 2$. If $h = \beta_3$ is another reflection, then $(r, \alpha_2, \alpha_3)(s, r_2, \beta_3)^2 = (rs^2, \alpha_2 r_2^2, \alpha_3) \in \Gamma$. As before, $\mathsf{ord}(rs^2) > 2$ but the other components are reflections contradicting Lemma 4.7(b). □

Lemma 4.21. *Let $\Gamma \leq \mathsf{C}_i \times \mathsf{D}_j \times \mathsf{D}_k$ be a 2-injective subdirect product, $i \leq 2$, and $j, k > 2$. Then $\overline{\pi}_1(\Gamma)$ is a rotate-or-reflect group.*

Proof. The case $i = 2$ follows immediately from Lemma 4.16 because $\mathsf{C}_2 \cong \mathsf{D}_1$. The case $i = 1$ implies that $H = H_1 = \Gamma$ (Theorem 4.9). Because H_1 defines an isomorphism between $\pi_2(H_1)$ and $\pi_3(H_1)$ (Lemma 4.8), $\pi_1(\Gamma)$ is a rotate-or-reflect-group. □

Lemma 4.22. *Let $\Gamma \leq \mathsf{C}_4 \times \mathsf{D}_{n_2} \times \mathsf{D}_{n_3}$ be a 2-injective subdirect product and $n_i > 2$ for $i \in \{2, 3\}$. Then $n_2 = n_3 = 4$ and $\Gamma \cong \Gamma_{\mathsf{2CFI}} \cap (\mathsf{Rot}(\mathsf{D}_4) \times \mathsf{D}_4 \times \mathsf{D}_4)$.*

Proof. We first show that $[\Gamma : H] > 1$. Suppose that $[\Gamma : H] = 1$. By Theorem 4.9 it holds that $\pi_i(H) = \mathsf{D}_{n_i}$ for every $i \in \{2, 3\}$ and hence that $\pi_i(H_1) = \mathsf{D}_{n_i}$ for every $i \in \{2, 3\}$ because H_2 and H_3 can only use rotations (they are normal subgroups and define isomorphisms to subgroups of C_4). Furthermore, $\pi_1(H_i) = \mathsf{C}_4$ for some $i \in \{2, 3\}$ because $\pi_1(H) = \mathsf{C}_4$, say w.l.o.g. for $i = 2$. So there is an element $g = (1, \alpha_2, \alpha_3) \in H_1$ and an element $h = (r_1, r_2, 1) \in H_3$ such that $\mathsf{ord}(r_1) > 2$. But then $gh = (r_1, \alpha_2 r_2, \alpha_3) \in \Gamma$ contradicting Lemma 4.7(b). So $[\Gamma : H] > 1$ and in particular $\pi_1(H) \in \{\mathsf{C}_2, \mathsf{C}_1\}$.

Next, we want to show that there is an element $(1, \alpha'_2, \alpha'_3) \in \Gamma$ and $n_2 \geq 4$. We know there is an element $g = (r_1, \alpha_2, g_3) \in \Gamma$. We make a case distinction on g_3.

- Assume $g_3 = \alpha_3$ and then, by Lemma 4.7(b), $\mathsf{ord}(r_1) = 2$. Let $s_1 \in \mathsf{C}_4$ be of order 4 and consider the element $h = (s_1, h_2, h_3) \in \Gamma$. Hence,

  $$gh^2 = (r_1 s_1^2, \alpha_2 h_2^2, \alpha_3 h_3^2) = (1, \alpha'_2, \alpha'_3).$$

  One of h_2 and h_3 must be of order greater 2, so be a rotation. Assume w.l.o.g. $h_2 = s_2$ (in both cases there is an element $(1, \alpha'_2, \alpha'_3)$). Now

  $$gh = (r_1 s_1, \alpha_2 s_2, \alpha_3 h_3) = (s^{-1}, \alpha_2 s_2, \alpha_3 h_3).$$

  Because $\mathsf{ord}(s_1^{-1}) = 4$ and $\mathsf{ord}(\alpha_2 s_2) = 2$, it holds that $\mathsf{ord}(\alpha_3 h_3) = 4$. In particular, $h_3 = \alpha'_3$ is a reflection and hence s_2 is of order 4. So we have $n_2 \geq 4$.

- Assume $g_3 = r_3$. Let $s_2 \in \mathsf{D}_{n_2}$ be a rotation of maximal order $\mathsf{ord}(s_2) = n_2$. Then $s_2\alpha_2 \neq \alpha_2 s_2$ and there is an element $h = (s_1, s_2, h_3) \in \Gamma$. If $h_3 = s_3$, then $gh = (r_1 s_1, \alpha_1 s_2, r_3 s_3)$ and $hg = (r_1 s_1, s_2 \alpha_1, r_3 s_3)$, which is a contradiction to 2-injectivity.

  So $h_3 = \beta_3$ must be a reflection. By Lemma 4.7(b), $\mathsf{ord}(s_1) = 4$ because $\mathsf{ord}(s_2) > 2$ (and C_4 contains only rotations of order 1, 2, and 4). Then also $\mathsf{ord}(s_2) = 4$ implying $n_2 \geq 4$. Considering $gh = (r_1 s_1, \alpha_1 s_2, r_3 \beta_3)$ and Lemma 4.7(b), we obtain $\mathsf{ord}(r_1 s_1) \in \{1, 2\}$. So either $r_1 s_1 = 1$ or $r_1 s_1^{-1} = 1$ (recall that $\mathsf{ord}(r_1) = 2$) and one of gh and gh^{-1} is equal to $(1, \alpha'_2, \alpha'_3) \in \Gamma$.

It follows that $\pi_2(H_1) = \pi_2(H) = \mathsf{D}_{n_2/2}$ (because it contains reflections) and, by Theorem 4.9, that $\pi_3(H_1) = \mathsf{D}_{n_3/2}$ and thus $n_2 = n_3$ (H_1 defines an isomorphism) and $\pi_1(H) = \mathsf{C}_2$.

We now show that $n_2 = n_3 = 4$. Assume $n_2 > 4$ and let $h = (s_1, s_2, \beta_3)$ be the element as in the case $g_3 = r_3$. Recall that $\mathsf{ord}(s_2) = n_2$. There is a rotation $s'_2 \in \mathsf{D}_{n_2}$ such that $\mathsf{ord}(s_2 s'_2) \notin \{1, 2, 4\}$: If $n_2 \neq 8$, we pick $s'_2 := s_2$. Then $\mathsf{ord}(s_2 s'_2) \in \{n_2, n_2/2\}$. If $n_2 = 8$, we pick $s'_2 := s_2^2$ and then $\mathsf{ord}(s_2 s'_2) = 8$. We consider an element $g' = (s'_1, s'_2, g'_3) \in \Gamma$ and $g'_2 = s'_3$ by Lemma 4.7 because $\mathsf{ord}(s'_1) \in \{1, 2, 4\}$. Finally, $hg' = (s_1 s'_1, s_2 s'_2, \beta_3 s'_3)$,

$\mathsf{ord}(s_2s_2') \notin \{1,2,4\}$, and $\mathsf{ord}(s_1s_1'), \mathsf{ord}(\beta_3s_3') \in \{1,2,4\}$ contradicting Lemma 4.7. The case $n_3 > 4$ follows by symmetry.

Now, Γ contains elements generating a group isomorphic to $\Gamma_{\mathsf{2CFI}} \cap (\mathsf{Rot}(\mathsf{D}_4) \times \mathsf{D}_4 \times \mathsf{D}_4)$. One can show this using the same argument with conjugacy classes of reflections as in the proof of Lemma 4.15. The order of this group is 16, the order of H is 8, so Γ cannot contain any other elements. □

Proof of Theorem 4.6. From Lemma 4.20 it follows that $n_1 \in \{1,2,4\}$. Assume $n_1 \leq 2$. If $n_2, n_3 > 2$, then $\overline{\pi}_1(\Gamma)$ is a rotate-or-reflect group due to Lemma 4.21. If $n_2 \leq 2$, then $n_3 \leq 2$ by Lemma 4.17 and Γ is abelian. Assume otherwise $n_1 = 4$. If $n_2, n_3 \leq 2$, then there is an element $(g_1, g_2, g_3) \in \Gamma$ and $\mathsf{ord}(g_1) = 4$, but $\mathsf{ord}(g_2), \mathsf{ord}(g_3) \leq 2$ contradicting Lemma 4.7(b). Otherwise, $n_2, n_3 > 2$ by Lemma 4.17. Then $n_2 = n_3 = 4$ and $\Gamma \cong \Gamma_{\mathsf{2CFI}} \cap (\mathsf{Rot}(\mathsf{D}_4) \times \mathsf{D}_4 \times \mathsf{D}_4)$ by Lemma 4.22. By Lemma 4.17, 2-injective subdirect products of a non-abelian dihedral group and two abelian groups do not exist. □

4.3 Normal Forms for Structures

In this section we will introduce a certain normal form for q-bounded relational structures. In said normal form, 2-injective subdirect products play an important role as local automorphism groups. We will show that arbitrary q-bounded τ-structures can be reduced to this normal form in CPT in the following canonization-preserving manner:

Definition 4.23 (Canonization-Preserving Reduction). A **canonization-preserving** CPT-**reduction** from a class of τ-structures $\mathcal{K}$ to a class of σ-structures $\mathcal{J}$ is a pair of CPT-interpretations (Θ, Π) with the following properties:

(a) Θ is a $\mathrm{CPT}[\tau,\sigma]$-interpretation mapping $\mathcal{K}$-structures to $\mathcal{J}$-structures.

(b) Π is a $\mathrm{CPT}[\tau \uplus (\sigma \uplus \{\leq\}), \tau \uplus \{\leq\}]$-interpretation mapping pairs of a $\mathcal{K}$-structure and an ordered $\mathcal{J}$-structure to ordered $\mathcal{K}$-structures.

(c) For every CPT-canonization Υ of $\mathcal{J}$, the CPT-interpretation mapping $\mathfrak{A} \in \mathcal{K}$ to $\Pi((\mathfrak{A}, \Upsilon(\Theta(\mathfrak{A}))))$ is a canonization of $\mathcal{K}$.

We also say that $\mathcal{K}$ can be reduced canonization preservingly in CPT to $\mathcal{J}$ if there is a canonization-preserving CPT-reduction from $\mathcal{K}$ to $\mathcal{J}$.

That is, if $\mathcal{K}$ can be reduced canonization preservingly to $\mathcal{J}$ in CPT, then a CPT-canonization of $\mathcal{J}$ implies a CPT-canonization of $\mathcal{K}$. Note that the interpretation Π not only takes the canonized $\mathcal{J}$-structure but also the original $\mathcal{K}$-structure as input. In particular, this allows for the possibility that Θ is not injective. For example, Θ can remove some parts of the input which can be canonized by Π and thus produce the same $\mathcal{J}$-structure for different $\mathcal{K}$-structures. The concept of a canonization-preserving reduction is akin to the concept of Levin reductions between problems in NPTIME that allow us to pull certificates back. These reductions are thus certificate-preserving.

Before we can turn to 2-injective subdirect products as local automorphism groups, we ensure that the structures satisfy some more basic properties. It will be necessary to distinguish between the arity of relations connecting different color classes and the ones only connecting tuples inside the same color class.

Definition 4.24 (Heterogeneous Tuples and Arity). A relation $R^{\mathfrak{A}}$ is **homogeneous** if, for every $\bar{u} \in R^{\mathfrak{A}}$, there is a color class $C \in \mathbb{C}_{\mathfrak{A}}$ such that $\bar{u} \in C^{\text{ar}(R)}$. Otherwise, $R^{\mathfrak{A}}$ is **heterogeneous**. The **heterogeneous arity** of a q-bounded relational structure $\mathfrak{A}$ is the minimal number r such that $\text{ar}(R) \leq r$ for every heterogeneous relation $R^{\mathfrak{A}}$ of $\mathfrak{A}$.

For all reductions we are going to define it will be important that we preserve the occurring automorphism groups of the color classes, that is, we do not change the structure of the color classes essentially.

Definition 4.25 (Preserving Automorphism Groups). A canonization-preserving CPT reduction (Θ, Π) from a class of q-bounded τ-structures $\mathcal{K}$ to another class of q'-bounded τ'-structures $\mathcal{J}$ **preserves the automorphism groups of the color classes** if, for every $\mathfrak{A} \in \mathcal{K}$ and every $C \in \mathbb{C}_{\Theta(\mathfrak{A})}$, there is a $C' \in \mathbb{C}_{\mathfrak{A}}$ such that $\mathsf{Aut}(\Theta(\mathfrak{A})[C])$ is a section (i.e., quotient group of a subgroup) of $\mathsf{Aut}(\mathfrak{A}[C'])$.

4.3.1 Transitivity and Clean Relations

In the first step, we will achieve that for every set I of s many color classes of the structure, the automorphism group of the substructure induced by the color classes in I acts transitively on every color class in I.

Definition 4.26 (Color-Class-Transitive). A q-bounded structure $\mathfrak{A}$ is called **s-color-class-transitive** if, for every $I \subseteq \mathbb{C}_{\mathfrak{A}}$ satisfying $|I| \leq s$, the group $\mathsf{Aut}(\mathfrak{A}[I])|_C$ is transitive for every $C \in I$.

Lemma 4.27. *For all $q, k \in \mathbb{N}$ and every signature τ,* CPT *distinguishes k-orbits for the class of τ-structures of order at most q.*

Proof. Recall that to distinguish k-orbits in CPT, one has to define the k-orbit partition and to order the k-orbits. The k-orbits of a structure $\mathfrak{A}$ are exactly the equivalence classes on k-tuples induced by first order logic with $|\mathfrak{A}|$ many variables. These classes can be ordered [98]. Because $|\mathfrak{A}|$ is bounded by q, there is a CPT-term defining the k-orbits and a CPT-formula defining a total order on them. □

We use the former lemma to preserve the automorphism groups of small structures when only considering some of their orbits: For a union of k-orbits $V \subseteq A^k$ of a structure $\mathfrak{A}$ of order at most q, we define ℓ additional relations that induce the same automorphism group on V as $\mathfrak{A}$ induces on V.

Lemma 4.28. *For all $q, k \in \mathbb{N}$, every signature τ, and $\ell = (q^k)^{(q^k)}$, there are* CPT-*formulas $\Phi_1(x, \bar{y}), \dots, \Phi_\ell(x, \bar{y})$ with $|\bar{y}| = q^k$ such that the following holds: For every τ-structure $\mathfrak{A}$ of order $|A| \leq q$ and every union of k-orbits $V \subseteq A^k$, the relations*

$$S_i^{\mathfrak{A}} = \left\{ \bar{u} \in (V^k)^{(q^k)} \;\middle|\; (V, \bar{u}) \in \Phi_i^{\mathfrak{A}} \right\}$$

(tuples in V are Kuratowski-encoded) satisfy for the permutation group Γ with domain A^k induced by $\mathsf{Aut}(\mathfrak{A})$ *that* $\mathsf{Aut}((V^k, S_1^{\mathfrak{A}}, \dots, S_\ell^{\mathfrak{A}})) = \Gamma|_V$.

Proof. We use Lemma 4.27 to define and order the $(k \cdot q^k)$-orbits of $\mathfrak{A}$. We identify $A^{k \cdot (q^k)}$ with $(A^k)^{q^k}$ via the natural bijection. In this way, we only consider the $(k \cdot q^k)$-orbits, which are a subset of $(V^k)^{(q^k)}$. Note that $|V| \leq |A|^k \leq q^k$ and that there are at most $|A|^{(k \cdot q^k)} \leq \ell$ many such orbits. For every $i \in [\ell]$, we define Φ_i to be satisfied for the tuples in the i-th such orbit or to be never satisfied if i exceeds the number of orbits.

By construction, every $\varphi \in \mathsf{Aut}((V^k, S_1^{\mathfrak{A}}, \dots, S_\ell^{\mathfrak{A}}))$ can only map q^k-tuples (of V^k) onto each other which are in the same q^k-orbit of Γ. So for every $\varphi \in \mathsf{Aut}((V^k, S_1^{\mathfrak{A}}, \dots, S_\ell^{\mathfrak{A}}))$, there is a $\psi \in \mathsf{Aut}(\mathfrak{A})$ inducing the action of φ on V^k, i.e., $\mathsf{Aut}((V^k, S_1^{\mathfrak{A}}, \dots, S_{q^q}^{\mathfrak{A}})) \subseteq \Gamma|_V$. We have equality because every $\psi \in \mathsf{Aut}(\mathfrak{A})$ set-wise stabilizes all $(k \cdot q^k)$-orbits, i.e., stabilizes all $S_i^{\mathfrak{A}}$. □

Note that the restriction that V is a union of orbits is no restriction at all because every CPT-definable set is a union of orbits.

Lemma 4.29. *For all numbers $q, r, s \in \mathbb{N}$ and every signature τ, there is a signature σ and a canonization-preserving* CPT*-reduction from q-bounded τ-structures of heterogeneous arity r to q-bounded and s-color-class-transitive σ-structures of heterogeneous arity r. The reduction preserves the automorphism groups of the color classes.*

Proof. Let $q, r, s \in \mathbb{N}$ and τ be a signature. Set $\sigma := \tau \uplus (\{S\} \cup \{S_i \mid i \in [q^q]\})$, where $\mathrm{ar}(S) = 2$ and $\mathrm{ar}(S_i) = q$ for every $i \in [q^q]$. We define a CPT$[\tau, \sigma]$-interpretation Θ as follows: Let $\mathfrak{A}$ be a q-bounded τ-structures of heterogeneous arity r. Set

$$J := \left\{ I \subseteq \mathbb{C}_{\mathfrak{A}} \;\middle|\; |I| \leq s \right\}$$

to be the set of all sets of up to s many color classes. The total order of the color classes extends to a total order on $J = \{I_1, \dots, I_m\}$. For every set $I \in J$, recall that $\mathsf{orb}(\mathfrak{A}[I])$ denotes the partition of the structure $\mathfrak{A}[I]$ into 1-orbits. For every atom $u \in A$, let $\mathsf{orb}_I(u)$ be the 1-orbit of $\mathfrak{A}[I]$ containing u if $u \in A[I]$ and $\emptyset$ otherwise. Finally, we define

$$\mathsf{orb}(u) := \Big(\mathsf{orb}_{I_1}(u), \dots, \mathsf{orb}_{I_m}(u)\Big).$$

Note that there is a lexicographical order $\preceq_{\mathrm{lex}}$ on $\bigotimes_{i \in [m]}(\mathsf{orb}(\mathfrak{A}[I_i]) \cup \{\emptyset\})$ given by the order on J and the orders on the $\mathsf{orb}(\mathfrak{A}[I_i])$ from Lemma 4.27.

We now define the σ-structure $\mathfrak{B} = \Theta(\mathfrak{A})$ on the universe $B := A$. For every relation $R \in \tau$, we define $R^{\mathfrak{B}} := R^{\mathfrak{A}}$. We set $S^{\mathfrak{B}} := \preceq^{\mathfrak{A}}$ and refine the preorder $\preceq^{\mathfrak{A}}$ by the preorder $u \preceq^{\mathfrak{B}} v := \mathsf{orb}(u) \preceq_{\mathrm{lex}} \mathsf{orb}(v)$. This preorder is CPT-definable using Lemma 4.27. To define the relations S_i, let C' be a color class of the preorder $\preceq^{\mathfrak{B}}$. Because $\preceq^{\mathfrak{B}}$ refines $\preceq^{\mathfrak{A}}$, there is a color class C of $\preceq^{\mathfrak{A}}$ such that $C' \subseteq C$. We exploit Lemma 4.28 (for $k = 1$) to define q-ary relations $S_{C',1}^{\mathfrak{B}}, \dots, S_{C',q^q}^{\mathfrak{B}}$ over C' such that

$$\mathsf{Aut}((C', S_{C',1}^{\mathfrak{B}}, \dots, S_{C',q^q}^{\mathfrak{B}})) = \mathsf{Aut}(\mathfrak{A}[C])|_{C'}.$$

We combine them for every color class C' of $\preceq^{\mathfrak{B}}$ into the q-ary relations $S_1^{\mathfrak{B}}, \dots, S_{q^q}^{\mathfrak{B}}$:

$$S_i^{\mathfrak{B}} := \bigcup_{C' \in \mathbb{C}_{\mathfrak{B}}} S_{C',i}^{\mathfrak{B}}$$

for every $i \in [q^q]$. The relation S_i is homogeneous for every $i \in [q^q]$. The structure $\mathfrak{B}$ is clearly q-bounded because we only split color classes. It has heterogeneous arity r

because we only added homogeneous relations. Moreover, $\mathfrak{B}$ is s-color-class-transitive by the definition of $\preceq^{\mathfrak{A}'}$.

It is easy to see that Θ can be extended to a canonization-preserving CPT-reduction (Θ, Π): The interpretation Π interprets $\preceq$ in the canon of $\mathfrak{A}$ as S in the canon of $\mathfrak{B}$ and all $R \in \tau$ are interpreted in the canon of $\mathfrak{A}$ as in the canon of $\mathfrak{B}$. To show that (Θ, Π) preserves the automorphism groups of the color classes, let $C' \in \mathbb{C}_{\mathfrak{B}}$. By construction, there is a $C \in \mathbb{C}_{\mathfrak{A}}$ such that $C' \subseteq C$. Because of the application of Lemma 4.28, it holds that $\mathsf{Aut}(\mathfrak{B}[C']) = \mathsf{Aut}(\mathfrak{A}[C])|_{C'}$. Thus, $\mathsf{Aut}(\mathfrak{B}[C'])$ is a quotient of the set-wise stabilizer $\mathsf{Stab}_{\mathsf{Aut}(\mathfrak{A}[C])}(C') \leq \mathsf{Aut}(\mathfrak{A}[C])$, i.e., $\mathsf{Aut}(\mathfrak{A}'[C'])$ is a section of $\mathsf{Aut}(\mathfrak{A}[C])$. □

In the previous lemma, we added homogeneous relations whose arity is possibly larger than the arity of the original structure (by applying Lemma 4.28). This was necessary to preserve the automorphism groups of the color classes. We remark that in the canonization for abelian color classes [103, Theorem 6.8] it is not necessary to augment the structure to maintain the automorphism groups. The next step is to ensure that every tuple in a heterogeneous relation contains at most one atom of the same color class. We formalize the notion as follows.

Definition 4.30 (Clean Relation). Let $\mathfrak{A}$ be a q-bounded τ-structure and $R \in \tau$. The relation $R^{\mathfrak{A}}$ is **clean** if, for every $\bar{u} \in R^{\mathfrak{A}}$, there are distinct $C_1, \dots, C_{\mathrm{ar}(R)} \in \mathbb{C}_{\mathfrak{A}}$ such that $\bar{u} \in C_1 \times \dots \times C_{\mathrm{ar}(R)}$. The structure $\mathfrak{A}$ is **clean** if $R^{\mathfrak{A}}$ is clean or homogeneous for every $R \in \tau$.

Definition 4.31 (Basic Constituent). Let $\mathfrak{A}$ be a q-bounded τ-structure. We call a tuple $T = (C_1, \dots, C_\ell) \in \mathbb{C}_{\mathfrak{A}}^\ell$ a **basic constituent** of $\mathfrak{A}$ if $R^{\mathfrak{A}} \cap (C_1 \times \dots \times C_\ell) \neq \emptyset$ for some $R \in \tau$ and there are i, j such that $C_i \neq C_j$. The set of all basic constituents of $\mathfrak{A}$ is denoted by $\mathbb{T}^{\mathfrak{A}}$. We omit the subscript if $\mathfrak{A}$ becomes clear from the context.

Note that tuples contained in different basic constituents can never be mapped onto each other by an automorphism.

Lemma 4.32. *For all numbers $q \in \mathbb{N}$ and $r \geq 2$ and every signature τ, there is a signature σ and a canonization-preserving* CPT*-reduction from q-bounded τ-structures of heterogeneous arity r to q-bounded and clean σ-structures of heterogeneous arity r. The reduction preserves s-color-class-transitivity for every $s \in \mathbb{N}$. It preserves the automorphism groups of the color classes.*

Proof. Let $q \in \mathbb{N}$, $r \geq 2$, and τ be a signature. Set $\sigma := \tau \uplus \{S\}$. We define a CPT$[\tau, \sigma]$-interpretation Θ. Let $\mathfrak{A}$ be a q-bounded τ-structure of heterogeneous arity r. The σ-structure $\mathfrak{B} = \Theta(\mathfrak{A})$ is defined as follows: We set $B := A \times [r]$ and define $\preceq^{\mathfrak{B}}$ such that $(u, i) \preceq^{\mathfrak{B}} (v, j)$ if and only if (u, i) is lexicographically smaller than (v, j) for all $u, v \in A$ and $i, j \in [r]$. We define

$$\begin{aligned}
S^{\mathfrak{B}} &:= \Big\{ \big((u,i),(u,j)\big) \;\Big|\; u \in A, i \neq j \in [r] \Big\}, \\
R^{\mathfrak{B}} &:= \Big\{ \big((u_1,1),\dots,(u_{\mathrm{ar}(R)},\mathrm{ar}(R))\big) \;\Big|\; \bar{u} \in R^{\mathfrak{A}} \Big\} && \text{if } R^{\mathfrak{A}} \text{ is heterogeneous,} \\
R^{\mathfrak{B}} &:= \Big\{ \big((u_1,i),\dots,(u_{\mathrm{ar}(R)},i)\big) \;\Big|\; \bar{u} \in R^{\mathfrak{A}}, i \in [r] \Big\} && \text{if } R^{\mathfrak{A}} \text{ is homogeneous,}
\end{aligned}$$

for every $R \in \tau$. The interpretation is clearly CPT-definable.

For all $C \in \mathbb{C}_{\mathfrak{A}}$ and $i \in [r]$, we denote by C_i the set $C \times \{i\} \subseteq B$. By construction of $\preceq^{\mathfrak{A}}$, the color classes of $\mathfrak{B}$ are exactly the C_i for all $C \in \mathbb{C}_{\mathfrak{A}}$ and $i \in [r]$. We see that $R^{\mathfrak{B}}$ is homogeneous if $R^{\mathfrak{A}}$ is homogeneous for every $R \in \tau$. We also see, for every $R \in \tau$, that the relation $R^{\mathfrak{B}}$ is clean if $R^{\mathfrak{A}}$ is heterogeneous because (u,i) and (v,j) are never in the same color class if $i \neq j$. Finally, $S^{\mathfrak{B}}$ is a heterogeneous relation of arity $2 \leq r$. So $\mathfrak{B}$ is a clean q-bounded σ-structure of heterogeneous arity r.

We complete Θ to a canonization-preserving CPT-reduction (Θ, Π) as follows: Note that, for all $C \in \mathbb{C}_{\mathfrak{A}}$ and $i \neq j \in [r]$, the relation $S^{\mathfrak{B}}$ connects C_i and C_j by a perfect matching identifying the copies (u,i) and (u,j) of the same atom $u \in A$. That is, when contracting S in a canon of $\mathfrak{B}$, we obtain a canon of $\mathfrak{A}$, which is easily CPT-definable. To see that (Θ, Π) preserves the automorphism groups of the color classes, simply note that $\mathfrak{B}[C_i] \cong \mathfrak{A}[C]$ by construction (homogeneous relations are copied into C_i) for every $C \in \mathbb{C}_{\mathfrak{A}}$ and $i \in [r]$.

Finally, let $s \in \mathbb{N}$ and assume that $\mathfrak{A}$ s-color-class-transitive. Let

$$I' = \left\{C_{i_1}^{(1)}, \dots, C_{i_k}^{(k)}\right\} \subseteq \mathbb{C}_{\mathfrak{B}}$$

be a set of $k \leq s$ many color classes for $C^{(1)}, \dots, C^{(k)} \in \mathbb{C}_{\mathfrak{A}}$ and $i_1, \dots, i_k \in [r]$. We consider the substructure $\mathfrak{B}[I']$ and contract $S^{\mathfrak{B}}$ yielding the structure $\mathfrak{B}_{\sim I'}$. It suffices to show that $\mathsf{Aut}(\mathfrak{B}_{\sim I'})$ acts transitively on every color class of $\mathfrak{B}_{\sim I'}$ because $S^{\mathfrak{B}}$ just identifies the copies of the atoms in $\mathfrak{B}$. Set $I := \{C^{(1)}, \dots, C^{(k)}\}$. Then $\mathfrak{B}_{\sim I'} \subseteq \mathfrak{A}[I]$ (after identifying contracted copies of u in $\mathfrak{B}_{\sim I'}$ with $u \in A$) by definition of $\mathfrak{B}$. The structure $\mathfrak{B}_{\sim I'}$ is not necessarily equal to $\mathfrak{A}[I]$ because some tuples in relations of $\mathfrak{B}_{\sim I'}$ are missing. However, every tuple $\bar{u} \in R^{\mathfrak{A}[I]} \setminus R^{\mathfrak{B}_{\sim I'}}$ is contained in a different basic constituent of $\mathfrak{A}$ than every tuple $\bar{v} \in R^{\mathfrak{A}[I]} \cap R^{\mathfrak{B}_{\sim I'}}$ for every $R \in \tau$. That is, $\mathsf{Aut}(\mathfrak{A}[I]) \subseteq \mathsf{Aut}(\mathfrak{B}_{\sim I'})$ and thus $\mathsf{Aut}(\mathfrak{B}_{\sim I'})$ acts transitively on every color class $C_{i_j}^{j}$ for $j \in [k]$ because $\mathsf{Aut}(\mathfrak{A}[I])$ does so. □

4.3.2 2-Injective Subdirect Products and Quotients

In the end, we want to achieve that the automorphism group of three color classes is always a 2-injective subdirect product. We also need the more general notion of r-injective subdirect products, which is a straightforward generalization: A group $\Gamma \leq \bigotimes_{i \in [r+1]} G_i$ is called a **subdirect product** if $\pi_i^{\Gamma}(\Gamma) = G_i$ for every $i \in [r+1]$. It is called r**-injective** if $\mathsf{ker}(\overline{\pi}_i^{\Gamma}) = 1$ for every $i \in [r+1]$. Before we can proceed to reduce to r-injective subdirect products, we need to modify the color classes to allow further operations on them.

Definition 4.33 (Regular Color Classes)**.** Let $\mathfrak{A}$ be a q-bounded structure. A color class $C \in \mathbb{C}_{\mathfrak{A}}$ is called **regular** if $\mathsf{Aut}(\mathcal{C})$ is a regular permutation group. The structure $\mathfrak{A}$ **has regular color classes** if every $C \in \mathbb{C}_{\mathfrak{A}}$ is regular.

Recall that $\mathcal{C}$ denotes the structure $\mathfrak{A}[C]$ if $\mathfrak{A}$ becomes clear from the context. Also, recall that a permutation group $\Gamma \leq \mathsf{Sym}(\Omega)$ is regular if Γ is transitive and $|\Omega| = |\Gamma|$. We now show that we can replace every color class C of a structure by a regular color class C' satisfying $\mathsf{Aut}(\mathcal{C}) \cong \mathsf{Aut}(\mathcal{C}')$.

Lemma 4.34. *For every permutation group $\Gamma \leq \mathsf{Sym}(\Omega)$ with domain Ω, there is a $|\Omega|$-orbit of Γ on which Γ acts regularly.*

Proof. Assume w.l.o.g. that $\Omega = [k]$ and consider the k-orbit O of Γ containing the k-tuple $(1, \dots, k)$. By definition of an orbit, Γ acts transitively on O. By construction of O, if $\varphi, \psi \in \Gamma$ satisfy $\varphi(\bar{u}) = \psi(\bar{u})$ for some $\bar{u} \in O$, then we have $\varphi = \psi$ because, for every $\bar{u}$, it holds that $\Omega = \{u_i \mid i \in [k]\}$ (otherwise $\bar{u}$ cannot be in the same orbit as $(1, \dots, k)$). $\square$

Lemma 4.35. *For all $q, r \in \mathbb{N}$ and every signature τ, there are q', σ, and a canonization-preserving* CPT*-reduction from q-bounded and clean τ-structures of heterogeneous arity r to q'-bounded and clean σ-structures of heterogeneous arity r with regular color classes. The reduction preserves s-color-class-transitivity for every $s \in \mathbb{N}$. It preserves the automorphism groups of the color classes.*

Proof. Let $q, r \in \mathbb{N}$ and τ be a signature. Set $q' := q^q$, $\ell := q'^{q'}$, and $\sigma := \tau \uplus (\{S\} \uplus \{S_1, \dots, S_\ell\})$, where $\mathrm{ar}(S) = 2$ and $\mathrm{ar}(S_i) = q'$ for every $i \in [\ell]$. We now define a $\mathrm{CPT}[\tau, \sigma]$-interpretation Θ. Let $\mathfrak{A}$ be a clean q-bounded τ-structure of heterogeneous arity r. The structure $\mathfrak{B} := \Theta(\mathfrak{A})$ is defined as follows: For every color class $C \in \mathbb{C}_{\mathfrak{A}}$, we define, using Lemma 4.27, the minimal q-orbit O_C of $\mathfrak{A}[C]$ such that the induced action of $\mathsf{Aut}(\mathfrak{A}[C])$ on O_C is regular. Such an orbit always exists by Lemma 4.34. Next, we define, using Lemma 4.28, for every color class $C \in \mathbb{C}_{\mathfrak{A}}$, q'-ary relations $S^{\mathfrak{B}}_{C,1}, \dots, S^{\mathfrak{B}}_{C,\ell}$ on O_C such that the group $\mathsf{Aut}((O_C^{q'}, S^{\mathfrak{B}}_{C,1}, \dots, S^{\mathfrak{B}}_{C,\ell}))$ is isomorphic to the permutation group Γ_C on $O_C^{q'}$ induced by $\mathsf{Aut}(\mathfrak{A}[C])$. We define

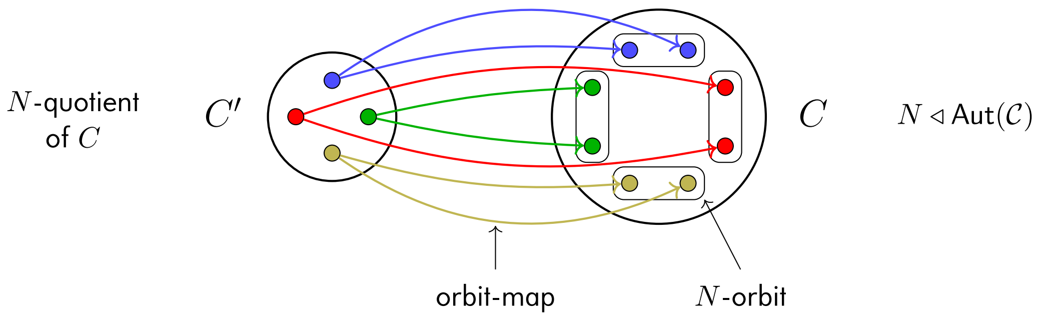
$$\begin{aligned}
B &:= \bigcup_{C \in \mathbb{C}_{\mathfrak{A}}} O_C,\\
S^{\mathfrak{B}} &:= \left\{ (\bar{u}, \bar{v}) \in B^2 \;\middle|\; u_1 = v_1 \right\},\\
S_i^{\mathfrak{B}} &:= \bigcup_{C \in \mathbb{C}_{\mathfrak{A}}} S^{\mathfrak{B}}_{C,i},\\
R^{\mathfrak{B}} &:= \left\{ (\bar{u}^1, \dots, \bar{u}^{\mathrm{ar}(R)}) \in B^{\mathrm{ar}(R)} \;\middle|\; (u^1_1, \dots, u_1^{\mathrm{ar}(R)}) \in R^{\mathfrak{A}} \right\},
\end{aligned}$$

for all $R \in \tau$ and $i \in [\ell]$. The preorder $\preceq^{\mathfrak{B}}$ is defined such that $\bar{u} \preceq^{\mathfrak{B}} \bar{v}$ if and only if $\bar{u} \in C_1^q$ and $\bar{v} \in C_2^q$ for some $C_1, C_2 \in \mathbb{C}_{\mathfrak{A}}$ with $C_1 \preceq^{\mathfrak{A}} C_2$. That is, we replace every color class $C \in \mathbb{C}_{\mathfrak{A}}$ with a color class $C' \in \mathbb{C}_{\mathfrak{B}}$ on which $\mathsf{Aut}(\mathfrak{A}[C])$ acts regularly (which is the group Γ_C). Hence, $\mathfrak{B}$ has regular color classes. Because we applied Lemma 4.28, we ensured that $\mathsf{Aut}(\mathfrak{B}[C']) = \Gamma_C$, which is isomorphic to $\mathsf{Aut}(\mathfrak{A}[C])$ because the action of $\mathsf{Aut}(\mathfrak{A}[C])$ is regular. Hence, we preserve the automorphism groups of the color classes.

The relation S relates q-tuples with equal first entry. Intuitively, we use the set of q-tuples with equal first entry u to represent the atom u in the definition of the $R^{\mathfrak{B}}$. That is, we can turn Θ into a canonization-preserving CPT-reduction (Θ, Π) as follows: Given a canon of $\mathfrak{B}$, the interpretation Π contracts S to obtain a canon of $\mathfrak{A}$, which is easily CPT-definable.

We finally show that the reduction preserves s-color-class-transitivity for every $s \in \mathbb{N}$. This can easily be seen because every automorphism of a $\mathfrak{B}$-color class is induced by an automorphism of the corresponding $\mathfrak{A}$-color class and the automorphism group of every $\mathfrak{B}$-color class is transitive by construction (because $\mathfrak{B}$ has regular color classes). Hence, if $\mathfrak{A}$ is s-color-class-transitive, so is $\mathfrak{B}$. $\square$

Next, we exploit regularity of the color classes to construct quotient groups. They will be important to reduce to 2-injective subdirect products.



4.2 Quotient color classes. The figure illustrates quotient color classes. On the right, a color class C containing 8 atoms. The automorphism group $\mathsf{Aut}(C)$ is regular. Assume $N \triangleleft \mathsf{Aut}(C)$ and $N \cong \mathsf{C}_2$. Thus, every N-orbit contains two atoms. The N-orbits are drawn in different colors. The color class C' on the left is an N-quotient of C: The color class C' contains, for every N-orbit O, one atom which is adjacent via the orbit-map to the atoms in O. These edges form the orbit-map. The different colors for the edges are only used for illustration.

Definition 4.36 (Quotient Color Class)**.** Let $\mathfrak{A}$ be a q-bounded τ-structure, $C, C' \in \mathbb{C}_{\mathfrak{A}}$ be distinct regular color classes, and $N \triangleleft \mathsf{Aut}(\mathcal{C})$. We say that C' is an **N-quotient of C** if $\mathsf{Aut}(\mathcal{C}') \cong \mathsf{Aut}(\mathcal{C})/N$ and there is an $R \in \tau$ such that $R^{\mathfrak{A}}|_{C \cup C'}$ is a function $C' \to C$ determining the orbit partition of N acting on C, i.e.,

$$\Big\{ \Big\{ u \in C \;\Big|\; (u,v) \in R^{\mathfrak{A}} \Big\} \;\Big|\; v \in C' \Big\} = \mathsf{orb}(N).$$

The relation $R^{\mathfrak{A}}$ is called the **orbit-map** of C (cf. Figure 4.2).

The following lemma states that quotient groups of regular permutation groups can be defined using the orbits of the normal subgroup.

Lemma 4.37. *Let $\Gamma \leq \mathsf{Sym}(\Omega)$ be a regular permutation group and $N \triangleleft \Gamma$. Then Γ acting on the set $\mathsf{orb}(N)$ forms a regular permutation group isomorphic to Γ/N.*

Proof. Since N is normal in Γ, every permutation $\sigma \in \Gamma$ induces a permutation σ_N of the N-orbits. Thus Γ induces a permutation group Γ' on $\mathsf{orb}(N)$. One easily checks that the map $\varphi\colon \Gamma/N \to \Gamma'$ defined by $\sigma N \mapsto \sigma_N$ is an isomorphism. The group Γ' is transitive because Γ is transitive. Assume there is an N-orbit of size k. Then $|N| = k$ because otherwise Γ was not regular. But so all N-orbits have size k and

$$|\mathsf{orb}(N)| = |\Omega|/k = |\Gamma|/k = |\Gamma/N| = |\Gamma'|.$$

Hence, Γ' is regular. □

We exploit the previous lemma to construct quotient color classes.

Lemma 4.38. *For every $q \in \mathbb{N}$ and every signature τ, there is a* CPT*-term $s(x,y)$ and* CPT*-formulas $\Phi(x,y,z_1,z_2)$ and $\Psi_i(x,y,\bar{z}')$ for all $i \in [q^q]$ such that, for every q-bounded τ-structure $\mathfrak{A}$, every regular color class $C \in \mathbb{C}_{\mathfrak{A}}$, and every $N \triangleleft \mathsf{Aut}(\mathcal{C})$, the following holds:*

1. *The term s defines a set $C' = s^{\mathfrak{A}}(C, N)$ (for some fixed encoding of permutation groups as hereditarily finite sets).*

2. *The formula Φ defines a relation*

$$R_{\mathsf{orb}}^{\mathfrak{A}} := \left\{ (u, v) \in C' \times C \;\middle|\; (C, N, u, \{v\}) \in \Phi^{\mathfrak{A}} \right\}.$$

3. *For every $i \in [q^q]$, the formula Ψ_i defines an q-ary relation on C'^q*

$$S_i^{\mathfrak{A}} := \left\{ \bar{u} \in (C'^q)^q \;\middle|\; (C, N, \bar{u}) \in \Psi_i^{\mathfrak{A}} \right\}.$$

4. *In the $(\tau \uplus (\{R_{\mathsf{orb}}\} \cup \{S_i \mid i \in [q^q]\}))$-structure obtained by extending $\mathfrak{A}$ with $R_{\mathsf{orb}}^{\mathfrak{A}}, S_1^{\mathfrak{A}}, \dots, S_\ell^{\mathfrak{A}}$ and adding C' (as new color class with fresh atoms) to $\mathfrak{A}$, the color class C' is an N-quotient of C and $R_{\mathsf{orb}}^{\mathfrak{A}}$ is the orbit-map. The color class C' is regular.*

Proof. Let $q \in \mathbb{N}$, τ be a signature, $\mathfrak{A}$ be a q-bounded τ-structure, $C \in \mathbb{C}_{\mathfrak{A}}$, $\Gamma = \mathsf{Aut}(\mathcal{C})$, and $N \triangleleft \Gamma$. The term s defines $C' := \mathsf{orb}(N)$ to be the N-orbits using Lemma 4.27. The orbit-map will be realized by

$$R_{\mathsf{orb}}^{\mathfrak{A}} := \left\{ (O, u) \in C' \times C \;\middle|\; u \in O \right\}.$$

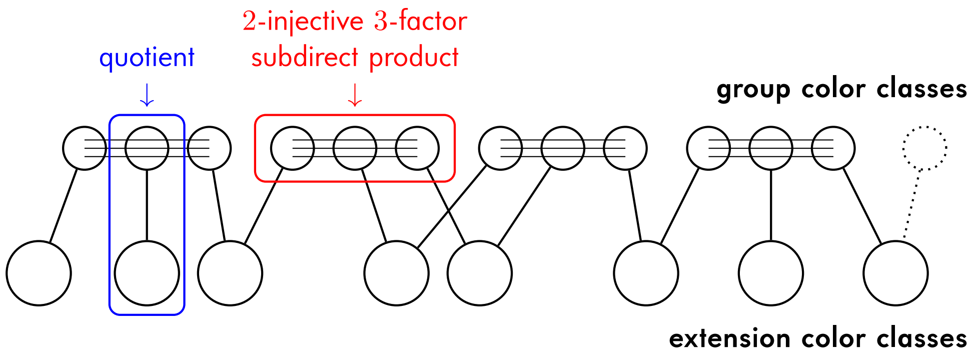
Let $\mathfrak{A}_C = (C \cup C', R_{\mathsf{orb}}^{\mathfrak{A}}, \preceq_C) \cup \mathcal{C}$ be the structure consisting of $\mathcal{C}$ and the attached new atoms and $\preceq_C$ be defined such that $C \prec_C C'$. Let $\Delta = \mathsf{Aut}(\mathfrak{A}_C) \leq \Gamma \times \mathsf{Sym}(C')$ and $(\varphi, \psi) \in \Delta$ for $\varphi \in \Gamma$ and $\psi \in \mathsf{Sym}(C')$. By Lemma 4.37, the automorphism φ permutes the N-orbits and its action is regular. Because every orbit $O \in C'$ is adjacent to exactly all atoms contained in O via R_{orb}, ψ has to be the permutation of N-orbits corresponding to φ. So $\Gamma' := \Delta|_{C'}$ is the permutation group of N-orbits given by Γ and from Lemma 4.37 it follows that $\Gamma' \cong \Gamma/N$. We apply Lemma 4.28 and define homogeneous q-ary relations $S_1^{\mathfrak{A}}, \dots, S_{q^q}^{\mathfrak{A}}$ on C' such that $\mathsf{Aut}((C', S_1^{\mathfrak{A}}, \dots, S_{q^q}^{\mathfrak{A}})) \cong \Gamma'$. Then, after adding C' and $R_{\mathsf{orb}}^{\mathfrak{A}}, S_1^{\mathfrak{A}}, \dots, S_{q^q}^{\mathfrak{A}}$ to $\mathfrak{A}$, the color class C' is an N-quotient of C and $R_{\mathsf{orb}}^{\mathfrak{A}}$ is the orbit-map. □

Now we are prepared to turn to structures with 2-injective subdirect products as local automorphism groups:

Definition 4.39 (Injective Quotient Structure). Let $\mathfrak{A}$ be a q-bounded structure. For a basic constituent $T = (C_1, \dots, C_j) \in \mathbb{T}^{\mathfrak{A}}$, we set $\Gamma_T^{\mathfrak{A}} := \mathsf{Aut}(\mathfrak{A}[\bigcup_{i \in [j]} C_i]) \leq \bigotimes_{i \in [j]} \mathsf{Aut}(\mathcal{C}_i)$. The structure $\mathfrak{A}$ is called an $(r-1)$**-injective quotient structure** if there is a partition $\{A_{\mathsf{gr}}, A_{\mathsf{ex}}\}$ of A with the following properties:

1. The structure $\mathfrak{A}$ is of heterogeneous arity r, is clean, and has regular color classes.

2. There is exactly one heterogeneous 2-ary relation $R_{\mathsf{orb}}^{\mathfrak{A}} \subseteq A_{\mathsf{ex}} \times A_{\mathsf{gr}}$ such that every other heterogeneous relation $R^{\mathfrak{A}}$ satisfies $\mathrm{ar}(R) = r$ and $R^{\mathfrak{A}} \subseteq A_{\mathsf{gr}}^r$. A color class $C \subseteq A_{\mathsf{gr}}$ (or $C \subseteq A_{\mathsf{ex}}$, respectively) is called a **group color class** (or an **extension color class**, respectively).



4.3 Injective quotient structures. The group color classes are at the top and the extension color classes at the bottom. The horizontal lines depict relations of arity 3. All three color classes connected by such a relation, that is, a basic constituent, form a 2-injective 3-factor subdirect product. A vertical line between a group and an extension color class depicts the orbit-maps.

3. For every group color class C, there is exactly one extension color class C' such that (C, C') is a basic constituent. Moreover, C is an N-quotient of C' for some $N \lhd \mathsf{Aut}(C')$ and $R^{\mathfrak{A}}_{\mathsf{orb}}$ is the orbit-map.

4. The **group constituents** $\mathbb{T}^{\mathfrak{A}}_{\mathsf{gr}} \subseteq \mathbb{T}^{\mathfrak{A}}$ of $\mathfrak{A}$ is the set of all basic constituents only consisting of group color classes. For every $T \in \mathbb{T}^{\mathfrak{A}}_{\mathsf{gr}}$, the group $\Gamma^{\mathfrak{A}}_{T}$ is an $(r-1)$-injective subdirect product.

We leave out the superscripts if the structure $\mathfrak{A}$ is clear from the context. Intuitively, an $(r-1)$-injective quotient structure consists of two different types of color classes, namely the group and extension color classes. Every group color class in a quotient of exactly one extension color class (thus the name extension color class) and connections between the group and extension color classes are established solely via the orbit-map R_{orb}. There are no connections between the extension color classes. All connections between group color classes are given by r-ary relations such that, for each group constituent $T \in \mathbb{T}^{\mathfrak{A}}_{\mathsf{gr}}$, the induced automorphism group $\Gamma^{\mathfrak{A}}_{T}$ on its r many color classes is an $(r-1)$-injective subdirect product. Also consider Figure 4.3, which shows a sketch of a 2-injective quotient structure. This means, in an $(r-1)$-injective quotient structure there are two types of local automorphism groups: $(r-1)$-injective subdirect products on r many color classes and "quotients" on two color classes, where the automorphism group of one is a quotient of the other realized by the orbit-map.

Lemma 4.40. *For all $q \in \mathbb{N}$, $2 \leq r$, and every signature τ, there is a signature σ and a canonization-preserving* CPT*-reduction* (Θ, Π)

- *from q-bounded, clean, and r-color-class-transitive τ-structures of heterogeneous arity r*
- *to q-bounded r-injective quotient σ-structures.*

If $\mathfrak{A}$ is actually of heterogeneous arity $r' \leq r$, then for every $T = (C_1, \dots, C_r) \in \mathbb{T}^{\Theta(\mathfrak{A})}_{\mathsf{gr}}$ and every $r' < i \leq r$, it holds that $|C_i| = 1$.

Proof. Let $q \in \mathbb{N}$, $2 \leq r$, and τ be a signature. We first assume that every relation in τ has arity at least r. If not, we increase the arity to r. Homogeneous relations for arity less than r are extended to arity r by repeating the last entry. For heterogeneous relations, r many new singleton color classes are added, whose atoms are used to extend tuples of length less than r to r. The atom in the i-th new singleton color classes is used in the i-th position to extend the tuples. This is clearly canonization-preserving and preserves r-color-class-transitivity and being clean.

Set $\sigma := \tau \uplus \{R_{\mathsf{orb}}, S_1, \dots, S_{q^q}\}$, where $\mathrm{ar}(R_{\mathsf{orb}}) = 2$ and $\mathrm{ar}(S_i) = q$ for every $i \in [q^q]$. We now define a CPT$[\tau,\sigma]$-interpretation Θ. Let $\mathfrak{A}$ be a q-bounded, clean, and r-color-class-transitive τ-structure of heterogeneous arity r. For every basic constituent $T = (C_1, \dots, C_r) \in \mathbb{T}^{\mathfrak{A}}$, we set $N_i^T := \mathsf{ker}(\pi_i^{\Gamma_T})$ for every $i \in [r]$. Note that $\Gamma_T^{\mathfrak{A}} := \mathsf{Aut}(\mathfrak{A}[\bigcup_{i \in [r]} C_i]) \leq \bigotimes_{i \in [r]} \mathsf{Aut}(\mathcal{C}_i)$. Because $\mathfrak{A}$ is r-color-class-transitive, it follows that $\Gamma_T^{\mathfrak{A}}$ is a subdirect product for every $T \in \mathbb{T}^{\mathfrak{A}}$. Moreover, $N_i^T \triangleleft \mathsf{Aut}(\mathcal{C}_i)$ via the canonical isomorphism $(g_1, \dots, g_r) \mapsto g_i$ (note that $g_j = 1$ for all $j \neq i$).

We first note that, for every $T \in \mathbb{T}^{\mathfrak{A}}$ and every $i \in [r]$, the group N_i^T is CPT-definable. For all $T = (C_1, \dots, C_r) \in \mathbb{T}^{\mathfrak{A}}$ and $i \in [r]$, we define using Lemma 4.38 disjoint sets $C_{T,i}$, binary relations $R^{\mathfrak{A}}_{\mathsf{orb};T,i} \subseteq C_i \times C_{T,i}$, and q-ary relations $S^{\mathfrak{A}}_{1;T,i}, \dots, S^{\mathfrak{A}}_{q^q;T,i} \subseteq C^q_{T,i}$ such that $C_{T,i}$ is the N_i^T-quotient of C and $R^{\mathfrak{A}}_{\mathsf{orb};T,i}$ is the orbit-map. We define the structure $\mathfrak{B} := \Theta(\mathfrak{A})$. Its universe is given by

$$B_{\mathsf{gr}} := \bigcup_{\substack{T \in \mathbb{T}^{\mathfrak{A}},\\ i \in [r]}} C_{T,i}, \qquad B_{\mathsf{ex}} := A, \text{ and } \qquad B := B_{\mathsf{gr}} \uplus B_{\mathsf{ex}}.$$

Relations are interpreted as follows:

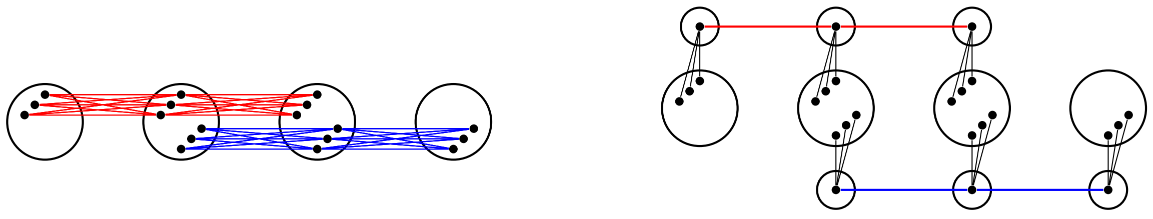
$$\begin{aligned}
R^{\mathfrak{B}}_{\mathsf{orb}} &:= \bigcup_{\substack{T \in \mathbb{T}^{\mathfrak{A}},\\ i \in [r]}} R^{\mathfrak{A}}_{\mathsf{orb};T,i} \\
S^{\mathfrak{B}}_j &:= \bigcup_{\substack{T \in \mathbb{T}^{\mathfrak{A}},\\ i \in [r]}} S^{\mathfrak{A}}_{j;T,i} && j \in [q^q], \\
R^{\mathfrak{B}} &:= \left\{ \bar{u} \in B^r_{\mathsf{gr}} \;\middle|\; \bar{v} \in R^{\mathfrak{A}}, (v_i, u_i) \in R^{\mathfrak{B}}_{\mathsf{orb}} \text{ for every } i \in [r] \right\} && \text{if } R^{\mathfrak{A}} \text{ is heterogeneous}, \\
R^{\mathfrak{B}} &:= R^{\mathfrak{A}} && \text{if } R^{\mathfrak{A}} \text{ is homogeneous},
\end{aligned}$$

for every $R \in \tau$. That is, we now relate the orbits in $R^{\mathfrak{B}}$ (see Figure 4.4). Finally, the preorder $\preceq^{\mathfrak{B}}$ is obtained from $\preceq^{\mathfrak{A}}$ by creating new color classes for every $C_{T,i}$ (which can obviously be ordered in CPT).

Claim 1. *The structure $\mathfrak{B}$ is an r-injective quotient structure.*

Proof. For Condition 1, $\mathfrak{B}$ is clean and of heterogeneous arity r. The "old" color classes are still regular and the new are so by Lemma 4.38. Condition 2 is satisfied because the only heterogeneous relation connecting B_{gr} and B_{ex} is by construction the binary relation R_{orb}. Every other heterogeneous relation is of arity r. To show Condition 3, let $C_{T,i} \in \mathbb{C}_{\mathfrak{B}}$ be a group color class for some $C \in \mathbb{C}_{\mathfrak{A}}$, $T \in \mathbb{T}^{\mathfrak{A}}$, and $i \in [3]$. Then, by construction, $C_{T,i}$ is only related to C (seen as extension color class of $\mathfrak{B}$) via R_{orb}.

To finally show Condition 4, let $T' \in \mathbb{T}^{\mathfrak{B}}_{\mathsf{gr}}$. By construction, there is a $T \in \mathbb{T}^{\mathfrak{A}}$ such that $T' = (C_{1,T,1}, \dots, C_{r,T,r})$ and $C_{i,T,i}$ is an N_i^T-quotient of C_i (note that the C_i are color



(a) The input structure

(b) The construction of Lemma 4.40

4.4 The construction in Lemma 4.40. Figure (a) shows the input structure. Each circle represents one color class with drawn tuples of two relations (red and blue) between two orbits of each color class (there can of course be more orbits and tuples of the red and blue relations). Figure (b) shows the altered structure: For the basic constituents of the red and blue relations, there are new group color classes (on the top for red and on the bottom for blue), where the orbits are contracted to a single atom. The "old" color classes became extension color classes.

classes of both $\mathfrak{A}$ and $\mathfrak{B}$). We argue that $\Gamma^{\mathfrak{B}}_{T'} \cong \Gamma^{\mathfrak{A}}_{T}/N^T_1/\dots/N^T_r$ and hence $\Gamma^{\mathfrak{B}}_{T'}$ is an r-injective subdirect product. First note that

$$\Gamma^{\mathfrak{A}}_{T}/N^T_1/\cdots/N^T_r = \Gamma^{\mathfrak{A}}_{T'}/(N^T_1\cdots N^T_r) \text{ and}$$
$$N^T := N^T_1\cdots N^T_r \lhd \Gamma^{\mathfrak{A}}_{T}.$$

Hence, $\Gamma^{\mathfrak{A}}_{T}/N^T$ defines a permutation group on $\mathsf{orb}(N^T)$, which are precisely the orbits of N^T_i on C_i for every $i \in [r]$. That is,

$$\mathsf{orb}(N^T) = \bigcup_{i\in[r]} \mathsf{orb}(N^T_i|_{C_i}).$$

We now study these orbits: Let $R \in \tau$, $\bar{u} \in R^{\mathfrak{A}} \cap \bigotimes_{i\in[r]} C_i$, and $\sigma_i \in N^T_i$ for every $i \in [r]$. Then $(\sigma_1\cdots\sigma_r)(\bar{u}) \in R^{\mathfrak{A}}$ because $N^T_i \leq \Gamma^{\mathfrak{A}}_{T}$ for every $i \in [3]$. It follows that $\bar{v} \in R^{\mathfrak{A}}$ whenever $v_i \in \mathsf{orb}_{N^T_i}(u_i)$ for every $i \in [3]$, that is,

$$\bigotimes_{i\in[r]} \mathsf{orb}_{N^T_i}(u_i) \subseteq R^{\mathfrak{A}}.$$

We define a map $\varphi\colon \Gamma^{\mathfrak{A}}_{T}/N^T \to \Gamma^{\mathfrak{B}}_{T'}$ as follows: Let $\sigma N^T \in \Gamma^{\mathfrak{A}}_{T}/N^T$, which induces a permutation on the N^T-orbits σ_{N^T}. Via R_{orb} (recall that the C_i are also color classes of $\mathfrak{B}$) the permutation σ_{N^T} induces a permutation σ'_{N^T} on the atoms of the $C_{i,T,i}$ for every $i \in [r]$. We have already seen that $\bigotimes_{i\in[r]} \mathsf{orb}_{N^T_i}(u_i) \subseteq R^{\mathfrak{A}}$ if and only if $\bar{v} \in R^{\mathfrak{A}}$. The heterogeneous relations in $\mathfrak{B}$ are defined accordingly, that is, heterogeneous relations are invariant under σ'_{N^T}. To show that $\sigma'_{N^T} \in \Gamma^{\mathfrak{B}}_{T'}$, it remains to consider the homogeneous relations. These are constructed precisely such that $\mathsf{Aut}(\mathcal{C}_{i,T,i}) \cong \mathsf{Aut}(\mathcal{C}_i)/N^T_i$ and are thus invariant under σ'_{N^T}, i.e., $\sigma'_{N^T} \in \Gamma^{\mathfrak{B}}_{T'}$. In particular, the prior reasoning also holds in the other direction, thus φ is an isomorphism, and thus $\Gamma^{\mathfrak{B}}_{T'}$ is an r-injective subdirect product. $\dashv$

We extend the interpretation Θ to a canonization-preserving reduction (Θ, Π): Given a canon of $\mathfrak{B}$, we can distinguish group from extension color classes using R_{orb}. The universe of the canon of $\mathfrak{A}$ are the extension color classes of the canon of $\mathfrak{B}$, homogeneous relations are just copied, and heterogeneous relations $R \in \tau$ are defined on the extension color classes using R_{orb}: A tuple $\bar{u}$ is in R in the canon of $\mathfrak{A}$ if there is a tuple $\bar{v}$ in R in the canon of $\mathfrak{B}$ such that (u_i, v_i) is in R_{orb} for every $i \in [r]$. This is clearly a CPT-definable interpretation.

The reduction obviously preserves the automorphism groups of the color classes. For every extension color class in $\mathfrak{B}$, there is by construction a color class in $\mathfrak{A}$ with the same automorphism group. For every group color class in $\mathfrak{B}$, its automorphism group is by construction the quotient of the automorphism group of some color class of $\mathfrak{A}$.

To the end, assume that $\mathfrak{A}$ is of heterogeneous arity $r' \leq r$. Then, by our assumption in the beginning, the heterogeneous relations were extended using a singleton color class to have arity r. That is, for every basic constituent $T = (C_1, \dots, C_r) \in \mathbb{T}^{\mathfrak{A}}$, the color classes $C_{r'+1}, \dots, C_r$ are distinct and new singleton color class by construction. Because every quotient of a singleton color class is a singleton color class, every group constituent of $\mathfrak{B}$ has the same property. $\square$

We conclude the section on normal forms:

Theorem 4.41. *For all $q \in \mathbb{N}$, $r \leq 2$, and every signature τ, q-bounded τ-structures of heterogeneous arity r can be canonization-preservingly reduced in* CPT *to*

(a) q'-bounded 2-injective quotient τ'-structures for some q' and τ' and to

(b) q'-bounded $(r-1)$-injective quotient τ'-structures for some q' and τ'. The latter reduction preserves the automorphism groups of the color classes.

Proof. We first show Part (b). By Lemma 4.29, we reduce to q-bounded and r-color-class transitive structures of heterogeneous arity r. We then further ensure that the structures are clean using Lemma 4.32 preserving r-color-class-transitivity and heterogeneous arity r. Next, we make the color classes regular by Lemma 4.35, now yielding q'-bounded structures. We still preserve r-color-class-transitivity, heterogeneous arity r, and cleanness. Finally, we reduce to r-injective quotient structures with Lemma 4.40. All reductions preserve the color classes of the automorphism groups and so does their composition, too.

For Part (a), we first need to reduce the arity of the structure to 3 and then apply Part (b). We sketch how a q-bounded structure $\mathfrak{A}$ can be reduced to have arity 3: For simplicity, we first apply Lemma 4.32 to obtain clean relations. We split a relation R of arity $r > 3$ in two relations R_1 and R_2 of arity $r-1$ and 3 as follows: Let $C_1, \dots, C_r \in \mathbb{C}_{\mathfrak{A}}$ be distinct color classes. For every $\bar{u} \in R^{\mathfrak{A}} \cap (C_1 \times \dots \times C_r)$, we add new atoms for the pairs (u_{r-1}, u_r) to the universe and define R_1 such that $(u_1, \dots, u_{r-1}, (u_{r-1}, \bar{u}_r)) \in R_1^{\mathfrak{A}}$ and $((u_{r-1}, \bar{u}_r), u_{r-1}, u_r) \in R_2^{\mathfrak{A}}$. The new atoms form a new color class. Its automorphism group can be ensured to be isomorphic to $\mathsf{Aut}(\mathfrak{A}[C_{r-1} \cup C_r])$ using Lemma 4.28. By iteratively applying this reduction, one can reduce the arity down to 3. $\square$

Of course, one could reduce the arity of the structures to 2 by encoding every tuple in a relation by a new atom adjacent to the atoms in the tuple. Obviously, the automorphism group of these new atoms must be the automorphism group of the tuples in the relations, so we have indeed encoded the structure in a graph, but are still faced with exactly the same groups. The new color classes obtained by our proposed construction can be much simpler, depending on the actual group. In the following, we will consider structures of arity at most 3.

4.4 Structures with Dihedral Colors

We now consider structures whose color classes have dihedral automorphism groups.

Definition 4.42 (Dihedral Colors). For a q-bounded structure $\mathfrak{A}$, a color class $C \in \mathbb{C}_{\mathfrak{A}}$ is called **dihedral**, **abelian**, or **cyclic** if $\mathsf{Aut}(\mathcal{C})$ is a dihedral, abelian, or cyclic group, respectively. The structure $\mathfrak{A}$ **has dihedral colors** or **is dihedral** if every $C \in \mathbb{C}_{\mathfrak{A}}$ is dihedral or cyclic.

Here, we include cyclic automorphisms group in the definition of structures with dihedral colors. We do so because we want that the class of groups admissible for the color classes is closed under taking subgroups and under taking quotient groups. So the class is preserved by the reduction in Theorem 4.41. We now show that the automorphism group of a regular and dihedral color class can be made explicit in the following sense:

Definition 4.43 (Color Class in Standard Form). Let $\mathfrak{A}$ be a q-bounded structure with dihedral colors. We say that **a color class $C \in \mathbb{C}_{\mathfrak{A}}$ is in standard form** if there are two relations $R^{\mathfrak{A}}$ and $S^{\mathfrak{A}}$ of arity 2 such that following holds:

- If $\mathsf{Aut}(\mathcal{C}) \cong \mathsf{C}_{|C|}$, then each of $R^{\mathfrak{A}}[C]$ and $S^{\mathfrak{A}}[C]$ forms a directed cycle of length $|C|$ on C.
- Otherwise, $\mathsf{Aut}(\mathcal{C}) \cong \mathsf{D}_{|C|/2}$, $R^{\mathfrak{A}}[C]$ defines two directed and disjoint cycles of length $|C|/2$, and $S^{\mathfrak{A}}[C]$ connects the cycles by a perfect matching such that the two cycles are directed into opposite directions (cf. Figure 4.5).[1]

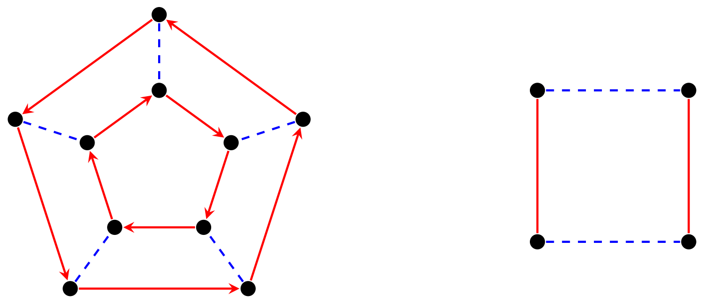
We say that the relations $R^{\mathfrak{A}}$ and $S^{\mathfrak{A}}$ **induce the standard form of $\mathcal{C}$**. The **color classes of $\mathfrak{A}$ are in standard form** if every color class is in standard form.

For cyclic groups, we require two relations only for technical reasons: We can treat the cyclic and dihedral case uniformly. In that case, one can always pick $R^{\mathfrak{A}} = S^{\mathfrak{A}}$ as we see in the following. For dihedral groups, we indeed need two relations for the group D_2 because for D_2 the directed cycles are just two undirected edges and they cannot not be distinguished from the perfect matching (cf. Figure 4.5).

Corollary 4.44. *Let $\mathfrak{A}$ be a q-bounded structure with dihedral colors. If $C \in \mathbb{C}_{\mathfrak{A}}$ is in standard form induced by the relations $R^{\mathfrak{A}}$ and $S^{\mathfrak{A}}$, then* $\mathsf{Aut}(\mathcal{C}) = \mathsf{Aut}((C, R^{\mathfrak{A}}[C], S^{\mathfrak{A}}[C]))$.

We show that every regular and dihedral color class can be converted to standard form in CPT.

[1]The two relations form the Cayley graph generated by a reflection and a rotation of maximum order, but we will not need this fact.



4.5 Dihedral color classes in standard form. The dihedral group D_5 on 10 atoms and the D_2 on 4 atoms in standard form. The two relations are drawn in different line styles.

Lemma 4.45. *For every $q \in \mathbb{N}$ and every signature τ, there are two* CPT*-formulas that given a q-bounded τ-structure $\mathfrak{A}$ and a regular and dihedral color class $C \in \mathbb{C}_{\mathfrak{A}}$ define homogeneous relations inducing the standard form of C.*

Proof. Let $q \in \mathbb{N}$, τ be a signature, $\mathfrak{A}$ be a q-bounded τ-structure, $C \in \mathbb{C}_{\mathfrak{A}}$ be a regular and dihedral color class, $\Gamma = \mathsf{Aut}(\mathcal{C})$, and $n = |C|$. We consider the 2-orbits of Γ. There are only two cases because C is regular:

1. Assume that $\Gamma \cong \mathsf{C}_n$. Let r be a rotation of order n. Let O be a 2-orbit such that $(u, r(u)) \in O$ for some $u \in C$. By iterating r, $(r^k(u), r^{k+1}(u)) \in O$ for every k and thus O contains a directed cycle of length n because Γ is transitive. Because C_n is of order n, every 2-orbit has size at most n. Hence, $|O| = n$ and O forms a directed cycle. To define such an orbit in CPT, we just pick the smallest orbit according to the defined order that has the mentioned properties (Lemma 4.27). We use O for both relations inducing the standard form.

2. Assume $\Gamma \cong \mathsf{D}_{n/2}$. By the reasoning for cyclic groups above, there is a 2-orbit O_1 containing a directed cycle $O' \subseteq O_1$ of length $n/2$. Because C is regular, all rotations $r \in \Gamma$ map an atom contained in O' to an atom contained in O' and all reflections map atoms in O' to atoms in $O_1 \setminus O'$. Hence, O_1 contains two disjoint directed cycles of length $n/2$. Because $|\Gamma| = n$, the orbit O_1 is the union of these two cycles.

   Let u and v be atoms not in the same cycle. Then (by the argument above), there is a reflection $\alpha \in \Gamma$ with $\alpha(u) = v$ and $\alpha(v) = u$. We pick the orbit O_2 with $(u, v), (u, v) \in O_2$. Let $r \in \Gamma$ be a rotation of order $n/2$. Then $(r^k(u), r^k(v)), (r^k(v), r^k(u)) \in O_2$ for every k. So $|O_2| \geq n$, that is, O_2 exactly contains these elements and thus is a perfect matching between the two directed cycles in O_1.

   In CPT, we first define the minimal orbit O_1 satisfying the required conditions, and then the minimal orbit O_2 satisfying the conditions with respect to O_1. The two orbits induce the standard form of C. □

By iteratively applying the previous lemma, we can assume that up to a CPT-definable interpretation the color classes of a structure with dihedral colors are in standard form. We just define two relations, each the union of the first (respectively, second) relation defined in the previous lemma for each color class. This reduction is clearly canonization-preserving because we just have to remove the added relations.

4.5 Treelike Cyclic Linear Equation Systems

Before we begin to canonize structures with dihedral colors, we need to discuss a special class of linear equation systems. These linear equation systems are later used to encode the canonical labelings, which is important for our canonization approach. We start with the definition of cyclic linear equations systems from [118].

Definition 4.46 (Cyclic Linear Equation System). Let V be a set of variables. A **cyclic constraint** on $W \subseteq V$ is a solvable set of linear equations containing, for each pair of variables $u, v \in W$, an equation of the form $u - v = d$ for some $d \in \mathbb{Z}_q$. A **cyclic linear equation system (CES)** over $\mathbb{Z}_q$ (for q a prime power) is a triple $(V, S, \preceq)$ where $\preceq \subseteq V^2$ is a total preorder and S is a linear equation system over V that contains a cyclic constraint on each $\preceq$-equivalence class.

As for structures, the total preorder $\preceq$ induces a total order on the $\preceq$-equivalence classes, which we call **variable classes**. For our use in CPT, the linear equation system S is represented by a set of constraints. A constraint $\sum_{u \in W} a_u u = d$ for $W \subseteq V$ itself is encoded by the tuple $(\{(a_u, u) \mid u \in W\}, d)$ (tuples themselves are Kuratowski-encoded).

Theorem 4.47 ([118]). *Solvability of CESs over the ring $\mathbb{Z}_q$ is* CPT*-definable for every prime power q.*

We relax the requirement on the order on the variables being total:

Definition 4.48 (Tree-Like Cyclic Linear Equation System). A **tree-like cyclic linear equation system (TCES)** over $\mathbb{Z}_q$ for a prime power q is a tuple $(V, S, \preceq)$ with the following properties:

- The variable classes form a rooted tree with respect to being a direct successor in $\preceq$. (That is, $\preceq$ is a preorder such that for all $u, v, w \in V$ with $u \prec v$, $u \prec w$, and v and w incomparable ($v \not\preceq w$ and $w \not\preceq v$), there is no $u' \in V$ with $v \preceq u'$ and $w \preceq u'$.)

- S is a linear equation system on V containing for every variable class a cyclic constraint.

- For every constraint $\sum_{u \in W} a_u u = d$ with $W \subseteq V$ and $a_i, d \in \mathbb{Z}_q$ in S, every pair of variables $u, v \in W$ is $\preceq$-comparable. (That is, a constraint can only use the variables from the classes on a root-to-leaf path of the tree.)

Note that TCESs are a strict generalization of CESs. We show that solvability of a certain subclass of TCESs can be defined in CPT. In principle, we follow the same strategy of [103, 118] to solve CESs. Thus, before we turn to solving TCESs, we sketch the method to solve CESs. As a first step, we preprocess a (T)CES, such that every variable class contains exactly q many variables [103, Lemma 5.3] and that $u - v = d \neq 0$ for all equations with $u \neq v$ in the cyclic constraints. This step is necessary to define hyperterms.

4.5.1 Hyperterms

Let $\mathcal{S} = (V, S, \preceq)$ be a CES over $\mathbb{Z}_q$ with variable classes $C_1 \prec \cdots \prec C_n$. For all variables $u, v \in C_i$ in the same variable class, there is a constraint $u - v = d$ for some $d \in \mathbb{Z}_q$. Hence, we can pick any variable $u \in C_i$ and substitute all other variables of the class C_i with a term of the form $u - d$. Of course, we cannot do this in CPT because we cannot choose u canonically. Assume for the moment that we picked for every variable class one variable and substituted all other variables in that fashion. Then we obtain a linear system of equations $\mathcal{S}'$ with a total order on the variables, which can be extended to a total order on the equations. So we can write the system in matrix form $M\bar{u} + \bar{b} = \bar{d}$, where M is the (ordered) $m \times n$ coefficient matrix and $\bar{b}, \bar{d} \in \mathbb{Z}_q^m$. Here $\bar{d}$ is the original right-hand side of the equation system S and $\bar{b}$ collects the constants that arose from substituting the variables. Now, consider another choice when picking the variables, yielding another system $\mathcal{S}''$. Then we can write $\mathcal{S}''$ as $M\bar{u} + \bar{b}' = \bar{d}$ with the same coefficient matrix M and the same right-hand side $\bar{d}$. The only change is the tuple $\bar{b}'$, the difference coming from the different constants arising when substituting two variables of the same class. **Hyperterms** are hereditarily finite sets built from variables and constants. They constitute a succinct, simultaneous encoding of all possible ways to pick the variables and the resulting tuple $\bar{b}'$. The encoding is performed in a way that allows us to use hyperterms to mimic algebraic operations involving linear terms over the variables in CPT. We summarize properties and capabilities of hyperterms, without explicitly describing how hyperterms are constructed and without providing proofs. For details, we refer to [103, 118].

Clearly, only assignments $V \to \mathbb{Z}_q$ satisfying the cyclic constraints of a (T)CES are candidates for solutions when checking whether a (T)CES is solvable. We call these assignments **reasonable assignments**, write $\mathbb{L}_{\mathcal{S}}$ for the reasonable assignments of $\mathcal{S}$, and leave out the subscript if $\mathcal{S}$ is clear from the context. Let T, T_1, and T_2 be hyperterms.

1. There is a CPT-term that, for each variable class C_i, defines the coefficient $\mathsf{c}_i(T)$ of C_i in T. Intuitively, $\mathsf{c}_i(T)$ is the coefficient of some (and thus every) variable $u \in C_i$ in the hyperterm T after all other variables of C_i have been substituted by u in the way explained before.

2. Under an assignment $\rho \in \mathbb{L}$, hyperterms can be evaluated to $T[\rho] \in \mathbb{Z}_q$. Given ρ, this can be done in CPT (but we usually do not have access to a single assignment ρ).

3. For every choice of a variable $u_i \in C_i$ for each $i \in [n]$, there is an equivalent linear term $t = \sum_{i \in [n]} a_i u_i + b$ with $a_i = \mathsf{c}_i(T)$ for T, that is $T[\rho] = t[\rho] := \sum_{i \in [n]} a_i \rho(u_i) + b$ for all assignments ρ.

4. Every constant $b \in \mathbb{Z}_q$ and every variable $u \in V$ is a hyperterm.

5. There are CPT-terms realizing addition and scalar multiplication on hyperterms that behave as expected with respect to evaluation and coefficients. This means, $T_1[\rho] + T_2[\rho] = (T_1 + T_2)[\rho]$ and $b \cdot T[\rho] = (b \cdot T)[\rho]$ for all $i \in [n]$, $b \in \mathbb{Z}_q$, and $\rho \in \mathbb{L}$. In particular, the coefficients of the variable classes satisfy $\mathsf{c}_i(T_1 + T_2) = \mathsf{c}_i(T_1) + \mathsf{c}_i(T_2)$ and $\mathsf{c}_i(b \cdot T) = b \cdot \mathsf{c}_i(T)$ for all $i \in [n]$ and $b \in \mathbb{Z}_q$. Addition on hyperterms is *not* associative (because of the involved set constructions) and we stipulate evaluation to be from left to right.

6. The hyperterm T is called **constant** if $T[\rho] = d$ for some constant $d \in \mathbb{Z}_q$ and all reasonable assignments $\rho \in \mathbb{L}$. We call d the **value** of T. There is a CPT-term that defines the value of constant hyperterms. Necessarily, $\mathsf{c}_i(T) = 0$ for all $i \in [n]$ if T is constant.

7. There is a CPT-term that extends the preorder $\preceq$ to all linear equations $t = d \in S$. Furthermore, there is a CPT-term that translates a linear term t to an equivalent hyperterm T_t (and hence a linear equation $t = d \in S$ into an equivalent hyperequation $T_t = d$) such that $\preceq$-equivalent linear equations are translated into the same hyperequation, in particular, the linear equations have the same solutions. In this way, we obtain an equivalent system of hyperequations $\mathrm{Hyp}(\mathcal{S})$ with a total order on the hyperequations.

8. The operations preserve set-theoretical membership. If the transitive closure of $T_1 + T_2$ (or of $d \cdot T$, respectively) contains a variable u, then the transitive closure of T_1 or T_2 (or of T, respectively) contains u.

Only Properties 6 and 7 depend on the totality of the preorder $\preceq$. Because $\mathrm{Hyp}(\mathcal{S})$ is equivalent to $\mathcal{S}$ (i.e., they have the same satisfying assignments) we can focus on checking ordered systems of hyperequations for solvability.

In general, a **system of hyperequations $\hat{\mathcal{S}}$ (for $\mathcal{S}$)** is an ordered set of hyperequations, where the hyperterms are constructed as above (so with respect to V and the cyclic constraints contained in $\mathcal{S}$). In particular, $\mathrm{Hyp}(\mathcal{S})$ is such a system. Because systems of hyperequations are ordered (both the variable classes and the hyperequation are), we can consider the ordered $m \times n$ coefficient matrix $M_{\hat{\mathcal{S}}}$ of $\hat{\mathcal{S}}$ defined via $M_{\hat{\mathcal{S}}}(j, i) := \mathsf{c}_i(T_j)$ for all $j \in [m]$ and $i \in [n]$, where m is the number of hyperequations in $\hat{\mathcal{S}}$ and $T_j = d_j$ is the j-th hyperequation for every $j \in [m]$.

4.5.2 Gaussian Elimination for Rings $\mathbb{Z}_q$

We want to apply a variant of Gaussian elimination adapted for the ring $\mathbb{Z}_q$ with q a prime power. Recall that Gaussian elimination for fields uses elementary row operations to convert the coefficient matrix to an upper triangular matrix (note that this is only a reasonable notion for matrices with a total order on both row and column indices). Then solvability can be tested by checking whether all **atomic** equations, i.e., equations with all coefficients 0, are solvable. From a CPT-perspective, it is important to have a total order on the variable classes to pick the next class whose coefficient should be eliminated. We also need the total order on the constraints to pick the unique constraint, in which the coefficient is not to be eliminated. We now sketch how the solvability check needs to be adapted for rings $\mathbb{Z}_q$ and, in particular, which additional constraints must be satisfied by the coefficient matrix such that we can check solvability similarly to fields. For simplicity, consider an ordinary and ordered linear system of equations in matrix form $M\bar{u} + \bar{b} = \bar{d}$

with upper triangular coefficient matrix

$$M = \begin{pmatrix} a_{1,1} & & & & \cdots & a_{1,n} \\ 0 & a_{2,2} & & & \cdots & a_{2,n} \\ \vdots & \ddots & \ddots & & & \vdots \\ 0 & \cdots & 0 & a_{k,k} & \cdots & a_{k,n} \\ 0 & \cdots & & 0 & \cdots & 0 \\ \vdots & & & & \ddots & \vdots \\ 0 & \cdots & & & & 0 \end{pmatrix}$$

such that $a_{i,i} \neq 0$ for $i \leq k$ and variables $\bar{u} = (u_1, \dots, u_n)$. Assume we found an assignment ρ for the variables $u_{j+1}, \dots, u_n$ ($j+1 \leq k$) satisfying all equations that use only these variables. We consider the j-th equation $\sum_{j \leq i \leq n} a_{j,i} u_i + b_j = d_j$. We solve for the variable u_j and obtain $a_{j,j} u_j = d_j - b_j - \sum_{j < i \leq n} a_{j,i} \rho(u_i)$. In a field, we can divide both sides by $a_{j,j}$, but in the ring $\mathbb{Z}_q$ this is not possible in general. To overcome this problem, [103] makes use of the fact that divisibility in $\mathbb{Z}_q$ for prime powers q is a total preorder and rearranges the order of the variables as follows: The variable u is considered smaller than v if there is some coefficient of u in some equation that divides every coefficient of v. If that is the case for both u and v, they are ordered according to the given order on the variables. Then the variables are eliminated according to this rearranged order. When eliminating variable u, the equation containing this minimal coefficient (with respect to divisibility) is picked to remain. In all other equations u is eliminated (if there are multiple such equations, we pick one using the total order on the equations). Consequently, it holds that $a_{j,j} \mid a_{j,i}$ for all $i \in [n]$ and that $a_{j,j} \mid a_{j+1,j+1}$ for all $j \in [k]$. Hence, once we ensure that $a_{j,j}$ also divides $(d_j - b_j)$, we can divide by $a_{j,j}$. Coefficient matrices satisfying this property are said to be in Hermite normal form:

Definition 4.49 (Hermite Normal Form, [103]). A system of hyperequations $\hat{\mathcal{S}}$ is in **Hermite normal form** if, for some total ordering of its variable classes $C_1 \prec \dots \prec C_n$ and some order of its equations, its $m \times n$ coefficient matrix $M_{\hat{\mathcal{S}}}$ satisfies the following conditions:

1. $M_{\hat{\mathcal{S}}} = (a_{i,j})_{i \in [m], j \in [n]}$ is upper triangular.
2. $a_{j,j} \mid a_{i,i}$ for all $j < i \leq k$, where $k \in [n]$ is maximal such that $a_{k,k} \neq 0$.
3. $a_{j,j} \mid a_{j,i}$ for all $j \in [k]$ and $i \in [n]$.

To check a system of hyperequations for solvability, it is not sufficient to check the atomic equations. Recall from the case of linear equations above, that we have to ensure that $a_{j,j} \mid (d_j - b_j)$ and that for a hyperequation $T_j = d_j$, there is always an equivalent linear equation $\sum_{i \in [n]} a_{j,i} u_i + b_j = d_j$. Also, recall that we do not have access to b_j directly because it depends on the choice of the u_i. It turns out that a system of hyperequation $\hat{\mathcal{S}}$ over $\mathbb{Z}_q$ in Hermite normal form is solvable if and only if the following two conditions hold [103, Lemma 5.21]:

1. Every atomic hyperequation in $\hat{\mathcal{S}}$ is solvable.
2. For every non-atomic hyperequation $T = d$ in $\hat{\mathcal{S}}$, it holds that $p^\ell T = p^\ell d$ is solvable, where q is a power of p and p^ℓ be the smallest power of p annihilating all coefficients of T, i.e., $p^\ell T$ is constant.

From the second condition we deduce for non-atomic hyperequations that $0 = p^\ell(d_j - b_j)$ and that $a_{j,j} \mid (d_j - b_j)$. We show this in more detail later in our extension of the algorithm for TCESs. It is clear that the former conditions can be checked in CPT because the value of constant hyperterms is CPT-definable.

4.5.3 Solving TCESs in CPT

Let $\mathcal{T} = (V, S, \preceq)$ be a TCES over $\mathbb{Z}_q$. We now write $\mathbb{C}_\mathcal{T}$ for the set of all variable classes and $\mathsf{c}_C(T)$ for the coefficient of the variable class C in a hyperterm T. We call the directed tree on the variable classes induced by $\preceq$ the **variable tree**, where there is an edge (C, C') if and only if $C \prec C'$ and there is no other variable class with $C \prec C'' \prec C'$.

We first discuss how a TCES $\mathcal{T}$ is translated into a system of hyperequations: Because every equation in $\mathcal{T}$ only uses variables on a root-to-leaf path in the variable tree, all relevant variable classes for a single linear equation in $\mathcal{T}$ are *totally* ordered. So we can apply the translation for CESs to the equation. We do this for all equations of $\mathcal{T}$. The obtained hyperequations form a system of hyperequations $\mathrm{Hyp}(\mathcal{T})$ equivalent to $\mathcal{T}$. However, the variable classes of $\mathrm{Hyp}(\mathcal{T})$ are not totally ordered as in the case for CES.

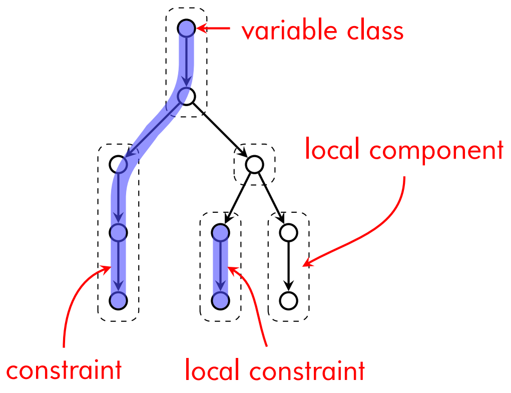
This complicates matters when checking such a system of hyperequations for solvability: We have seen in the previous section, that it is essential to reorder the variable classes to successfully apply Gaussian elimination to rings $\mathbb{Z}_q$ for prime powers q. In the case of CESs, this was possible because reordering a linear order yields a linear order again. However, if we reorder the preorder inducing the variable tree according to divisibility of the coefficients of the variables, then the result is not necessarily a tree anymore. In other words, reordering variables with regard to divisibility of the coefficients may not be compatible with the tree-like structure of the equation system.

Overall, we are unsure how to check solvability of a TCES in CPT. Hence, in the following, we will only look at a certain restricted class of TCESs, where divisibility of the coefficients of "critical" variable classes is not important. As we argue later, this suffices for our overall goal of canonizing structures with dihedral colors. Before defining this class of TCESs, we need to introduce some terminology for the structure of the variable tree.

Definition 4.50 (Local Component and Global/Local Variables). Let $\mathcal{T} = (V, S, \preceq)$ be a TCES. A set $L \subseteq \mathbb{C}_\mathcal{T}$ of variable classes is called a **local component** if it satisfies the following (cf. Figure 4.6):

1. The induced subgraph of L in the variable tree is a path.
2. For every $C \in L$, if C is of out-degree at least 2 in the variable tree, then the children of C are not in L.
3. L is maximal with respect to set inclusion.

A variable **occurs** in an equation if its coefficient in the equation is nonzero. A variable is **local** if in every equation in which it occurs, only variables of the same local component occur, too. Other variables are called **global**. An equation is **local** if it contains at least one local variable and **global** otherwise.



4.6 The variable tree of a TCES. The variable tree (a vertex represents a variable class) of an example TCES with all local components. A constraint is depicted by a blue line and only uses variables from vertex classes covered by the line. Constraints only use variables from root-to-leaf paths and local constraints are contained within a local component.

On the local components, the preorde $\preceq$ induces a tree in which every local component has degree at least 2 or is a leaf. Note that local equations can only use variables of the same local component. Also, note that a global equation may also do so namely if all variables occurring in the equation are global. Furthermore, in a local equation the coefficient of global variables of the same local component can be nonzero.

We extend the definition to variable classes: A variable class is called global if it contains some global variable. Likewise, the other variable classes are called local. Note that a global variable class can contain both, global and local variables, and that we cannot simply split the class because we need to maintain that every class contains q many variables. However, when we are working with hyperterms, it will not be important whether there is a local variable in a global variable class, but only that the class occurs in global equations.

Definition 4.51 (Weakly Global TCES). A TCES $\mathcal{T} = (V, S, \preceq)$ over $\mathbb{Z}_q$ is called **weakly global** if

- q is a power of an odd prime and every equation (equivalently every variable) is local or
- $q = 2^\ell$ is a power of 2 and, for every global variable $u \in V$, the equation $2u = 0$ is contained in S.

Note that every CES is a weakly global TCES because a CES only has one local component. We first note that weakly global TCESs for odd prime powers are easy:

Corollary 4.52. *Solvability of weakly global TCESs over $\mathbb{Z}_q$ for every odd prime power q is* CPT*-definable.*

Proof. A weakly global TCESs for odd prime powers consists only of local equations, so is just a disjoint union of CESs, one for each local component. Hence, we can solve each CES by the methods of [103, 118] in parallel. □

In the case that $q = 2^\ell$ is a power of 2, the constraints connecting the CESs induced by the local components are formed by variables that can be either assigned to 0 or $2^{\ell-1}$ in a satisfying assignment. Hence, these variables are actually over $\mathbb{Z}_2$, but embedded in $\mathbb{Z}_{2^\ell}$. In $\mathbb{Z}_2$, divisibility of coefficients does not become an issue because $\mathbb{Z}_2 = \mathbb{F}_2$ is a field. We now consider systems of hyperequations and adapt the notions of global and local variables and equations to these systems.

Definition 4.53 (Tree-Like System of Hyperequations). A **tree-like system of hyperequation** $\hat{\mathcal{T}}$ for a TCES $\mathcal{T}$ over $\mathbb{Z}_q$ is a set of hyperequations built from hyperterms constructed for $\mathcal{T}$ (so using the variable classes and cyclic constraints of $\mathcal{T}$) satisfying the following:

1. The preorder $\preceq$ of $\mathcal{T}$ induces a tree on the variable classes and, for every hyperequation $T = d$, the transitive closure of T (set-theoretically) only contains variables which are $\preceq$-comparable[2].
2. There is a preorder on the hyperequations such that, for every root-to-leaf path in the variable tree, the preorder is a total order on the subset $S' \subseteq S$ of hyperequations in which only variable classes contained in said path have nonzero coefficients.

A hyperequation $T = d$ in $\hat{\mathcal{T}}$ is **global** if it (set-theoretically) contains variables from different local components. Other hyperequations are **local**. A variable class is global if (some of) its variables are contained in a global hyperequation and local otherwise. Finally, $\hat{\mathcal{T}}$ is called **weakly global** if q is odd or, for every global variable class C, the system $\mathcal{T}$ contains a hyperequation $T = 0$ such that C is the only color class with nonzero coefficient in T and $\mathsf{c}_C(T) = 2$.

When converting a weakly global TCES to a system of hyperequations, the resulting system is tree-like and weakly global.

Lemma 4.54. *For every weakly global TCES $\mathcal{T}$, the system of hyperequations Hyp($\mathcal{T}$) is tree-like and weakly global.*

Proof. Let $\mathcal{T}$ be a weakly global TCES. Clearly, Hyp($\mathcal{T}$) is tree-like because a hyperequation $T = d$ obtained from a linear equation (set-theoretically) only contains variables of variable classes used by the linear equation. When converting an equation $2u = 0$ for $u \in C$ to a hyperequation $T = 0$, then $\mathsf{c}_C(T) = 2$ and the coefficient for every other color class is 0 (otherwise T would not be equivalent to $2u = 0$). Even more, T set-theoretically only contains variables from C. Because $\mathcal{T}$ contains for every global variable u the equation $2u = 0$ and because a variable class C in Hyp($\mathcal{T}$) is global if and only if C is global in $\mathcal{T}$, the system Hyp($\mathcal{T}$) is weakly global. □

4.5.4 Systems of Hyperequations over $\mathbb{Z}_{2^\ell}$

As explained before, we can solve weakly global TCESs over $\mathbb{Z}_q$ if q is an odd prime power. So fix $\ell \in \mathbb{N}$ and consider the power 2^ℓ of 2. We first make the following simple observation:

[2]This is a stronger condition than requiring that all variable classes with nonzero coefficient are $\preceq$-comparable, because some hyperterm operations may set the coefficient of a variable class to zero, but the hyperterm as a set only grows, so its transitive closure still contains the variables of that class.

Lemma 4.55. *Let $\mathcal{T}$ be a weakly global TCES. If $\mathcal{T}$ is solvable, then the following conditions hold for all global variable classes C:*

(a) There are at most two global variables $u, v \in C$.

(b) Let $u, v \in C$ be global variables with $u \neq v$. Then the equation $u - v = 2^{\ell-1}$ is contained in the cyclic constraint for C.

Proof. To show Part (a), assume there are 3 global variables $u_1, u_2, u_3 \in C \in \mathbb{C}_{\mathcal{T}}$ and hence there are equations $2u_i = 0$ for all $i \in [3]$, so $\rho(u_i) \in \{0, 2^{\ell-1}\}$ for all satisfying assignments ρ and $i \in [3]$. But by the cyclic constraints, we have that $\rho(u_i) \neq \rho(u_j)$ for all $i \neq j$ (recall that we ensured that for cyclic constraints $u_i - u_j = d$ it always holds that $d \neq 0$). To show Part (b), assume that the equation $u - v = d$ for $d \neq 2^{\ell-1}$ is contained in the cyclic constraints (again, we ensured $d \neq 0$). Then all reasonable assignments necessarily violate $\rho(u), \rho(v) \in \{0, 2^{\ell-1}\}$ and $\mathcal{T}$ cannot be solvable. $\square$

The prior lemma expresses formally that we embed $\mathbb{Z}_2$ into $\mathbb{Z}_{2^\ell}$. We surely can check in CPT whether a TCES satisfies the two properties granted by the lemma for solvable TCES. So we in the following always assume that all TCESs satisfy them. We adapt the required normal form for the coefficient matrix:

Definition 4.56 (Local Hermite Normal Form). Let $\hat{\mathcal{T}}$ be a tree-like system of hyperequations over $\mathbb{Z}_{2^\ell}$. We say that $\hat{\mathcal{T}}$ is in **local Hermite normal form** if there are (not necessarily CPT-definable) total orders on the variable classes $C_1 \prec \cdots \prec C_n$ and on the hyperequations in S such that the induced and ordered coefficient matrix $M_{\hat{\mathcal{T}}}$ of $\hat{\mathcal{T}}$ satisfies the following:

The coefficient matrix $M_{\hat{\mathcal{T}}}$ is upper triangular (whereby the coefficients of global variables are taken modulo 2) and, for every local component L, the submatrix M_L of all local equations of L has the following properties:

1. All global variable classes of L are at the end of the linear order.
2. M_L has the following shape:

$$M_L = \begin{pmatrix} M_L^{\text{local}} & M_L^{\text{global}} \end{pmatrix}$$

   where M_L^{local} is the submatrix of all columns of the local variable classes.

3. M_L^{local} is in Hermite normal form.

In that case we also say that L is in local Hermite normal form.

Note that M_L^{local} does not contain zero rows because the equation encoded by a zero row would be either global or atomic – both not local equations of L. We are only interested in the coefficients for global variable classes modulo 2 because the values of these variables are 0 or of order 2 and hence only the parity of coefficients is important. Note that we cannot simply replace the coefficients by the parity because there is no such operation on hyperterms.

Before we address solvability of weakly global TCESs in local Hermite normal form, we revisit the problem of defining the value of a hyperterm. Consider some TCES $\mathcal{T}$.

Now, we are not interested anymore in constant hyperterms T of $\mathcal{T}$, which are those for which $T[\rho]$ is constant for all reasonable assignments $\rho \in \mathbb{L}_{\mathcal{T}}$. Rather, we will consider an even more restricted set of assignments. In the case of global variables, only assignments are of interest that assign global variables values of order at most 2 because otherwise the constraints $2u = 0$ are violated. We refine our definition of reasonable assignments for weakly global TCESs.

Definition 4.57 (Reasonable Assignments of a Weakly Global TCES). For a weakly global TCES $\mathcal{T} = (V, S, \preceq)$ over $\mathbb{Z}^{2^\ell}$, an assignment $V \to \mathbb{Z}_{2^\ell}$ is called **reasonable** if it satisfies all cyclic constraints and all constraints $2u = 0$ for all global variables u. The set of the reasonable assignments of $\mathcal{T}$ is denoted by $\mathbb{L}_{\mathcal{T}}$.

Now, we can keep our definitions of constant hyperterms. Regarding the coefficient of constant hyperterms, we now obtain the following characterization:

Lemma 4.58. *Let $\hat{\mathcal{T}}$ be a weakly global system of hyperterms over $\mathbb{Z}^{2^\ell}$ and T be a hyperterm in $\hat{\mathcal{T}}$. The hyperterm T is constant if and only if $\mathsf{c}_C(T) = 0$ for all local variable classes C and $\mathsf{c}_C(T) \equiv 0 \pmod 2$ for all global variable classes C.*

Proof. The condition on the local variable classes is the same as for CES (recall that, restricted on a local component, a TCES is just a CES). So we can assume that $\mathsf{c}_C(T) = 0$ for all local variable classes C. Consider a linear term $t = \sum_{C \in \mathbb{C}_V} a_C u_C + b$ that is equivalent to T. For the sake of contradiction, suppose $a_C = \mathsf{c}_C(T) \equiv 1 \pmod 2$ for some global color class C. Let $\rho \in \mathbb{L}_{\mathcal{T}}$ be a reasonable assignment (which always exists due to the properties granted by Lemma 4.55 hold). Let ρ' be another assignment that assigns the same values as ρ apart from $\rho'(u) = \rho(u) + 2^{\ell-1}$ for all variables $u \in C$. Then the assignment ρ' is still reasonable, but evaluates differently for t: $t[\rho] = t[\rho'] + 2^{\ell-1}$. Hence, T was not constant.

For the other direction, assume that $\mathsf{c}_C(T) \equiv 0 \pmod 2$ for all global color classes C. Let $\rho \in \mathbb{L}_{\mathcal{T}}$ be a reasonable assignment. Then $t[\rho] = \sum_{C \in \mathbb{C}_V} a_C \rho(u_C) + b = b$ because the coefficients of all local variable classes are 0 and $\mathsf{c}_C(T)$ annihilates values of order at most 2 for all global color classes. □

Lemma 4.59. *There is a* CPT*-term that, for every weakly global system of hyperequations $\hat{\mathcal{T}}$ over $\mathbb{Z}_{2^\ell}$ and every constant hyperterm T of $\hat{\mathcal{T}}$ whose transitive closure (set-theoretically) contains only variables from a single root-to-leaf path in the variable tree, defines the value of T.*

Proof. Recall that a hyperterm is a nested set of variables and constants, in particular it can set-theoretically contain (at some level) a variable, even if its variable class has coefficient 0. A TCES restricted to one root-to-leaf path is a CES. We apply the same procedure to define the value of T as in [103, Lemma 5.11], but we have to argue that the procedure treats global variable classes correctly (which only have coefficient 0 modulo 2).

The procedure follows inductively the order of the CES. Let C be the smallest variable class (with respect to the order) of which a variable is set-theoretically contained in T. We consider all possible reasonable assignments of all variables in these class. For each such assignment, we syntactically replace all occurrences of the variables according to the assignments in T. Because T was constant, the result is the same for all assignments: By

Lemma 4.58, the values for global variables are always annihilated because the assignment only uses values of order at most 2 and the coefficient is 0 modulo 2. We then proceed with the next class. In the end, we are left with a variable free hyperterm whose value can be defined easily as shown in [103, Lemma 5.11]. □

We are ready to characterize solvable systems of tree-like hyperequations in local Hermite normal form:

Lemma 4.60. *A weakly global tree-like system of hyperequations $\hat{\mathcal{T}}$ over $\mathbb{Z}_{2^\ell}$ in local Hermite normal form is solvable if and only if the following holds:*

(a) Every atomic equation is solvable.

(b) For every non-atomic equation $T = d$ in $\hat{\mathcal{T}}$, the following holds: Let 2^k be the minimal power of 2 that annihilates the coefficients of every local variable class in T. If all local variables classes have coefficient zero, we set $k = 1$. Then $2^k T$ is constant and $2^k T = 2^k d$ is solvable.

Proof. If $\hat{\mathcal{T}}$ is solvable, then surely Condition (a) is satisfied. Condition (b) follows from Lemma 4.58 because $k \geq 1$ and $\hat{\mathcal{T}}$ is weakly global and solvable.

For the other direction, assume that Conditions (a) and (b) hold. Recall that changing the order of the variable classes or equations does not affect solvability. We order the variable classes $C_1 \prec \cdots \prec C_n$ as given by local Hermite normality and pick, for every variable class C_i, one variable $u_i \in C_i$. Let $M_{\hat{\mathcal{T}}}\bar{u} + \bar{b} = \bar{d}$ be the equivalent linear system, where $M_{\hat{\mathcal{T}}} = (a_{i,j})_{i\in[m],j\in[n]}$ is the $m \times n$ coefficient matrix as granted by local Hermite normality, $\bar{d}$ contains the same values as the right-hand side of $\hat{\mathcal{T}}$, and $\bar{u} = (u_1, \dots, u_n)$. Such a system exists by the properties of hyperterms. Note that we do not want to construct a satisfying assignment in CPT but only show that it exists and hence we can order the variable classes and pick the variables.

Let $\rho \in \mathbb{L}_{\mathcal{T}}$ be a reasonable assignment such that $\sum_{i\in[n]} a_{k,i}\rho(u_k) + b_k = d_k$ for all $j < k \leq n$ but not for $k = j$ (that is, all rows not among the first j rows in $M_{\hat{\mathcal{T}}}$ are satisfied) and j is minimal. If $j = 0$, then $\hat{\mathcal{T}}$ is solvable. So suppose that $j \geq 1$. We show that this contradicts minimality of j. By Condition (a), all atomic equations are solvable and so the j-th equation is not atomic. Because $M_{\hat{\mathcal{T}}}$ is upper triangular, ρ satisfies all equations in which only the variables $u_{j+1}, \dots, u_n$ occur. We consider the j-th equation $\sum_{i\in[n]} a_{j,i}u_i + b_j = d_j$, which additionally uses the variable u_j. Note that we can change $\rho(u_j)$ without affecting the equations with index greater than j because $M_{\hat{\mathcal{T}}}$ is upper triangular. We show that we can find a value $\rho(u_j)$ of appropriate order also satisfying the j-th equation. We make the following case distinction:

- The j-th equation is global. Then $2\rho(u_i) = 0$ for every $i \in [n]$ for which $a_{j,i} \neq 0$ because all variables classes C_i with nonzero coefficient $a_{j,i}$ are global. That is, $2(\sum_{i\in[n]} a_{j,i}\rho(u_i) + b_j) = 2b_j$. Because $2b_j = 2d_j$ is solvable by Condition (b), it follows that $2(d_j - b_j) = 0$. So $d_j - b_j$ and $b := (d_j - b_j) - \sum_{i\in[n],i\neq j} a_{j,i}\rho(u_i)$ are of order at most 2 because ρ is reasonable. Because $\hat{\mathcal{T}}$ is in local Hermite normal form, $a_{j,j} \equiv 1 \pmod 2$ and we set $\rho(u_i) := b$.

- The j-th equation is local. Then, by local Hermite normality, the variable u_j is local and $a_{j,j} \mid a_{j,i}$ for all i for which u_i is local.

  Let 2^k be the smallest power of 2 annihilating all these $a_{j,i}$. In particular, $k \geq 1$ and $2^k T_j = 2^k b_j$. So $2^k(d_j - b_j) = 0$ by Condition (b). We show $a_{j,j} \mid (d_j - b_j)$. Every $b \in \mathbb{Z}_{2^\ell}$ can be written as $b = 2^{k'} \cdot b'$, for a unit $b' \in \mathbb{Z}_{2^\ell}$ of order 2^ℓ and k' is independent of the choice of b'. The minimal power of 2 annihilating b is $2^{\ell - k'}$. Hence, $a_{j,j} = 2^{\ell-k}d$ for some unit d and $k \geq 0$ because $a_{j,j} \mid a_{j,i}$ and 2^ℓ is the smallest power of 2 annihilating the $a_{j,i}$. Next, $d_j - b_j = 2^{\ell-k'}d'$ for $0 \leq k' \leq k$ for a unit d' because 2^k annihilates $d_j - b_j$. This means that $a_{j,j} \mid (d_j - b_j)$ because $2^{\ell-k} \mid 2^{\ell-k'}$ and $d \mid d'$ (both d and d' are units).

  Lastly, $a_{j,j} \mid a_{j,i}\rho(u_i)$ for all i for which the variable class C_i is global because $a_{j,j} \neq 0$ and $\rho(u_i)$ and thus $a_{j,i}\rho(u_i)$ are of order at most 2. We conclude that $a_{j,j} \mid (d_j - b_j) - \sum_{i \in [n], i \neq j} a_{j,i}\rho(u_i) =: b$ and thus there is a value $\rho(u_j)$ satisfying $a_{j,j}\rho(u_j) = b$. □

Corollary 4.61. *Solvability of weakly global tree-like systems of hyperequations in local Hermite normal form is* CPT*-definable.*

Proof. Let $\hat{\mathcal{T}}$ be a weakly global system of hyperequations. Recall that we first check the conditions of Lemma 4.55. Assume that the check is successful and let $T = d$ be a hyperequation in $\hat{\mathcal{T}}$. Let k be the smallest $k \geq 1$ such that 2^k annihilates all coefficients of local variable classes in T. Then, by Lemma 4.58, we know that the term $2^k T$ is constant because $\hat{\mathcal{T}}$ is weakly global and $k \geq 1$. So we can use Lemma 4.59 to define the value of $2^k T$ and check the conditions of Lemma 4.60 to define whether $\hat{\mathcal{T}}$ is solvable. □

To define solvability of weakly global TCESs in CPT, it only remains to convert a weakly global tree-like system of hyperequations into local Hermite normal form. We treat local and global hyperequations differently. For the local ones we use Gaussian elimination for rings as described in Section 4.5.2. For the global ones, Gaussian elimination for fields suffices.

Lemma 4.62. *There is a* CPT*-term that converts a weakly global tree-like system of hyperequations over $\mathbb{Z}_{2^\ell}$ into an equivalent one in local Hermite normal form.*

Proof. Let $\hat{\mathcal{T}}$ be a weakly-global system of hyperequations. We first define the global variable classes and rearrange the orders of every local component such that the global variables are at the end. Second, we transform the local variables and local hyperequations of each local component into Hermite normal form. This can be done as described in [103] and outlined above where we ignore the coefficients of global variables (of course, they are manipulated when adding equations, but we do not care about divisibility for them and are allowed to do so as stated by local Hermite normality). The local components can be processed in parallel because they are independent of each other.

Third, we need to process the remaining global equations. The local components are processed inductively following the tree on the local components induced by the preorder $\preceq$ of $\hat{\mathcal{T}}$. We now write $C \preceq L$ if $C \preceq C'$ for some $C' \in L$ and $C \prec C'$ if additionally $C \notin L$. Let L be a local component. Assume that, for every local component L' that is a direct successor of L, the equation system of all equations with nonzero coefficient of at least one variable class $C \preceq L'$ is in local Hermite normal form. We can simply combine these systems (i.e., form the union of all equations) for all direct successors of L and still obtain an equation system in local Hermite normal form. We cannot define an order between these successors, but any order of them yields an upper triangular coefficient matrix because a branch of the variable tree cannot use variables of another one.

Let $G \subseteq L$ be the set of global variable classes of L. Let $S_G \subseteq S$ be the set of global equations in which some variable classes in G have nonzero coefficient but all variable classes $C \prec L$ have coefficient zero (the others with nonzero coefficient are already in local Hermite normal form by the induction hypothesis). The variable classes G are the largest ones (with respect to $\preceq$) ocurring with nonzero coefficent in S_G and thus S_G only uses variable classes on the path from L to the root of the variable tree. Because $\hat{\mathcal{T}}$ is tree-like, the order on the hyperequations is total on S_G. Also, the variable classes occurring with nonzero coefficients in S_G are ordered. Thus, we can apply Gaussian elimination on S_G (recall we are working modulo 2 in a field) to bring the equations in S_G into upper triangular form. Note that we only add hyperterms with variables of the same root-to-leaf path here, and hence the resulting equation system is tree-like again.

Finally, the coefficient matrix enlarged by the variable classes in L is in local Hermite normal form. Of course, we cannot process the local components one-by-one, but we have to process the local components of the same level in the tree in parallel. This is, as discussed before, possible because they are independent of each other. □

Theorem 4.63. *Solvability of weakly global TCESs over $\mathbb{Z}_q$ for every prime power q is* CPT-*definable.*

Proof. Let $\mathcal{T}$ be a TCES over $\mathbb{Z}_q$. If q is a power of an odd prime, it can be solved with the procedure for CESs (Corollary 4.52). If q is a power of 2, we first translate $\mathcal{T}$ into the weekly-global tree-like system of hyperequations $\text{Hyp}(\mathcal{T})$ (Lemma 4.54), apply Lemma 4.62 to convert $\text{Hyp}(\mathcal{T})$ to local Hermite normal form, and then check it for solvability using Corollary 4.61. □

We note that solving a linear equation system over $\mathbb{Z}_{p^\ell}$ can be reduced to solving multiple linear equation systems over $\mathbb{Z}_p$ for example by using Hensel's Lemma. But this technique requires not only to check an equation system for solvability, but also to compute a solution. This cannot be done in CPT in general because there are TCESs for which every solution is equivalent to exponentially many other solutions (under automorphisms of the system).

4.5.5 Intersecting Solution Spaces of TCESs

To canonize structures with dihedral colors later, we need to combine two TCESs $\mathcal{T}_1$ and $\mathcal{T}_2$ into a TCES $\mathcal{T}$, such that the solution space of $\mathcal{T}$ is the intersection of the solutions spaces of $\mathcal{T}_1$ and $\mathcal{T}_2$. For two CES with the same variable classes this is indeed easy: We just take the disjoint union of the equations and one of the preorders. For TCESs, the situation is more complicated because the tree structures might not be compatible (so we cannot just take the preorder of one TCES). We will now devise a strategy to combine TCESs under certain conditions. Our solution allows that the variables (and their orders) disagree in the different TCES to a certain extent. In the following we write $L(\mathcal{T})$ for the solution space of a TCES.

Definition 4.64 (Topmost Variables). Let $\mathcal{T} = (V, S, \preceq)$ be a TCES. We say that the variables in $V' \subseteq V$ are the **topmost variables** of $\mathcal{T}$ if V' is the set of all variables of the local component L_r that contains the root class of the variable tree (formally, $V' = \bigcup L_r$).

Definition 4.65 (Compatible TCES). Let $\mathcal{T}_i = (V_i, S_i, \preceq_i)$ be two TCESs over $\mathbb{Z}_q$ for $i \in [2]$ using variables V_i and with topmost variables V_i'. We call $\mathcal{T}_1$ and $\mathcal{T}_2$ **compatible** if $V_1 \cap V_2 = V_1' \cap V_2'$ and $V_1' \cap V_2'$ is a union of variable classes of $\mathcal{T}_i$ for all $i \in [2]$.

The common topmost variables are required to be a union of color classes of both TCES because the two TCES can define different orders on the topmost variables.

Definition 4.66 (Ordered Union of TCESs). Let $\mathcal{T}_i = (V_i, S_i, \preceq_i)$ be two TCESs with topmost variables V_i' for all $i \in [2]$. The **ordered union** of $\mathcal{T}_1$ and $\mathcal{T}_2$ is defined as

$$\mathcal{T}_1 \uplus \mathcal{T}_2 := (V_1 \cup V_2, S_1 \cup S_2, \preceq_1 \cup \preceq_2' \cup \preceq'),$$

where $\preceq_2' = \preceq_2[V_2 \setminus (V_2' \cap V_1')]$ and $\preceq'$ is defined by $V_1' \prec' V_2' \setminus V_1'$ and $V_i' \prec' V_j \setminus V_j'$ for all $i, j \in [2]$.

Note that given an order on the $\mathcal{T}_i$, that is, $\mathcal{T}_1$ can be CPT-distinguished from $\mathcal{T}_2$, then the ordered union $\mathcal{T}_1 \uplus \mathcal{T}_2$ is CPT-definable.

Lemma 4.67. *Let $\mathcal{T}_i = (V_i, S_i, \preceq_i)$ be two TCESs over $\mathbb{Z}_q$ for $i \in [2]$ using variables V_i and with topmost variables V_i'. If $\mathcal{T}_1$ and $\mathcal{T}_2$ are compatible, then $\mathcal{T}_1 \uplus \mathcal{T}_2$ is again a TCES with topmost variables $V_1' \cup V_2'$ and satisfies $L(\mathcal{T}_1 \uplus \mathcal{T}_2) = \bigcap_{i \in [2]} L(\mathcal{T}_i)|^{V_1 \cup V_2}$. If both $\mathcal{T}_1$ and $\mathcal{T}_2$ are weakly global, then $\mathcal{T}_1 \uplus \mathcal{T}_2$ is weakly global, too.*

Proof. The order $\preceq = \preceq_1 \cup \preceq_2' \cup \preceq'$ as defined above forms a tree on the variable classes: By the condition $V_1 \cap V_2 = V_1' \cap V_2'$, the only common variables of the two TCESs are the common topmost ones. On the common variables, we use the order $\preceq_1$. With $\preceq'$ we order the topmost variables of $\mathcal{T}_2$ not common with $\mathcal{T}_1$ after the topmost variables of $\mathcal{T}_1$. That is, considering $\mathcal{T}_2$, we just reorder the variable classes of the root local component because $V_1' \cap V_2'$ is a union of $\mathcal{T}_2$-variable classes. Reordering the variable classes of the root local component of $\mathcal{T}_2$ does not change its tree structure (i.e., its local components and the induced tree on them). By construction, $\mathcal{T}_1 \uplus \mathcal{T}_2$ has topmost variables $V_1' \cup V_2'$. One easily sees that the variables used in a constraint in $\mathcal{T}_i$ for every $i \in [2]$ are still contained in a root-to-leaf path in the variable tree of $\mathcal{T}_1 \uplus \mathcal{T}_2$. Because the set of equations of $\mathcal{T}_1 \uplus \mathcal{T}_2$ is the union of the equations of both TCESs,

$$L(\mathcal{T}_1 \uplus \mathcal{T}_2) = \bigcap_{i \in [2]} L(\mathcal{T}_i)|^{V_1 \cup V_2}$$

follows immediately.

Assume that $\mathcal{T}_1$ and $\mathcal{T}_2$ are weakly global. Because every local component of $\mathcal{T}_i$ is contained in a local component of $\mathcal{T}_1 \uplus \mathcal{T}_2$ for all $i \in [2]$, every global equation of $\mathcal{T}_1 \uplus \mathcal{T}_2$ is a global equation of $\mathcal{T}_1$ or $\mathcal{T}_2$. It follows that every global variable of $\mathcal{T}_1 \uplus \mathcal{T}_2$ is a global variable of $\mathcal{T}_1$ or $\mathcal{T}_2$ and thus that the required constraints of the form $2u = 0$ are present for all global variables. $\square$

We generalize our notation and write $\mathcal{T} = (\mathcal{T}_{p_1}, \dots, \mathcal{T}_{p_k})$ for a sequence of TCESs with pairwise disjoint variables over pairwise coprime prime powers p_i. We denote the solution space of $\mathcal{T}$ by $L(\mathcal{T})$. A series of TCESs $(\mathcal{T}_1, \dots, \mathcal{T}_k)$ has the topmost variables $V' = V'_1 \cup \dots \cup V'_k$ if $\mathcal{T}_i$ has the topmost variables V'_i for every $i \in [k]$.

We now want to form the union of two series of TCESs: Let $\mathcal{T} = (\mathcal{T}_{p_1}, \dots, \mathcal{T}_{p_\ell})$ and $\mathcal{T}' = (\mathcal{T}'_{q_1}, \dots, \mathcal{T}'_{q_{\ell'}})$ be two series of TCESs. We assume that if p_i and q_j are prime powers of the same prime, then actually $p_i = q_j$. This can always be ensured as follows. Assume w.l.o.g. that $p_i < q_j$. We turn $\mathcal{T}_{p_i}$ into a TCES over $\mathbb{Z}_{q_j}$ by adding constraints $p_i \cdot u = 0$ for all variables of $\mathcal{T}_{p_i}$. These constraints embed $\mathbb{Z}_{p_i}$ into $\mathbb{Z}_{q_j}$. We write $\mathcal{T} \uplus \mathcal{T}'$ for the series of TCESs obtained by forming the union of all $\mathcal{T}_{p_i}$ and $\mathcal{T}'_{q_j}$ whenever $p_i = q_j$. The remaining TCESs are just copied. Lemma 4.67 generalizes to series of TCESs by making the assumptions of the lemma for all TCESs $\mathcal{T}_{p_i}$ and $\mathcal{T}'_{q_j}$ for which p_i and q_i are powers of the same prime.

4.6 Canonizing Structures with Dihedral Color Classes

In this section, we finally consider canonization of structures with dihedral color classes. Recall that, for our canonization problem, the reduction to normal forms (Theorem 4.41) shows that we can assume the input structure to be a dihedral 2-injective quotient structure. Our further strategy is as follows: We want to reduce canonization of dihedral 2-injective quotient structures to that of structures with abelian color classes and then apply the canonization for abelian color classes. The main idea is to artificially prohibit reflections in one color class and then hope that this prohibits reflections in other color classes as well. For this, we want to exploit the classification of 2-injective subdirect products of dihedral groups (Theorems 4.5 and 4.6) saying that most 2-injective subdirect products are rotate-or-reflect groups. In particular, if we prohibit reflections in one color class of a rotate-or-reflect group, then reflections in the other color classes are prohibited, too. This effect of prohibiting reflection continues through most 2-injective subdirect products and quotient color classes. However, it does not have to reach all color classes since some 2-injective subdirect products are not rotate-or-reflect groups (for example if one factor is abelian). We will call the parts of the structure in which reflections are linked in this way and can only occur simultaneously reflection components. We analyze how reflection components can depend on each other. It will turn out that different reflection components can indeed only be connected through abelian color classes. We will call these color classes border color classes. Overall, we will follow a two-leveled approach: On the top-level, we deal with the dependencies between the border (and all other abelian) color classes and, on the second level, we consider each reflection component on its own and how it is embedded in its border color classes. To ensure that the border color classes are indeed all abelian, we have to forbid the single exception in Theorem 4.5, which is not a rotate-or-reflect group, namely the double CFI group.

Definition 4.68 (Double-CFI-Free Structure). We call a 2-injective dihedral quotient structure $\mathfrak{A}$ **double-CFI-free** if, for every basic constituent $T \in \mathbb{T}^{\mathfrak{A}}_{\mathsf{gr}}$, the group $\Gamma^{\mathfrak{A}}_T$ is neither isomorphic to the double CFI group Γ_{2CFI} nor to $\Gamma_{\mathsf{2CFI}} \cap (\mathsf{Rot}(\mathsf{D}_4) \times \mathsf{D}_4 \times \mathsf{D}_4)$.

We consider two natural classes of structures that are double-CFI-free after applying the preprocessing.

Definition 4.69 (Odd-Dihedral). A dihedral q-bounded structure $\mathfrak{A}$ is **odd-dihedral** if, for every dihedral color class $C \in \mathbb{C}_{\mathfrak{A}}$, there is an odd $k \in \mathbb{N}$ such that $\mathsf{Aut}(\mathcal{C}) \cong \mathsf{D}_k$.

Lemma 4.70. *Let $\mathfrak{A}$ be a dihedral q-bounded structure of arity at most 3. If $\mathfrak{A}$ satisfies one of the following conditions, then the 2-injective quotient structure $\mathfrak{A}'$ obtained after applying Theorem 4.41 is double-CFI-free:*

1. *$\mathfrak{A}$ is odd dihedral.*

2. *$\mathfrak{A}$ is a binary structure.*

Proof.

1. Let $\mathfrak{A}'$ be the 2-injective quotient structure obtained by the preprocessing steps and let $C' \in \mathbb{C}_{\mathfrak{A}'}$ have a dihedral automorphism group. By Theorem 4.41 and Lemma 4.40, there is a color class $C \in \mathbb{C}_{\mathfrak{A}}$ such that $\mathsf{Aut}(\mathfrak{A}'[C'])$ is a section of $\mathsf{Aut}(\mathfrak{A}[C])$. By assumption, there is an odd k such that $\mathsf{Aut}(\mathfrak{A}[C]) \cong \mathsf{D}_k$ and thus $\mathsf{Aut}(\mathfrak{A}'[C']) \cong \mathsf{D}_{k'}$ for an odd k'. So in $\mathfrak{A}'$ no color class has an automorphism group isomorphic to D_4 and thus $\mathfrak{A}'$ is double-CFI-free.

2. For every 2-injective subdirect product occurring in $\mathfrak{A}$, there is at least one factor such that the projection on it is the trivial group by Lemma 4.40. This is not the case for the two forbidden groups, so they cannot occur. □

4.6.1 Reflection Components

Let $q \in \mathbb{N}$, τ be a signature, and $\mathfrak{A} = (A_{\mathsf{gr}} \uplus A_{\mathsf{ex}}, R_1^{\mathfrak{A}}, \dots, R_k^{\mathfrak{A}}, \preceq^{\mathfrak{A}})$ be an arbitrary dihedral q-bounded 2-injective double-CFI-free quotient τ-structure in standard form, which we fix throughout this section. Whenever we construct a CPT-term in the following, it does not depend on $\mathfrak{A}$ but is evaluated on $\mathfrak{A}$ and it satisfies the claimed properties for all dihedral 2-injective double-CFI-free quotient τ-structures.

We introduce notation: We use the set $\mathbb{O} := \{\uparrow, \downarrow\}$ to denote orientations (e.g., think of the two possible ways to turn an undirected cycle into a directed cycle). For an orientation $o \in \mathbb{O}$, we set $\tilde{o} := o'$ as the reverse orientation, so that $\mathbb{O} = \{o, o'\}$.

Definition 4.71 (Orientation). We say that a structure $\mathfrak{A}' = (A_{\mathsf{gr}} \uplus A_{\mathsf{ex}}, R_1^{\mathfrak{A}}, \dots, R_k^{\mathfrak{A}}, \preceq^{\mathfrak{A}'})$ is an **orientation** of $\mathfrak{A}$ if $\preceq^{\mathfrak{A}'}$ refines $\preceq^{\mathfrak{A}}$ with the following property: For every color class $C \in \mathbb{C}_{\mathfrak{A}}$ that is split by $\preceq^{\mathfrak{A}'}$, the group $\mathsf{Aut}(\mathfrak{A}[C])$ is non-abelian and dihedral and C is split into two color classes $C^{\uparrow}$and $C^{\downarrow}$, such that each of the two classes contains one of the two oriented cycles inducing the standard form in C. We say that $\mathfrak{A}'$ **orients** C.

By splitting the color class C in the above manner, we precisely forbid the reflections in C. Note that an orientation modifies only the preorder of the structure. Hence, defining an orientation of $\mathfrak{A}$ is always canonization preserving because we can easily undo the changes in CPT.

For a dihedral color class C, we can define in CPT two orientations $\mathfrak{A}_C^o$ for both $o \in \mathbb{O}$ that only orient C (by the two possible orderings of $C^o \prec' C^{\tilde{o}}$). Of course, we cannot choose one orientation canonically. But the orientation of C can canonically be propagated to other color classes in the following cases:

(a) whenever C is part of a rotate-or-reflect group (because once we cannot reflect in one component, we cannot do so in the others) and

(b) whenever C' is a quotient of C or vice versa (because, unless C' is already abelian, removing reflections in C also removes reflection from quotients and vice versa).

We now formalize the propagation of orientations.

Definition 4.72. We define the relation $\Uparrow$ on the color classes of $\mathfrak{A}$ as follows: $C_1 \Uparrow C_2$ if and only if both C_1 and C_2 are dihedral and one of the following conditions hold:

(a) $C_1, C_2 \subseteq A_{\mathsf{gr}}$ and there is (up to reordering of the color classes) a group constituent $T = (C_1, C_2, C_3) \in \mathbb{T}_{\mathsf{gr}}$

(b) $C_i \subseteq A_{\mathsf{gr}}$, $C_j \subseteq A_{\mathsf{ex}}$, and C_i is a quotient of C_j for $\{i, j\} = [2]$.

The equivalence relation $\Uparrow\!\!\Uparrow$ is the reflexive and transitive closure of $\Uparrow$.

Note that if $C_1 \Uparrow\!\!\Uparrow C_2$ and $C_1 \neq C_2$, then both C_1 and C_2 are non-abelian. First, we show that if $C_1 \Uparrow\!\!\Uparrow C_2$ and given an orientation of C_1, we can define an orientation of both C_1 and C_2 in CPT. Second, we analyze how the structure $\mathfrak{A}$ decomposes into $\Uparrow\!\!\Uparrow$-equivalence classes.

4.6.1.1 Propagation of Orientations

To analyze the effect of orienting one component of a 2-injective subdirect product, we use the classification of 2-injective subdirect products of dihedral groups (Theorems 4.5 and 4.6).

Lemma 4.73. *Let $T = (C_1, C_2, C_3) \in \mathbb{T}_{\mathsf{gr}}$ be a group constituent. Then one of the following holds:*

1. *Γ_T is abelian, in particular C_i is abelian for all $i \in [3]$.*

2. *Γ_T is a rotate-or-reflect group, in particular C_i is non-abelian for all $i \in [3]$.*

3. *Up to permutation of the color classes, C_1 and C_2 are non-abelian, $\mathsf{Aut}(\mathcal{C}_3)$ is isomorphic to one of $\{\mathsf{D}_2, \mathsf{C}_2, \mathsf{C}_1\}$, and $\overline{\pi}_{C_3}(\Gamma_T)$ is a rotate-or-reflect group.*

Proof. If Conclusion 1 does not hold, then, w.l.o.g., C_1 is non-abelian. By the properties of 2-injective quotient structures, Γ_T is a 2-injective subdirect product. If C_i is non-abelian for every $i \in [3]$, then the group Γ_T is either a rotate-or-reflect group or the double CFI group by Theorem 4.5. The later one is impossible because $\mathfrak{A}$ is double-CFI-free.

Assume C_2 is non-abelian and C_3 is abelian. If $\mathsf{Aut}(\mathcal{C}_3)$ is a cyclic group, then $\overline{\pi}_{C_3}(\Gamma_T)$ is a rotate-or-reflect group by Theorem 4.6 and $\mathsf{Aut}(\mathcal{C}_3) \in \{\mathsf{C}_2, \mathsf{C}_1\}$. The double CFI case is not possible again because $\mathfrak{A}$ is double-CFI-free. If $\mathsf{Aut}(\mathcal{C}_3)$ is not a cyclic group, then $\mathsf{Aut}(\mathcal{C}_3) \cong \mathsf{D}_2$ because D_2 is the only non-cyclic abelian dihedral group. Finally, it cannot be the case that both C_2 and C_3 are abelian by Theorem 4.6. $\square$

Lemma 4.74. *There is a* CPT*-term that, for every $T = (C_1, C_2, C_3) \in \mathbb{T}_{\mathsf{gr}}$ and every orientation $\mathfrak{A}'$ orienting at least one of C_1, C_2, and C_3, defines another orientation $\mathfrak{A}''$ orienting all non-abelian color classes of C_1, C_2, and C_3.*

Proof. Let $T = (C_1, C_2, C_3) \in \mathbb{T}_{\mathsf{gr}}$ and $\mathfrak{A}'$ orient at least one of C_1, C_2, and C_3. If $\mathfrak{A}'$ orients all of the non-abelian color classes, we just set $\mathfrak{A}'' := \mathfrak{A}'$. Otherwise, we set $V := \bigcup_{i \in [3], C_i \text{ not abelian}} C_i$ to be the atoms in non-abelian color-classes (in $\mathfrak{A}$). Then $\Gamma_T^{\mathfrak{A}}|_V$ is a rotate-or-reflect group by Lemma 4.73 and thus $\Gamma_T^{\mathfrak{A}'}|_V$ cannot contain any reflections. We define and order the 2-orbits of $\bigcup_{i \in [3]} C_i$ under $\Gamma_T^{\mathfrak{A}'}$ using Lemma 4.27. Then, for each color class with dihedral automorphism group, the two directed cycles belong to different orbits and we define $\preceq''$ accordingly. □

It remains to consider quotient color classes:

Lemma 4.75. *There is a* CPT*-term that, for every dihedral extension color class $C \subseteq A_{\mathsf{ex}}$, every non-abelian N-quotient $C' \subseteq A_{\mathsf{gr}}$ of C (where $N \triangleleft \mathsf{Aut}(\mathcal{C})$), and every orientation $\mathfrak{A}'$ orienting one of C and C', defines an orientation $\mathfrak{A}''$ orienting both C and C'.*

Proof. Let C, C', N, and $\mathfrak{A}'$ be as required by the lemma. Note that $N \leq \mathsf{Rot}(\mathsf{Aut}(\mathcal{C}))$ because $\mathsf{Aut}(\mathcal{C}')$ is non-abelian and all normal subgroups of a dihedral group are abelian. Because N does not contain reflections, an atom in one standard form cycle of C is never in the same N-orbit as an atom of the other cycle. In particular, the atoms of one standard from cycle of C are adjacent via the orbit-map relation only to atoms in one standard form cycle of C' (and the same for the other standard from cycles of C and C'). Hence, if there is an order on the two cycles of C (or C', respectively), we can lift the order to the two cycles of C' (or C, respectively) via the orbit-map. This approach is clearly CPT-definable. □

Corollary 4.76. *There is a* CPT*-term that, for every $C_1 \Uparrow C_2 \in \mathbb{C}_{\mathfrak{A}}$ and every orientation $\mathfrak{A}'$ orienting C_1, defines an orientation $\mathfrak{A}''$ orienting both C_1 and C_2.*

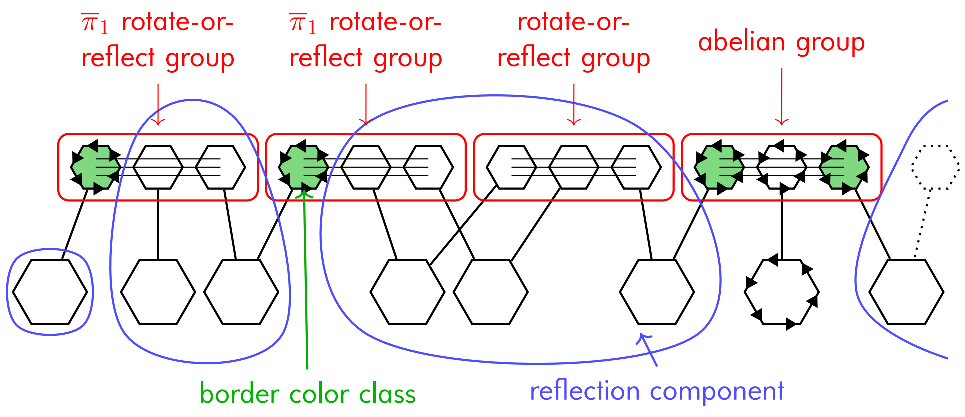
Proof. Because $\Uparrow$ is the reflexive and transitive closure of $\Uparrow$, it suffices to show that the lemma holds for the relation $\Uparrow$. Let $C_1 \Uparrow C_2 \in \mathbb{C}_{\mathfrak{A}}$ and $\mathfrak{A}'$ be an orientation orienting C_1. If C_1 and C_2 are non-abelian group color classes and up to reordering of the color classes $(C_1, C_2, C_3) \in \mathbb{T}_{\mathsf{gr}}$ is a group constituent, then the claim follows by Lemma 4.74. Otherwise, w.l.o.g., C_1 is a non-abelian quotient of C_2 and the claim follows by Lemma 4.75. □

4.6.1.2 Reflection Components and Border Color Classes

Corollary 4.76 shows that given an orientation of one color class C, we can orient the $\Uparrow$-equivalence class of C in CPT. This observation gives rise to the following definition:

Definition 4.77 (Reflection Component). We call a $\Uparrow$-equivalence class $D \subseteq \mathbb{C}_{\mathfrak{A}}$ containing only non-abelian color classes a **reflection component**.

Figure 4.7 shows an example of a dihedral 2-injective structure with its reflection components. Recall that by definition of $\Uparrow$, the abelian color classes are all in singleton equivalence classes. Because all color classes of a reflection component can be oriented by orienting only a single color class, we can speak of the two orientations of a reflection


$\overline{\pi}_1$ rotate-or-reflect group
$\overline{\pi}_1$ rotate-or-reflect group
rotate-or-reflect group
abelian group
border color class
reflection component

4.7 A 2-injective quotient structure with dihedral colors. A dihedral color class is drawn as hexagon, a cyclic color class as directed hexagon (cf. Figure 4.3). The group color classes are at the top and the extension classes are at the bottom. The horizontal lines depict relations of arity 3. A vertical lines between group and extension color classes depict the orbit-maps. The reflection components are encircled in blue and border color classes are green. By the classification of the 2-injective subdirect products of dihedral and cyclic groups, all occurring groups are either abelian or rotate-or-reflect-groups.

component D. They can be defined in CPT by first orienting the smallest color class (with respect to $\preceq$) contained in D and then applying Corollary 4.76. We make some simple observations:

Corollary 4.78. *There is a* CPT*-term defining the set of all reflection components of* $\mathfrak{A}$.

Corollary 4.79. *The two orientations* $\mathfrak{A}[D]^o$ *for* $o \in \mathbb{O}$ *of every reflection component* D *have abelian colors.*

Corollary 4.80. $\mathsf{Aut}(\mathfrak{A}[D])$ *is a rotate-or-reflect group for every reflection component* D.

Corollary 4.81. *Canonization of dihedral* q*-bounded* 2*-injective double-CFI-free quotient* τ*-structures containing at most one reflection component is* CPT*-definable for all* q *and* τ.

Proof. Assume that $\mathfrak{A}$ has only one reflection component D. We define the two orientations $\mathfrak{A}^o$ for $o \in \mathbb{O}$ of $\mathfrak{A}$ by orienting the reflection component D. Both $\mathfrak{A}^o$ have abelian color classes. We define canons $\mathsf{can}(\mathfrak{A}^o)$ using the CPT-canonization for q-bounded structures with abelian colors [118]. From these, we define two canons $\overline{\mathsf{can}}(\mathfrak{A}^o)$ of $\mathfrak{A}$ by undoing the orientation. If they are different, we output the lexicographically smaller one. Otherwise, we output the unique element contained in the set $\{\overline{\mathsf{can}}(\mathfrak{A}^\uparrow), \overline{\mathsf{can}}(\mathfrak{A}^\downarrow)\}$. □

We now deal with the case that $\mathfrak{A}$ contains multiple reflection components and thus have to deal with the restrictions that possible canons of one reflection component impose on other reflection components. To do so, we will define the canonical labelings (recall, the set of isomorphisms into the canon) of a reflection component. Because this set can be exponentially large, we will encode it as a solution of an equation system (similar to the approach to define canonization of abelian color classes). In fact, we will use the two equations systems given for the two orientations of a reflection component D to define an equation system for D. However, we will actually not be interested in the entire set of canonical labelings, but only in their restrictions onto the color classes connecting a reflection component to the rest of the structure. We now analyze properties of color classes that connect different reflection components.

Definition 4.82 (Border Color Class)**.** Let $D \subseteq \mathbb{C}_{\mathfrak{A}}$ be a reflection component. We call a color class $C \in \mathbb{C}_{\mathfrak{A}}$ a **border color class of** D if $C \notin D$ and C is related to a color class contained in D. We denote by $B(D)$ the set of all border color classes of D.

See Figure 4.7 for an illustration of border color classes.

Corollary 4.83. *There is a* CPT*-term that, for every reflection component D, defines the set of border color classes $B(D)$.*

Lemma 4.84. *Let $D \subseteq \mathbb{C}_{\mathfrak{A}}$ be a reflection component and $C \in B(D)$ be a border color class of D. Then $\mathsf{Aut}(\mathcal{C})$ is isomorphic to one of $\{\mathsf{C}_1, \mathsf{C}_2, \mathsf{D}_2\}$ and C is a group color class.*

Proof. Let $C' \in D$ be related to C (such a C' exists by the definition of a border color class). Because $C' \in D$, its automorphism group is non-abelian. We first make a case distinction on C. If $C \subseteq A_{\mathsf{gr}}$ is a group color class, we make a second case distinction on C'. If $C' \subseteq A_{\mathsf{gr}}$ is a group color class, too, let $C_1 = C$, $C_2 = C'$, and C_3 be the three related group color classes forming a 2-injective subdirect product. Because C' is non-abelian, at least two of the C_i are non-abelian by Lemma 4.73. So at least one of C_1 and C_3 is non-abelian (because $C_2 = C'$ is non-abelian). If $C_1 = C$ is non-abelian, then $C \Uparrow C'$ contradicting that $C \in B(D)$. So C must be abelian and hence $\mathsf{Aut}(\mathcal{C})$ is isomorphic to $\mathsf{D}_2, \mathsf{C}_2$, or C_1 by Lemma 4.73.

If $C' \subseteq A_{\mathsf{ex}}$ is an extension color class, then C is a quotient color class of C'. In particular, C is a group color class. If C is non-abelian, then $C \Uparrow C'$ contradicting $C \in B(D)$. The only abelian quotients of C are isomorphic to $\mathsf{D}_2, \mathsf{C}_2$, or C_1. Finally, consider the case that $C \subseteq A_{\mathsf{ex}}$ is an extension color class. Then $C' \subseteq A_{\mathsf{gr}}$ is a quotient of C and because C' is non-abelian, so is C. In particular, we have $C \Uparrow C'$. Once again, we obtain a contradiction to $C \in B(D)$. $\square$

We have seen that the border color classes of a reflection component D are all abelian group color classes. Additionally, two reflection components can only be connected through them (there could, e.g., be other cyclic groups between the border color classes). In other words, the reflection components are embedded in a global abelian part of the structure (cf. Figure 4.7).

Definition 4.85. For a reflection component D, we define $\mathfrak{A}_D := \mathfrak{A}[B(D) \cup \bigcup D]$. We denote the two CPT-definable (abelian) orientations of $\mathfrak{A}_D$ by $\mathfrak{A}_D^o$ for $o \in \mathbb{O}$.

4.6.1.3 Canonical Labelings of Reflection Components

Let D be a reflection component and assume we are given canons $\mathsf{can}(\mathfrak{A}_D^o)$ for all $o \in \mathbb{O}$ (with respect to some fixed canonization). We denote by $\overline{\mathsf{can}}(\mathfrak{A}_D^o)$ the structure obtained from $\mathsf{can}(\mathfrak{A}_D^o)$ by undoing the orientation as explained earlier, i.e., by undoing the refinement of the preorder. Then $\mathfrak{A}_D \cong \overline{\mathsf{can}}(\mathfrak{A}_D^o) \upharpoonright \tau$, i.e., $\overline{\mathsf{can}}(\mathfrak{A}_D^o)$ is a canon of $\mathfrak{A}_D$.

Let $\leq$ be the lexicographical order on ordered τ-structures. We define the canon $\mathsf{can}(\mathfrak{A}_D)$ to be the $\leq$-minimal canon $\overline{\mathsf{can}}(\mathfrak{A}_D^o)$ with $o \in \mathbb{O}$. We now analyze the canonical labelings of a reflection component.

Lemma 4.86. *If* $\mathsf{can}(\mathfrak{A}_D^o) < \mathsf{can}(\mathfrak{A}_D^{\tilde{o}})$ *for some* $o \in \mathbb{O}$*, then we have* $\mathsf{Iso}(\mathfrak{A}_D, \mathsf{can}(\mathfrak{A}_D)) = \mathsf{Iso}(\mathfrak{A}_D^o, \mathsf{can}(\mathfrak{A}_D^o))$.

Proof. Assume $\mathsf{can}(\mathfrak{A}_D^o) < \mathsf{can}(\mathfrak{A}_D^{\tilde{o}})$ for some $o \in \mathbb{O}$. We first show

$$\mathsf{Iso}(\mathfrak{A}_D^o, \mathsf{can}(\mathfrak{A}_D^o)) \subseteq \mathsf{Iso}(\mathfrak{A}_D, \mathsf{can}(\mathfrak{A}_D)).$$

Because $\mathfrak{A}_D^o$ has just a finer preorder than $\mathfrak{A}_D$ and by construction $\mathsf{can}(\mathfrak{A}_D) = \overline{\mathsf{can}}(\mathfrak{A}_D^o)$, a canonical labeling in $\mathsf{Iso}(\mathfrak{A}_D^o, \mathsf{can}(\mathfrak{A}_D^o))$ is also a canonical labeling in $\mathsf{Iso}(\mathfrak{A}_D, \mathsf{can}(\mathfrak{A}_D))$.

We now show the reverse direction. Because $\mathsf{Iso}(\mathfrak{A}_D, \mathsf{can}(\mathfrak{A}_D)) = \mathsf{Aut}(\mathfrak{A}_D)\varphi$ for all $\varphi \in \mathsf{Iso}(\mathfrak{A}_D, \mathsf{can}(\mathfrak{A}_D))$ and $\mathsf{Iso}(\mathfrak{A}_D^o, \mathsf{can}(\mathfrak{A}_D^o)) \subseteq \mathsf{Iso}(\mathfrak{A}_D, \mathsf{can}(\mathfrak{A}_D))$. In particular, we have that

$$\mathsf{Iso}(\mathfrak{A}_D, \mathsf{can}(\mathfrak{A}_D)) = \mathsf{Aut}(\mathfrak{A}_D)\varphi$$

for every $\varphi \in \mathsf{Iso}(\mathfrak{A}_D^o, \mathsf{can}(\mathfrak{A}_D^o))$. Fix an arbitrary $\varphi \in \mathsf{Iso}(\mathfrak{A}_D^o, \mathsf{can}(\mathfrak{A}_D^o))$ and assume $\varphi' \in \mathsf{Iso}(\mathfrak{A}_D, \mathsf{can}(\mathfrak{A}_D))$. Then there is a $\psi \in \mathsf{Aut}(\mathfrak{A}_D)$ such that $\varphi' = \psi \circ \varphi$. The automorphism ψ is either a rotation or a reflection on all group and extension color classes of D simultaneously by Corollary 4.80. If ψ is a reflection on one (and hence all by Corollary 4.80) color classes, then it exchanges the two cycles of the standard form within each color class and in particular $\mathfrak{A}_D^{\uparrow} \cong \mathfrak{A}_D^{\downarrow}$ contradicting our assumption. Hence, ψ is a rotation on all color classes and thus $\psi \in \mathsf{Aut}(\mathfrak{A}_D^o)$, i.e., $\mathsf{Aut}(\mathfrak{A}_D^o) = \mathsf{Aut}(\mathfrak{A}_D)$. Then

$$\varphi' = \psi \circ \varphi \in \mathsf{Aut}(\mathfrak{A}_D^o)\varphi = \mathsf{Iso}(\mathfrak{A}_D^o, \mathsf{can}(\mathfrak{A}_D^o)). \qquad \square$$

Lemma 4.87. *If* $\mathsf{can}(\mathfrak{A}_D^{\uparrow}) = \mathsf{can}(\mathfrak{A}_D^{\downarrow})$*, then* $\mathsf{Iso}(\mathfrak{A}_D, \mathsf{can}(\mathfrak{A}_D)) = \bigcup_{o \in \mathbb{O}} \mathsf{Iso}(\mathfrak{A}_D^o, \mathsf{can}(\mathfrak{A}_D^o))$.

Proof. The inclusion $\mathsf{Iso}(\mathfrak{A}_D^o, \mathsf{can}(\mathfrak{A}_D^o)) \subseteq \mathsf{Iso}(\mathfrak{A}_D, \mathsf{can}(\mathfrak{A}_D))$ follows because, for every $o \in \mathbb{O}$, we have $\mathsf{can}(\mathfrak{A}_D) = \overline{\mathsf{can}}(\mathfrak{A}_D^o)$ analogously as in Lemma 4.86. For the other direction, we have

$$\mathsf{Iso}(\mathfrak{A}_D, \mathsf{can}(\mathfrak{A}_D)) = \mathsf{Aut}(\mathfrak{A}_D)\varphi$$

for some fixed (and thus every) $\varphi \in \mathsf{Iso}(\mathfrak{A}_D^{\uparrow}, \mathsf{can}(\mathfrak{A}_D^{\uparrow}))$. Let $\varphi' \in \mathsf{Iso}(\mathfrak{A}_D, \mathsf{can}(\mathfrak{A}_D))$. Then there is a $\psi \in \mathsf{Aut}(\mathfrak{A}_D)$ such that $\varphi' = \psi \circ \varphi$. Now ψ is either a reflection or a rotation on all color classes of D simultaneously by Corollary 4.80. If ψ is a rotation everywhere, then $\psi \in \mathsf{Aut}(\mathfrak{A}_D^{\uparrow})$ and thus

$$\varphi' = \psi \circ \varphi \in \mathsf{Aut}(\mathfrak{A}_D^{\uparrow})\varphi = \mathsf{Iso}(\mathfrak{A}_D^{\uparrow}, \mathsf{can}(\mathfrak{A}_D^{\uparrow})).$$

If ψ is a reflection everywhere, then $\psi \in \mathsf{Iso}(\mathfrak{A}_D^{\uparrow}, \mathfrak{A}_D^{\downarrow}) = \mathsf{Iso}(\mathfrak{A}_D^{\downarrow}, \mathfrak{A}_D^{\uparrow})$ (equality holds because reflections are self-inverse). So $\varphi' = \psi \circ \varphi \in \mathsf{Iso}(\mathfrak{A}_D^{\downarrow}, \mathsf{can}(\mathfrak{A}_D^{\downarrow}))$. $\square$

4.6.2 Canonizing Structures with Abelian Color Classes

To encode the canonical labelings of reflection components with equation systems, we need more details on the CPT-canonization of structures with abelian color classes from [103, 118]. The canonization proceeds inductively: It canonizes an induced substructure consisting of at most r many color classes, where r is the bound on the arity, and adds it to the already computed partial canon computed so far. Because the added substructure is of constant size, one can consider all possible ordered versions of it, and pick the minimal one compatible with the existing canon. This compatibility check is done by checking whether an equation system encoding the canonical labelings of the canon so far and another one encoding the canonical labelings of the new substructure are solvable together. The following theorem is a precise statement of the canonization for bounded abelian colors with all properties needed in this thesis (cf. [103, Lemma 6.12]). For a finite set M, denote by $\mathcal{O}(M)$ the set of orderings $M \to [|M|]$.

Theorem 4.88 ([103, Section 6.2.2]). *For all $q, r \in \mathbb{N}$ and every signature τ, there is a* CPT*-term that, for q-bounded τ-structure $\mathfrak{A}$ with transitive and abelian colors of arity r, defines a canon $\mathsf{can}(\mathfrak{A})$ on the universe $[|A|]$, cosets of orderings $\Lambda_C = \mathsf{Aut}(\mathcal{C})\sigma_C \subseteq \mathcal{O}(C)$ with $\sigma_C \in \mathcal{O}(C)$ for every color class $C \subseteq \mathbb{C}_{\mathfrak{A}}$, pairwise coprime prime powers $p_1, \dots, p_\ell$, distinct variable sets $V_1, \dots, V_\ell$, and a sequence of CESs $\mathcal{S} = (\mathcal{S}_{p_1}, \dots, \mathcal{S}_{p_\ell})$ such that $\mathcal{S}_i$ is a CES over $\mathbb{Z}_{p_i}$ using variables V_i and there is a canonical embedding*

$$\iota : \bigotimes_{C \in \mathbb{C}_{\mathfrak{A}}} \Lambda_C \to \bigotimes_{i \in [\ell]} \mathbb{Z}_{p_i}^{V_i},$$

with the following properties:

1. *$L(\mathcal{S}) \subseteq \operatorname{im}(\iota)$ and $\iota^{-1}(L(\mathcal{S})) = \mathsf{Iso}(\mathfrak{A}, \mathsf{can}(\mathfrak{A}))$ via the canonical isomorphism between $\mathcal{O}(A)$ and $\bigotimes_{C \in \mathbb{C}_{\mathfrak{A}}} \mathcal{O}(C)$ given by the total order on the color classes.*

2. *For every $C \in \mathbb{C}_{\mathfrak{A}}$ the following holds: Let $q_1, \dots, q_k$ be the prime powers such that $\mathsf{Aut}(\mathcal{C}) \cong \bigotimes_{i \in [k]} \mathsf{C}_{q_i}$. Then there are numbers $j_1, \dots, j_k \in [\ell]$ and variable sets $V_1^C \subseteq V_{j_1}, \dots, V_k^C \subseteq V_{j_k}$ such that $q_i \mid p_j$ and there are constraints $q_i \cdot u = 0$ for all $u \in V_i^C$ in $\mathcal{S}_{p_j}$ embedding $\mathbb{Z}_{q_i}$ into $\mathbb{Z}_{p_j}$ for all $i \in [k]$.*

3. *There is a* CPT*-term (not depending on $\mathfrak{A}$) that, for every $C \in \mathbb{C}_{\mathfrak{A}}$, defines the variable sets V_i^C from above such that $V_i^C \subseteq \mathsf{HF}(C)$ and the cyclic constraints on these variable sets. It actually suffices to evaluate the term on $\mathcal{C}$.*

We do not write the cosets Λ_C as $\mathsf{Aut}(\mathcal{C})\sigma_C$, because we cannot choose some $\sigma_C \in \Lambda_C$ in CPT. The set $\bigotimes_{C \in \mathbb{C}_{\mathfrak{A}}} \Lambda_C$ naturally corresponds to a subset of $\mathcal{O}(A)$, namely to the orderings refining the preorder on A. In the following, we just identify these sets. Now, we show that we can start the canonization of abelian color classes with a TCES, that describes initial restrictions on the color classes. First, we define the set of orderings encoded by a TCES.

Definition 4.89 (Orderings Encoded by a TCES). Let $\mathfrak{A}$ be a q-bounded structure with abelian color classes, Λ_C be the cosets of orderings for every $C \in \mathbb{C}_{\mathfrak{A}}$, $p_1, \dots, p_\ell$ the prime powers, $V := V_1 \cup \dots \cup V_\ell$ the variables, and ι the embedding given by Theorem 4.88. Furthermore, let $\mathcal{T}$ be a series of TCESs using variables $V_{\mathcal{T}}$ such that V is contained in the topmost variables of $\mathcal{T}$. We say that $\mathcal{T}$ **encodes** the set of orderings $\Lambda \subseteq \mathcal{O}(A)$ if $L(\mathcal{T})|_V \subseteq \operatorname{im}(\iota)$ and $\iota^{-1}(L(\mathcal{T})|_V) = \Lambda$.

Lemma 4.90. *For all $q, r \in \mathbb{N}$ and every signature τ, there is a* CPT*-term that, for every q-bounded structure $\mathfrak{A}$ of arity r with transitive abelian color classes and every series of weakly global TCESs $\mathcal{T}$ encoding a nonempty set of orderings $\Lambda \subseteq \mathcal{O}(A)$, defines a canon $\mathsf{can}(\mathfrak{A}_{\mathcal{T}})$, cosets of orderings Λ_C for every $C \in \mathbb{C}_{\mathfrak{A}}$, pairwise coprime*

prime powers $p_1, \dots, p_\ell$, variables $V_1, \dots, V_\ell$, and a series of CESs $\mathcal{S}$ satisfying the conditions of Theorem 4.88 such that $\mathcal{S}$ encodes $\mathsf{Iso}(\mathfrak{A}, \mathsf{can}(\mathfrak{A}_\mathcal{T}))$ and $\mathcal{T} \uplus \mathcal{S}$ encodes the set $\mathsf{Iso}(\mathfrak{A}, \mathsf{can}(\mathfrak{A}_\mathcal{T})) \cap \Lambda \neq \emptyset$.

Proof Sketch. Note that the condition that $\mathcal{S}$ encodes $\mathsf{Iso}(\mathfrak{A}, \mathsf{can}(\mathfrak{A}_\mathcal{T}))$ is equivalent to the conditions on the embedding ι in Theorem 4.88. Let $q, r \in \mathbb{N}$, τ be a signature, and $\mathfrak{A}$ and $\mathcal{T}$ as required. We use $\mathcal{T}$ as an initial equation system in the canonization. Because $\mathcal{T}$ encodes a nonempty set of orderings, we just forbid some orderings initially, but at least one remains. So the canonization is done as before: We add induced substructures of constant size and accumulate all additional constraints in a series of CESs $\mathcal{S}$ using variables V. Then $\mathcal{S}$ encodes the set $\mathsf{Iso}(\mathfrak{A}, \mathsf{can}(\mathfrak{A}_\mathcal{T}))$. The union $\mathcal{T} \uplus \mathcal{S}$ encodes the intersection of both encoded sets by Lemma 4.67 and is weakly global because $\mathcal{S}$ is weakly global. Checks for solvability can be done in CPT by Theorem 4.63. □

4.6.3 Canonizing Structures with Dihedral Colors

For canonizing dihedral colors we want to maintain an equation system encoding all canonical labelings of all abelian color classes (and hence including all border color classes) that extend to canonical labelings of the input structure. This suffices to encode the dependencies between different reflection components because – as we have seen in the previous section – they can only be connected via abelian color classes. As initialization step, we apply the canonization for abelian colors to all abelian color classes. Then we want to inductively add one reflection component D in each step (possibly restricting the canonical labelings of the border color classes). But a reflection component is not of constant size, so we cannot try out all orderings. To overcome this limitation, we want to define a canon of the reflection component D by taking the existing partial canon into account. That is, given an equation system encoding all canonical labelings of the partial canon computed so far, we want to increase both, the equation system and the canon, by D in one step.

Fix $q \in \mathbb{N}$ and a signature τ. For a dihedral q-bounded structure $\mathfrak{A}$, we denote the set of abelian color classes of $\mathfrak{A}$ by $\mathbb{A}_\mathfrak{A} \subseteq \mathbb{C}_\mathfrak{A}$ and leave out the subscript if $\mathfrak{A}$ becomes clear from the context. From now, we assume that the abelian color classes $\mathbb{A}_\mathfrak{A}$ of $\mathfrak{A}$ are smaller than the non-abelian ones (according to the total preorder). If not, we can easily reorder them canonization-preservingly. The algorithmic approach for the canonization is given in Figure 4.8. Recall that for a set of I-indexed tuples $T \subseteq M^I$ and $J \supseteq I$, the extension of T to J is denoted by $T|^J$, that is, the maximal set of J-indexed tuples $T' \subseteq M^J$ such that $T'|_I = T$. For $I \subseteq J \subseteq K$ and $T \subseteq M^J$, we use $T|_I^K$ as shorthand notation for $(T|_I)|^K$. For a set I' of subsets of I, we write $T|_{J'}$ for $T|_{\bigcup J'}$. Likewise, for a set K' of subsets of K such that $\bigcup K' \supseteq J$, we write $T|^K$ for $T|^{\bigcup K'}$. We apply the notation also to functions $J \to M$. We first argue that this algorithm indeed is a canonization and second that it is CPT-definable. From now, we fix an arbitrary dihedral q-bounded double-CFI-free 2-injective quotient τ-structure $\mathfrak{A}$ with reflection components $D_1 < \dots < D_m$. The algorithm maintains canons $\mathsf{can}(\mathfrak{A}_i)$ of $\mathfrak{A}_i := \mathfrak{A}[\mathbb{A} \cup \bigcup_{j \in [i]} D_j]$ and sets Λ_i of canonical labelings.

```
Input: A dihedral $q$-bounded double-CFI-free 2-injective quotient $\tau$-structure $\mathfrak{A}$
Output: The canon $\mathsf{can}(\mathfrak{A})$ of $\mathfrak{A}$
1  Compute the set $\mathbb{A} \subseteq \mathbb{C}_{\mathfrak{A}}$ of abelian color classes;
2  Compute all reflection components $D_1 < \cdots < D_m$ of $\mathfrak{A}$;
3  Compute $\mathsf{can}(\mathfrak{A}_0) := \mathsf{can}(\mathfrak{A}[\mathbb{A}])$ and $\Lambda_0 := \mathsf{Iso}(\mathfrak{A}[\mathbb{A}], \mathsf{can}(\mathfrak{A}[\mathbb{A}]))$ using the canonization for
     abelian colors;
4  for $i \in [m]$ do
5      $D := D_i$;
6      Define the two orientations $\mathfrak{A}^o_D$ for $o \in \mathbb{O}$;
7      Compute $\mathsf{can}(\mathfrak{A}^o_D)$ and $\Lambda^o := \mathsf{Iso}(\mathfrak{A}^o_D, \mathsf{can}(\mathfrak{A}^o_D))$ such that $\Lambda_{i-1} \cap \Lambda^o|^{\mathbb{A}} \neq \emptyset$ with the
         canonization for abelian colors for both $o \in \mathbb{O}$;
8      if $\mathsf{can}(\mathfrak{A}^o_D) < \mathsf{can}(\mathfrak{A}^{\tilde{o}}_D)$ for some $o \in \mathbb{O}$ then
9          $\mathsf{can}(\mathfrak{A}_i) := \mathsf{can}(\mathfrak{A}_{i-1}) \cup \overline{\mathsf{can}}(\mathfrak{A}^o_D)$;
10         $\Lambda_i := \Lambda_{i-1} \cap \Lambda^o|^{\mathbb{A}}_{B(D)}$;
11     else
12         $\mathsf{can}(\mathfrak{A}_i) := \mathsf{can}(\mathfrak{A}_{i-1}) \cup \overline{\mathsf{can}}(\mathfrak{A}^{\uparrow}_D) \cup \overline{\mathsf{can}}(\mathfrak{A}^{\downarrow}_D)$;
13         $\Lambda_i := \Lambda_{i-1} \cap (\Lambda^{\uparrow} \cup \Lambda^{\downarrow})|^{\mathbb{A}}_{B(D)}$;
14 $\mathsf{can}(\mathfrak{A}) := \mathsf{can}(\mathfrak{A}_m)$;
```

4.8 CPT-canonization of 2-injective double-CFI-free structures. The pseudocode of the algorithmic approach to canonize q-bounded structures with dihedral colors in CPT. This section discusses how the algorithm can be implemented in CPT.

Lemma 4.91. *For every $i \leq m$, we have $\mathfrak{A}_i \cong \mathsf{can}(\mathfrak{A}_i) \upharpoonright \tau$ and $\Lambda_i = \mathsf{Iso}(\mathfrak{A}_i, \mathsf{can}(\mathfrak{A}_i))|_{\mathbb{A}}$.*

Proof. For the case $i = 0$, we have $\mathfrak{A}_0 = \mathfrak{A}[\mathbb{A}]$ and both $\mathsf{can}(\mathfrak{A}_0)$ and Λ_0 are given by the canonization of abelian colors as required (Theorem 4.88). So inductively assume $i > 0$, $\mathfrak{A}_{i-1} \cong \mathsf{can}(\mathfrak{A}_{i-1}) \upharpoonright \tau$, and $\Lambda_{i-1} = \mathsf{Iso}(\mathfrak{A}_{i-1}, \mathsf{can}(\mathfrak{A}_{i-1}))|_{\mathbb{A}}$. Let $D = D_i$ and $\mathfrak{A}^o_D$ be the two orientations of $\mathfrak{A}_D$ for $o \in \mathbb{O}$. Using the canonization for abelian colors, we obtain a canon $\mathsf{can}(\mathfrak{A}^o_D)$ of $\mathfrak{A}^o_D$ and the set $\Lambda^o = \mathsf{Iso}(\mathfrak{A}^o_D, \mathsf{can}(\mathfrak{A}^o_D))$ of canonical labelings such that $\Lambda_{i-1} \cap \Lambda^o|^{\mathbb{A}} \neq \emptyset$ for both $o \in \mathbb{O}$ (Lemma 4.90). We perform the same case distinction as in Line 8: If $\mathsf{can}(\mathfrak{A}^o_D) < \mathsf{can}(\mathfrak{A}^{\tilde{o}}_D)$ for some $o \in \mathbb{O}$, then $\mathsf{Iso}(\mathfrak{A}_D, \mathsf{can}(\mathfrak{A}_D)) = \mathsf{Iso}(\mathfrak{A}^o_D, \mathsf{can}(\mathfrak{A}^o_D))$ by Lemma 4.86 and for $\mathsf{can}(\mathfrak{A}_i) = \mathsf{can}(\mathfrak{A}_{i-1}) \cup \overline{\mathsf{can}}(\mathfrak{A}^o_D)$ it holds that

$$\mathsf{Iso}(\mathfrak{A}_i, \mathsf{can}(\mathfrak{A}_i)) = \mathsf{Iso}(\mathfrak{A}_{i-1}, \mathsf{can}(\mathfrak{A}_{i-1}))|^{A_i} \cap \mathsf{Iso}(\mathfrak{A}^o_D, \mathsf{can}(\mathfrak{A}^o_D))|^{A_i},$$

where A_i is the universe of $\mathfrak{A}_i$ and

$$\Lambda_i = \Lambda_{i-1} \cap \Lambda^o|_{B(D)}|^{\mathbb{A}} = \mathsf{Iso}(\mathfrak{A}_{i-1}, \mathsf{can}(\mathfrak{A}_{i-1}))|_{\mathbb{A}} \cap \mathsf{Iso}(\mathfrak{A}^o_D, \mathsf{can}(\mathfrak{A}^o_D))|^{\mathbb{A}}_{B(D)}.$$

The reflection component D is only connected to its border color classes $B(D) \subseteq \mathbb{A}$. So we finally have

$$\begin{aligned} &\Big(\mathsf{Iso}(\mathfrak{A}_{i-1}, \mathsf{can}(\mathfrak{A}_{i-1}))|^{A_i} \cap \mathsf{Iso}(\mathfrak{A}^o_D, \mathsf{can}(\mathfrak{A}^o_D))|^{A_i}\Big)\Big|_{\mathbb{A}} \\ &= \mathsf{Iso}(\mathfrak{A}_{i-1}, \mathsf{can}(\mathfrak{A}_{i-1}))|_{\mathbb{A}} \cap \mathsf{Iso}(\mathfrak{A}^o_D, \mathsf{can}(\mathfrak{A}^o_D))|^{\mathbb{A}}_{B(D)} \end{aligned}$$

and so $\Lambda_i = \mathsf{Iso}(\mathfrak{A}_i, \mathsf{can}(\mathfrak{A}_i))|_{\mathbb{A}}$. The case $\mathsf{can}(\mathfrak{A}^{\uparrow}_D) = \mathsf{can}(\mathfrak{A}^{\downarrow}_D)$ proceeds similarly using Lemma 4.87. □

To express the algorithm in CPT, the main obstacle is that the sets Λ_i cannot be defined directly in CPT because they are possibly exponentially large. Hence, we adapt the approach of the canonization for abelian color classes and encode the sets Λ_i with sequences of weakly global TCESs $\mathcal{T}_i$. We maintain that the variables $V_{\mathbb{A}}$ of the abelian color classes $\mathbb{A}$ are contained in the topmost variables of the $\mathcal{T}_i$. Again, all following CPT-terms do not depend on $\mathfrak{A}$ but are evaluated on $\mathfrak{A}$.

It is clear that Lines 1 to 6 are CPT-definable. The reflection components in Line 2 are CPT-definable by Corollary 4.78. In Line 3, we use the CPT-canonization for abelian color classes. It defines, apart from the canon, a series of CESs $\mathcal{T}_0$ encoding Λ_0 using variables $V_{\mathbb{A}}$ (and hence has topmost variables $V_{\mathbb{A}}$) by Theorem 4.88.

For Line 7, we use Lemma 4.90 to define the canons $\mathsf{can}(\mathfrak{A}^o_D)$ that are compatible with the canon $\mathsf{can}(\mathfrak{A}_{i-1})$ computed so far. We use $\mathcal{T}_{i-1}$ as an initial equation system to canonize $\mathfrak{A}^o_D$, which has topmost variables $V_{\mathbb{A}}$. Because there is at least one canonical labeling, $\emptyset \neq L(\mathcal{T}_{i-1}) \subseteq L(\mathcal{T}_0)$. So Lemma 4.90 can be applied and we obtain two sequences of CESs $\mathcal{S}^o_D$. Note that the variables $V_{B(D)}$ of the border color classes of D are contained in $V_{\mathbb{A}}$ and the variables of $\mathcal{S}^o_D$ for all $o \in \mathbb{O}$.

Lines 8 to 12 are CPT-definable: In Line 10, we set $\mathcal{T}_i := \mathcal{T}_{i-1} \uplus \mathcal{S}^o_D$. Because the common topmost variables of $\mathcal{T}_{i-1}$ and $\mathcal{S}^o_D$ are $V_{B(D)}$, we can apply Lemma 4.67 to show that $\mathcal{T}_i$ encodes Λ_i: The variables in $V_{\mathbb{A}}$ are contained in the topmost variables of $\mathcal{T}_i$, the TCES $\mathcal{T}_i$ encodes $\Lambda_{i-1} \cap \Lambda^o|_{\mathbb{A}}|^{\mathbb{A}}$, and $\mathcal{T}_i$ is weakly global because $\mathcal{T}_{i-1}$ is weakly global (and $\mathcal{S}^o_D$ is a CES).

The intersection in Line 13 can be performed again due to Lemma 4.67. So we are only left to show that, given a CES $\mathcal{S}^o_D$ that encodes $\Lambda^o = \mathsf{Iso}(\mathfrak{A}^o_D, \mathsf{can}(\mathfrak{A}^o_D))$ for a reflection component D and both $o \in \mathbb{O}$, we can define a series of weakly global TCESs with $V_{B(D)}$ contained in its topmost variables encoding $(\Lambda^{\uparrow} \cup \Lambda^{\downarrow})|^{\mathbb{A}}_{B(D)}$. We show this in the following section.

4.6.4 Equation Systems for Reflection Components

Fix an arbitrary $1 < i \leq m$, let $\mathcal{T} = \mathcal{T}_{i-1}$ be the series of weakly global TCESs for the partial canon $\mathsf{can}(\mathfrak{A}_{i-1})$ defined so far, and let $D = D_i$ be the next reflection component to canonize. Let the variables of the border color classes of D be $B = B_1 < \cdots < B_k$. Recall that these variables are CPT-definable only from the border color classes (Theorem 4.88) and that $\mathcal{T}$ already contains cyclic constraints for these variables (because $B \subseteq V_{\mathbb{A}}$). In particular, B is contained in the topmost variables of $\mathcal{T}$, which contain $V_{\mathbb{A}}$.

Next, let $\mathfrak{A}^o_D$ be the two orientations of D for both $o \in \mathbb{O}$. Further, let $\mathsf{can}(\mathfrak{A}^o_D)$ be the canons defined in Line 7 and let $\mathcal{S}^o$ be the series of CESs encoding the sets $\Lambda^o := \mathsf{Iso}(\mathfrak{A}^o_D, \mathsf{can}(\mathfrak{A}^o_D))$ for both $o \in \mathbb{O}$ as defined before (using Lemma 4.90). Note that the two series of CESs use the same variables for the border color classes (because they are equal in both orientations). We have to consider the remaining case $\mathsf{can}(\mathfrak{A}^{\uparrow}_D) = \mathsf{can}(\mathfrak{A}^{\downarrow}_D)$ and thus $\mathsf{can}(\mathfrak{A}_D) = \overline{\mathsf{can}}(\mathfrak{A}^{\uparrow}_D) = \overline{\mathsf{can}}(\mathfrak{A}^{\downarrow}_D)$. We would like to define a unique isomorphism in $\mathsf{Iso}(\mathfrak{A}^{\uparrow}_D, \mathfrak{A}^{\downarrow}_D) = \mathsf{Iso}(\mathfrak{A}^{\downarrow}_D, \mathfrak{A}^{\uparrow}_D)$. But this is not possible in general. Instead, we show that a unique restricted isomorphism in $\mathsf{Iso}(\mathfrak{A}^{\uparrow}_D, \mathfrak{A}^{\downarrow}_D)|_{B(D)}$ (which possibly extends to multiple isomorphisms between the orientations) is CPT-definable.

Note that, by Lemma 4.84, every border color class has an automorphism group isomorphic to C_2^{ℓ} for some $\ell \in \{0, 1, 2\}$ (and are in particular all abelian). Hence, all variables for the border color classes range over $\mathbb{Z}_2$ by Theorem 4.88. (Precisely, B are variables over $\mathbb{Z}_p$ for p a power of 2 and there are constraints $2u = 0$ for all $u \in B$ embedding $\mathbb{Z}_2$ in $\mathbb{Z}_p$, as already discussed for TCESs in Section 4.5.3).

We adapt both series of CESs such that their variables are different, but such that a variable of a border color class of $\mathcal{S}^o$ can still be identified with a variable of a border color class of $\mathcal{S}^{\tilde{o}}$. This can for example be achieved by renaming the variables as follows: $u \mapsto (u, A^o_D)$. We denote by V^o (and B^o, respectively) the changed variables for both $o \in \mathbb{O}$. For two tuples $\bar{b}^o \in \mathbb{Z}_2^{B^o}$, we write $\bar{b}^\uparrow = \bar{b}^\downarrow$ if $\bar{b}^\uparrow$ and $\bar{b}^\downarrow$ are equal up to the one-to-one identification of the variables.

Lemma 4.92. *There is a* CPT*-term that on input $\mathcal{T}$ and D defines two tuples $\bar{b}^o \in \mathbb{Z}_2^{B^o}$ for both $o \in \mathbb{O}$ (that are equal up to identification of the variables, i.e., $\bar{b}^\uparrow = \bar{b}^\downarrow$ in the above notational convention) such that, for every solution $\bar{a}^o$ of $\mathcal{S}^o$, there is a solution $\bar{a}^{\tilde{o}}$ of $\mathcal{S}^{\tilde{o}}$ such that $\bar{a}^o|_{B^o} + \bar{b}^o = \bar{a}^{\tilde{o}}|_{B^{\tilde{o}}}$.*

Proof. We first show that for two solutions $\bar{a}^o$ of $\mathcal{S}^o$ for both $o \in \mathbb{O}$, the tuple $\bar{a}^o|_{B^o} - \bar{a}^{\tilde{o}}|_{B^{\tilde{o}}}$ has the desired property. In the next step, we show how we can define such two tuples $\bar{a}^o|_{B^o}$ in CPT. For both $o \in \mathbb{O}$, we identify $\mathbb{Z}_2^{B^o}$ with $\mathbb{Z}_2^{B^o_1} \times \dots \times \mathbb{Z}_2^{B^o_k}$ via the canonical bijection $B^o \to \bigcup_{i\in[k]} B^o_i$. We will define two tuples of tuples

$$\bar{b}^o = \left(\bar{b}^o_1, \dots, \bar{b}^o_k\right) \in \mathbb{Z}_2^{B^o_1} \times \dots \times \mathbb{Z}_2^{B^o_k}.$$

Let $\bar{a}^o$ be a solution of $\mathcal{S}^o$ for both $o \in \mathbb{O}$. The two tuples $\bar{a}^o$ encode isomorphisms $\varphi^o\colon \mathfrak{A}^o_D \to \mathsf{can}(\mathfrak{A}^o_D) \upharpoonright \tau$ via the embedding ι given by canonization for abelian colors (cf. Theorem 4.88). They induce two isomorphisms $\psi^o := \varphi^o \circ (\varphi^{\tilde{o}})^{-1} : \mathfrak{A}^{\tilde{o}}_D \to \mathfrak{A}^o_D$ (recall that $\mathsf{can}(\mathfrak{A}^o_D) = \mathsf{can}(\mathfrak{A}^{\tilde{o}}_D)$). We cannot describe the action of the isomorphisms ψ^o on the reflection component D by tuples (because D is not abelian). But on the border color classes the action is precisely given by $\bar{a}^o|_{B^o} - \bar{a}^{\tilde{o}}|_{B^{\tilde{o}}}$. Restricted to the i-th C_2-group (recall that all border color classes have automorphism group C_2^ℓ for some $\ell \in \{0,1,2\}$), both ψ^o are the identity if $\bar{a}^o(u) - \bar{a}^{\tilde{o}}(u) = 0$ for some (and thus every) $u \in B^o_i$. If otherwise $\bar{a}^o(u) - \bar{a}^{\tilde{o}}(u) = 1$, then both are the nontrivial element of C_2. Now let $\bar{a}'^o$ be another solution of $\mathcal{S}^o$. It again encodes an isomorphism $\varphi'^o\colon \mathfrak{A}^o_D \to \mathsf{can}(\mathfrak{A}^o_D) \upharpoonright \tau$. Then $\varphi'^{\tilde{o}} := \varphi'^o \circ \psi^{\tilde{o}}\colon \mathfrak{A}^{\tilde{o}}_D \to \mathsf{can}(\mathfrak{A}^{\tilde{o}}_D)$ is an isomorphism that is encoded by a solution $\bar{a}^{\tilde{o}}$ of $\mathcal{S}^{\tilde{o}}$. It satisfies

$$(\varphi'^{\tilde{o}})^{-1} \circ \varphi'^o = (\varphi'^o \circ \psi^{\tilde{o}})^{-1} \circ \varphi'^o = (\psi^{\tilde{o}})^{-1} = \psi^o.$$

Hence, we have that

$$\bar{a}^o|_{B^o} - \bar{a}^{\tilde{o}}|_{B^{\tilde{o}}} = \bar{a}'^o|_{B^o} - \bar{a}'^{\tilde{o}}|_{B^{\tilde{o}}}.$$

Note that $(\bar{a}^o - \bar{a}^{\tilde{o}})(u) = (\bar{a}^o - \bar{a}^{\tilde{o}})(v)$ for all $u, v \in B^o_i$ and every $i \in [k]$ because whenever $u \neq v$, the cyclic constraints imply $\bar{a}^o(u) \neq \bar{a}^o(v) = \bar{a}^o(u) + 1$ and so

$$\left(\bar{a}^o - \bar{a}^{\tilde{o}}\right)(v) = \bar{a}^o(u) + 1 - (\bar{a}^{\tilde{o}}(u) + 1) = \left(\bar{a}^o - \bar{a}^{\tilde{o}}\right)(u)$$

(recall that we are working in $\mathbb{Z}_2$). Hence, the desired tuples $\bar{b}^o$ correspond to a tuple in $\mathbb{Z}_2^k$ because $\bar{b}^o_i$ contains either only zeros or only ones. To define the $\bar{b}^o$ in CPT, we want to choose the lexicographically minimal one. This cannot be done straightforwardly, because k is not bounded.

We construct the tuples $\bar{b}^o$ inductively and fix the entries for one variable set B_j^o per step. Assume we have defined $(\bar{b}_1^o, \dots, \bar{b}_{j-1}^o)$ such that there is a solution $\bar{a}^o|_{B^o} \in L(\mathcal{S}^o)$ for every $o \in \mathbb{O}$ such that

$$\left(\bar{b}_1^o, \dots, \bar{b}_{j-1}^o\right) = \bar{a}^o|_{B_1^o \cup \dots \cup B_{j-1}^o} - \bar{a}^{\tilde{o}}|_{B_1^{\tilde{o}} \cup \dots \cup B_{j-1}^{\tilde{o}}}.$$

To define $\bar{b}_j^o$, we want to solve the equation system

$$\begin{aligned} \bar{a}^\uparrow &\in L(\mathcal{S}^\uparrow) & & \\ \bar{a}^\downarrow &\in L(\mathcal{S}^\downarrow) & & \\ \bar{a}^\uparrow(u) - \bar{a}^\downarrow(u) &= \bar{b}_i^\uparrow(u) & & i \in [j], u \in B_i \\ \bar{a}^\downarrow(u) - \bar{a}^\uparrow(u) &= \bar{b}_i^\downarrow(u) & & i \in [j], u \in B_i \end{aligned}$$

for the two possible values of $\bar{b}_j^o$. Equivalently, we consider the two possible values of $\bar{b}_j^{\tilde{o}}$: Note that

$$\bar{b}_i^\uparrow(u) = \bar{a}^\uparrow(u) - \bar{a}^\downarrow(u) = \bar{a}^\downarrow(u) - \bar{a}^\uparrow(u) = \bar{b}_i^\downarrow(u)$$

because we are working in $\mathbb{Z}_2$. But the equations above do not define a series of TCESs because there is no order between the variables $B_i^\uparrow$ and $B_i^\downarrow$. Hence, we solve four equations systems, two for each possible value of $\bar{b}_i^o$ and two for each possible order setting $B_i^\uparrow < B_i^\downarrow$ or vice versa (for which the equation system has obviously the same solutions).

The equation system is solvable for at least one possible value of $\bar{b}_i^o$ because, by the induction hypothesis, there are solutions $\bar{a}^o|_{B^o} \in L(\mathcal{S}^o)$ such that $\bar{b}_i^o = (\bar{a}^o - \bar{a}^{\tilde{o}})|_{B_i^o}$ for every $i < j$. Thus, additionally setting $\bar{b}_j^o = (\bar{a}^o - \bar{a}^{\tilde{o}})|_{B_j^o}$ is a solution for the equation system above. If the system is solvable for both possible values of $\bar{a}^o$, we choose $\bar{b}_i^o(u) = 1$. The enlarged tuple $(\bar{b}_1^o, \dots, \bar{b}_j^o)$ satisfies the induction hypothesis, too, because by construction there are solutions $\bar{a}^o \in L(\mathcal{S}^o)$ that satisfy $(\bar{b}_1^o, \dots, \bar{b}_j^o) = \bar{a}^o|_{B_1^o \cup \dots \cup B_j^o} - \bar{a}^{\tilde{o}}|_{B_1^{\tilde{o}} \cup \dots \cup B_j^{\tilde{o}}}$ (as part of the solution of the equation system above). □

We now use the tuples $\bar{b}^o$ to represent the canonical labelings of the border color classes, which additionally extend to canonical labelings of the reflection component, as a TCES.

Lemma 4.93. *There is a* CPT*-term that on input $\mathcal{T}$ and D defines a series of weakly global TCESs $\mathcal{T}_D$ with the following properties:*

1. *B is contained in the topmost variables of $\mathcal{T}_D$.*
2. *$\mathcal{T}_D$ encodes the set $\mathsf{Iso}(\mathfrak{A}_D, \mathsf{can}(\mathfrak{A}_D))|_{B(D)} = \Lambda^\uparrow \cup \Lambda^\downarrow$.*
3. *The size of $\mathcal{T}_D$ is polynomial in $|D|$.*

Proof. Let $\bar{b}^o$ be the two tuples given by Lemma 4.92. We define a set of two variables $B_\alpha := \{\alpha^\uparrow, \alpha^\downarrow\}$ (and set $\alpha^o := A_D^o$), $V_D := B \cup B_\alpha \cup V^\uparrow \cup V^\downarrow$, and $\preceq_D$ such that it respects the orders on B and V^o and satisfies $B \prec B_\alpha \prec V^o$ for all $o \in \mathbb{O}$. The variable sets $V^\uparrow$ and $V^\downarrow$ of the CESs $\mathcal{S}^\uparrow$ and $\mathcal{S}^\downarrow$, respectively, are incomparable.

We want to define a TCES $\mathcal{T}_D$ such that $\bar{c} \in L(\mathcal{T}_D)$ if and only if then there is an $o \in \mathbb{O}$ and a $\bar{a}^o \in L(\mathcal{S}^o)$ such that $\bar{c} = \bar{a}^o|_B$. To do so, we guess two solutions $\bar{a}^o \in L(\mathcal{S}^o)$ (one for each $o \in \mathbb{O}$) with the property that $\bar{a}^o|_B + \bar{b}^o = \bar{a}^{\tilde{o}}|_B$ (Lemma 4.92). Then we want to ensure that $\bar{c} = \bar{a}^{\uparrow}|_B$ or $\bar{c} = \bar{a}^{\downarrow}|_B$. To allow that one equality does not hold, we use the additional variables $\alpha^{\uparrow}$ to express the constraints $\bar{c} = \bar{a}^o|_B + \alpha^o \cdot \bar{b}^o$. By enforcing that exactly one of $\alpha^{\uparrow}$ and $\alpha^{\downarrow}$ is 1, we obtain the desired system. Finally, to make the system linear, we encode the multiplication $\alpha^o \cdot \bar{b}^o$. This is possible because $\bar{b}^o$ does not depend on $\bar{a}^o$ and can be defined before defining the following TCES:

$$\begin{aligned}
\bar{a}^{\uparrow} &\in L(\mathcal{S}^{\uparrow}) & & \\
\bar{a}^{\downarrow} &\in L(\mathcal{S}^{\downarrow}) & & \\
\bar{c}(u) &= \bar{a}^{\uparrow}(u) = \bar{a}^{\downarrow}(u) & & \text{if } \bar{b}^{\uparrow}(u) = \bar{b}^{\downarrow}(u) = 0, u \in B \\
\bar{c}(u) &= \bar{a}^{\uparrow}(u) + \alpha^{\uparrow} = \bar{a}^{\downarrow}(u) + \alpha^{\downarrow} & & \text{if } \bar{b}^{\uparrow}(u) = \bar{b}^{\downarrow}(u) = 1, u \in B \\
1 &= \alpha^{\uparrow} + \alpha^{\downarrow} & &
\end{aligned}$$

where $\bar{a}^o$ is indexed by V^o and $\bar{c}$ is indexed by B and ranges over $\mathbb{Z}_2$. If the variable α^o is assigned to 1, then $\bar{c} = \bar{a}^o|_{B^o} + \bar{b}^o$ and otherwise $\bar{c} = \bar{a}^o|_{B^o}$. Because of the cyclic constraint $1 = \alpha^{\uparrow} + \alpha^{\downarrow}$, we add the tuple $\bar{b}^o$ to a solution $\bar{a}^o$ of $\mathcal{S}^o$ for exactly one orientation $o \in \mathbb{O}$.

Clearly, $\mathcal{T}_D$ has topmost variables B and, because the size of the $\mathcal{S}^o$ is polynomial $|D|$, so is the size of $\mathcal{T}_D$. We argue that $\mathcal{T}_D$ is weakly global: The only global equations are the equations relating $\bar{c}(u)$ and $\bar{a}^o(u)$. These variables and so all global variables are over $\mathbb{Z}_2$ (embedded in $\mathbb{Z}_{2^\ell}$ by equations $2u = 0$, as discussed earlier). It suffices to show that $\mathcal{T}_D$ encodes the set

$$\mathsf{Iso}(\mathfrak{A}_D, \mathsf{can}(\mathfrak{A}_D))|_{B(D)} = \bigcup_{o \in \mathbb{O}} \mathsf{Iso}(\mathfrak{A}_D^o, \mathsf{can}(\mathfrak{A}_D^o))|_{B(D)}$$

by Lemma 4.87. So we have to show that

$$L(\mathcal{T}_D)|_B = L(\mathcal{S}^{\uparrow})|_{B^{\uparrow}} \cup L(\mathcal{S}^{\downarrow})|_{B^{\downarrow}}$$

because $\mathcal{S}^o$ encodes $\mathsf{Iso}(\mathfrak{A}_D^o, \mathsf{can}(\mathfrak{A}_D^o))$. Let $\bar{c} \in L(\mathcal{T}_D)|_B$, so there is a solution consisting of $\bar{c}, \alpha^{\uparrow}, \alpha^{\downarrow}, \bar{a}^{\uparrow}$, and $\bar{a}^{\downarrow}$ of $\mathcal{T}_D$ with $\alpha^o = 0$ for some $o \in \mathbb{O}$ (which must be the case by the cyclic constraint on $\alpha^{\uparrow}$ and $\alpha^{\downarrow}$). In particular, we have $\bar{a}^o|_{B^o} = \bar{c}$ and $\bar{a}^o \in L(\mathcal{S}^o)$. Thus, $\bar{c} \in L(\mathcal{S}^o)|_{B^o}$.

For the reverse direction, let $\bar{a}^o \in L(\mathcal{S}^o)$ for some $o \in \mathbb{O}$. Then by Lemma 4.92 there is a solution $\bar{a}^{\tilde{o}} \in L(\mathcal{S}^o)$ such that $\bar{a}^o|_{B^o} - \bar{a}^{\tilde{o}}|_{B^{\tilde{o}}} = \bar{b}^o$. So $\bar{c} = \bar{a}^o|_{B^o}$, $\alpha^o = 0$, $\alpha^{\tilde{o}} = 1$, $\bar{a}^o$, and $\bar{a}^{\tilde{o}}$ form a solution of $\mathcal{T}_D$, in particular, $\bar{c} = \bar{a}^o|_{B^o} \in L(\mathcal{T}_D)|_B$. □

Now, we finally defined all operation on TCESs needed for our canonization and conclude:

Theorem 4.94. *Canonization of q-bounded dihedral double-CFI-free 2-injective quotient τ-structures is* CPT*-definable for every $q \in \mathbb{N}$ and every signature τ.*

Proof. By Lemma 4.91, we indeed are able to define a canon if we can encode the sets Λ_i in CPT. The discussion after Lemma 4.91 and Lemma 4.93 show that the sets can be encoded by CPT-definable series of TCESs. □

Theorem 4.95. *For every $q \in \mathbb{N}$ and every relational signature τ, canonization of the following classes of structures is* CPT*-definable:*

1. *q-bounded ternary relational τ-structures with odd dihedral or cyclic colors and*

2. *q-bounded binary τ-structures with dihedral or cyclic colors.*

Proof. The claim follows from Lemma 4.70 and Theorem 4.94. □

From the Immerman-Vardi Theorem (Theorem 2.1) it follows that CPT captures PTIME on these classes:

Corollary 4.96. *For every $q \in \mathbb{N}$ and every relational signature τ,* CPT *captures* PTIME *on the class of q-bounded ternary τ-structures with odd dihedral or cyclic colors and on the class of q-bounded binary τ-structures with dihedral or cyclic colors.*

4.7 Discussion

We separated a relational structure into 2-injective subdirect products and quotients, gave a classification of all 2-injective subdirect products of dihedral and cyclic groups, and used this classification to canonize relational structures with bounded dihedral colors of arity at most 3. We showed that the structure decomposes into reflection components and that in these components either all color classes have to be reflected or none. If we exclude a single 2-injective subdirect product, namely the double CFI group, the reflection components can only have abelian dependencies. This is always true for binary structures because the said group cannot be realized by binary structures with dihedral colors. In fact, we demonstrated the increase of complexity when considering structures of arity 3 instead of 2. Apart from the fact that the double CFI group does not appear, a classification of 1-injective 2-factor subdirect products of dihedral groups is much easier. Considering higher arity, already 3-injective 4-factor subdirect products of dihedral groups cannot be classified to be (almost) abelian or reflect-or-rotate groups. If one instead tries to reduce the arity of the structures, one needs not only to work with a class of groups closed under taking quotients and subgroups (which is the case for dihedral and cyclic groups), but also closed under taking direct products.

One natural way to exclude the double CFI group is a restriction to odd dihedral colors. The difficulty with even dihedral groups might indicate that looking at odd (non-dihedral) groups could be a reasonable next step. A natural graph class with odd automorphism groups are tournaments. Since such groups are solvable, there is hope for an inductive approach exploiting the abelian case. It could be possible that the techniques developed in this chapter transfer to this case. Just like dihedral groups, odd groups are closed under taking quotients and subgroups. However, they are also closed under direct products (and are solvable), which would allow a reduction of the arity. Thus, it is possible to apply our reduction to quotients and 2-injective groups. As a next step, one could try to follow a similar strategy as for dihedral colors: Identify components of the structure, in which the complexity of all color classes decreases simultaneously, when a single color class is made easier (similar to reflection components). This might not immediately result in abelian groups, but recursion on the complexity of the groups could be a reasonable option, e.g., on the length of the composition series or on the nilpotency class.

Chapter 5

Choiceless Polynomial Time with Witnessed Symmetric Choice

We now address the question of isomorphism testing versus canonization in logics and in particular in CPT. That is, we consider the question whether a definable isomorphism test implies a definable canonization. Grohe, Schweitzer, and Wiebking [59] introduced the Deep Weisfeiler Leman computation model (DeepWL) to show that a CPT-definable isomorphism test implies a CPT-definable *complete invariant*. A complete invariant is similar to a canonization: For every structure, a complete invariant defines an ordered object such that the ordered objects obtained for two structures are equal if and only if the two structures are isomorphic. The crucial difference to a canonization is that the object defined by the complete invariant is not an isomorphic (and ordered) copy of the input structure but just some ordered object. So in the setting of a complete invariant, the Immerman-Vardi Theorem cannot be exploited to capture PTIME. In an algorithmic context, Gurevich [62] showed how to turn a complete invariant into a canonization algorithm. The only requirement is that the considered class of structures is closed under individualization, so intuitively under coloring some atoms with unique colors. Gurevich's algorithm uses the complete invariant to compute a canonical orbit of the input structure, chooses an arbitrary atom in that orbit, and individualizes it. This process is continued until all atoms are individualized and thus a total order is computed. Because the algorithm only chooses from orbits, the resulting ordered object is indeed equal for all choices.

However, it is not clear whether Gurevich's algorithm can be implemented in CPT because it is based on making choices. To implement Gurevich's algorithm, we consider the extension of CPT with witnessed symmetric choice called CPT+WSC. *Witnessed symmetric choice* allows for arbitrary choices from *definable orbits*: One defines an orbit in the logic, from which an arbitrary element is chosen. The restriction to choose from orbits ensures isomorphism-invariance of the logic. To evaluate such a logic, we need to check whether the defined choice-sets are indeed orbits. Because it is unknown whether orbits can be computed in polynomial-time, the logic has to additionally provide witnessing automorphisms. That is, for every choice-set and all elements a and b in this choice-set, the logic has to define an automorphism mapping a to b. In this way, checking whether the choice-sets are indeed orbits becomes easy and the logic can be evaluated in polynomial-time.

In that fashion, Gurevich's canonization algorithm becomes CPT+WSC-definable because the algorithm indeed repeatedly defines and chooses from orbits. However, we have to extend the algorithm to provide witnessing automorphisms. It will turn out that Gurevich's algorithm can be exploited again to produce these witnessing automorphisms. So we can turn a CPT+WSC-definable complete invariant into a CPT+WSC-definable canonization. However, [59] only shows that a CPT-definable isomorphism test implies a CPT-definable complete invariant (and thus, by the prior reasoning, a CPT+WSC-definable canonization). Actually, we are interested in a logic in which isomorphism testing implies canonization. So we extend the results of [59] from CPT to CPT+WSC and indeed prove that for CPT+WSC isomorphism testing implies canonization and thus also implies that CPT+WSC captures PTIME.

Related Work. IFP extended with symmetric choice (IFP+SC), that is, with choices from definable orbits but without providing witnessing automorphisms, was first studied by Gire and Hoang [42]. Symmetric choice is integrated into a fixed-point operator to allow for repeated choices. The authors also considered witnessed symmetric choice (IFP+WSC) where certifying automorphisms have to be provided. As discussed before, these witnessing automorphisms are required to ensure polynomial-time evaluation. Dawar and Richerby [31] provided a more sophisticated definition of the semantics of this fixed-point operator with symmetric choice. We essentially follow this approach to integrate witnessed symmetric choice into CPT. We review further existing work on integrating different concepts of choice into fixed-point logic at the beginning of Chapter 6, which considers extensions of IFPC with witnessed symmetric choice.

As already discussed in Chapter 1, CPT defines a canonization for padded structures [19], bounded and abelian structures [118], and (some) bounded and dihedral structures as seen in Chapter 4. All these approaches are somehow orthogonal to witnessed symmetric choice. They exploit the fact that some set of objects, for which it is not known whether they form orbits, is small enough to try out all possible choices.

Overview of this Chapter. We first extend CPT with a fixed-point operator with witnessed symmetric choice in Section 5.1 and obtain the logic CPT+WSC. We make some small but important formal changes to the fixed-point operators with symmetric choice considered in [31,42]: The formula that defines the witnessing automorphisms has access to the defined fixed-point. This is necessary to exploit Gurevich's algorithm to define witnessing automorphisms. To allow access to the defined fixed-point, we have to put (potentially) stronger conditions on the defined orbits, which we will discuss later in more detail. The defined fixed-points are semideterministic, that is, all fixed-points resulting from different choices are related by an automorphism. These fixed-points have to be turned into an isomorphism-invariant value, to which the WSC-fixed-point operator evaluates. For IFP+WSC, this is achieved using quantifiers [31, 42]. We generalized the usage of quantifiers to arbitrary formulas called output formulas. While in the first-order setting the fixed-point operator can only define relations, in the CPT setting we can of course define arbitrary hereditarily finite sets. For these sets, using output formulas seems more suitable than solely using quantifiers. Equipped with these changes, defining Gurevich's canonization algorithm in CPT+WSC and extending it to provide witnessing automorphisms becomes rather straightforward in Section 5.2. We prove the equivalence between isomorphism testing and canonization in CPT+WSC, apart from the one crucial step, namely the one that definable isomorphism implies a definable complete invariant in CPT+WSC.

This step is shown in Section 5.3. For this, we consider the DeepWL computation model, which is used to show that a CPT-definable isomorphism test implies a CPT-definable complete invariant [59]. We extend DeepWL with witnessed symmetric choice. The proof in [59] is based on a translation of CPT to DeepWL, a normalization procedure in DeepWL yielding the complete invariant, and a translation back into CPT. Unfortunately, it turned out that this normalization procedure cannot be easily adapted to DeepWL with witnessed symmetric choice. At multiple points we have to change small but essential parts of definitions and so cannot reuse many results of [59]. The reason is that DeepWL always processes all elements of a hereditarily finite set in parallel, which is incompatible with choices: We cannot compute with multiple choices (out of the same choice-set) at the same time in the same structure because these choices influence each other. This will force us to nest DeepWL-algorithms to resemble nested fixed-point operators with witnessed symmetric choice. We end with a discussion and open questions in Section 5.4.

5.1 CPT with a Symmetric Choice Operator

We start by extending BGS with a **fixed-point operator with witnessed symmetric choice** (WSC-fixed-point operator). The logic BGS+WSC is the extension of BGS logic by the following operator to construct formulas. If

- $s_{\text{step}}(\bar{z}xy)$ is a BGS+WSC$[\tau]$-term,
- $s_{\text{choice}}(\bar{z}x)$ is a BGS+WSC$[\tau]$-term,
- $s_{\text{wit}}(\bar{z}xy)$ is a BGS+WSC$[\tau]$-term, and
- $\Phi_{\text{out}}(\bar{z}x)$ is a BGS+WSC$[\tau]$-formula,

then

$$\Psi(\bar{z}) = \mathsf{WSC}^*xy.\ \Big(s_{\text{step}}(\bar{z}xy), s_{\text{choice}}(\bar{z}x), s_{\text{wit}}(\bar{z}xy), \Phi_{\text{out}}(\bar{z}x)\Big)$$

is a BGS+WSC$[\tau]$-formula. The free variables of s_{step} and s_{wit} apart from x and y and the free variables of s_{choice} and Φ_{out} apart from x are free in Ψ. In particular, y is only bound in s_{step} and s_{wit}. We call s_{step} the **step term**, s_{choice} the **choice term**, s_{wit} the **witnessing term**, and Φ_{out} the **output formula**. Intuitively, we want to iterate the step term $s_{\text{step}}(x, y)$ until we reach a fixed-point for the set x. However, we choose before each step an element y of the choice-set defined by the choice term $s_{\text{choice}}(x)$. Once a fixed-point is reached, the witnessing term s_{wit} must provide automorphisms for every intermediate step witnessing that we indeed chose from orbits (details later). Finally, Ψ is satisfied if the output formula $\Phi_{\text{out}}(a)$ is satisfied where a is the set computed through iteration with choice. Because we always choose from orbits, $\Phi_{\text{out}}(a)$ is satisfied by some fixed-point a if and only it is satisfied for every possible fixed-point a (details also later). In this way, the evaluation of BGS+WSC-terms and formulas is still deterministic, so does not depend on any choices made in the fixed-point computation.

We remark that here it seems reasonable to allow free variables in iteration terms. Otherwise, they cannot be used in s_{step}, s_{choice}, s_{wit}, and Φ_{out}. While for CPT or BGS it is clear that nested iteration terms, that is, nested fixed-point computations, can be eliminated, this is not clear for BGS+WSC.

An Example. To illustrate the definition, we discuss an example. A *universal vertex* is a vertex adjacent to every other vertex. A graph $G = (V, E)$ is a *threshold* graph if we can reduce it to the empty graph by repeatedly removing a universal or an isolated vertex. We describe a CPT+WSC-sentence that defines the class of threshold graphs (which we only do for illustration as the class of threshold graphs is already IFP-definable). The intuitive idea is the following: The set of vertices that are universal or isolated form an orbit (note that a graph on more than one vertex cannot have a universal and an isolated vertex at the same time). Thus, we use a WSC-fixed-point operator to choose one such vertex, remove it, and repeat this, until no vertex can be removed anymore.

We start with the choice term $s_{\text{choice}}(x)$. For a set $x \subseteq V$ the following term defines the set of vertices that are universal or isolated in $G[V \setminus x]$:

$$\begin{aligned} r &:= \mathsf{Atoms} \setminus x, \\ s_{\text{choice}}(x) &:= \Big\{\, y \;\Big|\; y \in r, \big(\forall z \in r.\ y = z \lor E(y,z)\big) \lor \big(\forall z \in r.\ y = z \lor \neg E(y,z)\big) \Big\}. \end{aligned}$$

The step term $s_{\text{step}}(x, y)$ adds a chosen vertex y to those already removed:

$$s_{\text{step}}(x, y) := x \cup (\{y\} \cap \mathsf{Atoms}).$$

The intersection with Atoms is needed to reach a fixed-point when all vertices are removed from the graph. In that case, s_{choice} defines the empty choice-set, y is the empty set, and the intersection with Atoms prevents that $\emptyset$ is added to x. As certification, the term $t(z, z')$ defines the transposition of z and z' and the witnessing term $s_{\text{wit}}(x)$ collects all transpositions of pairs of vertices in the choice-set:

$$\begin{aligned} t(z, z') &:= \Big\{\, (x, x) \;\Big|\; x \in \mathsf{Atoms} \setminus \{z, z'\} \Big\} \cup \Big\{(z, z'), (z', z)\Big\} \\ s_{\text{wit}}(x) &:= \Big\{\, t(z, z') \;\Big|\; (z, z') \in x^2 \Big\}. \end{aligned}$$

Finally, the output formula $\Phi_{\text{out}}(x) := x = \mathsf{Atoms}$ checks whether all vertices have been removed. Overall, the following formula defines the class of threshold graphs:

$$\mathsf{WSC}^{*} xy.\ (s_{\text{step}}, s_{\text{choice}}, s_{\text{wit}}, \Phi_{\text{out}}).$$

The WSC-fixed-point operator will compute the fixed-point of the variable x starting with $a_1 = \emptyset$ as initial value for x. First, the term $s_{\text{choice}}(a_1)$ is evaluated to define the first choice-set b_1 of all universal or isolated vertices of G. One such vertex $u_1 \in b_1$ is chosen and the step term $s_{\text{step}}(a_1, \{u_1\})$ is evaluated yielding $a_2 = \{u_1\}$.

Now inductively assume that a_i contains all vertices removed so far. Then b_i is the set of all universal or isolated vertices of $G - a_i$. The set b_i is now an orbit of (G, a_i) (in fact, an orbit of $(G, a_1, \ldots, a_i)$). So again, a vertex $u_i \in b_i$ is chosen and added by s_{step} to a_i yielding the set $a_{i+1} = a_i \cup \{u_i\}$. Assume the case that b_i is empty in the end. Nothing is chosen and the step term is evaluated and yields $s_{\text{step}}(a_i, \emptyset) = a_i$, so a fixed-point is reached. The output formula defines whether it was possible to remove all vertices.

Finally, the WSC-fixed-point operator evaluates the witnessing term to certify that indeed all choices where made from orbits. As argued before, all choice-sets are indeed orbits. The term $s_{\text{wit}}(b_i)$ outputs, for every b_i, a set of automorphisms, that for all $u, v \in b_i$, contains an automorphism mapping u to v. Hence, it is certified that the choice-sets indeed are orbits.

5.1.1 Semantics of Symmetric Choice Operators

We now define the precise semantics of the WSC-fixed-point operator. We consider evaluation of WSC-fixed-point operators for arbitrary isomorphism-invariant functions in place of the choice, step, and witnessing terms. This makes the definition independent of the semantics of CPT and we can reuse it later in Section 5.3. For a finite set of atoms A, an automorphism of an $\mathsf{HF}(A)$-set a is a permutation φ of A such that $\varphi(a) = a$ by applying φ to every occurrence of an atom in a. Formally, for every $b \in \mathsf{HF}(A)$, we define

$$\varphi(b) := \begin{cases} \varphi(b) & \text{if } b \in A \\ \{\varphi(c_1), \dots, \varphi(c_k)\} & \text{if otherwise } b = \{c_1, \dots, c_k\}. \end{cases}$$

Definition 5.1 (Isomorphism-Invariant Function). For a structure $\mathfrak{A}$ and a tuple $\bar{a} \in \mathsf{HF}(A)^*$, a function $f \colon \mathsf{HF}(A)^k \to \mathsf{HF}(A)$ is called **$(\mathfrak{A}, \bar{a})$-isomorphism-invariant** if for every automorphism $\varphi \in \mathsf{Aut}((\mathfrak{A}, \bar{a}))$ and every $\bar{b} \in \mathsf{HF}(A)^k$ we have $f(\varphi(\bar{b})) = \varphi(f(\bar{b}))$.

Definition 5.2 (Witnessing an Orbit). For a τ-structure $\mathfrak{A}$ and a tuple $\bar{a} \in \mathsf{HF}(A)^*$, a set M **witnesses** an $\mathsf{HF}(A)$-set N as orbit of $(\mathfrak{A}, \bar{a})$ if $M \subseteq \mathsf{Aut}((\mathfrak{A}, \bar{a}))$ and, for all $b, c \in N$, there is a $\varphi \in M$ satisfying $\varphi(b) = c$.

Note that this definition in principle also allows witnessing proper subsets of orbits. However, the sets N of interest in the following will always be given by an isomorphism-invariant function and so N can never be a proper subset of an orbit. Now fix an arbitrary τ-structure $\mathfrak{A}$ and a tuple $\bar{a} \in \mathsf{HF}(A)^*$. Let $f^{\mathfrak{A},\bar{a}}_{\text{step}}, f^{\mathfrak{A},\bar{a}}_{\text{wit}} \colon \mathsf{HF}(A) \times \mathsf{HF}(A) \to \mathsf{HF}(A)$ and $f^{\mathfrak{A},\bar{a}}_{\text{choice}} \colon \mathsf{HF}(A) \to \mathsf{HF}(A)$ be $(\mathfrak{A}, \bar{a})$-isomorphism-invariant functions. We define the (possibly infinite) unique least rooted tree $\mathcal{T}(f^{\mathfrak{A},\bar{a}}_{\text{step}}, f^{\mathfrak{A},\bar{a}}_{\text{choice}})$ whose vertices are labeled with $\mathsf{HF}(A)$-sets (so two nodes in the tree can have the same label) and which satisfies the following:

(a) The root is labeled with $\emptyset$.

(b) A vertex labeled with $b \in \mathsf{HF}(A)$ has for every $c \in f^{\mathfrak{A}}_{\text{choice}}(b)$ a child labeled with $f^{\mathfrak{A}}_{\text{step}}(b, c)$.

Let $\mathcal{P}(f^{\mathfrak{A},\bar{a}}_{\text{step}}, f^{\mathfrak{A},\bar{a}}_{\text{choice}})$ be the set of tuples $p = (b_1, \dots, b_n)$ of $\mathsf{HF}(A)$-sets such that $n \geq 2$, $b_1 = \emptyset$, $b_{n-1} = b_n$, $b_{i-1} \neq b_i$ for all $1 < i < n$, there is a path of length n in $\mathcal{T}(f^{\mathfrak{A},\bar{a}}_{\text{step}}, f^{\mathfrak{A},\bar{a}}_{\text{choice}})$ starting at the root, and the i-th vertex in the path is labeled with b_i for all $i \in [n]$. The tree $\mathcal{T}(f^{\mathfrak{A},\bar{a}}_{\text{step}}, f^{\mathfrak{A},\bar{a}}_{\text{choice}})$ models the computation for all possible choices and $\mathcal{P}(f^{\mathfrak{A},\bar{a}}_{\text{step}}, f^{\mathfrak{A},\bar{a}}_{\text{choice}})$ is the set of all possible labels yielding a fixed-point. For the sake of readability, we call the elements of $\mathcal{P}(f^{\mathfrak{A},\bar{a}}_{\text{step}}, f^{\mathfrak{A},\bar{a}}_{\text{choice}})$ also paths.

We say that the function $f_{\text{wit}}^{\mathfrak{A},\bar{a}}$ **witnesses** a path $p = (b_1, \dots, b_n) \in \mathcal{P}(f_{\text{step}}^{\mathfrak{A},\bar{a}}, f_{\text{choice}}^{\mathfrak{A},\bar{a}})$ if, for every $i \in [n-1]$, it holds that $f_{\text{wit}}^{\mathfrak{A},\bar{a}}(b_n, b_i)$ witnesses $f_{\text{choice}}^{\mathfrak{A},\bar{a}}(b_i)$ as an $(\mathfrak{A}, \bar{a}, b_1, \dots, b_i)$-orbit. Finally, we define

$$\mathsf{WSC}^*(f_{\text{step}}^{\mathfrak{A},\bar{a}}, f_{\text{choice}}^{\mathfrak{A},\bar{a}}, f_{\text{wit}}^{\mathfrak{A},\bar{a}}) := \left\{ b_n \;\middle|\; (b_1, \dots, b_n) \in \mathcal{P}(f_{\text{step}}^{\mathfrak{A},\bar{a}}, f_{\text{choice}}^{\mathfrak{A},\bar{a}}) \right\}$$

if $f_{\text{wit}}^{\mathfrak{A},\bar{a}}$ witnesses all paths in $\mathcal{P}(f_{\text{step}}^{\mathfrak{A},\bar{a}}, f_{\text{choice}}^{\mathfrak{A},\bar{a}})$ and $\mathsf{WSC}^*(f_{\text{step}}^{\mathfrak{A},\bar{a}}, f_{\text{choice}}^{\mathfrak{A},\bar{a}}, f_{\text{wit}}^{\mathfrak{A},\bar{a}}) := \emptyset$ otherwise.

Lemma 5.3. *If $f_{wit}^{\mathfrak{A},\bar{a}}$ witnesses some path in $\mathcal{P}(f_{step}^{\mathfrak{A},\bar{a}}, f_{choice}^{\mathfrak{A},\bar{a}})$, then $\mathcal{P}(f_{step}^{\mathfrak{A},\bar{a}}, f_{choice}^{\mathfrak{A},\bar{a}})$ is an orbit of $(\mathfrak{A}, \bar{a})$.*

Proof. Let $p^* = (b_1^*, \dots, b_k^*) \in \mathcal{P}(f_{\text{step}}^{\mathfrak{A},\bar{a}}, f_{\text{choice}}^{\mathfrak{A},\bar{a}})$ be a witnessed path, let P_i be the set of prefixes of length i of the paths in $\mathcal{P}(f_{\text{step}}^{\mathfrak{A},\bar{a}}, f_{\text{choice}}^{\mathfrak{A},\bar{a}})$, and let p_i^* be the prefix of length i of p^*. We prove by induction on i that P_i is an orbit. For the root $\emptyset$, the claim trivially holds. We show that $p = (b_1, \dots, b_{i+1})$ is in P_{i+1} if and only if there is an automorphism $\varphi \in \mathsf{Aut}((\mathfrak{A}, \bar{a}))$ such that $\varphi(p_{i+1}^*) = p$. First, assume that $p \in P_{i+1}$. Let p_i be the prefix of length i of p. By the induction hypothesis, there is an automorphism $\varphi \in \mathsf{Aut}((\mathfrak{A}, \bar{a}))$ such that $\varphi(p_i^*) = p_i$. By definition of $\mathcal{T}(f_{\text{step}}^{\mathfrak{A},\bar{a}}, f_{\text{choice}}^{\mathfrak{A},\bar{a}})$, for some $c_i^* \in f_{\text{choice}}^{\mathfrak{A},\bar{a}}(b_i^*)$ and some $c_i \in f_{\text{choice}}^{\mathfrak{A},\bar{a}}(b_i)$ it holds that

$$b_{i+1}^* = f_{\text{step}}^{\mathfrak{A},\bar{a}}(b_i^*, c_i^*) \text{ and } b_{i+1} = f_{\text{step}}^{\mathfrak{A},\bar{a}}(b_i, c_i).$$

Because $f_{\text{choice}}^{\mathfrak{A},\bar{a}}$ is isomorphism-invariant, we have that $f_{\text{choice}}^{\mathfrak{A},\bar{a}}(b_i) = \varphi(f_{\text{choice}}^{\mathfrak{A},\bar{a}}(b_i^*))$. Because p^* is witnessed, $f_{\text{choice}}^{\mathfrak{A},\bar{a}}(b_i^*)$ is an orbit of $(\mathfrak{A}, \bar{a}p_i^*)$ and so $f_{\text{choice}}^{\mathfrak{A},\bar{a}}(b_i)$ is an orbit of $(\mathfrak{A}, \bar{a}p_i) = \varphi((\mathfrak{A}, \bar{a}p_i^*))$. It follows that $\varphi(c_i^*)$ and c_i are in the same orbit of $(\mathfrak{A}, \bar{a}p_i)$. So let $\psi \in \mathsf{Aut}((\mathfrak{A}, \bar{a}p_i))$ such that $\psi(\varphi(c_i^*)) = c_i$. We now have

$$\psi(\varphi(b_{i+1}^*)) = \psi(\varphi(f_{\text{step}}^{\mathfrak{A},\bar{a}}(b_i^*, c_i^*))) = f_{\text{step}}^{\mathfrak{A},\bar{a}}(\psi(\varphi(b_i^*)), \psi(\varphi(c_i^*))) = f_{\text{step}}^{\mathfrak{A},\bar{a}}(b_i, c_i) = b_{i+1}$$

because $f_{\text{step}}^{\mathfrak{A},\bar{a}}$ is isomorphism-invariant. Because ψ stabilizes $b_1, \dots, b_i$, we finally have that $(\psi \circ \varphi)(p_{i+1}^*) = p$.

Second, assume that $\varphi \in \mathsf{Aut}((\mathfrak{A}, \bar{a}))$ such that $p = \varphi(p_i^*)$. Then by the induction hypothesis, $p_i = \varphi(p_i^*) \in P_i$. As before, let $c_i^* \in f_{\text{choice}}^{\mathfrak{A},\bar{a}}(b_i^*)$ such that $b_{i+1}^* = f_{\text{step}}^{\mathfrak{A},\bar{a}}(b_i^*, c_i^*)$. Because $f_{\text{choice}}^{\mathfrak{A},\bar{a}}$ is isomorphism-invariant, it holds that $\varphi(c_{i+1}^*) \in f_{\text{choice}}^{\mathfrak{A},\bar{a}}(\varphi(b_i^*), \varphi(c_i^*))$. Because $f_{\text{step}}^{\mathfrak{A},\bar{a}}$ is isomorphism-invariant, it holds that

$$b_{i+1} = \varphi(b_{i+1}^*) = \varphi(f_{\text{step}}^{\mathfrak{A},\bar{a}}(b_i^*, c_{i+1}^*)) = f_{\text{step}}^{\mathfrak{A},\bar{a}}(\varphi(b_i^*), \varphi(c_{i+1}^*)) = f_{\text{step}}^{\mathfrak{A},\bar{a}}(b_i, \varphi(c_i^*).$$

That is, the vertex corresponding to b_i (when following the tree $\mathcal{T}(f_{\text{step}}^{\mathfrak{A},\bar{a}}, f_{\text{choice}}^{\mathfrak{A},\bar{a}})$ starting at the root) has a child labeled with b_{i+1} and hence $p \in P_{i+1}$. □

Corollary 5.4. *The function $f_{wit}^{\mathfrak{A},\bar{a}}$ either witnesses all paths in $\mathcal{P}(f_{step}^{\mathfrak{A},\bar{a}}, f_{choice}^{\mathfrak{A},\bar{a}})$ or none of them.*

Proof. If there is a witnessed path $p^* = (b_1^*, \dots, b_k^*)$ in $\mathcal{P}(f_{\text{step}}^{\mathfrak{A},\bar{a}}, f_{\text{choice}}^{\mathfrak{A},\bar{a}})$, then the set $\mathcal{P}(f_{\text{step}}^{\mathfrak{A},\bar{a}}, f_{\text{choice}}^{\mathfrak{A},\bar{a}})$ is an orbit. The claim follows because $f_{\text{choice}}^{\mathfrak{A},\bar{a}}$ is isomorphism-invariant:

So for every other path p, there is a $\varphi \in \mathsf{Aut}((\mathfrak{A}, \bar{a}))$ such that $p = \varphi(p^*)$ and, in particular, $p = (b_1, \dots, b_k)$ (i.e., p and p^* have the same length). Let $i \in [n-1]$. We have

$$\begin{aligned} f_{\mathrm{wit}}^{\mathfrak{A},\bar{a}}(b_n, b_i) = f_{\mathrm{wit}}^{\mathfrak{A},\bar{a}}(\varphi(b_n^*), \varphi(b_i^*)) = \varphi(f_{\mathrm{wit}}^{\mathfrak{A},\bar{a}}(b_n^*, b_i^*)) &\subseteq \varphi(\mathsf{Aut}((\mathfrak{A}, \bar{a}, b_1^*, \dots, b_i^*))) \\ &= \mathsf{Aut}(\mathfrak{A}, \bar{a}, b_1, \dots, b_i). \end{aligned}$$

Assume $d, e \in f_{\mathrm{choice}}^{\mathfrak{A},\bar{a}}(b_i, c_i)$. Then again, because $f_{\mathrm{choice}}^{\mathfrak{A},\bar{a}}$ is isomorphism-invariant, we have $\varphi^{-1}(d), \varphi^{-1}(e) \in f_{\mathrm{choice}}^{\mathfrak{A},\bar{a}}(b_i^*, c_i^*)$. Because p^* is witnessed, there is a $\psi \in f_{\mathrm{wit}}^{\mathfrak{A},\bar{a}}(b_n^*, c_i^*)$ such that $\psi(\varphi^{-1}(d)) = \varphi^{-1}(e)$. Hence, $\psi \circ \varphi$ is contained in $f_{\mathrm{choice}}^{\mathfrak{A},\bar{a}}(b_i, c_i)$ and maps d to e. □

Corollary 5.5. $\mathsf{WSC}^*(f_{step}^{\mathfrak{A},\bar{a}}, f_{choice}^{\mathfrak{A},\bar{a}}, f_{wit}^{\mathfrak{A},\bar{a}})$ *is an* $(\mathfrak{A}, \bar{a})$*-orbit.*

Proof. Assume that there is a non-witnessed path (or no path) in $\mathcal{P}(f_{\mathrm{step}}^{\mathfrak{A},\bar{a}}, f_{\mathrm{choice}}^{\mathfrak{A},\bar{a}})$. In this case, $\mathsf{WSC}^*(f_{\mathrm{step}}^{\mathfrak{A},\bar{a}}, f_{\mathrm{choice}}^{\mathfrak{A},\bar{a}}, f_{\mathrm{wit}}^{\mathfrak{A},\bar{a}}) = \emptyset$ trivially satisfies the claim. Otherwise, there is a witnessed path, $\mathcal{P}(f_{\mathrm{step}}^{\mathfrak{A},\bar{a}}, f_{\mathrm{choice}}^{\mathfrak{A},\bar{a}})$ is an orbit, all paths have the same length, and $\mathsf{WSC}^*(f_{\mathrm{step}}^{\mathfrak{A},\bar{a}}, f_{\mathrm{choice}}^{\mathfrak{A},\bar{a}}, f_{\mathrm{wit}}^{\mathfrak{A},\bar{a}})$ is the set of all sets b that are the last entry of some path in $\mathcal{P}(f_{\mathrm{step}}^{\mathfrak{A},\bar{a}}, f_{\mathrm{choice}}^{\mathfrak{A},\bar{a}})$. Because $\mathcal{P}(f_{\mathrm{step}}^{\mathfrak{A},\bar{a}}, f_{\mathrm{choice}}^{\mathfrak{A},\bar{a}})$ is an orbit, in particular, the last vertices in every root-to-leaf-path (which are necessarily at the same depth) form an orbit and the claim follows. □

Now we define the semantics of the WSC-fixed-point operator: For a BGS+WCS-term s with free variables $x_1, \dots, x_k$ and a tuple $\bar{a} \in \mathsf{HF}(A)^\ell$ for some $\ell \leq k$, we write $s^{\mathfrak{A}}(\bar{a})$ for the "partial" application of the function $s^{\mathfrak{A}}$, that is, for the function $\mathsf{HF}(A)^{k-\ell} \to \mathsf{HF}(A)$ defined by $\bar{b} \mapsto s^{\mathfrak{A}}(\bar{a}\bar{b})$. Let s_{step} and s_{wit} be BGS+WSC-terms with free variables $\bar{z}xy$, s_{choice} be a BGS+WSC-term with free variables $\bar{z}x$, and Φ_{out} be a BGS+WSC-formula with free variables $\bar{z}x$. We define

$$\begin{aligned} &\Big(\mathsf{WSC}^*xy.\ (s_{\mathrm{step}}, s_{\mathrm{choice}}, s_{\mathrm{wit}}, \Phi_{\mathrm{out}})\Big)^{\mathfrak{A}} := \\ &\qquad \Big\{ \bar{a} \in \mathsf{HF}(A)^{|\bar{z}|} \ \Big|\ \bar{a}b \in \Phi_{\mathrm{out}}^{\mathfrak{A}} \text{ for all } b \in \mathsf{WSC}^*(s_{\mathrm{step}}^{\mathfrak{A}}(\bar{a}), s_{\mathrm{choice}}^{\mathfrak{A}}(\bar{a}), s_{\mathrm{wit}}^{\mathfrak{A}}(\bar{a})) \Big\}. \end{aligned}$$

In [31], the fixed-point operator with symmetric choice is not evaluated on $\mathfrak{A}$, but on the reduct $\mathfrak{A} \upharpoonright \tau'$, where $\tau' \subseteq \tau$ is the subset of relations of the τ-structure $\mathfrak{A}$ used in the fixed-point operator. This ensures that adding unused relations to structures does not change the result of a formula (the additional relations potentially change the orbits of the structure and choices cannot be witnessed anymore), which is a desirable property of a logic [37]. We do not use the "reduct semantics" in this chapter. We could in principle use it but then Section 5.3 would get even more technical without providing further insights. In Chapter 6, which considers extensions of IFPC with witnessed symmetric choice, we consider the reduct semantics in more detail.

Failure on Non-Witnessed Choices. While the semantics defined as above results in a reasonable logic, we want a special treatment of the case when choices cannot be witnessed. Whenever during the evaluation of a formula, there is a path in $\mathcal{P}(f_{\mathrm{step}}^{\mathfrak{A},\bar{a}}, f_{\mathrm{choice}}^{\mathfrak{A},\bar{a}})$ that is not witnessed, we abort evaluation and output an error to indicate there was a non-witnessed choice. Formally, we extend the evaluation by an error-marker †. For a structure $\mathfrak{A}$, evaluating a term s becomes a function $s^{\mathfrak{A}}\colon \mathsf{HF}(A)^k \to \mathsf{HF}(A) \cup \{\dagger\}$ and evaluating a formula Φ becomes a function $\Phi^{\mathfrak{A}}\colon \mathsf{HF}(A)^k \to \{\top, \bot, \dagger\}$. Whenever a $\dagger$ occurs, it is just propagated. We omit the formal definitions here. Later, we will see that the error marker is necessary to guarantee polynomial-time evaluation.

Stabilizing Intermediate Steps. We defined the evaluation of choice terms similar to [31] using the tree $\mathcal{T}(f^{\mathfrak{A},\bar{a}}_{\text{step}}, f^{\mathfrak{A},\bar{a}}_{\text{choice}})$. However, our definition is different in one crucial aspect: In the setting of Lemma 5.3, we require that $f_{\text{choice}}(b_i)$ defines an orbit of $(\mathfrak{A}, \bar{a}, b_1, \dots, b_i)$, where in [31] an orbit of $(\mathfrak{A}, \bar{a}, b_i, c_i)$ is required. That is, in BGS+WSC one has to respect in some sense all choices made in previous intermediate steps during the fixed-point computation. This is crucial to prove Lemma 5.3. This is not required in [31] because the authors only need that the vertices in $\mathcal{T}(f^{\mathfrak{A},\bar{a}}_{\text{step}}, f^{\mathfrak{A},\bar{a}}_{\text{choice}})$ on the same level are in the same orbit. We actually need that the paths in the tree in their entirety form an orbit to establish Corollary 5.4. Due to this corollary, we can give the witnessing term access to b_n when witnessing $(\mathfrak{A}, \bar{a}, b_1, \dots, b_i)$-orbits. Accessing the defined fixed-point to witness intermediate choice-sets will become crucial in the following, namely to define Gurevich's algorithm in Section 5.2. We do not require that the actual chosen elements c_i are fixed by the automorphisms because, in contrast to [31], the choice term only gets the b_i as input and not the c_i. So if c_i and c_i' result in the same next intermediate step b_{i+1}, the subsequent computation will be the same for both choices. For the very same reason and again in contrast to [31], it is sufficient to label the vertices in the tree $\mathcal{T}(f^{\mathfrak{A},\bar{a}}_{\text{step}}, f^{\mathfrak{A},\bar{a}}_{\text{choice}})$ only with the intermediate steps b_i and not additionally with the chosen elements.

5.1.2 CPT+WSC

Similarly to how CPT is obtained from BGS, we obtain **Choiceless Polynomial Time with witnessed symmetric choice** (CPT+WSC) by enforcing polynomial bounds on BGS+WSC terms and formulas: A CPT+WSC-term (or formula, respectively) is a pair $(s, p(n))$ (or $(\Phi, p(n))$, respectively) of a BGS+WSC-term (or BGS+WSC-formula, respectively) and a polynomial. For BGS-operators, we add the same restrictions as in CPT. For a WSC-fixed-point operator $\mathsf{WSC}^*xy.\,(s_{\text{step}}, s_{\text{choice}}, s_{\text{wit}}, \Phi_{\text{out}})$, a structure $\mathfrak{A}$, and a tuple $\bar{a} \in \mathsf{HF}(A)^*$, we restrict $\mathcal{P}(f^{\mathfrak{A},\bar{a}}_{\text{step}}, f^{\mathfrak{A},\bar{a}}_{\text{choice}})$ to paths $(b_1, \dots, b_k)$ of length $k \leq p(|A|)$ for which $|\mathsf{TC}(b_i)| \leq p(|A|)$ for all $i \in [k]$. If there is a path in $\mathcal{P}(f^{\mathfrak{A},\bar{a}}_{\text{step}}, f^{\mathfrak{A},\bar{a}}_{\text{choice}})$ of length greater than $p(|A|)$ or in some path there is a set not bounded by p, then the WSC-fixed-point operator evaluates to $\dagger$.

It is important that $|\mathsf{WSC}^*(s^{\mathfrak{A}}_{\text{step}}(\bar{a}), s^{\mathfrak{A}}_{\text{choice}}(\bar{a}), s^{\mathfrak{A}}_{\text{wit}}(\bar{a}))|$ is not required to be bounded by $p(|A|)$. In fact, using WSC-fixed-point operators only makes sense if the set is allowed to be of superpolynomial size, as otherwise we could define it with a regular iteration term. It is also important to output $\dagger$ and not $\emptyset$ when the polynomial bound is exceeded because in that case we cannot validate whether all choice-sets are orbits (and so it might depend on the choices whether the bound is exceeded or not). To evaluate the witnessing term we need access to the fixed-point, which cannot be computed if the polynomial bound is exceeded. Because the WSC-fixed-point operator can only choose from orbits, CPT+WSC is isomorphism-invariant:

Lemma 5.6. *For every structure $\mathfrak{A}$, every* CPT+WSC*-term s, and every* CPT+WSC*-formula Φ, the evaluating functions $s^{\mathfrak{A}}$ and $\Phi^{\mathfrak{A}}$ are unions of $\mathfrak{A}$-orbits.*

Proof. The proof is straightforward by structural induction on terms and formulas using Corollary 5.5. □

While Lemma 5.6 only concerns automorphisms, it is easy to see that CPT+WSC also respects isomorphism between different structures. Using multiple structures would make Section 5.1.1 formally more complicated without providing new insights. Using Lemma 5.6 we can show that model checking for CPT+WSC can be done in polynomial time. Formulas and terms cannot be evaluated naively because, as we have seen earlier, the sets $\mathsf{WSC}^*(s^{\mathfrak{A}}_{\text{step}}(\bar{a}), s^{\mathfrak{A}}_{\text{choice}}(\bar{a}), s^{\mathfrak{A}}_{\text{wit}}(\bar{a}))$ are possibly not of polynomial size.

Lemma 5.7. *For every* CPT+WSC *term $(s, p(n))$ or formula $(\Phi, p(n))$, we can compute in polynomial time on input $\mathfrak{A}$ and $\bar{a} \in \mathsf{HF}(A)^k$ the set $s^{\mathfrak{A}}(\bar{a})$ or the truth-value $\Phi^{\mathfrak{A}}(\bar{a})$, respectively.*

Proof. The proof is by structural induction on terms and formulas. We show that $s^{\mathfrak{A}}(\bar{a})$ or $s^{\Phi}(\bar{a})$ can be computed in polynomial time for every tuple $\bar{a} \in \mathsf{HF}(A)$ of suitable length. Assume by the induction hypothesis that CPT+WSC-terms s and t and formulas Φ and Ψ can be evaluated in polynomial time. We can surely evaluate comprehension terms, iterations terms, and all formulas composed of s, t, Φ, and Ψ apart from WSC-fixed-point operators in polynomial time. So we have to consider a WSC-fixed-point operator

$$\mathsf{WSC}^* xy.\ (s_{\text{step}}, s_{\text{choice}}, s_{\text{wit}}, \Phi_{\text{out}}).$$

Because $W := \mathsf{WSC}^*(s^{\mathfrak{A}}_{\text{step}}(\bar{a}), s^{\mathfrak{A}}_{\text{choice}}(\bar{a}), s^{\mathfrak{A}}_{\text{wit}}(\bar{a}))$ is an orbit of $(\mathfrak{A}, \bar{a})$ by Corollary 5.5, it suffices to compute one $b \in W$ (or determine that none exists) and check whether $\bar{a}b \in \Phi^{\mathfrak{A}}_{\text{out}}$ by Lemma 5.6. Given b, the check can be done in polynomial time by the induction hypothesis. Some $b \in W$ can be computed by iteratively evaluating s_{choice} to define a choice-set, selecting one arbitrary element out of it, and then evaluating s_{step} with this choice until either a fixed-point b is reached or more than $p(|A|)$ iterations are performed. In the later case output †. If this is not the case, then we check whether s_{wit} witnesses the computed path. If the path is not witnessed, then we abort with output † and otherwise by Corollary 5.4 we computed one $b \in W$. If we have to construct a set c with $|\mathsf{TC}(c)| > p(|A|)$ at any point during the evaluation, then output † as well. If we always chose from orbits, then we would have constructed such an excessively large set c for all possible choices. Otherwise, if some choice-set would not be an orbit we would fail to witness the orbits and output † as well. We need to evaluate s_{step}, s_{choice}, and s_{wit} at most $p(|A|)$ many times, so computing b is also done in polynomial time. □

5.1.3 Defining Sets

The WSC-fixed-point operator can only output truth-values. These are, by design, isomorphism-invariant. We now discuss alternatives: Let $f\colon \mathsf{HF}(A)^k \to \mathsf{HF}(A)$ be some function that we want to define with an iteration term with choice (the domain is the set of possible values for the parameters). To obtain a logic with well-defined semantics, we need that $\varphi(f(\bar{a})) = f(\bar{a})$ for every $\varphi \in \mathsf{Aut}((\mathfrak{A}, \bar{a}))$. This clearly holds if f only returns truth-values (e.g., encoded by $\emptyset$ and $\{\emptyset\}$). We do not know how to decide in polynomial time whether the condition $\varphi(f(\bar{a})) = f(\bar{a})$ is satisfied for all $\varphi \in \mathsf{Aut}((\mathfrak{A}, \bar{a}))$ during the evaluation. So we consider functions $f\colon \mathsf{HF}(A)^k \to \mathsf{HF}(\emptyset)$, which generalizes the case of truth-values but still is syntactically isomorphism-invariant. We define an iteration term for this case:

$$\mathsf{WSC}^* xy.\ (s_{\text{step}}, s_{\text{choice}}, s_{\text{wit}}, s_{\text{out}}),$$

where s_{step}, s_{choice}, s_{wit}, and s_{out} are BGS+WSC-terms. The only difference to the iteration term with choice seen so far is that the output formula is replaced with an **output term**. Let $W = \mathsf{WSC}^*(s^{\mathfrak{A}}_{\text{step}}(\bar{a}), s^{\mathfrak{A}}_{\text{choice}}(\bar{a}), s^{\mathfrak{A}}_{\text{wit}}(\bar{a}))$ to define the evaluation as follows:

$$\begin{aligned}&\left(\mathsf{WSC}^* xy.\ (s_{\text{step}}, s_{\text{choice}}, s_{\text{wit}}, s_{\text{out}})\right)^{\mathfrak{A}}(\bar{a})\\ :={}& \begin{cases} \bigcup_{b \in W} s^{\mathfrak{A}}_{\text{out}}(\bar{a}b) & \text{if } s^{\mathfrak{A}}_{\text{out}}(\bar{a}b) \in \mathsf{HF}(\emptyset) \text{ for all } b \in W,\\ \emptyset & \text{otherwise.}\end{cases}\end{aligned}$$

If $s^{\mathfrak{A}}_{\text{out}}(\bar{a}b) \in \mathsf{HF}(\emptyset)$ for all $b \in W$, then $s^{\mathfrak{A}}_{\text{out}}(\bar{a}b) = s^{\mathfrak{A}}_{\text{out}}(\bar{a}b')$ for all $b, b' \in W$. If $W \neq \emptyset$, then the iteration term evaluates to $s^{\mathfrak{A}}_{\text{out}}(\bar{a}b)$ for some and thus every $b \in W$. Otherwise, it evaluates to $\emptyset$. We now show that the extended WSC-fixed-point operator does not increase the expressive power of CPT+WSC:

Lemma 5.8. *For every signature τ and for all* CPT$[\tau]$*-terms s_{step}, s_{choice}, s_{wit}, and s_{out}, there is a* CPT$[\tau]$*-term t such that, for all τ-structures $\mathfrak{A}$, it holds that*

$$\left(\mathsf{WSC}^* xy.\ (s_{step}, s_{choice}, s_{wit}, s_{out})\right)^{\mathfrak{A}} = t^{\mathfrak{A}}.$$

Proof. Let $p(n)$ be the polynomial bound of the CPT+WSC-term and $n = |A|$. Let a be a set constructed during the evaluation. Because of the polynomial bound, we have $|\mathsf{TC}(a)| \leq p(n)$. The set a corresponds to a directed acyclic graph (DAG), where the leaves are either atoms or $\emptyset$. By the condition $|\mathsf{TC}(a)| \leq p(n)$, the DAG has at most $p(n)$ many vertices. Note that $\mathsf{HF}(\emptyset)$ can be totally ordered in CPT. In particular, if $a \in \mathsf{HF}(\emptyset)$, then the DAG corresponding to a can be totally ordered. Given the DAG, we can reconstruct a in CPT. Let $t_{\text{toDAG}}(z)$ be a CPT-term, which, given a set $a \in \mathsf{HF}(\emptyset)$, outputs the totally ordered DAG corresponding to a (that is, we can assume that its vertex set is $[|\mathsf{TC}(a)|]$) as a set containing the edges of the DAG. If $a \notin \mathsf{HF}(\emptyset)$, then t_{toDAG} outputs $\emptyset$. Furthermore, let $t_{\text{fromDAG}}(z)$ be the term recovering a from this set. First,

$$\Phi := \mathsf{WSC}^* xy.\ \left(s_{\text{step}}, s_{\text{choice}}, s_{\text{wit}}, (s_{\text{out}} \neq \emptyset) \Rightarrow (t_{\text{toDAG}}(s_{\text{out}}) \neq \emptyset)\right)$$

defines whether s_{out} only outputs $\mathsf{HF}(\emptyset)$-sets. Second,

$$r := \left\{ (i,j) \;\middle|\; i,j \in [p(|\mathsf{Atoms}|)], \mathsf{WSC}^* xy.\ (s_{\text{step}}, s_{\text{choice}}, s_{\text{wit}}, (i,j) \in t_{\text{toDAG}}(s_{\text{out}})) \right\}$$

defines the DAG given by t_{toDAG} (with possible some isolated vertices, which can easily be ignored). Last, $\mathsf{Unique}(\{t_{\text{fromDAG}}(r) \mid \Phi\})$ is equivalent to $\mathsf{WSC}^* xy.\ (s_{\text{step}}, s_{\text{choice}}, s_{\text{wit}}, s_{\text{out}})$, where we use $p(n)^2$ as new polynomial bound (because we just try all pairs i, j). $\square$

With an easy inductive argument one sees that also nesting the extended iteration terms does not increase the expressive power.

5.2 Canonization in CPT+WSC

In this section, we work with classes of relational τ-structures $\mathcal{K}$. We always assume that these only contain connected structures because we are interested in isomorphism testing and canonization. The case of unconnected structures reduces to connected ones. We now introduce various notions related to defining canonization (Definition 2.3) and distinguishing k-orbits (Definition 2.5).

Definition 5.9 (Definable Isomorphism). A logic L **defines isomorphism** for a class of τ-structures $\mathcal{K}$ if there is an L-sentence Φ such that, for all $\mathfrak{A}, \mathfrak{B} \in \mathcal{K}$, the disjoint union $\mathfrak{A} \uplus \mathfrak{B}$ satisfies Φ if and only if $\mathfrak{A} \cong \mathfrak{B}$.

In the case that L is CPT+WSC, we require in the previous definition that Φ never outputs $\dagger$. In all following definitions, we also require without further mentioning that $\dagger$ never occurs for any input.

Definition 5.10 (Complete Invariant). For $L \in \{\text{CPT}, \text{CPT+WSC}\}$, an **$L$-definable complete invariant** of a class of τ-structures $\mathcal{K}$ is a closed L-term s which satisfies the following: $s^{\mathfrak{A}} = s^{\mathfrak{B}}$ if and only if $\mathfrak{A} \cong \mathfrak{B}$ for all $\mathfrak{A}, \mathfrak{B} \in \mathcal{K}$.

To show that a CPT+WSC-definable complete invariant implies a CPT+WSC-definable canonization, we need not only to consider a class of τ-structures $\mathcal{K}$, but all pairs $(\mathfrak{A}, \bar{u})$ for $\mathfrak{A} \in \mathcal{K}$ and $\bar{u} \in A^*$. Intuitively, we consider all possible individualizations of some atoms of $\mathfrak{A}$, i.e., assigning each atom in $\bar{u}$ a unique fresh color. In that sense, we want to work with a class of structures closed under individualization. Some formulas we are going to define in fact iteratively individualize atoms. So it will be convenient to capture the individualized atoms by CPT+WSC "internal" tuples, that is, many formulas will have a free variable ι to which we can pass a tuple (encoded using sets) containing the tuple of individualized atoms. Thus, instead of assuming that the classes of structures are closed under individualization, we work with the free variable ι to which all possible tuples can be passed and require in our definitions that certain properties hold for all possible tuples passed into ι.

In what follows, we will always assume that tuples do not contain duplicates. Moreover, we will freely switch between the "internal" representation of tuples in CPT and the "external" tuples of individualized atoms whenever needed. For the sake of shorter formulas, we introduce a slightly special concatenation operation for tuples of atoms in CPT+WSC. This operation is shorthand notation for a more complex but uninteresting CPT-term. Let x and y be variables. We write xy for a CPT term satisfying the following equations:

$$\begin{aligned}
(xy)^{\mathfrak{A}}(\bar{u}, v) &= \bar{u}v && \text{for all } \bar{u} \in A^*, v \in A, v \neq u_i \text{ for every } i \leq |\bar{u}|,\\
(xy)^{\mathfrak{A}}(\bar{u}, v) &= \bar{u} && \text{for all } \bar{u} \in A^*, v \in A, v = u_i \text{ for some } i \leq |\bar{u}|,\\
(xy)^{\mathfrak{A}}(\bar{u}, \{v\}) &= (xy)^{\mathfrak{A}}(\bar{u}, v) && \text{for all } \bar{u} \in A^*, v \in A,\\
(xy)^{\mathfrak{A}}(\bar{u}, \emptyset) &= \bar{u} && \text{for all } \bar{u} \in A^*.
\end{aligned}$$

We additionally use the notation xy to concatenate two tuples such that duplicates in the second tuple are removed. We write x_i for the term extracting the i-th position of a tuple or the empty set if i is larger than the length of the tuple (i is encoded as a von Neumann ordinal). Lastly, we extend the notation from variables to arbitrary CPT+WSC-terms s and t and write st for the CPT+WSC-term appending the result of t to the result of s in the way defined above. Clearly, all these terms can be defined in CPT+WSC (or in CPT if s and t are CPT-terms, too).

In the CPT-setting, individualized atoms can be encoded by tuples of unbounded length. But because we need the same notions also in Chapter 6, in which we work in a first-order setting, we encode the individualized atoms by an additional binary relation. We reserve the special relation symbol $\trianglelefteq$ for this relation in this Chapter.

Definition 5.11 (Individualization of Atoms)**.** Let $\mathfrak{A}$ be a relational structure. A binary relation $\trianglelefteq^{\mathfrak{A}} \subseteq A^2$ is an **individualization of** $V \subseteq A$ if $\trianglelefteq^{\mathfrak{A}}$ is a total order on V and $\trianglelefteq^{\mathfrak{A}} \subseteq V^2$. We say that $\trianglelefteq^{\mathfrak{A}}$ is an individualization if it is an individualization of some $V \subseteq A$ and that the atoms in V are **individualized** by $\trianglelefteq^{\mathfrak{A}}$.

Definition 5.12 (Closure under Individualization)**.** Let $\mathcal{K}$ be a class of τ-structures. The class $\mathcal{K}^{\trianglelefteq}$ is the class of $(\tau \cup \{\trianglelefteq\})$-structures such that $(\mathfrak{A}, \trianglelefteq^{\mathfrak{A}}) \in \mathcal{K}^{\trianglelefteq}$ for every $\mathfrak{A} \in \mathcal{K}$ and every individualization $\trianglelefteq^{\mathfrak{A}}$. A class of σ-structures $\mathcal{J}$ is **closed under individualization** if $\mathcal{J} = \mathcal{J}^{\trianglelefteq}$.

The relation $\trianglelefteq^{\mathfrak{A}}$ defines a total order on the individualized atoms, so intuitively it assigns unique colors to these atoms. Instead of $(\mathfrak{A}, \trianglelefteq^{\mathfrak{A}})$, one can think of $(\mathfrak{A}, \bar{u})$ where $u_1 \trianglelefteq^{\mathfrak{A}} \cdots \trianglelefteq^{\mathfrak{A}} u_{|\bar{u}|}$ are the pairwise distinct atoms individualized by $\trianglelefteq^{\mathfrak{A}}$. In this chapter we can represent tuples of arbitrary length by a hereditarily finite set. Hence, for the sake of readability, we continue to work with structures $(\mathfrak{A}, \bar{u})$. We add a free variable ι to formulas defining isomorphism, distinguishing orbits, or defining complete invariants and to interpretations defining a canonization. The free variable ι is used for the tuple of individualized atoms, e.g., in the following two lemmas.

Lemma 5.13. *If an L-term $s(\iota)$ is a complete invariant for a class of τ-structures $\mathcal{K}^{\trianglelefteq}$, then there is a* CPT*-formula $\Phi(x, y)$ such that, for every $\mathfrak{A} \in \mathcal{K}$, the relation $\leq$ defined by $a \leq b$ if $ab \in \Phi^{\mathfrak{A}}$ is a total order on $\{s^{\mathfrak{A}}(\bar{u}) \mid \bar{u} \in A^*\}$.*

Proof. Let $\mathfrak{A} \in \mathcal{K}$ and $\bar{u} \in A^*$ be arbitrary. Because classes of structures are isomorphism-closed, there is another structure $\mathfrak{B} \in \mathcal{K}$ and a tuple $\bar{v} \in B^*$ such that $(\mathfrak{A}, \bar{u}) \cong (\mathfrak{B}, \bar{v})$ and $A \cap B = \emptyset$. Let $\varphi\colon (\mathfrak{A}, \bar{u}) \to (\mathfrak{B}, \bar{v})$ be an isomorphism. Then $s^{\mathfrak{A}}(\bar{u}) = s^{\mathfrak{B}}(\bar{v}) = \varphi(s^{\mathfrak{A}}(\bar{u}))$, but this implies $s^{\mathfrak{A}}(\bar{u}) \in \mathsf{HF}(\emptyset)$ for all $\mathfrak{A} \in \mathcal{K}$ and $\bar{u} \in A^*$. Sets in $\mathsf{HF}(\emptyset)$ can easily be compared and ordered in CPT by a formula $\Phi(x, y)$ as long as there size is polynomial in the input structure. Because all sets in $\{s^{\mathfrak{A}}(\bar{u}) \mid \bar{u} \in A^*\}$ are L-definable, these sets are polynomially sized and $\Phi(x, y)$ indeed defines a total order on $\{s^{\mathfrak{A}}(\bar{u}) \mid \bar{u} \in A^*\}$ as required. □

Lemma 5.14. *If there is an L-definable complete invariant for a class of τ-structures $\mathcal{K}^{\trianglelefteq}$, then $\mathcal{K}^{\trianglelefteq}$ has L-distinguishable k-orbits for every $k \in \mathbb{N}$.*

Proof. Let $k \in \mathbb{N}$, $s_{\text{inv}}(\iota)$ be an L-definable complete invariant for $\mathcal{K}^{\trianglelefteq}$, and let $\Phi_{\text{inv}}(x, y)$ be a CPT-formula defining a total order on the invariant by Lemma 5.13. We define

$$\Phi_{\text{orb}}(\iota, x, y) := \Phi_{\text{inv}}\Big(s_{\text{inv}}(\iota x), s_{\text{inv}}(\iota y)\Big).$$

The L-formula $\Phi_{\text{orb}}(\iota, x, y)$ orders two k-tuples $\bar{u}$ and $\bar{v}$ according to the order of $\Phi_{\text{inv}}(x, y)$ on the complete invariants when individualizing $\bar{u}$ and $\bar{v}$, respectively. So Φ_{orb} is a total preorder. For a structure $\mathfrak{A} \in \mathcal{K}$ and a tuple $\bar{w} \in A^*$, two k-tuples $\bar{u}, \bar{v} \in A^k$ are in the same k-orbit of $(\mathfrak{A}, \bar{w})$ if and only if $(\mathfrak{A}, \bar{w}\bar{u}) \cong (\mathfrak{A}, \bar{w}\bar{v})$. This is the case if and only if $s^{\mathfrak{A}}_{\text{inv}}(\bar{w}\bar{u}) = s^{\mathfrak{A}}_{\text{inv}}(\bar{w}\bar{v})$, that is, Φ_{orb} defines and orders the k-orbits of $(\mathfrak{A}, \bar{w})$. $\square$

Before we show that defining isomorphism and defining canonization are equivalent in CPT+WSC, we need the following rather technical notion of defining certain 1-orbits, which will simplify the following proofs.

Definition 5.15 (Ready for Individualization). A class of τ-structures $\mathcal{K}^{\trianglelefteq}$ is **ready for individualization in a logic** L if there is an L-sentence Φ defining, for every $\mathfrak{A} \in \mathcal{K}^{\trianglelefteq}$, a set of atoms $O = \Phi^{\mathfrak{A}}$ such that

(a) O is a 1-orbit of $\mathfrak{A}$, i.e., $O \in \mathsf{orb}(\mathfrak{A})$ and

(b) if there is a 1-orbit disjoint with the $\trianglelefteq$-individualized atoms, then O is disjoint with the $\trianglelefteq$-individualized atoms.

Of course, we will use the additional variable ι in the CPT-setting also for this definition.

We now show that Gurevich's algorithm [62], which turns a complete invariant into a canonization, is CPT+WSC-definable. Intuitively, we iteratively individualize an atom of a nontrivial orbit which is minimal according to an isomorphism-invariant total order on the orbits. We continue this procedure, until all atoms are individualized and thereby defined a total order on the atoms (see Figure 5.1 for an example). While the order itself is not unique, the isomorphism type of the ordered structure is unique, because we always chose from orbits. This way, we obtain the canon by renaming the atoms to be just numbers. We now define this approach formally: Let $\mathcal{K}^{\trianglelefteq}$ be a class of τ-structures ready for individualization in CPT+WSC and let $s_{\text{orb}}(\iota)$ be the corresponding CPT+WSC-term defining a 1-orbit with the required properties.

For every $\mathfrak{A} \in \mathcal{K}$ and every $\bar{u} \in A^*$, we define a set $\mathsf{labels}(\mathfrak{A}, \bar{u})$ as follows: If all atoms are individualized, i.e., every atom is contained in $\bar{u}$, then we set

$$\mathsf{labels}(\mathfrak{A}, \bar{u}) := \{\bar{u}\}.$$

Otherwise, let $O = s^{\mathfrak{A}}_{\text{orb}}(\bar{u})$ be the 1-orbit given by s_{orb}. In particular, this orbit is disjoint with $\bar{u}$. We define

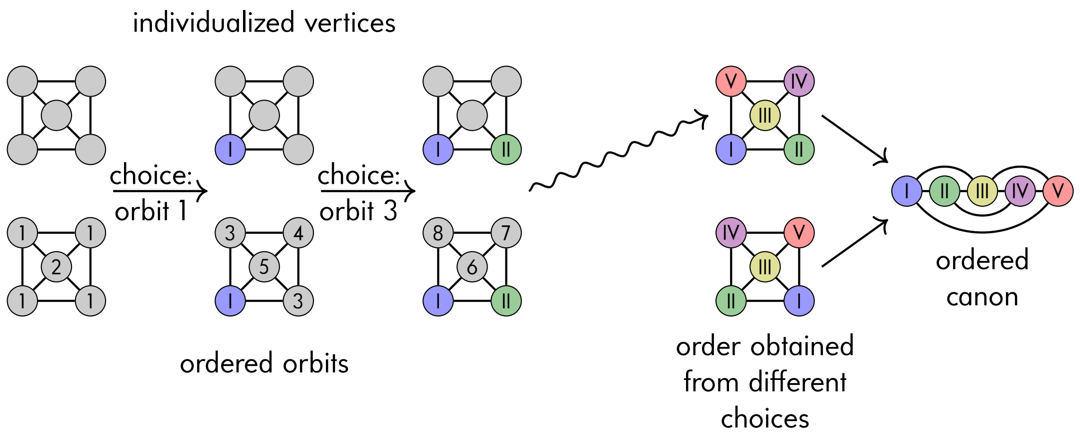
$$\mathsf{labels}(\mathfrak{A}, \bar{u}) := \bigcup_{w \in O} \mathsf{labels}(\mathfrak{A}, \bar{u}w).$$

For $\bar{w} \in \mathsf{labels}(\mathfrak{A}, \bar{u})$, let $\varphi_{\bar{w}} \colon A \to [|A|]$ be defined via $v \mapsto i$ if and only if $v = w_i$. It is easy to see that $\mathsf{labels}(\mathfrak{A}, \bar{u})$ is an $(\mathfrak{A}, \bar{u})$-orbit. Hence, the definition

$$\mathsf{canon}(\mathfrak{A}, \bar{u}) := \varphi_{\bar{w}}((\mathfrak{A}, \bar{u}))$$

is well-defined and independent of the choice of the atom $\bar{w} \in \mathsf{labels}(\mathfrak{A}, \bar{u})$ because $\varphi_{\bar{w}}((\mathfrak{A}, \bar{u})) = \varphi_{\bar{w}'}((\mathfrak{A}, \bar{u}))$ for all $\bar{w}, \bar{w}' \in \mathsf{labels}(\mathfrak{A}, \bar{u})$.

Lemma 5.16. *For all $\mathfrak{A}, \mathfrak{B} \in \mathcal{K}$, all $\bar{u} \in A^*$, and all $\bar{v} \in B^*$, we have* $\mathsf{canon}(\mathfrak{A}, \bar{u}) \cong (\mathfrak{A}, \bar{u})$ *and* $\mathsf{canon}(\mathfrak{A}, \bar{u}) = \mathsf{canon}(\mathfrak{B}, \bar{b})$ *if and only if* $(\mathfrak{A}, \bar{u}) \cong (\mathfrak{B}, \bar{b})$.



5.1 Gurevich's canonization algorithm on an example graph. The top row shows the sequence of graphs with individualized vertices (shown by Roman numerals) in the algorithm. The bottom row shows the orbit partition of the graphs (vertices with the same number are in the same orbit). In each step, one vertex of the orbit with minimal number is chosen and individualized. This process yields the total order shown on the top right. Another order obtained from different choices in shown below. Both orders induce to the same canon.

Proof. Let $\mathfrak{A}, \mathfrak{B} \in \mathcal{K}$, $\bar{u} \in A^*$, and $\bar{v} \in B^*$. First, because the $\varphi_{\bar{w}}$ are bijections for every $\bar{w} \in \mathsf{labels}(\mathfrak{A}, \bar{u})$, it follows that $\mathsf{canon}(\mathfrak{A}, \bar{u}) \cong (\mathfrak{A}, \bar{u})$. Second, because the order on the 1-orbits defined by s_{orb} is isomorphism-invariant, $\mathsf{canon}(\mathfrak{A}, \bar{u}) = \mathsf{canon}(\mathfrak{B}, \bar{v})$ if and only if $(\mathfrak{A}, \bar{u}) \cong (\mathfrak{B}, \bar{v})$. This is essentially the argument why Gurevich's canonization algorithm in [62] is correct. $\square$

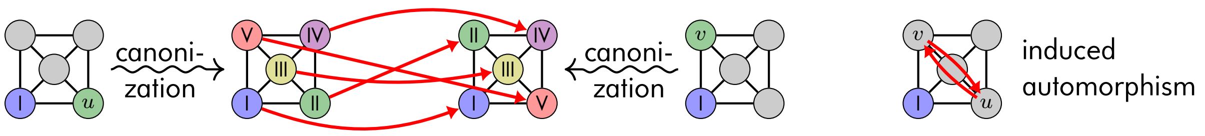
Lemma 5.17. *If a class of τ-structures $\mathcal{K}^{\trianglelefteq}$ is ready for individualization in* CPT+WSC*, then* CPT+WSC *defines a canonization for $\mathcal{K}^{\trianglelefteq}$-structures.*

Proof. To implement the former approach in CPT+WSC, we first introduce some notation. We define a fixed-point operator with deterministic choice, which behaves similarly to the WSC-fixed-point operator, but in which choices are resolved by a total order.

$$\mathsf{DC}^*xy.\ (s_{\text{step}}, s_{\text{choice}}, s_{\text{order}}),$$

where s_{step}, s_{choice}, and s_{order} are CPT+WSC-terms. The first terms $s_{\text{step}}(x, y)$ and $s_{\text{choice}}(x)$ behave exactly as in the symmetric choice operator: s_{step} defines a step function and s_{choice} a choice-set (of atoms). But the third term s_{order} defines a total order on the atoms and is used to resolve the choices deterministically by picking the minimal one. The operator evaluates to the fixed-point obtained in that manner or to $\emptyset$ if the polynomial bound is exceeded. Because all choices are resolved by a total order, this fixed-point operator can be simulated by a plain iteration term.

We define a CPT+WSC-interpretation $\Theta(\iota)$ whose universe is $[|A|]$. The total order $\leq$ is just the natural order on $[|A|]$. For every k-ary relation $E \in \tau$, we define a formula



5.2 Witnessing automorphisms for Gurevich's algorithm. Witnessing automorphisms for the example in Figure 5.1: The goal is to witness that $\{u, v\}$ is an orbit in the graph in which one vertex (I) is individualized. Gurevich's canonization algorithm is used to canonize the two graphs, in which u or v, respectively, is individualized (formally, we use the already defined order to resolve choices in this instance of the algorithm). The obtained total orders induce the witnessing automorphism.

$\Psi_R(\iota, i_1, \dots, i_k)$ as follows:

$$\begin{aligned} t_{\text{label}}(o, \iota) &:= \mathsf{DC}^* xy.\ \big(\iota xy, s_{\text{orb}}(\iota x), o\big), \\ t_{\text{wit}}(o, \iota) &:= \Big\{ \Big\{ (t_{\text{label}}(o, \iota x)_j, t_{\text{label}}(o, \iota y)_j) \;\Big|\; j \in [\mathsf{Card}(\mathsf{Atoms})] \Big\} \;\Big|\; x, y \in s_{\text{orb}}(\iota) \Big\}, \\ \Psi_R(\iota, i_1, \dots, i_k) &:= \mathsf{WSC}^* xy.\ \big(\iota xy, s_{\text{orb}}(\iota x), t_{\text{wit}}(y, x), R(x_{i_1}, \dots, x_{i_k})\big). \end{aligned}$$

Fix an arbitrary structure $\mathfrak{A} \in \mathcal{K}$. Set $A^<$ to be the set of all A-tuples of length $|A|$ containing all atoms exactly once (i.e., the set of all total orders on A). Additionally, fix an arbitrary $\bar{u} \in A^*$.

Claim 1. *If $\bar{w} \in A^<$, then $t^{\mathfrak{A}}_{label}(\bar{w}, \bar{u}) \in \mathsf{labels}(\mathfrak{A}, \bar{u})$.*

Proof. Recall here that the tuple operation ιxy discards duplicates from x and y. In the first iteration, we add $\bar{u}$ and another atom to x (or just $\bar{u}$ if already all atoms are individualized). In every iteration, we choose the minimal atom according to $\bar{w}$ from the orbit given by s_{orb} until all atoms are individualized. Then nothing is added to x anymore and a fixed-point is reached. This process follows exactly the definition of $\mathsf{labels}(\mathfrak{A}, \bar{u})$. ⊣

Claim 2. *If $\bar{w} \in A^<$, then $t^{\mathfrak{A}}_{wit}(\bar{w}, \bar{u})$ witnesses that $s^{\mathfrak{A}}_{orb}(\bar{u})$ is an orbit of $(\mathfrak{A}, \bar{u})$.*

Proof. Assume $O = s^{\mathfrak{A}}_{\text{orb}}(\bar{u})$, $v, v' \in O$, $\bar{w} \in \mathsf{labels}(\mathfrak{A}, \bar{u}v)$, and $\bar{w}' \in \mathsf{labels}(\mathfrak{A}, \bar{u}v')$. Then $\varphi_{\bar{w}}$ is an isomorphism $(\mathfrak{A}, \bar{u}v) \to \mathsf{canon}(\mathfrak{A}, \bar{u}v)$ and an isomorphism $(\mathfrak{A}, \bar{u}) \to \mathsf{canon}(\mathfrak{A}, \bar{u})$ because O is the orbit given by s_{orb}. Likewise, $\varphi_{\bar{w}'}$ is an isomorphism $(\mathfrak{A}, \bar{u}) \to \mathsf{canon}(\mathfrak{A}, \bar{u})$. We have $\varphi^{-1}_{\bar{w}'} \circ \varphi_{\bar{w}} = \{(c_j, d_j) \mid j \in |A|\} \in \mathsf{Aut}((\mathfrak{A}, \bar{u}))$ and $(\varphi^{-1}_{\bar{w}'} \circ \varphi_{\bar{w}})(v) = v'$. By Claim 1, it easy to see that, given some tuple in $A^<$, the term t_{wit} exactly defines such automorphisms for every pair of atoms in the orbit provided by s_{orb} (see Figure 5.2 for an illustration of witnessing automorphisms for the example in Figure 5.1). ⊣

Claim 3. $\mathsf{WSC}^*\big((\iota yz)^{\mathfrak{A}}(\bar{u}), (s_{orb}(\iota y))^{\mathfrak{A}}(\bar{u}), t^{\mathfrak{A}}_{wit}(\bar{u})\big) = \mathsf{labels}(\mathfrak{A}, \bar{u})$.

Proof. As in Claim 1, the WSC-fixed-point operator expresses precisely the definition of $\mathsf{labels}(\mathfrak{A}, \bar{u})$. By Claim 2, all choices are witnessed because every automorphism stabilizing some tuple $\bar{v} \in A^*$ stabilizes all prefixes of $\bar{v}$. Thus, it also stabilizes all tuples defined earlier in the iteration. Because we consider the result under all possible choices, the claim follows. ⊣

Finally, we show that $\Theta(\mathfrak{A}, \bar{u}) \upharpoonright \tau = \mathsf{canon}(\mathfrak{A}, \bar{u})$. By Claim 3, $(\bar{u}, i_1, \dots, i_\ell) \in \Psi_R^{\mathfrak{A}}$ if and only if $(i_1, \dots, i_\ell) \in R^{\varphi_{\bar{w}}(\mathfrak{A}, \bar{u})}$ for some (and thus every) $\bar{w} \in \mathsf{labels}(\mathfrak{A}, \bar{u})$. This is exactly the definition of $\mathsf{canon}(\mathfrak{A}, \bar{u})$. Hence, we obtained a CPT+WSC-definable canonization (Lemma 5.16). □

Lemma 5.18 ([59]). *If* CPT *defines isomorphism of a class of binary τ-structures $\mathcal{K}$ closed under individualization, then there is a* CPT*-term defining a complete invariant for $\mathcal{K}$.*

While this lemma is only for binary structures, it can also be applied to arbitrary structures. Every τ-structure can be encoded by a binary structure using a CPT-interpretation Θ (in fact, an FO-interpretation suffices) such that $\Theta(\mathfrak{A}) \cong \Theta(\mathfrak{B})$ if and only if $\mathfrak{A} \cong \mathfrak{B}$. Given a definable isomorphism test for a class of τ-structures $\mathcal{K}$, we can define an isomorphism test on $\Theta(\mathcal{K})$ and vice versa.

Corollary 5.19. *If* CPT *defines isomorphism of a class of τ-structures $\mathcal{K}$ closed under individualization, then* CPT+WSC *defines canonization of $\mathcal{K}$-structures and captures* PTIME *on $\mathcal{K}$-structures.*

This corollary is asymmetric in the sense that we turn an isomorphism-defining CPT-formula into a canonization-defining CPT+WSC-formula. The next goal is to prove the symmetric version, which starts with an isomorphism-defining CPT+WSC-formula (rather than a CPT-formula). We begin with the following theorem (which, similarly to Lemma 5.18 can also be used for non-binary structures).

Theorem 5.20. *If* CPT+WSC *defines isomorphism of a class of binary τ-structures $\mathcal{K}$, then* CPT+WSC *defines a complete invariant for $\mathcal{K}$-structures.*

The long proof of this theorem is deferred to Section 5.3. Assuming Theorem 5.20 for now, we conclude:

Theorem 5.21. *Let $\mathcal{K}$ be a class of τ-structures closed under individualization. The following are equivalent:*

1. *$\mathcal{K}$ is ready for individualization in* CPT+WSC*.*
2. *$\mathcal{K}$ has* CPT+WSC*-distinguishable* 1*-orbits.*
3. *$\mathcal{K}$ has* CPT+WSC*-distinguishable k-orbits for every $k \in \mathbb{N}$.*
4. CPT+WSC *defines isomorphism of $\mathcal{K}$.*
5. CPT+WSC *defines a complete invariant for $\mathcal{K}$.*
6. CPT+WSC *defines a canonization for $\mathcal{K}$.*

Proof. We show $4 \Rightarrow 5 \Rightarrow 3 \Rightarrow 2 \Rightarrow 1 \Rightarrow 6 \Rightarrow 4$. Theorem 5.20 proves $4 \Rightarrow 5$, Lemma 5.14 proves $5 \Rightarrow 3$, Lemma 5.17 proves $1 \Rightarrow 6$, and $3 \Rightarrow 2$ is trivial. To show $2 \Rightarrow 1$, one can pick the minimal orbit according to the given preorder satisfying the requirement of ready for individualization. Finally, $6 \Rightarrow 4$ is shown by comparing the two canons of the two structures given as the disjoint union. This is done as follows: Let s_{canon} be a closed CPT+WSC-term defining a canonization, that is, it evaluates the interpretation defining the ordered copy as an $\mathsf{HF}(\emptyset)$-set using numbers as atoms. Let s_{cc} be a CPT+WSC-term defining the set of the two connected components of the disjoint union, i.e., the set of the two universes of the structures to test for isomorphism. To evaluate s_{canon} on a single component of the disjoint union, we need to forbid automorphisms exchanging the components (in the case that they are isomorphic), so s_{canon} indeed can ignore one component and just canonize the other. The idea is to add a free variable x to s_{canon} which will hold a set of atoms forming a component. Let $s'_{\text{canon}}(x)$ be the CPT+WSC-term with a free variable x (unused in s_{canon}) obtained from s_{canon} in the following way: We replace every occurrence of Atoms in s_{canon} with x and in every WSC-fixed-point operator we add the tautology $x = x$ in the step, choice, and witnessing term. This way, every WSC-fixed-point operator has x as free variable and, in particular, all witnessing automorphisms need to stabilize the component held by x. Hence, the formula $\mathsf{Card}(\{s'_{\text{canon}}(x) \mid x \in s_{\text{cc}}\}) = 1$ is satisfied if and only if both components of the disjoint union are isomorphic. □

Finally, we can prove that a definable isomorphism test implies a definable canonization in CPT+WSC.

Theorem 5.22. *If* CPT+WSC *defines isomorphism of a class of τ-structures $\mathcal{K}$ closed under individualization, then* CPT+WSC *defines a canonization of $\mathcal{K}$-structures and captures* PTIME *on $\mathcal{K}$-structures.*

Proof. Let $\mathcal{K}$ be a class of τ-structures, for which CPT+WSC defines isomorphism. Canonization of $\mathcal{K}$ is CPT+WSC-definable by Theorem 5.21 for $\mathcal{K}$ and so by the Immerman-Vardi Theorem 2.1 CPT+WSC captures PTIME. □

Corollary 5.23. *If graph isomorphism is in* PTIME, *then* CPT+WSC *defines isomorphism on all structures if and only if* CPT+WSC *captures* PTIME.

5.3 Isomorphism Testing in CPT+WSC

The goal of this section is to prove Theorem 5.20, which states that a CPT+WSC-definable isomorphism test implies a CPT+WSC-definable complete invariant. The proof of Lemma 5.18 in [59], which proves the same statement for CPT, uses the equivalence between CPT and the DeepWL computation model. This model ensures that a Turing machine can only access and modify a relational structure in an isomorphism-invariant way. For DeepWL, the authors of [59] show that on input $\mathfrak{A} \uplus \mathfrak{B}$ a DeepWL-algorithm never needs to "mix" atoms of the two structures. This implies that if there is a DeepWL-algorithm to decide isomorphism, it essentially suffices not to compute on input $\mathfrak{A} \uplus \mathfrak{B}$ but to consider the output of another algorithm on input $\mathfrak{A}$ and on input $\mathfrak{B}$. The run of the Turing machine in the latter DeepWL-algorithm turns out to be a complete invariant if the DeepWL-algorithm decides isomorphism. To prove Theorem 5.20, we extend DeepWL with witnessed symmetric choice and essentially follow the same proof idea albeit with necessary adaptations. For a more elaborate introduction into DeepWL we refer to [59].

In the rest of this section, we assume that all structures are binary relational structures. Moreover, we see all relation symbols as binary strings such that Turing machines with a fixed alphabet can compute with relation symbols.

Coherent Configurations. Before introducing DeepWL, we need some background on coherent configuration. We start with introducing terminology for general binary relational structures, which can be seen as edge-colored graphs. Let $\mathfrak{A}$ be a binary τ-structure. The inverse of a relation $E^{\mathfrak{A}}$ (for some $E \in \tau$) is

$$\left(E^{\mathfrak{A}}\right)^{-1} := \left\{ (v,u) \;\middle|\; (u,v) \in E^{\mathfrak{A}} \right\}.$$

We call E **undirected** if $E^{\mathfrak{A}} = (E^{\mathfrak{A}})^{-1}$ and **directed** otherwise. We use $\{u,v\} \in E^{\mathfrak{A}}$ as notation for $\{(u,v),(v,u)\} \subseteq E^{\mathfrak{A}}$. For $\pi \subseteq \tau$, two atoms $u,v \in A$ are **π-connected** if there is a path from u to v only using edges contained in a relation in π. Similarly, we define π-connected components and strongly π-connected components (π-SCCs).

We now turn to coherent configurations. Let $\mathfrak{H}$ be a binary σ-structure. A relation $R \in \sigma$ is called **diagonal** if $R^{\mathfrak{H}} \subseteq \mathsf{diag}(H) := \{(u,u) \mid u \in H\}$. The structure $\mathfrak{H}$ is a **coherent configuration** if it satisfies the following properties:

1. The σ-relations partition H^2, that is, $\{R^{\mathfrak{H}} \mid R \in \sigma\}$ is a partition of H^2. In particular, all relations $R^{\mathfrak{H}}$ are nonempty.

2. Every relation $R \in \sigma$ is either disjoint from or a subset of $\mathsf{diag}(H)$.

3. Every relation $R \in \sigma$ has an inverse $R^{-1} \in \sigma$, i.e., $(R^{\mathfrak{H}})^{-1} = (R^{-1})^{\mathfrak{H}}$.

4. For every triple $(R,S,T) \in \sigma$, there is a number $q(R,S,T) \in \mathbb{N}$ such that whenever $(u,v) \in R^{\mathfrak{H}}$, there are exactly $q(R,S,T)$ many $w \in H$ such that $(u,w) \in S^{\mathfrak{H}}$ and $(w,v) \in T^{\mathfrak{H}}$.

The number $q(R,S,T)$ is called the **intersection number** of (R,S,T). The function $q\colon \sigma^3 \to \mathbb{N}$ given by $(R,S,T) \mapsto q(R,S,T)$ is called the **intersection function** of $\mathfrak{H}$. The σ-relations are called **colors**. Diagonal σ-relations are called **fibers**. A color $R \in \sigma$ **has an (S,T)-colored triangle** if $q(R,S,T) \geq 1$. This can be extended to paths: A color R **has an $(S_1,\dots,S_k)$-colored path** if, for every $(u,v) \in R^{\mathfrak{A}}$, there is a path $(w_1,\dots,w_{k+1})$ such that $w_1 = u$, $w_{k+1} = v$ and, for every $i \in [k]$, we have $(w_i,w_{i+1}) \in S_i^{\mathfrak{H}}$. By the properties of coherent configuration, this is either the case for every $(u,v) \in R^{\mathfrak{A}}$ or for no such edge. We say that a color has k many colored triangles or colored paths if we want to specify the exact number of these triangles or paths.

The coherent configuration $\mathfrak{H}$ **refines** a τ-structure $\mathfrak{A}$ if $H = A$ and, for every σ-relation R and every τ-relation E, either $R^{\mathfrak{H}} \subseteq E^{\mathfrak{A}}$ or $R^{\mathfrak{H}} \cap E^{\mathfrak{A}} = \emptyset$. A coherent configuration $\mathfrak{H}$ refining a structure $\mathfrak{A}$ is a **coarsest** coherent configuration refining $\mathfrak{A}$ if every coherent configuration $\mathfrak{H}'$ refining $\mathfrak{A}$ also refines $\mathfrak{H}$. Given a τ-structure $\mathfrak{A}$, a coarsest coherent configuration refining $\mathfrak{A}$ can be computed canonically with the two-dimensional Weisfeiler-Leman algorithm. We denote this configuration by $C(\mathfrak{A})$.

Structures with Sets as Vertices. In the following, relational structures in which some "atoms" are obtained as HF-sets of other atoms play an important role. We formalize this as follows: For a signature $\tau = \{E_1,\dots,E_k\}$, a finite binary **τ-HF-structure** $\mathfrak{A}$ is a tuple $(A, A_{HF}, E_1^{\mathfrak{A}},\dots,E_k^{\mathfrak{A}})$, where A is a finite set of atoms, $A_{HF} \in \mathsf{HF}(A) \setminus A$ is a finite set of $\mathsf{HF}(A)$-sets, and $E_i^{\mathfrak{A}} \subseteq (A \cup A_{HF})^2$ for all $i \in [k]$, that is, the universe of $\mathfrak{A}$ is a set of atoms A and some $\mathsf{HF}(A)$-sets A_{HF}. We call A **atoms** and $\mathsf{V}(\mathfrak{A}) := A \cup A_{HF}$ **vertices**.

In that sense, every τ-HF-structure $\mathfrak{A}$ can be turned into a τ-structure $\mathfrak{A}^{\text{flat}}$, where the sets in A_{HF} become fresh atoms. Conversely, every τ-structure is also a τ-HF-structure, where the set A_{HF} is empty.

An automorphism of the τ-HF-structure $\mathfrak{A}$ is a permutation φ of A such that

(a) φ is an automorphism of A_{HF}, that is, $\varphi(A_{HF}) = A_{HF}$, and

(b) $(u, v) \in E_i^{\mathfrak{A}}$ if and only if $\varphi(u, v) \in E_i^{\mathfrak{A}}$ for all $i \in [k]$ and all $u, v \in \mathsf{V}(\mathfrak{A})$.

A τ-HF-structure $\mathfrak{A}$ has potentially fewer automorphisms than the τ-structure $\mathfrak{A}^{\text{flat}}$ because the condition that φ is an automorphism of A_{HF} vanishes for $\mathfrak{A}^{\text{flat}}$. Using this notion of automorphisms, $\mathfrak{A}$-orbits and $(\mathfrak{A}, \bar{a})$-orbits (for a tuple $\bar{a}$ of $\mathsf{HF}(A)$-sets) are defined as before.

The disjoint union of two HF-structures $\mathfrak{A}$ and $\mathfrak{B}$ is the structure $\mathfrak{A} \uplus \mathfrak{B}$ with atom set $A \uplus B$ that is defined as expected. For an HF-structure $\mathfrak{A}$ and a set $M \subseteq \mathsf{V}(\mathfrak{A})$ such that $M \subseteq \mathsf{HF}(A \cap M)$, that is, M contains only HF-sets formed over atoms contained in M, the substructure of $\mathfrak{A}$ induced by M is denoted $\mathfrak{A}[M]$. We define $C(\mathfrak{A}) := C(\mathfrak{A}^{\text{flat}})$, that is, coherent configurations are always computed with respect to $\mathfrak{A}^{\text{flat}}$.

5.3.1 DeepWL

We are going to introduce the notion of a DeepWL-algorithm from [59]: A **DeepWL-algorithm** is a two-tape Turing machine using the alphabet $\{0, 1\}$ with three special states $q_{\texttt{addPair}}$, $q_{\texttt{scc}}$, and $q_{\texttt{create}}$. The first tape is called the **work-tape** and the second one the **interaction-tape**. The Turing machine computes on a binary relational τ-HF-structure $\mathfrak{A}$, but it has no direct access to it. Instead, the structure is put in the so-called "cloud" which maintains the pair $(\mathfrak{A}, C(\mathfrak{A}))$. The Turing machine only has access to the **algebraic sketch** $D(\mathfrak{A}) = (\tau, \sigma, \subseteq_{\tau,\sigma}, q)$, which gets written on the interaction-tape and consists of the following objects:

1. τ is the signature of the HF-structure $\mathfrak{A}$.

2. σ is the signature of the canonical coarsest coherent configuration $C(\mathfrak{A})$ refining $\mathfrak{A}^{\text{flat}}$.

3. $\subseteq_{\tau,\sigma} := \{(R, E) \in \tau \times \sigma \mid R^{C(\mathfrak{A})} \subseteq E^{\mathfrak{A}}\}$ is the **symbolic subset relation**. It relates a σ-color R to the τ-relation E which is refined by σ, i.e., $R^{C(\mathfrak{A})} \subseteq E^{\mathfrak{A}}$.

4. q is the intersection function of $C(\mathfrak{A})$.

In the following and unless stated otherwise, we always use τ for the signature of the HF-structure $\mathfrak{A}$ in the cloud and σ for the signature of $C(\mathfrak{A})$, which we assume to be disjoint from τ. We call relations $R \in \sigma$ **colors** and relations $E \in \tau$ just **relations**. If E (or R) is a diagonal relation, then we identify E (or R, respectively) with the set $\{u \mid (u, u) \in E^{\mathfrak{A}}\}$ and call E a **vertex class** (or R a **fiber**). In the rest of this chapter, we use the letters R, S, and T for colors and the letters E and F for relations. We use the letters C and D for vertex classes and the letters U and V for fibers. Although the cloud contains the pair $(\mathfrak{A}, C(\mathfrak{A}))$, we will just say that $\mathfrak{A}$ is in the cloud and interpret σ-colors R in $\mathfrak{A}$, i.e., just write $R^{\mathfrak{A}}$ for $R^{C(\mathfrak{A})}$.

With the special states $q_{\texttt{addPair}}$, $q_{\texttt{scc}}$, and $q_{\texttt{create}}$, the Turing machine can add vertices to the structure in the cloud in an isomorphism-invariant manner. If $\mathfrak{A}$ is the input structure to the DeepWL-algorithm, then the vertices of the HF-structure in the cloud will be pairs $\langle a, i\rangle$ of an $\mathsf{HF}(A)$-set (or atom) $a \in \mathsf{HF}(A)$ and a number $i \in \mathbb{N}$. The number i is encoded as an $\mathsf{HF}(\emptyset)$-set and $\langle a, i\rangle$ denotes the Kuratowski encoding of pairs[1]). Using the number i, we can create multiple vertices for the same $a \in \mathsf{HF}(A)$ as follows. Whenever i many vertices for the set a exist (possible zero many), we add the vertex $\langle a, i+1\rangle$. So, when describing how vertices are added, we can identify them with their $\mathsf{HF}(A)$-sets and assume that the numbers are picked as described.

Now assume that $\mathfrak{A}$ is the τ-HF-structure in the cloud at some point during the execution of the DeepWL-algorithm. To enter the states $q_{\texttt{addPair}}$ and $q_{\texttt{scc}}$, the Turing machine has to write a single relation symbol $X \in \tau \cup \sigma$ on the interaction-tape. To enter $q_{\texttt{create}}$, a set $\pi \subseteq \sigma$ has to be written on the interaction-tape. We say that the machine **executes** `addPair`(X), `scc`(X), and `create`(π).

(a) `addPair`(X): For every $(u, v) \in X^{\mathfrak{A}}$, a new vertex $\langle u, v\rangle$ is added to the structure (by the former convention, actually a vertex $\langle\langle u, v\rangle, i\rangle$ is added). Additionally, new relations E_{left} and E_{right} are added to τ containing the pairs $(\langle u, v\rangle, u)$ and $(\langle u, v\rangle, v)$, respectively. We call these relations the **component relations**.

(b) `scc`(X): For every strongly X-connected component c, a new vertex c is added (note that c is itself an $\mathsf{HF}(A)$-set). A new **membership relation** symbol E_{in} is added to τ containing the pairs (c, u) for every X-SCC c and every $u \in c$.

(c) `create`(π): A new relation symbol E is added to τ, which is interpreted as the union of all $R \in \pi$.

Whenever new relation symbols have to be picked, we choose the smallest unused one according to the lexicographical order (recall that relation symbols are binary strings). Each of these three operations modifies the HF-structure $\mathfrak{A}$ in the cloud. Afterwards, the coherent configuration $C(\mathfrak{A})$ is recomputed, the new algebraic sketch $D(\mathfrak{A})$ is written onto the interaction-tape, and the Turing machine continues. A DeepWL-algorithm accepts $\mathfrak{A}$ if the head of the work-tape points to a 1 when the Turing machine halts and rejects otherwise. For a more detailed definition and description of a DeepWL-algorithm we refer to [59].

Differences and Equivalence to [59]. Our definition of a DeepWL-algorithm differs at various places from the one given in [59], which we discuss now: We omit the `forget`-operation, which allows the machine to remove a τ-relation from the structure in the cloud. However, [59] proves that this operation is not needed because the algebraic sketch $D(\mathfrak{A}')$ of the structure $\mathfrak{A}'$ obtained from $\mathfrak{A}$ by removing some relations can be computed from the sketch $D(\mathfrak{A})$:

Lemma 5.24 ([59, Lemma 22]). *There is a polynomial-time algorithm that, for every algebraic sketch $D(\mathfrak{A})$ of a binary structure $\mathfrak{A}$, every subset $\tilde{\tau} \subseteq \tau$, and every vertex class $C \in \tau$, computes the algebraic sketch $D(\mathfrak{A}[C^{\mathfrak{A}}] \restriction \tilde{\tau})$.*

[1] While in the previous sections we only used the pair encoding implicitly, here we use the explicit $\langle\cdot,\cdot\rangle$ notation for the sake of readability.

Note that the algorithm from the lemma has no access to $\mathfrak{A}$ but solely to $D(\mathfrak{A})$ and so can be executed by a DeepWL-algorithm without modifying the cloud. So it suffices to remember the set of relations to remove. In particular, for every DeepWL-algorithm deciding isomorphism the equivalent DeepWL-algorithm constructed in Theorem 11 of [59] does not use the `forget`-operation.

Instead of the `scc`-operation a `contract`-operation is given in [59]. The `contract`-operation contracts the vertices of the SCCs and does not create a new one for each SCC. A structure $\mathfrak{A}$ resulting from a `contract`(X) operation can be obtained from the structure $\mathfrak{A}'$ resulting from the `scc`(X)-operation by removing the vertices incident to X. While our operations do not allow removing these vertices, they form a union of fibers and the algebraic sketch $D(\mathfrak{A})$ can be computed from $D(\mathfrak{A}')$ by Lemma 5.24. For polynomial-time DeepWL-algorithms, not removing these vertices only creates a polynomial overhead. The other way around, the `scc`-operation can be simulated by first copying the vertices incident to X (using an `addPair`-operation for the fibers incident to X) and second executing a `contract`-operation on the copies.

Our operations produce new component or membership relations, while in [59] a global component relation is maintained but the added vertices are put in a new vertex class. Surely, one is computable from the other. Our version of DeepWL computes on HF-structures while [59] uses plain relational structures and new vertices are just added as fresh atoms. For now, this does not make a difference because the coherent configuration $C(\mathfrak{A}) = C(\mathfrak{A}^{\text{flat}})$ is computed on $\mathfrak{A}^{\text{flat}}$. However, seeing the vertices as hereditarily finite sets possibly removes automorphisms of the structure. We will now show that with our altered operations the automorphisms of the cloud as HF-structure coincide with the automorphisms of the cloud as plain relational structure. This is one major reason why we use different operations when dealing with symmetric choice: Algebraic sketches can be computed with respect to plain relational structures while still maintaining automorphisms of the HF-structure. This will become crucial later when we will simulate an extension of DeepWL with witnessed symmetric choice in CPT+WSC.

Automorphisms and HF-Structures. The following lemma justifies to compute $C(\mathfrak{A})$ on the structure $\mathfrak{A}^{\text{flat}}$ and that DeepWL does not need to access the HF-structure of $\mathfrak{A}$ if the original input was a τ'-(non-HF)-structure.

Lemma 5.25. *Let $\mathfrak{A}_0$ be a binary relational structure and let $\mathfrak{A}$ be a HF-structure obtained by a DeepWL-algorithm in the cloud on input $\mathfrak{A}_0$. For every automorphism $\varphi \in \mathsf{Aut}(\mathfrak{A}^{\mathit{flat}})$, it holds that $\varphi|_A \in \mathsf{Aut}(\mathfrak{A})$.*

Proof. The vertex set of the HF-structure $\mathfrak{A}$ is the union of the atoms $A = A_0$ and some $\mathsf{HF}(A)$-sets A_{HF}. Every `scc`-operation creates, for every corresponding SCC c, a vertex for the HF-set c and the membership relation coinciding with "$\in$" on the HF-sets. Similarly, every `addPair`-operation introduces (Kuratowski-encoded) pairs and the component relations identifying the single entries in the pairs. That is, the structure of the vertices as HF-sets is encoded by the membership and component relations as a DAG using A as sinks. Because a DeepWL-algorithm only adds but never removes relations, the membership and component relations are still present in $\mathfrak{A}$.

Now let $\varphi \in \mathsf{Aut}(\mathfrak{A}^{\text{flat}})$. To show that $\varphi|_A \in \mathsf{Aut}(\mathfrak{A})$, we have to show that $\varphi|_A$ is a permutation of A that satisfies $\varphi|_A(A_{HF}) = A_{HF}$. We first note that φ cannot map an atom in A to a vertex in A_{HF} because atoms have no outgoing edge of a component relation or of a membership relation while all vertices in A_{HF} have one. Hence, φ setwise stabilizes A and A_{HF}, i.e., $\varphi(A) = A$ and $\varphi(A_{HF}) = A_{HF}$. In particular, $\varphi|_A$ is a permutation of A.

Let $a \in A_{HF}$. The $\mathsf{HF}(A)$-set a is the unique vertex of the DAG representing a via the membership and component relations. So $\varphi(a)$ is the unique vertex of an isomorphic DAG, which we obtain by applying φ to the DAG representing a. Thus, $\varphi(a) = \varphi|_A(a)$ (note that φ permutes $A \cup A_{HF}$ and $\varphi|_A$ permutes A and is applied to the atoms in a). Because, as already seen, $\varphi(A_{HF}) = A_{HF}$, it follows that $\varphi(A_{HF}) = \varphi|_A(A_{HF}) = A_{HF}$. $\square$

This lemma is important when we extend DeepWL with witnessed choice: To compute (HF-set respecting) orbits of $\mathfrak{A}$, it suffices to compute orbits of $\mathfrak{A}^{\text{flat}}$ and so to consider $C(\mathfrak{A}^{\text{flat}})$. Note that the lemma does not hold if the `forget`-operation of [59] is available (that can be used to forget a component or membership relation) or the `scc`-operation is replaced the by `contract`-operation (then also a membership-relation is lost).

5.3.2 DeepWL with Witnessed Symmetric Choices

We extend the DeepWL computation model with witnessed symmetric choice. Here, we need two different notions: We start with DeepWL+WSC-machines and use them afterwards to construct DeepWL+WSC-algorithms. A **DeepWL+WSC-machine** M is a DeepWL-algorithm, whose Turing machine has two additional special states $q_{\texttt{choice}}$ and $q_{\texttt{refine}}$. To enter $q_{\texttt{choice}}$, the machine M has to write a relation symbol $X \in \tau \cup \sigma$ on the interaction-tape. To enter $q_{\texttt{refine}}$, M has to write a relation symbol $X \in \tau \cup \sigma$ and a number $i \in \mathbb{N}$ on the interaction-tape. We say that the machine M executes $\texttt{choice}(X)$ and $\texttt{refine}(X, i)$. The DeepWL+WSC-machine M is **choice-free** if it cannot enter $q_{\texttt{choice}}$ syntactically. That is, $q_{\texttt{choice}}$ is not in the range of the transition function of the underlying Turing machine of M. DeepWL+WSC-algorithms are defined inductively:

Definition 5.26 (DeepWL+WSC-algorithm)**.** If M^{out} is a DeepWL+WSC-machine, M^{wit} is a choice-free DeepWL+WSC-machine, and $\mathcal{M}_1, \dots, \mathcal{M}_\ell$ is a possibly empty sequence of DeepWL+WSC-algorithms, then the tuple $\mathcal{M} = (M^{\text{out}}, M^{\text{wit}}, \mathcal{M}_1, \dots, \mathcal{M}_\ell)$ is a **DeepWL+WSC-algorithm**. The machine M^{out} is called the **output machine** of $\mathcal{M}$ and the machine M^{wit} is called the **witnessing machine** of $\mathcal{M}$.

Note that the base case of the former definition is the case $\ell = 0$, i.e., the sequence of nested DeepWL+WSC-algorithms is empty. The nested algorithms can be used by the machines M^{out} and M^{wit} as subroutines. We first discuss the execution of a DeepWL+WSC-algorithm and in particular the use of subroutines intuitively. A formal definition will follow. Let $\mathcal{M} = (M^{\text{out}}, M^{\text{wit}}, \mathcal{M}_1, \dots, \mathcal{M}_\ell)$ be a DeepWL+WSC-algorithm. As first step, we consider the executions of the DeepWL+WSC-machines M^{out} and M^{wit}.

(a) Assume a DeepWL+WSC-machine $M \in \{M^{\text{out}}, M^{\text{wit}}\}$ executes $\texttt{refine}(X, j)$. If $j > \ell$, then M just continues. If otherwise $j \in [\ell]$, then the DeepWL+WSC-algorithm $\mathcal{M}_j$ is used to refine the relation X: Let $\mathfrak{A}$ be the content of the cloud of M when M executes the `refine`-operation. If X is directed, the algorithm $\mathcal{M}_j$

is executed on $(\mathfrak{A}, uv)$ (u and v are individualized by putting them into singleton vertex classes) for each $(u, v) \in X^{\mathfrak{A}}$. Otherwise X is undirected, so $X^{\mathfrak{A}} = (X^{\mathfrak{A}})^{-1}$, then for each $\{u, v\} \in X^{\mathfrak{A}}$, the algorithm $\mathcal{M}_j$ is executed on $(\mathfrak{A}, \{uv\})$ (the undirected edge is individualized by creating a new vertex class only containing u and v). The algorithm $\mathcal{M}_j$ modifies its own cloud independently of the cloud of M. If $\mathcal{M}_j$ accepts $(\mathfrak{A}, uv)$ for every $(u, v) \in X^{\mathfrak{A}}$ (or $(\mathfrak{A}, \{uv\})$ for every $\{u, v\} \in X^{\mathfrak{A}}$), then nothing happens and M continues. Otherwise, a new relation E' is added to the cloud of M, where E' consists of all $(u, v) \in X^{\mathfrak{A}}$ (or all $\{u, v\} \in X^{\mathfrak{A}}$, respectively), for which $\mathcal{M}_j$ accepts the input.

(b) Assume a DeepWL+WSC-machine $M \in \{M^{\text{out}}, M^{\text{wit}}\}$ executes `choice`(X). If X is a directed relation, then an arbitrary $(u, v) \in X^{\mathfrak{A}}$ is individualized and M continues. If otherwise X is undirected, then an undirected edge $\{u, v\} \in X^{\mathfrak{A}}$ is individualized. The edges are individualized as described in the `refine`-operation.

The algebraic sketch is recomputed and written onto the interaction-tape whenever the structure in the cloud is modified by `refine`(X, j) or `choice`(X). The machine M accepts the input if the symbol under the head on the work-tape is a 1 when M halts and rejects otherwise. Later, we will formally define the execution with choices using a tree, similar to the definition of the iteration terms with choice in CPT+WSC in Section 5.1.1.

We now turn to the DeepWL+WSC-algorithm $\mathcal{M}$. To execute the algorithm $\mathcal{M}$ on input $\mathfrak{A}_0$, the output machine M^{out} is executed on $\mathfrak{A}_0$. Let $\mathfrak{A}$ be the content of the cloud when M^{out} halts. For every `choice`-operation executed by M^{out}, the witnessing machine M^{wit} is executed. Let k be a number not exceeding the number of `choice`-operations, `choice`(X_i) be the i-th executed `choice`-operation (for some X_i in the current signature) for every $i \in [k]$, and $\mathfrak{A}_i$ be the content of the cloud, when the i-th `choice`-operation is executed for every $i \in [k]$. For the k-th `choice`-operation `choice`(X_k), the machine M^{wit} has to provide automorphisms witnessing that $X^{\mathfrak{A}_k}$ is an $(\mathfrak{A}_0, \mathfrak{A}_1, \ldots, \mathfrak{A}_k)$-orbit (details follow later). That is, similarly to the WSC-fixed-point operator, all intermediate steps of the fixed-point computation have to be fixed by the witnessing automorphisms. Recall again that we are working with HF-structures with the same set of atoms, so all vertices are $\mathsf{HF}(A_0)$-sets, so the notion of an $(\mathfrak{A}_0, \mathfrak{A}_1, \ldots, \mathfrak{A}_k)$-orbit is well-defined.

The input of M^{wit} is the **labeled union** $\mathfrak{A} \uplus \mathfrak{A}_k$, which is the union $\mathfrak{A} \cup \mathfrak{A}_k$ equipped with two fresh relation symbols E_1 and E_2 labeling the vertices of $\mathfrak{A}$ and $\mathfrak{A}_k$, i.e.,

$$E_1^{\mathfrak{A} \uplus \mathfrak{A}_k} := \mathsf{V}(\mathfrak{A}) \text{ and}$$
$$E_2^{\mathfrak{A} \uplus \mathfrak{A}_k} := \mathsf{V}(\mathfrak{A}_k).$$

So M^{wit} is able to reconstruct $\mathfrak{A}$ and $\mathfrak{A}_k$ and to determine how $\mathfrak{A}$ and $\mathfrak{A}_k$ relate to each other: Since the atoms of both $\mathfrak{A}$ and $\mathfrak{A}_k$ are $\mathsf{HF}(\mathfrak{A}_0)$-sets, common vertices are "merged" in the union. When the witnessing machine M^{wit} halts, it has to write a relation symbol onto the interaction-tape. The relation has to encode a set of witnessing automorphisms (details on the encoding follow later).

If all choices are successfully witnessed, $\mathcal{M}$ accepts $\mathfrak{A}_0$ if M^{out} accepts $\mathfrak{A}_0$ the input and rejects otherwise. If some choice could not be witnessed, we abort the computation and output †. If an executed subalgorithm $\mathcal{M}_i$ outputs †, then $\mathcal{M}$ also outputs †. We also say that $\mathcal{M}$ **fails** if $\mathcal{M}$ outputs †.

Semantics of the new Operations. Both, the `refine`- and the `choice`-operation, contain some special cases in its semantics. First, both operations treat directed and undirected relations differently. For undirected relations X, an undirected edge $\{u,v\} \in X^{\mathfrak{A}}$ is individualized and not just a directed one. Second, a `refine`-operation does not create a new relation in the case that every $(u,v) \in X^{\mathfrak{A}}$ or every $\{u,v\} \in X^{\mathfrak{A}}$ are accepted by $\mathcal{M}_j$. Creating a new relation containing the same edges as X would seem more natural. Indeed, for general DeepWL+WSC-algorithms, these special cases do not change the expressive power. But later in Section 5.3.4, we will introduce the notion of a normalized DeepWL+WSC-algorithm, which will put additional restrictions on, e.g., `addPair`-operations. Here the precise semantics of the `refine`- and `choice`-operations will matter, in particular, it will be crucial to prove Lemma 5.57.

Encoding Automorphisms. We now discuss how sets of automorphisms are encoded. Let $\mathfrak{A}_0$ be the input structure and $\mathfrak{A}$ be the current content of the cloud. A tuple of relations $(E_{\text{aut}}, E_{\text{dom}}, E_{\text{img}})$ and a vertex w_φ **encode the partial map** $\varphi \colon A \to A$ as follows: We have $\varphi(u) = v$ in the case that v is the only vertex for which there exists exactly one w such that $(w_\varphi, w) \in E_{\text{aut}}^{\mathfrak{A}}$, $(w,u) \in E_{\text{dom}}^{\mathfrak{A}}$, and $(w,v) \in E_{\text{img}}^{\mathfrak{A}}$. A tuple of relations $(E_{\text{aut}}, E_{\text{dom}}, E_{\text{img}})$ **encodes the set of partial maps**

$$N = \left\{ \varphi \;\middle|\; w_\varphi \text{ encodes } \varphi \text{ for some } (w_\varphi, w) \in E_{\text{aut}}^{\mathfrak{A}} \right\}.$$

For a tuple $\bar{a} \in \mathsf{HF}(\mathfrak{A}_0)^*$, the tuple $(E_{\text{aut}}, E_{\text{dom}}, E_{\text{img}})$ witnesses a relation $E_{\text{orb}}^{\mathfrak{A}}$ as $(\mathfrak{A}_0, \bar{a})$-orbit if the set N witnesses $E_{\text{orb}}^{\mathfrak{A}}$ as $(\mathfrak{A}_0, \bar{a})$-orbit.

Execution with Choices. Intuitively, we have to nest DeepWL+WSC-algorithms to ensure that DeepWL+WSC-algorithms "return" an isomorphism-invariant result (accept or reject). This corresponds to the output formula in a WSC-fixed-point operator, which ensures that it defines an isomorphism-invariant query. We now define the execution of a DeepWL+WSC-algorithm formally. We need to deal with choices to obtain a well-defined notion. In the following, we will assume that all considered Turning machines always terminate. We can do so because we will be only interested in polynomial-time Turing machines in this chapter.

A **configuration** c of a DeepWL+WSC-machine M is a tuple of a state $q(c)$ of the Turing machine contained in M and the content of the two tapes of the machine. Assume $\mathcal{M} = (M^{\text{out}}, M^{\text{wit}}, \mathcal{M}_1, \dots, \mathcal{M}_\ell)$ is an arbitrary DeepWL+WSC-algorithm and $\mathfrak{A}_0$ is an input HF-structure to $\mathcal{M}$. Let $\delta^{\mathfrak{A}_0}$ be the transition function of M^{out}: For configurations c and c' of M^{out} and HF-structures $\mathfrak{A}$ and $\mathfrak{A}'$, both with atom set A_0, we have $\delta^{\mathfrak{A}_0}(c, \mathfrak{A}) = (c', \mathfrak{A}')$ if M^{out}, started in configuration c with $\mathfrak{A}$ in the cloud, executes the first `choice`-operation (or halts if no `choice`-operation is executed) in the configuration c' with $\mathfrak{A}'$ in the cloud. In particular, no choice operation is executed in the computation from $(c, \mathfrak{A})$ to $(c', \mathfrak{A}')$. If all DeepWL+WSC-algorithms $\mathcal{M}_1, \dots, \mathcal{M}_\ell$ are deterministic, i.e., accept, reject, or fail independent of the choices made during the execution of the $\mathcal{M}_i$, then all `refine`-operations executed by M^{out} are deterministic and so $\delta^{\mathfrak{A}_0}$ is indeed a well-defined function. We view a tuple $(c, \mathfrak{A})$ as an $\mathsf{HF}(A_0)$-set: The configuration c is encoded as an $\mathsf{HF}(\emptyset)$-set and the vertices of $\mathfrak{A}$ itself are $\mathsf{HF}(A_0)$-sets.

To define the run of the DeepWL+WSC-algorithm $\mathcal{M}$ in the presence of choices, we reuse the WSC^*-operator from Section 5.1.1. The set of possible runs of $\mathcal{M}$ is the set

$$\mathsf{WSC}^*(\delta^{\mathfrak{A}_0}_{\text{step}}, \delta^{\mathfrak{A}_0}_{\text{choice}}, \delta^{\mathfrak{A}_0}_{\text{wit}}),$$

where we define the functions $\delta^{\mathfrak{A}_0}_{\text{step}}$, $\delta^{\mathfrak{A}_0}_{\text{choice}}$, and $\delta^{\mathfrak{A}_0}_{\text{wit}}$ using $\delta^{\mathfrak{A}_0}$ as follows. In the beginning, the function $\delta^{\mathfrak{A}_0}_{\text{choice}}$ outputs the empty choice-set (i.e., makes no choice) and the function $\delta^{\mathfrak{A}_0}_{\text{step}}$ starts the output machine in the initial configuration c^{out}_0 (initial state, empty work-tape, and $D(\mathfrak{A}_0)$ written on the interaction-tape) with the initial input structure $\mathfrak{A}_0$ in the cloud, that is,

$$\begin{aligned}\delta^{\mathfrak{A}_0}_{\text{step}}(\emptyset, \emptyset) &:= (c^{\text{out}}_0, \mathfrak{A}_0),\\ \delta^{\mathfrak{A}_0}_{\text{choice}}(\emptyset) &:= \emptyset.\end{aligned}$$

Next, whenever a $\texttt{choice}(X)$-operation is executed, $\delta^{\mathfrak{A}_0}_{\text{choice}}$ outputs the relation $\tilde{X}^{\mathfrak{A}}$ in the current content of the cloud $\mathfrak{A}$, where $\tilde{X}^{\mathfrak{A}} := X^{\mathfrak{A}}$ if the relation X is directed and $\tilde{X}^{\mathfrak{A}} := \{\{u, v\} \mid (u, v) \in X^{\mathfrak{A}}\}$ if X is undirected:

$$\delta^{\mathfrak{A}_0}_{\text{choice}}((c^{\text{out}}, \mathfrak{A})) := \begin{cases}\tilde{X}^{\mathfrak{A}} & \text{if } q(c^{\text{out}}) = q_{\texttt{choice}} \text{ and } X \text{ is on interaction tape in } c^{\text{out}},\\ \emptyset & \text{otherwise.}\end{cases}$$

After choosing a set a (of size at most 1 encoding a directed or undirected edge), a is individualized as described earlier and the resulting structure is denoted by $(\mathfrak{A}, a)$. Then $\delta^{\mathfrak{A}_0}_{\text{step}}$ continues with the next configuration $\hat{c}^{\text{out}}$ according to the transition function of M^{out} in the structure (and the new algebraic sketch written onto the interaction-tape):

$$\delta^{\mathfrak{A}_0}_{\text{step}}((c^{\text{out}}, \mathfrak{A}), a) := \begin{cases}\delta^{\mathfrak{A}_0}(\hat{c}^{\text{out}}, (\mathfrak{A}, a)) & \text{if } q(c^{\text{out}}) = q_{\texttt{choice}},\\ (c^{\text{out}}, \mathfrak{A}) & \text{otherwise.}\end{cases}$$

If in the end of the computation a halting state is reached (so $q(c^{\text{out}}) \neq q_{\texttt{choice}}$), then $\delta^{\mathfrak{A}_0}_{\text{choice}}$ returns the empty set, nothing is chosen, and $\delta^{\mathfrak{A}_0}_{\text{step}}$ reaches a fixed-point.

The function $\delta^{\mathfrak{A}_0}_{\text{wit}}$ maps a pair $((c, \mathfrak{A}), (c', \mathfrak{A}_k))$ to the set of partial maps encoded by the relation which is written on the interaction-tape when M^{wit} halts on input $\mathfrak{A} \uplus \mathfrak{A}_k$. Because M^{wit} is choice-free, we do not have to deal with choices here. Recall from the WSC^*-operator, that for a $\texttt{choice}(X)$-operation, the relation X has to be an orbit which fixed all intermediate steps: Let $\texttt{choice}(X_1), \dots, \texttt{choice}(X_k)$ be the sequence of all already executed $\texttt{choice}$-operations and let $\mathfrak{A}_1, \dots, \mathfrak{A}_k$ be the contents of the cloud when the corresponding $\texttt{choice}$-operation is executed. Then the set $X_k^{\mathfrak{A}_k}$ has to be an $(\mathfrak{A}_0, \mathfrak{A}_1, \dots, \mathfrak{A}_k)$-orbit[2] (recall again, that all structures $\mathfrak{A}_0, \dots, \mathfrak{A}_k$ have the same atom set A_0), which $\delta^{A_0}_{\text{wit}}$ has to witness. Also, recall from the definition of the WSC^*-operator, that the input to $\delta^{\mathfrak{A}_0}_{\text{wit}}$ is indeed the pair $((c, \mathfrak{A}), (c', \mathfrak{A}_k))$ of the reached fixed-point $(c, \mathfrak{A})$ and the intermediate step $(c', \mathfrak{A}_k)$ on which the k-th $\texttt{choice}$-operation is executed.

[2]Formally, by the definition of the WSC^*-operator, $X_k^{\mathfrak{A}_k}$ has to be an $(\mathfrak{A}_0, (c_1, \mathfrak{A}_1), \dots, (c_k, \mathfrak{A}_k))$-orbit, where c_i is the configuration of the Turing machine at the moment when the i-th choice operation is executed. But since the c_i are encoded as $\mathsf{HF}(\emptyset)$-set, they are invariant under all permutations of the atoms.

Let $W = \mathsf{WSC}^*(\delta^{\mathfrak{A}_0}_{\text{step}}, \delta^{\mathfrak{A}_0}_{\text{choice}}, \delta^{\mathfrak{A}_0}_{\text{wit}})$. The algorithm $\mathcal{M}$ **accepts** $\mathfrak{A}_0$ if for some (and thus for every) $(c, \mathfrak{A}) \in W$, the head of the work-tape in c points to a 1, **fails** if $W = \emptyset$, and **rejects** otherwise. Note that $W = \emptyset$ if and only if there are non-witnessed choices because we assumed that our DeepWL+WSC-algorithms always terminate and so a nonempty fixed-point is always reached.

Lemma 5.27. *Every DeepWL+WSC-algorithm $\mathcal{M} = (M^{out}, M^{wit}, \mathcal{M}_1, \dots \mathcal{M}_\ell)$ satisfies the following:*

1. *The class of structures accepted by $\mathcal{M}$ is closed under isomorphisms.*
2. *The algorithm $\mathcal{M}$ always accepts, rejects, or fails independent of the choices made in the execution of M^{out}.*
3. *If all choices were witnessed, then the series of configurations of M^{out} is the same for all possible choices.*

Proof. The proof is by induction on the nesting depth of DeepWL+WSC-algorithms. Let $\mathcal{M} = (M^{\text{out}}, M^{\text{wit}}, \mathcal{M}_1, \dots \mathcal{M}_\ell)$ be a DeepWL+WSC-algorithm, $\mathfrak{A}_0$ be the input structure, and assume inductively that the claim holds for the DeepWL+WSC-algorithms $\mathcal{M}_1, \dots, \mathcal{M}_\ell$. So every `refine`-operation executed by M^{out} is isomorphism-invariant. Additionally, all other operations apart from `choice` modify the cloud in an isomorphism-invariant manner and the algebraic sketch itself is isomorphism-invariant [59]. Because $\delta^{\mathfrak{A}_0}$ "stops" the execution when the first `choice`-operation is encountered, $\delta^{\mathfrak{A}_0}$ is well-defined (i.e., deterministic) and isomorphism-invariant. That was exactly our assumption on $\delta^{\mathfrak{A}_0}$ in the former paragraph. Likewise, the function $\delta^{\mathfrak{A}_0}_{\text{wit}}$ is isomorphism-invariant because the machine M^{wit} is choice-free. Thus, $\delta^{\mathfrak{A}_0}_{\text{choice}}$ is isomorphism-invariant because $\delta^{\mathfrak{A}_0}_{\text{step}}$ is isomorphism-invariant, which is the case since the relation used as choice-set only depends on the configuration returned by $\delta^{\mathfrak{A}_0}_{\text{step}}$. That is, we can apply the lemmas in Section 5.1.1. By Corollary 5.5, the set $\mathsf{WSC}^*(\delta^{\mathfrak{A}_0}_{\text{step}}, \delta^{\mathfrak{A}_0}_{\text{choice}}, \delta^{\mathfrak{A}_0}_{\text{wit}})$ is an orbit of $\mathfrak{A}_0$. If $\mathsf{WSC}^*(\delta^{\mathfrak{A}_0}_{\text{step}}, \delta^{\mathfrak{A}_0}_{\text{choice}}, \delta^{\mathfrak{A}_0}_{\text{wit}}) = \emptyset$, then some choice could not be witnessed, which by Corollary 5.4 is either the case for all possible choices or never occurs. Otherwise, the configuration c is the same for all tuples $(c, \mathfrak{A}) \in \mathsf{WSC}^*(\delta^{\mathfrak{A}_0}_{\text{step}}, \delta^{\mathfrak{A}_0}_{\text{choice}}, \delta^{\mathfrak{A}_0}_{\text{wit}})$ because the configuration c is invariant under all permutations of the atoms (because c is an $\mathsf{HF}(\emptyset)$-set). So the algorithm either accepts or rejects for all possible choices (and Parts 1 and 2 are proved).

To see Part 3, note that we can replace the configuration c of the Turing machine in the tuples $(c, \mathfrak{A})$ with the sequence of all visited configurations so far without breaking one of the arguments before. Then not only the last configuration is an orbit, but also the sequence of visited configurations, which is thus invariant under all automorphisms (given that all choices were witnessed). $\square$

The former lemma has the consequence that, apart from possibly `choice`-operations, none of the operations change the automorphisms of the structure in the cloud.

Corollary 5.28. *Assume $\mathcal{M} = (M^{out}, M^{wit}, \mathcal{M}_1, \dots \mathcal{M}_\ell)$ is a DeepWL+WSC-algorithm, $M \in \{M^{out}, M^{wit}\}$, and $\mathfrak{A}$ is the current content of the cloud of M (at some point during its execution). Assume that M executes* `addPair`, `create`, `scc`, *or* `refine` *and let $\mathfrak{A}'$ be the content of the cloud of M after the execution. Then $\mathsf{Aut}(\mathfrak{A}) = \mathsf{Aut}(\mathfrak{A}')$.*

Proof. Recall that an automorphism of an HF-structure is a permutation of the *atoms*, which extends to all vertices. By construction, the `addPair`-, `create`-, and `scc`-operations are isomorphism-invariant. By Lemma 5.27, also the `refine`-operation is isomorphism-invariant because the class of structures accepted by DeepWL+WSC-algorithm used to refine a relation is isomorphism-closed. □

Internal Run. The internal run of a DeepWL-algorithm is the sequence of configurations of the Turing machine (the state and the content of the two tapes) during the run. Note that in particular the algebraic sketches computed during the computations are part of the internal run because they are written onto the interaction-tape. As we have already seen, the internal run of a DeepWL-algorithm is isomorphism-invariant [59]. To define the internal run of a DeepWL+WSC-algorithm, we have – beside the internal run of the output machine – to take all internal runs of the witnessing machine to witness the different choice-sets and the internal runs of the subalgorithms into account. The internal run is defined inductively over the nesting depth of DeepWL+WSC-algorithms.

Let $\mathcal{M} = (M^{\text{out}}, M^{\text{wit}}, \mathcal{M}_1, \dots, \mathcal{M}_\ell)$ be a DeepWL+WSC-algorithm and assume that we have defined the internal run $\mathsf{run}(\mathcal{M}_i, \mathfrak{A})$ for every $i \in [\ell]$ and every HF-structure $\mathfrak{A}$. The **internal run** $\mathsf{run}(M, \mathfrak{A})$ of a DeepWL+WSC-machine $M \in \{M^{\text{out}}, M^{\text{wit}}\}$ on input $\mathfrak{A}$ is the following: Let $c_1, \dots, c_n$ be the sequence of configurations of the Turing machine of M. By Lemma 5.27, the sequence is unique if all choices will be witnessed, which we assume for now. Furthermore, let $k_1 < \dots < k_m$ be all indices such that $q(c_{k_j}) = q_{\texttt{refine}}$ and M executes $\texttt{refine}(X_j, i_j)$ such that $i_j \leq \ell$. Let $\mathfrak{A}_j$ be the content of the cloud before executing $\texttt{refine}(X_j, i_j)$ for all $j \in [m]$. We define

$$\begin{aligned} r_j &:= \left\{\!\!\left\{ \mathsf{run}(\mathcal{M}_{i_j}, (\mathfrak{A}_j, x)) \;\middle|\; x \in \tilde{X}_j^{\mathfrak{A}_j} \right\}\!\!\right\} && \text{for every } j \in [m], \\ \mathsf{run}(M, \mathfrak{A}) &:= \begin{cases} \dagger & \\ c_1, \dots, c_{k_1}, r_1, c_{k_1+1}, \dots, c_{k_m}, r_m, c_{k_m+1}, \dots, c_\ell & \end{cases} && \begin{aligned} &\text{if } \dagger \in r_i \text{ for some } i \in [m], \\ &\text{otherwise.} \end{aligned} \end{aligned}$$

The internal run, denoted $\mathsf{run}(\mathcal{M}, \mathfrak{A})$, of the DeepWL+WSC-algorithm $\mathcal{M}$ on input $\mathfrak{A}$ is defined as follows: Let $c_1, \dots, c_\ell$ be the sequence of configurations of the output Turing machine M^{out}. Furthermore, let $k_1 < \dots < k_m$ be all indices such that $q(c_{k_j}) = c_{\texttt{choice}}$. Let the corresponding `choice`-executions be $\texttt{choice}(X_j)$, where the current content of the cloud is $\mathfrak{A}_j$, for every $j \in [m]$. Let $\mathfrak{A}_{m+1}$ be the final content of the cloud when M^{out} halts. We define

$$\begin{aligned} \hat{r}_j &:= \mathsf{run}(M^{\text{wit}}, \mathfrak{A}_{m+1} \uplus \mathfrak{A}_j) && \text{for every } j \in [m], \\ \mathsf{run}(\mathcal{M}, \mathfrak{A}) &:= \begin{cases} \mathsf{run}(M^{\text{out}}, \mathfrak{A}), \hat{r}_1, \dots, \hat{r}_m & \\ \dagger & \end{cases} && \begin{aligned} &\text{if all choices are witnessed,} \\ &\text{otherwise.} \end{aligned} \end{aligned}$$

Using Lemma 5.27, it is easy to see that $\mathsf{run}(\mathcal{M}, \mathfrak{A})$ is isomorphism-invariant and thus it can be canonically encoded as a 0/1-string.

Computability. Let $\mathcal{M} = (M^{\text{out}}, M^{\text{wit}}, \mathcal{M}_1, \dots, \mathcal{M}_\ell)$ be a DeepWL+WSC-algorithm. The algorithm $\mathcal{M}$ **decides a boolean query** Q of a class of τ-structures $\mathcal{K}$ if $\mathcal{M}$ accepts $\mathfrak{A} \in \mathcal{K}$ whenever $\mathfrak{A}$ satisfies Q and rejects otherwise (and in particular never fails). The algorithm $\mathcal{M}$ **computes a function** $f\colon \mathcal{K} \to \{0,1\}^*$ if, for every $\mathfrak{A} \in \mathcal{K}$, the machine M^{out} has written $f(\mathfrak{A})$ onto the work-tape when it halts on input $\mathfrak{A}$ and $\mathcal{M}$ never fails.

Definitions 5.9 and 5.10 of definable isomorphism and complete invariant are easily adapted to DeepWL+WSC: DeepWL+WSC decides isomorphism of a class of structures $\mathcal{K}$ if there is a DeepWL+WSC algorithm which accepts $\mathfrak{A} \uplus \mathfrak{B}$ if and only if $\mathfrak{A} \cong \mathfrak{B}$ for all $\mathfrak{A}, \mathfrak{B} \in \mathcal{K}$. DeepWL+WSC computes a complete invariant for $\mathcal{K}$ if there is a DeepWL+WSC algorithm computing a function $f\colon \mathcal{K} \to \{0,1\}^*$ such that $f(\mathfrak{A}) = f(\mathfrak{B})$ if and only if $\mathfrak{A} \cong \mathfrak{B}$ for all $\mathfrak{A}, \mathfrak{B} \in \mathcal{K}$.

Next, we define the **runtime** of a DeepWL+WSC-machine $M \in \{M^{\text{out}}, M^{\text{wit}}\}$ on input $\mathfrak{A}_0$. Every transition taken by the Turing machine counts as one time step. Whenever a cloud-modifying operating is executed and the algebraic sketch $D(\mathfrak{A})$ of the new structure in the cloud $\mathfrak{A}$ is written onto the interaction-tape, we count $|D(\mathfrak{A})|$ many time steps, where $|D(\mathfrak{A})|$ is the encoding length of $D(\mathfrak{A})$. Following [59], the encoding of $D(\mathfrak{A})$ is unary and so the runtime of M is at least $|A_0|$. When M executes $\texttt{refine}(X, i)$ for $i \leq \ell$ and the current content of the cloud is $\mathfrak{A}$, we count the sum of runtimes of $\mathcal{M}_i$ on input $(\mathfrak{A}, x)$ for every $x \in \tilde{X}^{\mathfrak{A}}$. The runtime of $\mathcal{M}$ is the sum of the runtime of the output machine M^{out} and the runtimes of the witnessing machine M^{wit} to witness all choices. A DeepWL+WSC-algorithm (or machine) runs in **polynomial time** if there exists a polynomial $p(n)$ such that $p(|D(\mathfrak{A}_0)|)$ bounds the runtime on input $\mathfrak{A}_0$ for every HF-structure $\mathfrak{A}_0$ (or possibly for every $\mathfrak{A}_0$ in a class of HF-structures of interest). Note that if $M^{\text{out}}, M^{\text{wit}}, \mathcal{M}_1, \dots, \mathcal{M}_\ell$ run in polynomial time, then $\mathcal{M}$ runs in polynomial time: The size of the structure in the cloud is polynomially bounded and the machines M^{out} and M^{wit} can execute only polynomially bounded many $\texttt{refine}$-operations.

5.3.2.1 Derived Operations

From the special operations `addPair`, `scc`, and `choice` we can derive additional operations, which then can be used for convenience. These are:

1. Ordered inputs: Currently, the structure $\mathfrak{A}$ put in the cloud is the only input to a DeepWL+WSC-machine or algorithm. We allow a pair $(\mathfrak{A}, \bar{z})$ of a structure $\mathfrak{A}$ and a binary string $\bar{z} \in \{0,1\}^*$ placed on the working tape as input.

2. Ordered input for `refine`-operations: We similarly extend `refine`-operations to support $\texttt{refine}(C, i, \bar{z})$ executions, where $\bar{z} \in \{0,1\}^*$ is an additional binary string initially put on the working tape.

3. $\texttt{rename}(E, F)$: If F is an unused relation symbol in the signature τ, the machine exchanges a relation symbol E with F. Otherwise, no change is made.

4. $\texttt{addUPair}(X)$: This operation creates *unordered pairs*, i.e., sets of size at most two for the X-edges: Let $\mathfrak{A}$ be the current structure in the cloud. For every $\{u, v\}$ such that $(u, v) \in X^{\mathfrak{A}}$, a new vertex is added and the membership relation $\bigcup_{(u,v) \in X^{\mathfrak{A}}} \{(\{u, v\}, u), (\{u, v\}, v)\}$ is created.

While adding the ability to rename relations seems useless, it simplifies proofs because we do not have to maintain bijections between relation symbols of different structures. Instead, we just can rename them accordingly.

Lemma 5.29. *Ordered input to DeepWL+WSC-algorithms and the operations* `addUPair`*,* `rename`*, and* `refine`$(C, i, \bar{z})$ *can be simulated with the existing operations with only a polynomial overhead.*

Proof. We describe how the new operations can be simulated.

1. Ordered inputs can be simulated in the original DeepWL+WSC model as follows: Let $\mathfrak{A}$ be an input HF-structure, $\bar{z} \in \{0,1\}^*$, and $m = |\bar{z}|$. We encode $\bar{z}$ into the string $\bar{z}' := 0^4 1^4 z_1 0 z_2 1 z_3 0 z_4 1 \dots z_m$. There is only one position at which $0^4 1^4$ occurs as substring of $\bar{z}'$, namely the first. We encode $\bar{z}'$ into $\mathfrak{A}$ using vertex classes. Let $m' = |\bar{z}'|$. We add m' many vertex classes $C_1, \dots, C_{m'}$ such that
$$C_i^{\mathfrak{A}} = \begin{cases} \emptyset & \text{if } z_i' = 0, \\ A & \text{if } z_i' = 1, \end{cases}$$
for every $i \in [m']$. The string $\bar{z}'$ can be read off the algebraic sketch: The algorithm starts at the lexicographically greatest relation symbol. Using the algebraic sketch the algorithm decides whether it encodes a 0 or a 1. This process is continued until the substring $0^4 1^4$ is found. The algorithm computes $\bar{z}$ from $\bar{z}'$. Surely, the new vertex classes do not change the automorphisms of the structure $\mathfrak{A}$, so exactly the same orbits are witnessed after adding $C_1, \dots, C_{m'}$. Using Lemma 5.24, we can always compute the algebraic sketch of the structure without the new vertex classes.

2. `refine`$(C, i, \bar{z})$: We proceed in the very similar way and encode $\bar{z}$ by the string $\bar{z}'$ as defined before using vertex classes. But now, the vertex classes $C_1, \dots, C_{m'}$, where $m' = |\bar{z}'|$, have to be created by the DeepWL+WSC-algorithm. This can easily be done by executing `create`(π_i) for every $i \in [m']$, where $\pi_i = \emptyset$ if $z_i' = 0$ and π_i is the set of all fibers if $z_i' = 1$.

   The DeepWL+WSC-algorithm used to refine C first reads off $\bar{z}'$ from the algebraic sketch and then, by Lemma 5.24, computes the algebraic sketch without the vertex classes $C_1, \dots, C_{m'}$.

3. `rename`(E, F): The DeepWL+WSC-machine maintains the bijection between current and renamed relation symbols on its tape. The bijection is passed through `refine`-operations using ordered input.

4. `addUPair`(X): We first execute `addPair`(X), so we obtain vertices $\langle u, v\rangle$ for every $(u, v) \in X^{\mathfrak{A}}$ and component relations E_{left} and E_{right}. Let π be the set of colors which have an $(E_{\text{left}}, E_{\text{right}}^{-1})$- and an $(E_{\text{right}}, E_{\text{left}}^{-1})$-colored triangle. Next, `create`(π) is executed and a relation F obtained. The relation F contains precisely all pairs $(\langle u, v\rangle, \langle v, u\rangle)$ for every $\{u, v\} \in X^{\mathfrak{A}}$. We execute `scc`$(E)$. We obtain, for each F-SCC, a new vertex and the membership relation E_{in}. Although the new vertices are created for the sets $\{\langle u, v\rangle, \langle v, u\rangle\}$, they resemble the sets $\{u, v\}$. The membership relation F_{in} for these sets is obtained as the union of colors, which have an $(E_{\text{in}}, E_{\text{left}})$- and an $(E_{\text{in}}, E_{\text{right}})$-colored triangle. For the pairs $(u, v) \in X^{\mathfrak{A}}$ for which $(v, u) \notin X^{\mathfrak{A}}$, we have not created an F-SCC vertex because these pair vertices are not incident to F. Instead, we can just use the pair vertices themselves

(no automorphism can map u to v or vice versa in this case). We can define the relations refining E_{left} and E_{right} incident to such pair vertices and also include them in F_{in}. As before, the sketch without the additionally created pair vertices can be computed using Lemma 5.24.

Regarding witnessing choices, the `addUPair`-operation is isomorphism-invaraint. Because we did not use a `choice`-operation in the simulation, we have not changed the automorphisms of the HF-structure in the cloud by Corollary 5.28. So exactly the choices in the execution with the `addUPair`-operation are witnessed as in the execution with the simulation.

Finally, it is easy to see that the simulations run in polynomial time, where the size of a tuple $(\mathfrak{A}, \bar{z})$ is $|D(\mathfrak{A})| + |\bar{z}|$. □

5.3.3 From CPT+WSC to DeepWL+WSC

In this section, we translate a CPT+WSC-formula into an equivalent polynomial-time DeepWL+WSC-algorithm. The following translation is based on the translation of CPT into interpretation logic in [46] and the translation of interpretation logic into DeepWL in [59]. However, we avoid the route through interpretation logic by directly implementing the ideas of [46] in DeepWL+WSC. To avoid case distinctions, we assume in the following that the occurring CPT+WSC-terms or formulas never output †. Although a †-respecting translation can be given, it is not needed for our purpose (all isomorphism-defining CPT+WSC-formulas never output †).

We simulate the evaluation of a BGS+WSC-term (or formula) with a DeepWL+WSC-algorithm. Recall that the vertices of the structure in the cloud during the execution of a DeepWL+WSC-machine on input structure $\mathfrak{A}$ are pairs $\langle a, i\rangle$ of a set $a \in \mathsf{HF}(A)$ and a number $i \in \mathbb{N}$, where $\langle a, i\rangle$ is encoded as an $\mathsf{HF}(A)$-set itself. For the simulation of a BGS+WSC-term (or formula) on input structure $\mathfrak{A}$, we encode two types of objects by vertices of the structure in the cloud:

1. A hereditarily finite set $a \in \mathsf{HF}(A)$ is encoded by a vertex $\langle a, i\rangle$ for some number i. We maintain a relation $E_\in$, which serves as **containment relation** between the vertices, that is, it corresponds to "$\in$" on the encoded sets. There is an exception for the empty set: Because a DeepWL+WSC-algorithm cannot create a vertex for the empty set, we encode $\emptyset$ by $\langle A, i\rangle$ for some number i. Here we require that $i \neq j$ for the number j for which $\langle A, j\rangle$ encodes A. We apply this recursively: Whenever $\emptyset$ is part of an $\mathsf{HF}(A)$-set, e.g., $\{\emptyset\}$, we consider the set obtained from replacing $\emptyset$ by A. We will be able to distinguish the encoding vertices, e.g., the vertices encoding $\{\emptyset\}$ and $\{A\}$, using the containment relation. Note that the containment relation is different from the membership relation obtained from `scc`-operations.

2. A tuple $\bar{a} \in \mathsf{HF}(A)^k$ of hereditarily finite sets is encoded by a vertex $\langle \bar{a}, i\rangle$ for some number i (for an appropriate encoding of tuples as hereditarily finite sets). We maintain a sequence of incident relations $E_1, \dots, E_k$, such that every E_j has out-degree 1 and associates to $\langle \bar{a}, i\rangle$ a vertex $\langle a_j, \ell\rangle$ for some number ℓ. We call these relations the **tuple relations**. We remark that the tuple relations in general will be different from the component relation of the `addPair`-operations.

We now introduce an intermediate version of DeepWL+WSC-algorithms. This intermediate version is needed for the recursive translation of CPT+WSC-terms and formulas. A **choice-free DeepWL+WSC-algorithm** is a tuple $(M^{\text{out}}, \mathcal{M}_1, \dots, \mathcal{M}_\ell)$ consisting of a choice-free DeepWL+WSC-machine M^{out} and DeepWL+WSC-algorithms $\mathcal{M}_1, \dots, \mathcal{M}_\ell$. The computation of a choice-free DeepWL+WSC-algorithm is exactly the same as the one of a regular DeepWL+WSC-algorithm, but since the output machine M^{out} is choice-free, no witnessing machine is needed. Note here that for a choice-free DeepWL+WSC-algorithm the subalgorithms $\mathcal{M}_1, \dots, \mathcal{M}_\ell$ are not choice-free. The benefit of choice-free algorithms is that they can be composed more easily, e.g., they can be executed one after the other without worrying about witnessing choices.

Definition 5.30 (Simulating CPT+WSC with DeepWL+WSC)**.** Let $\mathfrak{A}$ be a binary (non-HF) τ-structure. An HF-structure $\mathfrak{A}'$ is **compatible** with $\mathfrak{A}$ if $A' = A$ and $\mathsf{Aut}(\mathfrak{A}) = \mathsf{Aut}(\mathfrak{A}')$.

(a) A choice-free DeepWL+WSC-algorithm $\mathcal{M}$ **simulates a** CPT$[\tau]$**-term** $s(\bar{x})$ on $\mathfrak{A}$ if, for every HF-structure $\mathfrak{A}'$ compatible with $\mathfrak{A}$ and every vertex class C of $\mathfrak{A}'$ encoding $\mathsf{HF}(A)^{|\bar{x}|}$-tuples, the algorithm $\mathcal{M}$ on input $\mathfrak{A}'$ and C computes a vertex class C_s and a relation E_{io} with the following property:

- $E_{\mathsf{io}}^{\mathfrak{A}''} \subseteq C^{\mathfrak{A}'} \times C_s^{\mathfrak{A}'}$ is functional and surjective (where $\mathfrak{A}''$ is the content of the cloud after the execution of $\mathcal{M}$) and
- $(u, v) \in E_{\mathsf{io}}^{\mathfrak{A}''}$ if and only if the vertex u encodes a tuple $\bar{a} \in \mathsf{HF}(A)^{|\bar{x}|}$, the vertex v encodes a set $b \in \mathsf{HF}(A)$, and $b = s^{\mathfrak{A}}(\bar{a})$.

The relation E_{io} is called the **input-output relation**.

(b) A choice-free DeepWL+WSC-algorithm $\mathcal{M}$ **simulates a** CPT$[\tau]$**-formula** $\Phi(\bar{x})$ that is not a WSC-fixed-point operator (but may contain such operators as subformulas) on $\mathfrak{A}$ if, for every HF-structure $\mathfrak{A}'$ compatible with $\mathfrak{A}$ and every vertex class C of $\mathfrak{A}'$ encoding $\mathsf{HF}(A)^{|\bar{x}|}$-tuples, the algorithm $\mathcal{M}$ on input $\mathfrak{A}'$ and C defines a vertex class $C_\Phi^{\mathfrak{A}''} \subseteq C^{\mathfrak{A}'}$ of all C-vertices encoding a tuple $\bar{a} \in \Phi^{\mathfrak{A}}$.

(c) Last, a (non choice-free) DeepWL+WSC-algorithm $\mathcal{M}$ **simulates a WSC-fixed-point operator** $\Phi(\bar{x})$ on $\mathfrak{A}$ if, for every HF-structure $\mathfrak{A}'$ compatible with $\mathfrak{A}$ and every singleton vertex class C of $\mathfrak{A}'$ encoding an $\mathsf{HF}(A)^{|\bar{x}|}$-tuple $\bar{a}$, the algorithm $\mathcal{M}$ accepts on input $\mathfrak{A}'$ and C if and only if $\bar{a} \in \Phi^{\mathfrak{A}}$.

We say that a (choice-free) DeepWL+WSC-algorithm $\mathcal{M}$ simulates a CPT$[\tau]$-term or formula s if $\mathcal{M}$ simulates s on every τ-structure.

The former definition is intricate because of the free variables. For a BGS+WSC-sentence Φ, a DeepWL+WSC-algorithm $\mathcal{M}$ simulating Φ accepts exactly the structures satisfying Φ. We will translate CPT+WSC-formulas and terms to polynomial time DeepWL+WSC-algorithms by induction on the nesting structure of the WSC-fixed-point operators. We call a WSC-fixed-point operator Ψ **directly nested** in a CPT+WSC-formula (or a term) Φ if there is no WSC-fixed-point operator $\Psi' \neq \Psi$ such that Ψ is a subformula of Ψ' and Ψ' is a subformula of Φ.

Lemma 5.31. *For every* CPT+WSC*-formula* Φ *(or term) which is not a WSC-fixed-point operator, there is a choice-free polynomial-time DeepWL+WSC-algorithm simulating* Φ *if, for every WSC-fixed-point operator* Φ' *directly nested in* Φ*, there exists a polynomial-time DeepWL+WSC-algorithm simulating* Φ'*.*

Proof. In the following, we use relation symbols $X \in \tau \cup \sigma$ for the sets $X^{\mathfrak{A}}$, where $\mathfrak{A}$ is the current content of the cloud of the DeepWL+WSC-machine we are going to define. In that sense, for example, $X \setminus Y$ has a well-defined notion. This simplifies notation because we do not have to always introduce the current content of the cloud.

The proof is by induction on the CPT+WSC formula or term. Let Φ be a CPT+WSC formula (or r be a CPT+WSC-term). Let the directly nested WSC-fixed-point operators of Φ (or r, respectively) be $\hat{\Phi}_1, \dots, \hat{\Phi}_\ell$ and let, inductively, $\mathcal{M}_1, \dots, \mathcal{M}_\ell$ be DeepWL+WSC-algorithms, such that $\mathcal{M}_i$ simulates $\hat{\Phi}_i$ for every $i \in [\ell]$. We now define a DeepWL+WSC-machine M^{out} such that the choice-free DeepWL+WSC-algorithm $(M^{\text{out}}, \mathcal{M}_1, \dots, \mathcal{M}_\ell)$ simulates Φ (or r, respectively). We start with a vertex class C whose vertices encode the values for the free variables. Then we perform the computations to simulate the formulas and terms. During the simulation we want to ensure that for every encoded set a there is exactly one vertex of the form $\langle a, i\rangle$. We thus want to avoid duplicates (with the already mentioned exception for the empty set). So whenever we want to execute an `addPair`-, `addUPair`-, or `scc`-operation, we actually have to check whether the resulting vertices already exist. This can be done using the containment relation $E_\in$ (and for the `scc`-operation due to the fact that SCCs can be computed by DeepWL [59, Lemma 4]). From now on, we implicitly assume that these checks are always done and just say that we execute the `addPair`-, `addUPair`-, or `scc`-operations. We make the following case distinction for Φ and r:

- $\Psi = E_{\text{CPT}}(x, y)$: Here E_{CPT} is a relation symbol of the signature of the CPT+WSC-formula, which is thus contained in the current structure in the cloud. Given a vertex class C encoding the values for the pair xy, we return the vertex class C' consisting of all C-vertices for which the tuple relations are adjacent to atoms contained in the relation E_{CPT} as follows: Let E_1 and E_2 be the tuple relations for the first and second component of C. Then we want to return the vertex class containing the vertices which have an $(E_1, E_{\text{CPT}}, E_2^{-1})$-colored cycle. Due to coherence, there is a union of fibers $\pi \subseteq \sigma$, which contains precisely these vertices. We execute `create`(π) and output the resulting relation.

  In the following, we will for readability not mention the required `create`-operations or that we can find the union of fibers π if we want to obtain a relation of pairs with a $(F_1, \dots, F_k)$-colored path (for some relation symbols $F_1, \dots, F_k$).

- Ψ is $s(\bar{x}) = t(\bar{y})$: Given a vertex class C corresponding to the free variables $\bar{x} \cup \bar{y}$, we can use C for the free variables $\bar{x}$ and $\bar{y}$ by only using a subset of the tuple relations. We use the choice-free DeepWL+WSC-algorithms simulating s and t to obtain vertex classes C_s and C_t together with the input-output relations $E_{\mathsf{io},s}$ and $E_{\mathsf{io},t}$. Precisely one C_s-vertex u and one C_t-vertex v are obtained from the same C-vertex w if the edge (u, v) has an $(E_{\mathsf{io},s}^{-1}, E_{\mathsf{io},t})$-colored triangle. We now create a vertex class C' from the fibers refining C which have an $(E_{\mathsf{io},s}^{-1}, E_{\mathsf{io},t})$-colored cycle, that is, the two vertices u and v are actually the same. This is exactly then the case when the outputs of s and t are equal. Note that here it is crucial to perform the cleanup steps unifying vertices encoding the same set after every step.

- $\Psi = \neg\Phi$: Given a vertex class C, we obtain by induction a vertex class C_Φ. Then we return $C \setminus C_\Phi$.

- $\Psi = \Phi_1(\bar{x}) \land \Phi_2(\bar{x})$: Given a vertex class C, we obtain by induction vertex classes C_{Φ_1} and C_{Φ_2} and we just return $C_{\Phi_1} \cap C_{\Phi_2}$.

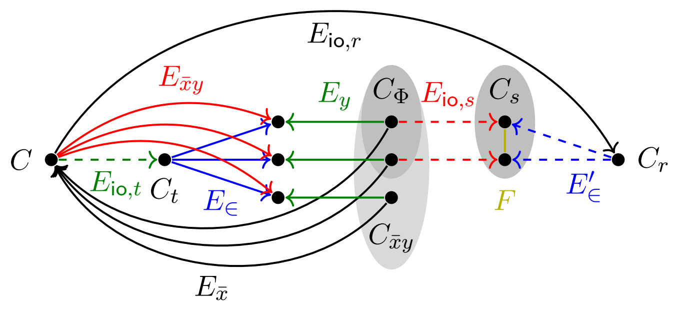
- $\Psi = \hat{\Phi}_i(\bar{x})$: Let C be a vertex class encoding the values for the variables $\bar{x}$. The machine executes $\texttt{refine}(C, i)$ and returns the resulting vertex class (note that C is a vertex class and so the fact that `refine`-operations treat directed and undirected relations differently does not matter here). By the induction hypothesis, this vertex class contains precisely the vertices encoding the tuples satisfying $\hat{\Phi}_i$.

- $r = \mathsf{Atoms}$: We create a relation E connecting all atoms of the input structure (all vertices, which have no outgoing edge of some membership or component relation) and execute $\texttt{scc}(E)$. We obtain a single additional vertex in a vertex class C_{Atoms} and a membership relation connecting it to the atoms, which is merged into the containment relation $E_\in$ (using a `create`- and a `rename`-operation). (Of course, as described before, we create the vertex class C_{Atoms} only once.)

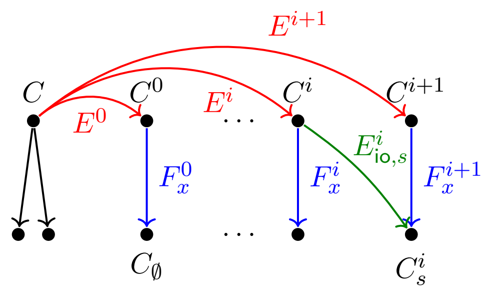
- $r = \emptyset$: In the same way as in the Atoms case, we obtain a single vertex in its own vertex class $C_\emptyset$, but now we do not merge the created membership relation into the containment relation $E_\in$. While $\emptyset$ is encoded by a vertex for the atoms, with respect to automorphism this does not make a difference because every automorphism stabilizes the set of all atoms and $C_\emptyset$ and C_{Atoms} contain different vertices $\langle A, i\rangle$ and $\langle A, j\rangle$ for $i \neq j$, where A is the set of atoms. So no automorphism can exchange these two vertices.

- $r = \mathsf{Pair}(s(\bar{x}), t(\bar{y}))$: Given a vertex class C encoding values for the free variables $\bar{x} \cup \bar{y}$, we can use C for the free variables $\bar{x}$ and $\bar{y}$ by only using a subset of the tuple relations. Then we use the choice-free DeepWL+WSC-algorithms simulating s and t to obtain vertex classes C_s and C_t together with the input-output relations $E_{\mathsf{io},s}$ and $E_{\mathsf{io},t}$. Then precisely the C_s-vertices u and C_t-vertices v are obtained for the same C-vertex w if the pair (u, v) has an $(E_{\mathsf{io},s}^{-1}, E_{\mathsf{io},t})$-colored triangle. That is, there is a union of colors $E_=$ containing exactly all such pairs (u, v). Then the machine executes $\texttt{addUPair}(E_=)$ and obtains the vertex class C_r and a new membership relation F. The membership relation F connects C_r-vertices to C_s- and C_t-vertices. The input-output relation $E_{\mathsf{io},r}$ is the union of colors which have an $(E_{\mathsf{io},s}, F^{-1})$- and an $(E_{\mathsf{io},t}, F^{-1})$-colored triangle. Finally, the machine merges F into the containment relation $E_\in$.

- $r = \mathsf{Unique}(s(\bar{x}))$: Given a vertex class C, we define the vertex class C_s with input-output relation $E_{\mathsf{io},s}$ and containment relation $E_\in$ using the induction hypothesis. The vertex class C_s can be partitioned into C_s^1 and $C_s^{\neq 1}$, where C_s^1-vertices have precisely one outgoing $E_\in$-edge and $C_s^{\neq 1}$-vertices do not. Let $E_{\mathsf{io},s}^1 := E_{\mathsf{io},s} \cap (C \times C_s^1)$ be the subset of $E_{\mathsf{io},s}$ leading to a C_s-vertex encoding a singleton set (which is



(a) Translation of $r = \{s(\bar{x}y) \mid y \in t(\bar{x}), \Phi(\bar{x}y)\}$



(b) Translation of $r = s[x]^*$

5.3 Translation of comprehension and iteration terms into DeepWL. Figure (a) shows the translation of a comprehension term into DeepWL. Figure (b) shows the translation of an iteration term. Both figures are drawn for a single vertex u in the input vertex class C and shows all vertices created for u.

DeepWL-computable). We similarly partition C into C^1 and $C^{\neq 1}$, where C^1 contains all C-vertices incident to $E^1_{\text{io},s}$.

The machine makes the following case distinction: If $|C^{\neq 1}| = 0$, set $C_r := C'$ and otherwise set $C_r := C' \cup C_\emptyset$. For the C^1-vertices, we obtain the input-output relation $E^1_{\text{io},r}$ as the relation with an $(E^1_{\text{io},s}, E_\in)$-colored triangle. For the $C^{\neq 1}$-vertices, let $E^{\neq 1}_{\text{io},r} = C^{\neq 1} \times C_\emptyset$ be the relation containing all edges between $C^{\neq 1}$ and the singleton vertex class $C_\emptyset$. Finally, output $E_{\text{io},r} := E^1_{\text{io},r} \cup E^{\neq 1}_{\text{io},r}$.

- $r = \mathsf{Card}(s(\bar{x}))$: Define for a given vertex class C the vertex class C_s, the input-output relation $E_{\text{io},s}$, and the containment relation $E_\in$ using the induction hypothesis. We partition C_s into $C^1_s, \dots, C^k_s$, such that C^i_s-vertices have i many outgoing $E_\in$-edges. Then the machine defines, for all $i \in [k]$ such that $C^i_s \neq \emptyset$, the i-th von Neumann ordinal, defines C_r as the union of these ordinals, and outputs $E_{\text{io},r}$ accordingly.

- $r = \{s(\bar{x}y) \mid y \in t(\bar{x}), \Phi(\bar{x}y)\}$: We start with the vertex class C and define by induction the vertex class C_t, the input-output relation $E_{\text{io},t}$, and the containment relation $E_\in$ (cf. Figure 5.3a). Then the machine defines vertex classes encoding the tuples for the variables $\bar{x}y$: Let $E_{\bar{x}y}$ be the relations with an $(E_{\text{io},t}, E_\in)$-colored triangle. These relations associate an input tuple with an element contained in the output set of t. The machine executes `addPair`$(E_{\bar{x}y})$ and obtains a vertex class $C_{\bar{x}y}$ with component relations $E_{\bar{x}} := E_{\text{left}}$ and $E_y := E_{\text{right}}$.

  By the induction hypothesis, we obtain the vertex class $C_\Phi \subseteq C_{\bar{x}y}$ of $C_{\bar{x}y}$-vertices satisfying Φ. With C_Φ as input vertex class, the machine obtains C_s again by induction and obtains the input-output relation $E_{\text{io},s}$.

  We define a relation F connecting C_s-vertices originating from the same C_t-vertex: F is given by the union of all colors R which have an $(E^{-1}_{\text{io},s}, E_y, E^{-1}_\in, E_\in, E^{-1}_y, E_{\text{io},s})$-colored path. We then execute `scc`(F) to obtain a vertex class C_r and a new membership relation $E'_\in$. Finally, the input-output relation $E_{\text{io},r}$ consists of the edges with an $(E_{\mathsf{io},t}, E_\in, E_y^{-1}, E_{\mathsf{io},s}, E'^{-1}_\in)$-colored path and the machine merges $E'_\in$ into the containment relation $E_\in$.

- $r = s[x]^*$: Let s have free variables $\bar{y}x$. We start with the vertex class C for the free variables $\bar{y}$ (cf. Figure 5.3b). We then execute `addPair`$(E_\emptyset)$ for the relation $E_\emptyset = C \times C_\emptyset$. We obtain the vertex class C^0 and the component relations $E^0 := E_{\text{left}}^{-1}$ between a C-vertex and the corresponding C^0-vertex and $F_x^0 := E_{\text{right}}$ defining the tuple relation to the entry for x.

  Let C^i be the vertex class containing the current input values after the i-th iteration. Let the relation E^i relate the values for $\bar{y}$ with the extended tuples $\bar{y}x$ in C^i (maybe not all $\bar{y}$ are related because a fixed-point for them is already computed). If i exceeds the maximal number of iterations (given by the polynomial bound of the CPT-term), set $C_*^{i+1} := \emptyset$ and E_*^{i+1} to be the relations with an (E^i, F)-colored triangle, where $F = C^i \times C_\emptyset$.

  Otherwise, the machine defines, starting with C^i, a vertex class C_s^i and the input-output relation $E_{\mathsf{io},s}^i$ using the induction hypothesis. Let F_x^i be the tuple relation of C^i for x. We partition E_s^i into $E_{\text{fix}}^i := E_s^i \cap F_x^i$ and $E_{\text{continue}}^i := E_s^i \setminus F_x^i$. In E_{fix}^i, the set for the variable x contained in C^i is equal to the output after applying s once more, so we reached a fixed-point. Set C_*^i to be the subset of C_s^i which is incident to E_{fix}^i and set E_*^i to be the relation with an (E^i, E_{fix}^i)-colored triangle incident (so E_*^i associates the input values of y with the computed fixed-point).

  If $E_{\text{continue}}^i = \emptyset$, the machine stops looping. Otherwise, the machine executes the operation `addPair`(F^i), where F^i are the edges with an $(E^i, E_{\text{continue}}^i)$-colored triangle, and obtains the vertex class C^{i+1}, the relation E^{i+1} relating a C-vertex with the corresponding C^{i+1}-vertex and the relation F_x^i for the value of x. That is, the machine combines the input variables for $\bar{y}$ with the new set for x. Now the machine continues looping.

  When the loop is finished, the machine outputs the union of the C_*^i and the union of the E_*^i is the input-output relation.

Note that we did not use a `choice`-operation and so, by Corollary 5.28, we indeed can use the induction hypothesis because we do not change the automorphisms of the input structure. Hence, the structure in the cloud is always compatible with the input structure.

We finally have to argue that the constructed choice-free DeepWL+WSC-algorithm runs in polynomial time. Let p be the polynomial given with the CPT+WSC-formula or term. In every translation step, the machine executes at most a polynomial number of steps because the number of iterations in the iteration term is bounded by p. Also, the size of the structure in the cloud is bounded by a polynomial in the size of the transitive closure of all sets encoded so far, which itself is bounded by a polynomial. Whenever a directly nested WSC-fixed-point operator is simulated, one of the simulating DeepWL+WSC-algorithms $\mathcal{M}_i$ is called a polynomial number of times. Because all the $\mathcal{M}_i$ run in polynomial time, the whole simulation runs in polynomial time. □

Lemma 5.32. *Assume that* $\Phi(\bar{z}) = \mathsf{WSC}^* xy.\, (s_{step}(\bar{z}xy), s_{choice}(\bar{z}x), s_{wit}(\bar{z}xy), \Phi_{out}(\bar{z}x))$ *is a* CPT+WSC*-formula and that, for every WSC-fixed-point operator* Φ' *directly nested in* Φ*, there is a polynomial-time DeepWL+WSC-algorithm simulating* Φ'*. Then there is a polynomial-time DeepWL+WSC-algorithm simulating* Φ*.*

Proof. We will construct DeepWL+WSC-machines M^{out} and M^{wit} in a way such that the DeepWL+WSC-algorithm $(M^{\text{out}}, M^{\text{wit}}, \mathcal{M}_1, \dots, \mathcal{M}_\ell)$ simulates Φ. The DeepWL+WSC-algorithms $\mathcal{M}_i$ are the ones simulating the directly nested WSC-fixed-point operators and will be used by choice-free DeepWL+WSC-algorithms simulating s_{step}, s_{choice}, s_{wit}, and Φ_{out}.

The machine M^{out} proceeds similarly to the plain iteration term in Lemma 5.31: For two singleton vertex classes C and D we say that M^{out} **creates a pair of** C and D when M^{out} executes `addPair` for the single edge between the C- and the D-vertex. Given a singleton vertex class C and the step term s_{step}, the machine first creates a pair of C and $C_\emptyset$ resulting in the singleton vertex class C^0. Now, in the i-th iteration, let C^i be the singleton vertex class encoding the current set in the fixed-point computation. The machine M^{out} first checks whether i exceeds the maximal number of iterations. If so, it sets $C_{\text{out}} := C^0$ (the input values paired with the empty set). Otherwise, it simulates, using Lemma 5.31, the choice term s_{choice} on input C^i yielding a singleton vertex class $C^i_{s_{\text{choice}}}$ encoding the choice-set. Let C^i_{orb} be the vertex class incident via the containment relation to the $C^i_{s_{\text{choice}}}$-vertex. Then the machine executes `choice`(C^i_{orb}) yielding a singleton vertex class C^i_{ind} (because we choose from a vertex class, we do not have to deal with the difference of the `choice`-operation for directed or undirected relations). Next, M^{out} creates a pair of C^i and C^i_{ind} and simulates the step term s_{step} (Lemma 5.31) with the newly created tuple vertex as input. This results in a singleton vertex class $C^i_{s_{\text{step}}}$. Then the machine creates again a pair of C and $C^i_{s_{\text{step}}}$ resulting in the singleton vertex class C^{i+1}. If $C^i = C^{i+1}$ (again it is important that vertices encoding the same set are not created twice), then a fixed-point is reached and the machine sets $C_{\text{out}} := C^{i+1}$. Otherwise, it starts the next iteration. Once C_{out} is computed (note that C_{out} is always a singleton vertex class because C is), M^{out} simulates Φ_{out} on input C_{out} (again using Lemma 5.31 for which we can w.l.o.g. assume that Φ_{out} is not a WSC-fixed-point operator by considering $\Phi_{\text{out}} \land \Phi_{\text{out}}$). If the simulation outputs C_{out} (so Φ_{out} is satisfied), M^{out} halts and accepts. Otherwise, M^{out} halts and does not accept. Note that the polynomial bound is not exceeded because Φ does not output $\dagger$ by assumption.

The machine M^{wit} just simulates the term s_{wit} (again by Lemma 5.31), given the vertex classes C^i and C^n, where n is the last iteration of the loop. At this point, there is a subtle difference between CPT+WSC and DeepWL+WSC. The CPT+WSC-term s_{wit} gets the i-th and n-th step of the fixed-point computation as input to witness the choices. In DeepWL+WSC, we get the labeled union $\mathfrak{A}_n \uplus \mathfrak{A}_i$ of the final content of the cloud $\mathfrak{A}_n$ and the content of the cloud $\mathfrak{A}_i$ of the i-th `choice`-execution, which is to be witnessed. In our setting, this means that $\mathfrak{A}_n \uplus \mathfrak{A}_i$ contains the vertex classes C^i and C^n but also the vertex class $C^i_{s_{\text{choice}}}$ encoding the choice-set[3]. The witnessing term s_{wit} defines automorphisms witnessing that $C^i_{s_{\text{choice}}}$ is an orbit of $\mathfrak{A}_i$ and stabilizes the C^j for all $j \in [i]$. But in

[3]If a relation named C^i is also present in $\mathfrak{A}_n$, then C^i is not a singleton vertex class in $\mathfrak{A}_n \uplus \mathfrak{A}_i$, but we can obtain the correct one by taking the C^i vertex which is also contained in the special E_2 relation given by the labeled union. We similarly process the other vertex classes.

DeepWL, also the choice-sets have to be fixed. By Lemma 5.6, the CPT+WSC term s_{choice} is isomorphism-invariant, so all the $C^j_{s_{\text{choice}}}$ are indeed fixed for all $j \in [i]$, too. That is, exactly those orbits are witnessed in the computation of the DeepWL+WSC-algorithm which are witnessed in the evaluation of the WSC-fixed-point operator (from which we assumed that all are witnessed). Arguing that the constructed DeepWL+WSC-algorithm runs in polynomial time is similar to Lemma 5.31. □

Lemma 5.33. *If a boolean query Q of a class of binary τ-structures $\mathcal{K}$-structures is* CPT+WSC*-definable, then there is a polynomial-time DeepWL+WSC-algorithm deciding Q.*

Proof. Let Φ be a CPT+WSC formula defining the query Q. In particular, Φ never outputs † on $\mathcal{K}$-structures. By applying Lemmas 5.31 and 5.32 recursively on the nesting structure of the WSC-fixed-point operators, we can translate Φ into a polynomial time and (possibly choice-free) DeepWL+WSC-algorithm $\mathcal{M}$ simulating Φ. Whenever needed, we extend choice-free DeepWL+WSC-algorithms to DeepWL+WSC-algorithms by adding a witnessing machine which immediately halts. □

5.3.4 Normalized DeepWL+WSC

For our aim to prove Theorem 5.20, we translated an isomorphism-defining CPT+WSC-formula into a polynomial time isomorphism-deciding DeepWL+WSC-algorithm in the last section. The next goal, which we address in this section, is to show that the isomorphism-deciding DeepWL+WSC-algorithm can be turned into a DeepWL+WSC-algorithm that computes a complete invariant. Recall that an isomorphism-deciding DeepWL+WSC-algorithm gets as input the disjoint union of the two connected structures, for which it has to decide whether they are isomorphic. Using the `addPair`- and `scc`-operations, the algorithms is able to create vertices composed of vertices from both structures, i.e., it mixes the structures. The ultimate goal in this section is to show that this mixing is not necessary: We will show that every DeepWL+WSC-algorithm computing on a disjoint union of structures can be simulated by a DeepWL+WSC-algorithm not mixing the two structures. We will call such non-mixing algorithms normalized. A normalized algorithm can compute on each structure separately. Exploiting this, we will obtain a complete invariant. We will follow the idea of [59] to show that every DeepWL+WSC-algorithm can be simulated by a normalized one. However, we have to differ from the construction in many points. These changes are crucial in the presence of choices.

5.3.4.1 Pure DeepWL+WSC

As first step to simulate one DeepWL+WSC-algorithm with another DeepWL+WSC-algorithm, we show that we can make some simplifying assumptions on the algorithm to be simulated. This will simplify simulating it. We adapt the notion of a pure DeepWL-algorithm from [59] to DeepWL+WSC-algorithms.

Definition 5.34 (Pure DeepWL+WSC-algorithm)**.** A DeepWL+WSC-algorithm is **pure** if `addPair`(R) and `scc`(R) are only executed for colors (or fibers) and `refine`(U, i) and `choice`(U) are only executed for fibers.

Lemma 5.35. *For every polynomial-time DeepWL+WSC-algorithm deciding a boolean query Q or computing a function f, there is a pure polynomial-time DeepWL+WSC-algorithm deciding Q or computing f.*

Proof. We show that `addPair`(X)-, `scc`(X)-, `refine`(X, i)-, and `choice`(X)-operations can be simulated by respective operations applied only to colors and fibers. The case for `addPair`(X) and `scc`(X) is similar to the proof of Lemma 7 in [59]:

`addPair`(X): First decompose X into its colors $R_1, \dots, R_k$ using the symbolic subset relation. Then execute `addPair`(R_i) for every $i \in [k]$, obtain membership relations E_i for every $i \in [k]$. Create a new relation which is the union of all E_i and serves as membership relation of the simulated `addPair`(X)-operation. This way, we create the same vertices in the cloud as the `addPair`(X)-execution would do. By Lemma 5.24, we can compute the algebraic sketch without the relations E_i.

`scc`(X): By Lemma 4 of [59], we can compute a relation which exactly contains the pairs of vertices in the same X-SCC using a pure DeepWL-algorithm (and this algorithm only executes `create` and no other operations, so can be directly transferred into our DeepWL-definition). If there is an X-SCC which is not discrete, i.e., it contains at least two vertices from the same fiber, we can identify a color R refining X, such that the R-SCCs are nontrivial. We execute `scc`(R) and obtain new vertices and a new membership relation F. Next, we consider the relation X', which consists of

(a) the edges of X not incident to an R-SCC and

(b) the edges (u, w) for which there is an edge (u, v) in X such that v is contained in an R-SCC and w is the R-SCC vertex for v, that is, v is adjacent to w via the membership relation F.

The relation X' is a union of colors, which can be identified using the symbolic subset relation. We create X' using `create` for the appropriate set of colors. If the X'-SCCs are still not all discrete, we repeat the procedure.

Now consider the case that all X-SCCs are discrete. Conceptually, we want to pick from every X-SCC the vertex in the minimal fiber U as a representative of that SCC. We create a copy of the U-vertices by executing `addPair`(U). While iteratively creating the SCC-vertices, we maintain a relation which associates the SCC-vertices to the vertices in the X-SCCs, for which they were created. Using this relation, we can define the membership relation.

So, to simulate `scc`(X), we do not create exactly the same vertex (as HF-set) because we pick the minimal fibers as representative. Regarding witnessing automorphisms, neither picking only a representative nor creating the intermediate SCC-vertices makes a difference: Because we did not use a `choice`-operation to simulate the `refine`-operation, the structure in the cloud has the same automorphisms as the structure that is simulated by Corollary 5.28. Again by Lemma 5.24, we can compute the algebraic sketch without the additional vertices and relations.

`refine`(X,k): Let X consist of the colors $R_1,\dots,R_m$, which we find using the symbolic subset relation. We execute `refine`$(R_1,k),\dots,$`refine`(R_m,k) and obtain new relations $E_1,\dots,E_m$. The union of these relations precisely corresponds to the relation outputted by `refine`(X,k). So we can assume that X is a color R. If R is not a fiber, then we do the following: If R is an undirected color, then execute `addUPair`(R) and obtain a new vertex class C and a membership relation F. We decompose C into its fibers $U_1,\dots,U_\ell$, execute `refine`(U_i,k) for every $i\in[\ell]$, and obtain the vertex classes $D_1,\dots,D_\ell$ (in case `refine`(U_i,k) does not create a vertex class, i.e., the algorithm used to refine U_i accepts all vertices in U_i, we set $D_i := U_i$). Let D be the union of the D_i for all $i\in[\ell]$. The relation E refining R is obtained as the set of R-edges with an (F^{-1},C,F)-colored path. If E contains the same edges as R, then we do nothing, otherwise E is the output of the simulated `refine`-operation.

If R is a directed color, then execute `addPair`(R) and obtain the vertex class C and the component relations F_{left} and F_{right}. Next, we execute `refine`(C,k) (for which we again decompose C into fibers), and obtain the vertex class D refining C. Similarly to the unordered case, we obtain the relation E refining R as R-edges with an $(F^{-1}_{\text{left}},D,F_{\text{right}})$-colored path. If E and R contain the same edges, we do nothing, otherwise E is the output of the simulated `refine`-operation.

The algorithm used to refine has to be altered to also perform the same operations to first create a new relation corresponding to the individualized D-vertex. Again by Corollary 5.28, the simulation does not alter the automorphisms of the structure in the cloud and by Lemma 5.24, the algebraic sketch without the additional vertices and relations can be computed.

`choice`(X): If `choice`(X) is executed for a relation E consisting of more than one color, then E is not an orbit and the given algorithm does not decide a query or computes a function because it is going to fail (recall that failing is not allowed when deciding a query or computing a function). So we can assume that X is a color R. Analogously to the `refine`-case, we execute `addUPair`(R) or `addPair`(R), depending on whether R is undirected or not, and obtain a vertex class C. Next, we execute `choice`(C) yielding a singleton vertex class containing a C-vertex u. Using the vertex u, we can define a relation containing a single (directed or undirected) R-edge analogously as in the `refine`-case.

It is easy to see that a set of automorphisms witnesses R (seen as directed or undirected edges depending on whether R is directed or not) as orbit if and only if it witnesses C as orbit. Additionally, both individualizing an R-edge and individualizing the corresponding C-vertex results in an HF-structure with the same automorphisms. By Lemma 5.24, we can compute the algebraic sketch without the additional vertices and relations. $\square$

5.3.4.2 Normalized DeepWL+WSC

We now formalize the notion of an DeepWL+WSC-algorithm not mixing the two connected components of the structure in the cloud. To do so, we adapt the notion of a normalized DeepWL-algorithm from [59] to DeepWL+WSC. Recall that, for the aim of testing isomorphisms, the input structure is the disjoint union of two connected structures.

Definition 5.36. (Normalized HF-Structure) An HF-structure $\mathfrak{A}$ is **normalized** if there are connected HF-structures $\mathfrak{A}_1$ and $\mathfrak{A}_2$ such that $\mathfrak{A} = \mathfrak{A}_1 \uplus \mathfrak{A}_2$. The $\mathfrak{A}_i$ are the **components** of $\mathfrak{A}$. The vertices of a normalized structure are called **plain**. The edges

$$\mathsf{E}_{\text{plain}}(\mathfrak{A}) := \mathsf{V}_1(\mathfrak{A})^2 \cup \mathsf{V}_2(\mathfrak{A})^2$$

are called **plain** and the edges

$$\mathsf{E}_{\text{cross}}(\mathfrak{A}) := \mathsf{V}_1(\mathfrak{A}) \times \mathsf{V}_2(\mathfrak{A}) \cup \mathsf{V}_2(\mathfrak{A}) \times \mathsf{V}_1(\mathfrak{A})$$

are called **crossing**. A relation or color is called plain (or crossing) if it only contains plain (or crossing, respectively) edges.

Observe that also the atom set of $\mathfrak{A}$ is the disjoint union $A = A_1 \uplus A_2$ and that the components $\mathfrak{A}_1$ and $\mathfrak{A}_2$ of $\mathfrak{A}$ are unique because the $\mathfrak{A}_i$ are connected. Also observe that if a DeepWL+WSC-machine is executed on $\mathfrak{A}_1 \uplus \mathfrak{A}_2$ and at some point during its execution the structure in the cloud is still normalized, then every vertex w is contained in $\mathsf{HF}(A_i)$ for some $i \in [2]$. That is, one of the following holds:

- $w \in \mathsf{V}(\mathfrak{A}_i)$,
- w was added by an `addPair`-execution for a pair (u, v) of vertices $u, v \in \mathsf{HF}(A_i)$, or
- w was obtained as a vertex for an SCC $c \subseteq \mathsf{HF}(A_i)$ during an `scc`-execution.

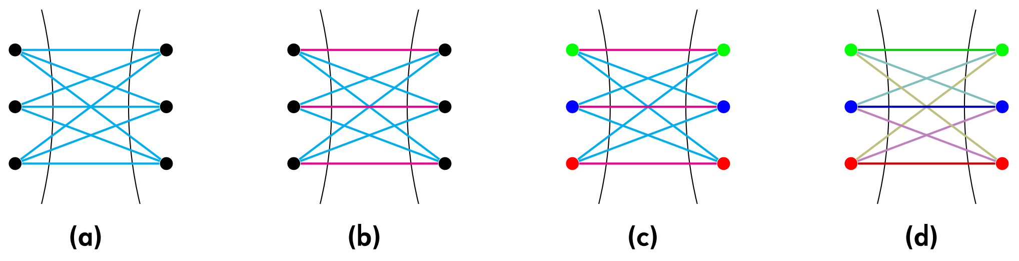
In particular, `addPair`(X)- and `scc`X-operations were only executed for plain X.

Lemma 5.37 ([59]). *The relation of all plain (respectively crossing) edges is DeepWL-computable.*

Definition 5.38 (Normalized DeepWL+WSC). A DeepWL+WSC-machine M is **normalized** if, for every normalized HF-structure $\mathfrak{A}$, the structure in the cloud of M is normalized at every point in time during the execution of M on $\mathfrak{A}$. A DeepWL+WSC-algorithm $\mathcal{M} = (M^{\text{out}}, M^{\text{wit}}, \mathcal{M}_1, \dots, \mathcal{M}_\ell)$ is normalized if M^{out}, M^{wit}, and $\mathcal{M}_1, \dots, \mathcal{M}_\ell$ are normalized.

The current definition has a severe issue, namely the way in which sets of witnessing automorphisms are encoded by the witnessing machine: With the encoding of automorphisms we previously described, a set of automorphisms can only be encoded by non-plain vertices. So normalized DeepWL+WSC-algorithms need to use a different encoding of automorphisms to witness choices. For the moment, we do not need to give a precise definition for this encoding because it is irrelevant for the following lemmas. We will thus define the encoding later in Section 5.3.4.6 once we have established the required formalism.

Normalized structures have the important property that every crossing color is actually a "direct product" of two fibers. That is, crossing colors do not provide additional information and this is the reason why general DeepWL+WSC-algorithms can ultimately be simulated by normalized ones.



5.4 Illustration of `refine`-operations and the "direct product" property. This figure shows an example for the problem with `refine`-operations and the "direct product" property. Figure (a) shows a coherent structure $\mathfrak{A}_1 \uplus \mathfrak{A}_2$ with a black fiber satisfying the "direct product" property, i.e., all edges between black vertices are light blue. Assume that in each component the black vertices can be ordered. Figure (b) shows the structure obtained after refining the light blue color with an algorithm accepting exactly the edges between the i-th black vertices in both structures. This yields a red color. The structure is again coherent but does not have the "direct product" property. Ordering the vertices is not passed through the `refine`-operation. Figure (c) shows the structure resulting after first ordering the black vertices and then executing the same `refine`-operation. This structure has the "direct product" property: Figure (d) shows the decomposition of the light blue relation and the red relation into colors.

Lemma 5.39 ([59, Lemma 8]). *Let $\mathfrak{A} = \mathfrak{A}_1 \uplus \mathfrak{A}_2$ be a normalized HF-structure.*

1. *For every crossing color R of $\mathfrak{A}$, there are two plain fibers U and V such that $R^{\mathfrak{A}} = (U^{\mathfrak{A}} \times V^{\mathfrak{A}}) \cap \mathsf{E}_{cross}(\mathfrak{A})$.*

2. *$C(\mathfrak{A}_i)$ is equal to $C(\mathfrak{A})[\mathsf{V}(\mathfrak{A}_i)] \upharpoonright \sigma_i$ up to renaming colors for every $i \in [2]$, where σ_i is the set of all colors $R \in \sigma$ for which $R^{\mathfrak{A}} \cap \mathsf{V}(\mathfrak{A}_i)^2 \neq \emptyset$.*

3. *There is a polynomial-time algorithm that on input $D(\mathfrak{A}_1)$ and $D(\mathfrak{A}_2)$ computes $D(\mathfrak{A})$.*

4. *The two sketches $D(\mathfrak{A}_1)$ and $D(\mathfrak{A}_2)$ determine $D(\mathfrak{A})$: if $D(\mathfrak{A}) \neq D(\mathfrak{B})$ for another HF-structure $\mathfrak{B} = \mathfrak{B}_1 \uplus \mathfrak{B}_2$, then $\{D(\mathfrak{A}_1), D(\mathfrak{A}_2)\} \neq \{D(\mathfrak{B}_1), D(\mathfrak{B}_2)\}$.*

5.3.4.3 The "Direct Product" Property and `refine`-Operations

We now develop tools to deal with normalized DeepWL+WSC-algorithms. First, we investigate the `refine`-operation. Intuitively, our goal is to preserve the "direct product" property for crossing colors as stated in Lemma 5.39. In principle, if we only create plain vertices, this property is preserved, but a `refine`-operation can violate it.

We give an illustrating example (cf. Figure 5.4): Assume we are given a normalized HF-structure $\mathfrak{A} = \mathfrak{A}_1 \uplus \mathfrak{A}_2$, both components of the same size, where all vertices are in the same fiber and consequently all crossing edges are in the same color R (Figure 5.4a) Further, assume that some DeepWL+WSC-algorithm can linearly order the vertices in each component. Then we can define an algorithm $\mathcal{M}$ with the following property. For $\{u, v\} \subseteq R^{\mathfrak{A}}$, the algorithm accepts $(\mathfrak{A}_1 \uplus \mathfrak{A}_2, \{u, v\})$ if and only if u and v are each the i-th vertex in their component for some i. If we refine R with $\mathcal{M}$, then a relation E containing a perfect matching between the components is added (Figure 5.4b). But the information, that the atoms can be ordered, is not "returned". The result is again a coherent configuration with only one fiber and so the "direct product" property is violated. To avoid this problem, we need to distinguish the atoms first and then execute the `refine`-operation (Figure 5.4c).

We will show now that this strategy generalizes to arbitrary `refine`-operations. To do so, we need to establish the following technical lemmas. The first lemma states that instead of computing on $\mathfrak{A}_1 \uplus \mathfrak{A}_2$, we can also compute on the structure $(\mathfrak{A}_1 \uplus \mathfrak{A}_2, A_1)$ (that is, we add a vertex class containing all vertices of $\mathfrak{A}_1$), where we artificially distinguish the two components, i.e., we remove automorphisms exchanging the components.

Lemma 5.40. *For every normalized DeepWL+WSC-algorithm $\mathcal{M}$, there exists a normalized algorithm $\hat{\mathcal{M}}$ such that, for all connected HF-structures $\mathfrak{A}_1$ and $\mathfrak{A}_2$, the algorithm $\mathcal{M}$ accepts (respectively rejects) $\mathfrak{A}_1 \uplus \mathfrak{A}_2$ if and only if $\hat{\mathcal{M}}$ accepts (respectively rejects) $(\mathfrak{A}_1 \uplus \mathfrak{A}_2, A_1)$. Polynomial running time is preserved. For all HF-structures $\mathfrak{A}_1$, $\mathfrak{A}_2$, $\mathfrak{B}_1$, and $\mathfrak{B}_2$, it holds that*

$$\textit{if } \mathsf{run}(\mathcal{M}, \mathfrak{A}_1 \uplus \mathfrak{A}_2) \neq \mathsf{run}(\mathcal{M}, \mathfrak{B}_1 \uplus \mathfrak{B}_2),$$
$$\textit{then } \mathsf{run}(\hat{\mathcal{M}}, (\mathfrak{A}_1 \uplus \mathfrak{A}_2, A_1)) \neq \mathsf{run}(\hat{\mathcal{M}}, (\mathfrak{B}_1 \uplus \mathfrak{B}_2, B_1)).$$

Proof. The proof is by induction on the nesting depth of DeepWL+WSC-algorithms. Let $\mathcal{M} = (M^{\text{out}}, M^{\text{wit}}, \mathcal{M}_1, \dots, \mathcal{M}_\ell)$ be a normalized DeepWL+WSC-algorithm. By the induction hypothesis, let $\hat{\mathcal{M}}_1, \dots, \hat{\mathcal{M}}_\ell$ be normalized DeepWL+WSC-algorithms satisfying the claim for the algorithms $\mathcal{M}_1, \dots, \mathcal{M}_\ell$. We construct DeepWL+WSC-machines $\hat{M}^{\text{out}}$ and $\hat{M}^{\text{wit}}$, i.e., for $M \in \{M^{\text{out}}, M^{\text{wit}}\}$ we construct a DeepWL+WSC-machine $\hat{M}$. Let $\mathfrak{A}_1 \uplus \mathfrak{A}_2$ be a normalized HF-structure on which M is executed. By Lemma 5.24, $\hat{M}$ can compute the algebraic sketch $D(\mathfrak{A}_1 \uplus \mathfrak{A}_2)$ from $D((\mathfrak{A}_1 \uplus \mathfrak{A}_2, A_1))$ and thus track the run of M. Note that by Lemma 5.39, the sketches of the individual components in $D(\mathfrak{A}_1 \uplus \mathfrak{A}_2)$ and $D((\mathfrak{A}_1 \uplus \mathfrak{A}_2, A_1))$ are equal possibly up to renaming of the colors. Additionally, the crossing colors of $(\mathfrak{A}_1 \uplus \mathfrak{A}_2, A_1)$ are always directed.

If M executes `create`(π) and π contains a color R occurring in both components, then $\hat{M}$ uses the two colors R_1 and R_2, each occurring in one component, whose union is equal to R. Every `addPair`-, `scc`-, and `refine`-operation executed by M is executed in the same way by $\hat{M}$ respecting the renamed colors: Whenever M uses a color or relation, which is now split into two sets (one in each component), $\hat{M}$ creates the union of these colors/relations and uses this union. The `refine`-operations yield the correct result by the induction hypothesis. To continue to track the run of M, we again exploit Lemma 5.24 to compute the sketch without the additionally created relations.

Whenever M executes `choice`(R) (if `choice` is executed for a relation consisting of multiple relations, then the algorithm will fail), we make the following case distinction. Let $\mathfrak{A}' = (\mathfrak{A}'_1 \uplus \mathfrak{A}'_2, A_1)$ be the current content of the cloud. If R is a crossing color, $\hat{M}$ executes `choice`(S) for the color S satisfying $S^{\mathfrak{A}'} = R^{\mathfrak{A}'} \cap (A'_1 \times A'_2)$ (which is DeepWL-computable). If R was a directed color, then there is nothing more to do because the individualized edge (u, v) is contained in $R^{\mathfrak{A}'}$. If R was an undirected color, then a directed edge (u, v) instead of the undirected edge $\{u, v\}$ is individualized. Recall that (u, v) is individualized by creating two singleton vertex classes and that $\{u, v\}$ is individualized by creating a two-element vertex class. So we create the union of the two singleton color classes. This corresponds to individualizing the undirected edge and we can continue to track the run of M.

If R is a plain color containing edges of both components, then $\hat{M}$ executes $\texttt{choice}(S)$ for the color S satisfying $S^{\mathfrak{A}'} = R^{\mathfrak{A}'} \cap (A_1')^2$. Otherwise, R is a plain color containing only edges of one component and $\hat{M}$ simply executes $\texttt{choice}(R)$. It is easy to see that if R is an orbit of $\mathfrak{A}_1' \uplus \mathfrak{A}_2'$, then S is an orbit of $(\mathfrak{A}_1' \uplus \mathfrak{A}_2', A_1)$. Again by Lemma 5.24, we compute the sketch without the additional created relations to track the run of M.

We finally have to modify the witnessing machine $\hat{M}^{\mathrm{wit}}$. Independent of the precise way we will encode sets of witnessing automorphisms for normalized DeepWL+WSC-machines, we may proceed as follows (an explicit description of the encoding is given in Section 5.3.4.6): The witnessing machine $\hat{M}^{\mathrm{wit}}$ first computes the set of automorphisms given by M^{wit}. Then it removes the automorphisms exchanging the two components and outputs the set of remaining automorphisms. This way, all choices are witnessed if all choices in the execution of $\mathcal{M}$ are witnessed.

Finally, assume $\mathfrak{A}_1, \mathfrak{A}_2, \mathfrak{B}_1$, and $\mathfrak{B}_2$ are connected HF-structures. Furthermore, suppose $\mathsf{run}(\mathcal{M}, \mathfrak{A}_1 \uplus \mathfrak{A}_2) \neq \mathsf{run}(\mathcal{M}, \mathfrak{B}_1 \uplus \mathfrak{B}_2)$. The machine $\hat{M}$ tracks the run of M. Hence, the run of $\hat{M}$ contains the configurations of the Turing machines in M and also the algebraic sketches in the run of M. These sketches were computed by $\hat{M}$ using Lemma 5.24. So $\mathsf{run}(\hat{\mathcal{M}}, \mathfrak{A}_1 \uplus \mathfrak{A}_2)$ can be extracted from $\mathsf{run}(\hat{\mathcal{M}}, (\mathfrak{A}_1 \uplus \mathfrak{A}_2, A_1))$ (and likewise for $\mathfrak{B}_1 \uplus \mathfrak{B}_2$). We conclude that $\mathsf{run}(\hat{\mathcal{M}}, (\mathfrak{A}_1 \uplus \mathfrak{A}_2, A_1)) \neq \mathsf{run}(\hat{\mathcal{M}}, (\mathfrak{B}_1 \uplus \mathfrak{B}_2, B_1))$. □

We now identify sets of runs as a possibility to distinguish vertices, so that executing a `refine`-operation on a crossing relation preserves the "direct product" property.

Lemma 5.41. *For every normalized DeepWL+WSC-algorithm $\mathcal{M}$, every normalized HF-structure $\mathfrak{A} = \mathfrak{A}_1 \uplus \mathfrak{A}_2$, every crossing color $R^{\mathfrak{A}} = (U^{\mathfrak{A}} \times V^{\mathfrak{A}}) \cap \mathsf{E}_{cross}(\mathfrak{A})$ for some fibers U and V, every vertex $u \in U^{\mathfrak{A}} \cap \mathsf{V}_i(\mathfrak{A})$ for some $i \in [2]$, and all edges $\{u,v\}, \{u,v'\} \in R^{\mathfrak{A}} \cup (R^{-1})^{\mathfrak{A}}$, we have the following: If $\mathcal{M}$ accepts $(\mathfrak{A}, \{u,v\})$ and $\mathcal{M}$ does not accept $(\mathfrak{A}, \{u,v'\})$, then*

$$\Big\{ \mathsf{run}(\mathcal{M}, (\mathfrak{A}, \{w,v\})) \,\Big|\, w \in U \cap \mathsf{V}_i(\mathfrak{A}) \Big\} \neq \Big\{ \mathsf{run}(\mathcal{M}, (\mathfrak{A}, \{w,v'\})) \,\Big|\, w \in U \cap \mathsf{V}_i(\mathfrak{A}) \Big\}.$$

Proof. Let $\mathcal{M}$ be a normalized DeepWL+WSC-algorithm, $\mathfrak{A} = \mathfrak{A}_1 \uplus \mathfrak{A}_2$ be a normalized HF-structure, and assume R, $u \in U^{\mathfrak{A}} \cap \mathsf{V}_i(\mathfrak{A})$, $\{u,v\}$, and $\{u',v'\}$ satisfy the conditions of the lemma. Assume $\mathcal{M}$ accepts $(\mathfrak{A}, \{u,v\})$ and $\mathcal{M}$ does not accept $(\mathfrak{A}, \{u,v'\})$. We prove that $\mathsf{run}(\mathcal{M}, (\mathfrak{A}, \{u,v\})) \notin \{\mathsf{run}(\mathcal{M}, (\mathfrak{A}, \{w,v'\})) \mid w \in U^{\mathfrak{A}} \cap \mathsf{V}_i(\mathfrak{A})\}$. Suppose, for the sake of contradiction, that there exists a vertex $u' \in U^{\mathfrak{A}} \cap \mathsf{V}_i(\mathfrak{A})$ such that $\mathsf{run}(\mathcal{M}, (\mathfrak{A}, \{u,v\})) = \mathsf{run}(\mathcal{M}, (\mathfrak{A}, \{u',v'\}))$. At some point during the execution of $\mathcal{M}$, the runs $\mathsf{run}(\mathcal{M}, (\mathfrak{A}, \{u',v'\}))$ and $\mathsf{run}(\mathcal{M}, (\mathfrak{A}, \{u,v'\}))$ have to differ because $\mathsf{run}(\mathcal{M}, (\mathfrak{A}, \{u,v\})) = \mathsf{run}(\mathcal{M}, (\mathfrak{A}, \{u',v'\}))$ is accepting and $\mathsf{run}(\mathcal{M}, (\mathfrak{A}, \{u,v'\}))$ is not accepting. Because $\mathcal{M}$ is normalized, at every point in time the current content of the cloud $\mathfrak{B}$ satisfies $\mathsf{V}(\mathfrak{B}) = \mathsf{V}_1(\mathfrak{B}) \cup \mathsf{V}_2(\mathfrak{B})$. Before the first moment where the two runs differ, the component of v' is equal in both clouds because the same operations were executed and the initial components of v' were equal (in $(\mathfrak{A}, \{u,v'\})$ and in $(\mathfrak{A}, \{u',v'\})$ the same vertex is individualized in this component). In particular, the components have the same sketches. So $\mathcal{M}$ can only have a run on $\{u',v'\}$ that is different from its run on

$\{u, v'\}$ if the sketches of the components of u and u' differ (Lemma 5.39). But this contradicts that $\mathsf{run}(\mathcal{M}, (\mathfrak{A}, \{u, v\})) = \mathsf{run}(\mathcal{M}, (\mathfrak{A}, \{u', v'\}))$ because the sketches are contained in the run. $\square$

Now we want to compute the set $\{\mathsf{run}(\mathcal{M}, (\mathfrak{A}, \{w, v\})) \mid w \in U \cap \mathsf{V}_i(\mathfrak{A})\}$ using a DeepWL+WSC-algorithm. We start with the case where a directed edge (w, v) instead of the undirected one $\{w, v\}$ is individualized.

Lemma 5.42. *For every normalized DeepWL+WSC-algorithm $\mathcal{M}$, every normalized HF-structure $\mathfrak{A} = \mathfrak{A}_1 \uplus \mathfrak{A}_2$, every crossing color $R^{\mathfrak{A}} = (U^{\mathfrak{A}} \times V^{\mathfrak{A}}) \cap \mathsf{E}_{cross}(\mathfrak{A})$ for some fibers U and V, all numbers i and j such that $\{i, j\} = [2]$, and every vertex $v \in V^{\mathfrak{A}} \cap \mathsf{V}_j(\mathfrak{A})$, there is a normalized DeepWL+WSC-algorithm computing on input v (given as a singleton vertex class) the set*

$$\left\{ \mathsf{run}(\mathcal{M}, (\mathfrak{A}, wv)) \;\middle|\; w \in U \cap \mathsf{V}_i(\mathfrak{A}) \right\}$$

if, for every $(u, v) \in R^{\mathfrak{A}}$, the algorithm $\mathcal{M}$ does not fail on input $(\mathfrak{A}, uv)$. Polynomial runtime of the algorithm is preserved.

Proof. Let $\mathcal{M}$ be a normalized DeepWL+WSC-algorithm. For $\mathcal{M}$, let $\mathcal{M}'$ be a normalized DeepWL+WSC-algorithm, which on input i, v, and w (given as singleton vertex classes) simulates $\mathcal{M}$ and decides whether the i-th position of the run $\mathsf{run}(\mathcal{M}, (\mathfrak{A}, wv))$ is a 1 if $\mathcal{M}$ does not fail. Similarly, let $\mathcal{M}''$ be a normalized DeepWL+WSC-algorithm, which on input i, v, and w decides whether $|\mathsf{run}(M, (\mathfrak{A}, wv))| \geq i$ unless M fails.

Let $\mathfrak{A} = \mathfrak{A}_1 \uplus \mathfrak{A}_2$ be a normalized HF-structure, $R^{\mathfrak{A}} = (U^{\mathfrak{A}} \times V^{\mathfrak{A}}) \cap \mathsf{E}_{\text{cross}}(\mathfrak{A})$ be a crossing color and $v \in V^{\mathfrak{A}} \cap \mathsf{V}_j(\mathfrak{A})$ for some $\{i, j\} = [2]$ such that $\mathcal{M}$ does not fail on input $(\mathfrak{A}, uv)$ for every $(u, v) \in R^{\mathfrak{A}}$. We construct a machine which computes the runs $\mathsf{run}(M, (\mathfrak{A}, wv))$ in parallel for all $w \in U^{\mathfrak{A}}$. The runs are encoded as binary 0/1-strings, which are encoded by vertex classes $C_1, \dots, C_m$, and $D_1, \dots, D_m$, where C_i contains all U-vertices for which the i-th bit of the run is a 1, and D_i contains all U-vertices for which the run has length at least i. Initialize $D_0 := U$, that is, D_0 contains exactly the U-vertices on which the run of $\mathcal{M}$ has length at least zero. Starting from $i = 0$, we first determine the vertex class D_{i+1} by executing $\texttt{refine}(D_i, \mathcal{M}'', i + 1)$ (we actually have to execute $\texttt{refine}(D_i, k, i + 1)$, where k is the index of $\mathcal{M}''$ in the list of nested subalgorithms). By definition of $\mathcal{M}''$, D_{i+1} contains exactly the U-vertices, on which the run of $\mathcal{M}$ has length at least i. If $D_{i+1} = \emptyset$, then we stop because we computed all runs. Otherwise, we determine C_{i+1} as a subset of D_{i+1} by executing $\texttt{refine}(D_{i+1}, \mathcal{M}', i + 1)$ (again, we need to replace $\mathcal{M}'$ by the corresponding number). By definition of $\mathcal{M}'$, the class C_{i+1} contains exactly the U-vertices for which the $(i + 1)$-th bit in the run of M is a 1. Finally, we increment i and repeat. We now have encoded the runs $\mathsf{run}(M, (\mathfrak{A}, wv))$ with the vertex classes $C_1, \dots, C_m$ and $D_1, \dots, D_m$. So we can determine which runs occur. Now it is easy to compute the set $\{\mathsf{run}(\mathcal{M}, (\mathfrak{A}, wv)) \mid w \in U\}$ and to write it onto the tape because the runs can be ordered lexicographically.

Regarding the runtime, its easy to see that $\mathcal{M}'$ and $\mathcal{M}''$ run in polynomial time because they just simulate $\mathcal{M}$ and check the run of $\mathcal{M}$. Because the length of the run of $\mathcal{M}$ is bounded by a polynomial, so is the number of iterations and the number of needed relations. That is, also the size of the algebraic sketch is bounded by a polynomial. $\square$

Lemma 5.43. *For every normalized DeepWL+WSC-algorithm $\mathcal{M}$, every normalized HF-structure $\mathfrak{A} = \mathfrak{A}_1 \uplus \mathfrak{A}_2$, every crossing color $R^{\mathfrak{A}} = (U^{\mathfrak{A}} \times V^{\mathfrak{A}}) \cap \mathsf{E}_{cross}(\mathfrak{A})$ for some fibers U and V, and every $i \in [2]$, we have the following: There exists a normalized DeepWL+WSC-algorithm computing a function f such that if, for every $(u,v) \in R^{\mathfrak{A}}$, the algorithm $\mathcal{M}$ does not fail on input $(\mathfrak{A}, uv)$, then*

$$f(v) \neq f(v') \text{ whenever}$$
$$\left\{ \mathsf{run}(\mathcal{M}, (\mathfrak{A}, \{w, v\})) \;\middle|\; w \in U \cap \mathsf{V}_i(\mathfrak{A}) \right\} \neq \left\{ \mathsf{run}(\mathcal{M}, (\mathfrak{A}, \{w, v'\})) \;\middle|\; w \in U \cap \mathsf{V}_i(\mathfrak{A}) \right\}$$

for all $v, v' \in V^{\mathfrak{A}} \cap \mathsf{V}_j(\mathfrak{A})$ such that $\{i, j\} = [2]$ (the input vertex to f is given as singleton vertex class). Polynomial runtime is preserved.

Proof. Let $\mathcal{M}$ be a normalized DeepWL+WSC-algorithm and let $\hat{\mathcal{M}}$ be the normalized DeepWL+WSC-algorithm given by Lemma 5.40 for $\mathcal{M}$, that is, $\hat{\mathcal{M}}$ accepts $(\mathfrak{A}, uv)$ if and only if $\mathcal{M}$ accepts $(\mathfrak{A}, \{u, v\})$ for every $(u,v) \in R^{\mathfrak{A}}$ and every crossing color $R^{\mathfrak{A}} = (U^{\mathfrak{A}} \times V^{\mathfrak{A}}) \cap \mathsf{E}_{\mathrm{cross}}(\mathfrak{A})$ of every normalized HF-structure $\mathfrak{A}$. Note here that individualizing u implicitly distinguishes the two components and we do not need to create a vertex class for A_1. Given v as singleton vertex class, we compute using Lemma 5.42 the set $f(v) := \{\mathsf{run}(\hat{\mathcal{M}}, (\mathfrak{A}, wv)) \mid w \in U \cap \mathsf{V}_i(\mathfrak{A})\}$. By the properties of $\hat{\mathcal{M}}$ granted by Lemma 5.40, it follows that

$$\text{if } \left\{ \mathsf{run}(\mathcal{M}, (\mathfrak{A}, \{w, v\})) \;\middle|\; w \in U \cap \mathsf{V}_i(\mathfrak{A}) \right\} \neq \left\{ \mathsf{run}(\mathcal{M}, (\mathfrak{A}, \{w, v'\})) \;\middle|\; w \in U \cap \mathsf{V}_i(\mathfrak{A}) \right\},$$
$$\text{then } \left\{ \mathsf{run}(\hat{\mathcal{M}}, (\mathfrak{A}, (w, v))) \;\middle|\; w \in U \cap \mathsf{V}_i(\mathfrak{A}) \right\} \neq \left\{ \mathsf{run}(\hat{\mathcal{M}}, (\mathfrak{A}, (w, v'))) \;\middle|\; w \in U \cap \mathsf{V}_i(\mathfrak{A}) \right\}.$$

□

Finally, we want to use this function f to refine vertex classes. That is, we want to execute a `refine`(C, f)-operation, which splits a vertex class C such that two C-vertices u and v end up in different classes if and only if $f(u) \neq f(v)$.

Lemma 5.44. *Let $\mathcal{M}$ be a normalized and polynomial-time DeepWL+WSC-algorithm, which computes a function f. Then we can simulate a* `refine`(C, f)*-execution using a normalized polynomial-time DeepWL+WSC-algorithm.*

Proof. The proof is similar to the one of Lemma 5.42. We use vertex classes encoding the values of f. We create a DeepWL+WSC-algorithm $\mathcal{M}'$, which, for every normalized structure $\mathfrak{A}$ and every vertex class C, takes a C-vertex u (as singleton vertex class) and an additional number i as input. It decides whether the i-th bit of $f(u)$ is 1. We create another algorithm $\mathcal{M}''$ that takes a C-vertex u and a number i as input and decides whether $|f(u)| \geq i$. Use $\mathcal{M}'$ and $\mathcal{M}''$ iteratively to obtain vertex classes refining C encoding $f(u)$ for every C-vertex u analogously to the proof of Lemma 5.42. We then create new vertex classes for distinct output of f containing the C-vertices of that value. Because f is computed by a polynomial-time machine, the length of f is bounded by a polynomial and so again is the number of iterations and created relations. □

In Section 5.3.4.6, we will use the lemmas of this section to simulate a `refine`-operation as already discussed: First split the vertex classes suitably and then execute the `refine`-operation.

5.3.4.4 Building Plans

To simulate arbitrary DeepWL+WSC-algorithms by normalized ones, we need to compute the algebraic sketch of the non-normalized HF-structure in the cloud of the simulated algorithm from the normalized HF-structure in the cloud of the simulating algorithm. For this, we introduce the notion of a building plan.

Definition 5.45 (Building Plan). Let $\mathfrak{A} = \mathfrak{A}_1 \uplus \mathfrak{A}_2$ be a normalized HF-structure. A **vertex plan** for $\mathfrak{A}$ is a set $\{C, D\}$ of vertex classes $C, D \in \tau$. A **relation plan** for $\mathfrak{A}$ defining a relation symbol $E \notin \tau$ is a pair $(E, \{\{F_{1,i}, F_{2,i}\} \mid i \in [k]\})$, where $F_{j,i} \in \tau$ for all $j \in [2]$ and $i \in [k]$. A **building plan** for $\mathfrak{A}$ is a pair $\Omega = (\omega, \kappa)$, where ω is a finite set of vertex plans and κ is a finite set of relation plans where each relation plan defines a different relation symbol.

Intuitively, a building plan Ω describes a non-normalized HF-structure $\Omega(\mathfrak{A})$ in terms of a normalized HF-structure $\mathfrak{A}$ and a recipe to construct vertices mixing the two components of $\mathfrak{A}$. A vertex plan $\{C, D\}$ says that, for every $u \in C^{\mathfrak{A}}$ and $v \in D^{\mathfrak{A}}$ in different components of $\mathfrak{A}$, the vertex $\{u, v\}$ is added to the structure (recall that we are defining an HF-structure). A relation plan $(E, \{\{F_{1,i}, F_{2,i}\} \mid i \in [k]\})$ specifies a new relation E between the created vertices: A pair $(\{u, v\}, \{u', v'\})$ is contained in $E^{\Omega(\mathfrak{A})}$ if and only if there is some $i \in [k]$ such that $(u, u') \in F_{1,i}^{\mathfrak{A}}$ and $(v, v') \in F_{2,i}^{\mathfrak{A}}$. Lastly, a special relation E_p relating every vertex $\{u, v\}$ that is created to the original vertices u and v is added. Formally, the structure $\Omega(\mathfrak{A})$ is defined as follows:

Definition 5.46 (Structure Defined by a Building Plan). Let $\Omega = (\omega, \kappa)$ be a building plan for a normalized HF-structure $\mathfrak{A}$. The structure $\Omega(\mathfrak{A})$ is defined as follows: For every vertex plan $\{C, D\} \in \omega$, define

$$\begin{aligned} E^{\mathfrak{A}}_{\{C,D\}} &:= (C^{\mathfrak{A}} \times D^{\mathfrak{A}} \cup D^{\mathfrak{A}} \times C^{\mathfrak{A}}) \cap \mathsf{E}_{\text{cross}}(\mathfrak{A}), \\ \tilde{E}^{\mathfrak{A}}_{\{C,D\}} &:= \Big\{ \{u, v\} \;\Big|\; (u, v) \in E^{\mathfrak{A}}_{\{C,D\}} \Big\}, \end{aligned}$$

and $\tilde{E}^{\mathfrak{A}}_{\omega} := \bigcup_{\{C,D\}\in\omega} \tilde{E}^{\mathfrak{A}}_{\{C,D\}}$. The atoms of $\Omega(\mathfrak{A})$ are the ones of $\mathfrak{A}$, the vertices are $\mathsf{V}(\Omega(\mathfrak{A})) := \mathsf{V}(\mathfrak{A}) \cup \tilde{E}^{\mathfrak{A}}_{\omega}$, and the signature of $\Omega(\mathfrak{A})$ is $\tau \uplus \{E_p\} \uplus \{E \mid (E, M) \in \kappa\}$. The relations are defined via

$$\begin{aligned} E_p^{\Omega(\mathfrak{A})} &:= \bigcup_{\{u,v\}\in\tilde{E}^{\mathfrak{A}}_{\omega}} \Big\{(\{u, v\}, u), (\{u, v\}, v)\Big\}, \\ E^{\Omega(\mathfrak{A})} &:= \Big\{ (\{u, v\}, \{u', v'\}) \;\Big|\; (u, u') \in F^{\mathfrak{A}}_{1,i}, (v, v') \in F^{\mathfrak{A}}_{2,i}, i \in [k] \Big\} \\ &\qquad\qquad \text{for every } \Big(E, \{\, \{F_{1,i}, F_{2,i}\} \mid i \in [k] \,\}\Big) \in \kappa. \end{aligned}$$

The added vertices $\{u, v\}$ are called **crossing**. The set of all crossing vertices of $\Omega(\mathfrak{A})$ is $\mathsf{V}_{\text{cross}}(\mathfrak{A})$. We call edges $(w, w') \in \mathsf{V}_{\text{cross}}(\mathfrak{A})^2$ **inter-crossing**. A relation or color is inter-crossing if it only contains inter-crossing edges. A vertex class or fiber is called crossing if it only contains crossing vertices.

Later, we are interested only in the substructure of $\Omega(\mathfrak{A})$ induced by the crossing vertices. But for HF-structures, this is actually ill-defined because the crossing vertices do not contain the atoms of $\mathfrak{A}$. So we turn to the non-HF-structure $\Omega(\mathfrak{A})^{\text{flat}}$ and define

$$\Omega^{\text{cross}}(\mathfrak{A}) := \Omega(\mathfrak{A})^{\text{flat}}[\mathsf{V}_{\text{cross}}(\Omega(\mathfrak{A})^{\text{flat}})].$$

Here, we refer with $\mathsf{V}_{\text{cross}}(\Omega(\mathfrak{A})^{\text{flat}})$ to the set of atoms in $\Omega(\mathfrak{A})^{\text{flat}}$ which are crossing vertices in $\Omega(\mathfrak{A})$. We now show that the fibers of crossing vertices, or the colors of inter-crossing edges, respectively, are already determined by plain fibers and plain relations.

Lemma 5.47. *For every normalized HF-structure $\mathfrak{A}$ and every building plan Ω for $\mathfrak{A}$, the structure $\Omega(\mathfrak{A})$ satisfies the following:*

(A1) Every vertex of $\Omega(\mathfrak{A})$ is either plain or crossing.

(A2) $\Omega(\mathfrak{A})[\mathsf{V}(\mathfrak{A})] = \mathfrak{A}$.

(A3) Every relation of $\Omega(\mathfrak{A})$ is either plain or inter-crossing or the special relation E_p.

(A4) For every crossing vertex $u \in \mathsf{V}_{cross}(\mathfrak{A})$ and every $i \in [2]$, there exists exactly one vertex $u^{(i)} \in \mathsf{V}_i(\mathfrak{A})$ such that $(u, u^{(i)}) \in E_p^{\mathfrak{A}}$. Moreover, $\{u^{(1)}, u^{(2)}\} \neq \{v^{(1)}, v^{(2)}\}$ for all distinct $u, v \in \mathsf{V}_{cross}(\mathfrak{A})$.

(A5) For every crossing vertex $u \in \mathsf{V}_{cross}(\mathfrak{A})$ its fiber U_u is uniquely determined by the set of fibers $\{U_{u^{(1)}}, U_{u^{(2)}}\}$ of the plain vertices $u^{(1)}$ and $u^{(2)}$, that is, whenever $\{U_{u^{(1)}}, U_{u^{(2)}}\} = \{U_{v^{(1)}}, U_{v^{(2)}}\}$, then $U_u = U_v$. Additionally, for every pair of vertices $v_1 \in U^{\mathfrak{A}}_{u^{(1)}} \cap \mathsf{V}_1(\mathfrak{A})$ and $v_2 \in U^{\mathfrak{A}}_{u^{(2)}} \cap \mathsf{V}_2(\mathfrak{A})$, there is a crossing vertex w in the fiber U_u such that $w^{(1)} = v_1$ and $w^{(2)} = v_2$.

(A6) In the same sense, the color $R_{(u,v)}$ of every inter-crossing edge (u, v) is uniquely determined by the set of plain colors $\{R_{(u^{(1)},v^{(1)})}, R_{(u^{(2)},v^{(2)})}\}$.

Proof. Let $\mathfrak{A} = \mathfrak{A}_1 \uplus \mathfrak{A}_2$ be a normalized HF-structure and $\Omega = (\omega, \kappa)$ be a building plan for $\mathfrak{A}$. Properties A1 to A4 immediately follow from the construction of $\Omega(\mathfrak{A})$. To show Property A5, let u be a crossing vertex in fiber U_u and let $U_{u^{(i)}}$ be the fiber of $u^{(i)}$ for every $i \in [2]$. First consider the claim that for all $v_1 \in U^{\mathfrak{A}}_{u^{(1)}} \cap \mathsf{V}_1(\mathfrak{A})$ and $v_2 \in U^{\mathfrak{A}}_{u^{(2)}} \cap \mathsf{V}_2(\mathfrak{A})$, there is a crossing vertex w such that $w^{(1)} = v_1$ and $w^{(2)} = v_2$. Because the crossing vertex u exists, there is a vertex plan $\{C, D\} \in \omega$ such that $\{u^{(1)}, u^{(2)}\} \in \tilde{E}^{\mathfrak{A}}_{\{C,D\}}$. Assume w.l.o.g. that $u^{(1)} \in C^{\mathfrak{A}}$ and that $u^{(2)} \in D^{\mathfrak{A}}$. Because $v_1 \in U^{\mathfrak{A}}_{u^{(1)}} \subseteq C^{\mathfrak{A}}$ and $v_2 \in U^{\mathfrak{A}}_{u^{(2)}} \subseteq D^{\mathfrak{A}}$, it follows that $\{v_1, v_2\} \in \tilde{E}^{\mathfrak{A}}_{\{C,D\}}$ and so by definition of $\Omega(\mathfrak{A})$ there is a crossing vertex w such that $w^{(1)} = v_1$ and $w^{(2)} = v_2$. To show that the fiber U_u is uniquely determined by $\{U_{u^{(1)}}, U_{u^{(2)}}\}$, it suffices to consider the special case of A6 when considering loop colors.

It remains to prove Property A6. Let (u, v) be an inter-crossing edge of color $R_{(u,v)}$. Let the color of $(u^{(i)}, v^{(i)})$ be $R_{(u^{(i)},v^{(i)})}$ for every $i \in [2]$. For the first direction, let (u', v') be a crossing edge such that $R_{(u',v')} = R_{(u,v)}$. Because every crossing vertex is adjacent via E_p to exactly one plain vertex in each component, every inter-crossing edge has exactly two (E_p, S_i, E_p^{-1})-colored paths, where S_i is a plain color, for every $i \in [2]$. For (u, v), we have that $\{S_1, S_2\} = \{R_{(u^{(1)},v^{(1)})}, R_{(u^{(2)},v^{(2)})}\}$. Because (u', v') has the same color as (u, v), the same holds for (u', v'), i.e.,

$$\left\{ R_{(u'^{(i)},v'^{(i)})} \;\middle|\; i \in [2] \right\} = \left\{ R_{(u^{(i)},v^{(i)})} \;\middle|\; i \in [2] \right\}.$$

For the remaining direction, we have to show that two inter-crossing edges (u_1, v_1) and (u_2, v_2), for which

$$\left\{ R_{u_1^{(i)},v_1^{(i)}} \;\middle|\; i \in [2] \right\} = \left\{ R_{u_2^{(i)},v_2^{(i)}} \;\middle|\; i \in [2] \right\},$$

have the same color $R_{(u_1,v_1)} = R_{(u_2,v_2)}$, where (u_1,v_1) and (u_2,v_2) are arbitrary. To do so, we construct a coherent configuration $\mathfrak{H}$, such that $\mathfrak{H}$ refines $C(\Omega(\mathfrak{A}))$, $\mathfrak{H}$ has the required property regarding colors of inter-crossing edges, and $\mathfrak{H}[\mathsf{V}_i(\mathfrak{A})] = C(\Omega(\mathfrak{A}))[\mathsf{V}_i(\mathfrak{A})]$ for every $i \in [2]$. Then every coarser coherent configuration has the same property for inter-crossing edges, too. We only sketch the construction, the idea is based on the proof of Lemma 10 in [59]. The main difference is that [59] gives the construction for the E_ω relation and not the $\tilde{E}_\omega$ relation (cf. the proof of Lemma 5.48). Essentially, we replace ordered pairs of two colors with a set of at most two colors. The coherent configuration is defined as follows: The plain edges are colored according to $C(\mathfrak{A})$. Every inter-crossing edge (u,v) is colored with the set of colors $\{R_{(u^{(1)},v^{(1)})}, R_{(u^{(2)},v^{(2)})}\}$ of its corresponding plain edges. Note that by Lemma 5.39, this also determines the color of the edges $(u'^{(i)}, v'^{(j)})$ whenever $u', v' \in \{u,v\}$ and $\{i,j\} = [2]$, i.e., $u'^{(i)}$ and $v'^{(j)}$ are in different components, because the color of an edge determines the fiber of its endpoints. Similarly, an edge (u,v) of a crossing vertex u and a plain vertex v is colored with the fiber of u (which is defined in the inter-crossing case for loops), the fiber of v, and whether (u,v) is contained in E_p. Clearly, the sketched coherent configuration has the required property by construction. □

We say that the plain vertices $u^{(1)}$ and $u^{(2)}$ are the vertices **corresponding to** the crossing vertex u. Likewise, the plain edges $(u^{(1)}, v^{(1)})$ and $(u^{(2)}, v^{(2)})$ correspond to the inter-crossing edge (u,v). The set of colors $\{E_{(u^{(1)},v^{(1)})}, E_{(v^{(2)},v^{(2)})}\}$ corresponds to the color $E_{(u,v)}$ (and similarly for crossing vertices). In some sense, Properties A5 and A6 mean that we actually do not need to construct the crossing vertices because all information is determined by the corresponding plain vertices and edges. We now show this formally.

Lemma 5.48. *There exists a normalized DeepWL-algorithm that for every normalized HF-structure $\mathfrak{A}$ and every building plan $\Omega = (\omega, \kappa)$ for $\mathfrak{A}$ computes $D(\Omega(\mathfrak{A}))$ in polynomial time.*

Proof. We show that there is a (non-normalized) DeepWL algorithm $\mathcal{M}$ which, on input $\mathfrak{A}$ in the cloud and $\Omega = (\omega, \kappa)$ on the work-tape, constructs the structure $\Omega(\mathfrak{A})$ in the cloud by first executing an `addPair`-operation on a crossing relation and then a `contract`-operation. This is done as follows. The crossing relations $E_{\{C,D\}}$ and $\tilde{E}_{\{C,D\}}$ are DeepWL-computable for every vertex plan $\{C,D\} \in \omega$. In particular, the crossing relation $\tilde{E}^{\mathfrak{A}}_\omega = \bigcup_{\{C,D\}\in\omega} \tilde{E}_{\{C,D\}}$ is DeepWL-computable. The algorithm $\mathcal{M}$ executes `addUPair`$(\tilde{E}_\omega)$ and obtains the relation E_p as membership relation. For every relation plan $(E, \{\{F_1^i, F_2^i\} \mid i \in [k]\}) \in \kappa$, the algorithm $\mathcal{M}$ defines $E^{\Omega(\mathfrak{A})}$ as follows. For every $i \in [k]$, the algorithm $\mathcal{M}$ defines the relation E_i to be the relation with an (E_p, F_1^i, E_p^{-1})-and an (E_p, F_2^i, E_p^{-1})-colored path. Then E is obtained as the union of all E_i. Similar to Lemma 5.29, the `addUPair`-operation can be simulated in the DeepWL-model of [59] by first an `addPair`-execution and then a `contract`-execution. Recall that `contract`(R) contracts every R-SCC to a singleton vertex. So we first execute `addPair`$(\tilde{E}_\omega)$ and then contract the edges between the two vertices $\langle u,v\rangle$ and $\langle v,u\rangle$ to obtain vertices $\{\langle u,v\rangle, \langle v,u\rangle\}$.

We now argue that there is a normalized DeepWL-algorithm $\hat{\mathcal{M}}$ computing the sketch of $\Omega(\mathfrak{A})$ without executing the operations (in fact, without modifying the cloud at all). Using Lemma 10 of [59], $\hat{\mathcal{M}}$ computes the sketch after the `addPair`-execution in polynomial time (which is possible because $\tilde{E}_\omega$ is crossing). Next, $\hat{\mathcal{M}}$ computes the the sketch after the `contract`-execution using Lemma 9 of [59]. Finally, the inter-crossing relations described by κ are unions of inter-crossing colors and the subset-relation can be computed in polynomial time. $\square$

Definition 5.49 (Efficient Building Plan). A building plan $\Omega = (\omega, \kappa)$ for a normalized τ-HF-structure $\mathfrak{A}$ is called **efficient** if, for every $u \in \mathsf{V}(\mathfrak{A})$, there is a vertex class $C \in \tau$ such that $u \in C^{\mathfrak{A}}$ and there is a vertex plan $\{C, D\} \in \omega$ for some $D \in \tau$.

Intuitively, an efficient building plan makes use of all plain vertices of $\mathfrak{A}$, which means that $\mathfrak{A}$ does not contain unnecessary vertices to construct $\Omega(\mathfrak{A})$.

Lemma 5.50. *If Ω is an efficient building plan for a normalized HF-structure $\mathfrak{A}$, then $|\mathfrak{A}| \leq 2|\mathsf{V}_{cross}(\Omega(\mathfrak{A}))|$.*

Proof. Every crossing vertex in $\mathsf{V}_{\text{cross}}(\Omega(\mathfrak{A}))$ is, by Property A4, incident to exactly two plain vertices. So there are at most twice as many plain as crossing vertices. $\square$

Lemma 5.51. *Let $\mathfrak{A}$ be a normalized HF-structure and Ω be a building plan for $\mathfrak{A}$. Then* $\mathsf{Aut}(\mathfrak{A}) = \mathsf{Aut}(\Omega(\mathfrak{A}))$ *(note that both structures are HF-structures with atom set A). For every crossing fiber U, it holds that if $U^{\Omega(\mathfrak{A})}$ is an $\Omega(\mathfrak{A})$-orbit, then for the corresponding plain fibers $\{U_1, U_2\}$ of U, the set $\tilde{E}^{\mathfrak{A}}_{\{U_1,U_2\}}$ is an $\mathfrak{A}$-orbit.*

Proof. We first show that $\mathsf{Aut}(\mathfrak{A}) \subseteq \mathsf{Aut}(\Omega(\mathfrak{A}))$. The structure $\Omega(\mathfrak{A})$ is defined in an isomorphism-invariant manner. Whenever a vertex or relation is added, it is done for all vertices/edges of a given vertex class/relation (cf. the proof of Lemma 5.48 that shows that $\Omega(\mathfrak{A})$ can be obtained from $\mathfrak{A}$ by a DeepWL-algorithm). So every automorphism of $\mathfrak{A}$ extends to an automorphism of $\Omega(\mathfrak{A})$. To show $\mathsf{Aut}(\Omega(\mathfrak{A})) \subseteq \mathsf{Aut}(\mathfrak{A})$, note that $\mathfrak{A}$ is contained in $\Omega(\mathfrak{A})$. Also, note that all relations added in $\Omega(\mathfrak{A})$ are new ones and no relation of $\mathfrak{A}$ is changed. It is never possible that an automorphism maps a crossing vertex to a plain vertex because the E_p relation is directed from crossing to plain vertices. So every automorphism of $\Omega(\mathfrak{A})$ is an automorphism of $\mathfrak{A}$.

For the second part, let U be a crossing fiber such that $U^{\Omega(\mathfrak{A})}$ is an $\Omega(\mathfrak{A})$-orbit. By construction of the E_p relation, an automorphism φ satisfies $\varphi(u) = v$ if and only if $\varphi(\{u^{(1)}, u^{(2)}\}) = \{v^{(1)}, v^{(2)}\}$ for all $u, v \in U^{\Omega(\mathfrak{A})}$ (cf. Property A4). Because $\tilde{E}^{\mathfrak{A}}_{\{U_1,U_2\}}$ contains exactly these pairs $\{u^{(1)}, u^{(2)}\}$, it is an $\Omega(\mathfrak{A})$-orbit, too, and thus also an $\mathfrak{A}$-orbit since $\mathsf{Aut}(\mathfrak{A}) = \mathsf{Aut}(\Omega(\mathfrak{A}))$. $\square$

Comparison to [59]. To show that every DeepWL-algorithm can be simulated by an equivalent normalized DeepWL-algorithm, almost normalized structures are used as intermediate step in [59]. The structures obtained from building plans differ at some points from the almost normalized structures:

- Our notion of crossing vertices is defined only for building plans and not for general DeepWL-algorithms (and is for *undirected* crossing edges).
- Our relation E_p does not distinguish between $u^{(1)}$ and $u^{(2)}$. This is needed so that a `choice`-operation on crossing vertices does not necessarily distinguish the two components.

- The relation E_p assigns to every crossing vertex exactly one plain vertex in each component. This ensures that choice-sets can still be witnessed.
- Properties similar to A5 and A6 are always satisfied for almost normalized structures in [59]. In combination with `refine`- and `choice`-operations, this would not be the case anymore.

5.3.4.5 Building Plans and scc-Operations

Before we can start to design normalized DeepWL+WSC-algorithms, we have to investigate `scc`-operations. Assume that for an inter-crossing relation E the `scc`(E)-operation is to be executed and we want to find a building plan simulating this. The challenge is to construct new plain vertices so that we find plain vertex classes which correspond to the vertex class obtained by `scc`(E). The first step is to analyze the SCCs of the corresponding plain colors of E in the components. In a second step, we show how we can define a building plan to simulate the `scc`(E)-operation. We start with a lemma regarding SCCs in coherent configurations.

Lemma 5.52. *Let $\mathfrak{H}$ be a coherent configuration with signature σ and $R, S \in \sigma$ be colors connecting vertices of the same fiber, i.e., $R^{\mathfrak{H}} \subseteq (U^{\mathfrak{H}})^2$ and $S^{\mathfrak{H}} \subseteq (U^{\mathfrak{H}})^2$ for some fiber $U \in \sigma$. Then*

1. *every R-connected component is strongly R-connected and*
2. *every $\{R, S\}$-connected component is strongly $\{R, S\}$-connected.*

Proof. We start with Claim 1: If R is itself a fiber, all connected components are trivial and the claim follows. Otherwise, let c be an R-connected component. Then we can assume that $\mathfrak{H}$ is a primitive coherent configuration, otherwise we can restrict $\mathfrak{H}$ to c (all edges leaving c have different colors than edges contained in c). Finally, it follows from Theorem 3.1.5. in [23] that c is strongly R-connected because R is not a fiber.

To show Claim 2, let c be an $\{R, S\}$-connected component. To prove that c is strongly $\{R, S\}$-connected, let $(u_1, \dots, u_k)$ be a $\{R, S\}$-path in c. We show that there is an $\{R, S\}$-path $(v_1, \dots, v_m)$ such that $v_1 = u_k$ and $v_m = u_1$. To do so, it suffices to show that, for every $i \in [k-1]$, there is a $\{R, S\}$-path from u_{i+1} to u_i. So let $i \in [k-1]$ and w.l.o.g. assume that $(u_i, u_{i+1}) \in R^{\mathfrak{H}}$. Then by Claim 1, the vertices u_i and u_{i+1} are in the same R-SCC. In particular, there is an R-path and so also an $\{R, S\}$-path from u_{i+1} to u_i. □

Lemma 5.53. *Let $\mathfrak{A} = \mathfrak{A}_1 \uplus \mathfrak{A}_2$ be a normalized HF-structure, Ω be a building plan for $\mathfrak{A}$, R be an inter-crossing color of $\mathfrak{B} = \Omega(\mathfrak{A})$ such that R-edges connect vertices of the same fiber U_R, i.e., $R^{\mathfrak{B}} \subseteq (U_R^{\mathfrak{B}})^2$, and $S, T \in \sigma$ be the corresponding plain colors of R. If c is an R-SCC, then $\{u^{(i)} \mid u \in c\}$ is an $\{S, T\}$-SCC for every $i \in [2]$.*

Proof. Define
$$K_i := \left\{ (u, v) \in R^{\mathfrak{B}} \;\middle|\; (u^{(i)}, v^{(i)}) \in S^{\mathfrak{A}} \right\}.$$
That is, if $S \neq T$, we partition $R^{\mathfrak{B}}$ into K_1 and K_2 depending on whether the corresponding S-edge is in $\mathfrak{A}_1$ or in $\mathfrak{A}_2$. If $S = T$, we just have $K_1 = K_2 = R^{\mathfrak{B}}$. For a set c of crossing vertices we define $c^{(i)} := \{u^{(i)} \mid u \in c\}$. We start to analyze the SCCs formed by the K_i-edges.

Claim 1. *Let $i \in [2]$ and c be a K_i-SCC. Then the set $c^{(i)}$ is an S-SCC. For $j \in [2]$ such that $\{i, j\} = [2]$, the set $c^{(j)}$ is a T-SCC.*

Proof. We consider the part of the claim regarding S-SCCs. The part regarding T-SCCs is symmetric. Clearly, if $(u_1, \ldots, u_k)$ is a K_i-path, then $(u_1^{(i)}, \ldots, u_k^{(i)})$ is an S-path. So $c^{(i)}$ is contained in an S-SCC. If S-edges connect vertices in different fibers, then all S-SCCs are singletons. That is $c^{(i)}$ cannot be strictly contained in an S-SCC.

So it remains to consider the case when S connects vertices in the same fiber. Then also T has to connect vertices in the same fiber (otherwise R would not be a color). Let $\{U, V\}$ be the fibers corresponding to U_R. If U has an incident S-edge, then V has an incident T-edge (because otherwise R would be empty, which is not allowed for colors). We show that $c^{(i)}$ is an S-connected component. This implies by Lemma 5.52 that $c^{(j)}$ is an S-SCC. Let $(u_1, u_2) \in S^{\mathfrak{A}}$ be such that $u_1 \in c^{(i)}$. We show that there is an R-edge (v_1, v_2) such that $(v_1^{(i)}, v_2^{(i)}) = (u_1, u_2)$ and $v_1, v_2 \in c$, which implies that $u_2 \in c^{(i)}$ and by induction that $c^{(i)}$ is an S-connected component. Because $u_1 \in c^{(i)}$, there is a vertex $v_1 \in c$ such that $v_1^{(i)} = u_1$. Assume w.l.o.g. that $u_1 \in U^{\mathfrak{A}}$. Then $v_1^{(j)} \in V^{\mathfrak{A}}$, where $\{i, j\} = [2]$. Because u_1 has an incident S-edge (namely (u_1, u_2)), $v_1^{(j)}$ has an incident T-edge $(v_1^{(j)}, w)$. Because S and T connect vertices of the same fiber, $u_2 \in U^{\mathfrak{A}}$ and $w \in V^{\mathfrak{A}}$. Then by Property A5 there is a vertex v_2 such that $v_2^{(i)} = u_2$ and $v_2^{(j)} = w$. Now, we have that $(v_1, v_2) \in R^{\mathfrak{B}}$ by Property A6, that $(v_1^{(i)}, v_2^{(i)}) = (u_1, u_2) \in S^{\mathfrak{A}}$, and that $(v_1^{(j)}, v_2^{(j)}) = (v_1^{(j)}, w) \in T^{\mathfrak{A}}$. That is, v_1 and v_2 are in the same R-connected component. This, by Lemma 5.52, implies that v_1 and v_2 are in the same R-SCC and thus $v_2 \in c$. $\dashv$

Claim 2. *Let S connect vertices of different fibers U and V and let T connect vertices of the same different fibers but in the other direction, i.e., $S^{\mathfrak{A}} \subseteq U^{\mathfrak{A}} \times V^{\mathfrak{A}}$ and $T^{\mathfrak{A}} \subseteq V^{\mathfrak{A}} \times U^{\mathfrak{A}}$ for $U \neq V$. Let $i \in [2]$ and c be an R-SCC. Then the set $c^{(i)}$ is an $\{S, T\}$-SCC.*

Proof. If $(u_1, \ldots, u_k)$ is an R-path, then $(u_1^{(i)}, \ldots, u_k^{(i)})$ is an $\{S, T\}$-path. So $c^{(i)}$ is contained in an $\{S, T\}$-SCC d. For the sake of contradiction, suppose that $c^{(i)}$ is strictly contained in d. Then there is an $\{S, T\}$-path $(v_1, \ldots, v_\ell)$ contained in d with $\ell \geq 3$, $v_1 \in c^{(i)}$, $v_\ell \in c^{(i)}$, and $v_k \notin c^{(i)}$ for every $1 < k < \ell$. We observe the following:

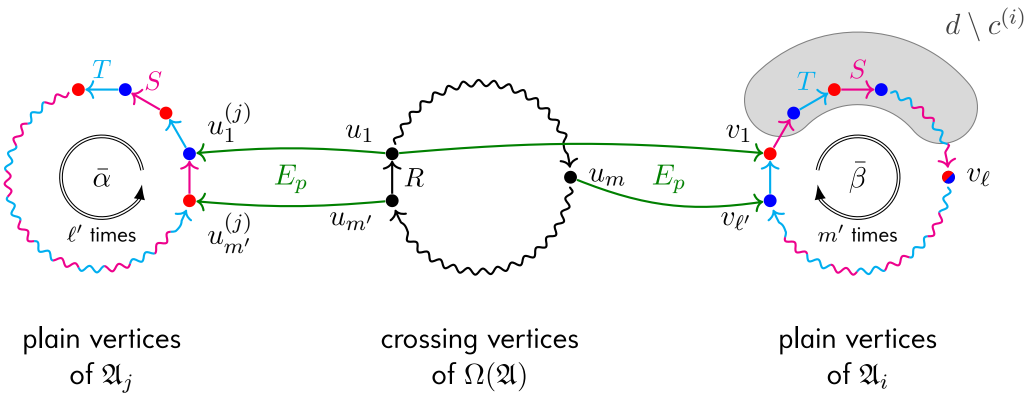
(a) There is a (possibly empty) $\{S, T\}$-path $(v_\ell, \ldots, v_{\ell'+1})$ contained in d such that $v_{\ell'+1} = v_1$ because d is an $\{S, T\}$-SCC.

(b) In every $\{S, T\}$-path the edge colors S and T alternate. Likewise, the fibers of the vertices alternate. Thus, every cycle consists of an even number of vertices.

Let $(u_1, \ldots, u_{m'}, u_1)$ be a nonempty R-cycle (which possibly uses vertices multiple times) such that $u_1^{(i)} = v_1$, $1 < m \leq m'$, and $u_m^{(i)} = v_{\ell'}$. Such a cycle exists because c is an R-SCC and $v_\ell \in c^{(i)}$. Consider the two following sequences $\bar{\beta} := (v_1, \ldots, v_{\ell'})^{m'}$ and $\bar{\alpha} := (u_1^{(j)}, \ldots, u_{m'}^{(j)})^{\ell'}$ of plain vertices of length $\ell' \cdot m'$, where j is chosen such that $\{i, j\} = [2]$, cf. Figure 5.5.

For a vertex $w \in \mathsf{V}(\mathfrak{A})$, denote by U_w the fiber containing w. For every pair $(w, w') \in \mathsf{V}(\mathfrak{A})^2$, denote by $R_{(w,w')}$ the color containing (w, w'). We show that

$$\{U_{\beta_k}, U_{\alpha_k}\} = \{U, V\}$$



5.5 The situation in Claim 2 in the proof of Lemma 5.53. All vertices in U are red, the ones in V are blue. Whether the vertex v_ℓ is in U or in V depends on whether ℓ is even or odd. The sequence $\bar{\alpha}$ iterates ℓ' times the cycle $(u_1^{(j)}, \dots, u_{m'}^{(j)}, u_1^{(j)})$ and the sequence $\bar{\beta}$ iterates m' times the cycle $(v_1, \dots, v_{\ell'}, v_1)$. This figure assumes that $v_1 \in U^{\mathfrak{A}}$. In the case that $v_1 \in V^{\mathfrak{A}}$, the colors S and T and the fibers U and V need to be swapped.

for every $k \in [\ell' \cdot m']$: First, consider the case $k = 1$. By construction,

$$(\beta_1, \alpha_1) = (v_1, u_1^{(j)}) = (u_1^{(i)}, u_1^{(j)}).$$

Because R connects vertices in the same fiber, the corresponding fibers of U_R must be $\{U, V\}$ and because $u_1 \in U_R^{\mathfrak{B}}$, it follows that $\{U_{\beta_1}, U_{\alpha_1}\} = \{U, V\}$. Second, assume that $\{U_{\beta_k}, U_{\alpha_k}\} = \{U, V\}$ for some $k < \ell' \cdot m'$. We already have seen that the fibers U and V alternate on $\{S, T\}$-paths, so in particular on the cycles $(v_1, \dots, v_{\ell'}, v_1)$ and $(u_1^{(j)}, \dots, u_{m'}^{(j)}, u_1^{(j)})$. So if $U_{\beta_k} = U$, then $U_{\beta_{k+1}} = V$ and vice versa and similar for α_k and α_{k+1}. Hence, $\{U_{\beta_{k+1}}, U_{\alpha_{k+1}}\} = \{U, V\}$.

By Property A5, there is a sequence of crossing vertices $(w_1, \dots, w_{\ell' \cdot m'})$ such that $w_k^{(i)} = \beta_k$ and $w_k^{(j)} = \alpha_k$ for all $k \in [\ell' \cdot m']$. We prove that $(w_1, \dots, w_{\ell' \cdot m'}, w_1)$ is an R-cycle. We consider the sequences of colors

$$R_{(w_1^{(i)}, w_2^{(i)})}, \dots, R_{(w_{\ell' \cdot m'-1}^{(i)}, w_{\ell' \cdot m'}^{(i)})}, R_{(w_{\ell' \cdot m'}^{(i)}, w_1^{(i)})} \text{ and}$$
$$R_{(w_1^{(j)}, w_2^{(j)})}, \dots, R_{(w_{\ell' \cdot m'-1}^{(j)}, w_{\ell' \cdot m'}^{(j)})}, R_{(w_{\ell' \cdot m'}^{(j)}, w_1^{(j)})}.$$

Similar to the case of the fibers, the colors S and T alternate in both sequences and $\{R_{(w_k^{(i)}, w_{k+1}^{(i)})}, R_{(w_k^{(j)}, w_2^{(k+1)})}\} = \{S, T\}$ for every $k < \ell' \cdot m'$ and

$$\left\{R_{(w_{\ell' \cdot m'}^{(i)}, w_1^{(i)})}, R_{(w_{\ell' \cdot m'}^{(j)}, w_1^{(j)})}\right\} = \{S, T\}.$$

Because S and T are the corresponding colors of R, $(w_k, w_{k+1}) \in R^{\mathfrak{B}}$ by Property A6 for every $k < \ell' \cdot m'$ and $(w_{\ell' \cdot m'}, w_1) \in R^{\mathfrak{B}}$. Thus, $(w_1, \dots, w_{\ell' \cdot m'}, w_1)$ is an R-cycle.

Because $\beta_1 = v_1 = u_1^{(i)}$ and $\alpha_1 = u_1^{(j)}$, we have that $w_1 = u_1 \in c$. In particular, the cycle $(w_1, \dots, w_{\ell' \cdot m'}, w_1)$ contains a vertex in c. By construction, $w_k^{(i)} = v_k$ for every $1 < k \leq \ell'$. And by assumption on the path $(v_1, \dots, v_\ell)$, $v_k \notin c^{(i)}$ for every $1 < k < \ell$.

Because $\ell \geq 3$, there is a $1 < k < \ell$ such that $v_k \notin c^{(i)}$ and thus $w_k \notin c$. But this means that there is an R-cycle containing a vertex in the R-SCC c and a vertex not in c, which is a contradiction. $\dashv$

First consider the case that $S = T$. Then $K_1 = K_2 = R^{\mathfrak{B}}$ and the claim of the lemma follows immediately from Claim 1. So consider the case that $S \neq T$. Let $S^{\mathfrak{A}} \subseteq U_S^{\mathfrak{A}} \times V_S^{\mathfrak{A}}$, $T^{\mathfrak{A}} \subseteq U_T^{\mathfrak{A}} \times V_T^{\mathfrak{A}}$, and recall that $R^{\mathfrak{B}} \subseteq (U_R^{\mathfrak{B}})^2$. Then, by Property A5, it follows that the corresponding fibers for U_R are $\{U_S, U_T\} = \{V_S, V_T\}$. We make the following case distinction:

- $U_S \neq V_S$ or $U_T \neq V_T$: We have $U_S = V_T \neq V_S = U_T$ because $\{U_S, U_T\} = \{V_S, V_T\}$. The assertion of the lemma follows immediately from Claim 2.

- $U_S = V_S = U_T = V_T$: Let c be an R-SCC. By Lemma 5.52, it suffices to show that $c^{(i)}$ is an $\{S,T\}$-connected component. So let (u,v) be an $\{S,T\}$-edge and let $u \in c^{(i)}$. We show that then also $v \in c^{(i)}$ which by induction shows that $c^{(i)}$ is an $\{S,T\}$-connected component. Assume w.l.o.g. that $(u,v) \in S^{\mathfrak{A}}$. Now for some K_i-SCC c' we have that $u \in c'^{(i)}$ and by Claim 1 we also have that $v \in c'^{(i)}$ because by Lemma 5.52 the vertices u and v are in the same S-SCC. Because $c' \subseteq c$, we have that $c'^{(i)} \subseteq c^{(i)}$ and thus that $v \in c^{(i)}$. $\square$

Now that we know that the SCCs of inter-crossing relations correspond to SCCs of the corresponding plain relations, we show that `scc`-operations on inter-crossing relations can be simulated using building plans.

Lemma 5.54. *There is a normalized polynomial-time DeepWL-algorithm that, for every normalized HF-structure $\mathfrak{A} = \mathfrak{A}_1 \uplus \mathfrak{A}_2$, every building plan Ω for $\mathfrak{A}$, and every inter-crossing color R of $\Omega(\mathfrak{A})$, halts with a normalized HF-structure $\mathfrak{B}$ in the cloud and writes a building plan Ω_R for $\mathfrak{B}$ onto the tape which satisfies $\Omega_R^{cross}(\mathfrak{B}) \cong \Omega^{cross}(\mathfrak{A})_R$, where $\Omega^{cross}(\mathfrak{A})_R$ denotes the structure obtained by executing `scc`(R) on $\Omega^{cross}(\mathfrak{A})$. If Ω is efficient, then Ω_R is efficient, too.*

Proof. Let $\mathfrak{A} = \mathfrak{A}_1 \uplus \mathfrak{A}_2$ be a normalized HF-structure, $\Omega = (\omega, \kappa)$ be a building plan for $\mathfrak{A}$, and R be an inter-crossing color of $\Omega(\mathfrak{A})$. We distinguish the following two cases: Assume first that the color R connects vertices in different fibers U and V, that is, $R^{\Omega(\mathfrak{A})} \subseteq U^{\Omega(\mathfrak{A})} \times V^{\Omega(\mathfrak{A})}$. Clearly, all R-SCCs are trivial and the `scc`-operation would create a new vertex for every U- and every V-vertex. To create these vertices with the building plan, let the corresponding fibers of U and V be U_1, U_2, V_1, and V_2. We create copies of these fibers by executing `addPair` for the loops and obtain component relations E_{left} and E_{right}, which coincide. The new vertices end up in new fibers U_1', U_2', V_1', and V_2'. The algorithm updates

$$\omega \leftarrow \omega \cup \Big\{\{U_1', U_2'\}, \{V_1', V_2'\}\Big\} \text{ and}$$
$$\kappa \leftarrow \kappa \cup \Big\{(F, \{E_{\text{left}}, E_{\text{right}}\})\Big\},$$

where F serves as new membership relation (and just relates a U-vertex to its copy).

Otherwise, R connects vertices in the same fiber. Let this fiber be U and let U_1 and U_2 be the fibers corresponding to U and R_1 and R_2 be the colors corresponding to R.

Note that in this case every $\{R_1, R_2\}$-SCCs is nontrivial: W.l.o.g. R_1 has to connect U_1 to U_2 and R_2 has to connect U_2 to U_1. Because R_1 and R_2 are fibers, every U_1-vertex and every U_2-vertex has one outgoing and one incoming $\{R_1, R_2\}$-edge, so there must be a cycle and, in particular, one nontrivial $\{R_1, R_2\}$-SCCs. But by the properties of a coherent-configuration, every $\{R_1, R_2\}$-SCCs has the same size and is thus nontrivial. By Lemma 5.53, we can construct vertices for SCCs of the corresponding plain vertices such that, for every R-SCC, there is a corresponding pair of $\{R_1, R_2\}$-SCCs. So we start with executing $\texttt{scc}(\{R_1, R_2\})$ (formally we have to create a relation as the union of R_1 and R_2) and obtain a new plain vertex class C containing the new SCC-vertices and a plain membership relation E. We then create, for each pair of C-vertices in different components, that is, for such a pair of $\{R_1, R_2\}$-SCCs, new crossing vertices and the inter-crossing membership relation F by updating

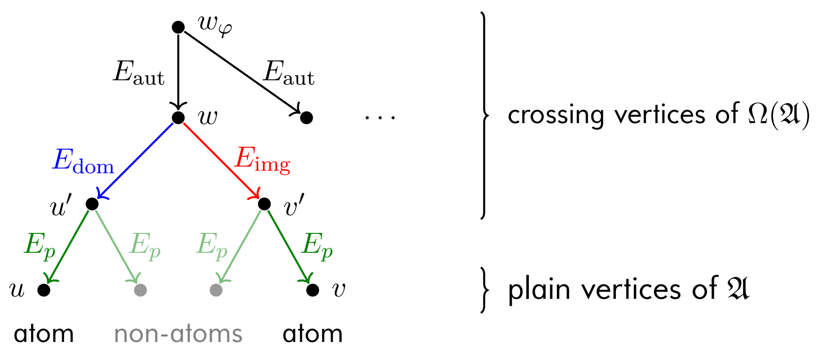
$$\omega \leftarrow \omega \cup \Big\{\{C, C\}\Big\} \text{ and}$$
$$\kappa \leftarrow \kappa \cup \Big\{(F, \{E, E\})\Big\}.$$

Indeed, every newly created vertex by ω corresponds to an R-SCC: Because the R-SCCs are nontrivial, every C-vertex was obtained from an $\{U_1, U_2\}$-SCC containing at least one U_1-vertex and at least one U_2-vertex (if $U_1 = U_2$ this is trivial and if $U_1 \neq U_2$ every nontrivial $\{R_1, R_2\}$-SCC has to contain one U_1- and one U_2-vertex because R_1 has to connect U_1 to U_2 and R_2 the other way around). Thus, for every pair of C-vertices in different components, we can find a U_1-vertex $u \in \mathsf{V}(\mathfrak{A}_1)$ in one component and a U_2-vertex $v \in \mathsf{V}(\mathfrak{A}_2)$ in the other component. By Property A5, there is a U-vertex w such that $w^{(1)} = u$ and $w^{(1)} = v$. But this means there is an R-SCC, which corresponds to the two C-vertices, namely the one containing w. In the same manner, F correctly defines the inter-crossing membership relation for the new R-SCC vertices.

Finally, to see that $\mathfrak{B}$ is normalized, note that we only executed a single $\texttt{addPair}$-operation or a single $\texttt{scc}$-operation for a plain relation. It is also clear that the property of being efficient is preserved because every newly created vertex is in a fiber U_1', U_2', V_1', and V_2' or in the vertex class C and all of them are used in the building plan. Obviously, the algorithm runs in polynomial time. $\square$

5.3.4.6 Simulation

Finally, we want to simulate arbitrary DeepWL+WSC-algorithms with normalized ones. Recall that we still have to define how normalized DeepWL+WSC-algorithms encode sets of witnessing automorphisms. We do this now. Let $\mathfrak{A}$ be a normalized HF-structure and Ω be a building plan for $\mathfrak{A}$. A tuple of crossing relations $(E_{\text{aut}}, E_{\text{dom}}, E_{\text{img}})$ and a crossing vertex w_φ **encode the partial map** $\varphi\colon A \to A$ as follows (cf. Figure 5.6): We have $\varphi(u) = v$ in case that there exists exactly one crossing vertex w such that $(w_\varphi, w) \in E_{\text{aut}}^{\Omega(\mathfrak{A})}$, $(w, u') \in E_{\text{dom}}^{\Omega(\mathfrak{A})}$, and $(w, v') \in E_{\text{img}}^{\Omega(\mathfrak{A})}$, where u' and v' are crossing vertices and u and v are the only atoms for which $(u', u) \in E_p^{\Omega(\mathfrak{A})}$ and $(v', v) \in E_p^{\Omega(\mathfrak{A})}$. Recall here that every crossing vertex has exactly two E_p-neighbors. So, the other E_p-neighbor of u' and v' must not be an atom. While introducing the vertices u' and v' into the definition seems odd at first, this will simplify technical aspects in the following. A tuple of relations $(E_{\text{aut}}, E_{\text{dom}}, E_{\text{img}})$ **encodes the set of partial maps** $\{\varphi \mid w_\varphi \text{ encodes } \varphi \text{ for some } (w_\varphi, w) \in E_{\text{aut}}^{\mathfrak{A}}\}$.



5.6 Encoding of automorphisms for normalized DeepWL+WSC algorithms. An automorphism φ of a normalized HF-structure $\mathfrak{A}$ is encoded by a building plan Ω, a tuple of crossing relations $(E_{\text{aut}}, E_{\text{dom}}, E_{\text{img}})$, and a vertex w_φ: The figure shows the encoding of $\varphi(u) = v$. The vertices w_φ, w, u', and v' are crossing vertices of $\Omega(\mathfrak{A})$. The vertices u and v are atoms and the other E_p-neighbor of u' and v', respectively, is not an atom.

The witnessing machine M^{wit} of a normalized DeepWL+WSC-algorithm on input $\mathfrak{A}$ outputs a set of witnessing automorphisms by writing a tuple $(\Omega, E_{\text{aut}}, E_{\text{dom}}, E_{\text{img}})$ of a building plan Ω for the final content of the cloud of M^{wit} and three relations for which Ω contains a relation plan on the interaction-tape. Note that, with this definition, a normalized DeepWL+WSC-algorithm is formally not a (non-normalized) DeepWL+WSC-algorithm anymore, because it encodes sets of witnessing automorphisms differently. Now, we are ready to simulate an arbitrary DeepWL+WSC-algorithm with a normalized one.

Definition 5.55 (Simulating a Structure). A pair $(\mathfrak{A}, \Omega)$ of a normalized HF-structure $\mathfrak{A}$ and a building plan $\Omega = (\omega, \kappa)$ for $\mathfrak{A}$ **simulates** an HF-structure $\hat{\mathfrak{A}}$ with $\hat{A} = A$ if

(S1) $\Omega^{\text{cross}}(\mathfrak{A}) \cong \hat{\mathfrak{A}}^{\text{flat}}$,

(S2) Ω is efficient,

(S3) there is a relation plan $(C, \{D_1, D_2\}) \in \kappa$ defining a crossing vertex class C, such that every isomorphism $\varphi\colon \hat{\mathfrak{A}}^{\text{flat}} \to \Omega^{\text{cross}}(\mathfrak{A})$ satisfies $\varphi(\hat{A}) = C^{\Omega(\mathfrak{A})}$, i.e., the atoms of $\hat{\mathfrak{A}}$ are mapped precisely onto the C-vertices,

(S4) the E_p-relation is a perfect matching between the atoms A of $\mathfrak{A}$ and $C^{\Omega(\mathfrak{A})}$, and

(S5) via this bijection between A and $C^{\Omega(\mathfrak{A})}$ we have that $\mathsf{Aut}(\mathfrak{A}^{\text{flat}}) = \mathsf{Aut}(\Omega^{\text{cross}}(\mathfrak{A}))$.

Note that the definition above only relates $\Omega^{\text{cross}}(\mathfrak{A})$ (and not $\Omega(\mathfrak{A})$) to $\hat{\mathfrak{A}}$. This definition reduces the need for case distinctions in the simulation: Crossing vertices of $\Omega(\mathfrak{A})$ are always used to simulate $\hat{\mathfrak{A}}$ and plain vertices are always used to create crossing vertices. Now that we have a notion of simulating a structure, we can also simulate DeepWL+WSC-algorithms:

Definition 5.56 (Simulating an Algorithm). Assume that $\mathcal{M} = (M^{\text{out}}, M^{\text{wit}}, \mathcal{M}_1, \dots, \mathcal{M}_\ell)$ and $\hat{\mathcal{M}} = (\hat{M}^{\text{out}}, \hat{M}^{\text{wit}}, \hat{\mathcal{M}}_1, \dots, \hat{\mathcal{M}}_\ell)$ are DeepWL+WSC-algorithms. The algorithm $\mathcal{M}$ **simulates** $\hat{\mathcal{M}}$ if $\mathcal{M}_i$ simulates $\hat{\mathcal{M}}_i$ for all $i \in [\ell]$ and, for every structure $\hat{\mathfrak{A}}$ and every pair $(\mathfrak{A}, \Omega)$ simulating $\hat{\mathfrak{A}}$, the algorithm $\mathcal{M}$ on input $(\mathfrak{A}, \Omega)$ accepts (respectively rejects) whenever $\hat{\mathcal{M}}$ on input $\hat{\mathfrak{A}}$ accepts (respectively rejects).

We do not care about $\hat{\mathcal{M}}$ failing because in the following we will always assume that this is not the case.

Lemma 5.57. *For every polynomial-time DeepWL+WSC-algorithm $\hat{\mathcal{M}}$, there is a normalized polynomial-time DeepWL+WSC-algorithm $\mathcal{M}$ simulating $\hat{\mathcal{M}}$.*

Proof. We will construct normalized DeepWL+WSC-algorithms, which uses the additional operations of Lemma 5.29. One easily sees that the reductions in the proof of Lemma 5.29 preserve being normalized. The proof of this lemma is by induction on nesting DeepWL+WSC-algorithms. For this, assume $\hat{\mathcal{M}} = (\hat{M}^{\text{out}}, \hat{M}^{\text{wit}}, \hat{\mathcal{M}}_1, \dots, \hat{\mathcal{M}}_\ell)$ is a polynomial-time DeepWL+WSC-algorithm and assume, by the induction hypothesis, that there are normalized and polynomial-time DeepWL+WSC-algorithms $\mathcal{M}_i$ simulating $\hat{\mathcal{M}}_i$ for every $i \in [\ell]$. By Lemma 5.35, we can assume that $\hat{\mathcal{M}}$ is pure.

Let $\hat{M} \in \{\hat{M}^{\text{out}}, \hat{M}^{\text{wit}}\}$. We will now construct a DeepWL+WSC-machine M simulating $\hat{M}$. Let $\hat{\mathfrak{A}}_0$ be the input HF-structure of $\hat{M}$. Let $\hat{\mathfrak{A}}_0, \dots, \hat{\mathfrak{A}}_k$ be the sequence of HF-structures in the cloud of $\hat{M}$ and let $(\mathfrak{A}_0, \Omega_0)$ simulate $\hat{\mathfrak{A}}_0$. We construct the machine M inductively (independently of the specific input $\mathfrak{A}_0$). We say that the machine M on input $(\mathfrak{A}_0, \Omega_0)$ **simulates the t-th step** for $t \in [k]$ if the content of the cloud of M is $\mathfrak{A}_t$ and there is a building plan $\Omega_t = (\omega_t, \kappa_t)$ for $\mathfrak{A}_t$ written onto the working tape such that $(\mathfrak{A}_t, \Omega_t)$ simulates $\hat{\mathfrak{A}}_t$. Assume that we constructed a machine M simulating the k-th step, then the content of the cloud does not change anymore and we can just track the run of the Turing machine of $\hat{M}$ until it halts.

We construct by induction on t a machine M simulating the t-th step for every $t \leq k$. For $t = 0$, the claim holds by assumption that $(\mathfrak{A}_0, \Omega_0)$ simulates $\hat{\mathfrak{A}}_0$. Now assume that M simulates the t-th step and that $(\mathfrak{A}_t, \Omega_t)$ simulates $\hat{\mathfrak{A}}_t$. By Lemma 5.48, the machine M computes $D(\Omega_t(\mathfrak{A}_t))$ in polynomial time. From this sketch, it computes $D(\Omega_t^{\text{cross}}(\mathfrak{A}_t))$ using Lemma 5.24. Because $(\mathfrak{A}_t, \Omega_t)$ simulates $\hat{\mathfrak{A}}_t$, by Property S1, it holds that $\Omega_t^{\text{cross}}(\mathfrak{A}_t) \cong \hat{\mathfrak{A}}_t^{\text{flat}}$ and thus $D(\Omega_t^{\text{cross}}(\mathfrak{A}_t)) = D(\hat{\mathfrak{A}}_t)$. So M can track the run of $\hat{M}$ until $\hat{M}$ executes an operation modifying the cloud. We make a case distinction on this operation:

`addPair`(R)**:** The color R is an inter-crossing color in $\Omega_t(\mathfrak{A}_t)$ because $\Omega_t^{\text{cross}}(\mathfrak{A}_t) \cong \hat{\mathfrak{A}}_t^{\text{flat}}$. Let R_1 and R_2 be the two plain colors corresponding to R. The machine M executes `addPair`(R_1) and `addPair`(R_2) (if $R_1 = R_2$, only one operation is executed) and obtains vertex classes C_1 and C_2 and component relations $E_{i,d}$ for all $i \in [2]$ and $d \in \{\text{left}, \text{right}\}$ (here if $R_1 = R_2$, we have $C_1 = C_2$ and $E_{1,d} = E_{2,d}$). We set

$$\omega_{t+1} := \omega_t \cup \Big\{\{C_1, C_2\}\Big\} \text{ and}$$
$$\kappa_{t+1} := \kappa_t \cup \Big\{ (R_d, \{\{E_{1,d}, E_{2,d}\}\}) \;\Big|\; d \in \{\text{left}, \text{right}\} \Big\}.$$

That is, precisely for every set $\{u, v\}$ of vertices $u \in C_1^{\mathfrak{A}_{t+1}}$ and $v \in C_2^{\mathfrak{A}_{t+1}}$, which means by construction for every set $\{e_1, e_2\}$ of edges $e_1 \in R_1^{\mathfrak{A}_t}$ and $e_2 \in R_2^{\mathfrak{A}_t}$, which again means by Property A6 for every edge $e \in R^{\Omega_t(\mathfrak{A}_t)}$, a vertex is added in $\Omega_{t+1}(\mathfrak{A}_{t+1})$. We create the component relations from the $E_{i,d}$. That is, $\Omega_{t+1}^{\text{cross}}(\mathfrak{A}_{t+1}) \cong \hat{\mathfrak{A}}_{t+1}^{\text{flat}}$ and so we maintained Property S1. Property S2 is satisfied because all newly created plain vertices are either in C_1 or C_2. Properties S3 and S4 will always be maintained if they initially hold (and we do not remove entries from the building plan). Finally, Property S5 is maintained because we did not make any choices and thus the automorphism of both structures stay exactly the same by Corollary 5.28 (note that by Lemma 5.25 both $\mathfrak{A}_{t+1}$ and $\mathfrak{A}_{t+1}^{\text{flat}}$ have the same automorphisms). So we simulated the $(t+1)$-th step.

`scc`(R): Again, R is an inter-crossing color in $\Omega_t(\mathfrak{A}_t)$. The machine M simulates the `scc`-operation using Lemma 5.54: Properties S1 and S2 are ensured by the lemma, all other properties hold by the same reasons as for the `addPair`-operation, and so the $(t+1)$-th step is simulated.

`create`(π): Let $\pi = \{R_1, \dots, R_k\}$ where all R_i are inter-crossing colors in $\Omega_t(\mathfrak{A}_t)$, and let E be the relation to be created. Let S_i and T_i be the corresponding colors for R_i for every $i \in [k]$ (Property A6). We update

$$\begin{aligned} \omega_{t+1} &:= \omega_t \text{ and} \\ \kappa_{t+1} &:= \kappa_t \cup \Big\{(E, \{\, \{S_i, T_i\} \mid i \in [k] \,\})\Big\}. \end{aligned}$$

Because the cloud of M is not modified, $\mathfrak{A}_{t+1} = \mathfrak{A}_t$ is still normalized. In particular, Property S2 is maintained because no new vertices are created. By construction Property S1 is satisfied. Again by the same reasons as before, the remaining properties are satisfied and the $(t+1)$-th step is simulated.

`refine`(U, k): Because $\hat{\mathcal{M}}$ is pure, `refine` is only executed for fibers. Let U_1 and U_2 be the plain fibers corresponding to U and let E_U be the crossing color such that $E_U^{\mathfrak{A}} = E_{\{U_1,U_2\}}^{\mathfrak{A}}$, that is, E_U precisely connects all the pairs of vertices corresponding to a U-vertex. We cannot simply execute `refine`(E_U, k, Ω_t) because this might create a relation which connects the two components of $\mathfrak{A}_t$ and so the content of the cloud would not be normalized anymore (recall the example in Section 5.3.4.2). We are going to refine U_1 and U_2, then decompose E_U into multiple colors such that either all edges of a color are accepted by $\mathcal{M}_k$ or no edge of the color is. This results in the structure $\mathfrak{A}_{t+1}$. We then refine one (crossing) color after the other. This way, for every color, either no new relation or an empty relation is created and the structure stays normalized.

Let $v, v' \in U_2^{\mathfrak{A}_t}$ be in the same component of $\mathfrak{A}_t$, say w.l.o.g. $v, v' \in U_2^{\mathfrak{A}_t} \cap \mathsf{V}_2(\mathfrak{A}_t)$. Then by Lemma 5.41, we can distinguish v and v' using the sets

$$\Big\{ \mathsf{run}(\mathcal{M}_k, (\mathfrak{A}, \{w, v\}) \;\Big|\; w \in U_1^{\mathfrak{A}_t} \cap \mathsf{V}_1(\mathfrak{A}_t) \Big\} \text{ and}$$
$$\Big\{ \mathsf{run}(\mathcal{M}_k, (\mathfrak{A}, \{w, v'\}) \;\Big|\; w \in U_1^{\mathfrak{A}_t} \cap \mathsf{V}_1(\mathfrak{A}_t) \Big\}$$

if for some $u \in U_1^{\mathfrak{A}_t}$ we have that $(\mathfrak{A}_t, \{u, v\})$ is accepted by $\mathcal{M}_k$ but $(\mathfrak{A}_t, \{u, v'\})$ is not. There is a normalized DeepWL+WSC-algorithm computing a function distinguishing the same (or more) vertices as these sets according to Lemma 5.43 (note that because we input all (u, v) or $\{u, v\}$ during the `refine`-operation, $\mathcal{M}_k$ indeed never fails because otherwise $\hat{\mathcal{M}}_k$ would have failed). We can refine U_2 according to these sets by Lemma 5.44. The same procedure is performed for U_1. After that, we obtain the structure $\mathfrak{A}'_{t+1}$ in the cloud.

Before we can use $\mathcal{M}_k$ to refine the edges, we have to slightly modify it. First assume that E_U is undirected. Then the input of $\mathcal{M}_k$ is $\mathfrak{A}'_t := ((\mathfrak{A}'_{t+1}, \{u,v\}), \Omega'_t)$ where some $\{u,v\} \in E_U^{\mathfrak{A}_t}$ is individualized, i.e., there is a new vertex class $C_{\{u,v\}}$ only containing u and v. This corresponds to individualizing one U-vertex (recall that E_p does *not* distinguish the two components of $\mathfrak{A}'_t$). In order to represent this individualization of the U-vertex, the algorithm $\mathcal{M}_k$ modifies the building plan as follows: $\kappa'_t := \kappa_t \cup \{(F', \{(C_{\{u,v\}}, C_{\{u,v\}})\})\}$ and $\Omega'_t := (\omega_t, \kappa'_t)$. In this way, the vertex w corresponding to $\{u,v\}$ in $\Omega'_t(\mathfrak{A}'_t)$ gets individualized by the relation F'. Now $(\mathfrak{A}'_t, \Omega'_t)$ simulates the structure $(\hat{\mathfrak{A}}_t, w)$, where the U-vertex w is individualized. All conditions for the induction hypothesis are satisfied and we can run the algorithm $\mathcal{M}_k$ by the induction hypothesis. If E_U is directed, the same argument applies: If E_U is directed, then $E_U^{\mathfrak{A}_t} \subseteq U_1^{\mathfrak{A}_t} \times U_2^{\mathfrak{A}_t}$ and thus the ordered tuple (u,v) can be recovered from the set $\{u,v\}$.

So we decompose E into colors $R_1, \dots, R_m$ and execute $\texttt{refine}(R_i, k, \Omega_t)$ for every $i \in [m]$. By Lemma 5.41, either all edges in a color are accepted or none of them. So the machine can compute the set $I := \{i \in [m] \mid \mathcal{M}_k \text{ accepts the } R_i\text{-edges}\}$ by distinguishing for every i whether no relation is created (so $i \in I$) or an empty relation is created (so $i \notin I$). After that, the structure $\mathfrak{A}_{t+1}$ is in the cloud. In particular, $\mathfrak{A}_{t+1}$ and all intermediate steps are normalized because we only create empty relations. Let $\{S_i, T_i\}$ be the corresponding colors for R_i for every $i \in [m]$. We create the vertex class D in $\Omega_t(\mathfrak{A}_t)$, which is the result of the $\texttt{refine}(U,k)$-operation of $\hat{M}$, as follows:

$$\begin{aligned}\omega_{t+1} &:= \omega_t,\\ \kappa_{t+1} &:= \kappa_t \cup \Big\{(D, \{\,\{S_i, T_i\} \mid i \in I\,\})\Big\}.\end{aligned}$$

This establishes Property S1. The other properties still hold for the same reasons as before and the $(t+1)$-th step is simulated.

$\texttt{choice}(U)$**:** Recall that in this case $\hat{M} = \hat{M}^{\text{out}}$ because $\hat{M}^{\text{wit}}$ is choice-free. Again because $\hat{\mathcal{M}}$ is pure, U is a crossing fiber in $\Omega_t(\mathfrak{A}_t)$. The machine defines the undirected relation E_U as in the $\texttt{refine}$-case and executes $\texttt{choice}(E_U)$. By the semantics of the $\texttt{choice}$-operator, this individualizes an undirected E_U-edge (recall that, to individualize this edge, we obtain a vertex class containing both endpoints of the edge and the two components of $\mathfrak{A}_t$ are not connected). Using this edge we can individualize the corresponding crossing vertex in $\Omega_t(\mathfrak{A}_t)$ by defining a singleton fiber V similar to the $\texttt{refine}$-case. This shows Property S1. Properties S2, S3, and S4 still hold as seen before. To show Property S5, observe that every automorphism of $\Omega_{t+1}^{\text{cross}}(\mathfrak{A}_{t+1})$ is an automorphism of $\Omega_t^{\text{cross}}(\mathfrak{A}_t)$, which additionally stabilizes the singleton crossing vertex in V. Thus, such an automorphism also has to stabilize the corresponding undirected E_U-edge and so it is an automorphism of $\Omega_{t+1}(\mathfrak{A})$. Vice versa, every automorphism stabilizing this edge also stabilizes the crossing vertex and the $(t+1)$-th step is simulated.

Finally, we have to alter M^{wit}: When $\hat{M}^{\text{wit}}$ writes the tuple of automorphism-encoding relations $(E_{\text{aut}}, E_{\text{dom}}, E_{\text{img}})$ onto the interaction-tape, M^{wit} only constructed relation plans for these relations. Now M^{wit} writes $(\Omega, E_{\text{aut}}, E_{\text{dom}}, E_{\text{img}})$ onto the interaction-tape, where Ω is the building plan maintained by M^{wit}. To show that $(M^{\text{out}}, M^{\text{wit}}, \mathcal{M}_1, \dots, \mathcal{M}_\ell)$ indeed simulates $(\hat{M}^{\text{out}}, \hat{M}^{\text{wit}}, \hat{\mathcal{M}}_1, \dots, \hat{\mathcal{M}}_\ell)$, it remains to show that all choices are witnessed (if not all choices of $\hat{M}$ are witnessed, M is allowed to do anything). First, the machine M^{out} only executes a $\texttt{choice}(E_U)$-operation when $\hat{M}^{\text{out}}$ executes the operation $\texttt{choice}(U)$. Because E_U is undirected and due to the semantics of the choice operator for undirected edges, we have to witness that $\tilde{E}_U := \tilde{E}_{\{U_1,U_2\}}$ is an orbit, where $\{U_1, U_2\}$ are the corresponding plain fibers of U (cf. the definition of $E_U = E_{\{U_1,U_2\}}$). Assume that $\texttt{choice}(U)$ is executed with the HF-structure $\hat{\mathfrak{A}}_t$ in the cloud. Let N be the set of automorphisms encoded by $(E_{\text{aut}}, E_{\text{dom}}, E_{\text{img}})$, which witnesses U as orbit stabilizing all previous steps $\hat{\mathfrak{A}}_{j_1}, \dots, \hat{\mathfrak{A}}_{j_\ell}$ (where $j_1 < \dots < j_\ell$), in which $\texttt{choice}$-operations where executed by $\hat{M}^{\text{out}}$. Then the tuple $(\Omega, E_{\text{aut}}, E_{\text{dom}}, E_{\text{img}})$ precisely encodes the same set of automorphisms N for $\mathfrak{A}_t$: For every vertex $\hat{w}_\varphi$ encoding a partial map φ via $(E_{\text{aut}}, E_{\text{dom}}, E_{\text{img}})$ in $\hat{\mathfrak{A}}_t$, the corresponding vertex w_φ in $\Omega(\mathfrak{A}_t)$ encodes the same partial map because of the following. By the definition of encoding automorphisms by DeepWL+WSC-algorithms, the vertex $\hat{w}_\varphi$ encodes that $\varphi(\hat{u}) = \hat{v}$ for atoms $\hat{u}$ and $\hat{v}$ if there is a unique vertex $\hat{w}$ such that $(\hat{w}_\varphi, \hat{w})$ is in E_{aut}, $(\hat{w}, \hat{u})$ is in E_{dom}, and $(\hat{w}, \hat{v})$ is in E_{img}. By Properties S3 and S4, the corresponding vertices w, u, and v in $\Omega(\mathfrak{A})$ are crossing vertices such that u has $\hat{u}$ as E_p-neighbor (and no other atom) and likewise v has $\hat{v}$ as E_p-neighbor. This exactly is the definition of encoding automorphisms for normalized DeepWL+WSC-algorithms. By Property S5 and Lemma 5.25, the automorphisms of $\mathfrak{A}_t$ and $\hat{\mathfrak{A}}_t$ are equal. By Lemma 5.51, N witnesses $\tilde{E}_U$ as orbit stabilizing $\tilde{E}_{U_1}^{\mathfrak{A}_{j_1}}, \dots, \tilde{E}_{U_\ell}^{\mathfrak{A}_{j_\ell}}$ in the simulation.

It remains to argue that the algorithm runs in polynomial time. Note that M needs polynomially many steps to simulate a single step of $\hat{M}$. So it remains to argue that the structure in the cloud is of polynomial size. By Property S2, the building plans are always efficient. Then, by Lemma 5.50, we have that $|\mathfrak{A}_t| \leq 2|\hat{\mathfrak{A}}_t|$. Because additionally the number of relations in $\mathfrak{A}_t$ is polynomially bounded, $|D(\mathfrak{A}_t)|$ is polynomially bounded, too. ☐

Corollary 5.58. *Let $\mathcal{K}$ be a class of binary structures. If there is a polynomial-time DeepWL+WSC-algorithm deciding isomorphism for $\mathcal{K}$, then there is a normalized polynomial-time DeepWL+WSC-algorithm deciding isomorphism for $\mathcal{K}$.*

Proof. Let $\hat{\mathcal{M}}$ be a polynomial-time DeepWL+WSC-algorithm deciding isomorphism for $\mathcal{K}$ and let $\mathcal{M}$ be a normalized polynomial-time DeepWL+WSC-algorithm simulation $\hat{\mathcal{M}}$ given by Lemma 5.57. Let $\mathfrak{A}_1, \mathfrak{A}_2 \in \mathcal{K}$ and $\mathfrak{A} = \mathfrak{A}_1 \uplus \mathfrak{A}_2$. To execute $\mathcal{M}$ to decide isomorphism, we first have to construct a building plan Ω such that $(\mathfrak{A}, \Omega)$ simulates $\mathfrak{A}$. This is done as follows: First, we define the vertex class D of all atoms. Second, we create a plain vertex u_i for both components $\mathfrak{A}_i$, and a vertex class D' only containing u_1 and u_2. To do so, we define a relation F connecting all atoms in the same component. We execute $\texttt{scc}(F)$ and obtain a vertex class D' containing one vertex per component. Then we rename every relation $E \in \tau$ to a fresh relation symbol E'. We set $\Omega := (\omega, \kappa)$, where

$$\omega := \Big\{\{D, D'\}\Big\} \text{ and}$$
$$\kappa := \Big\{(C, \{D, D'\})\Big\} \cup \Big\{ (E, \{\{E', D'\}\}) \,\Big|\, E \in \tau \Big\}.$$

We claim that $(\mathfrak{A}, \Omega)$ simulates $\mathfrak{A}$. With ω, we create, for every atom $u \in A_i$ in D, a single crossing vertex because D' contains exactly one vertex in $\mathfrak{A}_j$ (where $\{i, j\} = [2]$).

Two such crossing vertices u and v are in the relation E if between the corresponding vertices there is a D'-edge (the D'-loop) and an E'-edge. That is, $u^{(i)} = v^{(i)}$ is a D'-vertex, so u and v are copies of plain vertices of the same component $\mathfrak{A}_j$, and $(u^{(j)}, v^{(j)}) \in E^{\mathfrak{A}}$, where E was renamed to E'. So we have established Property S1. Clearly, every vertex is used in the building plan because every vertex is either in D or in D', thus Property S2 is satisfied. Every crossing vertex is adjacent via E_p to exactly one atom in A because the D'-vertices are not atoms. The relation plan $(C, \{D, D'\})$ creates a vertex class C containing all crossing vertices. Because each crossing vertex is E_p-connected to an atom and every atom to a C-vertex, we established Properties S3 and S4. Lastly, we also established Property S5 because $\Omega(\mathfrak{A})$ consists essentially of two copies of $\mathfrak{A}$ connected by a perfect matching. A similar construction (but without building plans) can be found in the proof of Lemma 11 in [59]. □

5.3.4.7 Deciding Isomorphism and Internal Runs

In this section we prove that, for every class of structures $\mathcal{K}$, isomorphism is decidable by a polynomial-time DeepWL+WSC-algorithm if and only if a complete invariant is computable by a polynomial-time DeepWL+WSC-algorithm. To do so, we consider the internal runs of normalized DeepWL+WSC-algorithms. We say that the components of a normalized HF-structure $\mathfrak{A}_1 \uplus \mathfrak{A}_1$ are **distinguished** if $D(\mathfrak{A}_1) \neq D(\mathfrak{A}_2)$.

Lemma 5.59. *For every normalized polynomial-time DeepWL+WSC-algorithm $\mathcal{M}$, there exists a normalized polynomial-time DeepWL+WSC-algorithm $\mathcal{M}'$ with the following properties:*

(a) For every normalized HF-structure $\mathfrak{A}$, the algorithm $\mathcal{M}'$ accepts (respectively rejects) $\mathfrak{A}$ whenever $\mathcal{M}$ accepts (respectively rejects) $\mathfrak{A}$.

(b) For all normalized HF-structures $\mathfrak{A} = \mathfrak{A}_1 \uplus \mathfrak{A}_2$ and $\mathfrak{B} = \mathfrak{B}_1 \uplus \mathfrak{B}_2$, it holds that if $\mathsf{run}(\mathcal{M}', \mathfrak{A}) = \mathsf{run}(\mathcal{M}', \mathfrak{B})$, then $\mathsf{run}(\mathcal{M}', \mathfrak{A}) \in \{\mathsf{run}(\mathcal{M}', \mathfrak{A}_1 \uplus \mathfrak{B}_i) \mid i \in [2]\}$.

Proof. A normalized DeepWL+WSC-algorithm is **nice** on input $\mathfrak{A}$ if

1. whenever `refine`(X, i) is executed and the components of the current structure in the cloud $\mathfrak{A}'$ are not distinguished, then $X^{\mathfrak{A}'} = \{u_1, u_2\}$ such that u_1 and u_2 are in different components of $\mathfrak{A}'$,
2. for every other `refine`(X, i)-execution, X is a plain color, and
3. whenever `choice`(R) is executed, then the two components of the current structure in the cloud are distinguished and R is a plain color.

Note that if $\mathcal{M} = (M^{\text{out}}, M^{\text{wit}}, \mathcal{M}_1, \dots, \mathcal{M}_\ell)$ is nice on input $\mathfrak{A}$, then $\mathcal{M}_1, \dots, \mathcal{M}_\ell$ must be nice on all inputs during `refine`-operations of M^{out} or M^{wit}.

Claim 1. *For every normalized polynomial-time DeepWL+WSC-algorithm $\mathcal{M}$, there is normalized polynomial-time DeepWL+WSC-algorithm $\mathcal{M}'$ such that, for every normalized HF-structure $\mathfrak{A}$ on which $\mathcal{M}$ does not fail, $\mathcal{M}'$ is nice on $\mathfrak{A}$ and $\mathcal{M}'$ accepts (respectively rejects) $\mathfrak{A}$ if $\mathcal{M}$ accepts (respectively rejects) $\mathfrak{A}$.*

Proof. Let $\mathcal{M}$ be a normalized polynomial-time DeepWL+WSC-algorithm. Let $\mathcal{M}''$ be the normalized polynomial-time DeepWL+WSC-algorithm given for $\mathcal{M}$ by Lemma 5.40. That is, for every normalized HF-structure $\mathfrak{A}_1 \uplus \mathfrak{A}_2$, $\mathcal{M}''$ accepts (respectively rejects) $(\mathfrak{A}_1 \uplus \mathfrak{A}_2, A_i)$ whenever $\mathcal{M}$ accepts (respectively rejects) $\mathfrak{A}_1 \uplus \mathfrak{A}_2$ for every $i \in [2]$ (the lemma states it only for $i = 1$, the case $i = 2$ follows from exchanging the two components). Then define the DeepWL+WSC-algorithm $\mathcal{M}' = (M^{\text{out}}, M^{\text{wit}}, \mathcal{M}'')$ as follows: First, the machine M^{out} creates a vertex class C containing one vertex per component by executing $\texttt{scc}(E)$ for the relation E of plain edges. Next, M^{out} executes $\texttt{refine}(C, 1)$, i.e., M^{out} refines C with $\mathcal{M}''$ (if C decomposes into two fibers U and V, then M^{out} executes $\texttt{refine}(U, 1)$ because we are only allowed to refine colors). Individualizing one C-vertex corresponds to creating a relation containing the vertices of one component, so $\mathcal{M}''$ will accept (respectively reject) the C-vertices whenever $\mathcal{M}$ accepts (respectively rejects) the input to $\mathcal{M}$. Then M accepts (respectively rejects) accordingly. The witnessing machine M^{wit} is not used and immediately halts. Clearly, $\mathcal{M}'$ accepts (respectively rejects) if $\mathcal{M}$ does so. We show how we have to modify $\mathcal{M}'$ such that $\mathcal{M}'$ is nice on every input:

1. By construction, $\mathcal{M}'$ executes a single $\texttt{refine}$-operation when the two components are not distinguished. Indeed, the refined relation is a vertex class containing a single vertex of each component. After that, the two components are distinguished (and remain so) because one of the two vertices is individualized.

2. A $\texttt{choice}(R)$-operation is only executed after the initial $\texttt{refine}$-operation, so the components of the structure $\mathfrak{A}$ in the cloud are distinguished. Assume that R is a crossing color, then $R^{\mathfrak{A}} \subseteq U^{\mathfrak{A}} \times V^{\mathfrak{A}}$ for two fibers U and V in different components because $\mathfrak{A}$ is normalized. We can equivalently execute $\texttt{choice}(U)$ and $\texttt{choice}(V)$. The automorphisms witnessing $\texttt{choice}(R)$ will also witness the two other operations because $R^{\mathfrak{A}}$ is an orbit if and only if $U^{\mathfrak{A}}$ and $V^{\mathfrak{A}}$ are orbits.

3. For all $\texttt{refine}(X, i)$-operations apart from the first one, the components of the structure $\mathfrak{A}$ in the cloud are distinguished. By decomposing X into its colors, we can assume that X is a color R. If R is a crossing color, then again $R^{\mathfrak{A}} \subseteq U^{\mathfrak{A}} \times V^{\mathfrak{A}}$ for two different fibers U and V in different components because $\mathfrak{A}$ is normalized and its components are distinguished. Let $\mathcal{M}_i$ be the DeepWL+WSC-algorithm used to refine R. We know that either every $(u, v) \in R^{\mathfrak{A}}$ is accepted by $\mathcal{M}_i$ or every $(u, v) \in R^{\mathfrak{A}}$ is rejected by $\mathcal{M}_i$ because $\mathcal{M}$ is normalized. These are the two only possibilities without creating a crossing relation, which would make the structure non-normalized. So we can equivalently execute two nested $\texttt{refine}$-operations: First, we refine U with a new algorithm $\mathcal{M}_i'$. The algorithm $\mathcal{M}_i'$, which gets $(\mathfrak{A}, u)$ for some $u \in U^{\mathfrak{A}}$ as input, immediately refines V with the algorithm $\mathcal{M}_i$, which then gets $(\mathfrak{A}, uv)$ for some $v \in V^{\mathfrak{A}}$ as input. The algorithm $\mathcal{M}_i'$ accepts $(\mathfrak{A}, u)$ if $\mathcal{M}_i$ accepts $(\mathfrak{A}, uv)$ for every $v \in V^{\mathfrak{A}}$. If all U-vertices are accepted by $\mathcal{M}_i'$, then $(\mathfrak{A}, uv)$ is accepted by $\mathcal{M}_i$ for every $(u, v) \in R^{\mathfrak{A}}$ and thus no relation is created. Otherwise, $\mathcal{M}_i$ rejects $(\mathfrak{A}, uv)$ for every $(u, v) \in R^{\mathfrak{A}}$. Thus, $\mathcal{M}_i'$ rejects $(\mathfrak{A}, u)$ for every $u \in U^{\mathfrak{A}}$ and an empty relation is created. $\dashv$

Claim 2. *For every normalized DeepWL+WSC-algorithm $\mathcal{M}$ and all normalized HF-structures $\mathfrak{A} = \mathfrak{A}_1 \uplus \mathfrak{A}_2$ and $\mathfrak{B} = \mathfrak{B}_1 \uplus A_2$, if $\mathcal{M}$ is nice on $\mathfrak{A}$ and $\mathfrak{B}$ such that* $\mathsf{run}(\mathcal{M}, \mathfrak{A}) = \mathsf{run}(\mathcal{M}, \mathfrak{B}) \neq \dagger$, *then* $\mathsf{run}(\mathcal{M}, \mathfrak{A}) \in \{\mathsf{run}(\mathcal{M}, \mathfrak{A}_1 \uplus \mathfrak{B}_i) \mid i \in [2]\}$.

Proof. The proof is by induction on the nesting depth of DeepWL+WSC-algorithms. Let $\mathcal{M} = (M^{\text{out}}, M^{\text{wit}}, \mathcal{M}_1, \dots, M_\ell)$ be a normalized DeepWL+WSC-algorithm and let $\mathfrak{A} = \mathfrak{A}_1 \uplus \mathfrak{A}_2$ and $\mathfrak{B} = \mathfrak{B}_1 \uplus \mathfrak{B}_2$ be two normalized HF-structures such that $\mathcal{M}$ is nice on $\mathfrak{A}$ and $\mathfrak{B}$ and $\mathsf{run}(\mathcal{M}, \mathfrak{A}) = \mathsf{run}(\mathcal{M}, \mathfrak{B})$. Let $\mathfrak{A}^1, \dots, \mathfrak{A}^m$ be the sequence of structures in the cloud of M^{out} on input $\mathfrak{A}$, let $\mathfrak{A}^i = \mathfrak{A}^i_1 \uplus \mathfrak{A}^i_2$ for every $i \in [m]$, and similarly let $\mathfrak{B}^1, \dots, \mathfrak{B}^m$ be the same sequence on input $\mathfrak{B}$ and $\mathfrak{B}^i = \mathfrak{B}^i_1 \uplus \mathfrak{B}^i_2$ for every $i \in [m]$. Because $\mathsf{run}(\mathcal{M}, \mathfrak{A}) = \mathsf{run}(\mathcal{M}, \mathfrak{B})$ implies $\mathsf{run}(M^{\text{out}}, \mathfrak{A}) = \mathsf{run}(M^{\text{out}}, \mathfrak{B})$, the two sequences have indeed the same length and satisfy $D(\mathfrak{A}^j) = D(\mathfrak{B}^j)$ for every $j \in [m]$. From Lemma 5.39 it follows that

$$\left\{ D(\mathfrak{A}^j_i) \;\middle|\; i \in [2] \right\} = \left\{ D(\mathfrak{B}^j_i) \;\middle|\; i \in [2] \right\}$$

for every $j \in [m]$. Moreover, if the components of $\mathfrak{A}^j$ are distinguished for some $j \in [m]$, then the components of $\mathfrak{A}^k$ are distinguished for every $k \geq j$ (and likewise for the $\mathfrak{B}^j$). So there is a permutation $\rho\colon [2] \to [2]$ such that $D(\mathfrak{A}^j_i) = D(\mathfrak{B}^j_{\rho(i)})$ for every $i \in [2]$ and $j \in [m]$. Assume w.l.o.g. that ρ is the identity map. Consider the HF-structure $\mathfrak{H} = \mathfrak{A}_1 \uplus \mathfrak{B}_2$. We claim that

$$\mathsf{run}(\mathcal{M}, \mathfrak{H}) = \mathsf{run}(\mathcal{M}, \mathfrak{A}) = \mathsf{run}(\mathcal{M}, \mathfrak{B}).$$

We first show that $\mathsf{run}(M^{\text{out}}, \mathfrak{H}) = \mathsf{run}(M^{\text{out}}, \mathfrak{A}) = \mathsf{run}(M^{\text{out}}, \mathfrak{B})$. In particular, the sequence of structures $\mathfrak{H}^1, \dots, \mathfrak{H}^{m'}$ in the cloud of M^{out} on input $\mathfrak{H}$ will satisfy $m' = m$ and $\mathfrak{H}^j_1 = \mathfrak{A}^j_1$ and $\mathfrak{H}^j_2 = \mathfrak{B}^j_2$ for every $j \in [m]$. This implies by Lemma 5.39 that $D(\mathfrak{H}^j) = D(\mathfrak{A}^j) = D(\mathfrak{B}^j)$ for every $j \in [m]$. We show by induction on j, that $\mathfrak{H}^j_1 = \mathfrak{A}^j_1$ and $\mathfrak{H}^j_2 = \mathfrak{B}^j_2$ and that the configuration of M^{out} is equal when the j-th cloud modifying operation is performed on input $\mathfrak{A}$, $\mathfrak{B}$, and $\mathfrak{H}$ for all $j \in [m]$ (for $\mathfrak{A}$ and $\mathfrak{B}$ the claim follows from $\mathsf{run}(M^{\text{out}}, \mathfrak{A}) = \mathsf{run}(M^{\text{out}}, \mathfrak{B})$).

For $j = 1$, this is the case by construction: $\mathfrak{H}^1 = \mathfrak{H} = \mathfrak{A}_1 \uplus \mathfrak{B}_2$ and M^{out} is started in its initial state. So assume that the claim holds for $j \geq 1$ and we show that it holds for $j + 1$. Because the sketches of the structures in the cloud are equal and the Turing machine of M^{out} is in the same configuration, the run of M^{out} on $\mathfrak{H}_j$ is equal to the one on $\mathfrak{A}_j$ (or equally on $\mathfrak{B}_j$) until the next cloud-modifying operation is executed. Because the Turing machines are in the same configuration, the same operation is executed for $\mathfrak{H}^j$ as for $\mathfrak{A}^j$ and $\mathfrak{B}^j$.

(a) If the operation modifying the cloud is an `addPair`-, `scc`-, or `create`-operation, then the effect of the operation is clearly given by the effect on each component because M^{out} is normalized. So, since $\mathfrak{H}^j_1 = \mathfrak{A}^j_1$ and $\mathfrak{H}^j_2 = \mathfrak{B}^j_2$, we have that $\mathfrak{H}^{j+1}_1 = \mathfrak{A}^{j+1}_1$ and $\mathfrak{H}^{j+1}_2 = \mathfrak{B}^{j+1}_2$.

(b) Assume that M^{out} executes `refine`(R, k). Because $\mathsf{run}(M^{\text{out}}, \mathfrak{A}) = \mathsf{run}(M^{\text{out}}, \mathfrak{B})$, we have

$$\left\{\!\!\left\{ \mathsf{run}(\mathcal{M}_k, (\mathfrak{A}^j, x)) \;\middle|\; x \in \tilde{R}^{\mathfrak{A}^j} \right\}\!\!\right\} = \left\{\!\!\left\{ \mathsf{run}(\mathcal{M}_k, (\mathfrak{B}^j, x)) \;\middle|\; x \in \tilde{R}^{\mathfrak{B}^j} \right\}\!\!\right\}.$$

Recall that $\tilde{R}^{\mathfrak{A}^j} = X^{\mathfrak{A}^j}$ if R is directed and $\tilde{R}^{\mathfrak{A}^j} = \{\{u,v\} \mid (u,v) \in R^{\mathfrak{A}^j}\}$ if R is undirected. So there is a bijection $\zeta\colon \tilde{R}^{\mathfrak{A}^j} \to \tilde{R}^{\mathfrak{B}^j}$ such that for every $x \in \tilde{R}^{\mathfrak{A}^j}$ we have $\mathsf{run}(\mathcal{M}_k, (\mathfrak{A}^j, x)) = \mathsf{run}(\mathcal{M}_k, (\mathfrak{B}^j, \zeta(x)))$.

Assume first that R is a crossing color and undirected in $\mathfrak{A}^j$, $\mathfrak{B}^j$, and thus in $\mathfrak{H}^j$. So the components are not distinguished in all three structures (otherwise, all crossing colors are directed). Because $\mathcal{M}$ is nice on $\mathfrak{A}$ and $\mathfrak{B}$, R is a fiber and contains one vertex per component. So assume $R^{\mathfrak{A}^j} = \{u_1, u_2\}$ and $R^{\mathfrak{B}^j} = \{v_1, v_2\}$, and thus $R^{\mathfrak{H}^j} = \{u_1, v_2\}$. We can assume that ζ satisfies $\zeta(u_i) = v_i$ for every $i \in [2]$ because we assumed that ρ is the identity map and so $D(\mathfrak{A}^j_i) = D(\mathfrak{B}^j_i)$ for every $i \in [2]$. That is, $\mathsf{run}(\mathcal{M}_k, (\mathfrak{A}^j, u_i)) = \mathsf{run}(\mathcal{M}_k, (\mathfrak{B}^j, v_i))$ for every $i \in [2]$.

Let $w \in R^{\mathfrak{H}_j}$ and w.l.o.g. assume that $w = u_1$ (the case $w = v_2$ is symmetric). So $(\mathfrak{H}^j_1, w) = (\mathfrak{A}^j_1, u_1)$, $\mathfrak{H}^j_2 = \mathfrak{B}^j_2$, and thus $(\mathfrak{H}^j, w) = (\mathfrak{A}^j_1, u_1) \uplus \mathfrak{B}^j_2$. Because $\mathsf{run}(\mathcal{M}_k, (\mathfrak{A}^j, u_1)) = \mathsf{run}(\mathcal{M}_k, (\mathfrak{B}^j, v_1))$ and $\mathcal{M}_k$ is nice on $\mathfrak{A}^j$ and $\mathfrak{B}^j$, we can apply the outer induction hypothesis and conclude that

$$\mathsf{run}(\mathcal{M}_k, (\mathfrak{H}^j, w)) = \mathsf{run}(\mathcal{M}_k, (\mathfrak{A}^j, u_1)) = \mathsf{run}(\mathcal{M}_k, (\mathfrak{B}^j, v_1)).$$

Note that we assumed that ρ is the identity map and hence

$$\begin{gathered}\text{if } \mathsf{run}(\mathcal{M}_k, \mathfrak{A}^j) \in \{\, \mathsf{run}(\mathcal{M}_k, \mathfrak{A}^j_1 \uplus \mathfrak{B}^j_i) \mid i \in [2]\,\},\\ \text{then } \mathsf{run}(\mathcal{M}_k, \mathfrak{A}^j) = \mathsf{run}(\mathcal{M}_k, \mathfrak{A}^j_1 \uplus \mathfrak{B}^j_2) = \mathsf{run}(\mathcal{M}_k, \mathfrak{H}^j).\end{gathered}$$

In particular, $\mathcal{M}_k$ accepts (respectively rejects) $(\mathfrak{H}^j, u_1)$ if and only if $\mathcal{M}_k$ accepts (respectively rejects) $(\mathfrak{A}^j, u_1)$. The same holds (by symmetry) for $(\mathfrak{H}^j, v_2)$ and $(\mathfrak{B}^j, u_2)$. Thus, the resulting vertex class D of the `refine`-operation satisfies $D^{\mathfrak{H}^{j+1}} = D^{\mathfrak{A}^{j+1}_1} \cup D^{\mathfrak{B}^{j+1}_2}$. That is, $\mathfrak{H}^{j+1}_1 = \mathfrak{A}^{j+1}_1$, $\mathfrak{H}^{j+1}_2 = \mathfrak{B}^{j+1}_2$, and

$$\begin{aligned}\left\{\!\!\left\{ \mathsf{run}(\mathcal{M}_k, (\mathfrak{H}^j, w)) \;\middle|\; w \in \{u_1, v_2\} \right\}\!\!\right\} &= \left\{\!\!\left\{ \mathsf{run}(\mathcal{M}_k, (\mathfrak{A}^j, w)) \;\middle|\; w \in \{u_1, u_2\} \right\}\!\!\right\}\\ &= \left\{\!\!\left\{ \mathsf{run}(\mathcal{M}_k, (\mathfrak{B}^j, w)) \;\middle|\; w \in \{v_1, v_2\} \right\}\!\!\right\}.\end{aligned}$$

If otherwise R is not crossing or not undirected, then R is a plain color in $\mathfrak{A}^j$ (and thus in $\mathfrak{B}^j$ and $\mathfrak{H}^j$) because $\mathcal{M}$ is nice. So every $x \in \tilde{R}^{\mathfrak{H}^j}$ consists solely of vertices of either $\mathsf{V}(\mathfrak{H}^j_1)$ or $\mathsf{V}(\mathfrak{H}^j_2)$. Assume w.l.o.g. that x consists of vertices of $\mathsf{V}(\mathfrak{H}^j_1)$. Then $(\mathfrak{H}^j_1, x) = (\mathfrak{A}^j_1, x)$, $\mathfrak{H}^j_2 = \mathfrak{B}^j_2$, and $(\mathfrak{H}^j, x) = (\mathfrak{A}^j_1, x) \uplus \mathfrak{B}^j_2$.

Because $\mathsf{run}(\mathcal{M}_k, (\mathfrak{A}^j, x)) = \mathsf{run}(\mathcal{M}_k, (\mathfrak{B}^j, \zeta(x)))$, we can apply the outer induction hypothesis to $\mathcal{M}_k$ and obtain that

$$\mathsf{run}(\mathcal{M}_k, (\mathfrak{A}^j, x)) = \mathsf{run}(\mathcal{M}_k, (\mathfrak{B}^j, \zeta(x))) = \mathsf{run}(\mathcal{M}_k, (\mathfrak{H}^j, x)).$$

In particular, $\mathcal{M}_k$ accepts $(\mathfrak{H}^j, x)$ if and only if it accepts $(\mathfrak{A}^j, x)$. So let Y be the relation obtained from `refine`(R, k). Then $Y^{\mathfrak{H}^{j+1}} = Y^{\mathfrak{H}^{j+1}_1} \cup Y^{\mathfrak{H}^{j+1}_2}$, $Y^{\mathfrak{H}^{j+1}_1} = Y^{\mathfrak{A}^{j+1}_1}$, and $Y^{\mathfrak{H}^{j+1}_2} = Y^{\mathfrak{B}^{j+1}_2}$. That is, $\mathfrak{H}^{j+1}_1 = \mathfrak{A}^{j+1}_1$ and $\mathfrak{H}^{j+1}_2 = \mathfrak{B}^{j+1}_2$.

Because the components of $\mathfrak{A}^j$ and $\mathfrak{B}^j$ are distinguished, ζ maps an edge of the i-th component of $\mathfrak{A}^j$ to the i-th component of $\mathfrak{B}^j$ (otherwise the sketches differ immediately). That is,

$$\begin{aligned}\left\{\!\!\left\{ \mathsf{run}(\mathcal{M}_k, (\mathfrak{A}^j, x)) \;\middle|\; x \in \tilde{R}^{\mathfrak{A}^j} \right\}\!\!\right\} = {}&\left\{\!\!\left\{ \mathsf{run}(\mathcal{M}_k, (\mathfrak{A}^j, x)) \;\middle|\; x \in \tilde{R}^{\mathfrak{A}^j} \cap \mathsf{V}(\mathfrak{A}^j_1) \right\}\!\!\right\} \cup \\ &\left\{\!\!\left\{ \mathsf{run}(\mathcal{M}_k, (\mathfrak{A}^j, x)) \;\middle|\; x \in \tilde{R}^{\mathfrak{A}^j} \cap \mathsf{V}(\mathfrak{A}^j_2) \right\}\!\!\right\}\end{aligned}$$

and likewise for $\mathfrak{B}_j$. So we have

$$\begin{aligned}\left\{\!\!\left\{ \mathsf{run}(\mathcal{M}_k, (\mathfrak{H}^j, x)) \;\middle|\; x \in \tilde{R}^{\mathfrak{H}^j} \right\}\!\!\right\} = {}&\left\{\!\!\left\{ \mathsf{run}(\mathcal{M}_k, (\mathfrak{A}^j, x)) \;\middle|\; x \in \tilde{R}^{\mathfrak{A}_j} \cap \mathsf{V}(\mathfrak{A}^j_1) \right\}\!\!\right\} \cup \\ &\left\{\!\!\left\{ \mathsf{run}(\mathcal{M}_k, (\mathfrak{B}^j, x)) \;\middle|\; x \in \tilde{R}^{\mathfrak{B}^j} \cap \mathsf{V}(\mathfrak{B}^j_2) \right\}\!\!\right\}\end{aligned}$$

and thus

$$\begin{aligned}\left\{\!\!\left\{ \mathsf{run}(\mathcal{M}_k, (\mathfrak{H}^j, x)) \;\middle|\; x \in \tilde{R}^{\mathfrak{H}^j} \right\}\!\!\right\} &= \left\{\!\!\left\{ \mathsf{run}(\mathcal{M}_k, (\mathfrak{A}^j, x)) \;\middle|\; x \in \tilde{R}^{\mathfrak{A}^j} \right\}\!\!\right\} \\ &= \left\{\!\!\left\{ \mathsf{run}(\mathcal{M}_k, (\mathfrak{B}^j, x)) \;\middle|\; x \in \tilde{R}^{\mathfrak{B}^j} \right\}\!\!\right\}.\end{aligned}$$

(c) Assume that M^{out} executes $\texttt{choice}(R)$. Because $\mathcal{M}$ is nice on $\mathfrak{A}$ and $\mathfrak{B}$, R is a plain color and the components of $\mathfrak{A}^j$ and $\mathfrak{B}^j$ are distinguished. That is,

$$D(\mathfrak{A}_1) = D(\mathfrak{B}_1) = D(\mathfrak{H}_1) \neq D(\mathfrak{A}_2) = D(\mathfrak{B}_2) = D(\mathfrak{H}_2).$$

Hence, the components of $\mathfrak{H}$ are distinguished, too. Because R is plain, R occurs solely in one component, say w.l.o.g. the first. Let $x \in \tilde{R}^{\mathfrak{H}^j}$ be a chosen element and $\mathfrak{H}^{j+1} = (\mathfrak{H}^j, x)$. Then $(\mathfrak{H}^j_1, x) = (\mathfrak{A}^j_1, x)$, $\mathfrak{H}^j_2 = \mathfrak{B}^j_2$, and $(\mathfrak{A}^j, x) = (\mathfrak{A}^j_1, x) \uplus \mathfrak{B}^j_2$ given that we also chose x in the execution of M^{out} on $\mathfrak{A}$ (which for the sketch of course does not matter if all choices are witnessed).

We have proved that M^{out} satisfies that $\mathsf{run}(M^{\text{out}}, \mathfrak{H}) = \mathsf{run}(M^{\text{out}}, \mathfrak{A}) = \mathsf{run}(M^{\text{out}}, \mathfrak{B})$. To show that $\mathsf{run}(\mathcal{M}, \mathfrak{H}) = \mathsf{run}(\mathcal{M}, \mathfrak{A}) = \mathsf{run}(\mathcal{M}, \mathfrak{B})$, it remains to show that all choices are witnessed and that whenever the j-th cloud-modifying operation was a $\texttt{choice}$-operation, then

$$\mathsf{run}(M^{\text{wit}}, \mathfrak{H}^m \Cup \mathfrak{H}^j) = \mathsf{run}(M^{\text{wit}}, \mathfrak{A}^m \Cup \mathfrak{A}^j) = \mathsf{run}(M^{\text{wit}}, \mathfrak{B}^m \Cup \mathfrak{B}^j).$$

So assume that the j-th operation is a $\texttt{choice}(R)$-operation for an arbitrary $j \in [m]$. We have already seen that $\mathfrak{H}^k_1 = \mathfrak{A}^k_1$ and $\mathfrak{H}^k_2 = \mathfrak{B}^k_2$ for every $k \in \{j, m\}$. The same applies to $\mathfrak{H}^m \Cup \mathfrak{H}^j$. By analogous reasoning as for M^{out}, the witnessing machine M^{wit} satisfies

$$\mathsf{run}(M^{\text{wit}}, \mathfrak{H}^m \Cup \mathfrak{H}^j) = \mathsf{run}(M^{\text{wit}}, \mathfrak{A}^m \Cup \mathfrak{A}^j) = \mathsf{run}(M^{\text{wit}}, \mathfrak{B}^m \Cup \mathfrak{B}^j).$$

In particular, the machine M^{wit} writes the same tuple $(\Omega, E_{\text{aut}}, E_{\text{dom}}, E_{\text{img}})$ onto the interaction-tape in all cases. Assume that $(\Omega, E_{\text{aut}}, E_{\text{dom}}, E_{\text{img}})$ witnesses $R^{\mathfrak{A}^j}$ and $R^{\mathfrak{B}^j}$ as orbit (if that is not the case, then $\mathcal{M}$ fails on $\mathfrak{A}$ and $\mathfrak{B}$ and there is nothing to show). Because $\mathcal{M}$ is nice on $\mathfrak{A}$ and $\mathfrak{B}$, a $\texttt{choice}$-operation is executed only if both components are distinguished. So no automorphism maps one component to the other and every automorphism φ of $\mathfrak{A}^j$ is induced by two automorphisms φ_1 and φ_2, one for each component. We write $\varphi = (\varphi_1, \varphi_2)$. Similarly, every automorphism ψ of $\mathfrak{B}^j$ decomposes into $\psi = (\psi_1, \psi_2)$. Then the map (φ_1, ψ_2) defined by φ_1 on the first and by ψ_2 on the second component is an automorphism of $\mathfrak{H}^j$. Because the defined relations E_{aut}, E_{dom}, and E_{img} are isomorphism-invariant (the witnessing machine is choice-free), they encode a set of automorphisms $N^{\mathfrak{A}^j}$ in $\Omega(\mathfrak{A}^m \uplus \mathfrak{A}^j)$ which decomposes into sets of automorphisms $N_1^{\mathfrak{A}^j} = \{\varphi \mid (\varphi, \psi) \in N_1^{\mathfrak{A}^j}\}$ and likewise in $N_2^{\mathfrak{A}^j}$ such that $N^{\mathfrak{A}^j} = N_1^{\mathfrak{A}^j} \times N_2^{\mathfrak{A}^j}$. Similarly, $N^{\mathfrak{B}^j} = N_1^{\mathfrak{B}^j} \times N_2^{\mathfrak{B}^j}$ for $\Omega(\mathfrak{B}^m \uplus \mathfrak{B}^j)$. So for the HF-structure $\mathfrak{H}^m \uplus \mathfrak{H}^j$, the tuple $(\Omega, E_{\mathrm{aut}}, E_{\mathrm{dom}}, E_{\mathrm{img}})$ encodes the set of automorphisms $N^{\mathfrak{H}^j} = N_1^{\mathfrak{A}^j} \times N_2^{\mathfrak{B}^j}$. Clearly, $N^{\mathfrak{H}^j}$ witnesses exactly the same plain relations (which are either contained in $\mathfrak{A}_1^j$ or in $\mathfrak{B}_2^j$) as orbit, which $N_1^{\mathfrak{A}}$ and $N_2^{\mathfrak{B}}$ witness as orbits. ⊣

Finally, let $\mathcal{M}$ be a normalized polynomial-time DeepWL+WSC-algorithm and let $\mathcal{M}'$ be the nice and normalized polynomial-time DeepWL+WSC-algorithm given by Claim 1 for $\mathcal{M}$. Let $\mathfrak{A} = \mathfrak{A}_1 \uplus \mathfrak{A}_2$ and $\mathfrak{B} = \mathfrak{B}_1 \uplus \mathfrak{B}_2$ be two normalized HF-structures on which $\mathcal{M}$ does not fail. Then $\mathcal{M}'$ accepts (respectively rejects) $\mathfrak{H}$ if $\mathcal{M}$ accepts (respectively rejects) $\mathfrak{H}$ for every $\mathfrak{H} \in \{\mathfrak{A}, \mathfrak{B}\}$. Assume that $\mathsf{run}(\mathcal{M}', \mathfrak{A}) = \mathsf{run}(\mathcal{M}', \mathfrak{B})$. Then

$$\mathsf{run}(\mathcal{M}', \mathfrak{A}) \in \left\{ \mathsf{run}(\mathcal{M}', \mathfrak{A}_1 \uplus \mathfrak{B}_i) \;\middle|\; i \in [2] \right\}$$

by Claim 2 because $\mathcal{M}'$ is nice. □

Theorem 5.60. *Let $\mathcal{K}$ be a class of binary τ-structures. Then the following are equivalent:*

1. *There is a polynomial-time DeepWL+WSC-algorithm deciding isomorphism on $\mathcal{K}$.*
2. *There is a polynomial-time DeepWL+WSC-algorithm computing some complete invariant for $\mathcal{K}$.*

Proof. To prove that Condition 2 implies Condition 1, let $\mathcal{M}$ be a DeepWL+WSC-algorithm computing a complete invariant for $\mathcal{K}$. We want, on input $\mathfrak{A} \uplus \mathfrak{B}$ for $\mathfrak{A}, \mathfrak{B} \in \mathcal{K}$, to run the algorithm $\mathcal{M}$ on both structures in parallel and accept if the invariants are equal. Here we are faced with a similar issue as in Theorem 5.21: We cannot simulate the computation on one component in the disjoint union if the components are not distinguished because we then possibly cannot witness orbits if the components are isomorphic. In the DeepWL+WSC setting, a complete invariant is a function $f\colon \mathcal{K} \to \{0,1\}^*$. We first execute $\mathtt{scc}(E)$ for the relation of plain edges in $\mathfrak{A} \uplus \mathfrak{B}$. This way, we obtain two vertices $u_{\mathfrak{A}}$ and $v_{\mathfrak{B}}$ related to all $\mathfrak{A}$-atoms respectively $\mathfrak{B}$-atoms in a vertex class C (similar to the proof of Lemma 5.59). Now, in $(\mathfrak{A} \uplus \mathfrak{B}, u_{\mathfrak{H}})$ for $\mathfrak{H} \in \{\mathfrak{A}, \mathfrak{B}\}$, the components are distinguished and we can execute $\mathcal{M}$ to compute $f(\mathfrak{H})$ by ignoring the other component. Using Lemma 5.44, we execute $\mathtt{refine}(C, f)$. If this results into two singleton vertex classes, then $f(\mathfrak{A}) \neq f(\mathfrak{B})$ and thus $\mathfrak{A} \not\cong \mathfrak{B}$. Otherwise, $\mathfrak{A} \cong \mathfrak{B}$.

To prove that Condition 1 implies Condition 2, let $\mathcal{M}$ be a DeepWL+WSC-algorithm deciding isomorphism. By Corollary 5.58, we can assume that $\mathcal{M}$ is normalized and, by Lemma 5.59, we can assume that if $\mathsf{run}(\mathcal{M}, \mathfrak{A}) = \mathsf{run}(\mathcal{M}, \mathfrak{B})$, then

$$\mathsf{run}(\mathcal{M}, \mathfrak{A}) \in \{\, \mathsf{run}(\mathcal{M}, \mathfrak{A}[\mathsf{V}_1(\mathfrak{A})] \uplus \mathfrak{B}[\mathsf{V}_i(\mathfrak{B})]) \mid i \in [2] \,\}$$

for all normalized HF-structures $\mathfrak{A}$ and $\mathfrak{B}$. We show that

$$f(\mathfrak{A}) := \mathsf{run}(\mathcal{M}, \mathfrak{A} \uplus \mathfrak{A})$$

for every $\mathfrak{A} \in \mathcal{K}$ is a complete invariant for $\mathcal{K}$. Clearly, if $\mathfrak{A} \cong \mathfrak{B}$, then $f(\mathfrak{A}) = f(\mathfrak{B})$ (runs are isomorphism-invariant). For the other direction, assume that

$$f(\mathfrak{A}) = \mathsf{run}(\mathcal{M}, \mathfrak{A} \uplus \mathfrak{A}) = \mathsf{run}(\mathcal{M}, \mathfrak{B} \uplus \mathfrak{B}) = f(\mathfrak{B}).$$

Because $\mathcal{M}$ decides isomorphism, $\mathcal{M}$ accepts $\mathfrak{A} \uplus \mathfrak{A}$ and $\mathfrak{B} \uplus \mathfrak{B}$. By Lemma 5.59, we have

$$\mathsf{run}(\mathcal{M}, \mathfrak{A} \uplus \mathfrak{A}) \in \{\mathsf{run}(\mathcal{M}, \mathfrak{A} \uplus \mathfrak{B})\},$$

i.e., $\mathsf{run}(\mathcal{M}, \mathfrak{A} \uplus \mathfrak{A}) = \mathsf{run}(\mathcal{M}, \mathfrak{A} \uplus \mathfrak{B})$. Thus, $\mathcal{M}$ also accepts $\mathfrak{A} \uplus \mathfrak{B}$ and we finally conclude $\mathfrak{A} \cong \mathfrak{B}$ because $\mathcal{M}$ decides isomorphism. □

5.3.5 From DeepWL+WSC to CPT+WSC

To finally prove Theorem 5.20, it remains to show that CPT+WSC simulates polynomial-time DeepWL+WSC-algorithms.

Lemma 5.61. *If a function f is computable (or a boolean query Q is decidable) by a polynomial-time DeepWL+WSC-algorithm, then f (or Q, respectively) is* CPT+WSC-*definable.*

Proof. We follow the same strategy as in Lemma 17 of [59]: Polynomial time DeepWL-algorithms can be simulated in CPT because CPT can execute the two-dimensional Weisfeiler-Leman algorithm to compute the needed coherent configurations and their algebraic sketches and because `addPair`- and `scc`-operations (the only ones modifying the vertex set of the cloud) can easily be simulated by set operations. So, we can maintain a set representing the vertex set of the structure in the cloud and further sets for the relations and colors in CPT. To extend this proof to DeepWL+WSC, we again proceed by induction on the nesting depth of the algorithm. We encode an HF-structure $\mathfrak{A} = (A, A_{HF}, E_1^{\mathfrak{A}}, \dots, E_k^{\mathfrak{A}})$ as an $\mathsf{HF}(A)$-set (using Kuratowski encoding for tuples). We then evaluate CPT+WSC-formulas or terms over the structure with atoms A and no relations. We denote this structure with $\mathfrak{A}_0$. The CPT+WSC-formulas and terms will have a free variable x, to which the $\mathfrak{A}$-encoding $\mathsf{HF}(A)$-set is passed.

We show that, for every polynomial-time DeepWL+WSC-algorithm $\mathcal{M}$, there is a CPT+WSC-term $s(x)$ such that, for every HF-structure $\mathfrak{A}$, the term $s(x)$ defines the final configuration of the output machine on input $\mathfrak{A}$ (using some appropriate encoding of 0/1-strings in $\mathsf{HF}(\emptyset)$-sets) if $\mathcal{M}$ does not fail on input $\mathfrak{A}$. To do so, assume $\mathcal{M} = (M^{\text{out}}, M^{\text{wit}}, \mathcal{M}_1, \dots, \mathcal{M}_\ell)$ is a DeepWL+WSC-algorithm and let by the induction hypothesis $s_1(x), \dots, s_\ell(x)$ be CPT+WSC-terms such that $s_i(x)$ defines the final configuration of the output machine of $\mathcal{M}_i$ for every $i \in [\ell]$. Let, for every $i \in [\ell]$, $\Phi_i(x)$ be a CPT+WSC-formula which defines whether the configuration defined by $s_i(x)$ is accepting, i.e., the head on the work-tape points to a 1. The Turing machine of M^{out} gets simulated using a WSC-fixed-point operator, which uses a variable y to maintain the structure in the cloud and the configuration of M^{out}. As for DeepWL, `addPair`- and `scc`-operations are simulated using set constructions on the structure in the cloud. The algebraic sketch is defined using the two-dimensional Weisfeiler-Leman algorithm.

A `refine`(E_j, i)-operation is simulated as follows: Let $\mathfrak{A} = (A, A_{HF}, E_1^{\mathfrak{A}}, \dots, E_k^{\mathfrak{A}})$ be the current HF-structure in the cloud as maintained by the WSC-fixed-point operator. First assume that E_j is directed. We then obtain with the term

$$\Big\{\, z \;\Big|\; z \in t_{2+j}(y), \Phi_i\Big(r(y, \{z_1\}, \{z_2\})\Big) \,\Big\}$$

the output of the `refine`-operation, where

- $t_{2+j}(y)$ defines the $(2+j)$-th entry in the tuple encoding $\mathfrak{A}$, i.e., the set $E_j^{\mathfrak{A}}$,
- z_i defines the i-th entry of the pair z, and
- $r(y, \{z_1\}, \{z_2\})$ extends the structure in y by the two new relations $\{z_1\}$ and $\{z_2\}$.

Here, we individualize z_1 and z_2 by putting them into new singleton relations. In the case that E is undirected, we proceed similarly but only create one new relation $\{z_1, z_2\}$. So we can simulate the machine M^{out} until it makes a `choice`- operation. We now use the variable

- x for the input structure,
- y for the pair of the current structure $\mathfrak{A}$ in the cloud and the current configuration c of M^{out} (as before), and
- z for the chosen element for the last `choice`-operation (or $\emptyset$ if the machine has to be started).

Let $s_{\text{step}}(x, y, z)$ be a CPT+WSC-term which simulates M^{out} in configuration c and $\mathfrak{A}$ in the cloud until M^{out} executes the next `choice`-operation or halts. The term $s_{\text{step}}(x, y, z)$ outputs the pair of the obtained configuration and the obtained structure in the cloud. If M^{out} has to be started, i.e., y is assigned to $\emptyset$, then the machine is started in the initial configuration on the structure passed to x. Second, let $s_{\text{choice}}(y)$ be a CPT+WSC-term which defines the choice-set of the next `choice`-operation to be made (or the empty set if the machine halted or was started). Third, let $s_{\text{wit}}(y, y')$ be the CPT+WSC-term which simulates the witnessing machine M^{wit} on the labeled union of the structures passed to y and y' and defines the set of automorphisms outputted by M^{wit} (or more precisely on the structures contained in the pairs passed to y and y'). Lastly, let $s_{\text{out}}(y)$ be a CPT+WSC-term extracting the configuration of the Turing machine passed to y, which is encoded as $\mathsf{HF}(\emptyset)$-set. We use the WSC-fixed-point operator defining $\mathsf{HF}(\emptyset)$-sets from Lemma 5.8 to define the final configuration of M^{out} when $\mathcal{M}$ is executed on the structure passed to x via

$$\Phi(x) := \mathsf{WSC}^* yz.\ \Big(s_{\text{step}}(x, y, z), s_{\text{choice}}(y), s_{\text{wit}}(y, z), s_{\text{out}}(y)\Big),$$

where we have to replace y' with z in $s_{\text{wit}}(y, y')$ to satisfy the formal requirements of the WSC-fixed-point operator. Because the elements of the structure in the cloud are obtained by the same HF-sets as by the DeepWL+WSC-algorithm, all choices are witnessed successfully in the formula if they are witnessed in the algorithm.

For the case of a DeepWL+WSC-computable function f, we extract the content of the work-tape from the set defined by Φ. For a DeepWL+WSC-decidable query, we check whether the defined configuration is accepting. □

Corollary 5.62. *A function f is computable (or a boolean query Q is decidable) by a polynomial-time DeepWL+WSC-algorithm if and only if f (or Q, respectively) are definable in* CPT+WSC.

Proof. The claim follows from Lemmas 5.33 and 5.61. □

We finally prove Theorem 5.20 and show that a CPT+WSC-definable isomorphism test implies a CPT+WSC-definable complete invariant.

Proof of Theorem 5.20. Let $\mathcal{K}$ be a class of binary τ-structures and let Φ be a CPT+WSC-formula defining isomorphism of $\mathcal{K}$. Then there is a polynomial-time DeepWL+WSC-algorithm deciding isomorphism of $\mathcal{K}$ by Lemma 5.33. By Theorem 5.60, there is a complete invariant computable by a polynomial-time DeepWL+WSC-algorithm. Finally, it follows from Lemma 5.61 that this complete invariant is CPT+WSC-definable. □

We note that the translation from polynomial-time DeepWL+WSC-algorithms into CPT-formulas cannot be effective because the polynomial bounding the running time of a DeepWL+WSC-algorithm is not given explicitly, but CPT-formulas have to provide them explicitly. However, this could be achieved by equipping DeepWL+WSC-algorithms with explicit polynomial bounds.

5.4 Discussion

We extended CPT with a witnessed symmetric choice operator and obtained the logic CPT+WSC. We proved that defining isomorphism in CPT+WSC is equivalent to defining canonization. A crucial point was to extend the DeepWL computation model to show that a CPT+WSC-definable isomorphism test yields a CPT+WSC-definable complete-invariant. Thereby, CPT+WSC can be viewed as a simplification step in the quest for a logic capturing PTIME as now only isomorphism needs to be defined to be able to apply the Immerman-Vardi Theorem.

To turn a complete invariant into a canonization within CPT+WSC, we used the canonization algorithm of Gurevich. To implement it in CPT+WSC, we have to extend it to provide witnessing automorphisms. For this to work, we needed to give to the witnessing terms the defined fixed-points as input. This is different in other extensions of first order logic with symmetric choice [31, 42]. We actually have to require that choice-sets are orbits respecting all previous intermediate steps in the fixed-point computation. It appears that this is only relevant if a formula actively forgets previous choices. But how could forgetting these be beneficial? In any case, we are not sure whether the modification changes the expressiveness of the logic.

Another question is the relation of CPT+WSC to other logics. Is CPT+WSC more expressive than CPT? We will see in Section 6.2.3 that CPT+WSC defines the CFI query for a class of base graphs for which it is not known that the CFI query is CPT-definable. Do nested WSC-fixed-point operators increase the expressiveness of CPT+WSC? In [31], it is proved that for fixed-point logic extended with (unwitnessed) symmetric choice, that nesting fixed-point operators with choice increases expressiveness. We should remark that any positive answer to our questions separates CPT from Ptime and hence all questions might be difficult to answer.

Finally, extending DeepWL with witnessed symmetric choice turned out to be extremely tedious. While proofs for DeepWL without choice are already complicated [59], for our extensions the proofs got even more involved. We would like to see more elegant techniques to prove Theorem 5.20 for CPT+WSC (or even for CPT).

Chapter 6

Fixed-Point Logic with Counting, Witnessed Symmetric Choice, and Interpretations

In the previous chapter, we have seen how witnessed symmetric choice in CPT can be used to show that a CPT+WSC-definable isomorphism test implies a CPT+WSC-definable canonization and hence that CPT+WSC captures PTime (for a class of structures). However, the question whether witnessed symmetric choice strictly increases the expressiveness of CPT remained open. In this section, we want to understand the expressiveness of witnessed symmetric choice and its combination with interpretations. However, a positive answer for CPT requires separating CPT from PTime, which has been open for a long time. So we will consider IFPC as base logic in the chapter.

(Witnessed) symmetric choice has the drawback that it can only choose from orbits of the input structure. If orbits cannot be defined or witnessed in the logic, then no choices can be made. Even if it is possible to reduce the input structure to a simpler one with definable orbits using an interpretation, it is not clear how choices can be made in the interpreted structure because the choice-sets have to be orbits of the input structure. To overcome this problem, Gire and Hoang [42] proposed an interpretation operator, which evaluates a formula in the interpreted structure. In this way, choices are made from the interpreted structure and not from the input structure. In particular, every logic equipped with the interpretation operator is, by construction, closed under interpretations. But for many logics, such an operator does not increase the expressiveness, e.g., for IFP, IFPC, and CPT. For IFP, the extension of IFP with witnessed symmetric choice (IFP+WSC) is less expressive than the additional extension with the interpretation operator (IFP+WSC+I). The interpretation operator combined with witnessed symmetric choice simulates counting. However, it was indicated that witnessed symmetric choice alone fails to simulate counting [42].

We are interested in the relation between witnessed symmetric choice and the interpretation operator not specifically for IFP but more generally. Most of the existing results by Gire and Hoang [42] and Dawar and Richerby [31], who consider the extension of IFP with non-witnessed symmetric choice (IFP+SC), show that (witnessed) symmetric choice or the interpretation operator can be used to define restricted forms of counting that cannot be defined without them. Essentially, the only non-counting based result shows that IFP+WSC defines the CFI query for order base graphs [42]. However, counting is not the actual reason for using witnessed symmetric choice. Counting can be achieved more naturally in IFPC. Thus, it is unknown whether the interpretation operator increases expressiveness of IFPC+WSC.

Overall, IFPC is a natural base logic for studying the interplay of witnessed symmetric choice and the interpretation operator. In IFPC, separation results based on counting are not applicable. But in contrast to CPT, there are known IFPC-undefinable PTIME properties, namely the CFI query from Section 2.8, which can be used analyze the expressiveness of extensions of IFPC. In this chapter, we show that the interpretation operator increases expressiveness for IFPC+WSC, too. Hence, IFPC+WSC+I is strictly more expressive than IFPC+WSC. In particular, we show that IFPC+WSC is not even closed under one-dimensional equivalence-free FO-interpretations. We thereby answer the question of Dawar and Richerby [31] whether IFP+SC is closed under interpretations.

Next, we show that, with respect to canonization in IFPC+WSC+I, a class of CFI graphs is not more difficult than the corresponding class of base graphs: If IFPC+WSC+I canonizes the class of base graphs, then IFPC+WSC+I also canonizes the class of CFI graphs. This result is special in the sense that it does not hold for many other logics [20, 28, 45, 84] (see also Chapter 7). We exploit this construction to establish a first step towards proving an operator nesting hierarchy for IFPC+WSC+I by essentially iteratively applying the CFI construction.

Related Work. The (witnessed) symmetric choice operator is combined with a fixed-point operator in [31, 42]. There are more approaches to integrate choices in first-order logic: Choice operators independent of a fixed-point operator were studied. Hilbert's ε-operator resolves choices using a global choice function, that is, when making twice a choice from the same choice-set, the same element is chosen. An ε-invariant formula is a formula invariant under all possible choice-functions. Blass and Gurevich [16] showed that for the class of all (not necessarily finite) structures, every ε-invariant FO(ε)-formula is equivalent to an FO-formula. For the restriction to finite structures, Otto [99] proved that this is not the case, i.e., FO(ε) is more expressive than FO on finite structures. Blass and Gurevich [16] also proposed a δ-operator that does not use a choice function and hence two choices from the same choice-set may result in two different elements. These logics are no candidates to capture PTIME because of nondeterminism, undecidable syntax, or too high complexity. A similar statement holds for the nondeterministic version of the fixed-point operator with choice, where choices can be made from arbitrary choice-sets and not only from orbits [32]. For a more detailed overview, we refer to Richerby's PhD thesis [105].

Symmetric choice becomes useless if a structure only has singleton orbits, or equivalently its only automorphism is the trivial one, because in this case there is nothing to choose. Such structures are called asymmetric. Multipedes [64] are a class of asymmetric structures that, for every fixed number of variables k, are not characterized up to isomorphism in k-variable counting logic. Asymmetry turns multipedes to hard examples for graph isomorphism algorithms in the individualization-refinement framework [4, 96]. The size of a multipede not identifiable in k-variable counting logic is large with respect to k. There exists also a class of asymmetric graphs [30] for which the number of variables needed for identification is linear. Both classes are based on the CFI construction.

There is another remarkable but not directly connected correspondence to lengths of resolution proofs. Resolution proofs for non-isomorphism of CFI-graphs have exponential size [112]. When adding a global symmetry rule (SRC-I), which exploits automorphisms of the formula (so akin to symmetric choice), the length becomes polynomial [108]. For asymmetric multipedes the length in the SRC-I system is still exponential [113]. But when considering the local symmetry rule (SRC-II), which exploits local automorphisms (so somewhat akin to symmetric choice after restricting to a substructure with an interpretation), the length becomes polynomial again [108].

Overview of this Chapter. We define the logics IFPC+WSC and IFPC+WSC+I in Section 6.1 by extending IFPC with a fixed-point operator featuring witnessed symmetric choice and further with the interpretation operator. We reuse the WSC^*-operator from Section 5.1.1, which was based on the techniques of [31], to formally define the semantics of these logics. Following the approach from Chapter 5, the formula that defines the witnessing automorphisms has access to the defined fixed-point. We show in Section 6.2 that, if a class of base graphs $\mathcal{K}$ can be canonized in IFPC+WSC+I, then the class of CFI graphs over $\mathcal{K}$ can be canonized in IFPC+WSC+I, too. Similar to Chapter 5, we again need to consider the closure under individualization. In this case, Gurevich's canonization algorithm [62] can be implemented in IFPC+WSC, too.

To prove that a definable canonization for a class of base graphs implies a definable canonization for the corresponding class of CFI graphs, we use the interpretation operator to define, given a CFI graph, the corresponding base graph and its orbits. We show that, if IFPC+WSC+I distinguishes orbits of the base graphs, then IFPC+WSC+I distinguishes also orbits of the CFI graphs and thus canonizes the CFI graphs using Gurevich's algorithm. In the CFI-graph-canonizing formula, the nesting depth of WSC-fixed-point operators increases by one (for Gurevich's algorithm) compared to the formula that distinguishes orbits of the base graphs. In the same sense, the nesting depth of interpretation operators increases by one (to define the orbits of the base graph), too.

In Section 6.3, we show that this increase of the nesting depth is necessary by constructing double CFI graphs. We start with a class of CFI graphs $\mathsf{CFI}(\mathcal{K}')$ for which a canonization is WSCI(IFPC)-definable, that is, definable in the fragment of IFPC+WSC+I in which at most one interpretation operator can be nested inside one WSC-fixed-point operator. We create a new class of base graphs $\mathcal{K}$ from the $\mathsf{CFI}(\mathcal{K}')$-graphs. Applying the CFI construction once more, $\mathsf{CFI}(\mathcal{K})$ is canonized in WSCI(WSCI(IFPC)) (that is, two operators can be nested) but not in WSCI(IFPC): To define orbits of $\mathsf{CFI}(\mathcal{K})$, we have to define orbits of the base graph, for which we need to distinguish the CFI graphs in $\mathsf{CFI}(\mathcal{K}')$.

Last, we prove IFPC+WSC $<$ IFPC+WSC+I in Section 6.4 and thereby separate IFPC+WSC from PTIME. We construct a class of asymmetric structures for which isomorphism is not IFPC-definable. Because asymmetric structures have only singleton orbits, witnessed symmetric choice is not beneficial, thus IFPC+WSC $=$ IFPC on these structures, and isomorphism is not IFPC+WSC-definable. These structures combine CFI graphs and the multipedes already mentioned [64], which are asymmetric and for which IFPC fails to distinguish orbits. An interpretation removes the multipedes and reduces the isomorphism problem to the one of CFI graphs. Thus, isomorphism of this class of structures is IFPC+WSC+I-definable. We end the chapter with a discussion in Section 6.5.

6.1 IFPC with Witnessed Symmetric Choice

We extend IFPC with an **inflationary fixed-point operator with witnessed symmetric choice**. Let τ be a relational signature and R, R^*, and S be new relation symbols not used in τ of arities $\mathrm{ar}(R) = \mathrm{ar}(R^*)$ and $\mathrm{ar}(S)$. The letter p is used for an element parameter in this section. We define the WSC-fixed-point operator with parameters $\bar{p}\bar{\nu}$. If

- $\Phi_{\text{step}}(\bar{p}\bar{x}\bar{\nu})$ is an IFPC+WSC$[\tau, R, S]$-formula such that $|\bar{x}| = \mathrm{ar}(R)$,
- $\Phi_{\text{choice}}(\bar{p}\bar{y}\bar{\nu})$ is an IFPC+WSC$[\tau, R]$-formula such that $|\bar{y}| = \mathrm{ar}(S)$,
- $\Phi_{\text{wit}}(\bar{p}\bar{y}\bar{y}'z_1z_2\bar{\nu})$ is an IFPC+WSC$[\tau, R, R^*]$-formula where $|\bar{y}| = |\bar{y}'| = \mathrm{ar}(S)$, and
- $\Phi_{\text{out}}(\bar{p}\bar{\nu})$ is an IFPC+WSC$[\tau, R^*]$-formula,

then

$$\Phi(\bar{p}\bar{\nu}) = \mathsf{ifp\text{-}wsc}_{R,\bar{x};R^*;S,\bar{y},\bar{y}';z_1z_2}.\ \Big(\Phi_{\text{step}}(\bar{p}\bar{x}\bar{\nu}), \Phi_{\text{choice}}(\bar{p}\bar{y}\bar{\nu}), \Phi_{\text{wit}}(\bar{p}\bar{y}\bar{y}'z_1z_2\bar{\nu}), \Phi_{\text{out}}(\bar{p}\bar{\nu})\Big)$$

is an IFPC+WSC$[\tau]$-formula. The formulas Φ_{step}, Φ_{choice}, Φ_{wit}, and Φ_{out} are called **step formula**, **choice formula**, **witnessing formula**, and **output formula**, respectively. The free variables of Φ are the ones of Φ_{step} apart from $\bar{x}$, the ones of Φ_{choice} apart from $\bar{y}$, the ones of Φ_{wit} apart from $\bar{y}$, $\bar{y}'$, z_1 and z_2, and the ones of Φ_{out}. That is, $\bar{x}$ is bound in Φ_{step}, $\bar{y}$ is bound in Φ_{choice} and Φ_{wit}, and $\bar{y}'$, z_1, and z_2 are bound in Φ_{wit}. Note that only element variables are used for defining the fixed-point in the WSC-fixed-point operator. This suffices for our purpose in this thesis and increases readability. We expect that our arguments also work with numeric variables in the fixed-point. For better readability, we will now omit the free numeric variables $\bar{\nu}$ when defining the semantics of the WSC-fixed-point operator. Fixing numeric parameters does not change orbits or automorphisms.

Evaluation with Choices. Intuitively, Φ is evaluated as follows: Let $\mathfrak{A}$ be a τ-structure and $\bar{u} \in A^{|\bar{p}|}$ be a tuple of parameters. We inductively define a sequence of relations called **stages**

$$\emptyset =: R_1^{\mathfrak{A}}, \dots, R_\ell^{\mathfrak{A}} = R_{\ell+1}^{\mathfrak{A}} =: (R^*)^{\mathfrak{A}}$$

as follows. Given the relation $R_i^{\mathfrak{A}}$, the choice formula defines the choice-set $T_{i+1}^{\mathfrak{A}}$ of the tuples $\bar{v}$ satisfying Φ_{choice}, i.e.,

$$T_{i+1}^{\mathfrak{A}} := \Big\{ \bar{v} \;\Big|\; \bar{u}\bar{v} \in \Phi_{\text{choice}}^{\mathfrak{A}, R_i^{\mathfrak{A}}} \Big\}.$$

We pick an arbitrary tuple $\bar{v} \in T_{i+1}^{\mathfrak{A}}$ and set $S_{i+1}^{\mathfrak{A}} := \{\bar{v}\}$ or $S_{i+1}^{\mathfrak{A}} := \emptyset$ if no such $\bar{v}$ exists. The step formula is used on the structure $(\mathfrak{A}, R_i^{\mathfrak{A}}, S_{i+1}^{\mathfrak{A}})$ to define the next stage in the fixed-point iteration:

$$R_{i+1}^{\mathfrak{A}} := R_i^{\mathfrak{A}} \cup \Big\{ \bar{w} \;\Big|\; \bar{u}\bar{w} \in \Phi_{\text{step}}^{(\mathfrak{A}, R_i^{\mathfrak{A}}, S_{i+1}^{\mathfrak{A}})} \Big\}.$$

We proceed in this way until a fixed-point $(R^*)^{\mathfrak{A}}$ is reached. This fixed-point and the sequence of stages is in general not isomorphism-invariant, i.e., not invariant under applying automorphisms of $(\mathfrak{A}, \bar{u})$.

We ensure that Φ is still isomorphism-invariant similar to the case of CPT in Section 5.1: First, we only allow choices from orbits, which the witnessing formula has to certify. Recall from Section 5.1.1 that a set $N \subseteq \mathsf{Aut}((\mathfrak{A}, \bar{u}))$ witnesses a relation $R \subseteq A^k$ as $(\mathfrak{A}, \bar{u})$-orbit if, for all $\bar{v}, \bar{v}' \in R$, there is an automorphism $\varphi \in N$ satisfying $\bar{v} = \varphi(\bar{v}')$. Because we need the notion of witnessing orbits only for isomorphism-invariant sets, we do not need to check whether R is a proper subset of an orbit. We require that Φ_{wit} defines a set of automorphisms. Intuitively, for $\bar{v}, \bar{v}' \in T_{i+1}^{\mathfrak{A}}$, a map $\varphi_{\bar{v},\bar{v}'}$ is defined via

$$w \mapsto w' \text{ whenever } \bar{u}\bar{v}\bar{v}'ww' \in \Phi_{\text{wit}}^{\left(\mathfrak{A}, R_i^{\mathfrak{A}}, (R^*)^{\mathfrak{A}}\right)}.$$

The set of all these maps for all $\bar{v}, \bar{v}' \in T_{i+1}^{\mathfrak{A}}$ has to witness $T_{i+1}^{\mathfrak{A}}$ as $(\mathfrak{A}, \bar{u}, R_1^{\mathfrak{A}}, \dots, R_i^{\mathfrak{A}})$-orbit. Note here, that, similar to CPT+WSC, the witnessing formula always has access to the fixed-point. Actually, we do not require that $\varphi_{\bar{v},\bar{v}'}$ maps $\bar{v}$ to $\bar{v}'$ but only the set of all $\varphi_{\bar{v},\bar{v}'}$ has to witness the orbit. If some choice is not witnessed, Φ is not satisfied by $\bar{u}$. Otherwise, the output formula is evaluated on the defined fixed-point:

$$\Phi^{\mathfrak{A}} := \Phi_{\text{out}}^{(\mathfrak{A}, (R^*)^{\mathfrak{A}})}.$$

Because all choices are witnessed, all possible fixed-points (for different choices) are related by an automorphism of $(\mathfrak{A}, \bar{u})$ and thus either all or none satisfy the output formula.

An Example. We adapt the example from Chapter 5 from CPT+WSC to IFP+WSC and show that the class of threshold graphs (i.e, graphs which can be reduced to the empty graph by iteratively deleting universal or isolated vertices) is IFPC+WSC-definable (it is actually IFP-definable, but illustrates the WSC-fixed-point operator). The set of all isolated or universal vertices (note that there cannot be an isolated and a universal vertex at the same time) of a graph forms a 1-orbit. We choose one vertex of that orbit, collect it in a unary relation R, and repeat. If all vertices are contained in the obtained fixed-point R^*, then the graph is a threshold graph. The choice formula Φ_{choice} defines the set of all isolated or universal vertices not in R. The step formula Φ_{step} adds the chosen vertex, which is the only vertex in the relation S, to R. The output formula Φ_{out} checks whether R^* contains all vertices and so defines whether it was possible to delete all vertices:

$$\begin{aligned}
\Phi_{\text{choice}}(y) &:= \neg R(y) \wedge \Big(\big(\forall z.\ \neg R(y) \Rightarrow E(y,z)\big) \vee \big(\forall z.\ \neg R(y) \Rightarrow \neg E(y,z)\big)\Big),\\
\Phi_{\text{step}}(x) &:= R(x) \vee S(x),\\
\Phi_{\text{out}} &:= \forall x.\ R^*(x).
\end{aligned}$$

Witnessing orbits is easy. To show that two isolated (or universal, respectively) vertices y and y' are related by an automorphism, it suffices to define their transposition as follows:

$$\Phi_{\text{wit}}(y, y', z_1, z_2) := (z_1 = y \wedge z_2 = y') \vee (z_2 = y \wedge z_1 = y') \vee (y \neq z_1 = z_2 \neq y').$$

The formula $\mathsf{ifp\text{-}wsc}_{R,x;R^*;S,y,y';z_1z_2}.\,(\Phi_{\text{step}}, \Phi_{\text{choice}}, \Phi_{\text{wit}}, \Phi_{\text{out}})$ defines the class of threshold graphs.

Formal Semantics. To define the semantics of the WSC-fixed-point operator formally, we use the WSC^*-operator defined Section 5.1.1. Recall that the WSC^*-operator captures fixed-point iterations with choices from orbits for arbitrary isomorphism-invariant functions. Let $\mathfrak{A}$ be a τ-structure and $\bar{u} \in A^k$. The WSC^*-operator defines, given isomorphism-invariant functions $f^{\mathfrak{A},\bar{u}}_{\text{step}}, f^{\mathfrak{A},\bar{u}}_{\text{wit}} \colon \mathsf{HF}(A) \times \mathsf{HF}(A) \to \mathsf{HF}(A)$ and $f^{\mathfrak{A},\bar{u}}_{\text{choice}} \colon \mathsf{HF}(A) \to \mathsf{HF}(A)$, the set $W = \mathsf{WSC}^*(f^{\mathfrak{A},\bar{u}}_{\text{step}}, f^{\mathfrak{A},\bar{u}}_{\text{wit}}, f^{\mathfrak{A},\bar{u}}_{\text{choice}})$ of all $\mathsf{HF}(A)$-sets obtained in the following way. Starting with $b_0 := \emptyset$, define a sequence of sets as follows: Given b_i, define the choice-set $c_i := f^{\mathfrak{A},\bar{u}}_{\text{choice}}(b_i)$, pick an arbitrary $d_i \in c_i$ (or $d = \emptyset$ if $c_i = \emptyset$), and set $b_{i+1} := f^{\mathfrak{A},\bar{u}}_{\text{step}}(b_i, d_i)$. Let $b^* := b_\ell$ for the smallest ℓ satisfying $b_\ell = b_{\ell+1}$ (if it exists, which in our case of inflationary fixed-points is always the case). Then we include b^* in W if $f^{\mathfrak{A},\bar{u}}_{\text{wit}}(b_i, b^*)$ defines a set of automorphisms witnessing c_i as $(\mathfrak{A}, \bar{u}, b_1, \dots, b_i)$-orbit for every $i \in [\ell]$. Of course, b^* is not unique and depends on the choices of the d_i and thus W is not necessarily a singleton set. We have seen in Corollary 5.4 that $f^{\mathfrak{A},\bar{u}}_{\text{wit}}$ witnesses the choices of all d_i for either every possible b^* obtained in the former way or for none of them. In particular, W is an $(\mathfrak{A}, \bar{u})$-orbit (Corollary 5.5). We now define the semantics of the WSC+fixed-point operator

$$\Phi(\bar{p}) = \mathsf{ifp\text{-}wsc}_{R,\bar{x};R^*;S,\bar{y},\bar{y}';z_1z_2}.\, \Big(\Phi_{\text{step}}(\bar{p}\bar{x}), \Phi_{\text{choice}}(\bar{p}\bar{y}), \Phi_{\text{wit}}(\bar{p}\bar{y}\bar{y}'z_1z_2), \Phi_{\text{out}}(\bar{p})\Big)$$

using tuples (implicitly encoded in $\mathsf{HF}(A)$-sets). Let $\bar{u} \in A^{|\bar{p}|}$. We set

$$\begin{aligned} f^{\mathfrak{A},\bar{u}}_{\text{step}}(R^{\mathfrak{A}}, S^{\mathfrak{A}}) &:= R^{\mathfrak{A}} \cup \left\{ \bar{w} \;\middle|\; \bar{u}\bar{w} \in \Phi^{(\mathfrak{A},R^{\mathfrak{A}},S^{\mathfrak{A}})}_{\text{step}} \right\}, \\ f^{\mathfrak{A},\bar{u}}_{\text{choice}}(R^{\mathfrak{A}}) &:= \left\{ \bar{v} \;\middle|\; \bar{u}\bar{v} \in \Phi^{(\mathfrak{A},R^{\mathfrak{A}})}_{\text{choice}} \right\}, \\ f^{\mathfrak{A},\bar{u}}_{\text{wit}}(R^{\mathfrak{A}}, (R^*)^{\mathfrak{A}}) &:= \left\{ \varphi_{\bar{v}\bar{v}'} \;\middle|\; \bar{v}, \bar{v}' \in f^{\mathfrak{A},\bar{u}}_{\text{choice}}(R^{\mathfrak{A}}) \right\} \text{ where} \\ & \qquad \varphi_{\bar{v}\bar{v}'} = \left\{ (w, w') \in A^2 \;\middle|\; \bar{u}\bar{v}\bar{v}'ww' \in \Phi^{\left(\mathfrak{A},R^{\mathfrak{A}},(R^*)^{\mathfrak{A}}\right)}_{\text{wit}} \right\}. \end{aligned}$$

The function $f^{\mathfrak{A},\bar{u}}_{\text{step}}(R^{\mathfrak{A}}, S^{\mathfrak{A}})$ evaluates the step formula Φ_{step} and adds its output to $R^{\mathfrak{A}}$, which defines the inflationary fixed-point. The function $f^{\mathfrak{A},\bar{u}}_{\text{choice}}(R^{\mathfrak{A}})$ defines the choice-set by evaluating the choice formula Φ_{choice}. Finally, the function $f^{\mathfrak{A},\bar{u}}_{\text{wit}}(R^{\mathfrak{A}}, (R^*)^{\mathfrak{A}})$ defines a set (possibly) of automorphisms by evaluating the witnessing formula Φ_{wit} for all tuples in the current relation. Now set $W^{(\mathfrak{A},\bar{u})}_{\Phi} := \mathsf{WSC}^*(f^{\mathfrak{A},\bar{u}}_{\text{step}}, f^{\mathfrak{A},\bar{u}}_{\text{choice}}, f^{\mathfrak{A},\bar{u}}_{\text{wit}})$.

We define the semantics of the WSC-fixed-point operator Φ as follows. Let $\mathfrak{A}$ be a τ-structure and let $\operatorname{sig}(\Phi) \subseteq \tau$ be the relation symbols mentioned in Φ (excluding those bound by fixed-point operators or WSC-fixed-point operators). We define

$$\Phi^{\mathfrak{A}} := \left\{ \bar{u} \;\middle|\; \bar{u} \in \Phi^{(\mathfrak{A},(R^*)^{\mathfrak{A}})}_{\text{out}} \text{ for some } (R^*)^{\mathfrak{A}} \in W^{(\mathfrak{A}\upharpoonright\operatorname{sig}(\Phi),\bar{u})}_{\Phi} \right\}.$$

Note that if $W^{(\mathfrak{A}\upharpoonright\operatorname{sig}(\Phi),\bar{u})}_{\Phi} = \emptyset$, that is, not all choices could be witnessed, then we have $\bar{u} \notin \Phi^{\mathfrak{A}}$. Also note that because $W^{(\mathfrak{A}\upharpoonright\operatorname{sig}(\Phi),\bar{u})}_{\Phi}$ is an $(\mathfrak{A} \upharpoonright \operatorname{sig}(\Phi), \bar{u})$-orbit, $\bar{u} \in \Phi^{(\mathfrak{A}\upharpoonright\operatorname{sig}(\Phi),(R^*)^{\mathfrak{A}})}_{\text{out}}$ holds for either every $(R^*)^{\mathfrak{A}} \in W^{(\mathfrak{A}\upharpoonright\operatorname{sig}(\Phi),\bar{u})}$ or for no $(R^*)^{\mathfrak{A}} \in W^{(\mathfrak{A}\upharpoonright\operatorname{sig}(\Phi),\bar{u})}$. Finally, note that the WSC-fixed-point operator for IFPC+WSC is evaluated on the reduct $\mathfrak{A} \upharpoonright \operatorname{sig}(\Phi)$. In this way, adding more relations to $\mathfrak{A}$ which are not mentioned in the formula do not change the orbit structure of $\mathfrak{A} \upharpoonright \operatorname{sig}(\Phi)$ and so does not change whether Φ is satisfied or not. This is a desirable property of a logic [37]. This **reduct semantics** of a choice operator can also be found in [31]. Finally, we conclude that IFPC+WSC is isomorphism-invariant:

Lemma 6.1. *For every* IFPC+WSC$[\tau]$*-formula* Φ *and every* τ*-structure* $\mathfrak{A}$*, the set* $\Phi^{\mathfrak{A}}$ *is a union of* $(\mathfrak{A} \upharpoonright \mathrm{sig}(\Phi))$*-orbits.*

Proof. The proof is straightforward by induction on the formula using the fact that $W_\Phi^{(\mathfrak{A},\bar{u})}$ is an $(\mathfrak{A}, \bar{u})$-orbit by Corollary 5.5. □

6.1.1 Extension with an Operator for Logical Interpretations

We now extend IFPC+WSC with another operator using interpretations. First, every IFPC+WSC-formula is an IFPC+WSC+I-formula. If $\Theta(\bar{p}\bar{\nu})$ is an IFPC+WSC+I$[\tau,\sigma]$-interpretation with parameters $\bar{p}\bar{\nu}$ and Φ is an IFPC+WSC+I$[\sigma]$-sentence, then the **interpretation operator**

$$\Psi(\bar{p}\bar{\nu}) = \mathsf{I}(\Theta(\bar{p}\bar{\nu}); \Phi)$$

is an IFPC+WSC+I$[\tau]$-formula with free variables $\bar{p}\bar{\nu}$. The semantics is defined as follows:

$$\mathsf{I}(\Theta(\bar{p}\bar{\nu}); \Phi)^{\mathfrak{A}} := \left\{ \bar{u}\bar{n} \in A^{|\bar{p}|} \times \mathbb{N}^{|\bar{\nu}|} \;\middle|\; \Phi^{\Theta(\mathfrak{A},\bar{u}\bar{n})} \neq \emptyset \right\}.$$

Note that $\Phi^{\Theta(\mathfrak{A},\bar{u}\bar{n})} \neq \emptyset$ if and only if Φ is satisfied. The interpretation operator allows to evaluate a subformula in the image of an interpretation. Thus, by definition, IFPC+WSC+I is closed under interpretations. For IFPC, such an operator does not increase the expressive power because IFPC is already closed under IFPC-interpretations (see [98]). For IFPC+WSC, this is not clear: Because $\Theta(\mathfrak{A}, \bar{u}\bar{n})$ may have a different automorphism structure, Φ may exploit the WSC-fixed-point operator in a way which is not possible in $\mathfrak{A}$. Indeed, we will later see that IFPC+WSC is not even closed under FO-interpretations. We now study the properties of IFPC+WSC+I and its relation to IFPC+WSC. To do so, we first consider canonization of CFI graphs in IFPC+WSC+I.

6.2 Canonization of CFI Graphs in IFPC+WSC+I

In this section we show that with respect to canonization the CFI construction from Section 2.8 "loses its power" in IFPC+WSC+I in the sense that canonizing CFI graphs is not harder than canonizing the base graphs.

CFI Construction in this Chapter. In this chapter, we only work with the CFI construction using gadget and edge vertices (cf. Section 2.8). For readability, we refer with $\mathsf{CFI}(G, g)$ to $\mathsf{CFI}_{\mathrm{ge}}(G, g)$ in this chapter. In case we are only interested in the isomorphism type of a CFI graph, so in whether $\sum g = 0$ or not, we also write $\mathsf{CFI}(G, 0)$ for the even and $\mathsf{CFI}(G, 1)$ for the odd CFI graph.

6.2.1 Canonization in IFPC+WSC+I

Similar to Chapter 5, we work with a class of base graphs closed under individualization. Intuitively, this means that the class is closed under assigning some vertices unique new colors. Recall Definition 5.11 of individualizing a tuple of vertices by a binary relation $\trianglelefteq$ which defines a total order on the individualized vertices. Also, recall the definition of the closure under individualization $\mathcal{K}^{\trianglelefteq}$ of a class of τ-structures $\mathcal{K}$ (Definition 5.12) that considers all possible individualizations of all $\mathcal{K}$-structures. We first establish a reduced version of Theorem 5.21 for IFPC+WSC and IFPC+WSC+I. Note that we do not include definable isomorphism here.

Lemma 6.2. *Let L be one of the logics* IFPC+WSC *or* IFPC+WSC+I *and let $\mathcal{K}$ be a class of τ-structures closed under individualization. The following are equivalent:*

1. *L defines a canonization for $\mathcal{K}$ (recall Definition 2.3).*
2. *L distinguishes the k-orbits of $\mathcal{K}$ for every $k \in \mathbb{N}$ (recall Definition 2.5).*
3. *$\mathcal{K}$ is ready for individualization in L (recall Definition 5.15).*

Proof. The proof is similar to the one of Theorem 5.21. We show 1 $\Rightarrow$ 2 $\Rightarrow$ 3 $\Rightarrow$ 1. To show 1 $\Rightarrow$ 2, one orders two k-tuples according to the lexicographical order on the canons when individualizing these tuples (similar to Lemma 5.13, where we use the canons instead of a complete invariant). For 2 $\Rightarrow$ 3, one orders the 1-orbits and picks the minimal (according to that order) orbit disjoint with the individualized vertices. If such an orbit does not exist, the minimal singleton 1-orbit is picked. Finally, to show 3 $\Rightarrow$ 1, one defines a variant of Gurevich's canonization algorithm [62] using a WSC-fixed-point operator in the same way as in Lemma 5.17. It is easy to see there that once the orbit-distinguishing formula is an L-formula, then the algorithm can easily be expressed in L (essentially collecting individualized vertices in a relation). $\square$

So, by Lemma 6.2, it suffices to define a single orbit disjoint with the individualized vertices (if it exists) to obtain a definable canonization, which uses the WSC-fixed-point operator. This approach simplifies defining canonization using WSC-fixed-point operators. Their use is hidden in Gurevich's canonization algorithm: We do not need to define witnessing automorphisms explicitly.

6.2.2 Defining Orbits of CFI Graphs

We now show that once we define orbits of base graphs (more precisely, of the closure under individualization of the base graphs), we can define orbits of CFI graphs and hence canonize them. Here we need the interpretation operator to reduce a CFI graph to its base graph to define its orbits.

Lemma 6.3. *Assume $\mathcal{K}$ is a class of colored base graphs with minimal degree 3. If* IFPC+WSC+I *distinguishes 2-orbits of $\mathcal{K}^{\trianglelefteq}$, then $\mathsf{CFI}(\mathcal{K})^{\trianglelefteq}$ is ready for individualization in* IFPC+WSC+I.

Proof. Let $G = (V, E, \preceq) \in \mathcal{K}$, $g\colon E \to \mathbb{F}_2$, and $\mathfrak{A} = (\mathsf{CFI}(G, g), \trianglelefteq^{\mathfrak{A}})$, where $\trianglelefteq^{\mathfrak{A}}$ individualizes some vertices of $\mathsf{CFI}(G, g)$. We assume that only edge vertices are individualized. Instead of individualizing a gadget vertex with origin $\mathfrak{u}$, one can equivalently individualize an edge vertex with origin $(\mathfrak{u}, \mathfrak{v})$ for every $\mathfrak{v} \in N_G(\mathfrak{u})$ by Lemma 2.12. This step is IFPC-definable. We denote by

$$\mathsf{orig}(\trianglelefteq^{\mathfrak{A}}) := \left\{ \mathsf{orig}(u) \;\middle|\; u \in A \text{ is individualized by } \trianglelefteq^{\mathfrak{A}} \right\}$$

the set of (directed) origin base edges of the individualized edge vertices and with $G - \mathsf{orig}(\trianglelefteq^{\mathfrak{A}})$ the graph obtained from G by deleting the edges in $\mathsf{orig}(\trianglelefteq^{\mathfrak{A}})$ viewed as undirected edges. Note that individualizing an edge vertex implies individualizing a directed base edge, which is equivalent to individualizing two base vertices. In this way, we denote by $(G, \mathsf{orig}(\trianglelefteq^{\mathfrak{A}}))$ the graph G when individualizing base vertices such that exactly the base edges in $\mathsf{orig}(\trianglelefteq^{\mathfrak{A}})$ are implicitly individualized.

We analyze the cases in which there are nontrivial 1-orbits of $(\mathfrak{A}, \trianglelefteq^{\mathfrak{A}})$. If there is a 2-orbit of $(G, \mathsf{orig}(\trianglelefteq^{\mathfrak{A}}))$ containing base edges part of a cycle in $G - \mathsf{orig}(\trianglelefteq^{\mathfrak{A}})$, we obtain a nontrivial 1-orbit of $(\mathfrak{A}, \trianglelefteq^{\mathfrak{A}})$:

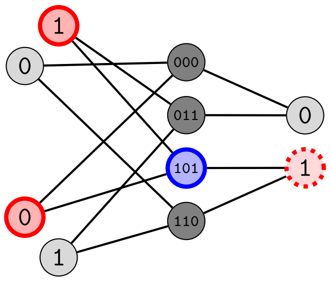
Claim 1. *Let O be a 2-orbit of $(G, \mathsf{orig}(\trianglelefteq^{\mathfrak{A}}))$. If every (directed) edge in O is part of a cycle in $G - \mathsf{orig}(\trianglelefteq^{\mathfrak{A}})$, then the set of edge-vertex-pairs $\{u \in A \mid \mathsf{orig}(u) \in O\}$ is a 1-orbit of $(\mathfrak{A}, \trianglelefteq^{\mathfrak{A}})$.*

Proof. Because O is a 2-orbit, either all or none of the directed base edges in O are part of a cycle in $G - \mathsf{orig}(\trianglelefteq^{\mathfrak{A}})$. By Lemma 2.11, the edge-vertex-pairs with origin in O are in the same 1-orbit. Using an automorphism of the base graph, the edge vertices with origin $(\mathfrak{u}, \mathfrak{v}) \in O$ can be mapped to the edge vertices with origin $(\mathfrak{u}', \mathfrak{v}') \in O$ for all $(\mathfrak{u}, \mathfrak{v}), (\mathfrak{u}', \mathfrak{v}') \in O$. That is, $\{u \in A \mid \mathsf{orig}(u) \in O\}$ is a subset of a 1-orbit of $(\mathfrak{A}, \trianglelefteq^{\mathfrak{A}})$. It cannot be a strict subset because then an edge vertex with origin in O has to be mapped to an edge vertex with origin not in O. This contradicts that O is an orbit. $\dashv$

In case that no such 2-orbit exists, there are possibly other nontrivial 1-orbits. They arise from automorphisms of the base graph. An **edge-vertex-pair-order** of a set of base edges $E' \subseteq E$ is a set R such that, for every edge-vertex-pair $\{u_1, u_2\}$ of origin $(\mathfrak{u}, \mathfrak{v})$ with $\{\mathfrak{u}, \mathfrak{v}\} \in E'$, exactly one of u_1 and u_2 is contained in R. Intuitively, R defines an order per edge-vertex-pair with origin in E', but does not order edge-vertex-pairs of different origins.

Claim 2. *There is an* IFPC*-formula (independent of G) that defines an edge-vertex-pair-order on all base edges E if $G - \mathsf{orig}(\trianglelefteq^{\mathfrak{A}})$ has no cycles.*

Proof. We first show the following: Whenever for a base vertex there is an edge-vertex-pair-order R of all incident edges apart from one, then R can be extended to the remaining incident edge. This is done as follows (cf. Figure 6.1): Let $\mathfrak{u} \in V$ be a base vertex and let $N_G(\mathfrak{u}) = \{\mathfrak{v}_1, \dots, \mathfrak{v}_d\}$. Assume w.l.o.g. that R orders the edge-vertex-pairs of the base edges $\{\mathfrak{u}, \mathfrak{v}_i\} \in E$ for every $i \in [d-1]$ and let $v_i \in R$ be the edge vertex with origin $(\mathfrak{u}, \mathfrak{v}_i)$ for every $i \in [d-1]$. By construction of the CFI graphs, there is exactly one gadget vertex u of the gadget of $\mathfrak{u}$ adjacent to v_i for all $i \in [d-1]$. Because every gadget vertex is adjacent to exactly one edge vertex per incident edge-vertex-pair, we can add



6.1 Extending edge-vertex-pair-orders. Edge-vertex-pair-orders are extended through one gadget as in Claim 2 in the proof of Lemma 6.3: Gadget vertices are drawn in gray and edge vertices are drawn in light gray (cf. Figure 2.1). The two red-circled vertices are contained in an edge-vertex-pair-order. Because the shown gadget is of degree 3, there is a unique blue gadget vertex adjacent to the two red vertices. The blue vertex has a unique additional adjacent vertex which is shown in red-dotted. By adding this canonical vertex to the edge-vertex-pair-order, it can be extended to the remaining edge-vertex-pair of the gadget.

this unique edge vertex v_d with origin $(\mathfrak{u}, \mathfrak{v}_d)$ and the unique edge vertex v'_d with origin $(\mathfrak{v}_d, \mathfrak{u})$ adjacent to u_d to R. These two vertices can clearly be defined in IFPC (without an order on the $\mathfrak{v}_i$).

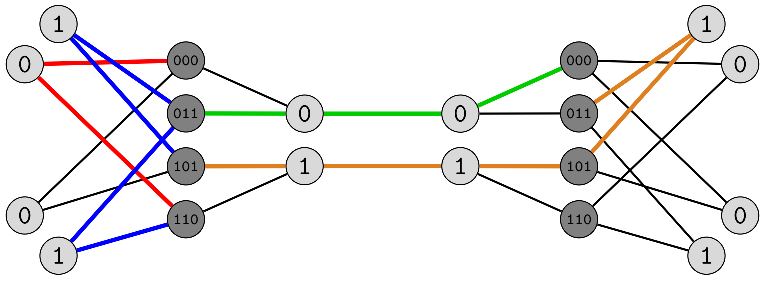
We propagate this approach through gadgets. Assume there is an edge-vertex-pair-order R of E' such that $G - E'$ has no cycles. Then $G - E'$ is a forest and there are vertices of degree one in $G - E'$ (unless $E' = E$). For all degree-one vertices of $G - E'$, we extend R to the remaining incident edge. So unless $E' = E$, we added more edges to E'. Surely, $G - E'$ has still no cycles. So we can repeat this process using a fixed-point operator to define an edge-vertex-pair-order of E.

It is clear that we can turn $\trianglelefteq^{\mathfrak{A}}$ into an edge-vertex-pair-order R of $\mathsf{orig}(\trianglelefteq^{\mathfrak{A}})$ (seen as undirected edges): For all $(\mathfrak{u}, \mathfrak{v}) \in \mathsf{orig}(\trianglelefteq^{\mathfrak{A}})$, at least for one of the edge-vertex-pairs with origin $(\mathfrak{u}, \mathfrak{v})$ is individualized by $\trianglelefteq^{\mathfrak{A}}$. We put the $\trianglelefteq^{\mathfrak{A}}$-minimal such vertex u into R. For the edge-vertex-pair with origin $(\mathfrak{v}, \mathfrak{u})$, we add the unique edge vertex adjacent to u to R (if both $(\mathfrak{u}, \mathfrak{v}), (\mathfrak{v}, \mathfrak{u}) \in \mathsf{orig}(\trianglelefteq^{\mathfrak{A}})$, we start with the directed edge containing the $\trianglelefteq^{\mathfrak{A}}$-minimal vertex). One easily sees that R is IFPC-definable. By assumption, $G - \mathsf{orig}(\trianglelefteq^{\mathfrak{A}})$ has no cycles and thus we can define an edge-vertex-pair-order of E. ⊣

Claim 3. *Assume R is an isomorphism-invariant edge-vertex-pair-order of E, that is, $\mathsf{Aut}((\mathfrak{A}, \trianglelefteq^{\mathfrak{A}})) = \mathsf{Aut}((\mathfrak{A}, \trianglelefteq^{\mathfrak{A}}, R))$, and O is a 2-orbit of $(G, \mathsf{orig}(\trianglelefteq^{\mathfrak{A}}))$. Then the set of edge vertices $\{u \in R \mid \mathsf{orig}(u) \in O\}$ is a 1-orbit of $(\mathfrak{A}, \trianglelefteq^{\mathfrak{A}})$.*

Proof. Let $(\mathfrak{u}, \mathfrak{v}), (\mathfrak{u}', \mathfrak{v}') \in O$, i.e., there is an automorphism $\varphi \in \mathsf{Aut}((G, \mathsf{orig}(\trianglelefteq^{\mathfrak{A}})))$ such that $\varphi((\mathfrak{u}, \mathfrak{v})) = (\mathfrak{u}', \mathfrak{v}')$. Every automorphism of $(G, \mathsf{orig}(\trianglelefteq^{\mathfrak{A}}))$ induces and automorphism of $(\mathfrak{A}, \trianglelefteq^{\mathfrak{A}})$. So there is an automorphism $\psi \in \mathsf{Aut}((\mathfrak{A}, \trianglelefteq^{\mathfrak{A}}))$ mapping the edge-vertex-pair with origin $(\mathfrak{u}, \mathfrak{v})$ to the one with origin $(\mathfrak{u}', \mathfrak{v}')$. Because R is isomorphism-invariant, ψ has to map edge vertices in R to edge vertices in R. Hence, $\{u \in R \mid \mathsf{orig}(u) \in O\}$ is a subset of a 1-orbit of $(\mathfrak{A}, \trianglelefteq^{\mathfrak{A}})$. This set cannot be a proper subset of an orbit because R is isomorphism-invariant. ⊣

Let $\Phi_{\text{2-orb}}(\bar{x}, \bar{y})$ be an IFPC+WSC+I-formula distinguishing 2-orbits of $\mathcal{K}^{\trianglelefteq}$. We cannot evaluate $\Phi_{\text{2-orb}}$ on $\mathfrak{A}$ to define 2-orbits of $(G, \mathsf{orig}(\trianglelefteq^{\mathfrak{A}}))$ because $\mathfrak{A}$ has a more complicated



6.2 Distances in CFI graphs. This figure illustrates the distances in CFI graphs in Claim 4 in the proof of Lemma 6.3: Gadget vertices are drawn in gray and edge vertices are drawn in light gray (cf. Figure 2.1). Two gadget vertices in the same gadgets have distance 2 or 4 as shown by the red and blue path. Two gadget vertices in gadgets for adjacent base vertices have distance 3 or 5 as shown by the green and brown path.

automorphism structure than G and it is not clear how to witness orbits. Here we use the interpretation operator. We define an IFPC-interpretation defining the base graph $(G, \mathsf{orig}(\trianglelefteq^{\mathfrak{A}}))$. Intuitively, we contract all gadgets to a single vertex, remove all edge vertices, and instead directly connect the contracted gadgets. To do so, we need the following claim:

Claim 4. *For every $u \in A$, it holds that u is a gadget vertex if and only if for every $v \in N_{\mathfrak{A}}(u)$ there are two distinct $w, w' \in N_{\mathfrak{A}}(v) \setminus \{u\}$ of different color. Every two distinct gadget vertices $u, v \in A$ with origins $\mathfrak{u} = \mathsf{orig}(u)$ and $\mathfrak{v} = \mathsf{orig}(v)$*

(a) have distance 2 or 4 if and only if $\mathfrak{u} = \mathfrak{v}$ and

(b) have distance 3 or 5 if and only if $\{\mathfrak{u}, \mathfrak{v}\} \in E$.

Proof. We show the first claim (cf. Figure 6.2). Let $u \in A$. Assume that u is a gadget vertex with origin $\mathfrak{u}$ and let $v \in N_{\mathfrak{A}}(u)$. Then v is an edge vertex with origin $(\mathfrak{u}, \mathfrak{v})$. Because the minimal degree of G is 3, there is another gadget vertex $u' \neq u$ with origin $\mathfrak{u}$ adjacent to v. The edge vertex with origin $(\mathfrak{v}, \mathfrak{u})$ adjacent to v has a different color than u'. To show the other direction, assume that for every neighbor $v \in N_{\mathfrak{A}}(u)$ there are two distinct $w, w' \in N_{\mathfrak{A}}(v) \setminus \{u\}$ of different color. For the sake of contradiction, suppose that u is an edge vertex with origin $(\mathfrak{u}, \mathfrak{v})$. Consider the unique edge v vertex with origin $(\mathfrak{v}, \mathfrak{u})$ adjacent to u. Every neighbor of v is either u or a gadget vertex with origin $\mathfrak{v}$, but which all have the same color. Hence, u is a gadget vertex.

To show the second claim (cf. Figure 6.2 again), let $u = (\mathfrak{u}, \bar{a})$ and $v = (\mathfrak{v}, \bar{b})$ be gadget vertices with origins $\mathfrak{u}$ and $\mathfrak{v}$, respectively, i.e., $\bar{a} \in \mathbb{F}_2^{N_G(\mathfrak{u})}$ and $\bar{b} \in \mathbb{F}_2^{N_G(\mathfrak{v})}$ such that $0 = \sum \bar{a} = \sum \bar{b}$. For Part (a), assume $\mathfrak{u} = \mathfrak{v}$. By construction of the CFI gadget, u and v are not adjacent and have a common neighbor if and only if $\bar{a}(\mathfrak{w}) = \bar{b}(\mathfrak{w})$ for some $\mathfrak{w} \in N_G(\mathfrak{u})$. If $\bar{a}(\mathfrak{w}) \neq \bar{b}(\mathfrak{w})$ for all $\mathfrak{w} \in N_G(\mathfrak{u})$, then there is another gadget vertex to which both u and v have distance 2 because the degree if $\mathfrak{u}$ is at least 3. For the other direction, assume u and v have distance 2 or 4. For every neighbor w of u, there is an $\{\mathfrak{u}, \mathfrak{w}\} \in E$ such that w is an edge vertex with origin $(\mathfrak{u}, \mathfrak{w})$ and every neighbor of w is a gadget vertex with origin $\mathfrak{u}$ or an edge vertex with origin $(\mathfrak{w}, \mathfrak{u})$. Hence, if u and v have distance 2, then $\mathfrak{v} = \mathfrak{u}$. The case of distance 4 is similar, here all distance 4 vertices are either vertices of the same gadget or edge vertices.

Proving Part (b) is similar: Assume $\{\mathfrak{u}, \mathfrak{v}\} \in E$. If $\bar{a}(\mathfrak{v}) = \bar{b}(\mathfrak{u})$, then u and v have distance 3 (via two edge vertices with origin $(\mathfrak{u}, \mathfrak{v})$ and $(\mathfrak{v}, \mathfrak{u})$), otherwise they have distance 5. For the other direction, all distance 3 vertices of u are either gadget vertices of gadgets whose origin $\mathfrak{w}$ satisfies $\{\mathfrak{u}, \mathfrak{w}\} \in E$ or edge vertices. $\dashv$

So the interpretation $\Theta = (\Phi_{\text{dom}}, \Phi_{\cong}, \Phi_E, \Phi_{\preceq}, \Phi_{\trianglelefteq})$ defines the base graph:

$$\begin{aligned}
\Phi_{\text{dom}}(x) &:= \forall y.\ E(x,y) \Rightarrow \exists z_1, z_2.\ z_1 \neq x \land z_2 \neq x \land E(y,z_1) \land E(y,z_2) \land z_1 \prec z_2,\\
\Phi_{\cong}(x,y) &:= x = y \lor \Phi^2_{\text{dist}}(x,y) \lor \Phi^4_{\text{dist}}(x,y),\\
\Phi_E(x,y) &:= \Phi^3_{\text{dist}}(x,y) \lor \Phi^5_{\text{dist}}(x,y),\\
\Phi_{\preceq}(x,y) &:= x \preceq y.
\end{aligned}$$

We used formulas $\Phi^\ell_{\text{dist}}(x,y)$ defining that x and y have distance ℓ. It remains to define $\Phi_{\trianglelefteq}$, for which we omit a formal definition: We use $\trianglelefteq$ to define an order on the base edges in $\mathsf{orig}(\trianglelefteq^{\mathfrak{A}})$ and individualize the corresponding base vertices using this order.

Now we are able to evaluate $\Phi_{2\text{-orb}}$ on the base graph. We extend Θ by two parameters z_1 and z_2 for two edge vertices, for which we want to know whether their origins are in the same 2-orbit of G. We add two fresh binary relation symbols S_1 and S_2 and let Θ define S_i to contain the origin of the edge vertex of z_i for every $i \in [2]$. We lift the total preorder of the base vertices to the edge vertices using the interpretation operator:

$$\Phi_{\text{base-orb}}(z_1, z_2) := \mathsf{I}(\Theta(z_1,z_2); \forall \bar{x}\bar{y}.\ S_1(\bar{x}) \Rightarrow S_2(\bar{y}) \Rightarrow \Phi_{2\text{-orb}}(\bar{x}, \bar{y})).$$

The formula $\Phi_{\text{base-orb}}$ defines whether the origins of two edge vertices are in the same orbit of $(G, \mathsf{orig}(\trianglelefteq^{\mathfrak{A}}))$. Note that $\Phi_{2\text{-orb}}$ does not mention the additional relations S_1 and S_2 and thus every WSC-fixed-point operator in $\Phi_{2\text{-orb}}$ is evaluated on a structure not containing S_1 and S_2 (by the reduct semantics) and so indeed on a $\mathcal{K}^{\trianglelefteq}$-graph.

Finally, we can indeed check whether there is an orbit as required by Claim 1: Consider the equivalence classes induced by $\Phi_{\text{base-orb}}$ on the edge vertices and check the existence of the required cycle. We define the minimal such orbit if such one exists. If no such orbit exists, then $G - \mathsf{orig}(\trianglelefteq^{\mathfrak{A}})$ contains no cycles and we can define an edge-vertex-pair-order R of E by Claim 2 in IFPC. Because R is IFPC-definable, R is in particular isomorphism-invariant. If there is a nontrivial 2-orbit of $G - \mathsf{orig}(\trianglelefteq^{\mathfrak{A}})$, we pick the minimal one to define a nontrivial 1-orbit of $(\mathfrak{A}, \trianglelefteq^{\mathfrak{A}})$ using R and Claim 3.

If that is also not the case, then there is no nontrivial 2-orbit of $G - \mathsf{orig}(\trianglelefteq^{\mathfrak{A}})$, that is, there is a definable total order on the (directed) base edges. Together with the edge-vertex-pair-order R, we define a total order on all edge-vertex-pairs, which can be extended to a total order on $\mathfrak{A}$. We define the minimal vertex which is not individualized in $\trianglelefteq^{\mathfrak{A}}$ if it exists. Otherwise, all vertices are individualized. We pick the $\trianglelefteq$-minimal one. $\square$

Now it is straightforward to show that if IFPC+WSC+I canonizes $\mathcal{K}$, then IFPC+WSC+I canonizes $\mathsf{CFI}(\mathcal{K})$, too.

Theorem 6.4. *If* IFPC+WSC+I *canonizes a class of colored base graphs $\mathcal{K}$ closed under individualization, then* IFPC+WSC+I *canonizes the class of CFI graphs* $\mathsf{CFI}(\mathcal{K})$ *over $\mathcal{K}$ (and actually its closure under individualization* $\mathsf{CFI}(\mathcal{K})^{\trianglelefteq}$*).*

Proof. Assume that IFPC+WSC+I defines a canonization for $\mathcal{K}$. Then IFPC+WSC+I distinguishes 2-orbits of $\mathcal{K}$ by Lemma 6.2. Hence, $\mathsf{CFI}(\mathcal{K})$ is ready for individualization in IFPC+WSC+I by Lemma 6.3 and so IFPC+WSC+I defines a canonization for $\mathsf{CFI}(\mathcal{K})$ by Lemma 6.2. □

Corollary 6.5. *If $\mathcal{K}$ is a class of base graphs of bounded degree, then* IFPC+WSC+I *defines canonization for $\mathcal{K}^{\trianglelefteq}$ if and only if it defines canonization for $\mathsf{CFI}(\mathcal{K})^{\trianglelefteq}$.*

Proof. One direction is by Theorem 6.4. For the other direction, let $\mathcal{K}$ be graph class of maximal degree d. Then there is a d-dimensional IFPC-interpretation Θ such that $\Theta(G)$ is the even CFI graph over G for every base graph $G \in \mathcal{K}^{\trianglelefteq}$. Individualized vertices are translated into the coloring and thus into gadgets of unique color. Together with the canonization-defining interpretation and the base-graph-defining one in the proof of Theorem 6.4, we obtain a canonization for $\mathcal{K}^{\trianglelefteq}$. □

Note that this approach of defining a CFI graph to canonize the base graph cannot work for arbitrary base graphs because for large degrees the CFI graphs get exponentially large (which cannot be defined by an IFPC-interpretation).

We observe: First, Theorem 6.4 can be applied iteratively. If IFPC+WSC+I canonizes $\mathcal{K}^{\trianglelefteq}$, then IFPC+WSC+I canonizes $\mathsf{CFI}(\mathcal{K})^{\trianglelefteq}$, and thus IFPC+WSC+I canonizes $\mathsf{CFI}(\mathsf{CFI}(\mathcal{K}))^{\trianglelefteq}$. Second, every application of Theorem 6.4 adds one WSC-fixed-point operator (to define Gurevich's algorithm in Lemma 6.2) and one interpretation operator (for the base-graph-defining interpretation in Lemma 6.3). More precisely, the nesting depth of these operators increases. We will show that this is in some sense necessary: The CFI query on a variation of $\mathsf{CFI}(\mathsf{CFI}(\mathcal{K}))$ cannot be defined without nesting two WSC-fixed-point and two interpretation operators.

6.2.3 Intermezzo: The CFI Query and CPT+WSC

We apply the former results to CPT+WSC to define the CFI query for a class of base graphs in CPT+WSC, for which it is not known that CPT defines the CFI query. Already defining the CFI-query for ordered base graphs is rather difficult in CPT [34]. The best currently known result is that the CFI-query for base graphs of logarithmic color class size or base graphs with linear maximal degree is CPT-definable [104]. The technique for logarithmic color class size is based on constructing deeply nested sets which are invariant under isomorphisms of the CFI graphs and encode their parity. For general base graphs, it is not clear how such sets can be constructed while still satisfying polynomial bounds.

Corollary 6.6. *Let $\mathcal{K}$ be a class of colored base graphs. If* CPT *distinguishes* 2*-orbits for $\mathcal{K}^{\trianglelefteq}$, then* CPT+WSC *defines the CFI-query for $\mathsf{CFI}(\mathcal{K})$.*

Proof. By Theorem 5.21, it suffices to show that $\mathsf{CFI}(\mathcal{K})^{\trianglelefteq}$ is ready for individualization in CPT+WSC, to obtain a CPT+WSC-definable canonization and in particular to define the CFI query. We define the orbits of $\mathsf{CFI}(\mathcal{K})^{\trianglelefteq}$ in the same way as in Lemma 6.3. We cannot immediately apply the lemma for two reasons:

1. CPT+WSC has no interpretation operator. The interpretation operator is used in Lemma 6.3 to define the base graph of a given CFI graph and to evaluate the

orbit-distinguishing formula on the base graph. This becomes important, when the orbit-distinguishing formula itself makes choices, so that they can successfully be witnessed. We, however, assume that the 2-orbits are CPT-distinguishable, so we can actually define them without using the interpretation operator (also cf. Corollary 6.8).

2. CPT+WSC is a three-valued logic using the error marker † when choices are not witnessed. However, we do not rely on this behavior in Lemma 6.3.

Hence, we can show that $\mathsf{CFI}(\mathcal{K})^{\trianglelefteq}$ is ready for individualization in CPT+WSC. □

We show that Corollary 6.6 covers a graph class whose color classes are not of logarithmic size and their maximal degree is not linear.

Corollary 6.7. *There is a class of colored base graphs $\mathcal{K} = \{G_n \mid n \in \mathbb{N}\}$, such that* CPT+WSC *defines the CFI-query for $\mathcal{K}$ and for every $n \in \mathbb{N}$, the graph G_n is $\mathcal{O}(|G_n|^{0.5})$-regular and every color class of G_n has size $\Omega(|G_n|^{0.5})$.*

Proof. It suffices to construct a class $\mathcal{K} = \{G_n \mid n \in \mathbb{N}\}$ of colored base graphs $\mathcal{K}$ with the desired regularity and color-class size properties and which has CPT-definable 2-orbits by Corollary 6.6. Let $n \in \mathbb{N}$ be arbitrary but fixed. We define $G_n = (V, E, \preceq)$ as follows: Start with n many disjoint cliques $K_1, \dots, K_n$ of size n. Then connect every K_i to K_{i+1} (and K_n to K_1) with a complete bipartite graph. Finally, color the graph so that each K_i is a color class. We now consider the 2-orbits of $(G_n, \bar{u})$ for some arbitrary $\bar{u} \in V^*$. For every vertex $v \in V$, it is easy to see that the 1-orbit $O(v)$ containing v is $O(v) = \{v\}$ if v is contained in $\bar{u}$ and otherwise

$$O(v) = \left\{ w \;\middle|\; v, w \in K_i, w \neq u_j \text{ for every } j \in [|\bar{u}|] \right\}.$$

Then the 2-orbit of a tuple (u, v) is

$$O(u,v) = \left\{ (w, w') \;\middle|\; w \in O(u), w' \in O(v), \text{ and } w \neq w' \text{ if and only if } u \neq v \right\}.$$

Clearly, the graph G_n has order n^2, is $(3n-1)$-regular, and has color class size n, satisfying the requirements of the lemma. It is also easy to see that CPT defines the partition of 2-orbits shown above and can order it using the total order on the cliques. □

6.3 The CFI Query and Nesting of Operators

In this section we show that the increased nesting depth of WSC-fixed-point and interpretation operators used to canonize CFI graphs in Theorem 6.4 is unavoidable. Because IFPC does not define the CFI query, it is clear that even if the class of base graph has IFPC-distinguishable orbits (e.g., on a class of ordered graphs of unbounded treewidth), the CFI query cannot be defined in IFPC. So, the nesting depth of WSC-fixed-point operators has to increase. However, for orbits distinguishable in IFPC+WSC+I but not in IFPC it is not clear whether the nesting depth has to increase. We now show that this is indeed the case.

Intuitively, we want to combine non-isomorphic CFI graphs into a new base graph and then apply the CFI construction again. To define orbits of these double CFI graphs, one has to define the CFI query for the base CFI graphs, which cannot be done without a WSC-fixed-point operator. However, parameters of WSC-fixed-point operators complicate matters. If parameter of a WSC-fixed-point operator is used to fix a vertex contained in of the base CFI graphs, then this base CFI graph can be distinguished from all the others. Hence, orbits of this base CFI graph can be defined without defining the CFI query. To overcome this problem, we use multiple copies of the base CFI graphs such that their number exceeds the number of parameters.

If we want to make choices from the base CFI graphs not containing a parameter, we have to define the CFI query for them which increases the nesting depth. If we do not do so, we can essentially only make choices in the base CFI graphs containing a parameter vertex. That is, we can move the twist to the other base CFI graphs. Proving that in this case the CFI query cannot be defined without more operators requires formal effort: We introduce a logic that allows for quantifying over all individualizations of the base CFI graphs containing parameters. This is (potentially) more powerful than making choices, but we still can prove non-definability of the CFI query. This logic has the benefit that we can characterize it via a pebble game.

First, we introduce a formalism for nesting WSC-fixed-point and interpretation operators in Section 6.3.1. Second, we provide a graph construction combining multiple base CFI graphs in Section 6.3.2. Third, we define a logic quantifying over certain individualizations in Section 6.3.3 and prove that it cannot distinguish CFI graphs from the prior graph construction. Last, we make the argument sketched above formal in Section 6.3.4 and show that these CFI graphs require nested WSC-fixed-point operators.

6.3.1 Nested WSC-Fixed-Point and Interpretation Operators

Let $\text{IFPC} \subseteq L \subseteq \text{IFPC+WSC+I}$ be a subset of IFPC+WSC+I. We write $\text{WSC}(L)$ for the set of formulas obtained from L using IFPC-formula-formation rules and WSC-fixed-point operators, for which the step, choice, witnessing, and output formulas are L-formulas. Likewise, we define $\text{I}(L)$: One can use an additional interpretation operator $I(\Theta, \Psi)$, where the interpretation Θ is an L-interpretation and Ψ is an L-formula. Note that $L \subseteq \text{WSC}(L)$, $L \subseteq \text{I}(L)$, and that $\text{I(IFPC)} = \text{IFPC}$ because IFPC is closed under IFPC-interpretations.

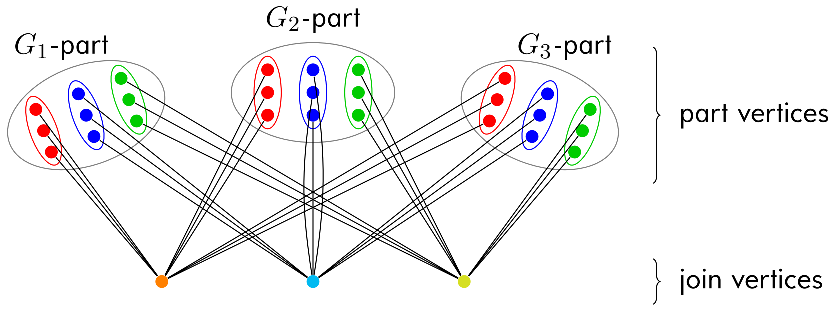
We abbreviate $\text{WSCI}(L) := \text{WSC}(\text{I}(L))$ and $\text{WSCI}^{k+1}(L) := \text{WSCI}(\text{WSCI}^k(L))$. Our goal is to provide CFI graphs proving $\text{WSCI(IFPC)} < \text{WSCI}^2\text{(IFPC)}$. Note the construction in Lemmas 6.2 and 6.3:

Corollary 6.8. *Let $\mathcal{K}$ be a class of colored base graphs.*

1. *If L distinguishes 2-orbits of $\mathcal{K}^{\trianglelefteq}$, then $\mathsf{CFI}(\mathcal{K})^{\trianglelefteq}$ is ready for individualization in $\text{I}(L)$.*

2. *If $\mathsf{CFI}(\mathcal{K})^{\trianglelefteq}$ is ready for individualization in L, then $\text{WSC}(L)$ defines a canonization for $\mathsf{CFI}(\mathcal{K})^{\trianglelefteq}$.*

3. *If L distinguishes 2-orbits of $\mathcal{K}^{\trianglelefteq}$, then $\text{WSCI}(L)$ defines a canonization for $\mathsf{CFI}(\mathcal{K})^{\trianglelefteq}$.*



6.3 Color class joins. The figure shows an example of a color class join. For three graphs G_1, G_2, and G_3, each with three color classes, the figure shows the color class join $J_{\text{cc}}(G_1, G_2, G_3)$. Edges inside the graphs G_1 to G_3 are not drawn. For each color class, one new vertex is added and connected to all existing vertices of that color class. The new vertices receive new unique colors.

Proof. The first claim follows from the proof of Lemma 6.3. The second claim follows from the proof of Lemma 6.2 and the fact that implementing Gurevich's canonization algorithm requires one WSC-fixed-point operator [89]. Combining the first and second claim yields the last one. □

6.3.2 Color Class Joins and CFI Graphs

We define a composition operation of graphs. Let $G_1, \dots, G_\ell$ be a sequence of connected colored graphs, such that all G_i have the same number of color classes c. The **color class join** $J_{\text{cc}}(G_1, \dots, G_\ell)$ is defined as follows: Start with the disjoint union of the G_i and add c many additional vertices $u_1, \dots, u_c$. Then add for every $i \in [c]$ edges between u_i and every vertex v in the i-th color class of every G_j (cf. Figure 6.3). The resulting colored graph $J_{\text{cc}}(G_1, \dots, G_\ell)$ has $2c$ many color classes: one color class for each u_i and the union of the color classes of the G_j. We call the G_j the **parts** of $J_{\text{cc}}(G_1, \dots, G_\ell)$. The vertices u_i are called the **join vertices** and the other ones the **part vertices**. The **part of** a part vertex v is the graph G_j containing v.

The color class join has two crucial properties: First, defining orbits of $J_{\text{cc}}(G_1, \dots, G_\ell)$ is at least as hard as defining isomorphism of the G_j.

Lemma 6.9. *If two part vertices v and v' are in the same orbit of $J_{\text{cc}}(G_1, \dots, G_\ell)$, then the part of v is isomorphic to the one of v'.*

Proof. Because every G_i is connected and the part and join vertices have different colors, an automorphism of $J_{\text{cc}}(G_1, \dots, G_\ell)$ mapping v to v' has to map the part of v to the part of v'. In particular, these parts are isomorphic. □

Second, the automorphism structure of a part G_j is independent of individualizing vertices in other parts (or join vertices).

Lemma 6.10. *Let $\bar{w}$ be a tuple of vertices of $J_{\text{cc}}(G_1, \dots, G_\ell)$ and $j \in [\ell]$. If there is no $i \in [|\bar{w}|]$ such that G_j is the part of w_i and if v and v' are in the same orbit of G_j, then v and v' are in the same orbit of $(J_{\text{cc}}(G_1, \dots, G_\ell), \bar{w})$.*

Proof. Because v and v' are in the same orbit of G_j, there is a $\varphi \in \mathsf{Aut}(G_j)$ such that $\varphi(v) = v'$. We extend φ to $J_{\text{cc}}(G_1, \dots, G_\ell)$ by the identity on all join vertices and all parts apart from G_j. Because φ respects the colors classes of G_j and the join vertex u_i is adjacent to all vertices in the i-th color class of G_j for every $i \in [c]$, adjacency of part vertices of G_j and join vertices is preserved by φ. Thus, $\varphi \in \mathsf{Aut}((J_{\text{cc}}(G_1, \dots, G_\ell), \bar{w}))$ because the part of w_i is not G_j for every $i \in [|\bar{w}|]$ and φ is the identity on all other parts. □

Color Class Joins of CFI Graphs. Now we apply color class joins to CFI graphs: For connected colored graphs G, H, and K with the same number of color classes, we define

$$J^k_{\text{cc}}(G, H, K) := J_{\text{cc}}(\underbrace{G, \dots, G}_{k \text{ many}}, \underbrace{H, \dots, H}_{k \text{ many}}, \underbrace{K, \dots, K}_{k \text{ many}}).$$

Let $\mathcal{K}$ be a class of colored base graphs. For $G \in \mathcal{K}$ and $g \in \mathbb{F}_2$, we define

$$\begin{aligned} \mathsf{CFI}^k(G, g) &:= J^k_{\text{cc}}(\mathsf{CFI}(G, 0), \mathsf{CFI}(G, g), \mathsf{CFI}(G, 1)), \\ \mathsf{CFI}^k(\mathcal{K}) &:= \left\{ \mathsf{CFI}^k(G, g) \;\middle|\; G \in \mathcal{K}, g \in \mathbb{F}_2 \right\}, \\ \mathsf{CFI}^\omega(\mathcal{K}) &:= \bigcup_{k \in \mathbb{N}} \mathsf{CFI}^k(\mathcal{K}). \end{aligned}$$

Lemma 6.11. *Let* IFPC $\subseteq L \subseteq$ IFPC+WSC+I *be a subset of* IFPC+WSC+I *closed under* IFPC*-formula-formation rules. If L canonizes $\mathsf{CFI}(\mathcal{K})^\trianglelefteq$, then L canonizes $\mathsf{CFI}^\omega(\mathcal{K})^\trianglelefteq$.*

Proof. First, we can easily distinguish join vertices from part vertices in IFPC for graphs in $\mathsf{CFI}^\omega(\mathcal{K})^\trianglelefteq$ because the join vertices are in singleton color classes and the part vertices are not. So for a given part vertex u of G, we can define the set of all part vertices contained in the same part as u (namely the ones reachable without using a join vertex), that is, we define the CFI graph in $\mathsf{CFI}(\mathcal{K})$ containing u.

Let Θ_{can} be an L-canonization of $\mathsf{CFI}(\mathcal{K})^\trianglelefteq$. Then we obtain an L-interpretation $\Theta_{\text{part-can}}(x)$, which given a $\mathsf{CFI}^\omega(\mathcal{K})$-graph canonizes the part of x. It essentially is Θ_{can} but only considers the (definable) set of part vertices in the same part as x. So every choice-set in the evaluation of $\Theta_{\text{part-can}}(x)$ will be a set of part vertices in the same part as x and thus a choice-set is an orbit if and only if the corresponding choice-set in the evaluation of Θ_{can} is an orbit. The witnessing automorphisms are obtained by extending the witnessing automorphisms defined in Θ_{can} with the identity on all vertices not in the part of x. In this way, exactly the same choices are successfully witnessed by $\Theta_{\text{part-can}}$ as by Θ_{can}. Note that $\Theta_{\text{part-can}}$ is an L-interpretation because L is closed under IFPC-formula-formation rules.

We use $\Theta_{\text{part-can}}(x)$ to define the canon of every part containing an $\trianglelefteq$-individualized vertex. These canons can be ordered according to the order of the individualized vertices. For the remaining parts not containing individualized vertices, we use $\Theta_{\text{part-can}}(x)$ to determine how many of these parts are even CFI graphs and how many are odd ones. We obtain a canon for the even and odd graph (if they occur) and define as many copies as needed (using numeric variables). The copies can be ordered. We take the disjoint union of all these canons, and lastly add the join vertices, which all is IFPC-definable. □

CFI Graphs of Color Class Joins. Now, we use color class joins as base graphs. We introduce terminology for CFI graphs over color class joins. Let $G_1, \dots, G_\ell$ be a sequence of base graphs and $h \in \mathbb{F}_2$. We transfer the notion of part and join vertices from $H := J_{\text{cc}}(G_1, \dots, G_\ell)$ to $\mathfrak{A} := \mathsf{CFI}(H, h)$. The G_i**-part** of $\mathfrak{A}$ is the set of vertices originating from a vertex or edge of G_i in H. These vertices are called **part vertices of** G_i. A vertex is just a part vertex if it is a part vertex of some G_i. The remaining vertices are the **join vertices**.

We consider a special class of individualizations of $\mathfrak{A}$. Let $\bar{u} \in A^*$. A part of $\mathfrak{A}$ is $\bar{u}$**-pebbled** if u_i is a part vertex of that part for some $i \in [k]$. Otherwise, the part is $\bar{u}$**-unpebbled**. The set of $\bar{u}$**-pebbled-part vertices** $V_{\bar{u}}(\mathfrak{A})$ is the set of all join vertices and all part vertices of every $\bar{u}$-pebbled part. The set of $\bar{u}$**-pebbled-part individualizations** $P_{\bar{u}}(\mathfrak{A})$ is the set of all individualizations of $V_{\bar{u}}(\mathfrak{A})$.

Definition 6.12 (Unpebbled-Part-Distinguishing). For an ℓ-tuple $\bar{u} \in A^\ell$, a k-ary relation $R \subseteq A^k$ is $\bar{u}$**-unpebbled-part-distinguishing** if there are $m \in [k]$ and $i \neq j \in [\ell]$ such that

1. both the G_i-part and the G_j-part of $\mathfrak{A}$ are $\bar{u}$-unpebbled,
2. there is a $\bar{v} \in R$ such that v_m is a part vertex of G_i, and
3. for every $\bar{w} \in R$, the vertex w_m is not a part vertex of G_j.

Lemma 6.13. *Let $\bar{u} \in A^\ell$, $G_i \not\cong G_j$ be $\bar{u}$-unpebbled, $R \subseteq A^k$, and some $\bar{v} \in R$ contain a vertex of a $\bar{u}$-unpebbled part. If R is an orbit of $(\mathfrak{A}, \bar{u})$, then R is $\bar{u}$-unpebbled-part-distinguishing.*

Proof. Let the part of v_m be $\bar{u}$-unpebbled and let this part be the G_n-part. Assume that R is an orbit of $(\mathfrak{A}, \bar{u})$. If the part of w_m is neither G_i nor G_j for every $\bar{w} \in R$, then $n \notin \{i, j\}$. Thus, R is $\bar{u}$-unpebbled-part-distinguishing because the G_i-part and the G_n-part are $\bar{u}$-unpebbled, v_m is a part vertex of G_n, and for every $\bar{w} \in R$, the vertex w_m is not a part vertex of G_i.

Otherwise, there is a $\bar{w} \in R$ such that the part of w_m is w.l.o.g. G_i. Then no automorphism can map w_m to a vertex whose part is G_j because $G_i \not\cong G_j$ (Lemma 6.9). Thus, w_m is not a part vertex of G_j for every $\bar{w} \in R$ because R is an orbit. It follows that R is $\bar{u}$-unpebbled-part-distinguishing. □

6.3.3 Quantifying over Pebbled-Part Individualizations

We now define an extension of $\mathcal{C}_k$ which allows for quantifying over pebbled-part individualizations. Our (unnatural) extension $\mathcal{P}_k$ can only be evaluated on CFI graphs over color class joins and will be a tool to show WSC(IFPC)-undefinability. The benefit of this logic is that we will characterize it via a pebble game.

Whenever $\Phi(\bar{x})$ is a $\mathcal{C}_k[\tau, \trianglelefteq_P]$-formula (for a binary relation symbol $\trianglelefteq_P \notin \tau$), then

$$(\exists^P \trianglelefteq_P .\, \Phi)(\bar{x})$$

is a $\mathcal{P}_k[\tau]$-formula. $\mathcal{P}_k[\tau]$-formulas can be combined as usual in $\mathcal{C}_k$ with boolean connectives and counting quantifiers. Note that $\exists^P$-quantifiers *cannot* be nested. Let $G_1, \dots, G_\ell$

be a sequence of colored base graphs with the same number of colors, $g \in \mathbb{F}_2$, and $\mathfrak{A} = \mathsf{CFI}(J_{\text{cc}}(G_1, \dots, G_n), g)$. The $\exists^P$-quantifier has the following semantics:

$$(\exists^P \trianglelefteq_P .\, \Phi)^{\mathfrak{A}} := \left\{ \bar{u} \in A^{|\bar{x}|} \;\middle|\; \bar{u} \in \Phi^{(\mathfrak{A}, \trianglelefteq_P^{\mathfrak{A}})} \text{ for some } \trianglelefteq_P^{\mathfrak{A}} \in P_{\bar{u}}(\mathfrak{A}) \right\}.$$

That is, the $\exists^P$-quantifier quantifies over a pebbled-part individualization for the free variables of Φ. Note that the quantifier does not bind first-order variables.

We now characterize $\mathcal{P}_k$ by an Ehrenfeucht-Fraïssé-like pebble game, which is an extension of the bijective k-pebble game. The **$\mathcal{P}_k$-game** is a game between two players called Spoiler and Duplicator. There are two types of pebbles. First, there a k many pebble pairs (p_i, q_i) for every $i \in [k]$. Second, there are pebble pairs (a_i, b_i) for every $i \in \mathbb{N}$. The game is played on CFI graphs over color class joins $\mathfrak{A}$ and $\mathfrak{B}$ satisfying $|A| = |B|$ (otherwise Spoiler wins immediately). A position in the game is a tuple $(\mathfrak{A}, \trianglelefteq_P^{\mathfrak{A}}, \bar{u}; \mathfrak{B}, \trianglelefteq_P^{\mathfrak{B}}, \bar{v})$, where $\bar{u} \in A^{\leq k}$ and $\bar{v} \in B^{\leq k}$ are of the same length, and $\trianglelefteq_P^{\mathfrak{A}}$ and $\trianglelefteq_P^{\mathfrak{B}}$ individualize vertices (possibly none) originating from up to k parts or join vertices in $\mathfrak{A}$ and $\mathfrak{B}$, respectively. A pebble p_j is placed on u_i and the pebble q_j is placed on v_i for some $j \in [k]$ (again, the exact pebble pair (p_j, q_j) placed on the i-th entries will not matter). The pebble a_i is placed on the i-th individualized vertex by $\trianglelefteq_P^{\mathfrak{A}}$ and b_i on the i-th individualized vertex by $\trianglelefteq_P^{\mathfrak{B}}$. The initial position is $(\mathfrak{A}, \emptyset, (); \mathfrak{B}, \emptyset, ())$, where $()$ denotes the empty tuple. Spoiler can perform two kinds of moves.

- A **regular move** proceeds as in the bijective k-pebble game: Spoiler picks up a pair of pebbles (p_i, q_i). Then Duplicator provides a bijection $\lambda \colon A \to B$. Spoiler places p_i on $u \in A$ and q_i on $\lambda(u)$.

- A **P-move** proceeds as follows and can only be performed once by Spoiler (that is, if $\trianglelefteq_P^{\mathfrak{A}} = \trianglelefteq_P^{\mathfrak{B}} = \emptyset$): Spoiler places the pebbles a_i (or the pebbles b_i, respectively) exactly on the $\bar{u}$-pebbled-part vertices of $\mathfrak{A}$ (or on the $\bar{v}$-pebbled-part vertices of $\mathfrak{B}$, respectively). Duplicator responds by placing the pebbles b_i (or the pebbles a_i, respectively) on exactly the $\bar{v}$-pebbled-part vertices of $\mathfrak{B}$ (or the $\bar{u}$-pebbled-part vertices of $\mathfrak{A}$, respectively).

If after a move there is no pebble-respecting local isomorphism of the pebble-induced substructures of $\mathfrak{A}$ and $\mathfrak{B}$, then Spoiler wins. Duplicator wins if Spoiler never wins.

Lemma 6.14. *For every $k \geq 3$, Spoiler has a winning strategy in the $\mathcal{P}_k$-game at position $(\mathfrak{A}, \emptyset, \bar{u}; \mathfrak{B}, \emptyset, \bar{v})$ if and only if the logic $\mathcal{P}_k$ distinguishes $(\mathfrak{A}, \bar{u})$ and $(\mathfrak{B}, \bar{v})$.*

Proof. Let $k \geq 3$. We first consider positions $(\mathfrak{A}, \trianglelefteq_P^{\mathfrak{A}}, \bar{u}; \mathfrak{B}, \trianglelefteq_P^{\mathfrak{B}}, \bar{v})$ where Spoiler already has performed the P-move, i.e., $\trianglelefteq_P^{\mathfrak{A}}, \trianglelefteq_P^{\mathfrak{B}} \neq \emptyset$. Then the remaining game is essentially just the bijective k-pebble game, where, for each $i \in \mathbb{N}$, the vertices pebbled by a_i and b_i are put in a unique singleton relation. Spoiler has a winning strategy at the position $(\mathfrak{A}, \trianglelefteq_P^{\mathfrak{A}}, \bar{u}; \mathfrak{B}, \trianglelefteq_P^{\mathfrak{B}}, \bar{v})$ if and only if $\mathcal{C}_k$ distinguishes $(\mathfrak{A}, \trianglelefteq_P^{\mathfrak{A}}, \bar{u})$ and $(\mathfrak{B}, \trianglelefteq_P^{\mathfrak{B}}, \bar{v})$ because for $k \geq 3$ we can define the i-th vertex individualized by $\trianglelefteq_P^{\mathfrak{A}}$ or $\trianglelefteq_P^{\mathfrak{B}}$.

We now prove by induction on the number of moves, that if Spoiler has a winning strategy in the $\mathcal{P}_k$-game at position $(\mathfrak{A}, \emptyset, \bar{u}; \mathfrak{B}, \emptyset, \bar{v})$, then $\mathcal{P}_k$ distinguishes $(\mathfrak{A}, \bar{u})$ and $(\mathfrak{B}, \bar{v})$. If Spoiler wins without performing a P-move, then Spoiler wins the bijective pebble game and $\mathcal{C}_k$ and thus also $\mathcal{P}_k$ distinguishes $(\mathfrak{A}, \bar{u})$ and $(\mathfrak{B}, \bar{v})$. So assume Spoiler eventually performs a P-move.

Assume that Spoiler performs a regular move and picks up the i-th pebble pair (p_i, q_i). The argument is essentially the same as for the bijective k-pebble game: For every bijection $\lambda\colon A \to B$, there is a $w \in A$ such that Spoiler wins the $\mathcal{P}_k$-game when placing p_i on w and q_i on $\lambda(w)$. For all these positions, there is a $\mathcal{P}_k$-formula distinguishing them by the induction hypothesis. So some boolean combination of these distinguishing formulas is satisfied by a different number of vertices in $(\mathfrak{A}, \bar{u})$ and $(\mathfrak{B}, \bar{v})$ and we can distinguish them using a counting quantifier.

Now assume that Spoiler performs a P-move: W.l.o.g. Spoiler places the pebbles a_i on all $\bar{u}$-pebbled-part vertices of $\mathfrak{A}$ (inducing the individualization $\trianglelefteq_P^{\mathfrak{A}}$). Then for every placement of the pebbles b_i by Duplicator on the $\bar{v}$-pebbled-part vertices of $\mathfrak{B}$ (inducing $\trianglelefteq_P^{\mathfrak{B}}$), Spoiler has a winning strategy in the bijective k-pebble game at position $(\mathfrak{A}, \trianglelefteq_P^{\mathfrak{A}}, \bar{u}; \mathfrak{B}, \trianglelefteq_P^{\mathfrak{B}}, \bar{v})$ (no P-move is allowed anymore). Then, as argued before, there is a $\mathcal{C}_k$-formula Φ distinguishing $(\mathfrak{A}, \trianglelefteq_P^{\mathfrak{A}}, \bar{u})$ and $(\mathfrak{B}, \trianglelefteq_P^{\mathfrak{B}}, \bar{v})$. So the $\mathcal{P}_k$-formula $\exists^P \trianglelefteq_P.\, \Phi$ distinguishes $(\mathfrak{A}, \bar{u})$ and $(\mathfrak{B}, \bar{v})$.

For the other direction, we prove by induction on the quantifier depth that if a $\mathcal{P}_k$-formula Φ distinguishes $(\mathfrak{A}, \bar{u})$ and $(\mathfrak{B}, \bar{v})$, then Spoiler has a winning strategy in the $\mathcal{P}_k$-game at position $(\mathfrak{A}, \emptyset, \bar{u}; \mathfrak{B}, \emptyset, \bar{v})$. If Φ is actually a $\mathcal{C}_k$-formula, then Spoiler wins the bijective k-pebble game so in particular the $\mathcal{P}_k$-game. If Φ is $\Psi \wedge \Psi'$, $\Psi \vee \Psi'$, or $\neg\Psi$, then one of Ψ and Ψ' distinguishes $(\mathfrak{A}, \bar{u})$ and $(\mathfrak{B}, \bar{v})$. If Φ is a counting quantifier $\exists^{\leq i} x.\, \Psi$, then Spoiler performs a regular move. Because Φ has at most $k-1$ free variables, Spoiler can pick up a pair of pebbles (p_i, q_i). Whatever bijection λ Duplicator chooses, there is a vertex w such that w.l.o.g. w satisfies Ψ in $\mathfrak{A}$ but $\lambda(w)$ does not satisfy Ψ in $\mathfrak{B}$. That is, Spoiler places p_i on w and q_i on $\lambda(w)$ and wins by the induction hypothesis.

To the end, assume that Φ is the $\mathcal{P}_k$-formula $\exists^P \trianglelefteq_P.\, \Psi$. W.l.o.g. we assume that $(\mathfrak{A}, \bar{u})$ satisfies Φ. So there is a $\trianglelefteq_P^{\mathfrak{A}} \in P_{\bar{u}}(\mathfrak{A})$ satisfying $\bar{u} \in \Psi^{(\mathfrak{A}, \trianglelefteq_P^{\mathfrak{A}})}$ such that for every $\trianglelefteq_P^{\mathfrak{B}} \in P_{\bar{v}}(\mathfrak{B})$, it holds that $\bar{v} \notin \Psi^{(\mathfrak{B}, \trianglelefteq_P^{\mathfrak{B}})}$. Because Ψ is a $\mathcal{C}_k$ formula, Spoiler has a winning strategy in the k-bijective pebble game at position $(\mathfrak{A}, \trianglelefteq_P^{\mathfrak{A}}, \bar{u}; \mathfrak{B}, \trianglelefteq_P^{\mathfrak{B}}, \bar{v})$. By performing a P-move and placing the pebbles a_i according to $\trianglelefteq_P^{\mathfrak{A}}$, Spoiler obtains a winning strategy in the $\mathcal{P}_k$ game at position $(\mathfrak{A}, \emptyset, \bar{u}; \mathfrak{B}, \emptyset, \bar{v})$. □

Note that the former lemma only holds for $k \geq 3$ because with fewer variables we cannot define the i-th individualized vertex in $\mathcal{C}_k$ and thus cannot check local isomorphisms. Extending the logic to make the lemma hold for every k only complicates matters and is not needed in the following.

Lemma 6.15. *Let $G_1, \dots, G_{k+1}$ be a sequence of colored base graphs, each with $c > k$ many color classes, such that $\mathsf{CFI}(G_i, 0) \equiv_{\mathcal{C}}^k \mathsf{CFI}(G_i, 1)$ for every $i \in [k+1]$. Then Duplicator has a winning strategy in the $\mathcal{P}_k$-game played on $\mathsf{CFI}(J_{cc}(G_1, \dots, G_{k+1}), 0)$ and $\mathsf{CFI}(J_{cc}(G_1, \dots, G_{k+1}), 1)$.*

Proof. Let $\mathfrak{A} = \mathsf{CFI}(J_{\mathrm{cc}}(G_1, \dots, G_{k+1}), 0)$ and $\mathfrak{B} = \mathsf{CFI}(J_{\mathrm{cc}}(G_1, \dots, G_{k+1}), 1)$. We call the G_j-part in $\mathfrak{A}$ and $\mathfrak{B}$ just the j-th part. Let $V_J \subseteq A = B$ be the set of join vertices and $V_j \subseteq A = B$ be the set of part vertices of the j-th part for every $j \in [k+1]$ (recall that CFI graphs are defined over the same vertex set).

Duplicator maintains the following invariant. At every position $(\mathfrak{A}, \trianglelefteq_P^{\mathfrak{A}}, \bar{u}; \mathfrak{B}, \trianglelefteq_P^{\mathfrak{B}}, \bar{v})$ during the game, there is an isomorphism $\psi\colon (\mathfrak{B}, \bar{v}, \trianglelefteq_P^{\mathfrak{B}}) \to (\mathfrak{B}', \bar{v}, \trianglelefteq_P^{\mathfrak{B}})$ for some $\mathfrak{B}'$ that moves the twisted edge, so $B' = B$, and there is a $j \in [k+1]$ satisfying the following:

1. If Spoiler has not performed the P-move (i.e., the pebbles a_i and b_i are not placed), then the part of every u_i and every v_i is not the j-th one.

2. If Spoiler has performed the P-move, then no vertex of the j-th part is individualized by $\trianglelefteq_P^{\mathfrak{A}}$ or $\trianglelefteq_P^{\mathfrak{B}}$.

3. There is an isomorphism $\varphi\colon (\mathfrak{A}, \trianglelefteq_P^{\mathfrak{A}}, \bar{u})[A \setminus V_j] \to (\psi(\mathfrak{B}), \trianglelefteq_P^{\mathfrak{B}}, \bar{v})[B \setminus V_j]$ respecting the parts, that is, φ maps the i-th part of $\mathfrak{A}$ to the i-th part of $\psi(\mathfrak{B})$.

4. $(\mathfrak{A}, \trianglelefteq_J, \bar{u})[V_J \cup V_j] \equiv_{\mathcal{C}}^k (\psi(\mathfrak{B}), \psi(\trianglelefteq_J), \bar{v})[V_J \cup V_j]$ for some individualization $\trianglelefteq_J$ of V_J.

Note that Property 4 is satisfied either by none or all such individualizations. Clearly, the invariant is satisfied initially.

Assume that it is Spoiler's turn and Spoiler performs a regular move. Spoiler picks up a pebble pair (p_i, q_i). Now Duplicator has to provide a bijection λ. The bijection λ is constructed in the following way. We start with the isomorphism φ of Property 3 and extend it on the j-th part as follows: If Spoiler has performed the P-move already, Duplicator uses the bijection λ' of Duplicator's winning strategy on $(\mathfrak{A}, \trianglelefteq_J, \bar{u})[V_J \cup V_j] \equiv_{\mathcal{C}}^k (\psi(\mathfrak{B}), \psi(\trianglelefteq_J), \bar{v})[V_J \cup V_j]$ for some and thus every $\trianglelefteq_J$ of V_J (Property 4) to extend φ on V_j:

$$\lambda(w) := \begin{cases} \varphi(w) & \text{if } w \notin V_j, \\ \lambda'(w) & \text{otherwise.} \end{cases}$$

Because in this game the individualization $\trianglelefteq_J$ is used for $\mathfrak{A}$ and $\psi(\trianglelefteq_J)$ is used in for $\mathfrak{B}$, the bijection λ' necessarily has to map the i-th $\trianglelefteq_J$-individualized vertex to the i-th $\psi(\trianglelefteq_J)$-individualized vertex. That is, λ and λ' necessarily agree on the join vertices, that is, $\lambda(w) = \lambda'(w)$ for all $w \in V_J$. Spoiler places p_i on w and q_i on $\lambda(w)$. The pebbles still induce a local isomorphism (because φ is an isomorphism and λ' is given by a winning strategy). Properties 1 and 2 are obviously satisfied. If $w \notin V_j$, then $\varphi(w) = \lambda(w)$ and the isomorphism φ still satisfies Property 3. If $w \notin V_J$, then Property 4 is satisfied because the new pebbles are not placed on $(\mathfrak{A}, \trianglelefteq_J, \bar{u})[V_J \cup V_j]$ and $(\psi(\mathfrak{B}), \psi(\trianglelefteq_J), \bar{v})[V_J \cup V_j]$. If $w \in V_j$, then the pebbles are placed according to a winning startegy of Duplicator and thus Property 4 is satisfied, too. If otherwise $w \in V_j$, then w is not in the domain of φ and thus φ satisfies Property 3. Property 4 is satisfied because $\lambda(w) = \lambda'(w)$ and λ' was obtained by a winning strategy of Duplicator.

If Spoiler has not performed the P-move, then Duplicator extends φ as follows. There is another $\bar{u}$-unpebbled part different from the j-th one because there are at most $k-1$ many pebbles placed (one pebble pair is picked up). Let this be the ℓ-th part for some $\ell \neq j$ and let $\{\mathfrak{u}, \mathfrak{v}\}$ be the twisted edge between $\mathfrak{A}$ and $\psi(\mathfrak{B})$, which is contained in the j-th part (by Property 3). There is a path from one of $\mathfrak{u}$ or $\mathfrak{v}$ into the ℓ-th part in $J_{\mathrm{cc}}(G_1, \dots, G_{k+1})$ only using vertices of the j-th and ℓ-th part and one join vertex $\mathfrak{w}$ such that it does not use the origin of the pebbled vertices: Both G_j and G_ℓ are connected, do not contain any pebbles, and there are $c > k$ color classes, so one join vertex is not pebbled. Hence, there is a path-isomorphism $\psi'\colon (\mathfrak{B}', \bar{v}) \to (\mathfrak{B}'', \bar{v})$ moving the twist from the j-th into the ℓ-th part along that path. Thus, we can extend the restriction

$$\psi'|_{A \setminus V_j \setminus V_\ell} \circ \varphi|_{A \setminus V_\ell}\colon (\mathfrak{A}, \bar{u})[A \setminus V_j \setminus V_\ell] \to (\psi'(\psi(\mathfrak{B})), \bar{v})[B \setminus V_j \setminus V_\ell]$$

to V_j because the twist is now in the ℓ-th part. That is, we obtain an isomorphism $\varphi' \colon (\mathfrak{A}, \bar{u})[A \setminus V_\ell] \to (\psi' \circ \psi(\mathfrak{B}), \bar{v})[B \setminus V_\ell]$ which agrees with φ on $A \setminus V_j \setminus V_\ell$ apart from gadget vertices originating from $\mathfrak{w}$ and the edge vertices originating from the edge incident to $\mathfrak{w}$ into the j-th and ℓ-th part. These are the only vertices in $A \setminus V_j \setminus V_\ell$ for which the isomorphism ψ' is not the identity. Duplicator extends φ on V_j using φ' to the bijection λ.

$$\lambda(w) := \begin{cases} \varphi(w) & \text{if } w \notin V_j, \\ \varphi'(w) & \text{otherwise.} \end{cases}$$

Spoiler places p_i on w and q_i on $\lambda(w)$. If $w \notin V_j$ (and so $\lambda(w) = \varphi(w) \notin V_j$), then Properties 1 to 4 are clearly satisfied (now for ℓ instead for j) because φ is an isomorphism. For the same reason, the pebbles induce a local isomorphism.

So assume $w \in V_j$. By Property 1, the first pebble is placed on the j-th part and the ℓ-th part does not contain a pebble. Then the restriction of λ to $A \setminus V_\ell$ can be turned into is an isomorphism $(\mathfrak{A}, \emptyset, \bar{u}w)[A \setminus V_\ell] \to (\mathfrak{B}'', \emptyset, \bar{v}\lambda(w))[B \setminus V_\ell]$ by applying a local automorphism of the gadget of $\mathfrak{w}$ (which is not pebbled) that moves the twist from the j-th into the ℓ-th part according to ψ'. In particular, the pebbles induce a local isomorphism and Property 3 holds. Property 4 is satisfied because the ℓ-th part does not contain a pebble: If $\mathsf{CFI}(G_\ell, 0) \equiv^k_{\mathcal{C}} \mathsf{CFI}(G_\ell, 1)$, then $(\mathfrak{A}, \trianglelefteq_J)[V_J \cup V_\ell] \equiv^k_{\mathcal{C}} (\mathfrak{B}'', \psi(\trianglelefteq_J))[V_J \cup V_\ell]$, too, because $(\mathfrak{A}, \trianglelefteq_J)[V_J \cup V_\ell]$ just extends $\mathsf{CFI}(G_\ell, 0)$ by gadgets for the join vertices, which are all fixed by $\trianglelefteq_J$ (and likewise for $(\mathfrak{B}'', \psi(\trianglelefteq_J))[V_J \cup V_\ell]$ and $\mathsf{CFI}(G_\ell, 1)$).

Finally, let Spoiler perform the P-move. Spoiler places w.l.o.g. the pebbles a_i on the $\bar{u}$-pebbled parts. Duplicator places the pebble b_i on $\varphi(a_i)$ for all i. Because the pebbles are placed according to the isomorphism φ, there is a pebble-respecting local isomorphism. Properties 1 and 2 are clearly satisfied. Property 3 is satisfied by φ and Property 4 is satisfied because no pebble is placed in the j-th part (similar to the ℓ-th part in the former case). □

6.3.4 Nesting Operators to Define the CFI Query is Necessary

We use CFI graphs over color class joins of CFI graphs to construct a class of base graphs, for which WSCI^2(IFPC) defines the CFI query but WSCI(IFPC) does not. Fix a class of totally ordered and 3-regular base graphs $\mathcal{K} := \{G_i \mid i \in \mathbb{N}\}$ such that G_i has treewidth at least i for every $i \in \mathbb{N}$. Such a class exists because we can obtain from some graph G_i' of treewidth i (e.g., a clique of size $i+1$) a 3-regular graph of treewidth at least i as follows: If G_i' has a vertex u of degree greater than 3, then we obtain a new graph G_i'' by splitting u off into two vertices (onto which we equally distribute the edges incident to u) and connecting them via an edge. Contracting this edge yields back G_i'. Thus, G_i' is a minor of G_i'' and thus the treewidth of G_i'' is at least the treewidth of G_i'. We repeat this procedure until every vertex has degree 3.

Lemma 6.16. $\mathsf{CFI}(\mathsf{CFI}(G_k, g), 0) \equiv^k_{\mathcal{C}} \mathsf{CFI}(\mathsf{CFI}(G_k, g), 1)$ *for all* $k \in \mathbb{N}$ *and* $g \in \mathbb{F}_2$.

Proof. Let $k \in \mathbb{N}$ and $g \in \mathbb{F}_2$. The graph G_k has treewidth at least k. The CFI construction does not decrease the treewidth because G_k is a minor of $\mathsf{CFI}(G_k, g)$ (cf. [33]). Hence, $\mathsf{CFI}(G_k, g)$ has treewidth at least k and $\mathsf{CFI}(\mathsf{CFI}(G_k, g), 0) \equiv^k_{\mathcal{C}} \mathsf{CFI}(\mathsf{CFI}(G_k, g), 1)$ by Lemma 2.15. □

Lemma 6.17. WSCI^2(IFPC) *defines the CFI query for* $\mathsf{CFI}(\mathsf{CFI}^{\omega}(\mathcal{K}))$.

Proof. IFPC distinguishes 2-orbits of $\mathcal{K}$ because $\mathcal{K}$-graphs are totally ordered. By Corollary 6.8, WSCI(IFPC) canonizes $\mathsf{CFI}(\mathcal{K})^{\trianglelefteq}$. From Lemma 6.11 it follows that WSCI(IFPC) canonizes $\mathsf{CFI}^{\omega}(G)^{\trianglelefteq}$ and so also distinguishes 2-orbits of $\mathsf{CFI}^{\omega}(G)^{\trianglelefteq}$. Again due to Corollary 6.8, WSCI^2(IFPC) canonizes $\mathsf{CFI}(\mathsf{CFI}^{\omega}(\mathcal{K}))^{\trianglelefteq}$. In particular, WSCI^2(IFPC) defines the CFI query for $\mathsf{CFI}(\mathsf{CFI}^{\omega}(\mathcal{K}))$. □

To show that WSCI(IFPC) does not define the CFI query for $\mathsf{CFI}(\mathsf{CFI}^{\omega}(\mathcal{K}))$, we will use the following idea: Assume that a WSC-fixed-point operator Φ defines the CFI query and we evaluate Φ on $\mathsf{CFI}(\mathsf{CFI}^{\ell}(G))$ for some $\ell > |\bar{p}|$. If Φ always defines choice-sets containing only tuples of vertices in the parameter-pebbled parts, then the twist can be moved in the parameter-unpebbled parts. Because all choices are made in parameter-pebbled parts, the output formula of Φ, which is an IFPC-formula, essentially has to define the CFI query for $\mathsf{CFI}(\mathsf{CFI}(\mathcal{K}))$, which is not possible (Lemma 6.16). Otherwise, Φ makes a choice in parameter-unpebbled parts. But for that, Φ has to distinguish $\mathsf{CFI}(G,0)$ from $\mathsf{CFI}(G,1)$ to define orbits of $\mathsf{CFI}(\mathsf{CFI}^{\ell}(G))$. So the choice IFPC-formula has to define the CFI query for $\mathsf{CFI}(\mathcal{K})$, which is also not possible. Making this idea formal turns out to be tedious.

Lemma 6.18. WSCI(IFPC) *does not define the CFI query for* $\mathsf{CFI}(\mathsf{CFI}^{\omega}(\mathcal{K}))$.

Proof. For a sake of contradiction, suppose that Φ is a WSCI(IFPC)-formula defining the CFI query for $\mathsf{CFI}(\mathsf{CFI}^{\omega}(\mathcal{K}))$. W.l.o.g. we assume that Φ binds no variable twice. Because I(IFPC) = IFPC, we assume that Φ is a WSC(IFPC)-formula. Let $\Psi_1(\bar{x}_1), \dots, \Psi_m(\bar{x}_m)$ be all WSC-fixed-point operators that are subformulas of Φ. For the moment assume that all free variables $\bar{x}_i$ are element variables. Let the number of distinct variables of Φ be k and let $\ell := \ell(k) \geq \max\{k, 3\}$ for some function $\ell(k)$ to be defined later. We consider the subclass $\mathsf{CFI}(\mathsf{CFI}^{\ell+1}(\mathcal{K})) \subseteq \mathsf{CFI}(\mathsf{CFI}^{\omega}(\mathcal{K}))$. We partition $\mathcal{K}$ as follows: First, let $\mathcal{K}_{\text{orb}}$ be the set of all $G \in \mathcal{K}$ such that, for every $g \in \mathbb{F}_2$, there are $h \in \mathbb{F}_2$, $j \in [m]$, and a $|\bar{x}_j|$-tuple $\bar{u}$ of vertices of $\mathsf{CFI}(\mathsf{CFI}^{\ell+1}(G,g),h))$ such that

1. all choice-sets during the evaluation of $\Psi_j(\bar{u})$ on $\mathsf{CFI}(\mathsf{CFI}^{\ell+1}(G,g),h)$ are indeed orbits and
2. one of these choice-sets is $\bar{u}$-unpebbled-part-distinguishing.

Second, set $\mathcal{K}_{\text{cfi}} := \mathcal{K} \setminus \mathcal{K}_{\text{orb}}$. Clearly, at least one of $\mathcal{K}_{\text{orb}}$ and $\mathcal{K}_{\text{cfi}}$ is infinite, which is a contradiction as shown in the two following claims.

Claim 1. $\mathcal{K}_{orb}$ *is finite.*

Proof. We claim that there is an IFPC-formula defining the CFI query for $\mathsf{CFI}(\mathcal{K}_{\text{orb}})$. First, we show that there is an IFPC-interpretation mapping for $G \in \mathcal{K}_{\text{orb}}$ (and even in $\mathcal{K}$) a CFI graph $\mathsf{CFI}(G,g)$ and an $h \in \mathbb{F}_2$ to the graph $(\mathsf{CFI}(\mathsf{CFI}^{\ell+1}(G,g),h), \trianglelefteq)$ such that $\trianglelefteq$ individualizes the vertices of $\ell+1$ many $\mathsf{CFI}(G,0)$-parts, $\ell+1$ many $\mathsf{CFI}(G,1)$-parts, and all join vertices of $\mathsf{CFI}(\mathsf{CFI}^{\ell+1}(G,g),h)$. The following mappings are definable by IFPC-interpretations:

(a) Map $\mathsf{CFI}(G,g)$ to the base graph G (which is an ordered graph).

(b) Map G and $g' \in \mathbb{F}_2$ to $(\mathsf{CFI}(G,g'),\leq)$ such that $\leq$ is a total order on $\mathsf{CFI}(G,g')$. This map is IFPC-definable because G is 3-regular.

(c) Map $\mathsf{CFI}(G,g)$, $(\mathsf{CFI}(G,0),\leq_0)$, and $(\mathsf{CFI}(G,1),\leq_1)$ to $(\mathsf{CFI}^{\ell+1}(G,g),\trianglelefteq')$, where $\leq_0$ and $\leq_1$ are total orders and $\trianglelefteq'$ individualizes all vertices of the $\ell+1$ many parts of $\mathsf{CFI}(G,0)$ and $\mathsf{CFI}(G,1)$ as well as the join vertices.

(d) Finally, map $(\mathsf{CFI}^{\ell+1}(G,g),\trianglelefteq')$ and $h \in \mathbb{F}_2$ to $(\mathsf{CFI}(\mathsf{CFI}^{\ell+1}(G,g),h),\trianglelefteq)$ such that $\trianglelefteq$ individualizes the required vertices. This map is IFPC-definable since $\mathsf{CFI}^{\ell+1}(G,g)$ is of bounded degree: The graph G has color class size 1 and is 3-regular, so $\mathsf{CFI}(G,h)$ has color class size 4 and degree at most 3. That is, $\mathsf{CFI}^{\ell+1}(G,g)$ has color class size $4\ell+4$ and degree at most $4\ell+4$ (the join vertices), which is a constant.

By composing these IFPC-interpretations, we obtain the required one. We now show that we can simulate each Ψ_j on $(\mathsf{CFI}(\mathsf{CFI}^{\ell+1}(G,g),h),\trianglelefteq)$ in IFPC such that we determine the parity of $\mathsf{CFI}(G,g)$.

For every $\Psi_j(\bar{x}_j)$, we consider all $h \in \mathbb{F}_2$ and all $|\bar{x}_j|$-tuples $\bar{u}$ of $\trianglelefteq$-individualized vertices for the parameters $\bar{x}_j$. We simulate the evaluation of the WSC-fixed-point operator Ψ_j in IFPC as follows: Because Φ is a WSC(IFPC) formula, the step, choice, witnessing, and output formula of Ψ_j are IFPC-formulas. We evaluate the choice formula and check whether all tuples in the defined relation are composed of $\bar{u}$-pebbled-part vertices. If that is the case, we resolve the choice deterministically using the lexicographical order of $\trianglelefteq$ on the tuples (recall that $\trianglelefteq$ individualizes all $\bar{u}$-pebbled-part vertices by construction). Then we evaluate the step formula. The simulation is continued until there is a choice-set not solely composed of vertices of $\bar{u}$-pebbled-part parts. Because the choice-set is by definition of $\mathcal{K}_{\mathrm{orb}}$ an orbit, it is $\bar{u}$-unpebbled distinguishing by Lemma 6.13. So the choice-set contains (at some index) vertices of the $\mathsf{CFI}(G,g)$-parts and either of the $\mathsf{CFI}(G,0)$-parts or of $\mathsf{CFI}(G,1)$-parts: Because isomorphic parts are in the same orbit, either vertices of all isomorphic parts or none of them occur because they cannot be distinguished. At least one of the $\mathsf{CFI}(G,0)$- and $\mathsf{CFI}(G,1)$-parts each is not $\bar{u}$-pebbled because $|\bar{u}| \leq k < \ell+1$. So one of these is isomorphic to the $\mathsf{CFI}(G,g)$-parts. The graphs $\mathsf{CFI}(G,0)$ and $\mathsf{CFI}(G,1)$ were added by the interpretation, so we can actually remember their parity and thus defined the parity of $\mathsf{CFI}(G,g)$.

It remains to prove that such a combination of j, h, and $\bar{u}$ always exists. By construction of $\mathcal{K}_{\mathrm{orb}}$, they exist when testing all possible $|\bar{x}_j|$-tuples for $\bar{u}$ (and not only those of $\trianglelefteq$-individualized vertices). But, because $k < \ell+1$, there is always an automorphism φ (ignoring the individualization) mapping $\bar{u}$ to the vertices of the $\mathsf{CFI}(G,0)$- and $\mathsf{CFI}(G,1)$-parts (because $\mathsf{CFI}(G,g)$ is isomorphic to one of them). Because Ψ_j has no access to the individualization, Ψ_j is satisfied by $\bar{u}$ if and only if Ψ_j is satisfied by $\varphi(\bar{u})$. So indeed IFPC defines the CFI query for $\mathsf{CFI}(\mathcal{K}_{\mathrm{orb}})$.

Finally, for the sake of contradiction, suppose that $\mathcal{K}_{\mathrm{orb}}$ is infinite. So for every k, there is a $j \geq k$ such that $G_j \in \mathcal{K}_{\mathrm{orb}}$ and $\mathsf{CFI}(G_j,0) \equiv^k_{\mathcal{C}} \mathsf{CFI}(G_j,1)$ by Lemma 2.15. This contradicts that IFPC defines the CFI query for $\mathsf{CFI}(\mathcal{K}_{\mathrm{orb}})$. $\dashv$

Claim 2. *$\mathcal{K}_{\text{cfi}}$ is finite.*

Proof. Assume that $\mathcal{K}_{\text{cfi}}$ is infinite. So there is an $\ell' > \ell$ such that $G = G_{\ell'} \in \mathcal{K}_{\text{cfi}}$. By definition of $\mathcal{K}_{\text{cfi}}$, there is a $g \in \mathbb{F}_2$ such that for all $h \in \mathbb{F}_2$, $j \in [m]$, and all $|\bar{x}_j|$-tuples $\bar{u}$ of $\mathsf{CFI}(\mathsf{CFI}^{\ell+1}(G, g), h))$

1. some choice-set during the evaluation of $\Psi_j(\bar{u})$ on $\mathsf{CFI}(\mathsf{CFI}^{\ell+1}(G, g), h)$ is not an orbit or

2. all choice-sets are not $\bar{u}$-unpebbled-part-distinguishing.

We claim that there exists a $\mathcal{P}_\ell$-formula equivalent to Φ on $\mathsf{CFI}(\mathsf{CFI}^{\ell+1}(G, g), 0)$ and $\mathsf{CFI}(\mathsf{CFI}^{\ell+1}(G, g), 1)$. We first translate every WSC-fixed-point operator $\Psi_i(\bar{x}_i)$ for $i \in [m]$ into an equivalent IFPC-formula which uses a fresh relation symbol $\trianglelefteq_P$. The relation $\trianglelefteq_P$ is intended to be interpreted as an $\bar{u}$-pebbled-part individualization when using $\bar{u}$ for the parameters $\bar{x}_i$. Let $i \in [m]$ be arbitrary. Again, the step, choice, witnessing, and output formulas of Ψ_i are IFPC-formulas because Ψ_i is a WSC(IFPC)-formula. We simulate Ψ_i by an IFPC-formula using the relation $\trianglelefteq_P$. If all choice-sets during the evaluation for $\bar{u}$ are not $\bar{u}$-unpebbled-part-distinguishing, then all choice-sets contain solely tuples composed of the vertices individualized by $\trianglelefteq_P$ (otherwise a choice-set would be $\bar{u}$-unpebbled-part-distinguishing by Lemma 6.13). So if all tuples in a choice-set are composed of the individualized vertices, we can resolve all choice deterministically using the lexicographical order on tuples given by $\trianglelefteq_P$. Otherwise, some choice-set during the evaluation will not be an orbit by definition of $\mathcal{K}_{\text{cfi}}$ and we immediately evaluate to false because we make (or will make) a choice out of a non-orbit. If this was never the case, we check in the end whether all choices were indeed witnessed. Let $\tilde{\Psi}_i(\bar{x}_i)$ be an IFPC-formula, which implements exactly this approach to simulate Ψ_i. The formula $\tilde{\Psi}_i(\bar{x}_i)$ can be constructed to use not more than $\ell(k)$ many distinct variables such that no variable is bound twice.

For every number n, every k-variable IFPC-formula not binding variables twice can be unwound into a $\mathcal{C}_k$-formula equivalent on structures of order up to n (see [98]). So, for $\ell = \ell(k)$ and $n = |\mathsf{CFI}(\mathsf{CFI}^{\ell+1}(G, g), 0)|$, we can unwind $\tilde{\Psi}_i(\bar{x}_i)$ to a $\mathcal{C}_\ell$-formula $\tilde{\Psi}_i^n(\bar{x}_i)$. Then the $\mathcal{P}_\ell$-formula

$$\Phi_i(\bar{x}_i) := \exists^P \trianglelefteq_P.\ \tilde{\Psi}_i^n(\bar{x}_i)$$

is equivalent to Ψ_i on $\mathsf{CFI}(\mathsf{CFI}^{\ell+1}(G, g), 0)$ and $\mathsf{CFI}(\mathsf{CFI}^{\ell+1}(G, g), 1)$. To see this, note that $\tilde{\Psi}_i$ evaluates equally for every pebbled-part individualization $\trianglelefteq_P$: The individualization $\trianglelefteq_P$ is only used to resolve choices. If all choice-sets were witnessed as orbits (not stabilizing $\trianglelefteq_P$), then indeed $\tilde{\Psi}_i$ evaluates equally for all $\trianglelefteq_P$ (neither the step, the choice, the witnessing, nor the output formula use $\trianglelefteq_P$). If a choice-set is not witnessed as orbit, this is surely also true for all $\trianglelefteq_P$.

We replace every WSC-fixed-point operator Ψ_i by Φ_i in Φ and continue to unwind the remaining IFPC-part of Φ yielding a $\mathcal{P}_\ell$-formula equivalent to Φ on $\mathsf{CFI}(\mathsf{CFI}^{\ell+1}(G, g), 0)$ and $\mathsf{CFI}(\mathsf{CFI}^{\ell+1}(G, g), 1)$, which by assumption distinguishes the two graphs. This contradicts Lemma 6.15: The ordered graph G has more than ℓ many vertices (and thus color classes), so $\mathsf{CFI}(G, g')$ has also more than ℓ many color classes for every $g' \in \mathbb{F}_2$, and $\mathsf{CFI}(\mathsf{CFI}(G, g'), 0) \equiv_{\mathcal{C}}^{k} \mathsf{CFI}(\mathsf{CFI}(G, g'), 1)$ by Lemma 6.16. So, by Lemma 6.15, Duplicator has a winning strategy in the $\mathcal{P}_\ell$-game played on the CFI graphs $\mathsf{CFI}(\mathsf{CFI}^{k+1}(G, g), 0)$ and $\mathsf{CFI}(\mathsf{CFI}^{k+1}(G, g), 1)$. Hence, $\mathcal{P}_\ell$ does not distinguish the graphs by Lemma 6.14. $\dashv$

Finally, we have to consider the case of free numeric variables. Let the numeric variables in Φ be $\bar{\nu}$. For each numeric variable, there is a closed numeric WSC(IFPC)-term bounding its value. Let these terms be $\bar{s}$. Because we cannot evaluate the WSC(IFPC)-terms, we construct upper-bound-defining IFPC-terms $\bar{t}$ only depending on the size of the input structure. To obtain these, we construct upper-bound-defining terms recursively: For 0, 1, $\cdot$, and $+$, defining the upper bound is obvious. For a counting quantifier $\#\bar{x}\bar{\nu} \leq \bar{s}'. \Phi$, the upper bound is defined by the IFPC-term $(\#\bar{x}. \bar{x} = \bar{x}) \cdot t'_1 \cdot \ldots \cdot t'_{|\nu|}$, where t'_i is the upper-bound-defining IFPC-term recursively obtained for s'_i for every $i \in [|\nu|]$. Note that we do not recurse on Φ and, in particular, not on a WSC-fixed-point operator. For $G \in \mathcal{K}$, set $N(G) := \{0, ..., t_1^{\mathfrak{A}}\} \times \cdots \times \{0, ..., t_{|\bar{\nu}|}^{\mathfrak{A}}\}$ to be the possible values for the numeric variables for $\mathfrak{A} = \mathsf{CFI}(\mathsf{CFI}^{\ell+1}(G, g), h)$ (which only depends on $|A|$).

To partition $\mathcal{K}$ into $\mathcal{K}_{\text{orb}}$ and $\mathcal{K}_{\text{cfi}}$, we not only consider $|\bar{x}_j|$-tuples $\bar{u}$ of vertices of $\mathsf{CFI}(\mathsf{CFI}^{\ell+1}(G, g), h)$ but tuples $\bar{u}\bar{a}$ for $\bar{a} \in N(G)$. To extend Claim 1 to numeric variables, we have to test all possible values for the free numeric variables according to the upper-bound-defining term $\bar{t}$ and find the unpebbled-part-distinguishing choice-set. To adapt the proof of Claim 2, we obtain, for every $i \in [m]$ and every tuple of values $\bar{a} \in N(G)$ for the free numeric variables, a $\mathcal{P}_\ell$-formula $\Phi_i^{\bar{a}}(\bar{x}_i) := \exists^P \trianglelefteq_P. \tilde{\Psi}_i^{n,\bar{a}}(\bar{x}_i)$ satisfying $\bar{u} \in (\Phi_i^{\bar{a}})^{\mathfrak{A}}$ if and only if $\bar{u}\bar{a} \in \Psi_i^{\mathfrak{A}}$ for every $\mathfrak{A} \in \{\mathsf{CFI}(\mathsf{CFI}^{\ell+1}(G, g), h) \mid h \in \mathbb{F}_2\}$. In the same way, free numeric variables of IFPC-formulas are eliminated and we can use these formulas to construct the $\mathcal{P}_\ell$-formula equivalent to Φ. □

Now we can show that we cannot avoid the additional operators to canonize CFI graphs (which implies defining the CFI query) as shown in Corollary 6.8.

Theorem 6.19. *There is a class of base graphs $\mathcal{K}$, such that*

(a) WSCI(IFPC) *defines a canonization for $\mathcal{K}$,*

(b) WSCI(IFPC) *does define the CFI query for $\mathsf{CFI}(\mathcal{K})$, and*

(c) WSCI(WSCI(IFPC)) *defines a canonization for $\mathsf{CFI}(\mathcal{K})$.*

Proof. We consider the class of base graphs $\mathsf{CFI}^\omega(\mathcal{K})$. WSCI^2(IFPC) defines the CFI query for $\mathsf{CFI}(\mathsf{CFI}^\omega(\mathcal{K}))$ by Lemma 6.17. But WSCI(IFPC) does not define the CFI query for $\mathsf{CFI}(\mathsf{CFI}^\omega(\mathcal{K}))$ by Lemma 6.18. □

Note that our proofs only use IFPC-interpretations in interpretation operators.

Corollary 6.20. IFPC $<$ WSCI(IFPC) $<$ WSCI^2(IFPC).

It seems natural that WSCI^n(IFPC) $<$ WSCI^{n+1}(IFPC) for every $n \in \mathbb{N}$. Possibly, this hierarchy can be shown by iterating our construction (e.g., $\mathsf{CFI}((\mathsf{CFI}^\omega)^n(\mathcal{K}))$, where $(\mathsf{CFI}^\omega)^n$ denotes n many applications of the CFI^ω-operator).

We have seen that nesting WSC-fixed-point and interpretation operators increases the expressiveness of IFPC+WSC+I. However, we have not shown that the interpretation operator is necessary for that or whether WSC-fixed-point operators suffice. We will show in the next section that the interpretation operator indeed increases the expressiveness.

6.4 Separating IFPC+WSC from IFPC+WSC+I

In this section we separate IFPC+WSC from IFPC+WSC+I, that is, the interpretation operator strictly increases expressiveness. To do so, we define a class of structures $\mathcal{K}$ without nontrivial automorphisms. Thus, there are only singleton orbits and the WSC-fixed-point operator can be expressed by a (non-WSC) fixed-point operator: Either all choice-sets are singletons or the WSC-fixed-point operator evaluates to false. We show that isomorphism of $\mathcal{K}$-structures is not definable in IFPC and thus not in IFPC+WSC. However, there is an IFPC-interpretation reducing $\mathcal{K}$-isomorphism to the one of CFI graphs (over ordered base graphs) and thus $\mathcal{K}$-isomorphism is IFPC+WSC+I-definable.

We will combine two known constructions. We start with CFI graphs. Then we will modify the CFI graphs to remove all automorphisms. This will be achieved by gluing a CFI graph to a so-called multipede [64]. The multipedes are asymmetric structures, for which IFPC fails to define the orbit partition. To prove that isomorphism of this gluing is not IFPC-definable, either, we will combine winning strategies of Duplicator in the bijective k-pebble game of CFI graphs and multipedes. In order to sucessfully combine the strategies, we will require that the base graphs of the CFI graphs have large connectivity. For the multipedes, we will show the existence of sets of vertices with pairwise large distance. The multipede can be removed by an FO-interpretation reducing the isomorphism problem of the gluing to the CFI query and hence the isomorphism problem is IFPC+WSC+I-definable.

6.4.1 Multipedes

We now review the multipede structures from [64]. These structures are also based on the CFI gadgets. Most importantly, these structures are **asymmetric**, i.e., their only automorphism is the trivial one.

The base graph of a multipede is a bipartite graph $G = (V, W, E, \leq)$, where $\leq$ is some total order on $V \cup W$ and every vertex in V has degree 3. We obtain the **multipede structure** $\mathsf{MP}(G)$ as follows. For every base vertex $\mathfrak{u} \in W$, there is a vertex pair $F(\mathfrak{u}) = \{u_0, u_1\}$ called a **segment**. We also call $\mathfrak{u} \in W$ itself a segment. A single vertex u_i is called a **foot**. Vertices $\mathfrak{v} \in V$ are called **constraint vertices**.

For every constraint vertex $\mathfrak{v} \in V$, a degree-3 CFI gadget with three edge-vertex-pairs is added. We use the relational CFI gadgets only with edge vertices (cf. Section 2.8.4). Let $N_G(\mathfrak{v}) = \{\mathfrak{u}^1, \mathfrak{u}^2, \mathfrak{u}^3\}$ (the order of the $\mathfrak{u}^i$ is given by $\leq$). The edge vertices are $\{(\mathfrak{v}, \mathfrak{u}^i, b)\}$ for $i \in [3]$ and $b \in \mathbb{F}_2$. Then $(\mathfrak{v}, \mathfrak{u}^i, b)$ is identified with the foot u^j_i for all $j \in [3]$ and $i \in \mathbb{F}_2$. That is, unlike for CFI graphs, a foot belongs to more than two gadgets. Because we use the relation-based CFI gadgets, we do not add further vertices to the feet but a ternary relation R containing all triples $(u^1_{i_1}, u^2_{i_2}, u^3_{i_3})$ with $i_1 + i_2 + i_3 = 0$. Because all constraint vertices in V have degree 3, the construction yields ternary structures of the fixed signature $\{R, \preceq\}$. The coloring $\preceq$ is again obtained from $\leq$ (the feet of a segment have the color of the segment). We collect properties of multipedes.

Definition 6.21 (Odd Graph). A bipartite graph $G = (V, W, E, \leq)$ is **odd** if, for every $\emptyset \neq X \subseteq W$, there exists a $\mathfrak{v} \in V$ such that $|X \cap N_G(\mathfrak{v})|$ is odd.

Lemma 6.22 ([64]). *If G is an odd and ordered bipartite graph, then* $\mathsf{MP}(G)$ *is asymmetric.*

Definition 6.23 (k-Meager). A bipartite graph $G = (V, W, E, \leq)$ is called **k-meager** if every set $X \subseteq W$ of size $|X| \leq 2k$ satisfies $|\{\mathfrak{v} \in V \mid N_G(\mathfrak{v}) \subseteq X\}| \leq 2|X|$.

Intuitively, if G is a k-meager bipartite graph, $\mathcal{C}_k$ cannot distinguish between the two feet of a segment in the structure $\mathsf{MP}(G)$. For a set $X \subseteq W$, we define the **feet-induced** subgraph

$$G[[X]] := G[X \cup \{\, \mathfrak{v} \in V \mid N_G(\mathfrak{v}) \subseteq X \,\}]$$

to be the induced subgraph by the feet in X and all constraint vertices only adjacent to feet in X. We extend the notation to the multipede: $\mathsf{MP}(G)[[X]]$ is the induced substructure of all feet whose segment is contained in X. For a tuple $\bar{u}$ of feet of $\mathsf{MP}(G)$, we define

$$S(\bar{u}) := \{\, \mathfrak{u} \in W \mid u_i \in F(\mathfrak{u}) \text{ for some } i \leq |\bar{u}| \,\}$$

to be the set of the segments of the feet in $\bar{u}$.

Lemma 6.24 ([64]). *Let G be a k-meager bipartite graph, $\mathfrak{A} = \mathsf{MP}(G)$, and $\bar{u}, \bar{v} \in A^k$. If there is a local automorphism $\varphi \in \mathsf{Aut}(\mathfrak{A}[[S(\bar{u}\bar{v})]])$ with $\varphi(\bar{u}) = \bar{v}$, then $(\mathfrak{A}, \bar{u}) \equiv_{\mathcal{C}}^{k} (\mathfrak{A}, \bar{v})$.*

It was shown [64] that meager bipartite graphs exists. However, we need a more detailed understanding of these graphs for our construction. In particular, we are interested in sets of segments which have pairwise large distance.

Definition 6.25 (k-Scattered). For a bipartite graph $G = (V, W, E)$, a set $X \subseteq W$ is **k-scattered** if every distinct $\mathfrak{u}, \mathfrak{v} \in X$ have distance at least $2k$ in G.

We require pairwise distance $2k$ for a k-scattered set X because we are actually only interested in segments (which alternate with constraint vertices in paths in G).

Lemma 6.26. *For every number $k \in \mathbb{N}$, there exists an odd and k-meager bipartite graph $G = (V, W, E)$ for which there is a k-scattered set $X \subseteq W$ of size $|X| \geq k^2$.*

Proof. The multipedes constructed in [64] are sparse graphs generated by a random process. Fix an arbitrary $k \in \mathbb{N}$. We start with a vertex set U of size n. An event $E(n)$ is called **almost sure** if the probability that E occurs tends to 1 when n grows to infinity. Pick $\varepsilon < (2k+3)^{-1}$ and add independently with probability $p = n^{-2+\varepsilon}$, for every 3-element subset $\{\mathfrak{u}_1, \mathfrak{u}_2, \mathfrak{u}_3\} \subseteq U$, a vertex $\mathfrak{v}$ to V adjacent to the $\mathfrak{u}_i$. Then we can almost surely remove at most $n/4$ many vertices from U forming subgraphs that are exceedingly dense (and all constraint vertices in $\mathfrak{v}$ incident to them) and obtain an odd and k-meager bipartite graph [64]. We show that in this graph a k-scattered set X of size k^2 exists almost surely.

Claim 1. *Almost surely, every vertex $\mathfrak{u} \in W$ has degree at most $n^{1.5k^{-1}}$.*

Proof. Let m be the least integer such that $m > n^{(1.5k)^{-1}}$. The probability that a given vertex has degree larger than $n^{(1.5k)^{-1}}$ is $\binom{n^2}{m} \cdot p^m$. So the probability that some vertex has degree larger than $n^{(1.5k)^{-1}}$ is at most the following:

$$\begin{aligned}
n \cdot \binom{n^2}{m} \cdot p^m &= n \cdot \binom{n^2}{m} \cdot (n^{-2+\varepsilon})^m \\
&\leq \binom{n^2}{m} \cdot n^{(-2+(2k+3)^{-1})m+1} \\
&\leq \left(\frac{en^2}{m}\right)^m \cdot n^{(-2+(2k+3)^{-1})m+1} \\
&\leq \left(en^{2-(1.5k)^{-1}}\right)^m \cdot n^{(-2+(2k+3)^{-1})m+1} \\
&= e^m \cdot n^{(2-(1.5k)^{-1})m} \cdot n^{(-2+(2k+3)^{-1})m+1} \\
&= e^m \cdot n^{((2k+3)^{-1}-(1.5k)^{-1})m+1} \\
&= n^{(\ln n)^{-1}\cdot m} \cdot n^{((2k+3)^{-1}-(1.5k)^{-1})m+1} \\
&= n^{((2k+3)^{-1}-(1.5k)^{-1}+(\ln n)^{-1})m+1} \\
&\leq n^{((2k+3)^{-1}-(1.5k)^{-1}+(\ln n)^{-1})(n^{(1.5k)^{-1}}+1)+1} = o(1).
\end{aligned}$$

The last step holds because $(2k+3)^{-1} - (1.5k)^{-1} < 0$ and thus for sufficiently large n it holds that $\ell := (2k+3)^{-1} - (1.5k)^{-1} + (\ln n)^{-1} < 0$ and $\ell(n^{(1.5k)^{-1}}+1)+1$ tends to $-\infty$. $\dashv$

Claim 2. *If every vertex $\mathfrak{u} \in W$ has degree at most $n^{(1.5k)^{-1}}$, then for each vertex $\mathfrak{u} \in W$ at most $3^k n^{\frac{2}{3}}$ many vertices $\mathfrak{u}' \in W$ have distance at most $2k$ to $\mathfrak{u}$.*

Proof. By construction, every vertex in V has degree 3. So at most $3n^{(1.5k)^{-1}}$ vertices have distance 2 to $\mathfrak{u}$. Repeating the argument, at most $(3n^{(1.5k)^{-1}})^k = 3^k n^{\frac{2}{3}}$ vertices in W have distance $2k$ to $\mathfrak{u}$. $\dashv$

Almost surely every vertex $\mathfrak{u} \in W$ has degree at most $n^{(1.5k)^{-1}}$ (Claim 1). We show that in this case a k-scattered set X of size at least k^2 exists. Note that we determined the probability before removing the $\frac{n}{4}$ "bad" vertices. So we first remove some set of size at most $\frac{n}{4}$ from U (which only decreases the degree of the remaining vertices). We now repeatedly apply Claim 2. If we have a k-scattered set X, then at most $|X| \cdot 3^k n^{\frac{2}{3}}$ many vertices have distance at most $2k$ to a vertex in X. Then pick one of the other vertices, add it to X, so X is still k-scattered, and repeat. In this way, we find a k-scattered set X of size $|X| \geq \frac{3}{4}n \cdot (3^k n^{\frac{2}{3}})^{-1} = \frac{3}{4}3^{-k}n^{\frac{1}{3}}$. Finally, for sufficiently large n we have that $k^2 \leq \frac{3}{4}3^{-k}n^{\frac{1}{3}}$. $\square$

We want to use a k-scattered set X to ensure that in the bijective k-pebble game placing a pebble on one foot of a segment in X has no effect on the other segments in X. To make this argument formal, we need to consider how information of pebbles is spread through the multipedes. For now, fix an arbitrary bipartite graph $G = (V, W, E, \leq)$.

Definition 6.27 (Closure). Let $X \subseteq W$. The **attractor** of X is

$$\mathsf{attr}(X) := X \cup \bigcup_{\substack{u \in V, \\ |N_G(u) \setminus X| \leq 1}} N_G(u).$$

The set X is **closed** if $X = \mathsf{attr}(X)$. The **closure** $\mathsf{cl}(X)$ of X is the inclusion-wise minimal closed superset of X.

Lemma 6.28 ([64]). *If G is k-meager and $X \subseteq W$ of size at most k, then $|\mathsf{cl}(X)| \leq 2|X|$.*

Let X be closed. A set $Y \subseteq X$ is a **component** of X if $G[[Y]]$ is a connected component of $G[[X]]$. That is, every constraint vertex contained in $G[[X]]$ is contained in $G[[Y]]$ or in $G[[X \setminus Y]]$.

Lemma 6.29. *Let $X \subseteq W$. If $\mathsf{cl}(X) = Y_1 \cup Y_2$ is the disjoint union of two components Y_1 and Y_2, then X can be partitioned into $X_1 \cup X_2$ such that $\mathsf{cl}(X_i) = Y_i$ for every $i \in [2]$.*

Proof. Define $X_i := Y_i \cap X$ for every $i \in [2]$. Let $i \in [2]$ be arbitrary. We show that $\mathsf{cl}(X_i) = Y_i$. It is clear that $\mathsf{cl}(X_i) \subseteq Y_i$. For the other direction, suppose that $\mathfrak{u} \in Y_i \setminus \mathsf{cl}(X_i)$. Then there must be a constraint vertex in $G[[Y]]$ which has a neighbor in Y_1 and another one in Y_2. This contradicts that the Y_i are components of $\mathsf{cl}(X)$. □

Lemma 6.30. *Let $Y \subseteq W$ be closed and $\mathfrak{u} \in W$ have distance at least 4 to Y. Then $\mathsf{cl}(Y \cup \{\mathfrak{u}\}) = Y \cup \{\mathfrak{u}\}$ and $\mathfrak{u}$ forms a singleton component of $Y \cup \{\mathfrak{u}\}$.*

Proof. Suppose for the sake of contradiction, that there exists a $\mathfrak{v} \in \mathsf{cl}(Y \cup \{\mathfrak{u}\}) \setminus (Y \cup \{\mathfrak{u}\})$. Then there is a constraint vertex $\mathfrak{w}$, of which one neighbor is $\mathfrak{v}$ and the two others are contained in $\mathsf{cl}(Y \cup \{\mathfrak{u}\})$. If the two other neighbors are contained in $\mathsf{cl}(Y)$, then $\mathfrak{v}$ is already contained $\mathsf{cl}(Y)$, which is a contradiction. So one of the two neighbors is $\mathfrak{u}$. But then $\mathfrak{u}$ has distance 2 to Y, which contradicts our assumption. □

Lemma 6.31. *Let $k \geq 2$, G be $2k$-meager, $Y \subseteq W$ be of size at most k, and $X \subseteq W$ be $6k$-scattered. Then at most $|Y|$ many vertices of X do not form singleton components in $\mathsf{cl}(X \cup Y)$.*

Proof. Because $X \subseteq W$ is $6k$-scattered, every distinct $\mathfrak{u}, \mathfrak{v} \in X$ have distance at least $12k$ in G. Hence, for every $\mathfrak{w} \in Y$, there is at most one $\mathfrak{u} \in X$ such that $\mathfrak{w}$ has distance less than $6k$ to $\mathfrak{u}$. Let $X' \subseteq X$ be the set vertices with distance at most $6k$ to Y and thus $|X'| \leq |Y|$. So, $|Y \cup X'| \leq 2k$ and thus $|\mathsf{cl}(Y \cup X')| \leq 4k$ by Lemma 6.28 because G is $2k$-meager. It follows that $|\mathsf{cl}(Y \cup X') \setminus Y \setminus X'| \leq 2k$. Because every component of $\mathsf{cl}(Y \cup X')$ contains a vertex of $Y \cup X'$, every vertex $\mathsf{cl}(Y \cup X')$ has distance at most $4k$ to every vertex in $Y \cup X'$. Every vertex $\mathfrak{w} \in X \setminus X'$ has distance at least $6k$ to Y and distance at least $12k$ to X'. So every vertex $\mathfrak{w} \in X \setminus X'$ has distance at least $2k \geq 4$ to $\mathsf{cl}(Y \cup X')$. That is, all vertices in $X \setminus X'$ form singleton components of $\mathsf{cl}(X \cup Y)$ by Lemma 6.30. □

Lemma 6.32. *Let $k \geq 2$, G be $2k$-meager, $X \subseteq W$ be $6k$-scattered, and let $Y \subseteq W$ be of size at most k. Then $X \cup Y$ can be partitioned into $Z_1, \dots, Z_\ell$ such that $|\mathsf{cl}(Z_i) \cap X| \leq 1$ for every $i \in [\ell]$ and $\mathsf{cl}(Z_1), \dots, \mathsf{cl}(Z_\ell)$ are the components of $\mathsf{cl}(X \cup Y)$.*

Proof. We partition $X \cup Y$ using Lemma 6.29 into $Z_1, \dots, Z_\ell$ such that $\mathsf{cl}(Z_1), \dots, \mathsf{cl}(Z_\ell)$ are precisely the components of $\mathsf{cl}(X \cup Y)$. Let $i \in [\ell]$ be arbitrary but fixed. We prove that $|\mathsf{cl}(Z_i) \cap X| \leq 1$. By Lemma 6.31, all apart from $|Y|$ many vertices in X are contained in a singleton component. If $\mathsf{cl}(Z_i)$ is such a singleton component, we are done. Otherwise,

$$|Z_i| \leq |Z_i \cap X| + |Y| \leq 2|Y| \leq 2k$$

and $|\mathsf{cl}(Z_i)| \leq 4k$ by Lemma 6.28 and because G is $2k$-meager. Because $\mathsf{cl}(Z_i)$ is a component, all vertices in $\mathsf{cl}(Z_i)$ have pairwise distance at most $8k$. So $|\mathsf{cl}(Z_i) \cap X| \leq 1$, because X is $6k$-scattered (and thus X-vertices have distance at least $12k$). □

Lemma 6.33 ([64]). *Let $X \subseteq W$, G be k-meager, $\varphi \in \mathsf{Aut}(\mathsf{MP}(G)[[X]])$, and $|X| \leq k$. Then φ extends to an automorphism of $\mathsf{MP}(G)[[\mathsf{cl}(X)]]$.*

Lemma 6.34. *Let $Y \subseteq W$, G be k-meager, $\varphi \in \mathsf{Aut}(\mathsf{MP}(G)[[\mathsf{cl}(Y)]])$, and $|Y| < k$. Then for every $\mathfrak{u} \in W \setminus \mathsf{cl}(Y)$ and every pair of feet $v, v' \in F(\mathfrak{u})$ (possibly $v = v'$), there is an extension ψ of φ to an automorphism of $\mathsf{MP}(G)[[\mathsf{cl}(Y \cup \{\mathfrak{u}\})]]$ satisfying $\psi(v) = v'$.*

Proof. The condition $\psi(v) = v'$ defines ψ uniquely on the feet of $\mathfrak{u}$, that is, given φ, v, and v' the map ψ is determined. If $v = v'$, then the ψ maps the feet of $\mathfrak{u}$ to itself and otherwise it exchanges them. Using Lemma 6.33 and noting that $|Y \cup \{\mathfrak{u}\}| \leq k$, it suffices to show that ψ is an automorphism of $\mathsf{MP}(G)[[Y \cup \{\mathfrak{u}\}]]$. Because $\mathfrak{u} \notin \mathsf{cl}(Y)$, there is no CFI gadget whose one edge-vertex-pair are the feet of $\mathfrak{u}$ and the other two are contained in $\mathsf{cl}(Y)$ (otherwise $\mathfrak{u}$ would be in the closure). So indeed, ψ is a local automorphism. □

Lemma 6.35. *Let $k \geq 2$, G be $2k$-meager, $X = \{\mathfrak{u}_1, \dots, \mathfrak{u}_\ell\} \subseteq W$ be $6k$-scattered, $Y \subseteq W$ such that $X \cap \mathsf{cl}(Y) = \emptyset$ and $|Y| < k$, and $\varphi \in \mathsf{Aut}(\mathsf{MP}(G)[[Y]])$. Then for every pair of tuples $\bar{u}, \bar{u}' \in F(\mathfrak{u}_1) \times \cdots \times F(\mathfrak{u}_\ell)$, there is an extension ψ of φ to and automorphism of $\mathsf{MP}(G)[[X \cup Y]]$ satisfying $\psi(\bar{u}) = \bar{u}'$.*

Proof. Let $\bar{u}, \bar{u}' \in F(\mathfrak{u}_1) \times \cdots \times F(\mathfrak{u}_\ell)$. For every $i \in [\ell]$, there is by Lemma 6.34 an extension ψ_i of φ to $\mathsf{MP}(G)[[\mathsf{cl}(Y \cup \{\mathfrak{u}_i\})]]$ satisfying $\psi_i(u_i) = u_i'$. By Lemma 6.32, all $\mathfrak{u}_i$ are in different components of $\mathsf{cl}(Y \cup X)$ because G is $2k$-meager and X is $6k$-scatted. So we can extend every ψ_i on all components of $\mathsf{cl}(Y \cup X)$, on which ψ_i is not defined (in particular the ones containing all other $\mathfrak{u}_j$ for $j \neq i$), by the identity map and obtain an automorphism of $\mathsf{MP}(G)[[\mathsf{cl}(Y \cup X)]]$ (two components are never connected by a CFI gadget). Hence, composing all the extended ψ_i yields the desired automorphism. □

The previous lemma will be extremely useful in the bijective k-pebble game: If the pebbles are placed on the feet in Y, then we can simultaneously for all feet in $X \setminus \mathsf{cl}(Y)$ place arbitrary pebbles and still maintain a local automorphism. Such sets X will allow us to glue another graph to the multipede at the feet in X. Whatever restrictions on placing pebbles are imposed by the other graph, we still can maintain local automorphisms in the multipede.

Lemma 6.36. *Let $k \geq 2$, G be $2k$-meager, $X \subseteq W$ be $6k$-scattered, and $Y \subseteq W$ be of size at most k. Then $|\mathsf{cl}(Y) \cap (X \setminus Y)| \leq |Y \setminus X|$.*

Proof. We partition $X \cup Y$ using Lemma 6.32 into $Z_1, \dots, Z_j$ such that the $\mathsf{cl}(Z_i)$ are the components of $\mathsf{cl}(X \cup Y)$ and at most one vertex of X is contained in one Z_i. Up to reordering, assume that for some $\ell \leq j$ the components $Z_1, \dots, Z_\ell$ are all components Z_i such that $Z_i \cap Y \neq \emptyset$ and for some $m \leq \ell$ the $Z_1, \dots, Z_m$ are all Z_i such that additionally $\mathsf{cl}(Z_i) \cap (X \setminus Y) \neq \emptyset$.

Then clearly $\mathsf{cl}(Y) \subseteq \bigcup_{i \in [\ell]} \mathsf{cl}(Z_i)$ and $\ell \leq |Y|$. Because G is $2k$-meager, X is $6k$-scattered, and $|Y| \leq k$, it follows from Lemma 6.32 that $|\mathsf{cl}(Z_i) \cap X| \leq 1$ for every $i \in [\ell]$. Thus, $|\mathsf{cl}(Z_i) \cap (X \setminus Y)| = 1$ for every $i \in [m]$ and $|\mathsf{cl}(Z_i) \cap (X \setminus Y)| = 0$ for every $m < i \leq \ell$. It follows that $\mathsf{cl}(Y) \cap (X \setminus Y) \subseteq \bigcup_{i=1}^{m} \mathsf{cl}(Z_i) \cap (X \setminus Y)$. Hence, $|\mathsf{cl}(Y) \cap (X \setminus Y)| \leq m$. Every vertex in $Y \cap X$ has to be contained in some Z_i such that $m < i \leq \ell$ because, for every $i \leq m$, we have $|\mathsf{cl}(Z_i) \cap X| \leq 1$ and so $Z_i \cap (Y \cap X) = \emptyset$ since Z_i contains a vertex of $X \setminus Y$. But for $i \leq \ell$, every Z_i contains at least one vertex of Y. That is, $m \leq |Y \setminus X|$. □

6.4.2 Gluing Multipedes to CFI Graphs

We now glue CFI graphs to multipedes. First, we alter the used CFI-construction. Instead of two edge-vertex-pairs for the same base edge $\{\mathfrak{u}, \mathfrak{v}\}$ (one with origin $(\mathfrak{u}, \mathfrak{v})$ and one with origin $(\mathfrak{v}, \mathfrak{u})$), we contract the edges between these two edge-vertex-pairs (which form a matching) and obtain a single edge-vertex-pair with origin $\{\mathfrak{u}, \mathfrak{v}\}$. This preserves all relevant properties of CFI graphs. In particular, there are parity-preserving IFPC-interpretations mapping a CFI graphs with two edge-vertex-pair per base edge to the corresponding CFI graphs with only one edge-vertex-pair per base edge and vice versa. In this section we refer with $\mathsf{CFI}(H, f)$ to CFI graphs of this modified construction. Using only one edge-vertex-pair per base edge removes technical details from the following.

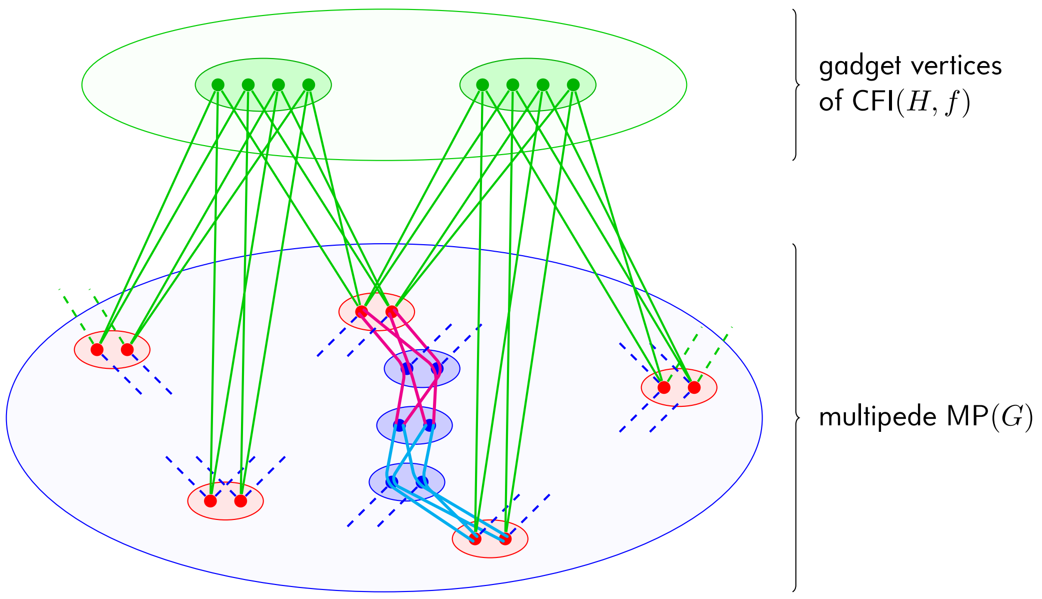
Let $G = (V^G, W^G, E^G, \leq^G)$ be an ordered bipartite graph, let $H = (V^H, E^H, \leq^H)$ be an ordered base graph, let $f\colon E^H \to \mathbb{F}_2$, and let $X \subseteq W^G$ have size $|X| = |E^H|$. We define the ternary structure $\mathsf{MP}(G) \cup_X \mathsf{CFI}(H, f)$ called the **gluing** of the multipede $\mathsf{MP}(G) = (A, R^{\mathsf{MP}(G)}, \preceq^{\mathsf{MP}(G)})$ and the CFI graph $\mathsf{CFI}(H, f) = (B, E^{\mathsf{CFI}(H,f)}, \preceq^{\mathsf{CFI}(H,f)})$ at X as follows (see Figure 6.4 for an illustration): The i-th edge-vertex-pair of $\mathsf{CFI}(H, f)$ is the edge-vertex-pair such that its origin $\{\mathfrak{u}, \mathfrak{v}\}$ is the i-th edge in H according to $\leq^H$. We start with the disjoint union of $\mathsf{MP}(G)$ and $\mathsf{CFI}(H, f)$ and identify the i-th edge-vertex-pair of $\mathsf{CFI}(H, f)$ with the i-th segment in X (according to $\leq^G$). We finally turn the edges $E^{\mathsf{CFI}(H,f)}$ into a ternary relation by extending every edge $(u, v) \in E^{\mathsf{CFI}(H,f)}$ to the triple (u, v, v). In this way, we obtain the $\{R, \preceq\}$-structure $\mathsf{MP}(G) \cup_X \mathsf{CFI}(H, f)$, where R is the union of $R^{\mathsf{MP}(G)}$ with the triples (u, v, v) defined before and $\preceq$ is the total preorder obtained from joining $\preceq^{\mathsf{MP}(G)}$ and $\preceq^{\mathsf{CFI}(H,f)}$ by moving all gadget vertices of $\mathsf{CFI}(H, f)$ to the end.

Lemma 6.37. *If* $\mathsf{MP}(G)$ *is asymmetric, then* $\mathsf{MP}(G) \cup_X \mathsf{CFI}(H, f)$ *is asymmetric.*

Proof. Every automorphism of $\mathsf{MP}(G) \cup_X \mathsf{CFI}(H, f)$ is, in particular, an automorphism of $\mathsf{CFI}(H, f)$. Every nontrivial automorphism of $\mathsf{CFI}(H, f)$ exchanges the two vertices of some edge-vertex-pairs because H is totally ordered. Because every edge-vertex-pair is identified with a segment of $\mathsf{MP}(G)$ and $\mathsf{MP}(G)$ is asymmetric, $\mathsf{MP}(G) \cup_X \mathsf{CFI}(H, f)$ is asymmetric, too. □

We now show that $\mathsf{MP}(G) \cup_X \mathsf{CFI}(H, 0) \equiv_C^k \mathsf{MP}(G) \cup_X \mathsf{CFI}(H, 1)$ if G and H satisfy certain conditions. Let $\bar{u}$ be a vertex-tuple of $\mathsf{MP}(G) \cup_X \mathsf{CFI}(H, f)$, i.e., $\bar{u}$ contains either gadget vertices of the gadgets in $\mathsf{CFI}(H, f)$ or feet of $\mathsf{MP}(G)$. We also write $S(\bar{u})$ for the set of segments of all feet in $\bar{u}$.

1. The segments $S(\bar{u})$ are **directly-fixed by** $\bar{u}$.
2. The segments $\mathsf{cl}(S(\bar{u})) \setminus S(\bar{u})$ are **closure-fixed by** $\bar{u}$.
3. A segment $\mathfrak{u} \in X$ is **gadget-fixed by** $\bar{u}$ if the feet of $\mathfrak{u}$ are identified with some edge-vertex-pair with origin $\{\mathfrak{v}, \mathfrak{w}\}$ in $\mathsf{CFI}(H, f)$ such that there is a gadget vertex with origin $\mathfrak{v}$ or $\mathfrak{w}$ in $\bar{u}$.
4. A segment is **fixed by** $\bar{u}$ if it is directly fixed, closure-fixed, or gadget-fixed by $\bar{u}$.



6.4 Gluing multipedes to CFI graphs. The figure shows the gluing $\mathsf{MP}(G) \cup_X \mathsf{CFI}(H, f)$ of the multipede $\mathsf{MP}(G)$ and the CFI graph $\mathsf{CFI}(H, f)$ at the set of segments X. The mutipede is drawn in blue, the segments in X are drawn in red, and the gadget vertices of the CFI graph are drawn in green. Only some segments and CFI gadgets are shown. Two relational CFI gadgets of the multipede are shown, where different colors are used for each gadget. The edge-vertex-pairs of the CFI graphs are identified with the vertex-pairs of the segments in X. Formally, every vertex pair of each segment has a unique color and there is only a single ternary relation.

Lemma 6.38. *Let $r \geq k \geq 2$ and $\bar{u}$ be a vertex-tuple of $\mathsf{MP}(G) \cup_X \mathsf{CFI}(H, f)$ of length at most k. If H is r-regular, G is $2k$-meager, and X is $6k$-scattered, then at most $r \cdot |\bar{u}|$ many segments are fixed by $\bar{u}$. If $\bar{u}$ contains i many gadget vertices and ℓ many segments in X are directly-fixed by $\bar{u}$, then at most $|\bar{u}| - i - \ell$ many segments in X are closure-fixed by $\bar{u}$.*

Proof. Assume that $\bar{u}$ contains j many feet and i many gadget vertices. So, $i+j = |\bar{u}| \leq k$. Then at most ri many segments are gadget-fixed by $\bar{u}$ because G is r-regular. From Lemma 6.28 it follows that $|\mathsf{cl}(S(\bar{u}))| \leq 2j$ because G is k-meager. So at most $2j$ many segments are directly-fixed or closure-fixed. Then $ri + 2j \leq ri + rj = r(i + j) = r|\bar{u}|$ because $i + j = |\bar{u}|$ and $r \geq k \geq 2$.

By Lemma 6.36, it holds that $|\mathsf{cl}(S(\bar{u})) \cap (X \setminus S(\bar{u}))| \leq |S(\bar{u}) \setminus X|$ because G is k-meager, X is $6k$-scattered, and $|S(\bar{u})| \leq k$. That is, the number of closure-fixed segments in X is bounded by the number of directly-fixed segments not in X. The number of directly-fixed segments not in X is $|\bar{u}| - i - \ell$. □

We now combine winning strategies of Duplicator in the bijective pebble game on multipedes and CFI graphs:

Lemma 6.39. *Let $r \geq k \geq 3$, G be $2rk$-meager, H be r-regular and at least $(k + 2)$-connected, and X be $6rk$-scattered. Then $\mathsf{MP}(G) \cup_X \mathsf{CFI}(H, 0) \equiv_{\mathcal{C}}^{k} \mathsf{MP}(G) \cup_X \mathsf{CFI}(H, 1)$.*

Proof. Let $\mathfrak{A} = \mathsf{MP}(G)$, $\mathfrak{B} = \mathsf{CFI}(H, 0)$, and $\mathfrak{B}' = \mathsf{CFI}(H, 1)$. We show that Duplicator has a winning strategy in the bijective k-pebble game played on $\mathfrak{A} \cup_X \mathfrak{B}$ and $\mathfrak{A} \cup_X \mathfrak{B}'$. For a set of segments Y and a tuple $\bar{u}$, we denote by $\bar{u}_Y$ the restriction of $\bar{u}$ to all feet whose segment is contained in Y, by $\bar{u}_\mathrm{G}$ the restriction of $\bar{u}$ to all gadget vertices, and by $\bar{u}_\mathrm{F}$ the restriction of $\bar{u}$ to all feet. Duplicator maintains the following invariant. At every position $(\mathfrak{A} \cup_X \mathfrak{B}, \bar{u}; \mathfrak{A} \cup_X \mathfrak{B}', \bar{u}')$ in the game there exist tuples $\bar{v}_\mathrm{gf}, \bar{v}_\mathrm{cf}$ of $\mathfrak{A} \cup_X \mathfrak{B}$ and $\bar{v}'_\mathrm{gf}, \bar{v}'_\mathrm{cf}$ of $\mathfrak{A} \cup_X \mathfrak{B}'$ satisfying the following:

1. Exactly one foot of exactly all the segments gadged-fixed by $\bar{u}$ (and so by $\bar{u}'$) is contained in $\bar{v}_\mathrm{gf}$ and in $\bar{v}'_\mathrm{gf}$.
2. Exactly one foot of exactly all the segments contained in X and closure-fixed by $\bar{u}$ (and so by $\bar{u}'$) is contained in $\bar{v}_\mathrm{cf}$ and in $\bar{v}'_\mathrm{cf}$.
3. There is a $\varphi \in \mathsf{Aut}(\mathfrak{A}[[S(\bar{u}_\mathrm{F}\bar{v}_\mathrm{gf}\bar{v}_\mathrm{cf})]])$ satisfying $\varphi(\bar{u}_\mathrm{F}\bar{v}_\mathrm{gf}\bar{v}_\mathrm{cf}) = \bar{u}'_\mathrm{F}\bar{v}'_\mathrm{gf}\bar{v}'_\mathrm{cf}$.
4. $(\mathfrak{B}, \bar{u}_X\bar{u}_\mathrm{G}\bar{v}_\mathrm{cf}) \equiv^k_\mathcal{C} (\mathfrak{B}', \bar{u}'_X\bar{u}'_\mathrm{G}\bar{v}'_\mathrm{cf})$.
5. For every base vertex $\mathfrak{u} \in V^H$, it holds that $(\mathfrak{B}', \bar{u}_\mathrm{G}\bar{v}_\mathrm{gf})[V_\mathfrak{u}] \cong (\mathfrak{B}, \bar{u}'_\mathrm{G}\bar{v}'_\mathrm{gf})[V_\mathfrak{u}]$, where $V_\mathfrak{u}$ is the set of all gadget vertices with origin $\mathfrak{u}$ and all edge vertices with origin $\{\mathfrak{u}, \mathfrak{v}\}$ for some $\mathfrak{v} \in N_G(\mathfrak{u})$.

Regarding Property 3, note that $S(\bar{u}_\mathrm{F}\bar{v}_\mathrm{gf}\bar{v}_\mathrm{cf}) = S(\bar{u}'_\mathrm{F}\bar{v}'_\mathrm{gf}\bar{v}'_\mathrm{cf})$ and $|\bar{u}_\mathrm{F}\bar{v}_\mathrm{gf}\bar{v}_\mathrm{cf}| = |\bar{u}'_\mathrm{F}\bar{v}'_\mathrm{gf}\bar{v}'_\mathrm{cf}| \leq rk$ by Lemma 6.38. Hence, the local automorphism φ extends to the closure of $S(\bar{u}_\mathrm{F}\bar{v}_\mathrm{gf})$ by Lemma 6.33 because G is rk-meager. Regarding Property 4, we argue that we have $|\bar{u}_X\bar{u}_\mathrm{G}\bar{v}_\mathrm{cf}| = |\bar{u}'_X\bar{u}'_\mathrm{G}\bar{v}'_\mathrm{cf}| \leq k$: By Lemma 6.38, the number of closure-fixed segments in X is at most $|\bar{v}_\mathrm{cf}| \leq k - |\bar{u}_\mathrm{G}| - |\bar{u}_X|$. Hence, $|\bar{u}_X\bar{u}_\mathrm{G}\bar{v}_\mathrm{cf}| = |\bar{u}_X| + |\bar{u}_\mathrm{G}| + |\bar{v}_\mathrm{cf}| \leq k$. (Note that $|\bar{u}_X| = |\bar{u}'_X|$, $|\bar{u}_\mathrm{G}| = |\bar{u}'_\mathrm{G}|$, etc., because otherwise Duplicator has already lost the game). Property 5 is needed because $|\bar{v}_\mathrm{gf}| \leq rk$ (possibly with equality), which exceeds k. Thus, Property 5 is not implied by Property 4. Property 5 guarantees that we pick the vertices $\bar{v}_\mathrm{gf}$ and $\bar{v}'_\mathrm{gf}$ consistently.

Intuitively, we want to play two games. Game I is played with rk many pebbles on the multipede at position $(\mathfrak{A}, \bar{u}_\mathrm{F}\bar{v}_\mathrm{gf}\bar{v}_\mathrm{cf}; \mathfrak{A}, \bar{u}'_\mathrm{F}\bar{v}'_\mathrm{gf}\bar{v}'_\mathrm{cf})$. Game II is played with k many pebbles on the CFI graphs at position $(\mathfrak{B}, \bar{u}_X\bar{u}_\mathrm{G}\bar{v}_\mathrm{cf}; \mathfrak{B}', \bar{u}'_X\bar{u}'_\mathrm{G}\bar{v}'_\mathrm{cf})$. We use the winning strategy of Duplicator in both games (Lemmas 2.15 and 6.24) to construct a winning strategy in the bijective k-pebble game played at position $(\mathfrak{A} \cup_X \mathfrak{B}, \bar{u}; \mathfrak{A} \cup_X \mathfrak{B}', \bar{u}')$. Intuitively, we can do so because in Game I we artificially fixed all gadget-fixed segments and in Game II we artificially fixed all closure-fixed segments in X (and only the segments in X are identified with edge-vertex-pairs of the CFI graphs).

Now assume that it is Spoiler's turn. When Spoiler picks up a pair of pebbles (p_i, q_i) from the structures, we first update the tuples $\bar{v}_\mathrm{gf}$, $\bar{v}'_\mathrm{gf}$, $\bar{v}_\mathrm{cf}$, and $\bar{v}'_\mathrm{cf}$: If a segment is no longer gadget-fixed or closure-fixed, then we remove the corresponding entries in the corresponding tuples. Clearly the invariant is maintained. We describe how Duplicator defines a bijection λ between $\mathfrak{A} \cup_X \mathfrak{B}$ and $\mathfrak{A} \cup_X \mathfrak{B}'$ by defining $\lambda(w)$ using the following case distinction:

(a) Assume the vertex w is a foot whose segment is contained in $\mathsf{cl}(S(\bar{u}_\mathrm{F}\bar{v}_\mathrm{gf}\bar{v}_\mathrm{cf}))$. The automorphism φ from Property 3 extends to an automorphism of $\mathfrak{A}[[\mathsf{cl}(S(\bar{u}_\mathrm{F}\bar{v}_\mathrm{gf}\bar{v}_\mathrm{cf}))]]$

by Lemma 6.33. We set $\lambda(w) := \varphi(w)$. (This is actually Duplicator's winning strategy in Game I [64], but the exact strategy is needed later).

(b) Assume w is a foot not covered by the previous case. The bijection given by Duplicator's winning strategy in Game I defines $\lambda(w)$ (actually, $\lambda(w)$ is an arbitrary foot of the same segment as w).

(c) Finally, assume w is a gadget vertex. We use the bijection given by Duplicator's winning strategy in Game II to define $\lambda(w)$.

Now Spoiler places the pebble pair (p_i, q_i) on the vertices w and $w' := \lambda(w)$. We update the tuples $\bar{v}_{\text{gf}}$, $\bar{v}'_{\text{gf}}$, $\bar{v}_{\text{cf}}$, and $\bar{v}'_{\text{cf}}$ as follows:

(a) Assume w (and thus also w') is a gadget vertex. Property 4 clearly holds because we followed Duplicator's winning strategy in Game II. No new segments in X get closure-fixed by $\bar{u}w$ and $\bar{u}'w'$,respectively, so we just do not change $\bar{v}_{\text{cf}}$ and $\bar{v}'_{\text{cf}}$, respectively, and Property 2 is satisfied. We satisfy Properties 1, 3, and 5 by picking feet from the new gadget-fixed segments as follows:

- Assume that a segment becomes gadget-fixed by $\bar{u}w$ (or $\bar{u}'w'$), which is already closure-fixed by $\bar{u}$ (or $\bar{u}'$, respectively). Then this segment is contained in X. We pick the same feet as in $\bar{v}_{\text{cf}}$ and $\bar{v}'_{\text{cf}}$, respectively, and append them to $\bar{v}'_{\text{gf}}$ and to $\bar{v}'_{\text{gf}}$, respectively. Thus, Property 1 holds. Because the local automorphism from Property 3 already maps $\bar{v}_{\text{cf}}$ to $\bar{v}'_{\text{cf}}$, appending the corresponding entries to $\bar{v}'_{\text{gf}}$ and $\bar{v}'_{\text{gf}}$ satisfies Property 3. Because the closure-fixed segments are part of the pebbled vertices in Game II (Property 4), appending these feet to $\bar{v}'_{\text{gf}}$ and $\bar{v}'_{\text{gf}}$ satisfies Property 5.
- Assume that a segment $\mathfrak{u}$ becomes gadget-fixed by $\bar{u}w$ (or $\bar{u}'w'$, respectively), which is not closure-fixed by $\bar{u}$ (or $\bar{u}'$, respectively). For $\mathfrak{B}$ (and $\mathfrak{B}'$, respectively), we pick the unique foot v and v', respectively, of the segment $\mathfrak{u}$ adjacent to the newly pebbled gadget vertex and append them to $\bar{v}_{\text{gf}}$ and $\bar{v}'_{\text{gf}}$, respectively. Hence, Property 1 is satisfied. Property 3 is satisfied by Lemma 6.35: We can pick for non-closure-fixed segments arbitrary feet and still find a local automorphism mapping them onto each other. For the sake of contradiction, suppose that Property 5 is not satisfied by appending v and v'. Then there is a base vertex $\mathfrak{v} \in V^H$ such that $(\mathfrak{B}', \bar{u}_{\text{G}}\bar{v}_{\text{gf}}v)[V_{\mathfrak{v}}] \not\cong (\mathfrak{B}, \bar{u}'_{\text{G}}\bar{v}'_{\text{gf}}v')[V_{\mathfrak{v}}]$. There must be a pebble placed on a gadget vertex with origin $\mathfrak{v}$, i.e., both $\bar{u}_{\text{G}}$ and $\bar{u}'_{\text{G}}$ are nonempty, because otherwise $|\bar{v}_{\text{gf}}| \leq k$ and $|\bar{u}'_{\text{gf}}| \leq k$ and thus the two edge vertices of the segment $\mathfrak{u}$ form an orbit by Lemma 2.14 since H is $(k+2)$-connected (the lemma also holds in the setting of a single edge-vertex-pair per base edge). In particular, $(\mathfrak{B}', \bar{u}_{\text{G}}v)[V_{\mathfrak{v}}] \not\cong (\mathfrak{B}, \bar{u}'_{\text{G}}v')[V_{\mathfrak{v}}]$. Note that v and v' are $\mathcal{C}_3$-definable because they are the unique vertices in the segment $\mathfrak{u}$ adjacent to w and w', respectively. So Spoiler can win Game II by picking up a pebble pair (which is neither placed on w nor the gadget of $\mathfrak{v}$) and placing it on v. Because $k \geq 3$, such a pebble pair actually exists. But that contradicts Property 4 and hence Property 5 is satisfied.

(b) Assume w (and thus also w') is a foot. Thus, no segments get gadget-fixed by $\bar{u}w$ and $\bar{u}'w'$, respectively, which was not already gadget-fixed by $\bar{u}$ and $\bar{u}'$, respectively. So we just do not change $\bar{v}_{\mathrm{gf}}$ and $\bar{v}'_{\mathrm{gf}}$ and Properties 1 and 5 are satisfied. Possibly a new segment $\mathfrak{u} \in X$ becomes closure-fixed. By Lemma 6.38, there can at most be one such segment. We satisfy Properties 2 to 4 as follows:

- Assume that $\mathfrak{u}$ is already gadget-fixed by $\bar{u}$ and $\bar{u}'$, respectively. We append the vertices v and v' whose segment is $\mathfrak{u}$ in $\bar{v}_{\mathrm{gf}}$ and $\bar{v}'_{\mathrm{gf}}$, respectively, to $\bar{v}_{\mathrm{cf}}$ and $\bar{v}'_{\mathrm{cf}}$, respectively. So Property 2 holds. Property 3 is satisfied because the local automorphism φ already maps $\bar{v}_{\mathrm{gf}}$ to $\bar{v}'_{\mathrm{gf}}$ and hence appending v and v', respectively, satisfies Property 2. Property 4 is satisfied because of Property 5: Fixing a single gadget vertex fixes all edge vertices adjacent to the gadget by Lemma 2.12 (which is $\mathcal{C}_3$-definable) and by Property 5 we have chosen $\bar{v}_{\mathrm{gf}}$ and $\bar{v}'_{\mathrm{gf}}$, respectively, consistently with the pebbles on the gadgets. Hence, we can actually place a pebble pair v and v' and Property 4 holds.
- Otherwise, $\mathfrak{u}$ is not gadget-fixed by $\bar{u}$ and $\bar{u}'$, respectively. By Lemma 2.14, the two feet of $\mathfrak{u}$ form an orbit in $(\mathfrak{B}, \bar{u}_X\bar{u}_{\mathrm{G}}\bar{v}_{\mathrm{cf}})$ and likewise in $(\mathfrak{B}, \bar{u}'_X\bar{u}'_{\mathrm{G}}\bar{v}'_{\mathrm{cf}})$ because H is $(k+2)$-connected. Hence, every choice of feet in $\mathfrak{u}$ satisfies Property 4. We make an arbitrary choice of a foot v of the segment $\mathfrak{u}$ in $\mathfrak{B}$, which we append to $\bar{v}_{\mathrm{cf}}$. Because $\mathfrak{u}$ is closure-fixed by $\bar{u}w$, we can extend φ to the segment $\mathfrak{u}$. We pick $v' = \varphi(u)$ and append it to $\bar{v}'_{\mathrm{cf}}$. So Property 3 is satisfied.

So Duplicator maintains the invariant. We update $\bar{u}$ and $\bar{u}'$ to include w and w', respectively. We show that the pebbles induce a local isomorphism: By Property 4, the pebbles induce a local isomorphism $\bar{u}_X\bar{u}_{\mathrm{G}}\bar{v}_{\mathrm{cf}} \mapsto \bar{u}'_X\bar{u}'_{\mathrm{G}}\bar{v}'_{\mathrm{cf}}$ on the CFI graphs. This local isomorphism extends to all gadget-fixed segments by Property 5, that is, $\bar{u}_X\bar{u}_{\mathrm{G}}\bar{v}_{\mathrm{cf}}\bar{v}_{\mathrm{gf}} \mapsto \bar{u}'_X\bar{u}'_{\mathrm{G}}\bar{v}'_{\mathrm{cf}}\bar{v}'_{\mathrm{gf}}$ is a local isomorphism. By Property 3, the map $\bar{u}_{\mathrm{F}}\bar{v}_{\mathrm{gf}}\bar{v}_{\mathrm{cf}} \mapsto \bar{u}'_{\mathrm{F}}\bar{v}'_{\mathrm{gf}}\bar{v}'_{\mathrm{cf}}$ is a local automorphism on the multipede. Because the maps agree on $\bar{v}_{\mathrm{gf}}\bar{v}_{\mathrm{cf}}$, the combined map $\bar{u}_X\bar{u}_{\mathrm{G}}\bar{v}_{\mathrm{cf}}\bar{v}_{\mathrm{gf}}\bar{u}_{\mathrm{F}} \mapsto \bar{u}'_X\bar{u}'_{\mathrm{G}}\bar{v}'_{\mathrm{cf}}\bar{v}'_{\mathrm{gf}}\bar{u}'_{\mathrm{F}}$ is a local isomorphism. So in particular, the map $\bar{u}_X\bar{u}_{\mathrm{G}}\bar{u}_{\mathrm{F}} \mapsto \bar{u}'_X\bar{u}'_{\mathrm{G}}\bar{u}'_{\mathrm{F}}$ is a local isomorphism, but that is just the map $\bar{u} \mapsto \bar{u}'$. So Duplicator does not lose in this round and by induction wins the bijective k-pebble game. □

Theorem 6.40. *There is an* FO*-interpretation* Θ *and, for every* $k \in \mathbb{N}$*, a pair of ternary* $\{R, \preceq\}$*-structures* $(\mathfrak{A}_k, \mathfrak{B}_k)$ *such that*

(a) $\preceq$ *is a total preorder on* $\mathfrak{A}$ *and* $\mathfrak{B}$*,*

(b) $\mathfrak{A}_k$ *and* $\mathfrak{B}_k$ *are asymmetric,*

(c) $\mathfrak{A}_k \equiv^k_{\mathcal{C}} \mathfrak{B}_k$*,*

(d) $\mathfrak{A}_k \not\cong \mathfrak{B}_k$*, and*

(e) $\Theta(\mathfrak{A}_k)$ *and* $\Theta(\mathfrak{B}_k)$ *are non-isomorphic CFI graphs over the same ordered base graph.*

The interpretation Θ *is* 1*-dimensional and equivalence-free.*

Proof. Let $k \geq 2$ and H be a clique of size $k+4$ (and thus H is $r := (k+3)$-regular, $(k+2)$-connected, and has $m := \frac{(k+4)(k+3)}{2} \leq r(k+2)$ many edges). There exists an odd and $(6r(k+2))$-meager bipartite graph $G = (V, W, E)$, which contains a $6r(k+2)$-scattered set $X \subseteq W$ of size $m \leq (6r(k+2))^2$ by Lemma 6.26. Equip the graphs G and H with an arbitrary total order.

Set $\mathfrak{A}_k := \mathsf{MP}(G) \cup_X \mathsf{CFI}(H,0)$ and $\mathfrak{B}_k := \mathsf{MP}(G) \cup_X \mathsf{CFI}(H,1)$. Clearly, $\mathfrak{A}_k \not\cong \mathfrak{B}_k$ and $\mathfrak{A}_k \equiv^k_{\mathcal{C}} \mathfrak{B}_k$ by Lemma 6.39 because H is $(k+2)$-connected and r-regular, G is $(6r(k+2))$-meager and so in particular $2rk$-meager, and X is $6r(k+2)$-scattered and so in particular $6rk$-scattered. By Lemma 6.22, the multipede $\mathsf{MP}(G)$ is asymmetric because G is odd. Thus, $\mathfrak{A}_k$ and $\mathfrak{B}_k$ are asymmetric by Lemma 6.37.

It remains to define the interpretation Θ, which maps $\mathfrak{A}_k$ to $\mathsf{CFI}(H,0)$ and $\mathfrak{B}_k$ to $\mathsf{CFI}(H,1)$. Recall that the gluing extends edges of the CFI graphs to triples by repeating the last entry and multipedes do not contain such triples. So we can easily define the vertices contained in the "CFI triples". By taking the induced graph on these vertices and by shortening the triples back to pairs, one defines the CFI graphs again. This is done by the following 1-dimensional and equivalence-free FO$[\{R, \preceq\}, \{E, \preceq\}]$-interpretation $\Theta = (\Phi_{\text{dom}}(x), \Psi_E(x,y), \Psi_{\preceq}(x,y))$:

$$\begin{aligned}
\Phi_{\text{dom}}(x) &:= \exists y.\ R(x,y,y) \lor R(y,x,x),\\
\Psi_E(x,y) &:= R(x,y,y),\\
\Psi_{\preceq}(x,y) &:= x \preceq y.
\end{aligned}$$

Clearly, $\Theta(\mathfrak{A}_k) = \mathsf{CFI}(H,0)$ and $\Theta(\mathfrak{B}_k) = \mathsf{CFI}(H,1)$. □

We now separate IFPC+WSC from IFPC+WSC+I.

Theorem 6.41. IFPC+WSC $<$ IFPC+WSC+I $\leq$ *Ptime*.

Proof. Consider the class of $\{R, \preceq\}$-structures $\mathcal{K}$ given by Theorem 6.40. To ensure that the reduct semantics does not add automorphisms when not using $\preceq$ in a formula, we additionally encode $\preceq$ into R. We add a directed path of length i and another one of length $i+1$ to every vertex in the i-th color class. In this way, we obtain the class of $\{R, \preceq\}$-structures $\mathcal{K}'$, for which also the $\{R\}$-reducts are asymmetric. Clearly, there is a 1-dimensional and equivalence-free FO-interpretation $\Theta_{\mathcal{K}}$ mapping a $\mathcal{K}'$-structure back to the corresponding $\mathcal{K}$-structure (remove all vertices of out-degree at most 1 because we attached to all original vertices two paths).

We argue that IFPC+WSC = IFPC on $\mathcal{K}'$. If a WSC-fixed-point operator mentions R, then the structure is asymmetric, i.e., there are only singleton orbits. Hence, choosing becomes useless and can be simulated by (non-WSC)-fixed-point operators. If a WSC-fixed-point operator does not mention R, then the $\{\preceq\}$-reduct is fully determined by the number and sizes of the color classes (which must be equal for $\mathfrak{A}_k$ and $\mathfrak{B}_k$ for every $k \geq 3$ because otherwise $\mathcal{C}_3$ distinguishes $\mathfrak{A}_k$ and $\mathfrak{B}_k$). The number of color classes and their sizes are clearly IFPC-definable.

For every $k \in \mathbb{N}$, there are two non-isomorphic structures $\mathfrak{A}_k \equiv^k_{\mathcal{C}} \mathfrak{B}_k$ in $\mathcal{K}'$ and thus IFPC does not define the isomorphism problem of $\mathcal{K}'$. It remains to show that IFPC+WSC+I defines the isomorphism problem. The CFI-query on ordered base graphs is definable in IFPC+WSC+I by Theorem 6.4. By Corollary 6.8, there is actually a WSCI(IFPC) = WSC(IFPC)-formula Φ_{CFI} defining the CFI query on ordered base graphs. Let Θ_{CFI} be the FO-interpretation extracting the CFI graphs from $\mathcal{K}$-structures given by Theorem 6.40. Then the I(WSC(IFPC))-formula

$$\mathsf{I}(\Theta_{\text{CFI}} \circ \Theta_{\mathcal{K}}; \Phi_{\text{CFI}})$$

defines the isomorphism problem of $\mathcal{K}'$-structures: Θ_{CFI} reduces by Theorem 6.40 the isomorphism problem of $\mathcal{K}$ to the isomorphism problem of CFI graph over ordered base graphs, which is defined by Φ_{CFI}. Note that because both Θ_{CFI} and $\Theta_{\mathcal{K}}$ are 1-dimensional and equivalence-free FO-interpretations, so is $\Theta_{\text{CFI}} \circ \Theta_{\mathcal{K}}$, too. □

Corollary 6.42. IFPC+WSC < *PTIME.*

Corollary 6.43. WSC(IFPC) < I(WSC(IFPC)).

Note that the prior corollary refines Corollary 6.20, which states that

$$\text{WSC(IFPC)} = \text{WSCI(IFPC)} < \text{WSCI}^2\text{(IFPC)} = \text{WSC(I(WSC(IFPC)))}.$$

We actually expect that

$$\text{WSC(IFPC)} < \text{I(WSC(IFPC))} < \text{WSC(I(WSC(IFPC)))}$$

because it seems unlikely that I(WSC(IFPC)) defines the CFI query of the base graphs of Theorem 6.19.

Corollary 6.44. IFPC+WSC *is not closed under* IFPC*-interpretations and not even under* 1*-dimensional and equivalence-free* FO*-interpretations.*

Proof. Note that the interpretation $\Theta_{\text{CFI}} \circ \Theta_{\mathcal{K}}$ in the proof of Theorem 6.41 is 1-dimensional and equivalence-free. The interpretation actually only removes vertices. The claim follows. □

Similary, we can answer the question of Dawar and Richerby [31] whether IFP+SC is closed under FO-interpretations in the negative:

Corollary 6.45. IFP+SC *is not closed under* 1*-dimensional and equivalence-free* FO*-interpretations.*

Proof. We consider the same structures: IFP = IFP+SC for the constructed structures because they only have trivial orbits and it does not make a difference whether we need to witness choices. Because IFPC does not define isomorphism, surely IFP does neither. Because IFP+WSC defines the CFI query for ordered base graphs [42], so does IFP+SC. We conclude that IFP+SC is not closed under FO-interpretations, too. □

6.5 Discussion

We defined the logics IFPC+WSC and IFPC+WSC+I to study the combination of witnessed symmetric choice and interpretations beyond simulating counting. Instead, we provided graph constructions to prove lower bounds. IFPC+WSC+I canonizes CFI graphs if it canonizes the base graphs, but operators have to be nested. We proved that this increase in nesting depth is unavoidable using double CFI graphs obtained by essentially applying the CFI construction twice. Does iterating our construction further show an operator nesting hierarchy in IFPC+WSC+I, i.e., $\mathrm{WSCI}^k(\mathrm{IFPC}) < \mathrm{WSCI}^{k+1}(\mathrm{IFPC})$ for every $k \in \mathbb{N}$? The main difficulty to extend our proof of Theorem 6.19 to an operator nesting hierarchy is the interpretation operator: For IFPC we used that $\mathrm{I}(\mathrm{IFPC}) = \mathrm{IFPC}$, but for $\mathrm{I}(\mathrm{WSCI}^k(\mathrm{IFPC}))$ this probably does not hold. So one needs to analyze the possible images of CFI graphs under $\mathrm{WSCI}^k(\mathrm{IFPC})$-interpretations and whether our proof idea can be extended to these images.

We have seen that also in the presence of counting the interpretation operator strictly increases the expressiveness. So indeed both, witnessed symmetric choice and interpretations are needed to possibly capture PTIME. This answers the question about the relation between witnessed symmetric choice and interpretations for IFPC. But it remains open whether IFPC+WSC+I captures PTIME. Here, iterating our CFI construction is of interest again: If one shows an operator nesting hierarchy using this construction, then one in particular will separate IFPC+WSC+I from PTIME because our construction does not change the signature of the structures.

Chapter 7

Separating Rank Logic from Polynomial Time

We lastly consider the approach of capturing PTIME by extending fixed-point logic with algebraic operators. One prominent approach is extending IFPC by a rank operator so that the logic has access to the rank of definable matrices over finite fields. For a finite structure $\mathfrak{A}$ with universe A, an $A^k \times A^k$ matrix is defined by a numeric term $s(\bar{x}, \bar{y})$ for $|\bar{x}| = |\bar{y}| = k$ by setting the entry indexed by $(\bar{u}, \bar{v})$ to the value $s(\bar{u}, \bar{v})^{\mathfrak{A}}$, to which s evaluates in the structure $\mathfrak{A}$. For such a matrix, we call the rank operator k-ary.

Multiple variants of rank logic were proposed. In its first version [27], rank logic comes with a rank operator rk_p for each prime p. The rank operator rk_p evaluates to the rank of a definable matrix over $\mathbb{F}_p$. These definable matrices are unordered in the sense that their rows and columns are indexed by tuples of atoms, on which in general no total order is definable. The rank of such matrices is not definable in IFPC. This is shown by the CFI graphs, which essentially are a graph encoding of a class of equation systems over $\mathbb{F}_2$ [45].

Using generalizations of the CFI construction from $\mathbb{F}_2$ (with two edge vertices per base edge, i.e., a directed cycle of length 2) to $\mathbb{F}_p$ (with a directed cycle of length p for every base edge), Grädel and Pakusa [45] showed that the rank operators rk_p is necessary to define the CFI query over $\mathbb{F}_p$. In particular, when considering all CFI graphs over all $\mathbb{F}_p$, no rank logic formula using the rank operators rk_p defines the CFI query because every formula only contains finitely many rk_p operators. An alternative version of rank logic was proposed in [45,68,83,103]. It replaces the rank operators rk_p by a uniform rank operator rk, which specifies the characteristic of the field in terms of the input structure. In this way, the CFI query over all $\mathbb{F}_p$ becomes definable. In this chapter, we consider this uniform rank operator and show that this more expressive variant of rank logic does not capture PTIME. To do so, we consider a generalization of CFI graphs over the rings $\mathbb{Z}_{2^i}$ for every i and prove that rank logic fails to define their isomorphism problem. We will exploit a recursive approach over the arity of the rank operators. We will actually show that rank logic does not even capture CPT.

Related Work. Hella [66] showed that for generalized Lindström quantifiers the expressiveness strictly increases with the arity of the quantifiers. A similar result was shown for rank logic [27,68,83]. It that light, it will not be surprising that general k-ary rank operators will require a much more complex approach than 1-ary rank operators.

In Chapter 6, we considered the multipedes [64], which are asymmetric structures with non-IFPC-definable orbits and a non-IFPC-definable isomorphism problem. However, this isomorphism problem is definable in rank logic [68] similar to the CFI query over $\mathbb{F}_2$. Moreover, rank logic captures PTIME on the class of structures with color class size 2 [118].

An open question [24] is whether rank logic defines the solvability of linear equation systems over finite rings rather than only over finite fields. As for $\mathbb{F}_2$, the isomorphism problem for CFI graphs over $\mathbb{Z}_{2^i}$ can be translated to a linear equation system over $\mathbb{Z}_{2^i}$. Hence, we also answer the question for solvability of linear equation systems over finite rings, where the ring is part of the input and so encoded in a relational structure, in the negative. The case of linear equation system for a fixed ring remains open.

Recall from Chapter 4 that the CPT-canonization for classes with bounded and abelian colors requires defining solvability over a certain cyclic class of linear equation systems over $\mathbb{Z}_{p^i}$ for prime powers p^i called cyclic linear equation systems. Grädel and Grohe [43] already suggested that defining solvability for this class of equation systems might be a candidate for separating CPT from rank logic.

Overview of this Chapter. In Section 7.1, we introduce rank logic with the uniform rank operator rk and also the more restrictive version using the rank operators rk_p for fixed characteristic p. Proving that rank logic fails to capture PTIME is based on pebble-game-arguments. There is a pebble game we call the rank-pebble game corresponding to rank logic [29]. However, the invertible-map game [29], which potentially distinguishes more structures than the rank-pebble game, is more suitable for our arguments. This game is introduced in Section 7.2. In the game, sequences of invertible matrices that are simultaneously similar play an important role.

In Section 7.3, we generalize the CFI construction of Section 2.8 from the field $\mathbb{F}_2$ to the rings $\mathbb{Z}_{2^i}$. We actually consider vertex and edge colored graphs, which are encoded into 4-ary structures. The goal is to show that the isomorphism problem of these generalized CFI structures over $\mathbb{Z}_{2^i}$ is not definable in rank logic. One key observation will be that it suffices to consider matrices and ranks over $\mathbb{F}_2$ because the automorphism groups of CFI structures over $\mathbb{Z}_{2^i}$ are 2-groups. This generalizes the result of Grädel and Pakusa [45]. Using the game-based approach, we will show that the characteristic 2 invertible-map game fails to distinguish all non-isomorphic CFI structures over $\mathbb{Z}_{2^i}$.

In Section 7.4, we consider properties of definable matrices over these CFI structures. A key point for a deeper understanding of these matrices is the following: Two entries in such a matrix that are indexed by tuples in the same orbit always contain the same value (so 0 or 1 in the case of $\mathbb{F}_2$). In that sense, definable matrices are orbit-respecting. We will consider base graphs of high connectivity so that the orbits of the CFI structure are structured as simple as possible. In particular, we establish a criterion for invertibility of such matrices. Then we focus on the arity 1 case in Section 7.5. We show that the 1-ary characteristic 2 invertible-map game fails to distinguish all CFI structures over $\mathbb{F}_4$. We introduce one key concept of our approach called blurrers and illustrate their usage in the easier arity 1 case. This case will also serve as base case for recursion on the arity.

Before we can turn to the general arity k case, we need to consider more properties of orbit-respecting matrices in Section 7.6. This section focuses on the notion of the active region of matrices, which intuitively is the part of matrix, in which it is locally not equal to the identity matrix. This notion becomes crucial in the arity k case, which is covered in Section 7.7. Here, we generalize our notion of blurrers to arity k and prove that the k-ary characteristic 2 invertible-map game fails to distinguish all CFI structures over $\mathbb{Z}_{2^i}$ for sufficiently large i. The proof will become formally intricate and combines blurrers, the active region of matrices, and a recursive approach over the arity. Sections 7.5 and 7.7 actually only consider a single round of the invertible-map game. In Section 7.8, we then show that the k-ary characteristic 2 invertible-map game fails to distinguish all CFI structures over all $\mathbb{Z}_{2^i}$ for every k. This separates rank logic from PTIME because the distinguishing power of the invertible-map game is at least as strong as the one of rank logic. To further separate rank logic from CPT, we prove that CPT defines the isomorphism problem of these CFI structures. For this, we use techniques from [103], which are related to canonizing structures with bounded and abelian color classes in CPT.

We lastly consider the more general linear-algebraic logic in Section 7.9. This logic subsumes every extension of IFPC by every linear-algebraic operator over finite fields. Together with results of Dawar, Grädel, and Pakusa [25], who show that the invertible-map game using all characteristics $p \neq 2$ fails to distinguish the CFI structures over $\mathbb{Z}_{2^i}$, linear-algebraic logic also fails to distinguish all CFI structures over all $\mathbb{Z}_{2^i}$ [28]. Hence, we finally show that extensions of fixed-point logic with linear-algebraic operators over finite fields cannot capture PTIME. We end this chapter with a discussion in Section 7.10.

7.1 Rank Logic

In this section we introduce **rank logic**. We consider the extension of IFPC by the **uniform rank operator** rk (IFPC+R) following the definition in [68]. The logic IFPC+R has the following additional rule to form numeric terms: Let $s(\bar{z}\bar{x}\bar{y})$ be a numeric IFPC+R-term such that $k := |\bar{x}| = |\bar{y}|$ and let t be a closed numeric IFPC+R-term. Then

$$r(\bar{z}) = \mathsf{rk}(\bar{x}, \bar{y}).\ (s, t)$$

is a numeric IFPC+R-term. For convenience, we call k the **arity** of the rank operator, although it is actually $2k$. We restricted the definition to square matrices, but this does not limit the expressive power. Let $\mathfrak{A}$ be a τ-structure and $\bar{w} \in A^{|\bar{z}|}$. The term s defines an $A^k \times A^k$ matrix $M_s^{(\mathfrak{A},\bar{w})}$ over $\mathbb{N}$ via

$$M_s^{(\mathfrak{A},\bar{w})}(\bar{u}, \bar{v}) := s^{\mathfrak{A}}(\bar{w}\bar{u}\bar{v}).$$

The rank operator is evaluated as follows:

$$\Big(\mathsf{rk}(\bar{x}, \bar{y}).\ (s, t)\Big)^{\mathfrak{A}}(\bar{w}) := \begin{cases} \text{the rank of } (M_s^{(\mathfrak{A},\bar{w})} \bmod p) \text{ over } \mathbb{F}_p & \text{if } p = t^{\mathfrak{A}} \text{ is a prime,} \\ 0 & \text{otherwise.} \end{cases}$$

We also need the more restrictive variant of rank logic with the **fixed-characteristic rank operator** rk_p for fields $\mathbb{F}_p$: For a set of prime numbers Ω, we define IFPC+R$_\Omega$ to be the variant of IFPC+R, in which we have a different rank operator $\mathsf{rk}_p(\bar{x}, \bar{y}).\ s$ for every $\mathbb{F}_p$ with $p \in \Omega$ instead of the uniform rank operator rk. That is, we have to fix the field in the formula independently of the structure. The rank operator rk_p always evaluates to the rank of the defined matrix over $\mathbb{F}_p$. This is not the case for the operator rk, where we can determine the value for p by the term t that may evaluate differently for different structures.

7.2 The Invertible-Map Game

This section considers Ehrenfeucht-Fraïssé-like pebble games with algebraic rules. They extend the bijective pebble game (cf. Section 2.3). The expressiveness of rank logic is related to the rank-pebble game (called matrix-equivalence game in [29]) in the same way the expressiveness of IFPC is related to the bijective pebble game via the embedding of IFPC into infinitary finite variable counting logic [29]. The rank-pebble game extends the bijective pebble game with ranks. Instead of looking at the rank-pebble game, we consider the invertible-map game [29]. Its distinguishing power is at least as strong as the one of the rank-pebble game in the sense that if Duplicator has a winning strategy in the invertible-map game, then Duplicator has a winning strategy in the rank-pebble game, too. Hence, to show that rank logic cannot distinguish two structures, it suffices to show that Duplicator has a winning strategy in the invertible-map game. The game is defined as follows:

Let k and m be two positive integers such that $2k \leq m$ and let Ω be a finite nonempty set of primes. Similar to the bijective pebble game, a position in the **invertible-map game** $\mathcal{M}^{m,k,\Omega}$ is a tuple $(\mathfrak{A}, \bar{u}; \mathfrak{B}, \bar{v})$ of two τ-structures $\mathfrak{A}$ and $\mathfrak{B}$, $\bar{u} \in A^{\leq m}$, and $\bar{v} \in B^{\leq m}$ such that $|\bar{u}| = |\bar{v}|$. For each structure, there are m many pebble pairs, where the pebbles of the i-th pair are labeled with the number i. In position $(\mathfrak{A}, \bar{u}; \mathfrak{B}, \bar{v})$, there are pebbles with the same label placed on u_i and v_i for all $i \in [|\bar{u}|]$. That is, if $|\bar{u}| < m$, some of the pebbles are not used. There are two players called Spoiler and Duplicator. If $|A| \neq |B|$, then Spoiler wins the game. Otherwise a round of the game proceeds as follows:

1. Spoiler chooses a prime $p \in \Omega$ and picks up $2k$ many pebbles from $\mathfrak{A}$ and the corresponding pebbles (with the same labels) from $\mathfrak{B}$.

2. Duplicator picks a partition $\mathbf{P}$ of $A^k \times A^k$ and another one $\mathbf{Q}$ of $B^k \times B^k$ such that $|\mathbf{P}| = |\mathbf{Q}|$. Furthermore, Duplicator picks an invertible $A^k \times B^k$ matrix S over $\mathbb{F}_p$, such that the matrix induces a total and bijective map $\lambda\colon \mathbf{P} \to \mathbf{Q}$ defined by

$$P \mapsto Q \text{ if and only if } \chi^P = S \cdot \chi^Q \cdot S^{-1}.$$

   Here χ^P (respectively χ^Q) is the characteristic $A^k \times A^k$ matrix over $\mathbb{F}_p$ of P (respectively the $B^k \times B^k$ matrix over $\mathbb{F}_p$ of Q) which satisfies that $\chi^P(\bar{u}', \bar{v}') = 1$ if $\bar{u}'\bar{v}' \in P$ and $\chi^P(\bar{u}', \bar{v}') = 0$ otherwise. To phrase it differently, Duplicator has to pick a bijection $\lambda\colon \mathbf{P} \to \mathbf{Q}$ and an invertible $A^k \times B^k$ matrix S satisfying

$$\chi^P = S \cdot \chi^{\lambda(P)} \cdot S^{-1}$$

   for all $P \in \mathbf{P}$, i.e., the characteristic matrices of $\mathbf{P}$ and $\mathbf{Q}$ are simultaneously similar.

3. Spoiler chooses a block $P \in \mathbf{P}$, a tuple $\bar{w} \in P$, and a tuple $\bar{w}' \in \lambda(P)$. For each $i \in [2k]$, Spoiler places a pebble on w_i and the corresponding pebble on w'_i.

After a round, Spoiler wins the game if the pebbles do not define a local isomorphism or if Duplicator was not able to respond with a matrix satisfying the conditions above. Duplicator wins the game if Spoiler forever fails to win. Spoiler has a winning strategy in position $(\mathfrak{A}, \bar{u}; \mathfrak{B}, \bar{v})$ if Spoiler can win independently of the actions of Duplicator.

Likewise, Duplicator has a winning strategy if Duplicator can always win the game. In that case, we write $(\mathfrak{A}, \bar{u}) \equiv_{\mathcal{M}}^{m,k,\Omega} (\mathfrak{B}, \bar{v})$. In the following, we use the invertible-map game instead of the rank-pebble game because it allows us to prove a stronger result and, at the same time, simplifies proofs.

Lemma 7.1 ([29]). *Let $\mathcal{K}$ be a class of finite τ-structures and let $Q \subseteq \mathcal{K}$ be a boolean query. If, for all $k, m \in \mathbb{N}$ with $2k \leq m$ and every finite and nonempty set of primes Ω, there is a pair of structures $(\mathfrak{A}, \mathfrak{B})$ such that $\mathfrak{A} \in Q$, $\mathfrak{B} \notin Q$, and $\mathfrak{A} \equiv_{\mathcal{M}}^{m,k,\Omega} \mathfrak{B}$, then Q is not* IFPC+R$_{\mathbb{P}}$ *definable, where $\mathbb{P}$ is the set of all primes.*

If we fix a finite set of primes Ω in Lemma 7.1, then P is not IFPC+R$_\Omega$ definable (see [29]) because a P-defining IFPC+R$_\Omega$-formula implies a winning strategy of Spoiler in the $\mathcal{M}^{m,k,\Omega}$ game. Lemma 7.1 is proved in [29] for the (m, k, Ω)-rank-pebble game, which induces the equivalence $\equiv_{\mathcal{R}}^{m,k,\Omega}$. Then the authors show that $\equiv_{\mathcal{M}}^{m,k,\Omega}$ refines $\equiv_{\mathcal{R}}^{m,k,\Omega}$. It is an open problem whether the equivalence $\equiv_{\mathcal{M}}^{m,k,\Omega}$ strictly refines $\equiv_{\mathcal{R}}^{m,k,\Omega}$.

7.3 CFI Structures over Rings $\mathbb{Z}_{2^q}$

We now generalize the CFI graphs over $\mathbb{F}_2$ from Section 2.8. The CFI construction generalizes to other finite fields than $\mathbb{F}_2$ and even to abelian groups [95]. We are interested in cyclic groups $\mathbb{Z}_{2^q}$ and use the variant of CFI gadgets only consisting of gadget vertices (cf. Section 2.8.4). Recall that a base graph is a simple, connected, and colored graph. We only consider ordered base graphs in this section. Let $G = (V, E, \leq)$ be a base graph. We consider the additive group of $\mathbb{Z}_{2^q}$. For each base vertex $\mathfrak{u} \in V$, we define a gadget consisting of a set of vertices $A_\mathfrak{u}$ and two families of relations. In the following definition, we need many pairs and tuples, and, for the sake of readability, we simply write $\mathfrak{u}\bar{a}$ for the pair $(\mathfrak{u}, \bar{a})$.

$$\begin{aligned}
A_\mathfrak{u} &:= \left\{ \mathfrak{u}\bar{a} \;\middle|\; \bar{a} \in \mathbb{Z}_{2^q}^{N_G(\mathfrak{u})}, \sum \bar{a} = 0 \right\}, && \text{for all } \mathfrak{u} \in V,\\
I_{\mathfrak{u},\mathfrak{v}} &:= \left\{ (\mathfrak{u}\bar{a}, \mathfrak{u}\bar{b}) \in A_\mathfrak{u}^2 \;\middle|\; \bar{a}(\mathfrak{v}) = \bar{b}(\mathfrak{v}) \right\}, && \text{for all } \mathfrak{u} \in V, \mathfrak{v} \in N_G(\mathfrak{u}),\\
C_{\mathfrak{u},\mathfrak{v}} &:= \left\{ (\mathfrak{u}\bar{a}, \mathfrak{u}\bar{b}) \in A_\mathfrak{u}^2 \;\middle|\; \bar{a}(\mathfrak{v}) + 1 = \bar{b}(\mathfrak{v}) \right\}, && \text{for all } \mathfrak{u} \in V, \mathfrak{v} \in N_G(\mathfrak{u}).
\end{aligned}$$

Consider the sets $A_{\mathfrak{u},\mathfrak{v},c} := \{\mathfrak{u}\bar{a} \in A_\mathfrak{u} \mid \bar{a}(\mathfrak{v}) = c\}$ for $\mathfrak{v} \in N_G(\mathfrak{u})$ and $c \in \mathbb{Z}_{2^q}$. The relation $I_{\mathfrak{u},\mathfrak{v}}$ realizes these sets by disjoint cliques, one for each $A_{\mathfrak{u},\mathfrak{v},c}$. The relation $C_{\mathfrak{u},\mathfrak{v}}$ induces a directed cycle $A_{\mathfrak{u},\mathfrak{v},c}, A_{\mathfrak{u},\mathfrak{v},c+1}, \dots, A_{\mathfrak{u},\mathfrak{v},c+2^q-1}, A_{\mathfrak{u},\mathfrak{v},c}$ on these sets for a fixed $\mathfrak{v}$ by adding directed complete bipartite graphs between subsequent cliques. In this way, the relation $C_{\mathfrak{u},\mathfrak{v}}$ realizes the group $\mathbb{Z}_{2^q}$ on the sets $A_{\mathfrak{u},\mathfrak{v},c}$. The condition $\sum \bar{a} = 0$ on the atoms in $A_\mathfrak{u}$ implies that the automorphism group of the gadget for $\mathfrak{u}$ is isomorphic to $\{\bar{a} \in \mathbb{Z}_{2^q}^d \mid \sum \bar{a} = 0\}$, where d is the degree of $\mathfrak{u}$.

Now we connect gadgets. We first extend the order $\leq$ on the base graph to the lexicographical order on tuples of base vertices of G and further to sets of such tuples. Let $g \colon E \to \mathbb{Z}_{2^q}$ be a function defining the values by which the base edges are twisted. For every base edge $\{\mathfrak{u}, \mathfrak{v}\} \in E$, we connect the gadgets of the incident base vertices. We obtain the **CFI structure**

$$\mathfrak{A} = \mathsf{CFI}_{2^q}(G,g) := \Big(A, R_I^{\mathfrak{A}}, R_C^{\mathfrak{A}}, R_{E,0}^{\mathfrak{A}}, \dots, R_{E,2^q-1}^{\mathfrak{A}}, \preceq^{\mathfrak{A}}\Big)$$

as follows:

$$\begin{aligned} A &:= \bigcup_{\mathfrak{u}\in V} A_{\mathfrak{u}}, \\ E_{\{\mathfrak{u},\mathfrak{v}\},c}^{\mathfrak{A}} &:= \Big\{ \{\mathfrak{u}\bar{a}, \mathfrak{v}\bar{b}\} \;\Big|\; \mathfrak{u}\bar{a} \in A_{\mathfrak{u}}, \mathfrak{v}\bar{b} \in A_{\mathfrak{v}}, \bar{a}(\mathfrak{v}) + \bar{b}(\mathfrak{u}) = c \Big\}, && c \in \mathbb{Z}_{2^q}, \\ R_{E,c}^{\mathfrak{A}} &:= \bigcup_{\mathfrak{e}\in E} E_{\mathfrak{e},c+g(\mathfrak{e})}, && c \in \mathbb{Z}_{2^q}, \\ \preceq^{\mathfrak{A}} &:= \Big\{ (\mathfrak{u}\bar{a}, \mathfrak{v}\bar{b}) \in A^2 \;\Big|\; \mathfrak{u} \leq \mathfrak{v} \Big\}. \end{aligned}$$

The relations $I_{\mathfrak{u},\mathfrak{v}}$ (and similarly $C_{\mathfrak{u},\mathfrak{v}}$) are encoded by the 4-ary relations R_I (and R_C) as follows: All pairs $\bar{u} \in A_{\mathfrak{u}}^2$, that is, $\bar{u} = (\mathfrak{u}\bar{a}, \mathfrak{u}\bar{b})$, are partitioned according to the set of base vertices $\mathfrak{v}$ such that $\bar{u} \in I_{\mathfrak{u},\mathfrak{v}}$. The partition is given by the equivalence classes of R_I (seen as equivalence on pairs). In this way, the relations $I_{\mathfrak{u},\mathfrak{v}}$ (respectively $C_{\mathfrak{u},\mathfrak{v}}$) are unions of R_I-equivalence classes (respectively R_C-equivalence classes). Intuitively, each pair $\bar{u} \in A_{\mathfrak{u}}^2$ is colored by all the set $I_{\mathfrak{u},\mathfrak{v}}$ (and $C_{\mathfrak{u},\mathfrak{v}}$, respectively) containing $\bar{u}$.

$$\begin{aligned} R_I^{\mathfrak{A}} &:= \Big\{ (\bar{u},\bar{v}) \in A^2 \times A^2 \;\Big|\; \Big\{ (\mathfrak{u},\mathfrak{v}) \in V^2 \;\Big|\; \bar{u} \in I_{\mathfrak{u},\mathfrak{v}} \Big\} \leq \Big\{ (\mathfrak{u}',\mathfrak{v}') \in V^2 \;\Big|\; \bar{v} \in I_{\mathfrak{u}',\mathfrak{v}'} \Big\} \Big\}, \\ R_C^{\mathfrak{A}} &:= \Big\{ (\bar{u},\bar{v}) \in A^2 \times A^2 \;\Big|\; \Big\{ (\mathfrak{u},\mathfrak{v}) \in V^2 \;\Big|\; \bar{u} \in C_{\mathfrak{u},\mathfrak{v}} \Big\} \leq \Big\{ (\mathfrak{u}',\mathfrak{v}') \in V^2 \;\Big|\; \bar{v} \in C_{\mathfrak{u}',\mathfrak{v}'} \Big\} \Big\}. \end{aligned}$$

The former definition, which only uses gadget vertices, could be simplified by using gadget and edge vertices and even more by only using edge vertices (cf. Section 2.8.4). Using only edge and gadget vertices creates unnecessary case distinctions (cf. the discussion in Section 2.8.4). Our argument separating rank logic from CPT requires base graphs of unbounded degree. So, only using edge vertices would create structures of different arity. Our construction always yields structures of arity 4, but the number of relations varies with the group $\mathbb{Z}_{2^q}$. Of course, we could use a single relation to encode the relations $R_{E,c}$. But in fact, it suffices only to use $R_{E,0}$ to obtain a structure with the same automorphism group. Then all $R_{E,c}$ are actually definable in 3-variable counting logic. For convenience, we include all $R_{E,c}$ in the structure. We transfer the notion of the origin to the generalized CFI structures.

Definition 7.2 (Origin). We say that the atom $\mathfrak{u}\bar{a} \in A$ **originates** from $\mathfrak{u}$ or that its **origin** is $\mathfrak{u}$ and write

$$\mathsf{orig}(\mathfrak{u}\bar{a}) := \mathfrak{u}.$$

We extend this to tuples and define the **origin** of $\bar{u} \in A^j$ as

$$\mathsf{orig}(\bar{u}) := \Big(\mathsf{orig}(u_1), \dots, \mathsf{orig}(u_j)\Big).$$

We will often view $\mathsf{orig}(\bar{u})$ as the set $\{\mathsf{orig}(u_1), \dots, \mathsf{orig}(u_j)\}$ and write $\mathfrak{u} \in \mathsf{orig}(\bar{u})$. If M is a set of tuples of the same origin, we set $\mathsf{orig}(M) := \mathsf{orig}(\bar{u})$ for some (and thus all) $\bar{u} \in M$. For a set $W \subseteq V$, we define the **origin-induced substructure**

$$\mathsf{CFI}_{2^q}(G,g)[W] := \mathsf{CFI}_{2^q}(G,g)[\{\, u \mid \mathsf{orig}(u) \in W \,\}]$$

to be the substructure induced by all atoms whose origin is contained in W.

It will be always clear from the context whether we refer to the origin-induced substructure (or just a standard induced substructure). In this case W and the universe of the CFI structure are disjoint.

For CFI structures it is well-known [20, 25, 45, 95] that $\mathsf{CFI}_{2^q}(G, g) \cong \mathsf{CFI}_{2^q}(G, f)$ if and only if $\sum g = \sum f$ (also cf. Lemma 2.10). That is, there are up to isomorphism 2^q many CFI structures of the base graph G.

Lemma 7.3. *The automorphism group* $\mathsf{Aut}(\mathsf{CFI}_{2^q}(G, g))$ *of the CFI structure* $\mathsf{CFI}_{2^q}(G, g)$ *is an abelian* 2*-group.*

Proof. Every automorphism of $\mathsf{CFI}_{2^q}(G, g)$ is origin-respecting, i.e., it maps an atom to an atom of the same origin because the preorder $\preceq$ on the atoms is obtained from the total order $\leq$ on G. It follows that $\mathsf{Aut}(\mathsf{CFI}_{2^q}(G, g))$ is a subgroup of the direct product of the automorphism groups of every gadget. Because the automorphism group of each degree d gadget is the abelian 2-group $\{\bar{a} \in \mathbb{Z}_{2^q}^d \mid \sum \bar{a} = 0\}$ as argued before, so is the direct product of them and in particular $\mathsf{Aut}(\mathsf{CFI}_{2^q}(G, g))$. □

7.3.1 Isomorphisms of CFI Structures

In this section we consider two classes of isomorphisms between CFI structures. These isomorphisms will get important later in Section 7.7. Let $q \in \mathbb{N}$ and $G = (V, E, \leq)$ be a base graph. In the following, we denote, for every $f \colon E \to \mathbb{Z}_{2^q}$, by $\mathfrak{A}_f$ the CFI structure $\mathsf{CFI}_{\mathbb{Z}_{2^q}}(G, f)$. By definition, these structures have the same universe A for every $f \colon E \to \mathbb{Z}_{2^q}$.

Definition 7.4 (Twisted Base Edge). Two functions $f, g \colon E \to \mathbb{Z}_{2^q}$ **twist** a base edge $\mathfrak{e} \in E$ if $f(\mathfrak{e}) \neq g(\mathfrak{e})$. We also say that $\mathfrak{e}$ is **twisted** by f and g. For a set $W \subseteq V$ we say that f and g *do not twist* W if no base edge in $G[W]$ is twisted by f and g.

We omit f and g if they are clear from the context. Let $\mathfrak{u} \in V$ and $\bar{a} \in \mathbb{Z}_{2^q}^{N_G(\mathfrak{u})}$ satisfy $\sum \bar{a} = 0$. We identify $\bar{a}$ with a permutation of atoms with origin $\mathfrak{u}$ as follows: If $\mathfrak{u}\bar{b}$ has origin $\mathfrak{u}$ (in some CFI structure over G), then we set $\bar{a}(\mathfrak{u}\bar{b}) := \mathfrak{u}\bar{c}$ for the $\bar{c} \in \mathbb{Z}_{2^q}^{N_G(\mathfrak{u})}$ such that $\bar{c}(\mathfrak{v}) = \bar{b}(\mathfrak{v}) + \bar{a}(\mathfrak{v})$ for all $\mathfrak{v} \in N_G(\mathfrak{u})$. Because $\sum \bar{a} = 0$, the tuple $\bar{a}(\mathfrak{u}\bar{b})$ is indeed an atom with origin $\mathfrak{u}$.

Definition 7.5 (Path-Isomorphism). Let $c \in \mathbb{Z}_{2^q}$ and $\bar{\mathfrak{s}} = (\mathfrak{u}_1, \dots, \mathfrak{u}_n)$ be a simple path in G. For every $1 < i < n$, let $\bar{a}_i \in \mathbb{Z}_{2^q}^{N_G(\mathfrak{u}_i)}$ such that $\bar{a}_i(\mathfrak{u}_{i-1}) = c$, $\bar{a}_i(\mathfrak{u}_{i+1}) = -c$, and $\bar{a}_i(\mathfrak{v}) = 0$ for all other $\mathfrak{v} \in N_G(\mathfrak{u}_i)$. The **path-isomorphism** $\vec{\pi}[c, \bar{\mathfrak{s}}]$ is defined by

$$\vec{\pi}[c, \bar{\mathfrak{s}}](u) := \begin{cases} \bar{a}_i(u) & \text{if } \mathsf{orig}(u) = \mathfrak{u}_i \text{ and } 1 < i < n, \\ u & \text{otherwise.} \end{cases}$$

Lemma 7.6. *Let* $f, g \colon E \to \mathbb{Z}_{2^q}$*, let* $\bar{\mathfrak{s}} = (\mathfrak{u}_1, \dots, \mathfrak{u}_n)$ *be a simple path in* G*, and set* $\mathfrak{e}_1 := \{\mathfrak{u}_1, \mathfrak{u}_2\}$*, and* $\mathfrak{e}_2 := \{\mathfrak{u}_{n-1}, \mathfrak{u}_n\}$*. If no base edge apart from* $\mathfrak{e}_1$ *and* $\mathfrak{e}_2$ *is twisted by* f *and* g*,* $g(\mathfrak{e}_1) = f(\mathfrak{e}_1) + c$*, and* $g(\mathfrak{e}_2) = f(\mathfrak{e}_2) - c$*, then* $\vec{\pi}[c, \bar{\mathfrak{s}}]$ *is an isomorphism* $(\mathfrak{A}_f, \bar{p}) \to (\mathfrak{A}_g, \bar{p})$ *for every tuple* $\bar{p} \in A^m$ *satisfying* $\mathsf{dist}_G(\mathsf{orig}(\bar{p}), \{\mathfrak{u}_1, \dots, \mathfrak{u}_n\}) > 1$*.*

Lemma 7.6 is a generalization of Lemma 2.9. The proof of Lemma 7.6 is an obvious adaptation of the proof of Lemma 3.11 in [45]. This lemma uses a variant of CFI structures with gadget vertices and relations, but the arguments are similar. We additionally require that the tuple $\bar{p}$ is fixed, but because its distance to the path $\bar{\mathfrak{s}}$ is greater than 1, it is not affected by the path-isomorphism at all, i.e., $\vec{\pi}[c, \bar{\mathfrak{s}}](\bar{p}) = \bar{p}$. Isomorphisms between CFI structures satisfying $\sum f = \sum g$, in which more than two base edges are twisted, can be composed of multiple path-isomorphisms. The following special case of such isomorphisms will play an important role later:

Definition 7.7 (Star-Isomorphism). Let $\mathfrak{w} \in V$ be of degree d, $\ell \leq d$, $\bar{\mathfrak{s}}_1, \dots, \bar{\mathfrak{s}}_\ell$ be simple paths, $\bar{\mathfrak{s}}_i = (\mathfrak{u}^i_1, \dots, \mathfrak{u}^i_{\ell_i})$, $\mathfrak{u}^i_{\ell_i} = \mathfrak{w}$ for all $i \in [\ell]$, and the $\bar{\mathfrak{s}}_i$ be disjoint apart from $\mathfrak{w}$. We call the sequence $\bar{\mathfrak{s}}_1, \dots, \bar{\mathfrak{s}}_\ell$ a **star** and $\mathfrak{w}$ the **center of the star**. For $\bar{c} \in \mathbb{Z}^\ell_{2^q}$ satisfying $\sum \bar{c} = 0$, we define the **star-isomorphism** $\pi^*[\bar{c}, \bar{\mathfrak{s}}_1, \dots, \bar{\mathfrak{s}}_\ell]$ via

$$\pi^*[\bar{c}, \bar{\mathfrak{s}}_1, \dots, \bar{\mathfrak{s}}_\ell](u) := \begin{cases} \bar{c}'(u) & \text{if } \mathsf{orig}(u) = \mathfrak{w}, \\ \vec{\pi}[c_i, \bar{\mathfrak{s}}_i](u) & \text{if } \mathsf{orig}(u) \neq \mathfrak{w} \text{ and } \mathsf{orig}(u) \text{ is contained in } \bar{\mathfrak{s}}_i, \\ u & \text{otherwise,} \end{cases}$$

where $\bar{c}' \in \mathbb{Z}^{N_G(\mathfrak{w})}_{2^q}$ such that $\bar{c}'(\mathfrak{u}^i_{\ell_i - 1}) = c_i$ for all $i \in [\ell]$ and $\bar{c}'(\mathfrak{v}) = 0$ for all other $\mathfrak{v} \in N_G(\mathfrak{w})$.

Lemma 7.8. *Let $f, g \colon E \to \mathbb{Z}_{2^q}$, $\bar{\mathfrak{s}}_1, \dots, \bar{\mathfrak{s}}_\ell$ be a star in G, $\bar{\mathfrak{s}}_i = (\mathfrak{u}^i_1, \dots, \mathfrak{u}^i_{\ell_i})$ for all $i \in [\ell]$, and $\bar{c} \in \mathbb{Z}^\ell_{2^q}$ such that $\sum \bar{c} = 0$. If no base edge apart from the base edges $\mathfrak{e}_i = \{\mathfrak{u}^i_1, \mathfrak{u}^i_2\}$ for every $i \in [\ell]$ is twisted by f and g and $g(\mathfrak{e}_i) = f(\mathfrak{e}_i) + c_i$ for all $i \in [\ell]$, then $\pi^*[\bar{c}, \bar{\mathfrak{s}}_1, \dots, \bar{\mathfrak{s}}_\ell]$ is an isomorphism $(\mathfrak{A}_f, \bar{p}) \to (\mathfrak{A}_g, \bar{p})$ for every tuple $\bar{p} \in A^m$ satisfying $\mathsf{dist}_G(\mathsf{orig}(\bar{p}), \{\mathfrak{u}^i_j \mid i \in [\ell], j \in [\ell_i]\}) > 1$.*

Proof. Let $\bar{p} \in A^m$ satisfy $\mathsf{dist}_G(\mathsf{orig}(\bar{p}), \{\mathfrak{u}^i_j \mid i \in [\ell], j \in [\ell_i]\}) > 1$ and let $\mathfrak{w}$ be the center of the $\bar{\mathfrak{s}}_1, \dots, \bar{\mathfrak{s}}_\ell$. For every $i \in [\ell - 1]$, let $\bar{\mathfrak{s}}'_i$ be the $\mathfrak{u}^i_1$-$\mathfrak{u}^{i+1}_1$-path obtained by stitching $\bar{\mathfrak{s}}_i$ and $\bar{\mathfrak{s}}_{i+1}$ together at $\mathfrak{w} := \mathfrak{u}^i_{\ell_i}$ (that is, the path $\bar{\mathfrak{s}}_{i+1}$ is attached in reversed direction). Furthermore, for every $i \in [\ell - 1]$, set $\varphi_i := \vec{\pi}[\sum_{j \in [i]} c_j, \bar{\mathfrak{s}}'_i]$, and let $f_i \colon E \to \mathbb{Z}_{2^q}$ be the function defined via $f_i(\mathfrak{e}_j) = f(\mathfrak{e}_j) + c_j$ for every $j \in [i]$, $f_i(\mathfrak{e}_{i+1}) = f(\mathfrak{e}_{i+1}) - \sum_{j \in [i]} c_j$, and $f_i(\mathfrak{e}) = f(\mathfrak{e})$ otherwise. Applying Lemma 7.6 inductively shows that $\varphi_1 \circ \dots \circ \varphi_i$ is an isomorphism $(\mathfrak{A}_f, \bar{p}) \to (\mathfrak{A}_{f_i}, \bar{p})$: For $i = 1$, the only twisted base edges are $\mathfrak{e}_1$ and $\mathfrak{e}_2$ satisfying $f_1(\mathfrak{e}_1) = f(\mathfrak{e}_1) + c_1$ and $f_1(\mathfrak{e}_2) = f(\mathfrak{e}_2) - c_1$ and $\varphi_1 \colon (\mathfrak{A}_f, \bar{p}) \to (\mathfrak{A}_{f_1}, \bar{p})$ is an isomorphism by Lemma 7.6. For every $2 \leq i \leq \ell - 2$, exactly the base edges $\mathfrak{e}_{i+1}$ and $\mathfrak{e}_{i+2}$ are twisted by f_i and f_{i+1}. It holds that

$$f_{i+1}(\mathfrak{e}_{i+1}) = f(\mathfrak{e}_{i+1}) + c_{i+1} = f_i(\mathfrak{e}_{i+1}) + \sum_{j \in [i+1]} c_j,$$
$$f_{i+1}(\mathfrak{e}_{i+2}) = f(\mathfrak{e}_{i+2}) - \sum_{j \in [i+1]} c_i = f_i(\mathfrak{e}_{i+2}) - \sum_{j \in [i+1]} c_i.$$

Thus, φ_{i+1} is an isomorphism $(\mathfrak{A}_{f_i}, \bar{p}) \to (\mathfrak{A}_{f_{i+1}}, \bar{p})$ by Lemma 7.6 and $\varphi_1 \circ \dots \circ \varphi_{i+1}$ is an isomorphism $(\mathfrak{A}_f, \bar{p}) \to (\mathfrak{A}_{f_{i+1}}, \bar{p})$ by induction.

Now let $\psi = \varphi_1 \circ \dots \circ \varphi_\ell$. To prove the claim it suffices to show that $f_{\ell-1} = g$ and that $\psi = \pi^*[\bar{c}, \bar{\mathfrak{s}}_1, \dots, \bar{\mathfrak{s}}_\ell]$. The former holds because $\sum \bar{c} = 0$. To show the latter, first consider the path $\bar{\mathfrak{s}}_1$. On atoms with origin in $\bar{\mathfrak{s}}_1$ different from the center $\mathfrak{w}$ the action of ψ is equal to the action of φ_1. This exactly equals the definition of $\pi^*[\bar{c}, \bar{\mathfrak{s}}_1, \dots, \bar{\mathfrak{s}}_\ell]$. For atoms with origin in $\bar{\mathfrak{s}}_\ell$ different from $\mathfrak{w}$ the argument is similar and the action of ψ is equal to the action of $\varphi_{\ell-1}$. The isomorphism $\varphi_{\ell-1}$ twists the base edge $\{\mathfrak{u}_1^\ell, \mathfrak{u}_2^\ell\}$ by $-\sum_{j\in[\ell-1]} c_j$, which, by assumption, is equal to c_ℓ because $\sum \bar{c} = 0$. Now consider atoms with origin in $\bar{\mathfrak{s}}_i$ different from $\mathfrak{w}$ for $i \notin \{1, \ell\}$. Here the action of ψ equals the action of $\varphi_{i-1} \circ \varphi_i$. The isomorphism φ_{i-1} twists the edge $\{\mathfrak{u}_1^i, \mathfrak{u}_2^i\}$ by $-\sum_{j\in[i-1]} c_j$ and φ_i twists the same base edge by $\sum_{j\in[i]} c_j$. Note that $\bar{\mathfrak{s}}_i'$ contains the base vertices of $\bar{\mathfrak{s}}_{i+1}$ in reversed order, so on all the base vertices with origin different from $\mathfrak{w}$ the action of $\varphi_{i-1} \circ \varphi_i$ becomes equal to the action of the path-isomorphism $\vec{\pi}[c_i, \bar{\mathfrak{s}}_i]$. Finally, by a similar argument, the action of ψ on atoms with origin $\mathfrak{w}$ coincides with the action of $\vec{c}'$ defined as in Definition 7.7. □

7.3.2 Orbits of CFI Structures

In this section we analyze the structure of k-orbits of CFI structures for highly connected base graphs. Let $q, k, m \in \mathbb{N}$ and $G = (V, E, \leq)$ be a $(k + m + 1)$-connected base graph. We denote again, for every $f \colon E \to \mathbb{Z}_{2^q}$, by $\mathfrak{A}_f$ the CFI structure $\mathsf{CFI}_{\mathbb{Z}_{2^q}}(G, f)$ with universe A. Let $\bar{p} \in A^m$ be arbitrary but fixed. We consider the k-orbits of structures $(\mathfrak{A}_f, \bar{p})$, i.e., orbits of k-tuples. Recall that $\mathsf{Aut}((\mathfrak{A}_f, \bar{p}))$ is the automorphism group of $(\mathfrak{A}_f, \bar{p})$ and that $\mathsf{orb}_k((\mathfrak{A}_f, \bar{p}))$ is the set of all k-orbits (cf. Section 2.1).

Definition 7.9 (Type of a Tuple). The **isomorphism type** of a structure is the class of all isomorphic structures. For $f \colon E \to \mathbb{Z}_{2^q}$, the **type of a tuple** $\bar{u} \in A^k$ in $(\mathfrak{A}_f, \bar{p})$ is the pair $(\mathsf{orig}(\bar{u}), T)$, where T is the isomorphism type of the origin-induced substructure $(\mathfrak{A}_f[\mathsf{orig}(\bar{p}\bar{u})], \bar{p}\bar{u})$.

We omit the structure $(\mathfrak{A}_f, \bar{p})$ if it is clear from the context. Including $\mathsf{orig}(\bar{u})$ in the type is needed because the isomorphism type T respects the relative order of the gadgets in $\preceq$ only. If $\mathfrak{A}_f$ was vertex-colored instead, this would not be a problem. We have to consider the origin-induced substructure of $\mathsf{orig}(\bar{p}\bar{u})$ and not the induced substructure of $\bar{p}\bar{u}$ because only in the origin-induced substructure the relations $I_{\mathfrak{u},v}$ and $C_{\mathfrak{u},v}$ can be recovered from R_I and R_C. Here, an edge coloring would resolve this issue.

Lemma 7.10. *For every $f \colon E \to \mathbb{Z}_{2^q}$ and every $\bar{u}, \bar{v} \in A^k$, there is an automorphism $\varphi \in \mathsf{Aut}((\mathfrak{A}_f, \bar{p}))$ such that $\varphi(\bar{u}) = \bar{v}$ if and only if $\bar{u}$ and $\bar{v}$ have the same type.*

Proof. A similar argument to the following can be found in Lemma 3.15 in [45]. Let $f \colon E \to \mathbb{Z}_{2^q}$ and $\bar{u}, \bar{v} \in A^k$. If $\varphi(\bar{u}) = \bar{v}$ for some automorphism $\varphi \in \mathsf{Aut}((\mathfrak{A}_f, \bar{p}))$, then surely $\bar{u}$ and $\bar{v}$ have the same type.

For the other direction, assume that $\bar{u}$ and $\bar{v}$ have the same type. Then, by definition, there is an isomorphism $\varphi \colon (\mathfrak{A}_f[\mathsf{orig}(\bar{p}\bar{u})], \bar{p}\bar{u}) \to (\mathfrak{A}_f[\mathsf{orig}(\bar{p}\bar{v})], \bar{p}\bar{v})$. Because $\bar{u}$ and $\bar{v}$ have the same type, it follows that $\mathsf{orig}(\bar{p}\bar{u}) = \mathsf{orig}(\bar{p}\bar{v})$ and in particular that φ is an automorphism of $(\mathfrak{A}_f[\mathsf{orig}(\bar{p}\bar{u})], \bar{p})$. We show that this local automorphism extends to an automorphism of $(\mathfrak{A}_f, \bar{p})$.

We extend φ by the identity map on all atoms with origin not in $\mathsf{orig}(\bar{p}\bar{u})$. Then φ is an isomorphism between $(\mathfrak{A}_f, \bar{p})$ and another CFI structure, where all twisted base edges $\mathfrak{e}_1, \dots, \mathfrak{e}_\ell$ leave $\mathsf{orig}(\bar{u})$ and are not incident to $\mathsf{orig}(\bar{p})$ (base edges incident to $\mathsf{orig}(\bar{p})$ cannot be twisted because φ fixes $\bar{p}$). Let N be the neighborhood of $\mathsf{orig}(\bar{u})$ (and thus of $\mathsf{orig}(\bar{v})$). Because G is $(k+m+1)$-connected, there is an $\mathfrak{u}$-$\mathfrak{v}$-path not using $\mathsf{orig}(\bar{p}\bar{u})$ for all $\mathfrak{u}, \mathfrak{v} \in N$ because $G - \mathsf{orig}(\bar{p}\bar{u})$ is still connected when removing at most $|\bar{p}\bar{u}| = k + m < k + m + 1$ many base vertices. Hence, we can use path-isomorphisms to move the twists at every $\mathfrak{e}_i$ all to $\mathfrak{e}_1$. But because φ was an automorphism of $(\mathfrak{A}_f[\mathsf{orig}(\bar{p}\bar{u})], \bar{p})$, the sum of the twists is 0. Hence, composing φ and the mentioned path-isomorphisms forms an automorphism $\psi \in \mathsf{Aut}((\mathfrak{A}_f, \bar{p}))$. Because the selected paths do not use $\mathsf{orig}(\bar{p}\bar{u})$, we still have $\psi(\bar{p}\bar{u}) = \bar{p}\bar{v}$. □

Corollary 7.11. *For every $f \colon E \to \mathbb{Z}_{2^q}$ and every $P \in \mathsf{orb}_k((\mathfrak{A}_f, \bar{p}))$, there is a type such that P contains exactly the tuples of that type.*

Definition 7.12 (Type of an Orbit). For $f \colon E \to \mathbb{Z}_{2^q}$, the **type** of a k-orbit in $(\mathfrak{A}_f, \bar{p})$ is the type of its contained tuples.

Corollary 7.13. *For every pair $f, g \colon E \to \mathbb{Z}_{2^q}$ that does not twist $\mathsf{orig}(\bar{p})$, it holds that*

$$\mathsf{orb}_k((\mathfrak{A}_f, \bar{p})) = \mathsf{orb}_k((\mathfrak{A}_g, \bar{p})) \text{ and}$$
$$\mathsf{Aut}((\mathfrak{A}_f, \bar{p})) = \mathsf{Aut}((\mathfrak{A}_g, \bar{p})).$$

While the orbit partitions of $(\mathfrak{A}_f, \bar{p})$ and $(\mathfrak{A}_g, \bar{p})$ are equal, it is in general not true that an orbit $P \in \mathsf{orb}_k((\mathfrak{A}_f, \bar{p}))$ has the same type in $(\mathfrak{A}_f, \bar{p})$ and in $(\mathfrak{A}_g, \bar{p})$.

Lemma 7.14. *Assume the functions $f, g \colon E \to \mathbb{Z}_{2^q}$ do not twist $\mathsf{orig}(\bar{p})$. Then, for every k-orbit $P \in \mathsf{orb}_k((\mathfrak{A}_f, \bar{p}))$, there is a $Q \in \mathsf{orb}_k((\mathfrak{A}_g, \bar{p}))$ that has the same type.*

Proof. It suffices to consider the case that exactly one base edge $\mathfrak{e} = \{\mathfrak{u}, \mathfrak{v}\}$ is twisted: All twists can be moved to a single base edge using isomorphisms because isomorphisms preserve types and because no base edge contained in $\mathsf{orig}(\bar{p})$ is twisted.

Let $P \in \mathsf{orb}_k((\mathfrak{A}_f, \bar{p}))$. If $\{\mathfrak{u}, \mathfrak{v}\} \not\subseteq \mathsf{orig}(P)$, then P has the same type in $(\mathfrak{A}_f, \bar{p})$ and in $(\mathfrak{A}_g, \bar{p})$. Otherwise, let $\{\mathfrak{u}, \mathfrak{v}\} \subseteq \mathsf{orig}(P)$ and assume w.l.o.g. that $\mathfrak{v} \notin \mathsf{orig}(\bar{p})$ (if $\{\mathfrak{u}, \mathfrak{v}\} \subseteq \mathsf{orig}(\bar{p})$, then $\{\mathfrak{u}, \mathfrak{v}\} \not\subseteq \mathsf{orig}(P)$ because the twisted base edge is not contained in $\mathsf{orig}(\bar{p})$). Furthermore, choose a path $\bar{\mathfrak{s}} = (\mathfrak{u}, \mathfrak{v}, \dots, \mathfrak{w})$, such that $\mathfrak{w} \notin \mathsf{orig}(P)$ and the path, possibly apart from $\mathfrak{u}$, is disjoint from $\mathsf{orig}(\bar{p})$. Such a path exists, because $G - (\mathsf{orig}(\bar{p}) \cup \mathsf{orig}(P))$ is connected (at most $m + k < m + k + 1$ many base vertices are removed) and $\mathfrak{v} \notin \mathsf{orig}(\bar{p})$ by assumption. So we can pick some base vertex $\mathfrak{w}$ that is neither contained in $\mathsf{orig}(P)$ nor in $\mathsf{orig}(\bar{p})$. Now, we move the twist to a base edge incident to $\mathfrak{w}$ with the path-isomorphism $\varphi := \vec{\pi}[g(\mathfrak{e}) - f(\mathfrak{e}), \bar{\mathfrak{s}}]$. Then P has the same type in $(\mathfrak{A}_f, \bar{p})$ as in $\varphi((\mathfrak{A}_g, \bar{p})) = (\varphi(\mathfrak{A}_g), \bar{p})$ because

$$\mathfrak{A}_f[\mathsf{orig}(\bar{p}) \cup \mathsf{orig}(P)] = \varphi(\mathfrak{A}_g)[\mathsf{orig}(\bar{p}) \cup \mathsf{orig}(P)].$$

Because isomorphisms preserve types, there is an orbit $Q \in \mathsf{orb}_k((\mathfrak{A}_g, \bar{p}))$ with the same type in $(\mathfrak{A}_g, \bar{p})$ as P has in $(\mathfrak{A}_f, \bar{p})$. □

Lemma 7.15. *Let $f \colon E \to \mathbb{Z}_{2^q}$ and $P \in \mathsf{orb}_k((\mathfrak{A}_f, \bar{p}))$. Then the permutation group Γ on P induced by $\mathsf{Aut}((\mathfrak{A}_f, \bar{p}))$ is a regular abelian 2-group.*

Proof. We first argue that the automorphism group of a gadget is a regular abelian 2-group. Recall that the atoms of a gadget for the base vertex $\mathfrak{u} \in V$ are defined as $A_\mathfrak{u} = \{u\bar{a} \mid \bar{a} \in \mathbb{Z}_{2^q}^{N_G(\mathfrak{u})}, \sum a = 0\}$. So $|A_\mathfrak{u}| = (2^q)^{d-1}$, where d is the degree of $\mathfrak{u}$. We saw in Section 7.3.1 that the automorphism group of a gadget is transitive. We already argued that the automorphism group is isomorphic to $\{\bar{a} \in \mathbb{Z}_{2^q}^d \mid \sum \bar{a} = 0\}$. Thus, the automorphism group is a 2-group and has order $(2^q)^{d-1}$. Hence, it is a regular abelian 2-group.

The claim for k-orbits follows from the case of a gadget: The group Γ is a subgroup of the direct product of the automorphism groups of the gadgets of $\mathsf{orig}(P)$. That is, Γ is an abelian 2-group. By definition of a k-orbit, Γ is transitive. For regularity, note that a gadget is partitioned into singleton orbits once one atom of the gadget is fixed (cf. Lemma 3.13 in [45]). So if we fix a $\bar{u} \in P$, all gadgets in the origin of $\bar{u}$ are fixed. Hence, if an automorphism φ maps $\bar{u}$ to $\bar{v}$, then its action on P is fixed, i.e., there is exactly one permutation in Γ that maps $\bar{u}$ to $\bar{v}$. Hence, $|\Gamma| = |P|$ and Γ is regular. $\square$

7.3.3 Composition of Orbits

Composing k-orbits out of k'-orbits for $k' < k$ will play a special role later. We further analyze the structure of k-orbits and identify cases in which such a composition in possible. As in the previous section, let $q, k, m \in \mathbb{N}$ and $G = (V, E, \leq)$ be a $(k+m+1)$-connected base graph, denote for $f\colon E \to \mathbb{Z}_{2^q}$ by $\mathfrak{A}_f$ the CFI structure $\mathsf{CFI}_{\mathbb{Z}_{2^q}}(G, f)$ with universe A, and let $\bar{p} \in A^m$.

Let $\bar{u} \in A^k$ and $\mathsf{orig}(\bar{u})$ (viewed as a set) be partitioned into M and N. We now introduce notation for splitting $\bar{u}$ into its parts belonging to M and N and for recovering $\bar{u}$ from these two parts again.

1. The tuple $\bar{u}_M$ obtained from $\bar{u}$ by deleting all entries whose origin is not in M (respectively for N), is
$$\bar{u}_M := \bar{u}|_{\{i \in [k] \mid \mathsf{orig}(u_i) \in M\}}.$$

2. We define a concatenation operation for a permutation σ of $[k]$ as follows:
$$\bar{u}_M \cdot_\sigma \bar{u}_N := \sigma(\bar{u}_M \bar{u}_N).$$
For a suitable σ we have $\bar{u} = \bar{u}_M \cdot_\sigma \bar{u}_N$. In this chapter, we are only interested in permutations satisfying the former equation. Then σ is almost always fixed by the context and we use the juxtaposition $\bar{u}_M\bar{u}_N$. *We will never refer with $\bar{u}_M\bar{u}_N$ to ordinary concatenation.*

3. We define similar operations for orbits: For $P \in \mathsf{orb}_k((\mathfrak{A}, \bar{p}))$ we set
$$P|_M := \left\{ \bar{u}_M \;\middle|\; \bar{u} \in P \right\},$$
$$P|_M \times_\sigma P|_N := \left\{ \bar{u}_M \cdot_\sigma \bar{u}_N \;\middle|\; \bar{u}_M \in P|_M, \bar{u}_N \in P|_N \right\},$$
and omit σ if clear from the context. This intentionally overloads notation. Because the tuples in P are indexed by $[k]$, the sets $P|_M$ and $P|_K$ for $M \subseteq \mathsf{orig}(P) \subseteq V$ and $K \subseteq [k]$ can always be distinguished.

We also use this notation if M and N are sets of sets, such that $\mathsf{orig}(\bar{u})$ is partitioned into $\bigcup M$ and $\bigcup N$.

Definition 7.16 (Components of Tuples and Orbits)**.** Assume $f\colon E \to \mathbb{Z}_{2^q}$, $\bar{u} \in A^k$, and $N \subseteq \mathsf{orig}(\bar{u})$. We call N a **component** of $\bar{u}$ if N is a connected component of $G[\mathsf{orig}(\bar{u})]$. We call $\bar{u}$ **disconnected** if it has more than one component. Likewise, we call a k-orbit $P \in \mathsf{orb}_k((\mathfrak{A}_f, \bar{p}))$ is disconnected if P contains some (and thus only) disconnected tuples. A set $N \subseteq \mathsf{orig}(P)$ is a component of P if N is a connected component of $G[\mathsf{orig}(P)]$.

If a k-orbit P is disconnected, then we can split P into multiple k'-orbits for $k' < k$ as follows.

Lemma 7.17. *Let $f\colon E \to \mathbb{Z}_{2^q}$, $P \in \mathsf{orb}_k((\mathfrak{A}_f, \bar{p}))$, and the components of P be partitioned into M and N. Then*

(a) $P = P|_M \times P|_N$,

(b) $P|_M \in \mathsf{orb}_{k_M}((\mathfrak{A}_f, \bar{p}))$, *and*

(c) $P|_N \in \mathsf{orb}_{k_N}((\mathfrak{A}_f, \bar{p}))$

for suitable k_M and k_N such that $k_M + k_N = k$.

Proof. The claim can easily be seen by Corollary 7.11. Because M and N are sets of components, the type of $\bar{u} \in P$ is given by the disjoint union of the types of $\bar{u}_M$ and $\bar{u}_N$ (even if $\mathsf{orig}(\bar{p})$ overlaps with M and N because $\bar{p}$ has to be fixed by every automorphism). $\square$

Next, we show how to obtain k'-orbits from k-orbits with $k' < k$ by fixing an atom.

Lemma 7.18. *Let $f\colon E \to \mathbb{Z}_{2^q}$, $P \in \mathsf{orb}_k((\mathfrak{A}_f, \bar{p}))$, $K \subseteq [k]$, and $\mathsf{orig}(P|_K) = \{\mathfrak{w}\}$. For all $\bar{v} \in A^{|K|}$ and $w \in A$ such that $\mathsf{orig}(\bar{v}) = \{\mathfrak{w}\}$ and $\mathsf{orig}(w) = \mathfrak{w}$, the set*

$$Q := \left\{ \bar{u}|_{[k]\setminus K} \;\middle|\; \bar{u} \in P, \bar{u}|_K = \bar{v} \right\} \text{ satisfies}$$
$$Q \in \mathsf{orb}_{k-|K|}((\mathfrak{A}_f, \bar{p}w)) \cup \{\emptyset\}.$$

If $\bar{v}$ has the same type as $\bar{u}|_K$ for some (and thus every) $\bar{u} \in P$, then $Q \neq \emptyset$.

Proof. We assume up to reordering that $K = [|K|]$. Let $\bar{v} \in A^{|K|}$ such that $\mathsf{orig}(\bar{v}) = \{\mathfrak{w}\}$. Every atom v_i forms a singleton orbit in $\mathsf{orb}_1((\mathfrak{A}_f, w))$ and in particular in $\mathsf{orb}_1((\mathfrak{A}_f, \bar{p}w))$ because v_i and w have the same origin $\mathfrak{w}$ (all atoms with origin $\mathfrak{w}$ can be distinguished by their distances to w in the $C_{u,v}$ relation, cf. Lemma 3.13 in [45]). So it holds that

$$\mathsf{Aut}((\mathfrak{A}_f, \bar{p}\bar{v})) = \mathsf{Aut}((\mathfrak{A}_f, \bar{p}w)).$$

Assume that $Q \neq \emptyset$. Because P is an orbit, if $\bar{v}\bar{u}, \bar{v}\bar{u}' \in P$, then there is an automorphism $\varphi \in \mathsf{Aut}((\mathfrak{A}_f, \bar{p}))$ such that $\varphi(\bar{v}\bar{u}) = \bar{v}\bar{u}'$. That is, $\varphi \in \mathsf{Aut}((\mathfrak{A}_f, \bar{p}\bar{v})) = \mathsf{Aut}((\mathfrak{A}_f, \bar{p}w))$ and thus Q is a subset of an orbit in $\mathsf{orb}_{k-|K|}((\mathfrak{A}_f, \bar{p}w))$. To show that Q is indeed an orbit, assume $\bar{u} \in Q$ and $\varphi \in \mathsf{Aut}((\mathfrak{A}_f, \bar{p}w)) = \mathsf{Aut}((\mathfrak{A}_f, \bar{p}\bar{v}))$. Because $\bar{u} \in Q$, we have $\bar{v}\bar{u} \in P$ and $\varphi(\bar{v}\bar{u}) = \bar{v}\varphi(\bar{u}) \in P$. Hence, we have $\varphi(\bar{u}) \in Q$ and so $Q \in \mathsf{orb}_{k-|K|}((\mathfrak{A}_f, \bar{p}w))$.

Now assume that there is some $\bar{u} \in P$ such that $\bar{u}|_K$ has the same type as $\bar{v}$. That is, there is an automorphism $\varphi \in \mathsf{Aut}((\mathfrak{A}_f, \bar{p}))$ such that $\varphi(\bar{u})|_K = \bar{v}$ (Lemma 7.10). Hence, $\varphi(\bar{u}) \in P$ and $\varphi(\bar{u})|_{[k]\setminus K} \in Q$. $\square$

Note that the set Q is independent of w (as long as $\mathsf{orig}(w) = \mathfrak{w}$) but the type of Q in $(\mathfrak{A}_f, \bar{p}w)$ depends on w.

Corollary 7.19. *Let $f\colon E \to \mathbb{Z}_{2^q}$, $P \in \mathsf{orb}_k((\mathfrak{A}_f, \bar{p}))$, $i \in [k]$, $\mathsf{orig}(P|_{\{i\}}) = \{\mathfrak{w}\}$, and let $\mathsf{dist}_G(\mathfrak{w}, \mathsf{orig}(\bar{p})) > 1$. For all $v, w \in A$ such that $\mathsf{orig}(v) = \mathsf{orig}(w) = \mathfrak{w}$, it holds that $\{\bar{u}|_{[k]\setminus\{i\}} \mid \bar{u} \in P, u_i = v\} \in \mathsf{orb}_{k-1}((\mathfrak{A}_f, \bar{p}w))$.*

Proof. We apply Lemma 7.18: Because $\mathsf{dist}_G(\mathfrak{w}, \mathsf{orig}(\bar{p})) > 1$, the type of w is the same as the type of every v with origin $\mathfrak{w}$, in particular the same as u_i for every $\bar{u} \in P$. □

7.3.4 Rank Logic on CFI Structures

In this section we refine a result of [45] and show that on CFI structures over $\mathbb{Z}_{2^q}$ the uniform rank logic IFPC+R has the same expressiveness as the rank logic IFPC+R$_{\{2\}}$ that only has the rank operator over $\mathbb{F}_2$.

Definition 7.20. For a class of base graphs $\mathcal{K}$,

$$\mathsf{CFI}_{2^\omega}(\mathcal{K}) := \left\{ \mathsf{CFI}_{2^q}(G, f) \;\middle|\; q \in \mathbb{N}, G = (V, E, \leq) \in \mathcal{K}, f\colon E \to \mathbb{Z}_{2^q} \right\}$$

is the class of all CFI structures over $\mathcal{K}$.

Lemma 7.21. *Let $\mathcal{K}$ be a class of base graphs. For every* IFPC+R*-formula Φ, there is an* IFPC+R$_{\{2\}}$*-formula Ψ that is equivalent to Φ on $\mathsf{CFI}_{2^\omega}(\mathcal{K})$.*

Proof. Let **solvability logic** IFPC+S be the extension of IFPC by the uniform solvability quantifier slv [45]. If $s(\bar{x}, \bar{y})$ is a numeric term and t is a closed numeric term, then

$$\mathsf{slv}(\bar{x}, \bar{y}).\ (s, t)$$

is a formula. Similar to the rank operator rk, the numeric term t defines a number p. If p is prime, then the solvability quantifier is satisfied if the linear system $M_s^{\mathfrak{A}} x = 1$ is solvable over $\mathbb{F}_p$. If otherwise p is not prime, then the operator is not satisfied. Let IFPC+S$_\Omega$ be the extension of IFPC by solvability quantifiers slv_p for each fixed field $\mathbb{F}_p$ with $p \in \Omega$. We again omit parameters for readability.

Grädel and Pakusa [45] give a translation of IFPC+R$_\Omega$-formulas to IFPC-formulas equivalent on CFI structures over $\mathbb{F}_2$ for every set of primes Ω satisfying $2 \notin \Omega$. The crucial point in their proofs is that the automorphism groups of these CFI structures are abelian 2-groups and that their k-orbits can be defined and ordered in IFPC, that is, there is an IFPC-definable total preorder on all k-tuples whose equivalence classes coincide with the k-orbits. Their construction is not specific to $\mathbb{F}_2$ but works generally for $\mathbb{F}_p$ whenever $p \notin \Omega$ and p is the characteristic of the CFI structures. These assumptions are made explicit in Section 3.2 in [45]. Hence, the arguments also work for CFI structures over $\mathbb{Z}_{2^q}$ instead of $\mathbb{F}_2$. In [45], the authors use solvability logic as an intermediate step and first show that, for all sets of primes Ω (even with $2 \in \Omega$), it holds that IFPC+R$_\Omega$ = IFPC+S$_\Omega$ on $\mathsf{CFI}_{2^\omega}(\mathcal{K})$ (Lemma 3.7 in [45]). This reduction works as well for the uniform case and shows IFPC+R = IFPC+S on $\mathsf{CFI}_{2^\omega}(\mathcal{K})$.

The second step in [45] is a recursive translation of IFPC+R$_\Omega$-formulas to IFPC-formulas if $2 \notin \Omega$ (Lemmas 3.4 to 3.6 in [45]). For every IFPC-term s, the solvability quantifier $\Psi = \mathsf{slv}_p(\bar{x}, \bar{y}).\, s$ over $\mathbb{F}_p$ can be simulated in IFPC by computing the rank of the matrix $M := M_s^{\mathfrak{A}}$ orbit-wise. This is expressible in IFPC because the automorphism group is a 2-group and $p \neq 2$. This process works as follows: There is an IFPC-formula that, for every prime p and every term s, exploits the orbits of the structure to define a matrix E such that $Mx = 1$ is solvable if and only if $(M \cdot E)x = 1$ is solvable (Lemma 3.6 in [45]). Now, E is defined such that the columns of $M \cdot E$ are totally ordered and thus the solution can be obtained in IFPC.

Now, we translate an IFPC+S$_\Omega$-formula (respectively term) with $2 \in \Omega$ recursively into an IFPC+S$_{\{2\}}$-formula (respectively term). Again consider a solvability quantifier $\Psi = \mathsf{slv}_p(\bar{x}, \bar{y}).\, s$. If $p = 2$, then we recurse on s but do not replace the solvability quantifier. If otherwise $p \neq 2$, then we recurse on s and obtain an IFPC+S$_{\{2\}}$-term equivalent to s, define the matrix E with the IFPC-formula from above, and construct a formula defining whether $M \cdot E = 1$ is solvable. Because this check can be done in IFPC and M is defined by an IFPC+S$_{\{2\}}$-term, we obtain an IFPC+S$_{\{2\}}$-formula equivalent to Ψ.

We finally deal with the case of an IFPC+S formula, where the prime is defined by a numeric term t. Checking an ordered equation system for solvability is IFPC-definable when the prime is given by a term, too. Let $\Psi = \mathsf{slv}(\bar{x}, \bar{y}).\ (s, t)$ be a uniform solvability quantifier. Let Ψ_2 be the formula obtained for $\mathsf{slv}_2(\bar{x}, \bar{y}).\ s$ in the former case and $\Psi_{\neq 2}$ be the formula for the case $p \neq 2$, where we already used t to obtain the prime. Indeed, $\Psi_{\neq 2}$ is independent of p because defining the matrix E is independent of p and checking the linear equation system for consistency is already done using the prime-defining term t. Then the uniform solvability quantifier Ψ is equivalent to the IFPC+S$_{\{2\}}$-formula

$$(t = 2 \rightarrow \Psi_2) \wedge (t \neq 2 \rightarrow \Psi_{\neq 2}).$$

Obviously, an IFPC+S$_{\{2\}}$-formula can be translated back into an IFPC+R$_{\{2\}}$-formula. □

7.4 Matrices over CFI Structures

In the invertible-map game, Duplicator has to partition the $2k$-tuples of CFI structures and to provide a similarity matrix. For our arguments, we would like Duplicator to play with the $2k$-orbit partitions. To construct the required similarity matrices, we will now develop a criterion for invertibility of matrices over $\mathbb{F}_2$ and we will show that this criterion is preserved by matrix multiplication.

Let $q, k, m \in \mathbb{N}$ and $G = (V, E, \leq)$ be a $(k + m + 1)$-connected base graph. The connectivity is needed to apply the lemmas of Section 7.3.2. Again, we denote for a function $f\colon E \to \mathbb{Z}_{2^q}$ by $\mathfrak{A}_f$ the CFI structure $\mathsf{CFI}_{\mathbb{Z}_{2^q}}(G, f)$ with universe A (which is the same for every $f\colon E \to \mathbb{Z}_{2^q}$). Let $\bar{p} \in A^m$ be arbitrary but fixed in this section.

Definition 7.22 (Blurring the Twist). For $f, g\colon E \to \mathbb{Z}_{2^q}$ not twisting $\mathsf{orig}(\bar{p})$, an $A^k \times A^k$ matrix S over $\mathbb{F}_2$ k-**blurs the twist** between $(\mathfrak{A}_f, \bar{p})$ and $(\mathfrak{A}_g, \bar{p})$ if S is invertible and $\chi^P \cdot S = S \cdot \chi^Q$ for all $P \in \mathsf{orb}_{2k}((\mathfrak{A}_f, \bar{p}))$ and $Q \in \mathsf{orb}_{2k}((\mathfrak{A}_g, \bar{p}))$ that are of the same type.

Note that, by Corollary 7.11, two different orbits have different types. Also note that, by Lemma 7.14, for each $P \in \mathsf{orb}_{2k}((\mathfrak{A}_f, \bar{p}))$, there is a $Q \in \mathsf{orb}_{2k}((\mathfrak{A}_g, \bar{p}))$ of the same type. So there is indeed a type-preserving bijection between the orbits and Duplicator can use the matrix S in the invertible-map game. Because S is invertible, $\chi^P \cdot S = S \cdot \chi^Q$ is equivalent to $\chi^P = S \cdot \chi^Q \cdot S^{-1}$. Showing the former has the benefit that we do not need the inverse S^{-1}.

Lemma 7.23. *Let $f, g, h \colon E \to \mathbb{Z}_{2^q}$ pairwise not twist $\mathsf{orig}(\bar{p})$ and S, T be $A^k \times A^k$ matrices over $\mathbb{F}_2$. If S blurs the twist between $(\mathfrak{A}_f, \bar{p})$ and $(\mathfrak{A}_g, \bar{p})$ and T blurs the twist between $(\mathfrak{A}_g, \bar{p})$ and $(\mathfrak{A}_h, \bar{p})$, then $S \cdot T$ blurs the twist between $(\mathfrak{A}_f, \bar{p})$ and $(\mathfrak{A}_h, \bar{p})$.*

Proof. Let $P \in \mathsf{orb}_{2k}((\mathfrak{A}_f, \bar{p}))$, $Q \in \mathsf{orb}_{2k}((\mathfrak{A}_g, \bar{p}))$, and $R \in \mathsf{orb}_{2k}((\mathfrak{A}_h, \bar{p}))$ be of the same type. Recall that given P, the orbits Q and R are determined uniquely (Corollary 7.11). Then $\chi^P \cdot S \cdot T = S \cdot \chi^Q \cdot T = S \cdot T \cdot \chi^R$. □

Now we want to develop combinatorial conditions ensuring that an $A^k \times A^k$ matrix S over $\mathbb{F}_2$ is invertible. The k-orbits (for given $f, g \colon E \to \mathbb{Z}_{2^q}$) partition S into a block matrix. Each $P \in \mathsf{orb}_k((\mathfrak{A}_f, \bar{p}))$ corresponds to a subset of the rows of S and each $Q \in \mathsf{orb}_k((\mathfrak{A}_g, \bar{p}))$ corresponds to a subset of the columns of S. We denote by $S_{P \times Q}$ the corresponding submatrix of S.

Definition 7.24 (Orbit-Diagonal Matrix). For $f, g \colon E \to \mathbb{F}_2$ not twisting $\mathsf{orig}(\bar{p})$, we call an $A^k \times A^k$ matrix S over $\mathbb{F}_2$ **orbit-diagonal** over $(\mathfrak{A}_f, \bar{p})$ and $(\mathfrak{A}_g, \bar{p})$ if, for every $P \in \mathsf{orb}_k((\mathfrak{A}_f, \bar{p}))$ and every $Q \in \mathsf{orb}_k((\mathfrak{A}_g, \bar{p}))$, it holds that if $S_{P \times Q} \neq \mathbb{0}$, then P has the same type in $(\mathfrak{A}_f, \bar{p})$ as Q has in $(\mathfrak{A}_g, \bar{p})$.

We have seen that, for every $P \in \mathsf{orb}_k((\mathfrak{A}_f, \bar{p}))$, there is exactly one $Q \in \mathsf{orb}_k((\mathfrak{A}_g, \bar{p}))$ of the same type. So orbit-diagonal matrices are block-diagonal matrices, where orbits of the same type form the nonzero blocks. A permutation σ of A^k is applied to an $A^k \times A^k$ matrix S in the natural way: $(\sigma(S))(\bar{u}, \bar{v}) = S(\sigma(\bar{u}), \sigma(\bar{v}))$. Of particular interest are automorphisms.

Definition 7.25 (Orbit-Invariant Matrix). For $f, g \colon E \to \mathbb{Z}_{2^q}$ that do not twist $\mathsf{orig}(\bar{p})$, an $A^k \times A^k$ matrix S over $\mathbb{F}_2$ is called **orbit-invariant** over $(\mathfrak{A}_f, \bar{p})$ and $(\mathfrak{A}_g, \bar{p})$ if, for all $P \in \mathsf{orb}_k((\mathfrak{A}_f, \bar{p}))$, $Q \in \mathsf{orb}_k((\mathfrak{A}_f, \bar{p}))$, and $\varphi \in \mathsf{Aut}((\mathfrak{A}_f, \bar{p})) = \mathsf{Aut}((\mathfrak{A}_g, \bar{p}))$ (cf. Corollary 7.13), the matrix S satisfies $\varphi(S_{P \times Q}) = S_{P \times Q}$.

Lemma 7.26. *Let $f, g, h \colon E \to \mathbb{Z}_{2^q}$ not twist $\mathsf{orig}(\bar{p})$ and S, T be $A^k \times A^k$ matrices over $\mathbb{F}_2$. If S is orbit-diagonal and orbit-invariant over $(\mathfrak{A}_f, \bar{p})$ and $(\mathfrak{A}_g, \bar{p})$ and T is orbit-diagonal and orbit-invariant over $(\mathfrak{A}_g, \bar{p})$ and $(\mathfrak{A}_h, \bar{p})$, then $S \cdot T$ is orbit-diagonal and orbit-invariant over $(\mathfrak{A}_f, \bar{p})$ and $(\mathfrak{A}_h, \bar{p})$.*

Proof. It is clear that $S \cdot T$ is orbit-diagonal over $(\mathfrak{A}_f, \bar{p})$ and $(\mathfrak{A}_h, \bar{p})$. For k-orbits $P \in \mathsf{orb}_k((\mathfrak{A}_f, \bar{p}))$, $Q \in \mathsf{orb}_k((\mathfrak{A}_g, \bar{p}))$, and $R \in \mathsf{orb}_k((\mathfrak{A}_h, \bar{p}))$ of the same type it holds that

$$(S \cdot T)_{P \times R}(\bar{u}, \bar{w}) = \sum_{\bar{v} \in Q} S_{P \times Q}(\bar{u}, \bar{v}) \cdot T_{Q \times R}(\bar{v}, \bar{w}).$$

Let $\varphi \in \mathsf{Aut}((\mathfrak{A}, \bar{p}))$. Then

$$\begin{aligned}(\varphi(S \cdot T))_{P\times R}(\bar{u}, \bar{w}) &= (S \cdot T)_{P\times R}(\varphi(\bar{u}), \varphi(\bar{w}))\\ &= \sum_{\bar{v}\in Q} S_{P\times Q}(\varphi(\bar{u}), \bar{v}) \cdot T_{Q\times R}(\bar{v}, \varphi(\bar{w}))\\ &= \sum_{\bar{v}\in Q} S_{P\times Q}(\varphi(\bar{u}), \varphi(\bar{v})) \cdot T_{Q\times R}(\varphi(\bar{v}), \varphi(\bar{w}))\\ &= \sum_{\bar{v}\in Q} S_{P\times Q}(\bar{u}, \bar{v}) \cdot T_{Q\times R}(\bar{v}, \bar{w})\\ &= (S \cdot T)_{P\times R}(\bar{u}, \bar{w}).\end{aligned}$$

Applying φ to $\bar{v}$ is valid because φ is a permutation of Q and thus only permutes the summands. Then $S_{P\times Q}(\varphi(\bar{u}), \varphi(\bar{v})) = S_{P\times Q}(\bar{u}, \bar{v})$ because S is orbit-invariant (and likewise for T). □

Definition 7.27 (Odd-Filled Matrix). A matrix over $\mathbb{F}_2$ is called **odd-filled** if every row contains an odd number of ones.

Lemma 7.28. *If two $A^k \times A^k$ matrices S and T over $\mathbb{F}_2$ are odd-filled, then so is $S \cdot T$.*

Proof. Let $R = S \cdot T$ and denote by $r_{\bar{u}}$ and $t_{\bar{v}}$ the rows of R and T indexed by $\bar{u} \in A^k$ and $\bar{v} \in A^k$. Then

$$r_{\bar{u}} = \sum_{\bar{v}\in A^k} S(\bar{u}, \bar{v}) \cdot t_{\bar{v}}.$$

The number of ones modulo 2 is given by

$$\sum r_{\bar{u}} = \sum_{\bar{v}\in A^k} S(\bar{u}, \bar{v}) \cdot \sum t_{\bar{v}}.$$

Now $S(\bar{u}, \bar{v}) = 1$ for an odd number of $\bar{v} \in A^k$, because S is odd-filled. Hence, $\sum r_{\bar{u}}$ is the sum of an odd number of $\sum t_{\bar{v}}$, of which each is odd because T is odd-filled. So $\sum r_{\bar{u}} = 1$ and $r_{\bar{u}}$ contains an odd number of ones. □

Lemma 7.29. *Let $f, g\colon E \to \mathbb{Z}_{2^q}$ not twist $\mathsf{orig}(\bar{p})$ and S be an $A^k \times A^k$ matrix over $\mathbb{F}_2$. If S is odd-filled and both orbit-diagonal and orbit-invariant over $(\mathfrak{A}_f, \bar{p})$ and $(\mathfrak{A}_g, \bar{p})$, then every column of S contains an odd number of ones.*

Proof. Consider the block $S_{P\times Q}$ for arbitrary $P \in \mathsf{orb}_k((\mathfrak{A}_f, \bar{p}))$ and $Q \in \mathsf{orb}_k((\mathfrak{A}_g, \bar{p}))$ of the same type. Let $P = \{\bar{u}_1, \dots, \bar{u}_n\}$ and $Q = \{\bar{v}_1, \dots, \bar{v}_n\}$. Then consider automorphisms φ_i such that $\varphi_i(\bar{u}_1) = \bar{u}_i$. Because the induced action of $\mathsf{Aut}((\mathfrak{A}, \bar{p}))$ on P (and on Q) is regular (Lemma 7.15), the action of φ_i on P (and so Q) is uniquely determined. W.l.o.g., we consider the column indexed by $\bar{v}_1$: We have

$$S(\bar{u}_i, \bar{v}_1) = \varphi_i^{-1}(S)(\bar{u}_i, \bar{v}_1) = S(\bar{u}_1, \varphi_i^{-1}(\bar{v}_1))$$

because S is orbit-invariant. So the column indexed by $\bar{v}_1$ contains exactly the entries of the row indexed by $\bar{u}_1$. That is, the number of ones in every column is odd. □

Lemma 7.30. *Let $\bar{a} \in \mathbb{F}_2^N$ for some finite set N and $\Gamma < \mathsf{Sym}(N)$ be a regular and abelian 2-group. If the number of ones in $\bar{a}$ is odd, then the set $B := \{\sigma(\bar{a}) \mid \sigma \in \Gamma\}$ is a basis of $\mathbb{F}_2^N$.*

Proof. Assume w.l.o.g. that $N = [\ell]$ for some $\ell \in \mathbb{N}$ and let $W \subseteq \mathbb{F}_2^N$ be the linear space spanned by B. Because Γ is regular, it consists of ℓ many permutations $\Gamma = \{\sigma_1, \dots, \sigma_\ell\}$ such that $\sigma_i(1) = i$ for all $i \in [\ell]$. By definition, W is invariant under permutations of Γ. In coding theory, such a linear space is called an abelian code. It is known that W can be identified with an ideal of the group algebra $\mathbb{F}_2[\Gamma]$ [15], which is the set of formal sums

$$\left\{ \sum_{g \in \Gamma} b_g g \;\middle|\; b_g \in \mathbb{F}_2 \right\}.$$

This set is naturally an $\mathbb{F}_2$-vector space indexed by Γ. To turn it into an $\mathbb{F}_2$-algebra, multiplication is defined via

$$\left(\sum_{g \in \Gamma} b_g g\right) \cdot \left(\sum_{g \in \Gamma} c_g g\right) := \sum_{g,h \in \Gamma} (b_g \cdot c_h)(g \cdot h).$$

A nonempty set $I \subseteq \mathbb{F}_2[\Gamma]$ is a (left) ideal of the algebra $\mathbb{F}_2[\Gamma]$ if $g + h \in I$ for all $g, h \in I$ and $g \cdot h \in I$ for all $g \in \mathbb{F}_2[\Gamma]$ and $h \in I$, i.e., $\mathbb{F}_2[\Gamma] \cdot I = I$. The abelian code W is identified with its image under the linear map $(b_1, \dots, b_\ell) \mapsto \sum_{i=1}^{\ell} b_i \sigma_i$ for every $\bar{b} \in W$. This image is an ideal of $\mathbb{F}_2[\Gamma]$.

Let $I \subseteq \mathbb{F}_2[\Gamma]$ be the corresponding ideal of W and let the number of ones of $\bar{a} \in W$ be odd. Because Γ is a 2-group, there is a $k \in \mathbb{N}$ such that $\sigma_i^{(2^k)} = 1_\Gamma$ for all $i \in [\ell]$. Because Γ is abelian and we consider $\mathbb{F}_2$, we have $(b\sigma_i)(c\sigma_j) + (c\sigma_j)(b\sigma_i) = 2(b\sigma_i)(c\sigma_j) = 0$. So $(b\sigma_i + c\sigma_j)^2 = (b\sigma_i)^2 + (c\sigma_j)^2$ and $(b\sigma_i + c\sigma_j)^{(2^k)} = (b\sigma_i)^{(2^k)} + (c\sigma_j)^{(2^k)}$. It follows that

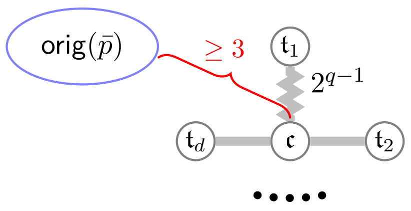
$$\left(\sum_{i=1}^{\ell} a_i \sigma_i\right)^{(2^k)} = \sum_{i=1}^{\ell} (a_i \sigma_i)^{(2^k)} = \sum_{i=1}^{\ell} a_i^{(2^k)} 1_\Gamma = \sum_{i=1}^{\ell} a_i 1_\Gamma = 1_\Gamma.$$

The last step holds because the number of ones in $\bar{a}$ is odd. So $\sum_{i=1}^{\ell} a_i \sigma_i$ is a unit with inverse $(\sum_{i=1}^{\ell} a_i \sigma_i)^{2^k - 1}$. First, $\sum_{i=1}^{\ell} a_i \sigma_i \in I$ because $\bar{a} \in W$. Second, $1_\Gamma \in I$ because the inverse of $\sum_{i=1}^{\ell} a_i \sigma_i \in I$ is clearly contained in $\mathbb{F}_2[\Gamma]$ and $\mathbb{F}_2[\Gamma] \cdot I = I$. Thus, $I = \mathbb{F}_2[\Gamma]$ and $W = \mathbb{F}_2^N$. Finally, B must be a basis of W because $|B| = |N|$. $\square$

Lemma 7.31. *Let $f, g \colon E \to \mathbb{Z}_{2^q}$ not twist $\mathsf{orig}(\bar{p})$ and S be an $A^k \times A^k$ matrix over $\mathbb{F}_2$. If S is odd-filled and both orbit-diagonal and orbit-invariant over $(\mathfrak{A}_f, \bar{p})$ and $(\mathfrak{A}_g, \bar{p})$, then S is invertible.*

Proof. It suffices to show that each block on the diagonal of S is invertible because S is orbit-diagonal. Let $P \in \mathsf{orb}_k((\mathfrak{A}_f, \bar{p}))$ and $Q \in \mathsf{orb}_k((\mathfrak{A}_g, \bar{p}))$ be of the same type. Because S is odd-filled and orbit-diagonal, $S_{P \times Q}$ is also odd-filled. By Lemma 7.15, the action of $\mathsf{Aut}((\mathfrak{A}_f, \bar{p}))$ on P induces a regular and abelian 2-group Γ. By Corollary 7.13, the action of $\mathsf{Aut}((\mathfrak{A}_g, \bar{p}))$ on Q yields the same group Γ. Let $n := |P|$, $P = \{\bar{u}_1, \dots, \bar{u}_n\}$, and s_i be the row of $S_{P \times Q}$ indexed by $\bar{u}_i$. We want to show that $s_i = \varphi_i(s_1)$ for a unique $\varphi_i \in \Gamma$. Each $\varphi \in \Gamma$ acts as a permutation on the entries of each s_i, that is $(\varphi(s_i))(\bar{v}) = s_i(\varphi(\bar{v}))$. Let $\Gamma = \{\varphi_1, \dots, \varphi_n\}$ such that $\varphi_i^{-1}(\bar{u}_1) = \bar{u}_i$ for every $i \in [n]$ (this is possible because Γ is regular). Then

$$(\varphi_i(s_1))(\bar{v}) = S_{P \times Q}(\bar{u}_1, \varphi_i(\bar{v})) = S_{P \times Q}(\varphi_i^{-1}(\bar{u}_1), \bar{v})$$



7.1 The base vertex used to blur the twist in the 1-ary case. The base edge $\{\mathfrak{c}, \mathfrak{t}\}$ in the graph G is twisted by f and g by value 2^{q-1}. The neighborhood of $\mathfrak{c}$ is $N_G(\mathfrak{c}) = \{\mathfrak{t}_1, \dots, \mathfrak{t}_d\}$ with $\mathfrak{t} = \mathfrak{t}_1$. The origin of the parameters $\mathsf{orig}(\bar{p})$ has distance at least 3 to $\mathfrak{c}$.

because S is orbit-invariant. Hence,

$$(\varphi_i(s_1))(\bar{v}) = S_{P\times Q}(\varphi_i^{-1}(\bar{u}_1), \bar{v}) = s_i(\bar{v}),$$

i.e., $\varphi_i(s_1) = s_i$. Finally, $\{\varphi_i(s_1) \mid i \in [n]\} = \{s_1, \dots, s_n\}$ forms a basis of $\mathbb{F}_2^n$ by Lemma 7.30. That is, $S_{P\times Q}$ has full rank and is invertible. □

7.5 The Arity 1 Case

To separate rank logic from PTIME, we want to show that, for every arity k and every number of pebbles $2k+m$, there are two non-isomorphic CFI structures over $\mathbb{Z}_{2^q}$ for a suitable $q \in \mathbb{N}$ for which Duplicator has a winning strategy in the invertible-map game $\mathcal{M}^{2k+m,k,\{2\}}$. This implies IFPC+R$_{\{2\}}$-undefinability of the CFI query by Lemma 7.1 and IFPC+R-undefinability by Lemma 7.21. The most challenging part of constructing winning strategies for Duplicator in the invertible-map game is to provide similarity matrices. Indeed, our goal is to construct matrices blurring the twist. Once we achieve this, it suffices to ensure that the pebbled tuples in both structures always have the same type. This final step is made formal in Section 7.8. Constructing matrices blurring the twists for an arbitrary arity k turns out to be formally intricate and is in particular recursive on the arity. In this section, we start with constructing matrices for arity 1, which serve as a base case for the recursion. We introduce basic techniques that we generalize to higher arities later in Section 7.7.

Let $q \geq 2$, $m \in \mathbb{N}$, $G = (V, E, \leq)$ be an $(m+3)$-connected base graph, $\mathfrak{c} \in V$ be a base vertex of degree d, and $\{\mathfrak{c}, \mathfrak{t}\} \in E$. Let $f, g \colon E \to \mathbb{Z}_{2^q}$ such that $\{\mathfrak{c}, \mathfrak{t}\}$ is the only twisted base edge and $g(\{\mathfrak{c}, \mathfrak{t}\}) = f(\{\mathfrak{c}, \mathfrak{t}\}) + 2^{q-1}$ (cf. Figure 7.1). The number m is the number of pebbles remaining on the structure when Spoiler picks up the $2 = 2k$ many pebbles before Duplicator needs to provide the similarity matrix (we consider arity $k = 1$ in this section). From another perspective, m corresponds to the number of free variables of a rank operator. Set $\mathfrak{A}_f := \mathsf{CFI}_{2^q}(G, f)$ and $\mathfrak{A}_g := \mathsf{CFI}_{2^q}(G, g)$, both with universe A. Let $\bar{p} \in A^m$ such that $\mathsf{dist}_G(\mathfrak{c}, \mathsf{orig}(\bar{p})) \geq 3$, in particular, g and f do not twist $\mathsf{orig}(\bar{p})$. The tuple $\bar{p}$ is the tuple of atoms on which the pebbles remain. It suffices to consider only a single tuple $\bar{p}$ for both structures because we will ensure that the pebbled tuples always have the same type in both structures. Whenever the pebbled tuples have the same type but are not equal, we can consider an isomorphic structure in which we moved the twist to a base edge far apart from the pebbled tuples. Then the tuples are equal and we ensured that the distance between $\mathsf{orig}(\bar{p})$ and the twisted base edge is sufficiently large (details in Section 7.8).

For $\mathfrak{u} \in V$, let $A_{\mathfrak{u}}$ be the set of atoms originating from $\mathfrak{u}$, i.e., the atoms of the gadget for $\mathfrak{u}$. The key idea is to "distribute" the twist among multiple base edges, such that it cannot be detected by Spoiler. For this, we introduce blurrers, the key ingredient to define the desired similarity matrix.

Definition 7.32 (1-ary Blurrer). Let $\Xi \subseteq \mathbb{Z}_{2^q}^d$. For $b \in \mathbb{Z}_{2^q}$ and $j \in [d]$ we define

$$\#_{j,b}(\Xi) := \left|\left\{ \bar{a} \in \Xi \;\middle|\; a_j = b \right\}\right| \bmod 2.$$

The set Ξ is called a (q,d)**-blurrer** if it satisfies

1. $\sum \bar{a} = 0$ for all $a \in \Xi$,
2. $\#_{1,2^{q-1}}(\Xi) = 1$,
3. $\#_{j,0}(\Xi) = 1$ for all $1 < j \leq d$, and
4. $\#_{j,b}(\Xi) = 0$ for all other pairs of $b \in \mathbb{Z}_{2^q}$ and $j \in [d]$.

From now on, we use the letter ξ for elements of a blurrer Ξ. Note that Ξ consists solely of tuples satisfying $\sum \xi = 0$, i.e., we can later turn every $\xi \in \Xi$ into an automorphism. But intuitively, when looking at a single index and summing over all $\xi \in \Xi$, there seems to be a twist at index 1 and no twist at all other indices.

Lemma 7.33. *The size $|\Xi|$ of every (q,d)-blurrer is odd. For every $d \geq 3$, there is a (q,d)-blurrer.*

Proof. By Conditions 2 and 4 of the blurrer, it holds that

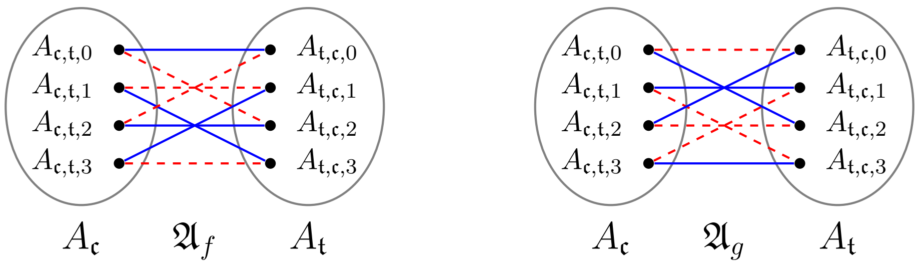
$$|\Xi| = \sum_{b \in \mathbb{Z}_{2^q}} \#_{1,b}(\Xi) = \#_{1,2^{q-1}}(\Xi) + \sum_{b \in \mathbb{Z}_{2^q} \setminus \{2^{q-1}\}} \#_{1,b}(\Xi) = 1 \bmod 2.$$

For $d \geq 3$, set $\Xi := 2^{q-2} \cdot \{(3,0,1,0,\dots,0),(3,1,0,0,\dots,0),(2,1,1,0,\dots,0)\}$. □

Set $\mathbf{P}_j := \mathsf{orb}_j((\mathfrak{A}_f,\bar{p}))$ and $\mathbf{Q}_j := \mathsf{orb}_j((\mathfrak{A}_g,\bar{p}))$ for every $j \in [2]$. For $P \in \mathbf{P}_2$ we set $P_i := P|_i$ for every $i \in [2]$ and likewise for $Q \in \mathbf{Q}_2$. By Corollary 7.11, P_i satisfies $P_i = A_{\mathfrak{u}}$ if $\mathfrak{u} = \mathsf{orig}(P_i)$ and $\mathsf{dist}_G(\mathfrak{u}, \mathsf{orig}(\bar{p})) > 1$. Moreover, every $P \in \mathbf{P}_1$ is also contained in $\mathbf{Q}_1$ and has the same type in $(\mathfrak{A}_f,\bar{p})$ as in $(\mathfrak{A}_g,\bar{p})$.

Let Ξ be a (q,d)-blurrer (note that $\mathfrak{c}$ is of degree $d \geq 3$ because G is $(m+3)$-connected) and $N_G(\mathfrak{c}) = \{\mathfrak{t}_1,\dots,\mathfrak{t}_d\}$ such that $\mathfrak{t}_1 = \mathfrak{t}$ (cf. Figure 7.1). Then we can view $\xi \in \Xi$ also as a tuple $\xi \in \mathbb{Z}_{2^q}^{N_G(\mathfrak{c})}$. Thus, ξ acts on atoms u originating from $\mathfrak{c}$ and we denote this action by $\xi(u)$ (cf. Section 7.3.1 for a definition of the action). Note that every $\xi \in \Xi$ extends to an automorphism of $(\mathfrak{A}_f,\bar{p})$ (and so of $(\mathfrak{A}_g,\bar{p})$): By Corollary 7.11, the gadget of $\mathfrak{c}$ consists of a single orbit because $\mathsf{dist}_G(\mathfrak{c},\bar{p}) \geq 3$, i.e., $A_{\mathfrak{c}} \in \mathbf{P}_1$. We define an $A \times A$ matrix S over $\mathbb{F}_2$, which is orbit-diagonal over $(\mathfrak{A}_f,\bar{p})$ and $(\mathfrak{A}_g,\bar{p})$. We set $S_P := S_{P \times P}$ and define

$$S_P(u,v) := \begin{cases} 1 & \text{if } \mathsf{orig}(P) \neq \mathfrak{c} \text{ and } u = v, \\ 1 & \text{if } \mathsf{orig}(P) = \mathfrak{c} \text{ and } \xi(u) = v \text{ for some } \xi \in \Xi, \\ 0 & \text{otherwise.} \end{cases}$$



$A_{\mathfrak{c},\mathfrak{t},0}$
$A_{\mathfrak{c},\mathfrak{t},1}$
$A_{\mathfrak{c},\mathfrak{t},2}$
$A_{\mathfrak{c},\mathfrak{t},3}$
$A_{\mathfrak{t},\mathfrak{c},0}$
$A_{\mathfrak{t},\mathfrak{c},1}$
$A_{\mathfrak{t},\mathfrak{c},2}$
$A_{\mathfrak{t},\mathfrak{c},3}$
$A_{\mathfrak{c}}$
$\mathfrak{A}_f$
$A_{\mathfrak{t}}$
$A_{\mathfrak{c}}$
$\mathfrak{A}_g$
$A_{\mathfrak{t}}$

7.2 The twist between CFI structures. This figure shows the twist between $\mathfrak{A}_f$ and $\mathfrak{A}_g$. The figure considers $\mathbb{Z}_4$ and assumes that $f(\{\mathfrak{c},\mathfrak{t}\}) = 0$ and $g(\{\mathfrak{c},\mathfrak{t}\}) = 2$. The twisted connection for the base edge $\{\mathfrak{c},\mathfrak{t}\}$ is shown. On the left, there are the two gadgets for the base vertices $\mathfrak{c}$ and $\mathfrak{t}$ in $\mathfrak{A}_f$ and on the right there are the same gadgets in $\mathfrak{A}_g$. Every vertex represents a clique corresponding to the set $A_{\mathfrak{c},\mathfrak{t},c}$ and every base edge a complete bipartite graph (cf. Section 7.3). The relation $R_{E,0}$ is drawn in blue and $R_{E,2}$ in red and dashed style. Restricted to the connection between $\{\mathfrak{c},\mathfrak{t}\}$, we have in $\mathfrak{A}_f$ that $R_{E,0} = E_{\{\mathfrak{c},\mathfrak{t}\},0}$ and $R_{E,2} = E_{\{\mathfrak{c},\mathfrak{t}\},2}$. In $\mathfrak{A}_g$, we have that $R_{E,0} = E_{\{\mathfrak{c},\mathfrak{t}\},2}$ and $R_{E,2} = E_{\{\mathfrak{c},\mathfrak{t}\},0}$.

Of particular interest is the unique 1-orbit $P_\mathfrak{c}$ with origin $\mathfrak{c}$. We have already seen that $P_\mathfrak{c} = A_\mathfrak{c} \in \mathbf{P}_1$, because $\mathsf{dist}_G(\mathfrak{c}, \mathsf{orig}(\bar{p})) \geq 3$ by assumption. For all other orbits $P \in \mathbf{P}_1$, it is easy to see that $S_P = \mathbb{1}$.

Lemma 7.34. *The matrix S is orbit-invariant over $(\mathfrak{A}_f, \bar{p})$ and $(\mathfrak{A}_g, \bar{p})$.*

Proof. Let $P \in \mathbf{P}_1$, $\varphi \in \mathsf{Aut}((\mathfrak{A}_f, \bar{p}))$, $u \in P$, and $v \in Q = P \in \mathbf{Q}_1$. If $P \neq P_\mathfrak{c}$, then clearly $\varphi(S_P) = \varphi(\mathbb{1}) = \mathbb{1} = S_P$. Otherwise, $P = P_\mathfrak{c}$. Because the automorphism group of $\mathfrak{A}_f$ is abelian (Lemma 7.3) and every $\xi \in \Xi$ extends to an automorphism, it holds that $\xi(\varphi(u)) = \varphi(\xi(u))$. So $S_{P_\mathfrak{c}}(\varphi(u), \varphi(v)) = 1$ if and only if $\xi(\varphi(u)) = \varphi(\xi(u)) = \varphi(v)$ for some $\xi \in \Xi$ if and only if $\xi(u) = v$ for some $\xi \in \Xi$, i.e, $S_{P_\mathfrak{c}}(u, v) = 1$. □

Lemma 7.35. *The matrix S is odd-filled.*

Proof. Let $P \in \mathbf{P}_1$. For $P \neq P_\mathfrak{c}$, the number of ones in a row of $S_P = \mathbb{1}$ is one and thus odd. In $S_{P_\mathfrak{c}}$, the number of ones in a row is $|\Xi|$ because $\xi(u) \neq \xi'(u)$ if $\xi \neq \xi'$ (Lemma 7.15) and if $u \in P_\mathfrak{c}$, then $\xi(u) \in P_\mathfrak{c}$ for every $\xi \in \Xi$. From Lemma 7.33 it follows that $|\Xi|$ is odd. □

Corollary 7.36. *The matrix S is invertible.*

Proof. Apply Lemmas 7.31, 7.34, and 7.35. □

We want to define a bijection $\lambda\colon \mathbf{P}_2 \to \mathbf{Q}_2$ such that λ maps an orbit to another orbit of the same type. By Corollary 7.13, we know that $\mathbf{P}_2 = \mathbf{Q}_2$ and, by Lemma 7.14, that a type-preserving bijection exists. Let $P \in \mathbf{P}_2$ with origin $(\mathfrak{u}, \mathfrak{v})$. If $\{\mathfrak{c},\mathfrak{t}\} \neq \{\mathfrak{u},\mathfrak{v}\}$, we set $\lambda(P) := P$. Otherwise if $(\mathfrak{t},\mathfrak{c}) = (\mathfrak{u},\mathfrak{v})$, then P has a different type in $(\mathfrak{A}_f, \bar{p})$ than in $(\mathfrak{A}_g, \bar{p})$: Every atom in P_1 is related to every atom in P_2 via some $R_{E,c}$. By Corollary 7.11, we have that $P = E_{\{\mathfrak{c},\mathfrak{t}\},a}$ for some $a \in \mathbb{Z}_{2^q}$ (recall our assumption $\mathsf{dist}_G(\mathfrak{c}, \bar{p}) \geq 3$ and thus a determines the type of P). We set $\lambda(P) := E_{\{\mathfrak{c},\mathfrak{t}\},a+2^{q-1}}$, which then has the same type in $(\mathfrak{A}_g, \bar{p})$ because of the twist (cf. Figure 7.2). The case of $(\mathfrak{c},\mathfrak{t})$ is analogous.

Lemma 7.37. $\chi^P \cdot S = S \cdot \chi^{\lambda(P)}$ *for every* $P \in \mathbf{P}_2$.

Proof. Let $P \in \mathbf{P}_2$ and $\mathsf{orig}(P) = (\mathfrak{u}, \mathfrak{v})$ and set $Q := \lambda(P)$. Clearly, $P \subseteq P_1 \times P_2$. We also have $P_1 = Q_1$ and $P_2 = Q_2$ (as seen earlier by Corollary 7.11). Then the $P_1 \times P_2$ block is the only nonzero block of χ^P. Because S is orbit-diagonal, $\chi^P \cdot S$ has only one nonzero block, namely the $P_1 \times Q_2$ block, which satisfies

$$(\chi^P \cdot S)_{P_1 \times Q_2} = \chi^P_{P_1 \times P_2} \cdot S_{P_2 \times Q_2}.$$

Likewise, $(S \cdot \chi^Q)_{P_1 \times Q_2} = S_{P_1 \times Q_1} \cdot \chi^Q_{Q_1 \times Q_2}$. Recall that we have set $S_{P_2} = S_{P_2 \times Q_2}$. We identify χ^P with $\chi^P_{P_1 \times P_2}$ and likewise for χ^Q. So we are left to show that $\chi^P \cdot S_{P_2} = S_{Q_1} \cdot \chi^Q$.

(a) Case $\mathfrak{c} \notin \{\mathfrak{u}, \mathfrak{v}\}$: Then $Q = \lambda(P) = P$ and $\chi^P \cdot S_{P_2} = \chi^P \cdot \mathbb{1} = \mathbb{1} \cdot \chi^Q = S_{Q_1} \cdot \chi^Q$.

(b) Case $\mathfrak{u} = \mathfrak{v} = \mathfrak{c}$: Then $Q = \lambda(P) = P$. As already seen, $P_1 = P_2 = Q_1 = Q_2 = P_\mathfrak{c}$. So if $u \in Q_2$, then $\xi^{-1}(u) \in Q_2$ for every $\xi \in \Xi$. We obtain

$$\begin{aligned}(\chi^P \cdot S_{P_2})(u, v) &= \sum_{w \in P_2} \chi^P(u, w) \cdot S_{P_\mathfrak{c}}(w, v) \\ &= \sum_{\xi \in \Xi} \chi^P(u, \xi^{-1}(v)) \\ &= \sum_{\xi \in \Xi} \chi^P(\xi(u), v).\end{aligned}$$

The last step uses that ξ extends to an automorphism and thus we have that $\chi^P(u, \xi^{-1}(v)) = \xi(\chi^P)(u, \xi^{-1}(v)) = \chi^P(\xi(u), v)$. The reverse direction is similar:

$$\begin{aligned}&\sum_{\xi \in \Xi} \chi^P(\xi(u), v) \\ &= \sum_{\xi \in \Xi} \chi^Q(\xi(u), v) \\ &= \sum_{w \in Q_1} S_{Q_1}(u, w) \cdot \chi^Q(w, v) \\ &= (S_{Q_1} \cdot \chi^Q)(u, v).\end{aligned}$$

(c) Case $\mathfrak{v} = \mathfrak{c}$ and $\{\mathfrak{u}, \mathfrak{c}\} \in E$ (the case $\mathfrak{u} = \mathfrak{c}$ and $\{\mathfrak{c}, \mathfrak{v}\} \in E$ is analogous): Again $P_2 = P_\mathfrak{c}$. We have

$$\begin{aligned}(\chi^P \cdot S_{P_2})(u, v) &= \sum_{w \in P_\mathfrak{c}} \chi^P(u, w) \cdot S_{P_\mathfrak{c}}(w, v) \\ &= \sum_{\xi \in \Xi} \chi^P(u, \xi^{-1}(v)) \\ &= \begin{cases} 1 & \text{if } |\{\xi \in \Xi \mid (u, \xi^{-1}(v)) \in P\}| \text{ is odd}, \\ 0 & \text{otherwise.}\end{cases}\end{aligned}$$

Let $P = E_{\{\mathfrak{u},\mathfrak{c}\},a}$ for some $a \in \mathbb{Z}_{2^q}$ (cf. the definition of λ) and $(u, v) \in E_{\{\mathfrak{u},\mathfrak{c}\},b}$ for some $b \in \mathbb{Z}_{2^q}$. Then, by definition of $E_{\{\mathfrak{u},\mathfrak{c}\},b}$, it holds that $u(\mathfrak{c}) + v(\mathfrak{u}) = b$. Let $i \in [d]$ such that $\mathfrak{u} = \mathfrak{t}_i$ (recall that $N_G(\mathfrak{c}) = \{\mathfrak{t}_1, \ldots, \mathfrak{t}_d\}$ and $\mathfrak{t}_1 = \mathfrak{t}$). For every

$\xi \in \Xi$, it holds that $(u, \xi^{-1}(v)) \in P = E_{(\mathfrak{u},\mathfrak{c}),a}$ if and only if $u(\mathfrak{c}) + \xi^{-1}(v)(\mathfrak{u}) = a$ if and only if $\xi(i) = b - a$ because

$$u(\mathfrak{c}) + \xi^{-1}(v)(\mathfrak{u}) = u(\mathfrak{c}) + v(\mathfrak{u}) - \xi(i) = b - \xi(i).$$

We see that

$$\left|\left\{ \xi \in \Xi \;\middle|\; (u, \xi^{-1}(v)) \in P \right\}\right| = \#_{i,b-a}(\Xi).$$

Set $c := 2^{q-1}$ if $i = 1$ (and so $\mathfrak{u} = \mathfrak{t}$) and $c := 0$ otherwise. Using the properties of a blurrer, we see that $\#_{i,b-a}(\Xi) = 1$ holds if and only if $b - a = c$. It follows that

$$(\chi^P \cdot S_{P_2})(u, v) = \begin{cases} 1 & \text{if } b - a = c, \\ 0 & \text{otherwise.} \end{cases}$$

- If $i \neq 1$ (so $\mathfrak{u} \neq \mathfrak{t}'$), then $c = 0$ and $(\chi^P \cdot S_{P_2})(u, v) = 1$ if and only if $b = a$, but that holds if and only if $(u, v) \in P$. So

  $$\chi^P \cdot S_{P_2} = \chi^P = \mathbb{1} \cdot \chi^Q = S_{Q_1} \cdot \chi^Q$$

  because $Q = \lambda(P) = P$.
- If $i = 1$ (so $\mathfrak{u} = \mathfrak{t}$), then $(\chi^P \cdot S_{P_2})(u, v) = 1$ if and only if $b - a = 2^{q-1}$, i.e., $a + 2^{q-1} = b$. But that holds by definition of λ if and only if

  $$(u, v) \in Q = \lambda(P) = E_{(\mathfrak{u},\mathfrak{c}),a+2^{q-1}}$$

  and so

  $$\chi^P \cdot S_{P_2} = \mathbb{1} \cdot \chi^Q = S_{Q_1} \cdot \chi^Q.$$

(d) Case $\mathfrak{v} = \mathfrak{c}$ and $\{\mathfrak{u}, \mathfrak{c}\} \notin E$ (the case $\mathfrak{u} = \mathfrak{c}$ and $\{\mathfrak{c}, \mathfrak{v}\} \notin E$ is analogous): By the assumption that $\mathsf{dist}_G(\mathfrak{c}, \mathsf{orig}(\bar{p})) \geq 3$, the type of (u, v) and (u, v') for $u \in A_{\mathfrak{u}}$ and $v, v' \in A_{\mathfrak{c}}$ is equal. So $(u, v) \in P$ if and only if $(u, v') \in P$ by Corollary 7.11. In particular, $(u, v) \in P$ if and only if $(u, \xi^{-1}(v)) \in P$ for every $\xi \in \Xi$. Set

$$D := \left\{ \xi \in \Xi \;\middle|\; (u, \xi^{-1}(v)) \in P \right\}.$$

Then we have

$$\begin{aligned}(\chi^P \cdot S_{P_2})(u, v) &= \begin{cases} 1 & \text{if } |D| \text{ is odd,} \\ 0 & \text{otherwise} \end{cases} \\ &= \chi^P(u, v).\end{aligned}$$

The last step holds because if $(u, v) \in P$, then $D = \Xi$ and $|D| = |\Xi|$ is odd (Lemma 7.33), and if $(u, v) \notin P$, then $D = \emptyset$ and $|D| = 0$. As seen before,

$$\chi^P \cdot S_{P_2} = \chi^P = \chi^Q = S_{Q_2} \cdot \chi^Q$$

because $Q = \lambda(P) = P$. □

Corollary 7.38. *The matrix S 1-blurs the twist between $(\mathfrak{A}_f, \bar{p})$ and $(\mathfrak{A}_g, \bar{p})$.*

We summarize the results of this section:

Lemma 7.39. *For all $q \geq 2, m \in \mathbb{N}$,*

- *every $(m+3)$-connected base graph $G = (V, E, \leq)$,*
- *all functions $f, g \colon E \to \mathbb{Z}_{2^q}$ which twist only a single base edge $\{\mathfrak{c}, \mathfrak{t}\}$ such that $f(\{\mathfrak{c}, \mathfrak{t}\}) = g(\{\mathfrak{c}, \mathfrak{t}\}) + 2^{q-1}$,*
- *and every m-tuple $\bar{p} \in A^m$ of $\mathfrak{A}_f := \mathsf{CFI}_{2^q}(G, f)$ and $\mathfrak{A}_g := \mathsf{CFI}_{2^q}(G, f)$,*

there is an odd-filled matrix S, both orbit-diagonal and orbit-invariant over $(\mathfrak{A}_f, \bar{p})$ and $(\mathfrak{A}_g, \bar{p})$, that 1-blurs the twist between $(\mathfrak{A}_f, \bar{p})$ and $(\mathfrak{A}_g, \bar{p})$ and satisfies $S_{P,Q} = \mathbb{1}$ for all k-orbits $P \in \mathsf{orb}_1((\mathfrak{A}_f, \bar{p}))$ and $Q \in \mathsf{orb}_1((\mathfrak{A}_g, \bar{p}))$ which are of the same type and have origin $\mathsf{orig}(P) = \mathsf{orig}(Q) \neq \mathfrak{c}$.

Constructing matrices blurring the twist for higher arities is more difficult: First, we have to generalize our notion of a blurrer to arity k. Second, we are faced with disconnected orbits, which do not pose a problem in the 1-ary case, but complicate matters in the general case. To deal with these orbits, we need to establish more technical lemmas for matrices over CFI structures.

7.6 The Active Region of a Matrix

In this section, we consider the part of a matrix S where S "has a nontrivial effect". Intuitively, this means that S is locally not the identity matrix. We will call the set of origins of all atoms, of which the tuples in these parts are composed, the active region. We now formalize this idea.

As in Section 7.4, let $q, k, m \in \mathbb{N}$ and $G = (V, E, \leq)$ be a $(k+m+1)$-connected base graph. Again, we denote, for every $f \colon E \to \mathbb{Z}_{2^q}$, the CFI structure $\mathsf{CFI}_{\mathbb{Z}_{2^q}}(G, f)$ by $\mathfrak{A}_f$ and the universe of these CFI structures by A. Let $\bar{p} \in A^m$ be arbitrary but fixed in this section. For a set $N \subseteq V$, we denote by $C_N(P)$ the **N-components** of an orbit P, that is, the set of components C of P that satisfy $C \subseteq N$ (recall Definition 7.16 of a component of an orbit).

Definition 7.40 (Active Region)**.** Let $f, g \colon E \to \mathbb{Z}_{2^q}$ not twist $\mathsf{orig}(\bar{p})$, S be an $A^k \times A^k$ matrix over $\mathbb{F}_2$, and $\mathbf{P}_k = \mathsf{orb}_k((\mathfrak{A}_f, \bar{p}))$ and $\mathbf{Q}_k = \mathsf{orb}_k((\mathfrak{A}_g, \bar{p}))$. For $P \in \mathbf{P}_k$ and $Q \in \mathbf{Q}_k$ of the same type, the matrix S is **active** (with respect to $(\mathfrak{A}_f, \bar{p})$ and $(\mathfrak{A}_g, \bar{p})$) on a component C of P (and so of Q) if there are $\bar{u} \in P$ and $\bar{v} \in Q$ such that $\bar{u}_C \neq \bar{v}_C$ and $S(\bar{u}, \bar{v}) = 1$. We write $\mathsf{A}^{f,g,\bar{p}}(S, P) = \mathsf{A}^{f,g,\bar{p}}(S, Q)$ for the set of components of P on which S is active, and $\mathsf{N}^{f,g,\bar{p}}(S, P) = \mathsf{N}^{f,g,\bar{p}}(S, Q)$ for the remaining components. The **active region $\mathsf{A}^{f,g,\bar{p}}(S) \subseteq V$ of S** is the inclusion-wise smallest set satisfying the following:

(A1) $C \subseteq \mathsf{A}^{f,g,\bar{p}}(S)$ for every $C \in \mathsf{A}^{f,g,\bar{p}}(S,P)$ and every $P \in \mathbf{P}_k$.

(A2) For all $P, P' \in \mathbf{P}_k$ and $Q, Q' \in \mathbf{Q}_k$ such that

$$C_{\mathsf{A}^{f,g,\bar{p}}(S)}(P) = C_{\mathsf{A}^{f,g,\bar{p}}(S)}(P') = C_{\mathsf{A}^{f,g,\bar{p}}(S)}(Q) = C_{\mathsf{A}^{f,g,\bar{p}}(S)}(Q') =: \mathsf{A},$$

both P and Q (respectively P' and Q') have the same type and thus $\mathsf{N}^{f,g,\bar{p}}(S,P) = \mathsf{N}^{f,g,\bar{p}}(S,Q) =: \mathsf{N}$ (respectively $\mathsf{N}^{f,g,\bar{p}}(S,P') = \mathsf{N}^{f,g,\bar{p}}(S,Q') =: \mathsf{N}'$), and for all $\bar{u} \in P$, $\bar{u}' \in P'$, $\bar{v} \in Q$, and $\bar{v}' \in Q'$, we have if $\bar{u}_\mathsf{A} = \bar{u}'_\mathsf{A}$, $\bar{v}_\mathsf{A} = \bar{v}'_\mathsf{A}$, $\bar{u}_\mathsf{N} = \bar{v}_\mathsf{N}$, and $\bar{u}'_{\mathsf{N}'} = \bar{v}'_{\mathsf{N}'}$, then $S(\bar{u}, \bar{v}) = S(\bar{u}', \bar{v}')$.

The active region is well-defined: Clearly, V itself satisfies Conditions A1 and A2. If two sets $X \subseteq V$ and $Y \subseteq V$ satisfy the two conditions, then also $X \cap Y$. Note that $C_{\mathsf{A}^{f,g,\bar{p}}(S)}(P)$ and $\mathsf{N}^{f,g,\bar{p}}(S,P)$ are not necessarily disjoint, but $\mathsf{N}^{f,g,\bar{p}}(S,P)$ contains all components of P not contained in $C_{\mathsf{A}^{f,g,\bar{p}}(S)}(P)$. Condition A2 can equivalently be stated only considering the components apart from the ones in $C_{\mathsf{A}^{f,g,\bar{p}}(S)}(P)$ instead of the set of components $\mathsf{N}^{f,g,\bar{p}}(S,P)$.

Although Condition A2 is rather technical, it ensures that the "non-identity-part" of S only depends on the active region: The entry $S(\bar{u}, \bar{v})$ only depends on the components of $\bar{u}$ and $\bar{v}$ on which S is active, as long as the entries for the other components are equal (otherwise $S(\bar{u}, \bar{v}) = 0$ anyway by Condition A1). That is $S(\bar{u}, \bar{v})$ only depends on whether $\bar{u}_{\mathsf{N}^{f,g,\bar{p}}(S,P)} = \bar{v}_{\mathsf{N}^{f,g,\bar{p}}(S,P)}$ but not on, for example, the type of $\bar{u}_{\mathsf{N}^{f,g,\bar{p}}(S,P)}$. We first consider the matrix blurring the twist defined in Section 7.5:

Lemma 7.41. *The matrix S given in the setting of Lemma 7.39 satisfies $\mathsf{A}^{f,g,\bar{p}}(S) = \{\mathfrak{c}\}$.*

Proof. Let $P \in \mathsf{orb}_1((\mathfrak{A}_f, \bar{p}))$ and $Q \in \mathsf{orb}_1((\mathfrak{A}_g, \bar{p}))$ be of the same type with origin $\mathfrak{u} = \mathsf{orig}(P) = \mathsf{orig}(Q) \neq \mathfrak{c}$. Then $S_{P \times Q} = \mathbb{1}$ by Lemma 7.39, i.e., S is clearly not active on $\{\mathfrak{u}\}$. The matrix S has to be active on $\{\mathfrak{c}\}$ because otherwise $S = \mathbb{1}$ and the structures would be isomorphic. This proves Condition A1. In the 1-ary case, a 1-orbit can only have one component, so Condition A2 of the active region is trivially satisfied. □

We now continue in the general case. The rest of this section establishes rather technical lemmas needed in Section 7.7. It is easy to see that if P and Q have the same type, whose origins contain no base vertex of $\mathsf{A}^{f,g,\bar{p}}(S)$, then $S_{P \times Q} = \mathbb{1}$. In the region of a twist, S has to be active:

Lemma 7.42. *Let $f, g \colon E \to \mathbb{Z}_{2^q}$ not twist $\mathsf{orig}(\bar{p})$, S be an $A^k \times A^k$ matrix over $\mathbb{F}_2$, and $P \in \mathsf{orb}_k((\mathfrak{A}_f, \bar{p}))$ and $Q \in \mathsf{orb}_k((\mathfrak{A}_g, \bar{p}))$ have the same type. If the block $S_{P \times Q}$ is nonzero and C is a component of P (and thus of Q) such that $P|_C \neq Q|_C$, then $C \in \mathsf{A}^{f,g,\bar{p}}(S,P)$.*

Proof. Let $\bar{u} \in P$ and $\bar{v} \in Q$ such that $S(\bar{u}, \bar{v}) = 1$. Such an entry must exist because $S_{P \times Q}$ is nonzero. If $\bar{u}_C = \bar{v}_C$, then $P|_C = Q|_C$ by Lemma 7.17 and Corollary 7.11, which contradicts our assumption. So $\bar{u}_C \neq \bar{v}_C$ and $C \in \mathsf{A}^{f,g,\bar{p}}(S,P)$. □

The next lemma shows that, as long as f and g agree on the base edges in $\mathsf{orig}(\bar{p})$, the values that f and g assign to base edges $\mathfrak{e}$ are actual irrelevant and only the difference $f(\mathfrak{e}) - g(\mathfrak{e})$ matters.

Lemma 7.43. *Let $f, g \colon E \to \mathbb{Z}_{2^q}$ not twist $\mathsf{orig}(\bar{p})$ and S be an $A^k \times A^k$ matrix over $\mathbb{F}_2$. Furthermore, let $f', g' \colon E \to \mathbb{Z}_{2^q}$ such that $f'(\mathfrak{e}) = f(\mathfrak{e})$ and $g'(\mathfrak{e}) = g(\mathfrak{e})$ for every $\mathfrak{e} \in E$ with $\mathfrak{e} \cap \mathsf{orig}(\bar{p}) \neq \emptyset$ and $f'(\mathfrak{e}) - f(\mathfrak{e}) = g'(\mathfrak{e}) - g(\mathfrak{e})$ for every other $\mathfrak{e} \in E$. Then $\mathsf{A}^{f,g,\bar{p}}(S) = \mathsf{A}^{f',g',\bar{p}}(S)$ and if S is orbit-diagonal (respectively orbit-invariant) over $(\mathfrak{A}_f, \bar{p})$ and $(\mathfrak{A}_g, \bar{p})$, then S is orbit-diagonal (respectively orbit-invariant) over $(\mathfrak{A}_{f'}, \bar{p})$ and $(\mathfrak{A}_{g'}, \bar{p})$.*

Proof. Note that if $P \in \mathsf{orb}_k((\mathfrak{A}_f, \bar{p}))$ has the same type in $(\mathfrak{A}_f, \bar{p})$ as $Q \in \mathsf{orb}_k((\mathfrak{A}_g, \bar{p}))$ has in $(\mathfrak{A}_g, \bar{p})$, then $P \in \mathsf{orb}_k((\mathfrak{A}_{f'}, \bar{p}))$ and $Q \in \mathsf{orb}_k((\mathfrak{A}_{g'}, \bar{p}))$ (Corollary 7.13) and P has the same type in $(\mathfrak{A}_{f'}, \bar{p})$ as Q has in $(\mathfrak{A}_{g'}, \bar{p})$. So we only change the type of the orbits, but not the correspondence between orbits of the same type. That is, if S is orbit-diagonal (respectively orbit-invariant) over $(\mathfrak{A}_f, \bar{p})$ and $(\mathfrak{A}_g, \bar{p})$, then S is orbit-diagonal (respectively orbit-invariant) over $(\mathfrak{A}_{f'}, \bar{p})$ and $(\mathfrak{A}_{g'}, \bar{p})$. Furthermore, S is active on the same components with respect to $(\mathfrak{A}_f, \bar{p})$ and $(\mathfrak{A}_g, \bar{p})$ as S is with respect to $(\mathfrak{A}_{f'}, \bar{p})$ and $(\mathfrak{A}_{g'}, \bar{p})$. This implies that $\mathsf{A}^{f,g,\bar{p}}(S) = \mathsf{A}^{f',g',\bar{p}}(S)$. □

We now show that the active region of products $S \cdot T$ is bounded by the active regions of S and T. For two k-tuples $\bar{u}, \bar{v} \in A^k$ we use the Kronecker delta $\delta_{\bar{u},\bar{v}}$, which is 1 if and only if $\bar{u} = \bar{v}$ and 0 otherwise.

Lemma 7.44. *Let $f, g, h \colon E \to \mathbb{Z}_{2^q}$ pairwise not twist $\mathsf{orig}(\bar{p})$ and S, T be $A^k \times A^k$ matrices over $\mathbb{F}_2$. If S is orbit-diagonal over $(\mathfrak{A}_f, \bar{p})$ and $(\mathfrak{A}_g, \bar{p})$ and T is orbit-diagonal over $(\mathfrak{A}_g, \bar{p})$ and $(\mathfrak{A}_h, \bar{p})$, then $\mathsf{A}^{f,h,\bar{p}}(S \cdot T) \subseteq \mathsf{A}^{f,g,\bar{p}}(S) \cup \mathsf{A}^{g,h,\bar{p}}(T)$.*

Proof. In this proof we omit the superscripts f, g, h, and $\bar{p}$ for readability: For S we always refer to f and g, for T to g and h, and for $S \cdot T$ to f and h. We show that $\mathsf{A}(S) \cup \mathsf{A}(T)$ satisfies Conditions A1 and A2. Because the active region is the inclusion-wise minimal set satisfying the two conditions, it then follows that $\mathsf{A}(S \cdot T) \subseteq \mathsf{A}(S) \cup \mathsf{A}(T)$. Let $\mathbf{P}_k = \mathsf{orb}_k((\mathfrak{A}_f, \bar{p}))$, $\mathbf{Q}_k = \mathsf{orb}_k((\mathfrak{A}_g, \bar{p}))$, and $\mathbf{R}_k = \mathsf{orb}_k((\mathfrak{A}_h, \bar{p}))$.

We show Condition A1 by contraposition. Let $P \in \mathbf{P}_k$ and C be a connected component of $G[\mathsf{orig}(P)]$. We show that

$$\mathsf{A}(S \cdot T, P) \subseteq \mathsf{A}(S, P) \cup \mathsf{A}(T, P).$$

Let $C \notin \mathsf{A}(S, P) \cup \mathsf{A}(T, P)$, $Q \in \mathbf{Q}_k$ and $R \in \mathbf{R}_k$ be of the same type as P, $\bar{u} \in P$, and $\bar{w} \in R$. Because S and T are orbit-diagonal,

$$(S \cdot T)(\bar{u}, \bar{w}) = \sum_{\bar{v} \in Q} S(\bar{u}, \bar{v}) \cdot T(\bar{v}, \bar{w}).$$

If $S(\bar{u}, \bar{v}) = 1$ (i.e., $S(\bar{u}, \bar{v}) \neq 0$), then $\bar{u}_C = \bar{v}_C$ because $C \notin \mathsf{A}(S, P)$. Similarly, $\bar{v}_C = \bar{w}_C$ if $T(\bar{v}, \bar{w}) = 1$. This implies $\bar{u}_C = \bar{w}_C$ if $(S \cdot T)(\bar{u}, \bar{w}) = 1$ (so there is at least one nonzero summand). Hence, $C \notin \mathsf{A}(S \cdot T, P)$.

To show Condition A2, let $P, P' \in \mathbf{P}_k$ and $R, R' \in \mathbf{R}_k$ be arbitrary k-orbits, such that

$$\mathsf{A} := C_{\mathsf{A}(S) \cup \mathsf{A}(T)}(P) = C_{\mathsf{A}(S) \cup \mathsf{A}(T)}(P') = C_{\mathsf{A}(S) \cup \mathsf{A}(T)}(R) = C_{\mathsf{A}(S) \cup \mathsf{A}(T)}(R'),$$

the orbits P and R have the same type, and P' and R' have the same type. Let N be the set of remaining components of P (and so of R) apart from A. Similarly, let N' be the set of remaining components of P' (and so of R') apart from A . Let $\bar{u} \in P$, $\bar{u}' \in P'$, $\bar{w} \in R$, and $\bar{w}' \in R'$ such that $\bar{u}_{\mathsf{A}} = \bar{u}'_{\mathsf{A}}$, $\bar{w}_{\mathsf{A}} = \bar{w}'_{\mathsf{A}}$, $\bar{u}_{\mathsf{N}} = \bar{w}_{\mathsf{N}}$, and $\bar{u}'_{\mathsf{N}'} = \bar{w}'_{\mathsf{N}'}$. We have to show that $(S \cdot T)(\bar{u}, \bar{w}) = (S \cdot T)(\bar{u}', \bar{w}')$.

By assumption, $\bar{u}_{\mathsf{A}} \in P|_{\mathsf{A}}$, $\bar{u}'_{\mathsf{A}} \in P'|_{\mathsf{A}}$, and $\bar{u}_{\mathsf{A}} = \bar{u}'_{\mathsf{A}}$. So $P|_{\mathsf{A}} = P'|_{\mathsf{A}}$ by Corollary 7.11 because they have the same type and contain the same tuple. Let $Q \in \mathbf{Q}_k$ be of the same type as P and $Q' \in \mathbf{Q}_k$ be of the same type as P'. Then $Q|_{\mathsf{A}} = Q'|_{\mathsf{A}}$ and A and N (respectively N') are sets of components of Q (respectively Q'). We first assume that the blocks $S_{P\times Q}$ and $T_{Q\times R}$ are nonzero. We apply Lemma 7.17: $Q = Q|_{\mathsf{A}} \times Q|_{\mathsf{N}}$, $Q' = Q'|_{\mathsf{A}} \times Q'|_{\mathsf{N}'}$, and likewise for P and P'.

$$\begin{aligned}(S \cdot T)(\bar{u}, \bar{w}) &= \sum_{\bar{v} \in Q} S(\bar{u}, \bar{v}) \cdot T(\bar{v}, \bar{w}) \\ &= \sum_{\bar{v}_{\mathsf{A}} \in Q|_{\mathsf{A}}} \sum_{\bar{v}_{\mathsf{N}} \in Q|_{\mathsf{N}}} S(\bar{u}_{\mathsf{A}}\bar{u}_{\mathsf{N}}, \bar{v}_{\mathsf{A}}\bar{v}_{\mathsf{N}}) \cdot T(\bar{v}_{\mathsf{A}}\bar{v}_{\mathsf{N}}, \bar{w}_{\mathsf{A}}\bar{w}_{\mathsf{N}}). \qquad (\star)\end{aligned}$$

From Lemma 7.42 it follows that $P|_{\mathsf{N}} = Q|_{\mathsf{N}} = R|_{\mathsf{N}}$ (recall that the blocks $S_{P\times Q}$ and $T_{Q\times R}$ were assumed to be nonzero), in particular, $\bar{u}_{\mathsf{N}}, \bar{w}_{\mathsf{N}} \in Q|_{\mathsf{N}}$. We use again that the N-components are not in the active region of S and T. We continue the equation $(\star)$:

$$\begin{aligned}(\star) &= \sum_{\bar{v}_{\mathsf{A}} \in Q|_{\mathsf{A}}} \sum_{\bar{v}_{N} \in Q|_{\mathsf{N}}} \delta_{\bar{u}_{\mathsf{N}},\bar{v}_{\mathsf{N}}} \cdot S(\bar{u}_{\mathsf{A}}\bar{u}_{\mathsf{N}}, \bar{v}_{\mathsf{A}}\bar{u}_{\mathsf{N}}) \cdot \delta_{\bar{v}_{\mathsf{N}},\bar{w}_{\mathsf{N}}} \cdot T(\bar{v}_{\mathsf{A}}\bar{w}_{\mathsf{N}}, \bar{w}_{\mathsf{A}}\bar{w}_{\mathsf{N}}) \\ &= \sum_{\bar{v}_{\mathsf{A}} \in Q|_{\mathsf{A}}} \delta_{\bar{u}_{\mathsf{N}},\bar{w}_{\mathsf{N}}} \cdot S(\bar{u}_{\mathsf{A}}\bar{u}_{\mathsf{N}}, \bar{v}_{\mathsf{A}}\bar{u}_{\mathsf{N}}) \cdot T(\bar{v}_{\mathsf{A}}\bar{w}_{\mathsf{N}}, \bar{w}_{\mathsf{A}}\bar{w}_{\mathsf{N}}) \\ &= \sum_{\bar{v}_{\mathsf{A}} \in Q'|_{\mathsf{A}}} \delta_{\bar{u}'_{\mathsf{N}'},\bar{w}'_{\mathsf{N}'}} \cdot S(\bar{u}_{\mathsf{A}}\bar{u}'_{\mathsf{N}'}, \bar{v}_{\mathsf{A}}\bar{u}'_{\mathsf{N}'}) \cdot T(\bar{v}_{\mathsf{A}}\bar{w}'_{\mathsf{N}'}, \bar{w}_{\mathsf{A}}\bar{w}'_{\mathsf{N}'}) \\ &= \sum_{\bar{v}'_{\mathsf{A}} \in Q'|_{\mathsf{A}}} \sum_{\bar{v}'_{\mathsf{N}'} \in Q'|_{\mathsf{N}'}} \delta_{\bar{u}'_{\mathsf{N}'},\bar{v}'_{\mathsf{N}'}} \cdot S(\bar{u}_{\mathsf{A}}\bar{u}'_{\mathsf{N}'}, \bar{v}'_{\mathsf{A}}\bar{u}'_{\mathsf{N}'}) \cdot \delta_{\bar{v}'_{\mathsf{N}'},\bar{w}'_{\mathsf{N}'}} \cdot T(\bar{v}'_{\mathsf{A}}\bar{w}'_{\mathsf{N}'}, \bar{w}_{\mathsf{A}}\bar{w}'_{\mathsf{N}'}).\end{aligned}$$

Here, we used the identity $Q|_{\mathsf{A}} = Q'|_{\mathsf{A}}$ and the inclusion $\mathsf{A}(S \cdot T, P) \subseteq \mathsf{A}(S, P) \cup \mathsf{A}(T, P)$, where the latter was already shown for Condition A1. So $\bar{u}_{\mathsf{N}}$ can be exchanged with $\bar{u}'_{\mathsf{N}'}$ and $\bar{w}_{\mathsf{N}}$ with $\bar{w}'_{\mathsf{N}'}$, respectively. In the next step we use that $\bar{u}_{\mathsf{A}} = \bar{u}'_{\mathsf{A}}$ and $\bar{w}_{\mathsf{A}} = \bar{w}'_{\mathsf{A}}$ (by assumption) and again that S and T are not active on the N'-components.

$$\begin{aligned}(\star) &= \sum_{\bar{v}'_{\mathsf{A}} \in Q'|_{\mathsf{A}}} \sum_{\bar{v}'_{\mathsf{N}} \in Q'|_{\mathsf{N}'}} S(\bar{u}'_{\mathsf{A}}\bar{u}'_{\mathsf{N}'}, \bar{v}'_{\mathsf{A}}\bar{v}'_{\mathsf{N}'}) \cdot T(\bar{v}'_{\mathsf{A}}\bar{v}'_{\mathsf{N}'}, \bar{w}'_{\mathsf{A}}\bar{w}'_{\mathsf{N}'}) \\ &= \sum_{\bar{v}' \in Q'} S(\bar{u}', \bar{v}') \cdot T(\bar{v}', \bar{w}') \\ &= (S \cdot T)(\bar{u}', \bar{w}').\end{aligned}$$

If $S_{P\times Q}$ or $T_{Q\times R}$ is zero, then $S_{P'\times Q'}$ or $T_{Q'\times R'}$ is zero because $S(\bar{u}, \bar{v}) = S(\bar{u}', \bar{v}') = 0$ and likewise for T. The claim follows because $(S \cdot T)_{P\times Q} = \mathbb{0}$ and $(S \cdot T)_{P'\times Q'} = \mathbb{0}$. $\square$

We now consider products $S \cdot T$ in the case that the active regions of S and T are disjoint. Intuitively, our goal is to prove that then $S \cdot T$ is given by S on the active region of S and by T on the active region of T.

Lemma 7.45. *Let $f, g, h \colon E \to \mathbb{Z}_{2^q}$ pairwise not twist* $\mathsf{orig}(\bar{p})$ *and S, T be $A^k \times A^k$ matrices over $\mathbb{F}_2$. Let S be orbit-diagonal over $(\mathfrak{A}_f, \bar{p})$ and $(\mathfrak{A}_g, \bar{p})$, T be orbit-diagonal* *over $(\mathfrak{A}_g, \bar{p})$ and $(\mathfrak{A}_h, \bar{p})$, both be odd-filled, $\mathsf{A}^{f,g,\bar{p}}(S) \cap \mathsf{A}^{g,h,\bar{p}}(T) = \emptyset$, $P \in \mathsf{orb}_k((\mathfrak{A}_f, \bar{p}))$, $Q \in \mathsf{orb}_k((\mathfrak{A}_g, \bar{p}))$, and $R \in \mathsf{orb}_k((\mathfrak{A}_h, \bar{p}))$ be of the same type, and the components of P (and thus the components of Q and R) be partitioned into set M and N such that $C_{\mathsf{A}^{f,g,\bar{p}}(S)}(P) \subseteq M$ and $C_{\mathsf{A}^{g,h,\bar{p}}(T)}(Q) \subseteq N$.*

(a) *For every $\bar{u} \in P$ and $\bar{w} \in R$, it holds that*

$$(S \cdot T)(\bar{u}, \bar{w}) = S(\bar{u}_M \bar{u}_N, \bar{w}_M \bar{u}_N) \cdot T(\bar{w}_M \bar{u}_N, \bar{w}_M \bar{w}_N).$$

(b) *If S is orbit-invariant over $(\mathfrak{A}_f, \bar{p})$ and $(\mathfrak{A}_g, \bar{p})$, then for every $\bar{u} \in P$ and $\bar{w} \in R$, it holds that*

$$\sum_{\bar{u}'_M \in P|_M} (S \cdot T)(\bar{u}'_M \bar{u}_N, \bar{w}_M \bar{w}_N) = T(\bar{w}_M \bar{u}_N, \bar{w}_M \bar{w}_N).$$

Proof. We first show Part (a). Let $\bar{u} \in P$ and $\bar{w} \in R$.

$$\begin{aligned}(S \cdot T)(\bar{u}, \bar{w}) &= \sum_{\bar{v} \in Q} S(\bar{u}_M \bar{u}_N, \bar{v}_M \bar{v}_N) \cdot T(\bar{v}_M \bar{v}_N, \bar{w}_M \bar{w}_N) \\ &= \sum_{\bar{v} \in Q} \delta_{\bar{u}_N, \bar{v}_N} \cdot S(\bar{u}_M \bar{u}_N, \bar{v}_M \bar{u}_N) \cdot \delta_{\bar{v}_M, \bar{w}_M} \cdot T(\bar{w}_M \bar{v}_N, \bar{w}_M \bar{w}_N). \qquad (\star)\end{aligned}$$

The last step uses that components of $\bar{u}_N$ and $\bar{w}_N$ consist only of vertices not contained in $\mathsf{A}(S)$ and likewise for $\bar{v}_M$ and $\bar{w}_M$.

$$\begin{aligned}(\star) &= \sum_{\substack{\bar{v} \in Q, \\ \bar{v} = \bar{w}_M \bar{u}_N}} S(\bar{u}_M \bar{u}_N, \bar{w}_M \bar{u}_N) \cdot T(\bar{w}_M \bar{u}_N, \bar{w}_M \bar{w}_N) \\ &= S(\bar{u}_M \bar{u}_N, \bar{w}_M \bar{u}_N) \cdot T(\bar{w}_M \bar{u}_N, \bar{w}_M \bar{w}_N).\end{aligned}$$

For the last step, we have to argue that $\bar{w}_M \bar{u}_N \in Q$. From Lemma 7.17 it follows that $P = P|_M \times P|_N$, $Q = Q|_M \times Q|_N$, and $R = R|_M \times R|_N$. Because S is not active on the components in N and T is not active on the components in M, it follows from Lemma 7.42 that $P|_N = Q|_N$ and that $Q|_M = R|_M$ (the corresponding blocks of S and T are nonzero because S and T are odd-filled). Hence, $\bar{w}_M \bar{u}_N \in Q$ because $\bar{w}_M \in R|_M$ and $\bar{u}_N \in P|_N$.

We now show Part (b). We apply Part (a):

$$\begin{aligned}&\sum_{\bar{u}'_M \in P|_M} (S \cdot T)(\bar{u}'_M \bar{u}_N, \bar{w}_M \bar{w}_N) \\ &= \sum_{\bar{u}'_M \in P|_M} S(\bar{u}'_M \bar{u}_N, \bar{w}_M \bar{u}_N) \cdot T(\bar{w}_M \bar{u}_N, \bar{w}_M \bar{w}_N) \\ &= T(\bar{w}_M \bar{u}_N, \bar{w}_M \bar{w}_N) \cdot \sum_{\bar{u}'_M \in P|_M} S(\bar{u}'_M \bar{u}_N, \bar{w}_M \bar{u}_N).\end{aligned}$$

It suffices to show that the value of the sum is 1. We rewrite the sum using $P = P|_M \times P|_N$ (Lemma 7.17):

$$\sum_{\bar{u}'_M \in P|_M} S(\bar{u}'_M \bar{u}_N, \bar{w}_M \bar{u}_N) = \sum_{\bar{u}'_M \bar{u}'_N \in P} S(\bar{u}'_M \bar{u}'_N, \bar{w}_M \bar{u}_N) - \sum_{\substack{\bar{u}'_M \bar{u}'_N \in P, \\ \bar{u}'_N \neq \bar{u}_N}} S(\bar{u}'_M \bar{u}'_N, \bar{w}_M \bar{u}_N).$$

In the right sum it always holds that $S(\bar{u}'_M \bar{u}'_N, \bar{w}_M \bar{u}_N) = 0$ because $\bar{u}'_N \neq \bar{u}_N$ and N is not in the active region of S. So the right sum is zero. Finally, the left summation $\sum_{\bar{u}'_M \bar{u}'_N \in P} S(\bar{u}'_M \bar{u}'_N, \bar{w}_M \bar{u}_N)$ runs over a column of S because S is orbit-diagonal. Since S is orbit-invariant and odd-filled, this summation is 1 by Lemma 7.29. □

Finally, we show the result of Lemma 7.45(b) for a product of three matrices $S_1 \cdot S_2 \cdot S_3$.

Lemma 7.46. *Let $g_i \colon E \to \mathbb{Z}_{2^q}$ pairwise not twist $\mathsf{orig}(\bar{p})$ for every $i \in [4]$. Let S_i be an $A^k \times A^k$ matrix over $\mathbb{F}_2$ that is odd-filled and both orbit-diagonal and orbit-invariant over $(\mathfrak{A}_{g_i}, \bar{p})$ and $(\mathfrak{A}_{g_{i+1}}, \bar{p})$ for every $i \in [3]$. If the active regions $\mathsf{A}^{g_i, g_{i+1}, \bar{p}}(S_i)$ are pairwise disjoint, then for all k-orbits $P_i \in \mathsf{orb}_k((\mathfrak{A}_{g_i}, \bar{p}))$ of the same type for all $i \in [4]$, every partition of the components of the P_i into M_1, M_2, and M_3 such that $C_{\mathsf{A}^{g_i, g_{i+1}, \bar{p}}(S_i)}(P_i) \subseteq M_i$ for every $i \in [3]$, and every $\bar{u} \in P_1$ and $\bar{w} \in P_4$ it holds that*

$$\sum_{\bar{u}'_{M_2} \in P_1|_{M_2}} (S_1 \cdot S_2 \cdot S_3)(\bar{u}_{M_1}\bar{u}'_{M_2}\bar{u}_{M_3}, \bar{w}_{M_1}\bar{w}_{M_2}\bar{w}_{M_3}) = (S_1 \cdot S_3)(\bar{u}_{M_1}\bar{w}_{M_2}\bar{u}_{M_3}, \bar{w}_{M_1}\bar{w}_{M_2}\bar{w}_{M_3}).$$

Proof. By Lemma 7.26, the matrix $(S_1 \cdot S_2)$ is orbit-diagonal and orbit-invariant over $(\mathfrak{A}_{g_1}, \bar{p})$ and $(\mathfrak{A}_{g_3}, \bar{p})$ and the matrix $(S_2 \cdot S_3)$ is orbit-diagonal and orbit-invariant over $(\mathfrak{A}_{g_2}, \bar{p})$ and $(\mathfrak{A}_{g_4}, \bar{p})$. Both matrices are odd-filled by Lemma 7.28. We now apply Lemma 7.45(a) for the partition of the components of P into $M_1 \cup M_3$ and M_2:

$$\begin{aligned}
&\sum_{\bar{u}'_{M_2} \in P_1|_{M_2}} (S_1 \cdot S_2 \cdot S_3)(\bar{u}_{M_1}\bar{u}'_{M_2}\bar{u}_{M_3}, \bar{w}_{M_1}\bar{w}_{M_2}\bar{w}_{M_3}) \\
= &\sum_{\bar{u}'_{M_2} \in P_1|_{M_2}} S_1(\bar{u}_{M_1}\bar{u}'_{M_2}\bar{u}_{M_3}, \bar{w}_{M_1}\bar{u}'_{M_2}\bar{u}_{M_3}) \cdot (S_2 \cdot S_3)(\bar{w}_{M_1}\bar{u}'_{M_2}\bar{u}_{M_3}, \bar{w}_{M_1}\bar{w}_{M_2}\bar{w}_{M_3}) \\
= &\sum_{\bar{u}'_{M_2} \in P_1|_{M_2}} S_1(\bar{u}_{M_1}\bar{w}_{M_2}\bar{u}_{M_3}, \bar{w}_{M_1}\bar{w}_{M_2}\bar{u}_{M_3}) \cdot (S_2 \cdot S_3)(\bar{w}_{M_1}\bar{u}'_{M_2}\bar{u}_{M_3}, \bar{w}_{M_1}\bar{w}_{M_2}\bar{w}_{M_3}). \qquad (\star)
\end{aligned}$$

The last step uses that M_2 consists only of components not contained in $\mathsf{A}^{g_1, g_2, \bar{p}}(S_1)$. We continue the equation by moving S_1 out of the sum and applying Lemma 7.45(b) for the partition of the components of P into $M_1 \cup M_3$ and M_2:

$$\begin{aligned}
(\star) &= S_1(\bar{u}_{M_1}\bar{w}_{M_2}\bar{u}_{M_3}, \bar{w}_{M_1}\bar{w}_{M_2}\bar{u}_{M_3}) \cdot \sum_{\bar{u}'_{M_2} \in P|_{M_2}} (S_2 \cdot S_3)(\bar{w}_{M_1}\bar{u}'_{M_2}\bar{u}_{M_3}, \bar{w}_{M_1}\bar{w}_{M_2}\bar{w}_{M_3}) \\
&= S_1(\bar{u}_{M_1}\bar{w}_{M_2}\bar{u}_{M_3}, \bar{w}_{M_1}\bar{w}_{M_2}\bar{u}_{M_3}) \cdot S_3(\bar{w}_{M_1}\bar{w}_{M_2}\bar{u}_{M_3}, \bar{w}_{M_1}\bar{w}_{M_2}\bar{w}_{M_3}) \\
&= (S_1 \cdot S_3)(\bar{u}_{M_1}\bar{w}_{M_2}\bar{u}_{M_3}, \bar{w}_{M_1}\bar{w}_{M_2}\bar{w}_{M_3}).
\end{aligned}$$

The last step follows from applying Lemma 7.43 and, using the partition of the components into M_1 and $M_2 \cup M_3$, from Lemma 7.45(a). □

7.7 The Arity k Case

We now construct a similarity matrix for the k-ary invertible-map game. Constructing this matrix and verifying its suitability will be quite technical and intricate. We first discuss the difficulties we have to overcome and why the approach for arity 1 cannot simply be generalized to arity k easily. In the following, we provide high-level intuition for constructing the similarity matrix for arity k. This prepares us for the lengthy formal definition of this matrix, which follows subsequently.

7.7.1 Overview of the Construction

Orbits of the Same Type. Let $\mathfrak{A}_f$ and $\mathfrak{A}_g$ be two CFI structures such that a single base edge $\{\mathsf{t}, \mathsf{t}'\}$ of the base graph G is twisted by f and g. Let $\bar{p}$ be a tuple of parameters, whose origin has sufficiently large distance to the twisted base edge. We have seen in Section 7.5 that every 1-orbit has the same type in $(\mathfrak{A}_f, \bar{p})$ as it has in $(\mathfrak{A}_g, \bar{p})$. For a k-orbit P, this is not the case whenever $\{\mathsf{t}, \mathsf{t}'\} \subseteq \mathsf{orig}(P)$. Ultimately, our goal is to construct an orbit-invariant, orbit-diagonal, and odd-filled similarity matrix S that k-blurs the twist. Because the blocks on the diagonal of S arise from orbits of the same type and because the characteristic matrices of orbits of the same type have to be simultaneously similar, we first want to define a bijection $\mathsf{orb}_{k'}((\mathfrak{A}_f, \bar{p})) \to \mathsf{orb}_{k'}((\mathfrak{A}_g, \bar{p}))$ for every $k' \leq 2k$ that preserves the orbit types. For this, we want to construct a function $\tau \colon A^{\leq 2k} \to A^{\leq 2k}$ that preserves the type of tuples. Then τ preserves orbit types, too. To do so, we pick a base vertex $\mathfrak{c}$ satisfying $\mathsf{dist}_G(\mathsf{t}, \mathfrak{c}) > 2k$ and a path $(\mathfrak{c}, \dots, \mathsf{t}', \mathsf{t})$. We consider the path-isomorphism φ_τ that twists the base edge $\{\mathsf{t}, \mathsf{t}'\}$ and the base edge incident to $\mathfrak{c}$ in the chosen path. That is, between $\varphi_\tau(\mathfrak{A}_f)$ and $\mathfrak{A}_g$ a base edge incident to $\mathfrak{c}$ is twisted but the base edge $\{\mathsf{t}, \mathsf{t}'\}$ is not. For the moment assume that we only consider connected tuples and thus only connected orbits. Let τ be the function that applies the path-isomorphism φ_τ to every tuple $\bar{u}$ with $\{\mathsf{t}, \mathsf{t}'\} \subseteq \mathsf{orig}(\bar{u})$ and is the identity function on all others. Let $\bar{u} \subseteq A^{\leq 2k}$ be such a tuple with $\{\mathsf{t}, \mathsf{t}'\} \subseteq \mathsf{orig}(\bar{u})$. Because $\mathsf{dist}_G(\mathsf{t}, \mathfrak{c}) > 2k$ and because we consider connected tuples, we have that $\mathfrak{c} \notin \mathsf{orig}(\bar{u})$. Hence, we have $\mathfrak{A}_g[\mathsf{orig}(\bar{u})] = \varphi_\tau(\mathfrak{A}_f)[\mathsf{orig}(\bar{u})]$ and $\bar{u}$ has the same type in $(\mathfrak{A}_f, \bar{p})$ as $\tau(\bar{u})$ has in $(\mathfrak{A}_g, \bar{p})$. Consequently, for all $k' \leq 2k$ and $P \in \mathsf{orb}_{k'}((\mathfrak{A}_f, \bar{p}))$ it holds that $\tau(P) \in \mathsf{orb}_{k'}((\mathfrak{A}_g, \bar{p}))$ and $\tau(P)$ has the same type in $(\mathfrak{A}_g, \bar{p})$ as P has in $(\mathfrak{A}_f, \bar{p})$.

Generalized Blurrers. Next we transfer the concept of a blurrer to the k-ary case. Definition 7.32 of a (q, d)-blurrer requires that there seems to be a twist at index 1 but none at the other indices when considering only one of the d entries of the blurrer elements. Although, all tuples ξ in a blurrer satisfy $\sum \xi = 0$. We require the same property in the k-ary case, but now not only consider a single index at a time but sets of k many indices. We will generalize (q, d)-blurrers to (k, q, a, d)-blurrers, where k is the arity, q specifies the ring $\mathbb{Z}_{2^q}$, the length of the tuples in the blurrer is d, and $a \in \mathbb{Z}_{2^q}$ is the value of the twist (which was fixed to 2^{q-1} before). Showing the existence of such blurrers will be more difficult, in particular, we will have to consider, for a given k, the ring $\mathbb{Z}_{2^q}$ for a sufficiently large $q = q(k)$.

In the 1-ary case, we identified a tuple $\xi \in \Xi$ with a local automorphism of the gadget of a base vertex $\mathfrak{c}$. We now describe the approach in the k-ary case. Assume we are given a generalized (k, q, a, d)-blurrer Ξ for arity k for some suitable q, $a \in \mathbb{Z}_{2^q}$, and d. We now require that the base graph G is regular of degree d. Recall that in Section 7.5 we blurred the twist between the base edges incident to $\mathfrak{c}$, of which one was the twisted base edge: We used multiple local automorphisms (one for each $\xi \in \Xi$) to distribute the twist among these base edges. When considering connected $2k$-tuples, we want to ensure that the origin of every $2k$-tuple contains at most one of the base edges between which we blur the twist. So it is not possible to blur the twist between the incident base edges of a single base vertex. Instead, we will choose base vertices $\mathsf{t}_1, \dots, \mathsf{t}_d$ and $\mathsf{t}'_1, \dots, \mathsf{t}'_d$, such that $\mathsf{t} = \mathsf{t}_1$, $\mathsf{t}' = \mathsf{t}'_1$, and such that, for every $i \in [d]$, there is a simple path $\bar{\mathfrak{s}}_i$ of length at most $2k$ starting at $\mathfrak{c}$ and ending with $(\mathfrak{t}_i', \mathfrak{t}_i)$. These paths are chosen to form a star, i.e., the paths $\bar{\mathfrak{s}}_i$ are disjoint apart from $\mathfrak{c}$ (see Figure 7.3 and compare to Figure 7.1). Here it will be important to choose $\bar{\mathfrak{s}}_1$ to be the path we used to define the tuple-type-preserving map τ in the previous paragraph. We will ensure that such paths exist by requiring that the girth of G is large enough. We will blur the twist between the base edges $\{\mathfrak{t}_i, \mathfrak{t}_i'\}$. In the 1-ary case, an element in a blurrer corresponded to an automorphism of the gadget of $\mathfrak{c}$, or equivalently to a star-isomorphism, where the paths of the star have length 1. In the k-ary case we will identify a tuple $\xi \in \Xi$ with the star-isomorphism $\varphi_\xi := \pi^*[\xi, \bar{\mathfrak{s}}_1, \dots, \bar{\mathfrak{s}}_d]$. Again to preserve the type of tuples, we will only apply ξ to tuples $\bar{u}$ satisfying $\{\mathfrak{t}_i, \mathfrak{t}_i'\} \not\subseteq \mathsf{orig}(\bar{u})$ for all $i \in [d]$. That is, on such a $\bar{u}$, the action of ξ could also be defined by an automorphism. This turns ξ into a "star-automorphism". Using a star in combination with the large girth ensures that the tips of the star, the base edges $\{\mathfrak{t}_i, \mathfrak{t}_i'\}$, are sufficiently far apart. If we only had to deal with connected tuples, this approach would be sufficient to construct a similarity matrix (and in particular, we could even use easier blurrers). However, disconnected tuples complicate matters.

Disconnected Tuples and Orbits. We have to consider disconnected tuples and orbits. While for connected tuples the approach just described is local (we only considered the $2k$-neighborhood of $\mathfrak{c}$), there are disconnected tuples containing atoms scattered in the structure. However, these atoms belong to different components of the tuple (cf. Definition 7.16). Lemma 7.17 tells us that the components of disconnected orbits are independent whenever the connectivity of G is sufficiently large. In a first step, we will salvage the previous approach by applying the path- and the star-isomorphism to components of tuples and not to entire tuples. That is, if a component C contains the twisted base edge $\{\mathfrak{t}, \mathfrak{t}'\} = \{\mathfrak{t}_1, \mathfrak{t}_1'\}$, then we apply the type-preserving map τ to this component. If $\{\mathfrak{t}_i, \mathfrak{t}_i'\} \not\subseteq C$ for all $i \in [d]$, i.e, C contains none of the base edges between which we blur the twist, we apply the star-automorphisms ξ to this component. (Note that ξ is the identity map unless C intersects nontrivially with at least one path $\bar{\mathfrak{s}}_i$.)

This approach fails when for a $2k$-orbit P the two k-orbits $P_1 := P|_{\{1,\dots,k\}}$ and $P_2 := P|_{\{k+1,\dots,2k\}}$ contain the center $\mathfrak{c}$ of the star and some of the base edges $\{\mathfrak{t}_i, \mathfrak{t}_i'\}$ in their origin. Because the base edges $\{\mathfrak{t}_i, \mathfrak{t}_i'\}$ are contained in the origin, we need the blurrer properties to show that we blur the twist. This is only possible if for two k-tuples $\bar{u} \in P_1$ and $\bar{v} \in P_2$ it only depends on up to k indices of a tuple $\xi \in \Xi$ whether $\xi(\bar{u})\bar{v}$ is in the same orbit as $\bar{u}\bar{v}$. But because the center $\mathfrak{c}$ is in the origin, this actually depends on all d entries of ξ and the blurrer properties do not apply. This is why we will have to distinguish two kinds of k-orbits. We call a k-orbit P **blurrable** if $\mathfrak{c} \notin \mathsf{orig}(P)$. For non-blurrable orbits, we need another technique that we describe in the following.

Recursive Blurring. Now consider a $2k$-orbit P, such that both $P_1 := P|_{\{1,\dots,k\}}$ and $P_2 := P|_{\{k+1,\dots,2k\}}$ are non-blurrable k-orbits. Let us quickly recall the 1-ary case and assume for the moment that P is a 2-orbit. It was possible to blur the twist in Lemma 7.37 because we summed over the tuples $\xi(\bar{u})\bar{v}$ for all $\xi \in \Xi$. Assume that the origin of P is the twisted edge $\{\mathfrak{c}, \mathfrak{t}\}$ and that the origin of P_2 is $\mathfrak{c}$. Then, for every $v \in P$, it held that $\xi(v) \neq \xi'(v)$ for every $\xi \neq \xi'$ in the blurrer. But for P_1 only one index of the blurrer was relevant. That is, for all $u \in P_1$, $v \in P_2$, and $\xi, \xi' \in \Xi$ such that $\xi(1) = \xi'(1)$, we had that $\xi(u) = \xi'(u)$ and that $\xi(u)v$ is in the same orbit as $\xi'(u)v$. So we were able to apply the properties of a blurrer, i.e., when summing over $\xi(u)v$ for all $\xi \in \Xi$ and if only one index matters, then the twist vanishes. The 2-orbits for which both P_1 and P_2 have origin $\{\mathfrak{c}\}$ did not cover the twisted base edge and so did not pose a problem in the 1-ary case.

Now consider the k-ary case again. Here of course there are orbits P such that both P_1 and P_2 are non-blurrable and they contain the twisted base edge and the center $\mathfrak{c}$ in their origins. Let $\bar{u} \in P_1$ and $\bar{v} \in P_2$. Both $\bar{u}$ and $\bar{v}$ contain an atom with origin $\mathfrak{c}$ and the blurrer properties do not apply because the orbit of $\xi(\bar{u})\bar{v}$ is different for every $\xi \in \Xi$ (fixing one atom of origin $\mathfrak{c}$ separates the gadget of $\mathfrak{c}$ into singleton orbits). That is, when summing over all $\xi \in \Xi$, we map every $\bar{u}\bar{v}$ to the tuple $\xi(\bar{u})\bar{v}$, whose type in $(\varphi_\xi(\mathfrak{A}_g), \bar{p})$ is the same as the type of $\bar{u}\bar{v}$ in $(\mathfrak{A}_f, \bar{p})$. But in $(\mathfrak{A}_g, \bar{p})$ the tuple $\xi(\bar{u})\bar{v}$ has a different type. Between $(\mathfrak{A}_g, \bar{p})$ and $(\varphi_\xi(\mathfrak{A}_g), \bar{p})$ the base edges $\{\mathfrak{t}_i, \mathfrak{t}_i'\}$ are additionally twisted and the values of the twists depend on ξ. This, in some sense, introduces other twists, but only for said $2k$-orbits P, where both P_1 and P_2 are non-blurrable.

The idea to solve is problem is to fix an arbitrary atom $p_\mathfrak{c}$ with origin $\mathfrak{c}$ and consider $(k-1)$-orbits of $(\mathfrak{A}, \bar{p}p_\mathfrak{c})$, which is justified by Corollary 7.19. This can be done because all non-blurrable orbits contain $\mathfrak{c}$ in its origin. For every $\xi \in \Xi$, we will recursively obtain a matrix S^ξ that $(k-1)$-blurs the twist between $(\mathfrak{A}_f, \bar{p}p_\mathfrak{c})$ and $(\varphi_\xi^{-1}(\mathfrak{A}_g), \bar{p}p_\mathfrak{c})$. We use the inverse φ_ξ^{-1} of the star-isomorphism φ_ξ because we want to revert the twists introduced by φ_ξ. Here the need arises to blur a twist of value $a \neq 2^{q-1}$. Combining the blurrer Ξ with the matrices S^ξ to a matrix S that k-blurs the twist will become formally tedious. In particular, we will need to ensure that the S^ξ act "independently" on the $\{\mathfrak{t}_i, \mathfrak{t}_i'\}$, which we discuss next.

Active Region and Blurrers. The matrix S is defined for blocks of k-orbits. Blocks for blurrable k-orbits will be defined using the blurrer Ξ, blocks for non-blurrable k-orbits will be defined using Ξ and the matrices S^ξ. With this approach we will show that S is a similarity matrix for all orbits P, for which either P_1 and P_2 are both blurrable or P_1 and P_2 are both non-blurrable. In the former case, we will use the blurrer property, in the latter case, we will use induction. The case that P_1 is blurrable and P_2 is not or vice versa remains. We have to show that $\chi^P \cdot S = S \cdot \chi^Q$ (for $Q = \tau(P)$, which has the same type as P). Assume that P_1 is blurrable and P_2 is not. For $S \cdot \chi^Q$ solely the block $S_{P_1 \times Q_1}$ of S is relevant. This block is defined using the blurrer Ξ because P_1 is blurrable. Similarly, for $\chi^P \cdot S$ solely the block $S_{P_2 \times Q_2}$ is relevant. This block is defined using the blurrer Ξ and the matrices S^ξ because P_2 is non-blurrable.

To use the blurrer properties also for P_2, we will define matrices S^ξ, which blur multiple twists at the base edges $\{\mathfrak{t}_i, \mathfrak{t}_i'\}$, as $S^\xi := S^{\xi,1} \cdot \dots \cdot S^{\xi,d}$, where each $S^{\xi,i}$ only blurs a single twist at the base edge $\{\mathfrak{t}_i, \mathfrak{t}_i'\}$. We will ensure that the active region of $S^{\xi,i}$ is bounded by the $r(k)$-neighborhood of $\mathfrak{t}_i$ for some suitable $r(k)$. We then enlarge the star such that the paths $\bar{\mathfrak{s}}_i$ have length greater than $\max\{2k, r(k)\}$. Now, the active regions of the $S^{\xi,i}$ are disjoint and we can use Lemma 7.46 to show that indeed all except k many of the $S^{\xi,i}$ cancel. So finally, we can use the blurrer properties to show that S is a similarity matrix for orbits P for which P_1 is blurrable and P_2 is not. We now start with generalizing blurrers and then show the existence of the required similarity matrix.

7.7.2 Blurrers

When dealing with arity k, the properties of a blurrer must be generalized from a single index to sets of indices of size at most k. Let $q, d \in \mathbb{N}$ and $\Xi \subseteq \mathbb{Z}_{2^q}^d$. For $K \subseteq [d]$ and $\bar{b} \in \mathbb{Z}_{2^q}^{|K|}$ we count the tuples contained in Ξ whose restriction to K equals $\bar{b}$. We define

$$\#_{K,\bar{b}}(\Xi) := \left|\left\{ \bar{c} \in \Xi \;\middle|\; \bar{c}|_K = \bar{b} \right\}\right| \bmod 2.$$

Definition 7.47 (k-ary Blurrer). Let $d \geq k$, $\Xi \subseteq \mathbb{Z}_{2^q}^d$, and $a \in \mathbb{Z}_{2^q}$. The set $\Xi \subseteq \mathbb{Z}_{2^q}^d$ is called a (k, q, a, d)**-blurrer** if it satisfies the following for all $K \subseteq [d]$ with $|K| = k$:

(B1) $\sum \xi = 0$ for all $\xi \in \Xi$.

(B2) If $1 \in K$, then $\#_{K,(a,0,\dots,0)}(\Xi) = 1$.

(B3) If $1 \notin K$, then $\#_{K,\bar{0}}(\Xi) = 1$.

(B4) $\#_{K,\bar{b}}(\Xi) = 0$ for all other pairs of K and $\bar{b}$.

The crucial property of a blurrer is the following:

Lemma 7.48. *Let Ξ be a (k, q, a, d)-blurrer, $K \subseteq [d]$ such that $|K| = k$, and define $\xi_{\mathsf{twst}} := (a, 0, \dots, 0) \in \mathbb{Z}_{2^q}^d$. Every function $f \colon \Xi|_K \to \mathbb{F}_2$ satisfies*

$$\sum_{\xi \in \Xi} f(\xi|_K) = f(\xi_{\mathsf{twst}}|_K).$$

In particular, there is a $\xi_{\mathsf{tw}} \in \Xi$ such that $\xi_{\mathsf{tw}}|_K = \xi_{\mathsf{twst}}|_K$.

Proof. By Conditions B2 and B3, we have $\xi_{\mathsf{twst}}|_K \in \Xi|_K$. Because f takes k-tuples as input, it cannot distinguish whether it is applied to $\xi|_K$ or to $\xi_{\mathsf{twst}}|_K$. Conditions B2, B3, and B4 ensure that when summing over all $\xi \in \Xi$, all summands $f(\xi|_K)$ apart from $f(\xi_{\mathsf{twst}}|_K)$ cancel (by Condition B4 and B2 if $1 \in K$ or by Condition B3 if $1 \notin K$). The existence of a tuple $\xi_{\mathsf{tw}} \in \Xi$ as required follows from Conditions B2 and B3. $\square$

Note that while ξ only contains tuples satisfying $\sum \xi = 0$, we have that $\sum \xi_{\mathsf{twst}} = a$.

Lemma 7.49. *Let Ξ be a (k, q, a, d)-blurrer. Then $|\Xi|$ is odd.*

Proof. Let $K \subseteq [d]$ with $|K| = k$. We partition $\Xi = M \cup N$ into

$$M := \left\{ \xi \in \Xi \;\middle|\; \xi|_K = \xi_{\mathsf{twst}}|_K \right\},$$
$$N := \left\{ \xi \in \Xi \;\middle|\; \xi|_K \neq \xi_{\mathsf{twst}}|_K \right\},$$

where $\xi_{\mathsf{twst}} := (a, 0, \dots, 0)$ is the tuple from Lemma 7.48. The size of $|M|$ is odd by Condition B2 if $1 \in K$ and otherwise by Condition B3. By Condition B4, the size $|N|$ is even. If it was odd, then some $\bar{b}$ would violate Condition B4. $\square$

We now construct blurrers.

Lemma 7.50. *If there is a (k, q, a, d)-blurrer Ξ, then*

1. *there is a (k, q, a, d')-blurrer for every $d' \geq d$,*
2. *Ξ is a (k', q, a, d)-blurrer for every $k' \leq k$, and*
3. *there is a $(k, q, c \cdot a, d)$-blurrer for every $c \in \mathbb{Z}_{2^q}$.*

Proof. To prove the first statement, we just fill up the tuples of Ξ with zeros to be of length d'. To prove the second statement, let $K' \subseteq K \subseteq [d]$ such that $|K| = k$ and let $\bar{b}' \in \mathbb{Z}_{2^q}^{|K'|}$. Then

$$\#_{K',\bar{b}'}(\Xi) = \sum_{\substack{\bar{b} \in \mathbb{Z}_{2^q}^k, \\ \bar{b}|_{K'} = \bar{b}'}} \#_{K,\bar{b}}(\Xi).$$

Assume $1 \in K'$ and $\bar{b}' = (a, 0, \dots, 0)$. Then for $\bar{b} = (a, 0, \dots, 0) \in \mathbb{Z}_{2^q}^k$ we have $\bar{b}|_{K'} = \bar{b}'$ and $\#_{K,\bar{b}}(\Xi) = 1$ by Condition B2. For all other $\bar{b}$, we have $\#_{K,\bar{b}}(\Xi) = 0$ by Condition B4. So the sum is 1. The case that $1 \notin K'$ and $\bar{b} = \bar{0}$ is similar using Condition B3. In the remaining case all summands are 0 by blurrer Condition B4.

To prove the last statement, let $c \in \mathbb{Z}_{2^q}$ and set $\Xi' := \{c \cdot \xi \mid \xi \in \Xi\}$. If $\sum \xi = 0$, then clearly $\sum c \cdot \xi = 0$. We verify blurrer Condition B2, the others are similar. Let $K \subseteq [d]$ of size k and $\bar{b} = (a, 0, \dots, 0) \in \mathbb{Z}_{2^q}^k$. From Conditions B2 and B3 it follows that

$$\#_{K,c\cdot\bar{b}}(\Xi') = \#_{K,\bar{b}}(\Xi) + \sum_{\substack{\bar{b}' \in \mathbb{Z}_{2^q}^k, \\ \bar{b}' \neq \bar{b}, \\ c\cdot\bar{b}' = c\cdot\bar{b}}} \#_{K,\bar{b}'}(\Xi) = 1 + \sum_{\substack{\bar{b}' \in \mathbb{Z}_{2^q}^k, \\ \bar{b}' \neq \bar{b}, \\ c\cdot\bar{b}' = c\cdot\bar{b}}} 0 = 1.$$

□

Lemma 7.51. *Let $m, n \in \mathbb{N}$. If $0 < m < 2^n$, then $\binom{2^n}{m}$ is even. If $m \leq 2^n - 1$, then $\binom{2^n-1}{m}$ is odd.*

Proof. Let $k \in \mathbb{N}$ and consider $\binom{k}{m}$. We write k and m in base 2 representation

$$m = \sum_{i=0}^{j} m_i 2^i, \qquad k = \sum_{i=0}^{j} k_i 2^i$$

for some suitable j and $m_i, k_i \in \{0, 1\}$ for all $i \in \{0, \dots, j\}$. We apply Lucas's Theorem [21]:

$$\binom{k}{m} \bmod 2 = \prod_{i=0}^{j} \binom{k_i}{m_i} \bmod 2,$$

where $\binom{0}{1} = 0$ and $\binom{0}{0} = \binom{1}{0} = \binom{1}{1} = 1$. That is, $\binom{k}{m} \bmod 2 = 0$ if and only if there is an $i \in \{0, \dots, j\}$ such that $k_i = 0$ and $m_i = 1$.

(a) Let $k = 2^n$ and $0 < m < k$. Then there is an $i < n$ such that $m_i = 1$. Because $k = 2^n$, $k_i = 0$ and so $\binom{k}{m}$ is even.

(b) Let $k = 2^n - 1$ and $m \leq k$. For every $i < n$, we have $k_i = 1$. For every i such that $m_i = 1$, it holds that $i < n$. That is, $\binom{k}{m}$ is odd. □

Lemma 7.52. *For every $i \in \mathbb{N}$, there is a $(2^{i-1}-1, i, 2^{i-1}, 2^i-1)$-blurrer.*

Proof. We set $k := 2^{i-1}-1$, $d := 2^i - 1$, and define Ξ as follows:

$$\begin{aligned}
\Xi_1 &:= \left\{ (2^{i-1}, a_2, \dots, a_{2^i-1}) \;\middle|\; \sum_{j=2}^{2^i-1} a_j = 2^{i-1}, a_j \in \{0,1\} \text{ for every } j \right\},\\
\Xi_2 &:= \left\{ (2^{i-1}+1, a_2, \dots, a_{2^i-1}) \;\middle|\; \sum_{j=2}^{2^i-1} a_j = 2^{i-1}-1, a_j \in \{0,1\} \text{ for every } j \right\},\\
\Xi &:= \Xi_1 \cup \Xi_2.
\end{aligned}$$

To verify that Ξ is indeed a $(k, i, 2^{i-1}, d)$-blurrer, let $K \subseteq [d]$ be of size k and $\bar{b} \in \mathbb{Z}_{2^i}^k$. Set $\overline{K} := [d] \setminus K$.

- Let $1 \in K$ and $\bar{b} = (2^{i-1}, 0, \dots, 0)$. Every $\xi \in \Xi$ with $\xi|_K = \bar{b}$ is contained in Ξ_1. Because every $\xi \in \Xi_1$ contains 2^{i-1} many 1-entries, ξ is of length $d = 2^i - 1$, and because $\bar{b}$ contains $k - 1 = 2^{i-1} - 2$ many 0-entries, every $\xi \in \Xi_1$ such that $\xi|_K = \bar{b}$ satisfies $\xi|_{\overline{K}} = (1, \dots, 1)$. So there can be at most one such $\xi \in \Xi_1$. It exists by construction of Ξ_1. Hence, $\#_{K,\bar{b}}(\Xi) = 1$.

- Let $1 \in K$ and $\bar{b} = (2^{i-1}, b_2, \dots, b_{2^i-1})$, let $b_j \in \{0,1\}$ for all j, and let not all b_j equal zero. Again, every $\xi \in \Xi$ such that $\xi|_K = \bar{b}$ is contained in Ξ_1. Because

$$\sum \bar{b} \in \left\{2^{i-1}+1, \dots, 2^{i-1}+k-1\right\} = \left\{2^{i-1}+1, \dots, 2^i-2\right\}$$

  it holds that

$$m := -\sum \bar{b} = \sum \xi|_{\overline{K}} \in \left\{2, \dots, 2^{i-1}-1\right\}$$

  for every $\xi \in \Xi_1$. By construction, $\{\xi \in \Xi_1 \mid \xi|_K = \bar{b}\}|_{[2^i-1]\setminus K}$ is the set of all 0/1-tuples of length 2^{i-1} that contain exactly m many ones. There are exactly $\binom{2^{i-1}}{m}$ such 0/1-tuples of length 2^{i-1}. For all possible values of m, the number $\binom{2^{i-1}}{m}$ is even by Lemma 7.51. We conclude that $\#_{K,\bar{b}}(\Xi) = 0$.

- Let $1 \in K$ and $\bar{b} = (2^{i-1}+1, b_2, \dots, b_{2^i-1})$. Now, every $\xi \in \Xi$ with $\xi|_K = \bar{b}$ is contained in Ξ_2. Then

$$m := -\sum \bar{b} = \sum \xi|_{\overline{K}} \in \left\{1, \dots, 2^{i-1}-1\right\}$$

  for every $\xi \in \Xi_2$ because

$$\sum \bar{b} \in \left\{2^{i-1}+1, \dots, 2^{i-1}+1+(k-1)\right\} = \left\{2^{i-1}+1, \dots, 2^i-1\right\}.$$

  Again, there are $\binom{2^{i-1}}{m}$ many 0/1-tuples extending $\bar{b}$ to a tuple $\xi \in \Xi$, which is an even number by Lemma 7.51 and thus $\#_{K,\bar{b}}(\Xi) = 0$.

- Let $1 \in K$ and $\bar{b}$ not be covered by two cases before. Then there is no ξ satisfying $\xi|_K = \bar{b}$ and so $\#_{K,\bar{b}}(\Xi) = 0$.

- Now the case that $1 \notin K$ remains. If there is no $\xi \in \Xi$ satisfying $\xi|_K = \bar{b}$, then clearly $\#_{K,\bar{b}}(\Xi) = 0$. So assume that there is such a $\xi \in \Xi$.

  – Let us first consider Ξ_1. Let $\xi \in \Xi_1$. Then

  $$0 = \sum \xi = \sum \bar{b} + \sum \xi|_{\overline{K}} = \sum \bar{b} + 2^{i-1} + \sum \xi|_{\overline{K}\setminus\{1\}}.$$

  The tuple $\xi|_{\overline{K}\setminus\{1\}}$ is a 0/1-tuple of length $d - k - 1 = 2^{i-1} - 1$. So

  $$\sum \xi|_{\overline{K}\setminus\{1\}} \in \left\{0, \ldots, 2^{i-1} - 1\right\}.$$

  That is, if $\sum \bar{b} = 0$, we obtain a contradiction because there is no $\xi \in \Xi_1$ satisfying $\xi|_K = \bar{b}$ and $\#_{K,\bar{b}}(\Xi_1) = 0$. Otherwise, all $\xi \in \Xi_1$ satisfying $\xi|_K = \bar{b}$ extend $\bar{b}$ by a 0/1-tuple of length $2^{i-1} - 1$ containing

  $$m := -\sum \bar{b} - 2^{i-1} \in \left\{0, \ldots, 2^{i-1} - 1\right\}$$

  many ones. There are $\binom{2^{i-1}-1}{m}$ many 0/1-tuples of length $d - k - 1 = 2^{i-1} - 1$ that contain exactly m ones, which is an odd number by Lemma 7.51. Hence, $\#_{N,\bar{b}}(\Xi_1) = 1$.

  – Now consider Ξ_2. For every $\xi \in \Xi_2$ such that $\xi|_K = \bar{b}$, it similarly holds that

  $$0 = \sum \bar{b} + 2^{i-1} + 1 + \sum \xi|_{\overline{K}\setminus\{1\}}$$

  and thus $\sum \xi|_{\overline{K}\setminus\{1\}} \in \{0, \ldots, 2^{i-1}\}$. Every $\xi \in \Xi_1$ satisfying $\xi|_K = \bar{b}$ extends $\bar{b}$ by a 0/1-tuple of length $2^{i-1} - 1$ containing

  $$m := -\sum \bar{b} - 2^{i-1} - 1 \in \left\{0, \ldots, 2^{i-1} - 1\right\}$$

  many ones. The number m is again odd by Lemma 7.51. Hence, $\#_{N,\bar{b}}(\Xi_2) = 1$.

  Together, if $\bar{b} = 0$, then $\#_{K,\bar{b}}(\Xi) = \#_{K,\bar{b}}(\Xi_1) + \#_{K,\bar{b}}(\Xi_2) = 0 + 1 = 1$. Otherwise, $\#_{K,\bar{b}}(\Xi_1) + \#_{K,\bar{b}}(\Xi_2) = 1 + 1$ which is 0 modulo 2. □

Computer experiments suggest that, for a given $2^{i-2} \leq k \leq 2^{i-1} - 1$, our choice of $q = i$ is minimal to construct a $(k, q, 2^{i-1}, d)$-blurrer and that $d = 2^i - 1$ could be improved in the case that $k \neq 2^{i-1} - 1$, but that d is minimal in the case that $k = 2^{i-1} - 1$.

We now lift a (k, q, a, d)-blurrer from the ring $\mathbb{Z}_{2^q}$ to the ring $\mathbb{Z}_{2^{q+\ell}}$. Here we have two choices, both of which we need later: The first is via the embedding of $\mathbb{Z}_{2^q}$ into $\mathbb{Z}_{2^{q+\ell}}$ whereas the second will not change the value a.

Lemma 7.53. *Let $q, \ell \in \mathbb{N}$ and $\iota \colon \mathbb{Z}_{2^q} \to \mathbb{Z}_{2^{q+\ell}}$ be the embedding of $\mathbb{Z}_{2^q}$ in $\mathbb{Z}_{2^{q+\ell}}$ defined by $a \mapsto 2^\ell a$. If Ξ is a (k, q, a, d)-blurrer, then $\iota(\Xi)$ is a $(k, q + \ell, \iota(a), d)$-blurrer.*

Proof. The proof is straightforward from the definition. □

Lemma 7.54. *For all $i, \ell \in \mathbb{N}$, there is a $(2^{i-1}-1, i+\ell, 2^{i-1}, 2^i-1)$-blurrer.*

Proof. Let Ξ be the $(2^{i-1}-1, i, 2^{i-1}, 2^i-1)$-blurrer given by Lemma 7.52 and assume $c = 2^{\ell+1}-1 \in \mathbb{Z}_{2^{i+\ell}}$. Let h be the following function that maps Ξ to $\Xi' := \{h(\xi) \mid \xi \in \Xi\}$:

$$\xi \mapsto \Big(-c \cdot \sum_{j=2}^{d} \xi_j, c \cdot \xi_2, \dots, c \cdot \xi_d \Big).$$

The operations are all in $\mathbb{Z}_{2^{i+\ell}}$. By definition $\sum \xi' = 0$ for every $\xi' \in \Xi'$. Let $K \subseteq [d]$ be of size k and $\bar{b} \in \mathbb{Z}_{2^{i+\ell}}^k$. Note that c is a unit because c is odd.

- Let $1 \notin K$. Because c is a unit, multiplication with c is a bijection and thus we have $\#_{K,\bar{b}}(\Xi') = \#_{K,c^{-1}\cdot\bar{b}}(\Xi)$, which is 1 if and only if $c^{-1} \cdot \bar{b} = \bar{b} = \bar{0}$.

- Let $1 \in K$. We argue that also the action of h on the first position is a bijection. Because $\sum \xi = 0$ for all $\xi \in \Xi$, the map $\xi_1 \mapsto \sum_{j=2}^{d} \xi_j$ is a bijection and so is the action of h because c is a unit. So we have $\#_{K,\bar{b}}(\Xi') = \#_{K,\bar{a}}(\Xi)$ for some $\bar{a} \in \mathbb{Z}_{2^i}^k$.

  It holds that $-ca_1 = -(2^{\ell+1}-1)a_1 = a_1 - 2^{\ell+1}a_1$ (over $\mathbb{Z}_{2^{i+\ell}}$). So $a_1 = 2^{i-1}$ if and only if $-ca_1 = 2^{i-1} - 2^{\ell+1}2^{i-1} = 2^{i-1} - 2^{\ell+i} = 2^{i-1} - 0 = 2^{i-1}$. Hence, $\bar{b} = (2^i, 0, \dots, 0)$ if and only if $\bar{a} = (2^i, 0, \dots, 0)$, which is the case if and only if $\#_{K,\bar{b}}(\Xi') = 1$. □

7.7.3 Similarity Matrix for One Round

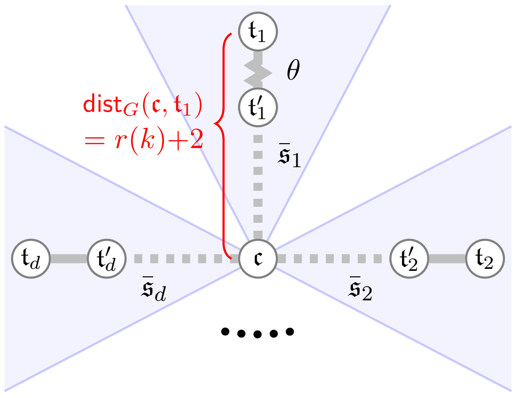
We now construct a similarity matrix k-blurring the twist. To be able to define this matrix, we need bounds on the degree, the girth, and the connectivity of the base graph as well as certain guarantees for the placement of the pebbles. Therefore, we define the following functions that will give us the needed bounds for the given arity k and a number of parameters m. In the following definitions, let $i \in \mathbb{N}$ be the unique number that satisfies $2^{i-1}-1 < k \leq 2^i - 1$ for the given value of k.

$$\begin{aligned}
r(k) &:= \begin{cases} 1 & \text{if } k = 1, \\ \max\{4 \cdot r(k-1)+2, 2k+2\} & \text{otherwise,} \end{cases} \\
\theta(k) &:= \begin{cases} 1 & \text{if } k = 1, \\ i + \theta(k-1) & \text{otherwise,} \end{cases} \\
d(k,m) &:= \begin{cases} 3+m & \text{if } k = 1, \\ \max\{2^{i+1}+m-1, d(k-1, m+1)\} & \text{otherwise,} \end{cases} \\
q(k) &:= 1 + \theta(k).
\end{aligned}$$

Lemma 7.55. *For all $k, m \in \mathbb{N}$,*

- *every regular and $(2k+m+1)$-connected base graph $G = (V, E, \leq)$ of degree $d \geq d(k,m)$ and girth at least $2r(k+1)$,*

- *every base edge $\{\mathsf{t}, \mathsf{t}'\} \in E$,*



7.3 The star used to blur the twist in the k-ary case. The star $\bar{\mathfrak{s}}_1,\dots,\bar{\mathfrak{s}}_d$ in the base graph G. Each path $\bar{\mathfrak{s}}_i$ is contained in its own tree rooted at $\mathfrak{c}$ (depicted in blue) because G has girth greater than $4r(k)$. The base edge $\{\mathfrak{t}_1,\mathfrak{t}_1'\}$ is twisted by f and g by the value θ.

- *every $q \geq q(k)$,*
- *every $\theta = a \cdot 2^{\theta(k)} \in \mathbb{Z}_{2^q}$ (for an arbitrary $a \in \mathbb{Z}_{2^q}$),*
- *all $f, g\colon E \to \mathbb{Z}_{2^q}$ such that $g(\{\mathfrak{t}',\mathfrak{t}\}) = f(\{\mathfrak{t}',\mathfrak{t}\}) + \theta$ and $f(\mathfrak{e}) = g(\mathfrak{e})$ for all $\mathfrak{e} \in E \setminus \{\{\mathfrak{t},\mathfrak{t}'\}\}$, and*
- *every m-tuple $\bar{p} \in A^m$ of $\mathfrak{A}_f := \mathsf{CFI}_{2^q}(G,f)$ and $\mathfrak{A}_g := \mathsf{CFI}_{2^q}(G,g)$, both with universe A, for which $\mathsf{dist}_G(\mathfrak{t},\mathsf{orig}(\bar{p})) > r(k+1)$,*

there is an odd-filled $A^k \times A^k$ matrix S, both orbit-diagonal and orbit-invariant over $(\mathfrak{A}_f,\bar{p})$ and $(\mathfrak{A}_g,\bar{p})$, that k-blurs the twist between $(\mathfrak{A}_f,\bar{p})$ and $(\mathfrak{A}_g,\bar{p})$ and those active region satisfies $\mathsf{A}^{f,g,\bar{p}}(S) \subseteq N_G^{r(k+1)}(\mathfrak{t})$.

Proof. The proof is by induction on k and spans the rest of this section. We already proved the case $k=1$ in Lemmas 7.39 and 7.41 for $q \geq 2$ and $\theta = 2^{q-1}$ using a $(1,q,2^{q-1},d)$ blurrer. This can easily be adapted for the case that $\theta = a \cdot 2^{\theta(1)} = 2a$ for some $a \in \mathbb{Z}_{2^q}$. We start with a $(1,q,2,3)$-blurrer given by Lemma 7.54 and turn it into a $(1,q,2a,d)$-blurrer using Lemma 7.50. Then the proof proceeds exactly as before.

So assume $k > 1$. Let $m \in \mathbb{N}$, $G = (V,E,\leq)$ be a regular and $(2k+m+1)$-connected base graph of degree $d \geq d(k,m)$ and girth at least $2r(k+1)$, and $\{\mathfrak{t},\mathfrak{t}'\} \in E$. The bound on the connectivity of G is needed in the following to apply the results from Sections 7.3.2, 7.4, and 7.6 and we will not further mention this when applying them. Let $q \geq q(k)$. As before, we denote, for every $f\colon E \to \mathbb{Z}_{2^q}$, by $\mathfrak{A}_f$ the CFI structure $\mathsf{CFI}_{2^q}(G,f)$ with universe A (which is equal for all such f). Let $f,g\colon E \to \mathbb{Z}_{2^q}$ such that $f(\mathfrak{e}) = g(\mathfrak{e})$ for all $\mathfrak{e} \in E \setminus \{\{\mathfrak{t},\mathfrak{t}'\}\}$ and $g(\{\mathfrak{t},\mathfrak{t}'\}) = f(\{\mathfrak{t},\mathfrak{t}'\}) + \theta$, where $\theta = a \cdot 2^{\theta(k)}$ for some $a \in \mathbb{Z}_{2^q}$. Furthermore, let $\bar{p} \in A^m$ with $\mathsf{dist}_G(\mathfrak{t},\mathsf{orig}(\bar{p})) > r(k+1)$ be arbitrary but fixed. In particular, the functions f and g do not twist $\mathsf{orig}(\bar{p})$.

Let $\mathfrak{c}$ be a base vertex with $\mathsf{dist}_G(\mathfrak{c},\mathfrak{t}') = r(k)+1$ and $\mathsf{dist}_G(\mathfrak{c},\mathfrak{t}) = r(k)+2$. Choose base vertices $\mathfrak{t} = \mathfrak{t}_1,\dots,\mathfrak{t}_d$ and $\mathfrak{t}' = \mathfrak{t}_1',\dots,\mathfrak{t}_d'$ such that there are simple paths $\bar{\mathfrak{s}}_i = (\mathfrak{c},\dots,\mathfrak{t}_i',\mathfrak{t}_i)$ of length $r(k)+2 > 2k+1$ for all $i \in [d]$ forming a star. Such paths exist because the girth of G is at least $2r(k+1) > 2r(k)+4$ and the degree is d (cf. Figure 7.3).

Claim 1. *We have* $\mathsf{dist}_G(\mathfrak{t}'_i, \mathsf{orig}(\bar{p})) \geq 2r(k)$ *for every* $i \in [d]$ *and* $\mathsf{dist}_G(\mathfrak{t}'_i, \mathfrak{t}'_j) = 2r(k) + 2$ *for every* $i \neq j$.

Proof. By choice of $\mathfrak{c}$, $\mathsf{dist}_G(\mathfrak{c}, \mathfrak{t}'_i) = r(k) + 1$. Let $i, j \in [d]$ such that $i \neq j$. The $\mathfrak{t}'_i$-$\mathfrak{t}'_j$-path obtained by joining $\bar{\mathfrak{s}}_i$ and $\bar{\mathfrak{s}}_j$ (after removing $\mathfrak{t}_i$ and $\mathfrak{t}_j$, respectively) has length $2 \cdot (r(k) + 1)$ by construction. If this path was not a shortest path, then we would obtain a cycle of length less than $4 \cdot (r(k) + 1)$. This contradicts that G has girth at least $2r(k + 1) \geq 4 \cdot (r(k) + 1)$. It follows that

$$\mathsf{dist}_G(\mathfrak{t}'_i, \mathfrak{t}'_j) = \mathsf{dist}_G(\mathfrak{t}'_i, \mathfrak{c}) + \mathsf{dist}_G(\mathfrak{c}, \mathfrak{t}'_j) = 2r(k) + 2.$$

Let $\mathfrak{v} \in \mathsf{orig}(\bar{p})$. Then $\mathsf{dist}_G(\mathfrak{v}, \mathfrak{t}'_1) \leq \mathsf{dist}_G(\mathfrak{v}, \mathfrak{t}'_i) + \mathsf{dist}_G(\mathfrak{t}'_i, \mathfrak{t}'_1)$. By assumption, we have $\mathsf{dist}_G(\mathfrak{v}, \mathfrak{t}'_1) \geq r(k + 1) = 4r(k) + 2$ and by the former argument $\mathsf{dist}_G(\mathfrak{t}'_i, \mathfrak{t}'_1) = 2r(k) + 2$. It follows that

$$\mathsf{dist}_G(\mathfrak{v}, \mathfrak{t}'_i) \geq \mathsf{dist}_G(\mathfrak{v}, \mathfrak{t}'_1) - \mathsf{dist}_G(\mathfrak{t}'_1, \mathfrak{t}'_i) = 4r(k) + 2 - 2r(k) - 2 \geq 2r(k). \qquad \dashv$$

We call a set $C \subseteq V$ a **$2k$-component** if $|C| \leq 2k$ and $G[C]$ is connected. Every $2k$-component is a component of some $2k$-orbit (cf. Definition 7.16). Because k is fixed in this proof, we just call them components in the following.

Definition 7.56. We call a component C

- an ***i*-tip component** if $\{\mathfrak{t}_i, \mathfrak{t}'_i\} \subseteq C$ and $i \in [d]$,
- a **star component** if C intersects the star $\bar{\mathfrak{s}}_1, \dots, \bar{\mathfrak{s}}_d$ nontrivially and $\mathfrak{t}_i \notin C$ for all $i \in [d]$,
- an ***i*-star component** if C is a star component, C intersects the path $\bar{\mathfrak{s}}_i$ nontrivially, and for every other $j \neq i$, C intersects the path $\bar{\mathfrak{s}}_j$ trivially,
- a **star center component** if C is a star component but not an i-star component for every $i \in [d]$,
- and otherwise a **sky component**.

Claim 2. *Let C and C' be components such that $C' \subseteq C$.*

1. *If C is an i-star component, then C' is an i-star component or a sky component*
2. *If C is a star component, then C' is a star component or a sky component.*
3. *If C is an i-tip component, then C' is an i-tip component, an i-star component, or a sky component.*
4. *If C is a star component, then C is an i-star component for some i if and only if $\mathfrak{c} \notin C$.*

Proof. To show Assertion 1, let C be an i-star component. By definition, C only has a nontrivial intersection with $\bar{\mathfrak{s}}_i$ and $\mathfrak{t}_i \notin C$. So every $C' \subseteq C$ either has a trivial intersection with every $\bar{\mathfrak{s}}_j$, i.e., C' is a sky component, or a nontrivial intersection only with $\bar{\mathfrak{s}}_i$ and $\mathfrak{t}_i \notin C' \subseteq C$, i.e., C' is an i-star component. The Assertion 2 where C is a star component is similar.

For Assertion 3, let C be an i-tip component. Because $\mathsf{dist}_G(\mathfrak{t}'_i, \mathfrak{t}'_j) = 2r(k)+2 > 4k+4$ (Claim 1), $\mathsf{dist}_G(\mathfrak{t}'_i, \mathfrak{c}) = r(k)+1 > 2k$, $|C| \leq 2k$, and because $G[C]$ is connected, C has a trivial intersection with every $\bar{\mathfrak{s}}_j$ for all $j \neq i$. Let $C' \subseteq C$. Now C' is an i-tip component if $\{\mathfrak{t}_i, \mathfrak{t}'_i\} \subseteq C'$, an i-star component if otherwise C' intersects nontrivially with $\bar{\mathfrak{s}}_i$, or otherwise a sky component.

Finally, to prove Assertion 4, let C be a star component. If C is also an i-star component, then $\mathfrak{c} \notin C$ because otherwise C intersects nontrivially with all $\bar{\mathfrak{s}}_j$. For the reverse direction, let $\mathfrak{c} \notin C$ and C have a nontrivial intersection with $\bar{\mathfrak{s}}_i$. Because $\mathsf{dist}_G(\mathfrak{t}'_i, \mathfrak{t}'_j) = 2r(k)+2 > 4k+4$, the component C has size $|C| \leq 2k$, its induced subgraph $G[C]$ is connected, and because $\mathfrak{c} \notin C$, the component C cannot have a nontrivial intersection with another $\bar{\mathfrak{s}}_j$. Hence, C is an i-star component. ⊣

To blur the twist, we want to distribute it among the base edges $\{\mathfrak{t}_i, \mathfrak{t}'_i\}$ (for all $i \in [d]$) similar to the 1-ary case. Here we blurred the twist between all base edges adjacent to $\mathfrak{c}$. In the 1-ary case, 1-tuples always had the same type in $(\mathfrak{A}_f, \bar{p})$ and in $(\mathfrak{A}_g, \bar{p})$. However, for k-tuples, this is no longer the case. We now want to construct a function that maps a tuple $\bar{u}$ to a tuple $\bar{v}$ such that $\bar{v}$ has the same type in $(\mathfrak{A}_g, \bar{p})$ as $\bar{u}$ has in $(\mathfrak{A}_f, \bar{p})$. To do so, we use a path-isomorphism on the path $\bar{\mathfrak{s}}_1$. We generalize this to not only "repair" the types for a twist at the base edge $\{\mathfrak{t}_1, \mathfrak{t}'_1\}$ but to consider multiple twists at all base edges $\{\mathfrak{t}_i, \mathfrak{t}'_i\}$.

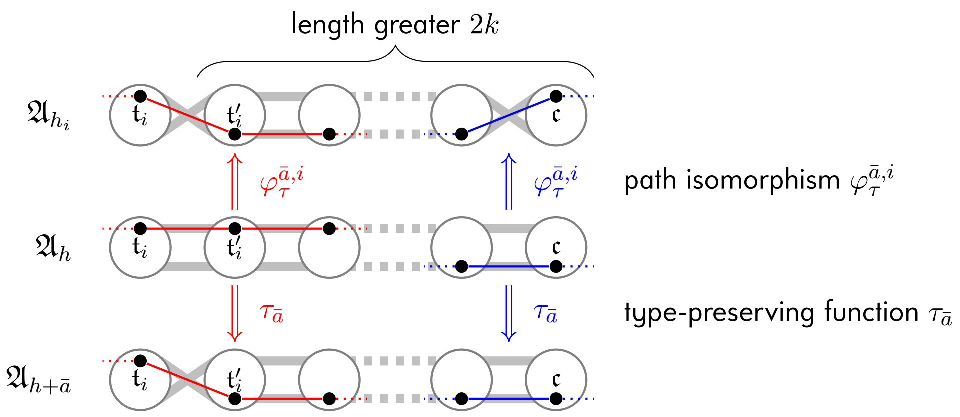
Let $\bar{a} \in \mathbb{Z}_{2^q}^d$. We define a function $\tau_{\bar{a}} \colon A^{\leq 2k} \to A^{\leq 2k}$ that preserves the size of tuples. Set $\varphi_\tau^{\bar{a},i} := \vec{\pi}[a_i, \bar{\mathfrak{s}}_i]$ (cf. Definition 7.5). The function $\tau_{\bar{a}}$ applies $\varphi_\tau^{\bar{a},i}$ to tuples, but only to those components containing some of the base edges $\{\mathfrak{t}_i, \mathfrak{t}'_i\}$. These are precisely the i-tip components:

$$\tau_{\bar{a}}(\bar{u}) := (v_1, \dots, v_{|\bar{u}|}), \text{where } \bar{u} = (u_1, \dots, u_{|\bar{u}|}) \text{ and}$$
$$v_j := \begin{cases} \varphi_\tau^{\bar{a},i}(u_j) & \text{if } \mathsf{orig}(u_j) \in C \text{ and } C \text{ is an } i\text{-tip component of } \bar{u}, \\ u_j & \text{otherwise.} \end{cases}$$

Given a function $h \colon E \to \mathbb{Z}_{2^q}$, we write $h + \bar{a}$ for the function $h' \colon E \to \mathbb{Z}_{2^q}$ such that $h'(\{\mathfrak{t}_i, \mathfrak{t}'_i\}) = h(\{\mathfrak{t}_i, \mathfrak{t}'_i\}) + a_i$ for all $i \in [d]$, and $h'(\mathfrak{e}) = h(\mathfrak{e})$ otherwise.

Claim 3. *Assume $\bar{a} \in \mathbb{Z}_{2^q}^d$, $h \colon E \to \mathbb{Z}_{2^q}$, and $k' \leq 2k$. If $P \in \mathsf{orb}_{k'}((\mathfrak{A}_h, \bar{p}))$, then $\tau_{\bar{a}}(P) \in \mathsf{orb}_{k'}((\mathfrak{A}_{h+\bar{a}}, \bar{p}))$ and $\tau_{\bar{a}}(P)$ has the same type in $(\mathfrak{A}_{h+\bar{a}}, \bar{p})$ as P has in $(\mathfrak{A}_h, \bar{p})$.*

Proof. Let $P \in \mathsf{orb}_{k'}((\mathfrak{A}_h, \bar{p}))$. It suffices to consider the case that P has a single component C because $\tau_{\bar{a}}$ is defined component-wise and, by Lemma 7.17, the type of a disconnected orbit is given by the types of the restrictions to the components. If C does not contain $\mathfrak{t}_i$ and $\mathfrak{t}'_i$ for some $i \in [d]$, then $\tau_{\bar{a}}$ is the identity function (cf. the blue tuple in Figure 7.4). Because P does not cover the twisted base edge, it has the same type in $(\mathfrak{A}_h, \bar{p})$ and in $(\mathfrak{A}_{h+\bar{a}}, \bar{p})$.



7.4 Preserving the types of 2k-orbits. The figure shows the path $\mathfrak{t}_i$ and the three structures $\mathfrak{A}_h$, $\mathfrak{A}_{h+\bar{a}}$, and $\mathfrak{A}_{h_i}$. A big circle depicts a gadget for the given base vertex. An edge twisted by h and $h+\bar{a}$ is depicted by a twisted connection in $\mathfrak{A}_{h+\bar{a}}$. Respectively, an edge twisted by h and h_i is depicted by a twisted connection in $\mathfrak{A}_{h_i}$. The figure shows a red and a blue connected tuple of $\mathfrak{A}_h$ of length at most $2k$. A vertex represents some entry in the tuple. The image of the tuples under the path isomorphism $\varphi_\tau^{\bar{a},i}$ is shown in $\mathfrak{A}_{h_i}$ and the image under the type-preserving function $\tau_{\bar{a}}$ is shown in $\mathfrak{A}_{h+\bar{a}}$. The function $\tau_{\bar{a}}$ applies $\varphi_\tau^{\bar{a},i}$ only to tuples whose origin contains the base edge $\{\mathfrak{t}_i, \mathfrak{t}'_i\}$.

So assume $\{\mathfrak{t}_i, \mathfrak{t}'_i\} \subseteq C$. This is the case for exactly one $i \in [d]$ by Claim 2. Let h_i be the function equal to h for all base edges apart from $h_i(\{\mathfrak{t}_i, \mathfrak{t}'_i\}) := h(\{\mathfrak{t}_i, \mathfrak{t}'_i\}) + a_i$ and $h_i(\{\mathfrak{c}, \mathfrak{c}_i\}) := h(\{\mathfrak{c}, \mathfrak{c}_i\}) - a_i$, where $\mathfrak{c}_i$ is the neighbor of $\mathfrak{c}$ used in the path $\bar{\mathfrak{s}}_i$. By Claim 1, the parameters $\bar{p}$ have distance $2r(k)$ to $\mathfrak{t}'_i$. In particular, the parameters $\bar{p}$ are not contained in $\bar{\mathfrak{s}}_i$. So $\varphi_\tau^{\bar{a},i}$ is an isomorphism between $(\mathfrak{A}_h, \bar{p})$ and $(\mathfrak{A}_{h_i}, \bar{p})$ by Lemma 7.6. Because P is a k'-orbit and $k' \leq 2k$, neither $\mathfrak{c}$ nor its neighbors (in particular not $\mathfrak{c}_i$) are contained in C, because $2k < \mathsf{dist}_G(\mathfrak{c}, \mathfrak{t}'_i) = r(k) + 1$. So $\tau_{\bar{a}}(P) = \varphi_\tau^{\bar{a},i}(P)$ has the same type in $(\mathfrak{A}_{h_i}, \bar{p})$ as P has in $(\mathfrak{A}_h, \bar{p})$. Between $(\mathfrak{A}_{h+\bar{a}}, \bar{p})$ and $(\mathfrak{A}_{h_i}, \bar{p})$ all base edges $\{\mathfrak{t}_\ell, \mathfrak{t}'_\ell\}$ for $\ell \neq i$ and the base edge $\{\mathfrak{c}, \mathfrak{c}_i\}$ are potentially twisted. Because $\mathfrak{c} \notin C$ and $\{\mathfrak{t}_\ell, \mathfrak{t}'_\ell\} \not\subseteq C$ for every $\ell \neq i$, we have $(\mathfrak{A}_{h_i}, \bar{p})[C] = (\mathfrak{A}_{h+\bar{a}}, \bar{p})[C]$ (cf. the red tuple in Figure 7.4). Hence, the type of $\varphi_\tau^{\bar{a},i}(P)$ in $(\mathfrak{A}_{h_i}, \bar{p})$ is equal to the type of $\tau_{\bar{a}}(P)$ in $(\mathfrak{A}_{h+\bar{a}}, \bar{p})$. $\dashv$

We now construct a blurrer for our setting. Let $i \in \mathbb{N}$ such that $2^{i-1} - 1 < k \leq 2^i - 1$.

1. By Lemma 7.54, there is a $(2^i - 1, q - \theta(k-1), 2^i, 2^{i+1} - 1)$-blurrer (note that $q - \theta(k-1) \geq q(k) - \theta(k-1) = i + 1$).
2. We use Lemma 7.53 to turn it into a $(2^i - 1, q, 2^{i+\theta(k-1)}, 2^{i+1} - 1)$-blurrer by embedding $\mathbb{Z}_{2^{q-\theta(k-1)}}$ into $\mathbb{Z}_{2^q}$.
3. We use Lemma 7.50 to get a $(k, q, 2^{i+\theta(k-1)}, 2^{i+1} - 1)$-blurrer because $k \leq 2^i - 1$,
4. then a $(k, q, 2^{i+\theta(k-1)}, d)$-blurrer because $d \geq d(k, m) \geq 2^{i+1} - 1$, and finally
5. a $(k, q, a \cdot 2^{i+\theta(k-1)}, d)$-blurrer Ξ.

By this construction, for every $\xi \in \Xi$ and every $j \in [d]$, there is some $b \in \mathbb{Z}_{2^q}$ such that $\xi(j) = b \cdot 2^{\theta(k-1)}$ because we embedded $\mathbb{Z}_{2^{q-\theta(k-1)}}$ in $\mathbb{Z}_{2^q}$. Let $\xi_{\mathsf{twst}} = (a \cdot 2^{i+\theta(k-1)}, 0, \dots, 0)$ be the tuple given for Ξ by Lemma 7.48. Then $g = f + \xi_{\mathsf{twst}}$. We set $\tau := \tau_{\xi_{\mathsf{twst}}}$.

Corollary 7.57. *If $k' \leq 2k$ and $P \in \mathsf{orb}_{k'}((\mathfrak{A}_f, \bar{p}))$, then $\tau(P) \in \mathsf{orb}_{k'}((\mathfrak{A}_g, \bar{p}))$ and $\tau(P)$ has the same type in $(\mathfrak{A}_g, \bar{p})$ as P has in $(\mathfrak{A}_f, \bar{p})$.*

Proof. The assertion follows from Claim 3. ⊣

We have seen that with the function τ we can "repair" the types of the orbits, but τ introduces inconsistencies between tuples along the path $\bar{\mathfrak{s}}_1$. This can already be seen for $k = 2$: Consider a 2-tuple (u, v) with origin $(\mathfrak{t}_1, \mathfrak{t}'_1)$ and the 2-tuple (v, w) with origin $(\mathfrak{t}'_1, \mathfrak{u})$, where $\mathfrak{u}$ is the next base vertex in the path $\bar{\mathfrak{s}}_1$. Then $\tau((u, v)) = (u, v')$ for some $v' \neq v$ but $\tau((v, w)) = (v, w)$. So clearly the composed 4-tuple (u, v, v, w) has a different type than (u, v', v, w). For these inconsistencies, we use the blurrer Ξ as follows.

We now define another function which according to a $\xi \in \Xi$ "distributes" the twists among the base edges $\{\mathfrak{t}_i, \mathfrak{t}'_i\}$ using a star isomorphism. We associate with each $\xi \in \Xi$ a function $\psi_\xi \colon A^{\leq k} \to A^{\leq k}$ that preserves tuple sizes: Set $\varphi_\xi := \pi^*[\xi, \bar{\mathfrak{s}}_1, \ldots, \bar{\mathfrak{s}}_d]$ (cf. Definition 7.7). Then define $\psi_\xi \colon A^{\leq k} \to A^{\leq k}$ as follows: Let $\bar{u} \in A^{\leq k}$. We set

$$\psi_\xi(\bar{u}) := (v_1, \ldots, v_{|\bar{u}|}), \text{where } \bar{u} = (u_1, \ldots, u_{|\bar{u}|}) \text{ and}$$
$$v_i := \begin{cases} \varphi_\xi(u_i) & \text{if } \mathsf{orig}(u_i) \in C \text{ and } C \text{ is a star component of } \bar{u}, \\ u_i & \text{otherwise.} \end{cases}$$

That is, we apply φ_ξ to all components of $\bar{u}$ that do not contain the tips of the star $\bar{\mathfrak{s}}_1, \ldots, \bar{\mathfrak{s}}_d$. On the sky components φ_ξ is the identity function anyway. From now on, we will identify ξ with ψ_ξ and write $\xi(\bar{u})$.

Claim 4. *Let $\bar{u} \in A^\ell$ and the components of $\bar{u}$ be partitioned into D and D'. Then $\tau_{\bar{a}}(\bar{u}) = \tau_{\bar{a}}(\bar{u}_D)\tau_{\bar{a}}(\bar{u}_{D'})$ and $\xi(\bar{u}) = \xi(\bar{u}_D)\xi(\bar{u}_{D'})$ for all $\bar{a} \in \mathbb{Z}_{2^q}^d$ and $\xi \in \Xi$.*

Proof. The claim is immediate because $\tau_{\bar{a}}$ and ψ_ξ are defined component-wise. ⊣

Claim 5. *For all $\bar{a} \in \mathbb{Z}_{2^q}^d$ and $\xi \in \Xi$, the functions $\tau_{\bar{a}}$, ψ_ξ, and every automorphism $\varphi \in \mathsf{Aut}((\mathfrak{A}_f, \bar{p})) = \mathsf{Aut}((\mathfrak{A}_g, \bar{p}))$ commute.*

Proof. The functions $\tau_{\bar{a}}$ and ψ_ξ are defined component-wise by isomorphisms. We saw in Section 7.3.1 that isomorphisms between CFI structures are composed of automorphisms of each gadget. Because the automorphism group of a gadget is abelian (Lemma 7.15), the said functions commute. ⊣

Definition 7.58 (Orbit-Automorphism). A function $\zeta \colon A^{\leq 2k} \to A^{\leq 2k}$ is called an **orbit-automorphism** if, for every $P \in \mathsf{orb}_{k'}((\mathfrak{A}_f, \bar{p})) = \mathsf{orb}_{k'}((\mathfrak{A}_g, \bar{p}))$ with $k' \leq 2k$, there is an automorphism $\varphi_P \in \mathsf{Aut}((\mathfrak{A}_f, \bar{p})) = \mathsf{Aut}((\mathfrak{A}_g, \bar{p}))$ such that $\zeta(\bar{u}) = \varphi_P(\bar{u})$ for all $\bar{u} \in P$.

That is, an orbit-automorphism is a function whose action on a single orbit is the action of an automorphism. For different orbits, the corresponding automorphisms may be different. This matches the definition of an orbit-invariant matrix (cf. Definition 7.25), which is invariant under all orbit-automorphisms.

Claim 6. *Every $\xi \in \Xi$ is an orbit-automorphism.*

Proof. By Lemma 7.17, it suffices to show the claim for connected orbits $P \in \mathsf{orb}_{k'}((\mathfrak{A}_f, \bar{p}))$ with $k' \leq 2k$. If the origin of P is not a star component, then ξ is the identity function on P and so clearly an orbit-automorphism. Otherwise, the origin $C := \mathsf{orig}(P)$ is a star component. Then $\mathfrak{t}_i \notin C$ for all $i \in [d]$. We show that there are paths $\bar{\mathfrak{s}}'_i = (\mathfrak{t}'_i, \mathfrak{t}_i, \dots, \mathfrak{t}_1, \mathfrak{t}'_1)$ that are completely disjoint from $\mathsf{orig}(\bar{p})$ and possibly apart from $\mathfrak{t}'_i, \mathfrak{t}'_1$ disjoint from C. Set $C' := C \setminus \{\mathfrak{t}'_i \mid i \in [d]\}$. Consider the graph $G - (C' \cup \mathsf{orig}(\bar{p}))$. We removed at most $|\mathsf{orig}(\bar{p})| + k' \leq 2k + m$ many base vertices. Because G is $(2k+m+1)$-connected, the claimed paths exist in $G - (C' \cup \mathsf{orig}(\bar{p}))$. We then use path-isomorphisms $\varphi_i := \vec{\pi}[\xi(i), \bar{\mathfrak{s}}'_i]$ for all $i \in [d] \setminus \{1\}$ to move the twist introduced by ψ_ξ at the base edge $\{\mathfrak{t}_i, \mathfrak{t}'_i\}$ to the base edge $\{\mathfrak{t}_1, \mathfrak{t}'_1\}$. We set $\psi := \psi_\xi \circ \varphi_2 \circ \dots \circ \varphi_d$. Now, we have that $(\mathfrak{A}_f, \bar{p}) \cong \psi((\mathfrak{A}_f, \bar{p})) = (\mathfrak{A}_f, \bar{p})$ by Lemmas 7.6 and 7.8. Let $\bar{u} \in P$. The isomorphisms φ_i are the identity on atoms in $\mathsf{orig}(P)$ because φ_i is the identity on $\mathfrak{t}'_i$ and $\mathfrak{t}'_1$ (cf. Definition 7.5) and, for every $i \in [d]$, other base vertices in $\mathsf{orig}(P)$ are not contained in $\bar{\mathfrak{s}}'_i$. That is, $P = \psi(P) = \psi_\xi(P) = \xi(P)$ and ξ is an orbit-automorphism. $\dashv$

We show that ξ and τ_ξ together preserve types. That is, for $\bar{u}\bar{v}$ of length at most $2k$, the tuple $\tau_\xi(\xi(\bar{u}))\tau_\xi(\xi(\bar{v}))$ has the same type as the tuple $\tau_\xi(\bar{u}\bar{v})$. This is not true when applying only τ_ξ to $\bar{u}$ and $\bar{v}$ because their origins may overlap.

Claim 7. *Let $h\colon E \to \mathbb{Z}_{2^q}$, $k' \leq 2k$, $P \in \mathsf{orb}_{k'}((\mathfrak{A}_h, \bar{p}))$, and $\xi \in \Xi$. For all $\bar{u}\bar{v} \in A^{k'}$, we have $\bar{u}\bar{v} \in P$ if and only if $\tau_\xi(\xi(\bar{u}))\tau_\xi(\xi(\bar{v})) \in \tau_\xi(P)$.*

Proof. Set $P_1 := P|_{\{1,\dots,|\bar{u}|\}}$ and $P_2 := P|_{\{|\bar{u}|+1,\dots,k'\}}$. Let $C^i_1, \dots, C^i_{\ell_i}$ be the components of P_i for every $i \in [2]$. Because τ_ξ and ξ are defined component-wise, it suffices by Lemma 7.17 to assume that P has a single component C. We need to verify that $\bar{u}\bar{v} \in P$ if and only if $\tau_\xi(\xi(\bar{u}))\tau_\xi(\xi(\bar{v})) \in \tau_\xi(P)$. The component C is the union of the C^i_j. We make the following case distinction:

(a) Assume C is an n-tip component. For every $i \in [2]$, let D^T_i be the set of the n-tip components in C^i_j, let D^S_i be the set of the n-star components in C^i_j, and let D^R_i be the set of sky components in C^i_j. This yields a partition of all C^i_j by Claim 2.

Then we have, by the definitions of τ_ξ and ξ, that ξ is the identity function on D^T_i, τ_ξ is the identity on D^S_i, and both are the identity on D^R_i. That is,

$$\begin{aligned}
&\tau_\xi\big(\xi(\bar{u})\big)\tau_\xi\big(\xi(\bar{v})\big)\\
&= \tau_\xi\Big(\xi\big(\bar{u}_{D^T_1}\bar{u}_{D^S_1}\bar{u}_{D^R_1}\big)\Big)\tau_\xi\Big(\xi\big(\bar{v}_{D^T_2}\bar{v}_{D^S_2}\bar{v}_{D^R_2}\big)\Big)\\
&= \tau_\xi(\bar{u}_{D^T_1})\xi(\bar{u}_{D^S_1})\tau_\xi(\bar{u}_{D^R_1})\tau_\xi(\bar{v}_{D^T_2})\xi(\bar{v}_{D^S_2})\tau_\xi(\bar{v}_{D^R_2}). \qquad (\star)
\end{aligned}$$

When working on the whole component C, τ_ξ applies $\varphi^{\xi,n}_\tau$ to atoms in components of D^S_1 and D^S_2 because C is an n-tip component. We see that $\varphi_\xi|_{D^S_i} = \varphi^{\xi,n}_\tau|_{D^S_i}$ because D^S_i is an n-star-component and so does not contain $\mathfrak{c}$ (cf. Definitions 7.5 and 7.7 and the definitions of φ_ξ and $\varphi^{\xi,n}_\tau$). It follows that

$$(\star) = \tau_\xi(\bar{u}_{D^T_1}\bar{u}_{D^S_1}\bar{u}_{D^R_1}\bar{v}_{D^T_2}\bar{v}_{D^S_2}\bar{v}_{D^R_2}) = \tau_\xi(\bar{u}\bar{v}).$$

So $\bar{u}\bar{v} \in P$ if and only if $\tau_\xi(\xi(\bar{u}))\tau_\xi(\xi(\bar{v})) = \tau_\xi(\bar{u}\bar{v}) \in \tau_\xi(P)$.

(b) Otherwise, C is not a tip component. We distinguish two more cases:

- If C is a star component, let D_i^S be the set of star components C_j^i and D_i^R be the set of sky components C_j^i for all $i \in [2]$. There are no tip components among the C_j^i by Claim 2. So again, we partitioned all components C_j^i. Now τ_ξ is the identity function on all D_i^S and D_i^R and ξ is the identity on all D_i^R. So we have $\bar{u}\bar{v} \in P$ if and only if $\tau_\xi(\xi(\bar{u}))\tau_\xi(\xi(\bar{v})) = \xi(\bar{u})\xi(\bar{v}) = \xi(\bar{u}\bar{v}) \in \tau_\xi(P) = P$ by Claim 6.
- Otherwise, C is a sky component and both τ_ξ and ξ are the identity function on C and all C_j^i and the claim follows immediately. ⊣

Using the blurrer Ξ, we will be able to blur the twist in many cases, but not in all. The problem is the following: If we only look at k many of the $\{\mathfrak{t}_i, \mathfrak{t}_i'\}$ base edges, then the blurrer properties will ensure that we cannot see the twist, i.e., "summing" over all elements in the blurrer maps a $2k$-orbit of $(\mathfrak{A}_f, \bar{p})$ to a $2k$-orbit of the same type in $(\mathfrak{A}_g, \bar{p})$ similar to the 1-ary case. Let us briefly recall the arguments to prove Lemma 7.37, which shows that summing over blurrer elements indeed yields a matrix blurring the twist for arity 1. There are two cases: First, for a tuple (u, v) with origin $(\mathfrak{c}, \mathfrak{c})$ the action of every $\xi \in \Xi$ was the action of an automorphism, so $(\xi(u), v)$ and $(u, \xi^{-1}(v))$ were in the same orbit. Second, for a tuple (u, v) with origin $(\mathfrak{c}, \mathfrak{t}_1)$ only one index (namely the first for $\mathfrak{t}_1$) was relevant: $\xi(u)v$ and $\xi'(u)v$ are in the same orbit if $\xi(1) = \xi'(1)$. So whenever $\xi(1) = \xi'(1)$, the terms for $\xi(u)v$ and $\xi'(u)v$ canceled in the summation. The blurrer properties ensured that only one term for $\xi(u)v$ of the same type in $(\mathfrak{A}_g, \bar{p})$ as uv in $(\mathfrak{A}_f, \bar{p})$ remained.

For arity k the two cases (automorphism or blurrer properties) can be mixed. Consider $k = 2$ and a 4-tuple $\bar{u}$ with origin $(\mathfrak{c}, \mathfrak{t}_1', \mathfrak{c}, \mathfrak{t}_1)$. Then, for every $\xi, \xi' \in \Xi$, both $\xi(u_1u_2)u_3u_4$ and $\xi'(u_1u_2)u_3u_4$ are in the same orbit if and only if $\xi = \xi'$ (because fixing one atom with origin $\mathfrak{c}$ splits the gadget of $\mathfrak{c}$ into singleton orbits). That is, we cannot argue only with the blurrer properties. In general, however, the two tuples $\xi(u_1u_2)u_3u_4$ and $u_1u_2\xi^{-1}(u_3u_4)$ are not in the same orbit because no automorphism that maps u_1u_2 to $\xi(u_1u_2)$ is the identity on u_4 but $\xi^{-1}(u_3u_4)$ is the identity on u_4. So we also cannot argue only with automorphisms. The techniques of the 1-ary case can only be applied if $\mathfrak{c}$ is not in the origin of at least one of u_1u_2 and u_3u_4.

In general, let $P \in \mathsf{orb}_{2k}((\mathfrak{A}_f, \bar{p}))$ and $\bar{u} \in P$. Then in χ^P the first k positions of $\bar{u}$ will serve as row index and the remaining k positions as column index. The problem with the blurrer only occurs if both the first and second half of $\bar{u}$ contain $\mathfrak{c}$ in its origin. So we make a case distinction on whether a k-orbit contains $\mathfrak{c}$ in its origin.

Definition 7.59 (Blurrable Orbit). An orbit $P \in \mathsf{orb}_k((\mathfrak{A}_f, \bar{p}))$ is **blurrable** if $\mathfrak{c} \notin \mathsf{orig}(P)$.

In order to blur the twist for non-blurrable orbits, we use a recursive approach. Because $\sum \xi = 0$, $\mathfrak{A}_{g-\xi}$ is isomorphic to $\mathfrak{A}_g$ for every $\xi \in \Xi$. Let $p_\mathfrak{c}$ be an arbitrary atom with origin $\mathfrak{c}$. Our goal now is to blur the twist between $(\mathfrak{A}_f, \bar{p}p_\mathfrak{c})$ and $(\mathfrak{A}_{g-\xi}, \bar{p}p_\mathfrak{c})$. This will exactly undo the action of a $\xi \in \Xi$, when we consider an orbit that fixes an atom with origin $\mathfrak{c}$. We exploit the high girth of G to blur the twists at each $\{\mathfrak{t}_i, \mathfrak{t}_i'\}$ independently. Before we start to blur twists between $(\mathfrak{A}_f, \bar{p}p_\mathfrak{c})$ and $(\mathfrak{A}_{g-\xi}, \bar{p}p_\mathfrak{c})$, we first have to show that τ_ξ and ξ are compatible with orbits when fixing the additional atom $p_\mathfrak{c}$. The following two claims are in some sense refinements of Claims 3 and 7.

Claim 8. *For all $\bar{a} \in \mathbb{Z}_{2^q}$ and $k' \leq 2k$, every orbit $P \in \mathsf{orb}_{k'}((\mathfrak{A}_{f+\bar{a}}, \bar{p}p_\mathfrak{c}))$ satisfying $\mathsf{orig}(P) \cap N^1_G(\mathfrak{c}) = \emptyset$ is contained in $\mathsf{orb}_{k'}((\mathfrak{A}_{f+\bar{a}}, \bar{p}))$.*

Proof. Let $P \in \mathsf{orb}_{k'}((\mathfrak{A}_{f+\bar{a}}, \bar{p}p_\mathfrak{c}))$ such that $\mathsf{orig}(P) \cap N^1_G(\mathfrak{c}) = \emptyset$ (recall that $N^\ell_G(\mathfrak{c})$ is the ℓ-neighborhood of $\mathfrak{c}$). By Lemma 7.10, it suffices to show that two tuples $\bar{u}, \bar{v} \in A^{k'}$, such that $\mathsf{orig}(\bar{u}) = \mathsf{orig}(\bar{v})$ is disjoint from $N^1_G(\mathfrak{c})$, have the same type in $(\mathfrak{A}_{f+\bar{a}}, \bar{p})$ if and only if they have the same type in $(\mathfrak{A}_{f+\bar{a}}, \bar{p}p_\mathfrak{c})$. Because $\mathsf{orig}(P) \cap N^1_G(\mathfrak{c}) = \emptyset$, the components of $\bar{u}\bar{p}p_\mathfrak{c}$ are the components of $\bar{u}\bar{p}$ and $p_\mathfrak{c}$. Hence, if $\bar{u}$ and $\bar{v}$ have the same type in $(\mathfrak{A}_{f+\bar{a}}, \bar{p})$, then they also have the same type in $(\mathfrak{A}_{f+\bar{a}}, \bar{p}p_\mathfrak{c})$. The other inclusion is trivial. ⊣

Claim 9. *Let $\bar{a} \in \mathbb{Z}^d_{2^q}$, $h \colon E \to \mathbb{Z}_{2^q}$, and $k' \leq 2k$. Assume $P \in \mathsf{orb}_{k'}((\mathfrak{A}_h, \bar{p}p_\mathfrak{c}))$. Then $\tau_{\bar{a}}(P) \in \mathsf{orb}_{k'}((\mathfrak{A}_{h+\bar{a}}, \bar{p}p_\mathfrak{c}))$ and $\tau_{\bar{a}}(P)$ has the same type in $(\mathfrak{A}_{h+\bar{a}}, \bar{p}p_\mathfrak{c})$ as P has in $(\mathfrak{A}_h, \bar{p}p_\mathfrak{c})$.*

Proof. Let $P \in \mathsf{orb}_{k'}((\mathfrak{A}_h, \bar{p}p_\mathfrak{c}))$, R be the set of components C of P with $C \cap N^1_G(\mathfrak{c}) \neq \emptyset$, and D be the set of all remaining components of P. Then $P = P|_R \times P|_D$ by Claim 7.17. Similarly, $\tau_{\bar{a}}(P) = \tau_{\bar{a}}(P)|_R \times \tau_{\bar{a}}(P)|_D$. Every component C in R does not, for every $i \in [d]$, contain the base edge $\{\mathfrak{t}_i, \mathfrak{t}'_i\}$ because $|C| \leq k'$ but every path $\bar{\mathfrak{s}}_i$ has length $r(k)+2 > 2k+1 \geq k'+1$. That is, $\tau_{\bar{a}}$ is the identity on $P|_R$ and so $P|_R = \tau_{\bar{a}}(P)|_R$ and $P|_R$ has the same type in $(\mathfrak{A}_{h+\bar{a}}, \bar{p}p_\mathfrak{c})$ as it has in $(\mathfrak{A}_h, \bar{p}p_\mathfrak{c})$. By Claim 8, the orbit $\tau_{\bar{a}}(P)|_D$ is an orbit of $(\mathfrak{A}_h, \bar{p})$ and has, by Claim 3, the same type in $(\mathfrak{A}_h, \bar{p})$ as $\tau_{\bar{a}}(P)|_D$ has in $(\mathfrak{A}_{h+\bar{a}}, \bar{p})$. It follows that P has the same type in $(\mathfrak{A}_h, \bar{p}p_\mathfrak{c})$ as $\tau_{\bar{a}}(P)$ has in $(\mathfrak{A}_{h+\bar{a}}, \bar{p}p_\mathfrak{c})$. ⊣

Claim 10. *Assume that $Q \in \mathsf{orb}_{k'}((\mathfrak{A}_{g-\xi}, \bar{p}p_\mathfrak{c}))$ for some $\xi \in \Xi$ and $k' \leq 2k-2$. Then $\tau_\xi(Q) \in \mathsf{orb}_{k'}((\mathfrak{A}_g, \bar{p}p_\mathfrak{c}))$ and $\tau_\xi(Q)$ has the same type as Q. Let C be the connected component of $G[\mathsf{orig}(Q) \cup \{\mathfrak{c}\}]$ containing $\mathfrak{c}$, let R be the set of components of Q contained in C, let D be the set of all other components of Q, and let $\bar{w}\bar{v} \in A^{k'}$. Then $\bar{w}\bar{v} \in Q$ if and only if $\bar{w}_R\xi(\tau_\xi(\bar{w}_D))\bar{v}_R\xi(\tau_\xi(\bar{v}_D)) \in \tau_\xi(Q)$.*

Proof. We split Q in $Q = Q|_R \times Q|_D$ by Lemma 7.17. Because $k' \leq 2k-2$, components in R cannot be tip components. Hence, $\tau_\xi(Q) = Q|_R \times \tau_\xi(Q|_D)$. By Claim 8, $Q|_D$ is also an orbit of $(\mathfrak{A}_{g-\xi}, \bar{p})$ because its origin has distance greater than 1 to $\mathsf{orig}(p_\mathfrak{c})$. Then $\tau_\xi(Q|_D)$ has the same type in $(\mathfrak{A}_g, \bar{p})$ and is also an orbit of $(\mathfrak{A}_g, \bar{p}p_\mathfrak{c})$ of the same type by Claim 3. It follows that $\tau_\xi(Q)$ has the same type as Q. Let $\bar{w}\bar{v} \in A^{k'}$. Using the splitting above, from Claim 7 it follows that $\bar{w}_D\bar{v}_D \in Q|_D$ if and only if $\xi(\tau_\xi(\bar{w}_D))\xi(\tau_\xi(\bar{v}_D)) \in \tau_\xi(Q|_D)$. The claim follows because $Q = Q|_R \times Q|_D$. ⊣

Now we construct matrices $(k-1)$-blurring the twist between $(\mathfrak{A}_f, \bar{p}p)$ and $(\mathfrak{A}_{g-\xi}, \bar{p}p_\mathfrak{c})$. For all $\xi \in \Xi$ and $j \in [d+1]$, we define $g^{\xi,j} \colon E \to \mathbb{Z}_{2^q}$ to be the following function:

$$g^{\xi,j}(\mathfrak{e}) := \begin{cases} f(\{\mathfrak{t}_i, \mathfrak{t}'_i\}) & \text{if } \mathfrak{e} = \{\mathfrak{t}_i, \mathfrak{t}'_i\} \text{ for some } i \geq j, \\ (g-\xi)(\mathfrak{e}) & \text{otherwise.} \end{cases}$$

Note that $f(\mathfrak{e}) = g(\mathfrak{e}) = g^{\xi,j}(\mathfrak{e})$ for all $\mathfrak{e}$ different from the base edges $\{\mathfrak{t}_i, \mathfrak{t}'_i\}$, $g^{\xi,1} = f$, $g^{\xi,d+1} = g - \xi$, and the only possibly twisted base edge by $g^{\xi,j}$ and $g^{\xi,j+1}$ is $\{\mathfrak{t}_j, \mathfrak{t}'_j\}$ for every $j \in [d]$. Define $N_j := N^{r(k)}_G(\mathfrak{t}_j)$ for every $j \in [d]$.

Claim 11. *For every $\xi \in \Xi$ and every $j \in [d]$, there is an odd-filled $A^{k-1} \times A^{k-1}$ matrix $S^{\xi,j}$, both orbit-diagonal and orbit-invariant over $(\mathfrak{A}_{g^{\xi,j}}, \bar{p}p_{\mathfrak{c}})$ and $(\mathfrak{A}_{g^{\xi,j+1}}, \bar{p}p_{\mathfrak{c}})$, which $(k-1)$-blurs the twist between $(\mathfrak{A}_{g^{\xi,j}}, \bar{p}p_{\mathfrak{c}})$ and $(\mathfrak{A}_{g^{\xi,j+1}}, \bar{p}p_{\mathfrak{c}})$, and whose active region satisfies $\mathsf{A}^{g^{\xi,j},g^{\xi,j+1},\bar{p}p_{\mathfrak{c}}}(S^{\xi,j}) \subseteq N_j$. In particular, $S^{\xi,j} = \mathbb{1}$ if $\xi(j) = 0$.*

Proof. Let $\xi \in \Xi$ and $j \in [d]$. If $\xi(j) = 0$, then $g^{\xi,j} = g^{\xi,j+1}$ and $S^{\xi,j} := \mathbb{1}$ trivially satisfies the claim. Otherwise, the matrix $S^{\xi,j}$ is obtained from the induction hypothesis: The number of parameters increased by one, but we consider tuples of length $k-1$. We continue to consider $\mathbb{Z}_{2^q}$.

- Clearly, $q \geq q(k) \geq q(k-1)$, the degree of G is $d \geq d(k,m) \geq d(k-1, m+1)$, and the girth of G is at least $4r(k) + 2 > 2r(k)$.
- We have $2k + m + 1 \geq 2(k-1) + (m+1) + 1$ and so G satisfies the connectivity condition.
- By construction, we have that $g^{\xi,j}(\mathfrak{e}) = g^{\xi,j+1}(\mathfrak{e})$ for every $j \in [d]$ and every $\mathfrak{e} \in E \setminus \{\{\mathfrak{t}_j, \mathfrak{t}'_j\}\}$.
- We consider the value of the twist: Let $j \in [d]$. Then $\xi(j) = b \cdot 2^{\theta(k-1)}$ for some $b \in \mathbb{Z}_{2^q}$ (as shown when constructing the blurrer Ξ before Corollary 7.57). If $j \neq 1$, then it holds that

  $$g^{\xi,j+1}(\{\mathfrak{t}_j, \mathfrak{t}'_j\}) = g^{\xi,j}(\{\mathfrak{t}_j, \mathfrak{t}'_j\}) - b \cdot 2^{\theta(k-1)}.$$

  If otherwise $j = 1$, then we have

  $$g^{\xi,2}(\{\mathfrak{t}_1, \mathfrak{t}'_1\}) = g(\{\mathfrak{t}_1, \mathfrak{t}'_1\}) - \xi(1) = g^{\xi,1}(\{\mathfrak{t}_1, \mathfrak{t}'_1\}) - \xi(1) + \theta$$

  because $g^{\xi,1} = f$ and $g(\{\mathfrak{t}_1, \mathfrak{t}'_1\}) = f(\{\mathfrak{t}_1, \mathfrak{t}'_1\}) + \theta$. By assumption, we have that $\theta = a \cdot 2^{\theta(k)} = a \cdot 2^{i+\theta(k-1)}$ for some $a \in \mathbb{Z}_{2^q}$. Clearly,

  $$-\xi(1) + \theta = -b \cdot 2^{\theta(k-1)} + a \cdot 2^{i+\theta(k-1)} = (a \cdot 2^i - b) \cdot 2^{\theta(k-1)}.$$

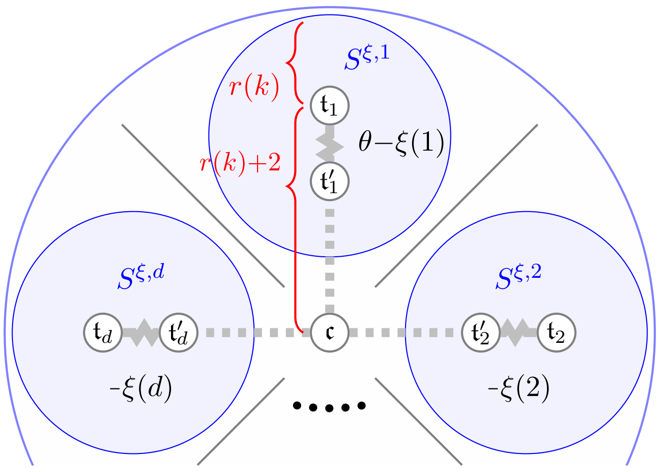
  So, in all cases, each edge $\{\mathfrak{t}_j, \mathfrak{t}'_j\}$ is twisted by a value $c \cdot 2^{\theta(k-1)}$ for some $c \in \mathbb{Z}_{2^q}$.
- Assume $i \in [d]$. We have $\mathsf{dist}_G(\mathfrak{t}'_i, \mathsf{orig}(\bar{p})) \geq 2r(k)$ by Claim 1. Thus, we have $\mathsf{dist}_G(\mathfrak{t}_i, \mathsf{orig}(\bar{p})) > r(k)$. We have $\mathsf{dist}_G(\mathfrak{t}'_i, \mathsf{orig}(p_{\mathfrak{c}})) = \mathsf{dist}_G(\mathfrak{t}'_i, \mathfrak{c}) = r(k) + 1$ by construction. Hence, we have $\mathsf{dist}_G(\mathfrak{t}_i, \mathsf{orig}(\bar{p}p_{\mathfrak{c}})) > r(k)$. ⊣

We now define, for every $\xi \in \Xi$, the $A^{k-1} \times A^{k-1}$ matrix S^{ξ} as follows:

$$S^{\xi} := S^{\xi,1} \cdot \ldots \cdot S^{\xi,d},$$

where $S^{\xi,j}$ is the matrix given by Claim 11 for ξ and j.

Claim 12. *For every $\xi \in \Xi$, the matrix S^{ξ} is odd-filled and both orbit-diagonal and orbit-invariant over $(\mathfrak{A}_f, \bar{p}p_{\mathfrak{c}})$ and $(\mathfrak{A}_{g-\xi}, \bar{p}p_{\mathfrak{c}})$. Moreover, it $(k-1)$-blurs the twist between $(\mathfrak{A}_f, \bar{p}p_{\mathfrak{c}})$ and $(\mathfrak{A}_{g-\xi}, \bar{p}p_{\mathfrak{c}})$, and satisfies $\mathsf{A}^{f,g-\xi,\bar{p}p_{\mathfrak{c}}}(S^{\xi}) \subseteq \bigcup_{i=1}^{d} N_i$.*



7.5 The active region of the k-ary similarity matrix. The active region of $S^{\xi,i}$ is contained in $N_i = N_G^{r(k)}(\mathfrak{t}_i)$ for every i (drawn in blue). Because the blurrer only acts nontrivially on tuples contained in $N_G^{r(k)}(\mathfrak{c})$ and the type-preserving function acts only nontrivially on tuples contained in $N_G^{r(k)+2+k}(\mathfrak{c})$, the active region of S^{ξ} is contained in $N_G^{2r(k)+2}(\mathfrak{c})$ (drawn in light blue). Distances in the star are drawn in red. The values of the twist at the base edges $\{\mathfrak{t}_i, \mathfrak{t}_i'\}$ are drawn below or right of them.

Proof. For every $j \in [d]$, the matrix $S^{\xi,j}$ is odd-filled and both orbit-diagonal and orbit-invariant over $(\mathfrak{A}_{g^{\xi,j}}, \bar{p}p_{\mathfrak{c}})$ and $(\mathfrak{A}_{g^{\xi,j+1}}, \bar{p}p_{\mathfrak{c}})$, it $(k-1)$-blurs the twist between $(\mathfrak{A}_{g^{\xi,j}}, \bar{p}p_{\mathfrak{c}})$ and $(\mathfrak{A}_{g^{\xi,j+1}}, \bar{p}p_{\mathfrak{c}})$, and it satisfies $\mathsf{A}^{g^{\xi,j}, g^{\xi,j+1}, \bar{p}p_{\mathfrak{c}}}(S^{\xi,j}) \subseteq N_j$ by Claim 11. Because $f = g^{\xi,1}$ and $g - \xi = g^{\xi,d+1}$, the matrix $S^{\xi} = S^{\xi,1} \cdot \ldots \cdot S^{\xi,d}$ is orbit-diagonal and orbit-invariant over $(\mathfrak{A}_f, \bar{p}p_{\mathfrak{c}})$ and $(\mathfrak{A}_{g-\xi}, \bar{p}p_{\mathfrak{c}})$ by Lemmas 7.45 and 7.26. By Lemma 7.28, the matrix S^{ξ} is odd-filled. It $(k-1)$-blurs the twist between $(\mathfrak{A}_f, \bar{p}p_{\mathfrak{c}})$ and $(\mathfrak{A}_{g-\xi}, \bar{p}p_{\mathfrak{c}})$ by Lemma 7.23. Finally, it follows from Lemma 7.44 that

$$\mathsf{A}^{f, g-\xi, \bar{p}p_{\mathfrak{c}}}(S^{\xi}) \subseteq \bigcup_{i=1}^{d} \mathsf{A}^{g^{\xi,j}, g^{\xi,j+1}, \bar{p}p_{\mathfrak{c}}}(S^{\xi,j}) \subseteq \bigcup_{i=1}^{d} N_i. \qquad \dashv$$

Claim 13. *For every pair of distinct $i, j \in [d]$, it holds that $N_i \cap N_j = \emptyset$.*

Proof. Let $i \neq j$. Assume that there is an $\mathfrak{u} \in N_i \cap N_j$. Then there is a path from $\mathfrak{t}_i$ to $\mathfrak{t}_j$ of length at most $2r(k)$. By construction $\mathsf{dist}_G(\mathfrak{t}_i, \mathfrak{c}) = \mathsf{dist}_G(\mathfrak{t}_j, \mathfrak{c}) = r(k) + 2$ and so $\mathfrak{c} \notin N_i$ and $\mathfrak{c} \notin N_j$. But that means that there is a cycle of length at most $\mathsf{dist}_G(\mathfrak{t}_i, \mathfrak{c}) + \mathsf{dist}_G(\mathfrak{t}_j, \mathfrak{c}) + \mathsf{dist}_G(\mathfrak{t}_i, x) + \mathsf{dist}_G(\mathfrak{t}_j, x) \leq 4r(k) + 4$ contradicting that G has girth at least $2r(k+1) \geq 8r(k) + 4$ (cf. Figure 7.5). $\dashv$

Claim 14. *For every $\xi \in \Xi$, every $P \in \mathsf{orb}_{k-1}((\mathfrak{A}_f, \bar{p}p_{\mathfrak{c}}))$, every $Q \in \mathsf{orb}_{k-1}((\mathfrak{A}_{g-\xi}, \bar{p}p_{\mathfrak{c}}))$ of the same type in $(\mathfrak{A}_{g-\xi}, \bar{p}p_{\mathfrak{c}})$ as P in $(\mathfrak{A}_f, \bar{p}p_{\mathfrak{c}})$, all $\bar{u} \in P$, $\bar{v} \in Q$, and $\varphi \in \mathsf{Aut}((\mathfrak{A}_f, \bar{p}))$, the matrix S^{ξ} satisfies $S^{\xi}(\bar{u}, \bar{v}) = S^{\xi}(\varphi(\bar{u}), \varphi(\bar{v}))$.*

Proof. Let $\xi \in \Xi$. By Claim 12, the matrix S^{ξ} is orbit-invariant over $(\mathfrak{A}_f, \bar{p}p_{\mathfrak{c}})$ and $(\mathfrak{A}_{g-\xi}, \bar{p}p_{\mathfrak{c}})$ and thus satisfies the claim for all $\varphi \in \mathsf{Aut}((\mathfrak{A}_f, \bar{p}p_{\mathfrak{c}}))$. But now we also want to consider automorphisms not stabilizing $p_{\mathfrak{c}}$. So let $P \in \mathsf{orb}_{k-1}((\mathfrak{A}_f, \bar{p}p_{\mathfrak{c}}))$, let $Q \in \mathsf{orb}_{k-1}((\mathfrak{A}_{g-\xi}, \bar{p}p_{\mathfrak{c}}))$ be of the same type in $(\mathfrak{A}_{g-\xi}, \bar{p}p_{\mathfrak{c}})$ as P in $(\mathfrak{A}_f, \bar{p}p_{\mathfrak{c}})$, $\bar{u} \in P$,

$\bar{v} \in Q$, and $\varphi \in \mathsf{Aut}((\mathfrak{A}_f, \bar{p}))$. Let R be the set of components of $\bar{u}$ (and thus of $\bar{v}$) containing a base vertex of $N_G^1(\mathfrak{c})$ (so in particular $\mathfrak{c}$ itself). Let D be the set of remaining components. Because $\bar{u}_D\bar{v}_D$ and $\varphi(\bar{u}_D\bar{v}_D)$ are in the same orbit in $(\mathfrak{A}_{f+\bar{a}}, \bar{p})$, they are also in the same orbit in $(\mathfrak{A}_{f+\bar{a}}, \bar{p}p_\mathfrak{c})$ by Claim 8. Hence, there is a $\psi \in \mathsf{Aut}((\mathfrak{A}_{f+\bar{a}}, \bar{p}p_\mathfrak{c}))$ satisfying $\varphi(\bar{u}_D\bar{v}_D) = \psi(\bar{u}_D\bar{v}_D)$. We now use the fact that S^ξ is orbit-invariant over $(\mathfrak{A}_f, \bar{p}p_\mathfrak{c})$ and $(\mathfrak{A}_{g-\xi}, \bar{p}p_\mathfrak{c})$:

$$\begin{aligned} S^\xi(\bar{u}, \bar{v}) &= S^\xi(\bar{u}_R\bar{u}_D, \bar{v}_R\bar{v}_D) \\ &= S^\xi\Big(\bar{u}_R\psi(\bar{u}_D), \bar{v}_R\psi(\bar{v}_D)\Big) \\ &= S^\xi\Big(\bar{u}_R\varphi(\bar{u}_D), \bar{v}_R\varphi(\bar{v}_D)\Big). \end{aligned} \tag{$\star$}$$

For every $C \in R$, it holds that $C \not\subseteq \mathsf{A}^{f,g-\xi,\bar{p}p_\mathfrak{c}}(S^\xi) \subseteq \bigcup_{i=1}^d N_i$ because $N_G^1(\mathfrak{c}) \cap \bigcup_{i=1}^d N_i = \emptyset$, which follows from $N_i = N_G^{r(k)}(\mathfrak{t}_i)$ and $\mathsf{dist}_G(\mathfrak{c}, \mathfrak{t}_i) = r(k) + 2$. So we can apply Condition A2 of the active region because $\bar{u}_R = \bar{v}_R$ if and only if $\varphi(\bar{u}_R) = \varphi(\bar{v}_R)$:

$$\begin{aligned} (\star) &= S^\xi\Big(\varphi(\bar{u}_{D_\mathfrak{c}})\bar{u}_{D_R}, \varphi(\bar{v}_{D_\mathfrak{c}})\bar{v}_{D_R}\Big) \\ &= S^\xi\Big(\varphi(\bar{v}), \varphi(\bar{v})\Big). \end{aligned}$$ ⊣

Claim 15. *Let $k' \leq k-1$, $\xi \in \Xi$, $P' \in \mathsf{orb}_{k'}((\mathfrak{A}_f, \bar{p}p_\mathfrak{c}))$, $K \subseteq [d]$, D be the set of all components C of P' satisfying $C \subseteq N_i$ for some $i \in K$, and R be the set of remaining components. Let $Q' = \tau(P') \in \mathsf{orb}_{k'}((\mathfrak{A}_g, \bar{p}p_\mathfrak{c}))$, $\bar{u} \in P'$, and $\bar{v} \in Q'$. Then*

$$\sum_{\bar{w}_D \in P'|_D} S^\xi(\bar{w}_D\bar{u}_R, \bar{v}_D\bar{v}_R) = \left(\prod_{i \in [d]\setminus K} S^{\xi,i}\right)(\bar{v}_D\bar{u}_R, \bar{v}_D\bar{v}_R)$$

and $\mathsf{A}^{f,f+\bar{a},\bar{p}p_\mathfrak{c}}(\prod_{i\in[d]\setminus K} S^{\xi,i}) \subseteq \bigcup_{i\in[d]\setminus K} N_i$ for the tuple $\bar{a} \in \mathbb{Z}_{2^q}^d$ whose entries satisfy $a_i = (g-\xi)(\{\mathfrak{t}_i, \mathfrak{t}'_i\}) - f(\{\mathfrak{t}_i, \mathfrak{t}'_i\})$ if $i \notin K$ and $a_i = 0$ otherwise for every $i \in [d]$.

Proof. Recall that $S^\xi = S^{\xi,1} \cdot \ldots \cdot S^{\xi,d}$ and that $\mathsf{A}^{g^{\xi,j},g^{\xi,j+1},\bar{p}p_\mathfrak{c}}(S^{\xi,j}) \subseteq N_j$ for every $j \in [d]$ by Claim 11. The first part of the claim follows from repeated application of Lemma 7.46 using that the sets N_j are disjoint (Claim 13). The second part follows from repeated application of Lemmas 7.43 and 7.44. ⊣

We introduce more notation. Let $\bar{u} \in A^{k'}$ such that $\mathfrak{c} \in \mathsf{orig}(\bar{u})$. Then $\bar{u}^{-\mathfrak{c}} \in A^{k'-1}$ is the tuple obtained from $\bar{u}$ by deleting the first entry with origin $\mathfrak{c}$. This first entry is denoted by $\bar{u}_\mathfrak{c}$. Similarly to our convention for $\bar{u}_C$ for a component C in Section 7.3.2, we write $\bar{u}_\mathfrak{c}\bar{u}^{-\mathfrak{c}}$ not for concatenation but for inserting $\bar{u}_\mathfrak{c}$ at the correct position such that $\bar{u}_\mathfrak{c}\bar{u}^{-\mathfrak{c}} = \bar{u}$. Now we are ready to define the $A^k \times A^k$ matrix S. For $j \in \{k, 2k\}$ we set $\mathbf{P}_j := \mathsf{orb}_j((\mathfrak{A}_f, \bar{p}))$. We define the $P \times \tau(P)$ block of S for every $P \in \mathbf{P}_k$.

$$S_{P\times\tau(P)}(\bar{u}, \bar{v}) := \begin{cases} \displaystyle\sum_{\substack{\xi\in\Xi,\\ \tau(\xi(\bar{u}))=\bar{v}}} 1 & \text{if } P \text{ is blurrable,} \\ \displaystyle\sum_{\substack{\xi\in\Xi,\\ \tau_\xi(\xi(\bar{u}_\mathfrak{c}))=\bar{v}_\mathfrak{c}}} S^\xi\Big(\xi(\bar{u}^{-\mathfrak{c}}), \tau_\xi^{-1}(\bar{v}^{-\mathfrak{c}})\Big) & \text{if } P \text{ is non-blurrable.} \end{cases}$$

All other blocks are zero. We first check that we used the matrices S^ξ only for row and column indices of the same type:

Claim 16. *Let $P \in \mathbf{P}_k$ be non-blurrable, $\bar{u} \in P$, and $\bar{v} \in \tau(P)$. If $\bar{u}$ has the same type in $(\mathfrak{A}_f, \bar{p})$ as $\bar{v}$ has in $(\mathfrak{A}_g, \bar{p})$, then for every $\xi \in \Xi$ such that $\tau_\xi(\xi(\bar{u}_\mathfrak{c})) = \bar{v}_\mathfrak{c}$, the tuple $\xi(\bar{u}^{-\mathfrak{c}})$ has the same type in $(\mathfrak{A}_f, \bar{p}p_\mathfrak{c})$ as $\tau_\xi^{-1}(\bar{v}^{-\mathfrak{c}})$ has in $(\mathfrak{A}_{g-\xi}, \bar{p}p_\mathfrak{c})$.*

Proof. Let $\xi \in \Xi$ such that $\tau_\xi(\xi(\bar{u}_\mathfrak{c})) = \bar{v}_\mathfrak{c}$. Let R be the star component of $\mathsf{orig}(P)$ containing $\mathfrak{c}$ and let D be the set of remaining components of $\mathsf{orig}(P)$. Then $P = P|_R \times P|_D$ by Lemma 7.17. In particular, $\tau(P) = P|_R \times \tau(P|_D)$ because τ is the identity on star components. The orbit $\tau(P|_D)$ has the same type in $(\mathfrak{A}_g, \bar{p})$ as $P|_D$ has in $(\mathfrak{A}_f, \bar{p})$ (Corollary 7.57) and $\tau_\xi(\tau(P|_D))$ has thus the same type in $(\mathfrak{A}_{g-\xi}, \bar{p})$ as $P|_D$ has in $(\mathfrak{A}_f, \bar{p})$ (Claim 3). Because $\mathfrak{c} \in R$ (P is non-blurrable), all components in D have distance at least 2 to $\mathfrak{c}$. Thus, $P|_D$ is also an orbit of $(\mathfrak{A}_f, \bar{p}p_\mathfrak{c})$ (Claim 8) and has the same type in $(\mathfrak{A}_f, \bar{p}p_\mathfrak{c})$ as $\tau_\xi(\tau(P|_D))$ has in $(\mathfrak{A}_{g-\xi}, \bar{p}p_\mathfrak{c})$. Because ξ is an orbit-automorphism (Claim 6), $\xi(P|_D) = P|_D$. It follows that $\xi(\bar{u}_D) \in P|_D$ and $\tau_\xi(\bar{v}_D) \in \tau_\xi(\tau(P|_D))$ have the same type.

It suffices to show that $\xi(\bar{u}_R)$ has the same type in $(\mathfrak{A}_f, \bar{p}p_\mathfrak{c})$ as $\tau_\xi(\bar{v}_R)$ has in $(\mathfrak{A}_{g-\xi}, \bar{p}p_\mathfrak{c})$. Because R contains star components, τ_ξ is the identity on R and so $\bar{v}_R \in P|_R$. Further, $\xi(\bar{u}_R) \in P|_R$ because ξ is an orbit-automorphism. That is, there is an automorphism $\psi \in \mathsf{Aut}((\mathfrak{A}_f, \bar{p}))$ such that $\psi(\xi(\bar{u}_R)) = \bar{v}_R$. Because by assumption $\tau_\xi(\xi(\bar{u}_\mathfrak{c})) = \xi(\bar{u}_\mathfrak{c}) = \bar{v}_\mathfrak{c}$ (τ_ξ is the identity on atoms with origin $\mathfrak{c}$), ψ is the identity on atoms with origin $\mathfrak{c}$ and thus $\psi \in \mathsf{Aut}((\mathfrak{A}_f, \bar{p}p_\mathfrak{c}))$. That is, $\xi(\bar{u}_R)$ and $\bar{v}_R$ are in the same orbit of $(\mathfrak{A}_f, \bar{p}p_\mathfrak{c})$. Because R contains star components, it does not contain any base edge twisted by f and $g - \xi$ and thus $\xi(\bar{u}_R)$ and $\bar{v}_R$ have the same type in $(\mathfrak{A}_f, \bar{p}p_\mathfrak{c})$ and in $(\mathfrak{A}_{g-\xi}, \bar{p}p_\mathfrak{c})$. ⊣

Claim 17. *The matrix S is orbit-diagonal over $(\mathfrak{A}_f, \bar{p})$ and $(\mathfrak{A}_g, \bar{p})$.*

Proof. By definition, the only nonzero blocks of S are the $P \times \tau(P)$ blocks. By Corollary 7.57, P and $\tau(P)$ have the same type for every $P \in \mathbf{P}_k$. ⊣

Claim 18. *The matrix S is orbit-invariant over $(\mathfrak{A}_f, \bar{p})$ and $(\mathfrak{A}_g, \bar{p})$.*

Proof. Let $\varphi \in \mathsf{Aut}((\mathfrak{A}_f, \bar{p})) = \mathsf{Aut}((\mathfrak{A}_g, \bar{p}))$, $P \in \mathbf{P}_k$, $\bar{u} \in P$, and $\bar{v} \in Q := \tau(P)$. We make a case distinction: Assume that P is blurrable. The functions τ, ξ, and φ commute (Claim 5) and thus $\tau(\xi(\varphi(\bar{u}))) = \varphi(\tau(\xi(\bar{u})))$ for every $\xi \in \Xi$. Because φ is a bijection, $\varphi(\tau(\xi(\bar{u}))) = \varphi(\bar{v})$ if and only if $\tau(\xi(\bar{u})) = \bar{v}$. So

$$S\Big(\varphi(\bar{u}), \varphi(\bar{v})\Big) = \sum_{\substack{\xi \in \Xi,\\ \tau(\xi(\varphi(\bar{u})))=\varphi(\bar{v})}} 1 = \sum_{\substack{\xi \in \Xi,\\ \varphi(\tau(\xi(\bar{u})))=\varphi(\bar{v})}} 1 = \sum_{\substack{\xi \in \Xi,\\ \tau(\xi(\bar{u}))=\bar{v}}} 1 = S(\bar{u}, \bar{v}).$$

Otherwise, assume that P is non-blurrable. Then, for every $\xi \in \Xi$, the following holds because τ_ξ, ξ, and φ commute by Claim 5 and because S^ξ is invariant under φ by Claim 14:

$$\begin{aligned} &S^\xi\Big(\xi(\varphi(\bar{u}^{-\mathfrak{c}})), \tau_\xi^{-1}(\varphi(\bar{v}^{-\mathfrak{c}}))\Big) \\ &= S^\xi\Big(\varphi(\xi(\bar{u}^{-\mathfrak{c}})), \varphi(\tau_\xi^{-1}(\bar{v}^{-\mathfrak{c}}))\Big) \\ &= S^\xi\Big(\xi(\bar{u}^{-\mathfrak{c}}), \tau_\xi^{-1}(\bar{v}^{-\mathfrak{c}})\Big). \end{aligned}$$

Because τ_ξ, ξ, and φ commute and because φ is a bijection, we have $\tau_\xi(\xi(\varphi(\bar{u}_\mathfrak{c}))) = \varphi(\bar{v}_\mathfrak{c})$

if and only if $\tau_\xi(\xi(\bar{u}_\mathfrak{c})) = \bar{v}_\mathfrak{c}$. Hence,

$$\begin{aligned} S\big(\varphi(\bar{u}), \varphi(\bar{v})\big) &= \sum_{\substack{\xi\in\Xi,\\ \tau_\xi(\xi(\varphi(\bar{u}_\mathfrak{c})))=\varphi(\bar{v}_\mathfrak{c})}} S^\xi\Big(\xi(\varphi(\bar{u}^{-\mathfrak{c}})), \tau_\xi^{-1}(\varphi(\bar{v}^{-\mathfrak{c}}))\Big) \\ &= \sum_{\substack{\xi\in\Xi,\\ \tau_\xi(\xi(\bar{u}_\mathfrak{c}))=\bar{v}_\mathfrak{c}}} S^\xi\Big(\xi(\bar{u}^{-\mathfrak{c}}), \tau_\xi^{-1}(\bar{v}^{-\mathfrak{c}})\Big) \\ &= S(\bar{u}, \bar{v}). \end{aligned}$$ ⊣

Claim 19. *The matrix S is odd-filled.*

Proof. Let $P \in \mathbf{P}_k$ and $\bar{u} \in P$. Then $\tau(\bar{u}) \in \tau(P)$ and because every $\xi \in \Xi$ is an orbit-automorphism (Claim 6), we have $\xi(\bar{u}) \in P$ and $\tau(\xi(\bar{u})) \in \tau(P)$ for every $\xi \in \Xi$. Assume first that P is blurrable. Now, we sum the entries in the row indexed by $\bar{u}$ (in $\mathbb{F}_2$):

$$\sum_{\bar{v}\in\tau(P)} S(\bar{u},\bar{v}) = \sum_{\bar{v}\in\tau(P)} \sum_{\substack{\xi\in\Xi,\\ \tau(\xi(\bar{u}))=\bar{v}}} 1 = \sum_{\substack{\xi\in\Xi,\\ \tau(\xi(\bar{u}))\in\tau(P)}} 1 = |\Xi| \bmod 2.$$

The last step holds because $\tau(\xi(\bar{u})) \in \tau(P)$ for every $\xi \in \Xi$. Finally, $|\Xi|$ is odd by Lemma 7.49 and so the number of ones in the row indexed by $\bar{u}$ is odd, too.

Assume otherwise that P is non-blurrable and set $Q := \tau(P)$, which is of the same type as P (Corollary 7.57). For every $\xi \in \Xi$, we set $Q_\xi := \{\bar{v}^{-\mathfrak{c}} \mid \bar{v} \in Q, \tau_\xi(\xi(\bar{u}_\mathfrak{c})) = \bar{v}_\mathfrak{c}\}$. Then $Q_\xi \in \mathsf{orb}_{k-1}((\mathfrak{A}_g, \bar{p}p_\mathfrak{c})) = \mathsf{orb}_{k-1}((\mathfrak{A}_{g-\xi}, \bar{p}p_\mathfrak{c}))$ by Corollaries 7.19 and 7.13 (the center $\mathfrak{c}$ has distance greater than 1 to $\mathsf{orig}(\bar{p})$).

$$\begin{aligned} \sum_{\bar{v}\in Q} S(\bar{u},\bar{v}) &= \sum_{\bar{v}\in Q} \sum_{\substack{\xi\in\Xi,\\ \tau_\xi(\xi(\bar{u}_\mathfrak{c}))=\bar{v}_\mathfrak{c}}} S^\xi\Big(\xi(\bar{u}^{-\mathfrak{c}}), \tau_\xi^{-1}(\bar{v}^{-\mathfrak{c}})\Big) \\ &= \sum_{\xi\in\Xi} \sum_{\substack{\bar{v}\in Q,\\ \tau_\xi(\xi(\bar{u}_\mathfrak{c}))=\bar{v}_\mathfrak{c}}} S^\xi\Big(\xi(\bar{u}^{-\mathfrak{c}}), \tau_\xi^{-1}(\bar{v}^{-\mathfrak{c}})\Big) \\ &= \sum_{\xi\in\Xi} \sum_{\bar{v}\in Q_\xi} S^\xi\Big(\xi(\bar{u}^{-\mathfrak{c}}), \tau_\xi^{-1}(\bar{v})\Big) \\ &= \sum_{\xi\in\Xi} \sum_{\bar{v}\in\tau_\xi^{-1}(Q_\xi)} S^\xi\Big(\xi(\bar{u}^{-\mathfrak{c}}), \bar{v}\Big). \qquad (\star) \end{aligned}$$

In the first line of the equation, $\xi(\bar{u}^{-\mathfrak{c}})$ has the same type in $(\mathfrak{A}_f, \bar{p}p_\mathfrak{c})$ as $\tau_\xi^{-1}(\bar{v}^{-\mathfrak{c}})$ has in $(\mathfrak{A}_{g-\xi}, \bar{p}p_\mathfrak{c})$ for every $\xi \in \Xi$ such that $\tau_\xi(\xi(\bar{u}_\mathfrak{c})) = \bar{v}_\mathfrak{c}$ (Claim 16). One sees that we always sum over the same column indices of S^ξ (for a fixed ξ) and we only manipulate the way in which we express the sum. Hence, in the last line, $\xi(\bar{u}^{-\mathfrak{c}})$ has the same type in $(\mathfrak{A}_f, \bar{p}p_\mathfrak{c})$ as $\bar{v}$ has in $(\mathfrak{A}_{g-\xi}, \bar{p}p_\mathfrak{c})$. By Claim 9, $\tau_\xi^{-1}(Q_\xi) \in \mathsf{orb}_{k-1}((\mathfrak{A}_{g-\xi}, \bar{p}p_\mathfrak{c}))$, so $\sum_{\bar{v}\in\tau_\xi^{-1}(Q_\xi)} S^\xi(\xi(\bar{u}^{-\mathfrak{c}}), \bar{v}) = 1$ because we sum over all entries in one row of a block on the diagonal of S^ξ and S^ξ is odd-filled (Claim 12). It follows that

$$(\star) = \sum_{\xi\in\Xi} 1 = |\Xi| \bmod 2 = 1.$$

For the last step we used again that $|\Xi|$ is odd by Lemma 7.49. ⊣

Now it follows from Lemma 7.31 that S is invertible.

Claim 20. $\mathsf{A}^{f,g,\bar{p}}(S) \subseteq N_G^{r(k+1)}(\mathfrak{t})$.

Proof. We show that $N_G^{r(k+1)}(\mathfrak{t})$ satisfies the conditions of the active region. This implies that $\mathsf{A}^{f,g,\bar{p}}(S) \subseteq N_G^{r(k+1)}(\mathfrak{t})$. To show Condition A1 of the active region, we have to show that $C \subseteq N_G^{r(k+1)}(t)$ for every component $C \in \mathsf{A}^{f,g,\bar{p}}(S,P)$ for every $P \in \mathbf{P}_k$. Let $P \in \mathbf{P}_k$ be blurrable and C be a component of P. By definition of ξ and τ, the matrix S is only active on C if C is a star or a tip component. But this means that $C \subseteq N_G^{r(k)+2+k}(\mathfrak{c})$ because C is connected, of size at most k, and contains a base vertex $\mathfrak{u}$ of some $\bar{\mathfrak{s}}_i$ (which have length $r(k)+2$, cf. Figure 7.5). Because $\mathsf{dist}_G(\mathfrak{c},\mathfrak{t}) = r(k)+2$, every base vertex with distance at most $r(k)+2+k$ to $\mathfrak{c}$ has distance at most $2r(k)+4+k \leq 4r(k)+2 = r(k+1)$ to $\mathfrak{t}$ (one immediately sees that $r(k) \geq k \geq 2$). Thus,

$$C \subseteq N_G^{r(k)+2+k}(\mathfrak{c}) \subseteq N_G^{r(k+1)}(\mathfrak{t}).$$

Let otherwise $P \in \mathbf{P}_k$ be non-blurrable and C be a component of P. Then S is possibly active on C if $C \subseteq N_G^{r(k)+2+k}(\mathfrak{t})$ (as seen in the blurrable case) or

$$C \subseteq \mathsf{A}^{f,g-\xi,\bar{p}p_{\mathfrak{c}}}(S^{\xi}) \subseteq \bigcup_{i \in d} N_i$$

for some $\xi \in \Xi$ (Claim 12). Recall that $N_i = N_G^{r(k)}(\mathfrak{t}_i)$ and that $\mathsf{dist}_G(\mathfrak{t}_i,\mathfrak{c}) = r(k)+2$ by construction. If follows that

$$\bigcup_{i \in d} N_i \subseteq N_G^{2r(k)+2}(\mathfrak{c}) \subseteq N_G^{r(k+1)}(\mathfrak{t})$$

because every base vertex with distance $r(k)$ to some base vertex $\mathfrak{t}_i$ has distance at most $3r(k)+4 \leq 4r(k)+2$ to $\mathfrak{t}$ (we again use that $r(k) \geq 2$ for $k \geq 2$). To prove Condition A2 of the active region, we see that ξ, τ, and τ_ξ are defined component-wise (for every $\xi \in \Xi$) and that $\mathsf{A}^{f,g-\xi,\bar{p}p_{\mathfrak{c}}}(S^{\xi}) \subseteq \bigcup_{i \in d} N_i$. ⊣

Now, we want to show that S actually k-blurs the twist. For a $2k$-orbit $P \in \mathbf{P}_{2k}$, we set

$$P_1 := P|_{\{1,\dots,k\}} \in \mathbf{P}_k,$$
$$P_2 := P|_{\{k+1,\dots,2k\}} \in \mathbf{P}_k$$

to be the unique k-orbits such that $P \subseteq P_1 \times P_2$ (and similar for $Q \in \mathbf{Q}_{2k}$). Our aim is to prove that $\chi^P \cdot S = S \cdot \chi^Q$ for every $P \in \mathbf{P}_{2k}$ and $Q := \tau(P)$. Because S is orbit-diagonal and $\chi^P_{P_1 \times P_2}$ and $\chi^Q_{Q_1 \times Q_2}$ are the only nonzero blocks of χ^P and χ^Q, it suffices to show that

$$\chi^P_{P_1 \times P_2} S_{P_2 \times Q_2} = S_{P_1 \times Q_1} \chi^Q_{Q_1 \times Q_2}.$$

We begin with blurrable orbits and define the set of indices $i \in [d]$ of the blurrer Ξ which are relevant for a blurrable orbit.

Definition 7.60 (Occupied Indices). The set of **occupied indices** $\mathsf{Occ}(P)$ of a blurrable orbit $P \in \mathbf{P}_k$ is the set of indices $i \in [d]$, such that there is a component C of P that satisfies $C \subseteq N_i$ or C is an i-star component.

Note that the definition also covers i-tip components because every i-tip component is contained in N_i. Also note that $|\mathsf{Occ}(P)| \leq k$ because P is a k-orbit. The following lemma states that if one of P_1 and P_2 is blurrable, say P_1, then it does not matter whether we apply ξ or ξ' to P_2 as long as ξ and ξ' agree on $\mathsf{Occ}(P_1)$.

Claim 21. *Assume $P \in \mathbf{P}_{2k}$, $\xi, \xi' \in \Xi$, and $\bar{u}, \bar{v} \in A^k$. If $P_1 \in \mathbf{P}_k$ is blurrable and $\xi|_{\mathsf{Occ}(P_1)} = \xi'|_{\mathsf{Occ}(P_1)}$, then $\bar{u}\xi(\bar{v}) \in P$ if and only if $\bar{u}\xi'(\bar{v}) \in P$. Likewise, if P_2 is blurrable and $\xi|_{\mathsf{Occ}(P_2)} = \xi'|_{\mathsf{Occ}(P_2)}$, then $\xi(\bar{u})\bar{v} \in P$ if and only if $\xi'(\bar{u})\bar{v} \in P$. The same holds for $Q := \tau(P)$.*

Proof. Consider the case that P_1 is blurrable. Assume that $\bar{u}\xi(\bar{v}) \in P$. Let $C^i_1, \dots, C^i_{\ell_i}$ be the components of P_i for every $i \in [2]$. Because τ and ξ are defined component-wise, it suffices by Lemma 7.17 to assume that P is a k'-orbit of $(\mathfrak{A}_f, \bar{p})$ for some $k' \leq k$ and has a single component C. The component C is the union of all C^i_j. First consider the case that C is a star component. Because P_1 is blurrable, we can partition all components C^1_j by Claim 2 into $D^1_1, \dots, D^1_d$ and D^1_R such that D^1_i contains all i-star components for all $i \in [d]$ and D^1_R the remaining sky components. Set $\xi'' := \xi' - \xi$. Then $\xi''|_{\mathsf{Occ}(P_1)} = 0$, $\xi''(\xi(\bar{v})) = \xi'(\bar{v})$, and ξ'' is the identity function on the components in all D^1_i because every D^1_i only contains i-star components and is nonempty only if $i \in \mathsf{Occ}(P_1)$. Additionally, ξ'' is the identity on the sky components D^1_R. Hence $\varphi_{\xi''}(\bar{u}) = \bar{u}$ by Definition 7.7. Because C is a star component, we have that

$$\xi''\Big(\bar{u}\xi(\bar{v})\Big) = \varphi_{\xi''}\Big(\bar{u}\xi(\bar{v})\Big) = \varphi_{\xi''}(\bar{u})\varphi_{\xi''}\Big(\xi(\bar{v})\Big) = \bar{u}\xi''\Big(\xi(\bar{v})\Big) = \bar{u}\xi'(\bar{v}).$$

Because ξ'' is an orbit-automorphism by Claim 6, it holds that $\bar{u}\xi(\bar{v}) \in P$ if and only if $\xi''(\bar{u}\xi(\bar{v})) = \bar{u}\xi'(\bar{v}) \in P$. If otherwise C is not a star component, then none of the C^i_j is a star component, ξ and ξ' are the identity function on $\bar{v}$, and the claim follows immediately. The cases for P_2 and Q are analogous. $\dashv$

Claim 22. *For every $P \in \mathbf{P}_{2k}$, $\xi \in \Xi$ such that $\xi|_{\mathsf{Occ}(P)} = \xi_{\mathsf{twst}}|_{\mathsf{Occ}(P)}$, and $\bar{u}, \bar{v} \in A^k$ it holds that $\bar{u}\bar{v} \in P$ if and only if $\tau(\xi(\bar{u}))\tau(\xi(\bar{v})) \in \tau(P)$.*

Proof. Let $P \in \mathbf{P}_{2k}$, $\xi \in \Xi$ such that $\xi|_{\mathsf{Occ}(P)} = \xi_{\mathsf{twst}}|_{\mathsf{Occ}(P)}$, and $\bar{u}, \bar{v} \in A^k$. Using Claim 7 we obtain $\bar{u}\bar{v} \in P$ if and only if $\tau_\xi(\xi(\bar{u}))\tau_\xi(\xi(\bar{v})) \in \tau_\xi(P)$. Because $\xi|_{\mathsf{Occ}(P)} = \xi_{\mathsf{twst}}|_{\mathsf{Occ}(P)}$, the action of τ_ξ and τ is equal on the i-tip components of P for $i \in \mathsf{Occ}(P)$. Thus, $\tau_\xi(P) = \tau(P)$. Similarly, $\tau_\xi(\xi(\bar{u})) = \tau(\xi(\bar{u}))$ and $\tau_\xi(\xi(\bar{v})) = \tau(\xi(\bar{v}))$. $\dashv$

For every blurrable orbit $P \in \mathbf{P}_k$, it holds that $|\mathsf{Occ}(P)| \leq k$ and so we can use the blurrer properties as follows:

Claim 23. *Let $P \in \mathbf{P}_{2k}$ and $Q = \tau(P)$. If P_1 (and so Q_1) and P_2 (and so Q_2) are blurrable, then $\chi^P \cdot S = S \cdot \chi^Q$.*

Proof. With the definition of S on blurrable orbits we obtain for $\bar{u} \in P_1$ and $\bar{v} \in Q_2$ that

$$\begin{aligned}
(\chi^P \cdot S)(\bar{u}, \bar{v}) &= \sum_{\bar{w} \in P_2} \chi^P(\bar{u}, \bar{w}) \cdot S_{P_2 \times Q_2}(\bar{w}, \bar{v}) \\
&= \sum_{\bar{w} \in P_2} \chi^P(\bar{u}, \bar{w}) \cdot \sum_{\substack{\xi \in \Xi, \\ \tau(\xi(\bar{w})) = \bar{v}}} 1 \\
&= \sum_{\xi \in \Xi} \chi^P\Big(\bar{u}, \xi^{-1}(\tau^{-1}(\bar{v}))\Big). \qquad (\star)
\end{aligned}$$

Now, for every $\xi \in \Xi$, the entry $\chi^P(\bar{u}, \xi^{-1}(\tau^{-1}(\bar{v})))$ depends only on $\xi|_{\mathsf{Occ}(P_1)}$ by Claim 21, i.e., $\xi \mapsto \chi^P(\bar{u}, \xi^{-1}(\tau^{-1}(\bar{v})))$ is actually a function $\Xi|_{\mathsf{Occ}(P_1)} \to \mathbb{F}_2$. Because P_1 is a blurrable k-orbit, it holds that $|\mathsf{Occ}(P_1)| \leq k$. Then, by Lemma 7.48, it follows that for some $\xi_{\mathsf{tw}} \in \Xi$ with $\xi_{\mathsf{tw}}|_{\mathsf{Occ}(P_1)} = \xi_{\mathsf{twst}}|_{\mathsf{Occ}(P_1)}$ we have that

$$\begin{aligned}(\star) &= \chi^P\Big(\bar{u}, \xi_{\mathsf{tw}}^{-1}(\tau^{-1}(\bar{v}))\Big)\\ &= \chi^Q\Big(\tau(\xi_{\mathsf{tw}}(\bar{u})), \bar{v}\Big)\\ &= \sum_{\xi \in \Xi} \chi^Q\Big(\tau(\xi(\bar{u})), \bar{v}\Big)\\ &= \sum_{\bar{w} \in Q_1} S_{P_1 \times Q_1}(\bar{u}, \bar{w}) \cdot \chi^Q(\bar{w}, \bar{v})\\ &= (S \cdot \chi^Q)(\bar{u}, \bar{v}),\end{aligned}$$

where the transition from P to Q is by Claim 22. The last step is the inverse reasoning as for P using Claim 21. ⊣

Next, we want to consider the case that P_1 is blurrable and P_2 is not (or vice versa, which is symmetric). We would like to use Claim 22 as in the case that both P_1 and P_2 are blurrable. But this is not sufficient because $S_{P_2 \times \tau(P_2)}$ is defined using the matrices S^ξ. We show that the matrices S^ξ cancel in this case. Intuitively, the idea is to use that the active regions of the $S^{\xi,i}$ are disjoint (Claim 13). As a consequence the matrices $S^{\xi,i}$ for all $i \notin \mathsf{Occ}(P_1)$ cancel. For the remaining $S^{\xi,i}$ we use the blurrer properties to show that they vanish.

Claim 24. *Assume $P \in \mathbf{P}_{2k}$ and $Q = \tau(P)$. If P_1 is blurrable and P_2 is not, then $\chi^P \cdot S = S \cdot \chi^Q$.*

Proof. Let $\bar{u} \in P_1$ and $\bar{v} \in Q_2$. By the definition of S we have

$$\begin{aligned}(\chi^P \cdot S)(\bar{u}, \bar{v}) &= \sum_{\bar{w} \in P_2} \chi^P(\bar{u}, \bar{w}) \cdot S_{P_2 \times Q_2}(\bar{w}, \bar{v})\\ &= \sum_{\bar{w} \in P_2} \chi^P(\bar{u}, \bar{w}) \cdot \sum_{\substack{\xi \in \Xi,\\ \xi(\bar{w}_{\mathfrak{c}}) = \tau_\xi^{-1}(\bar{v}_{\mathfrak{c}})}} S^\xi\Big(\xi(\bar{w}^{-\mathfrak{c}}), \tau_\xi^{-1}(\bar{v}^{-\mathfrak{c}})\Big)\\ &= \sum_{\xi \in \Xi} \sum_{\substack{\bar{w} \in P_2,\\ \xi(\bar{w}_{\mathfrak{c}}) = \tau_\xi^{-1}(\bar{v}_{\mathfrak{c}})}} \chi^P(\bar{u}, \bar{w}) \cdot S^\xi\Big(\xi(\bar{w}^{-\mathfrak{c}}), \tau_\xi^{-1}(\bar{v}^{-\mathfrak{c}})\Big). \qquad (\star)\end{aligned}$$

For every $\xi \in \Xi$, we set $P'_{\xi,2} := \{\bar{w}^{-\mathfrak{c}} \mid \bar{w} \in P_2, \bar{w}_{\mathfrak{c}} = \xi(\bar{v}_{\mathfrak{c}})\}$ (note that $\tau_\xi^{-1}(\bar{v}_{\mathfrak{c}}) = \bar{v}_{\mathfrak{c}}$ because $\mathsf{orig}(\bar{v}_{\mathfrak{c}}) = \mathfrak{c}$). Here $\bar{w}\xi^{-1}(\tau_\xi^{-1}(\bar{v}_{\mathfrak{c}}))$ denotes the tuple $\bar{w}'$ such that $\bar{w}'^{-\mathfrak{c}} = \bar{w}$ and $\bar{w}'_{\mathfrak{c}} = \xi^{-1}(\tau_\xi^{-1}(\bar{v}_{\mathfrak{c}}))$. It holds that $P'_{\xi,2} \in \mathsf{orb}_{k-1}((\mathfrak{A}_f, \bar{p}p_{\mathfrak{c}}))$ by Corollary 7.19. Then

$$(\star) = \sum_{\xi \in \Xi} \sum_{\bar{w} \in P'_{\xi,2}} \chi^P\Big(\bar{u}, \bar{w}\xi^{-1}(\tau_\xi^{-1}(\bar{v}_{\mathfrak{c}}))\Big) \cdot S^\xi\Big(\xi(\bar{w}), \tau_\xi^{-1}(\bar{v}^{-\mathfrak{c}})\Big).$$

Let $K \subseteq [d]$ be the maximal set of indices such that there is no component C of P_1 satisfying $C \subseteq N_i$. Hence, $\mathsf{Occ}(P_1) \subseteq [d] \setminus K$. We partition the components of P_2 as follows:

- Let D be the set of components C of P_2 that are also components of P and satisfy $C \subseteq N_i$ for some $i \in K$.
- Let E be the set of C of P_2 that are not contained in D and satisfy $C \subseteq N_i$ for some $i \in [d]$.
- Let R be the set of all remaining components of P_2.

We split $\bar{w} = \bar{w}_D\bar{w}_E\bar{w}_R$ into the components belonging to D, E, and R. We split $\bar{v} = \bar{v}_D\bar{v}_E\bar{v}_R\bar{v}_{\mathfrak{c}}$ likewise, where, for simplicity, we set $\bar{v}_D := \bar{v}_D^{\mathfrak{c}}$ and similar for $\bar{v}_E$ and $\bar{v}_R$. From Lemma 7.17 it follows that $P'_{\xi,2} = P'_{\xi,2}|_D \times P'_{\xi,2}|_{E\cup R}$ and $P = P|_D \times P|_{R'}$, where R' are the components of P not contained in D. By the definition of K and D, we have that $P|_D = P'_{\xi,2}|_D$ because the components in D are components of P, are disjoint with $\mathsf{orig}(P_1)$, and do not contain $\mathfrak{c}$ ($\mathfrak{c} \notin N_i$ for all $i \in [d]$). We obtain that

$$\begin{aligned}(\star)\\ &= \sum_{\xi\in\Xi}\sum_{\bar{w}_D\in P'_{\xi,2}|_D}\sum_{\bar{w}_E\bar{w}_R\in P'_{\xi,2}|_{E\cup R}} \chi^P\Big(\bar{u}, \bar{w}_D\bar{w}_E\bar{w}_R\xi^{-1}(\tau_\xi^{-1}(\bar{v}_{\mathfrak{c}}))\Big)\cdot S^\xi\Big(\xi(\bar{w}_D\bar{w}_E\bar{w}_R), \tau_\xi^{-1}(\bar{v}_D\bar{v}_E\bar{v}_R)\Big)\\ &= \sum_{\xi\in\Xi}\sum_{\bar{w}_E\bar{w}_R\in P'_{\xi,2}|_{E\cup R}} \chi^{P|_{R'}}\Big(\bar{u}, \bar{w}_E\bar{w}_R\xi^{-1}(\tau_\xi^{-1}(\bar{v}_{\mathfrak{c}}))\Big)\cdot \sum_{\bar{w}_D\in P'_{\xi,2}|_D} S^\xi\Big(\xi(\bar{w}_D\bar{w}_E\bar{w}_R), \tau_\xi^{-1}(\bar{v}_D\bar{v}_E\bar{v}_R)\Big).\end{aligned}$$

By Claim 4, the functions ξ and τ_ξ can be applied component-wise. For every ξ, the matrix S^ξ is not active on the components in R (by definition of R and Claim 12). Thus, $\xi(\bar{w}_R) = \tau_\xi^{-1}(\bar{v}_R)$ unless $S^\xi(\xi(\bar{w}_D\bar{w}_E\bar{w}_R), \tau_\xi^{-1}(\bar{v}_D\bar{v}_E\bar{v}_R)) = 0$. We again use Lemma 7.17 to split $P'_{\xi,2}|_{E\cup R} = P'_{\xi,2}|_E \times P'_{\xi,2}|_R$. Such a split of P is not possible because the components of $P'_{\xi,2}$ in E and R might not be components of P. Here, for every $\xi \in \Xi$, we have $\xi^{-1}(\tau_\xi^{-1}(\bar{v}_R))\xi^{-1}(\tau_\xi^{-1}(\bar{v}_{\mathfrak{c}})) = \xi^{-1}(\tau_\xi^{-1}(\bar{v}_R\bar{v}_{\mathfrak{c}}))$ because if $C \cup \{\mathfrak{c}\}$ is a component, then $C \cup \{\mathfrak{c}\}$ is a star component and C is a union of star and sky components. Thus,

$$\begin{aligned}(\star)\\ &= \sum_{\xi\in\Xi}\sum_{\bar{w}_E\in P'_{\xi,2}|_E} \chi^{P|_{R'}}\Big(\bar{u}, \bar{w}_E\xi^{-1}(\tau_\xi^{-1}(\bar{v}_R\bar{v}_{\mathfrak{c}}))\Big)\cdot \sum_{\bar{w}_D\in P'_{\xi,2}|_D} S^\xi\Big(\xi(\bar{w}_D\bar{w}_E)\tau_\xi^{-1}(\bar{v}_R), \tau_\xi^{-1}(\bar{v}_D\bar{v}_E\bar{v}_R)\Big).\end{aligned}$$

We know that $P'_{\xi,2}|_D \in \mathsf{orb}_{k'}((\mathfrak{A}_f, \bar{p}p_{\mathfrak{c}}))$ for some $k' \leq k-1$ by Lemma 7.17. Because ξ is an orbit-automorphism for every $\xi \in \Xi$ (Claim 6) and D contains no star center components, every $\xi \in \Xi$ permutes $P'_{\xi,2}|_D$. Hence, we can sum over $\xi(\bar{w}_D) \in P'_{\xi,2}|_D$ instead over $\bar{w}_D \in P'_{\xi,2}|_D$. Then we can apply Claim 15:

$$\begin{aligned}(\star) = \sum_{\xi\in\Xi}\sum_{\bar{w}_E\in P'_{\xi,2}|_E} &\chi^{P|_{R'}}\Big(\bar{u}, \bar{w}_E\xi^{-1}(\tau_\xi^{-1}(\bar{v}_R\bar{v}_{\mathfrak{c}}))\Big)\cdot\\ &\qquad\Bigg(\prod_{i\in[d]\setminus K} S^{\xi,i}\Bigg)\Big(\tau_\xi^{-1}(\bar{v}_D)\xi(\bar{w}_E)\tau_\xi^{-1}(\bar{v}_R), \tau_\xi^{-1}(\bar{v}_D\bar{v}_E\bar{v}_R)\Big).\end{aligned}$$

We show that this term only depends on $\xi|_{[d]\setminus K}$ as follows: Let $\xi, \xi' \in \Xi$ such that $\xi|_{[d]\setminus K} = \xi'|_{[d]\setminus K}$.

(a) Consider the right term $(\prod_{i\in[d]\setminus K} S^{\xi,i})(\cdot,\cdot)$ in the equation. First, $S^{\xi,i} = S^{\xi',i}$ for every $i \in [d] \setminus K$ by construction of the matrices $S^{\xi,i}$ because $\xi(i) = \xi'(i)$. Because R does not contain tip components, it holds that $\tau_\xi(\bar{v}_R) = \bar{v}_R$. Second, all components in E are contained in some N_i for $i \in [d] \setminus K$ by definition of E. That is, $\xi(\bar{w}_E) = \xi'(\bar{w}_E)$ and $\tau_\xi(\bar{v}_E) = \tau_{\xi'}(\bar{v}_E)$. Third, the active region of $\prod_{i\in[d]\setminus K} S^{\xi,i}$ is bounded by $\bigcup_{i\in[d]\setminus K} N_i$ by Claim 15. Because components in $D \cup R$ are not contained in $\bigcup_{i\in[d]\setminus K} N_i$ by definition of D and R, we can exploit Condition A2 of the active region and apply τ_ξ to $\bar{v}_D$ and $\bar{v}_R$ on both sides (because τ_ξ is a bijection), that is,

$$\begin{aligned}
&\left(\prod_{i\in[d]\setminus K} S^{\xi,i}\right)\Big(\tau_\xi^{-1}(\bar{v}_D)\xi(\bar{w}_E)\tau_\xi^{-1}(\bar{v}_R), \tau_\xi^{-1}(\bar{v}_D\bar{v}_E\bar{v}_R)\Big)\\
=&\left(\prod_{i\in[d]\setminus K} S^{\xi,i}\right)\Big(\bar{v}_D\xi(\bar{w}_E)\bar{v}_R, \bar{v}_D\tau_\xi^{-1}(\bar{v}_E)\bar{v}_R\Big)\\
=&\left(\prod_{i\in[d]\setminus K} S^{\xi,i}\right)\Big(\bar{v}_D\xi'(\bar{w}_E)\bar{v}_R, \bar{v}_D\tau_{\xi'}^{-1}(\bar{v}_E)\bar{v}_R\Big)\\
=&\left(\prod_{i\in[d]\setminus K} S^{\xi',i}\right)\Big(\tau_{\xi'}^{-1}(\bar{v}_D)\xi'(\bar{w}_E)\tau_{\xi'}^{-1}(\bar{v}_R), \tau_{\xi'}^{-1}(\bar{v}_D\bar{v}_E\bar{v}_R)\Big).
\end{aligned}$$

(b) Now consider the left term $\chi^{P|_{R'}}(\cdot,\cdot)$. First, note that R does not contain tip components and thus $\tau_\xi^{-1}(\bar{v}_R\bar{v}_\mathfrak{c}) = \bar{v}_R\bar{v}_\mathfrak{c}$. Second, $\bar{u}\bar{w}_E\xi^{-1}(\bar{v}_R\bar{v}_\mathfrak{c}) \in P|_{R'}$ if and only if $\bar{u}\bar{w}_E\xi'^{-1}(\bar{v}_R\bar{v}_\mathfrak{c}) \in P|_{R'}$ by Claim 21: By repeating entries, one sees that Claim 21 also holds for k'-orbits with $k' \le 2k$ and a partition of $[k']$ into two parts each of size at most k. Hence,

$$\chi^{P|_{R'}}\Big(\bar{u}, \bar{w}_E\xi^{-1}(\tau_\xi^{-1}(\bar{v}_R\bar{v}_\mathfrak{c}))\Big) = \chi^{P|_{R'}}\Big(\bar{u}, \bar{w}_E\xi'^{-1}(\tau_{\xi'}^{-1}(\bar{v}_R\bar{v}_\mathfrak{c}))\Big).$$

Hence, the Equation $(\star)$ is of the form $\sum_{\xi\in\Xi} h(\xi|_{[d]\setminus K})$ for some function $h\colon \Xi|_{[d]\setminus K} \to \mathbb{F}_2$. Because P_1 contains k-tuples, it holds that $|K| \ge d-k$ and hence that $|[d]\setminus K| \le k$. That is, we can apply Lemma 7.48 and obtain for some $\xi_\mathsf{tw} \in \Xi$, which satisfies that $\xi_\mathsf{tw}|_{[d]\setminus K} = \xi_\mathsf{twst}|_{[d]\setminus K}$, the following:

$$\begin{aligned}
(\star) = \sum_{\bar{w}_E\in P'_{\xi_\mathsf{tw},2}|_E} &\chi^{P|_{R'}}\Big(\bar{u}, \bar{w}_E\xi_\mathsf{tw}^{-1}(\tau_{\xi_\mathsf{tw}}^{-1}(\bar{v}_R\bar{v}_\mathfrak{c}))\Big)\cdot\\
&\left(\prod_{i\in[d]\setminus K} S^{\xi_\mathsf{tw},i}\right)\Big(\tau_{\xi_\mathsf{tw}}^{-1}(\bar{v}_D)\xi_\mathsf{tw}(\bar{w}_E)\tau_{\xi_\mathsf{tw}}^{-1}(\bar{v}_R), \tau_{\xi_\mathsf{tw}}^{-1}(\bar{v}_D\bar{v}_E\bar{v}_R)\Big).
\end{aligned}$$

Now, the matrices $S^{\xi_\mathsf{tw},i}$ in the equation blur a twist of value 0 for every $i \in [d]\setminus K$: The matrix $S^{\xi_\mathsf{tw},i}$ blurs a twist of value $(g - \xi_\mathsf{tw} - f)(\{\mathfrak{t}_i, \mathfrak{t}'_i\}) = (\xi_\mathsf{twst} - \xi_\mathsf{tw})(i)$, which is 0 for every $i \in [d]\setminus K$ because $\xi_\mathsf{tw}|_{[d]\setminus K} = \xi_\mathsf{twst}|_{[d]\setminus K}$. That is, $S^{\xi_\mathsf{tw},i} = \mathbb{1}$ by definition (cf. Claim 11) and $\xi_\mathsf{tw}(\bar{w}_E) = \tau_{\xi_\mathsf{tw}}^{-1}(\bar{v}_E)$ unless the product is zero. Hence,

$$(\star) = \chi^{P|_{R'}}\Big(\bar{u}, \xi_\mathsf{tw}^{-1}(\tau_{\xi_\mathsf{tw}}^{-1}(\bar{v}_E\bar{v}_R\bar{v}_\mathfrak{c}))\Big).$$

On the components of E, the functions τ and $\tau_{\xi_{\mathsf{tw}}}$ coincide because $\xi_{\mathsf{tw}}|_{[d]\setminus K} = \xi_{\mathsf{twst}}|_{[d]\setminus K}$. On R, both are the identity. Hence, $\tau_{\xi_{\mathsf{tw}}}^{-1}(\bar{v}_E\bar{v}_R\bar{v}_{\mathfrak{c}}) = \tau^{-1}(\bar{v}_E\bar{v}_R\bar{v}_{\mathfrak{c}})$ and

$$(\star) = \chi^{P|_{R'}}\Big(\bar{u}, \xi_{\mathsf{tw}}^{-1}(\tau^{-1}(\bar{v}_E\bar{v}_R\bar{v}_{\mathfrak{c}}))\Big).$$

We show that $\bar{u}\xi_{\mathsf{tw}}^{-1}(\tau^{-1}(\bar{v}_E\bar{v}_R\bar{v}_{\mathfrak{c}})) \in P|_{R'}$ if and only if $\bar{u}\xi_{\mathsf{tw}}^{-1}(\tau^{-1}(\bar{v}_D\bar{v}_E\bar{v}_R\bar{v}_{\mathfrak{c}})) \in P$. This holds because D is a set of components of P, $\bar{v}_D \in Q|_D$, and $\tau^{-1}(\bar{v}_D) \in P|_D$ if and only if $\bar{v}_D \in Q|_D$ by Claim 22 and $\tau^{-1}(\bar{v}_D) \in P|_D$ if and only if $\xi_{\mathsf{tw}}^{-1}(\tau^{-1}(\bar{v}_D)) \in P|_D$ by Claim 6. It follows that

$$(\star) = \chi^{P}\Big(\bar{u}, \xi_{\mathsf{tw}}^{-1}(\tau^{-1}(\bar{v}_D\bar{v}_E\bar{v}_R\bar{v}_{\mathfrak{c}}))\Big) = \chi^{P}\Big(\bar{u}, \xi_{\mathsf{tw}}^{-1}(\tau^{-1}(\bar{v}))\Big).$$

We finish the proof similar to Claim 23 because P_1 (and thus Q_1) is blurrable:

$$\begin{aligned}(\star) &= \chi^{Q}\Big(\tau(\xi_{\mathsf{tw}}(\bar{u})), \bar{v}\Big)\\ &= \sum_{\xi\in\Xi} \chi^{Q}\Big(\tau(\xi(\bar{u})), \bar{v}\Big)\\ &= \sum_{\bar{w}\in Q_1} S_{P_1\times Q_1}(\bar{u},\bar{w}) \cdot \chi^{Q}(\bar{w},\bar{v})\\ &= (S\cdot\chi^{Q})(\bar{u},\bar{v}).\end{aligned}$$ ⊣

The case when P_2 is blurrable and P_1 is not is analogous. Finally, to solve the problem in the case where both P_1 and P_2 are non-blurrable, we exploit the induction hypothesis for the matrices S^{ξ} (Claim 12). This argument becomes elaborate for two reasons. First, we need to argue that the types of occurring orbits are the same. Second, we need to treat the components containing $\mathfrak{c}$ differently (Claim 10) because in the recursive step we have $p_{\mathfrak{c}}$ as an additional parameter and thus cannot apply the orbit-automorphisms $\xi \in \Xi$ freely (because they are not the identity function on $p_{\mathfrak{c}}$). These components can be treated specially because they are not contained in the active region of the matrices S^{ξ}.

Claim 25. *Let $P \in \mathbf{P}_{2k}$ and $Q = \tau(P)$. If P_1 and P_2 are non-blurrable, then $\chi^P\cdot S = S\cdot\chi^Q$.*

Proof. Let $\bar{u} \in P_1$ and $\bar{v} \in Q_2$. We expand the definition of S:

$$\begin{aligned}(\chi^P\cdot S)(\bar{u},\bar{v}) &= \sum_{\bar{w}\in P_2} \chi^P(\bar{u},\bar{w}) \cdot S_{P_2\times Q_2}(\bar{w},\bar{v})\\ &= \sum_{\bar{w}\in P_2} \chi^P(\bar{u},\bar{w}) \cdot \sum_{\substack{\xi\in\Xi,\\ \xi(\bar{w}_{\mathfrak{c}})=\tau_{\xi}^{-1}(\bar{v}_{\mathfrak{c}})}} S^{\xi}\Big(\xi(\bar{w}^{-\mathfrak{c}}), \tau_{\xi}^{-1}(\bar{v}^{-\mathfrak{c}})\Big)\\ &= \sum_{\xi\in\Xi} \sum_{\substack{\bar{w}\in P_2,\\ \bar{w}_{\mathfrak{c}}=\xi^{-1}(\tau_{\xi}^{-1}(\bar{v}_{\mathfrak{c}}))}} \chi^P(\bar{u},\bar{w}) \cdot S^{\xi}\Big(\xi(\bar{w}^{-\mathfrak{c}}), \tau_{\xi}^{-1}(\bar{v}^{-\mathfrak{c}})\Big). \qquad (\star)\end{aligned}$$

We define, for every $\xi \in \Xi$,

$$\begin{aligned}P_{\xi} &:= \Big\{\, \bar{u}'^{-\mathfrak{c}}\bar{w}'^{-\mathfrak{c}} \;\Big|\; \bar{u}' \in P_1, \bar{w}' \in P_2, \bar{u}'\bar{w}' \in P, \bar{u}'_{\mathfrak{c}} = \bar{u}_{\mathfrak{c}}, \bar{w}'_{\mathfrak{c}} = \xi^{-1}(\tau_{\xi}^{-1}(\bar{v}_{\mathfrak{c}}))\,\Big\},\\ P_{\xi,2} &:= \Big\{\, \bar{w}'^{-\mathfrak{c}} \;\Big|\; \bar{w}' \in P_2, \bar{w}'_{\mathfrak{c}} = \xi^{-1}(\tau_{\xi}^{-1}(\bar{v}_{\mathfrak{c}}))\,\Big\}.\end{aligned}$$

By Lemma 7.18, we have that $P_\xi \in \mathsf{orb}_{2k-2}((\mathfrak{A}_f, \bar{p}p_\mathfrak{c})) \cup \{\emptyset\}$ and, by Corollary 7.19, that $P_{\xi,2} \in \mathsf{orb}_{k-1}((\mathfrak{A}_f, \bar{p}p_\mathfrak{c}))$. It depends on $\xi \in \Xi$ whether the set P_ξ is empty. We continue the equation:

$$(\star) = \sum_{\xi\in\Xi}\sum_{\bar{w}\in P_{\xi,2}} \chi^{P_\xi}(\bar{u}^{\text{-}\mathfrak{c}}, \bar{w}) \cdot S^\xi\Big(\xi(\bar{w}), \tau_\xi^{-1}(\bar{v}^{\text{-}\mathfrak{c}})\Big).$$

We use that S^ξ is invariant under automorphism of $(\mathfrak{A}_f, \bar{p})$ (Claim 14) and that ξ is an orbit-automorphism (Claim 6) for every $\xi \in \Xi$.

$$\begin{aligned}(\star) &= \sum_{\xi\in\Xi}\sum_{\bar{w}\in P_{\xi,2}} \chi^{P_\xi}(\bar{u}^{\text{-}\mathfrak{c}}, \bar{w}) \cdot S^\xi\Big(\bar{w}, \xi^{-1}(\tau_\xi^{-1}(\bar{v}^{\text{-}\mathfrak{c}}))\Big)\\ &= \sum_{\xi\in\Xi}(\chi^{P_\xi} \cdot S^\xi)\Big(\bar{u}^{\text{-}\mathfrak{c}}, \xi^{-1}(\tau_\xi^{-1}(\bar{v}^{\text{-}\mathfrak{c}}))\Big)\\ &= \sum_{\xi\in\Xi}(S^\xi \cdot \chi^{Q_\xi})\Big(\bar{u}^{\text{-}\mathfrak{c}}, \xi^{-1}(\tau_\xi^{-1}(\bar{v}^{\text{-}\mathfrak{c}}))\Big)\\ &= \sum_{\xi\in\Xi}\sum_{\bar{w}\in Q_{\xi,1}} S^\xi(\bar{u}^{\text{-}\mathfrak{c}}, \bar{w}) \cdot \chi^{Q_\xi}\Big(\bar{w}, \xi^{-1}(\tau_\xi^{-1}(\bar{v}^{\text{-}\mathfrak{c}}))\Big),\end{aligned}$$

where $Q_\xi \in \mathsf{orb}_{2k-2}((\mathfrak{A}_{g-\xi}, \bar{p}p_\mathfrak{c}))$ has the same type in $(\mathfrak{A}_{g-\xi}, \bar{p}p_\mathfrak{c})$ as P_ξ has in $(\mathfrak{A}_f, \bar{p}p_\mathfrak{c})$ and $Q_{\xi,1} := Q_\xi|_{[k-1]}$ for every $\xi \in \Xi$ (or $Q_\xi = Q_{\xi,1} = \emptyset$ if $P_\xi = \emptyset$). The step from P_ξ to Q_ξ is possible because S^ξ blurs the twist between $(\mathfrak{A}_f, \bar{p}p_\mathfrak{c})$ and $(\mathfrak{A}_{g-\xi}, \bar{p}p_\mathfrak{c})$ for every $\xi \in \Xi$ (Claim 12).

We analyze the structures of these orbits. Let R be the star center component of P and so of Q. Note that R may be split into multiple components for P_1, P_2, P_ξ, and the others. We apply Lemma 7.17 to split $P = P|_R \times P|_D$, $P_\xi = P_\xi|_R \times P_\xi|_D$, and $Q_\xi = Q_\xi|_R \times Q_\xi|_D$ for every $\xi \in \Xi$, where D is the set of components of P apart from R. The components in D have distance greater than 1 to $\mathfrak{c}$ because $\mathfrak{c} \in R$. Hence, $P|_D = P_\xi|_D$ and

$$P = P|_R \times P_\xi|_D$$

for every $\xi \in \Xi$. Because $P_\xi|_R$ has the same type in $(\mathfrak{A}_f, \bar{p}p_\mathfrak{c})$ as $Q_\xi|_R$ has in $(\mathfrak{A}_{g-\xi}, \bar{p}p_\mathfrak{c})$ and their origins do not contain any of the base vertices $\mathfrak{t}_i'$, it even follows that $P_\xi|_R = Q_\xi|_R$ for every $\xi \in \Xi$ because $(\mathfrak{A}_f, \bar{p}p_\mathfrak{c})[R] = (\mathfrak{A}_{g-\xi}, \bar{p}p_\mathfrak{c})[R]$. So

$$Q_\xi = P_\xi|_R \times Q_\xi|_D$$

for every $\xi \in \Xi$. For readability, we set $\bar{u}_D := \bar{u}_D^{\text{-}\mathfrak{c}}$ and $\bar{u}_R := \bar{u}_R^{\text{-}\mathfrak{c}}$. Then $\bar{u}^{\text{-}\mathfrak{c}} = \bar{u}_R\bar{u}_D$. We do the same for $\bar{v}^{\text{-}\mathfrak{c}} = \bar{v}_R\bar{v}_D$ and $\bar{w} = \bar{w}_R\bar{w}_D$. So we obtain in the next step that

$$(\star) = \sum_{\xi\in\Xi}\sum_{\bar{w}_R\bar{w}_D\in Q_{\xi,1}} S^\xi(\bar{u}_R\bar{u}_D, \bar{w}_R\bar{w}_D) \cdot \chi^{Q_\xi}\Big(\bar{w}_R\bar{w}_D, \xi^{-1}(\tau_\xi^{-1}(\bar{v}_R\bar{v}_D))\Big).$$

We set

$$Q_D := \tau_\xi(Q_\xi|_D)$$

for some $\xi \in \Xi$. Claim 10 states that Q_D is an orbit of $(\mathfrak{A}_g, \bar{p}p_\mathfrak{c})$ and has the same type in $(\mathfrak{A}_g, \bar{p}p_\mathfrak{c})$ as $Q_\xi|_D$ has in $(\mathfrak{A}_{g-\xi}, \bar{p}p_\mathfrak{c})$. As seen before, this is the same type as $P_\xi|_D$ has in $(\mathfrak{A}_f, \bar{p}p_\mathfrak{c})$. Because, as already seen, $P_\xi|_D = P|_D$ for every $\xi \in \Xi$, the type of Q_D is independent of ξ and Q_D is well-defined. So Q_D is also an orbit of $(\mathfrak{A}_g, \bar{p})$ and $\xi(Q_D) = Q_D$ for every $\xi \in \Xi$ because ξ is an orbit-automorphism (Claim 6). Thus, $Q_D = \xi(\tau_\xi(Q_\xi|_D))$ for every $\xi \in \Xi$. We set for every $\xi \in \Xi$

$$Q'_\xi := Q_\xi|_R \times Q_D = P_\xi|_R \times Q_D.$$

By Claim 10, for every $\xi \in \Xi$ it holds that

$$\begin{aligned}
&\chi^{Q_\xi}\left(\bar{w}_R\bar{w}_D, \xi^{-1}(\tau_\xi^{-1}(\bar{v}_R\bar{v}_D))\right)\\
&= \chi^{Q'_\xi}\left(\bar{w}_R\xi(\tau_\xi\big(\bar{w}_D)\big), \xi^{-1}\big(\tau_\xi^{-1}(\bar{v}_R)\big)\xi\big(\tau_\xi(\xi^{-1}(\tau_\xi^{-1}(\bar{v}_D)))\big)\right)\\
&= \chi^{Q'_\xi}\left(\bar{w}_R\xi(\tau_\xi(\bar{w}_D)), \xi^{-1}(\tau_\xi^{-1}(\bar{v}_R))\bar{v}_D\right).
\end{aligned}$$

We used that ξ and τ_ξ commute (Claim 5) and that they can be applied component-wise (Claim 4). With $Q'_{\xi,1} := Q'_\xi|_{[k-1]}$ for every $\xi \in \Xi$, we obtain that

$$\begin{aligned}
(\star) &= \sum_{\xi\in\Xi}\sum_{\bar{w}_R\bar{w}_D\in Q_{\xi,1}} S^\xi\left(\bar{u}_R\bar{u}_D, \bar{w}_R\bar{w}_D\right)\cdot\chi^{Q'_\xi}\left(\bar{w}_R\xi(\tau_\xi(\bar{w}_D)), \xi^{-1}(\tau_\xi^{-1}(\bar{v}_R))\bar{v}_D\right)\\
&= \sum_{\xi\in\Xi}\sum_{\bar{w}_R\bar{w}_D\in Q'_{\xi,1}} S^\xi\left(\bar{u}_R\bar{u}_D, \bar{w}_R\xi^{-1}(\tau_\xi^{-1}(\bar{w}_D))\right)\cdot\chi^{Q'_\xi}\left(\bar{w}_R\bar{w}_D, \xi^{-1}(\tau_\xi^{-1}(\bar{v}_R))\bar{v}_D\right).
\end{aligned}$$

We claim that

$$Q = P|_R \times Q_D.$$

First, $P|_R \times Q_D \in \mathsf{orb}_{2k}((\mathfrak{A}_g, \bar{p}))$ by Lemma 7.17. Both parts are orbits and their origins are not connected. Second, because Q_D has the same type in $(\mathfrak{A}_g, \bar{p}p_\mathfrak{c})$ as $P_\xi|_D = P|_D$ in $(\mathfrak{A}_f, \bar{p}p_\mathfrak{c})$, Q_D has also the same type in $(\mathfrak{A}_g, \bar{p})$ as $P_\xi|_D = P|_D$ in $(\mathfrak{A}_f, \bar{p})$. Because the components in R do not contain a twisted base edge, $P|_R = Q|_R$ and so indeed $Q = P|_R \times Q_D$. Because $Q'_\xi|_R = P_\xi|_R$ and by the definition of $P_\xi|_R$, it holds that $\bar{w}_R\bar{w}_D\xi^{-1}(\tau_\xi^{-1}(\bar{v}_R)) \in Q'_\xi$ if and only if $\bar{w}_\mathfrak{c}\bar{w}_R\bar{w}_D, \xi^{-1}(\tau_\xi^{-1}(\bar{v}_\mathfrak{c}\bar{v}_R)) \in Q$. We continue as follows:

$$\begin{aligned}
(\star) &= \sum_{\xi\in\Xi}\sum_{\bar{w}_R\bar{w}_D\in Q'_{\xi,1}} S^\xi\left(\bar{u}_R\bar{u}_D, \bar{w}_R\xi^{-1}(\tau_\xi^{-1}(\bar{w}_D))\right)\cdot\chi^{Q}\left(\bar{u}_\mathfrak{c}\bar{w}_R\bar{w}_D, \xi^{-1}(\tau_\xi^{-1}(\bar{v}_\mathfrak{c}\bar{v}_R))\bar{v}_D\right)\\
&= \sum_{\xi\in\Xi}\sum_{\substack{\bar{w}_\mathfrak{c}\bar{w}_R\bar{w}_D\in Q_1,\\ \bar{w}_\mathfrak{c}=\bar{u}_\mathfrak{c}}} S^\xi\left(\bar{u}_R\bar{u}_D, \bar{w}_R\xi^{-1}(\tau_\xi^{-1}(\bar{w}_D))\right)\cdot\chi^{Q}\left(\bar{w}_\mathfrak{c}\bar{w}_R\bar{w}_D, \xi^{-1}(\tau_\xi^{-1}(\bar{v}_\mathfrak{c}\bar{v}_R))\bar{v}_D\right).
\end{aligned}$$

For every $\xi \in \Xi$, it holds that $\xi^{-1}(\tau_\xi^{-1}(Q|_R)) = Q|_R$ because τ_ξ is the identity function on R-base vertices (R is a star center component) and because ξ is an orbit-automorphism (Claim 6). So by Lemma 7.17, $\xi^{-1}(\tau_\xi^{-1}(\bar{w}_R))\bar{w}_D \in Q$ if and only if $\bar{w}_R\bar{w}_D \in Q$ for every

$\xi \in \Xi$. We substitute $\bar{w}_R\bar{w}_D \mapsto \xi^{-1}(\tau_\xi^{-1}(\bar{w}_R))\bar{w}_D$ (this is a bijection).

$$
\begin{aligned}
&(\star)\\
&= \sum_{\xi\in\Xi} \sum_{\substack{\bar{w}_\mathfrak{c}\bar{w}_R\bar{w}_D \in Q_1,\\ \xi^{-1}(\tau_\xi^{-1}(\bar{w}_\mathfrak{c}))=\bar{u}_\mathfrak{c}}} S^\xi\Big(\bar{u}_R\bar{u}_D, \xi^{-1}\Big(\tau_\xi^{-1}(\bar{w}_R)\Big)\xi^{-1}\Big(\tau_\xi^{-1}(\bar{w}_D)\Big)\Big) \cdot \\
&\qquad\qquad\qquad\qquad \chi^Q\Big(\xi^{-1}\Big(\tau_\xi^{-1}(\bar{w}_\mathfrak{c}\bar{w}_R)\Big)\bar{w}_D, \xi^{-1}\Big(\tau_\xi^{-1}(\bar{v}_\mathfrak{c}\bar{v}_R)\Big)\bar{v}_D\Big)\\
&= \sum_{\xi\in\Xi} \sum_{\substack{\bar{w}_\mathfrak{c}\bar{w}_R\bar{w}_D \in Q_1,\\ \xi^{-1}(\tau_\xi^{-1}(\bar{w}_\mathfrak{c}))=\bar{u}_\mathfrak{c}}} S^\xi\Big(\bar{u}_R\bar{u}_D, \xi^{-1}(\tau_\xi^{-1}(\bar{w}_R\bar{w}_D))\Big) \cdot \chi^Q\Big(\xi^{-1}(\tau_\xi^{-1}(\bar{w}_\mathfrak{c}\bar{w}_R))\bar{w}_D, \xi^{-1}(\tau_\xi^{-1}(\bar{v}_\mathfrak{c}\bar{v}_R))\bar{v}_D\Big)\\
&= \sum_{\xi\in\Xi} \sum_{\substack{\bar{w}_\mathfrak{c}\bar{w}_R\bar{w}_D \in Q_1,\\ \xi^{-1}(\tau_\xi^{-1}(\bar{w}_\mathfrak{c}))=\bar{u}_\mathfrak{c}}} S^\xi\Big(\bar{u}_R\bar{u}_D, \xi^{-1}(\tau_\xi^{-1}(\bar{w}_R\bar{w}_D))\Big) \cdot \chi^Q\Big(\bar{w}_\mathfrak{c}\bar{w}_R\bar{w}_D, \bar{v}_\mathfrak{c}\bar{v}_R\bar{v}_D\Big).
\end{aligned}
$$

In the last step we again used that τ_ξ is the identity on the R-component and that ξ is an orbit-automorphism. We finish the proof with the reverse steps as to how we started it.

$$
\begin{aligned}
(\star) &= \sum_{\xi\in\Xi} \sum_{\substack{\bar{w}\in Q_1,\\ \xi^{-1}(\tau_\xi^{-1}(\bar{w}_\mathfrak{c}))=\bar{u}_\mathfrak{c}}} S^\xi\Big(\bar{u}^{-\mathfrak{c}}, \xi^{-1}(\tau_\xi^{-1}(\bar{w}^{-\mathfrak{c}}))\Big) \cdot \chi^Q\Big(\bar{w}, \bar{v}\Big)\\
&= \sum_{\bar{w}\in Q_1} \sum_{\substack{\xi\in\Xi,\\ \xi(\bar{u}_\mathfrak{c})=\tau_\xi^{-1}(\bar{w}_\mathfrak{c})}} S^\xi\Big(\bar{u}^{-\mathfrak{c}}, \xi^{-1}(\tau_\xi^{-1}(\bar{w}^{-\mathfrak{c}}))\Big) \cdot \chi^Q\Big(\bar{w}, \bar{v}\Big)\\
&= \sum_{\bar{w}\in Q_1} S(\bar{u}, \bar{w}) \cdot \chi^Q(\bar{w}, \bar{v})\\
&= (S \cdot \chi^Q)(\bar{u}, \bar{v}). \qquad \dashv
\end{aligned}
$$

Claims 23, 24, and 25 show that S indeed k-blurs the twist between $(\mathfrak{A}_f, \bar{p})$ and $(\mathfrak{A}_g, \bar{p})$. This finishes the proof of Lemma 7.55. □

We checked our construction in the proof for $k \leq 2$ on the computer. For larger k, it was computationally not tractable.

7.8 Separating Rank Logic from CPT

In this section we finally separate rank logic from CPT. To apply Lemma 7.55, we need to construct a suitable class of base graphs.

Lemma 7.61. *For every $n \in \mathbb{N}$, there is a regular graph that has degree at least n, girth at least n, and is n-connected.*

Proof. A (d, g)-cage is a d-regular graph with girth g of minimum order. For every odd $g \geq 7$, every (d, g)-cage is $\lceil \frac{d}{2} \rceil$-connected [11]. So it suffices to show that, for every n, there is a $d \geq 2n$ and an odd $g \geq n$ such that there is a (d, g)-cage. For all $d \geq 2$ and $g \geq 3$, there is a d-regular graph of girth g [107] and so in particular a (d, g)-cage. □

Lemma 7.62. *Let $G = (V, E)$ be a d-regular graph of girth at least $2(\ell + 2) + 1$ for some $\ell \in \mathbb{N}$. Then, for every set $V' \subseteq V$ of size $|V'| < d$, there is a base vertex $\mathfrak{u} \in V$ such that $\mathsf{dist}_G(V', \mathfrak{u}) > \ell$.*

Proof. Because G is d-regular and has girth at least $2(\ell + 2) + 1$, for every $\mathfrak{v} \in V$, the induced subgraph $G[N^i_G(\mathfrak{v})]$ is a tree in which the root $\mathfrak{v}$ has d many subtrees, which are all complete $(d-1)$-ary trees of height $i-1$. Thus, we have $|N^i_G(\mathfrak{v})| = 1 + d\sum_{j=0}^{i-1}(d-1)^j$ for every $\mathfrak{v} \in V$. Let $\mathfrak{v} \in V'$. Then

$$\begin{aligned}\Big|N^{\ell+2}_G(\mathfrak{v}) \setminus \bigcup_{\mathfrak{w}\in V'} N^{\ell}_G(\mathfrak{w})\Big| &\geq 1 + d\sum_{j=0}^{\ell+1}(d-1)^j - |V'| \cdot \left(1 + d\sum_{j=0}^{\ell-1}(d-1)^j\right) \\ &\geq d\sum_{j=0}^{\ell+1}(d-1)^j - (d-1) - d\sum_{j=0}^{\ell}(d-1)^j \\ &\geq d(d-1)^{\ell+1} - (d-1) > 0.\end{aligned}$$

Hence, there is at least one $\mathfrak{u} \in V$ satisfying the claim. □

Theorem 7.63. *There is a class of base graphs $\mathcal{K}$, such that, for all $k, m \in \mathbb{N}$, there are $G = (V, E, \leq) \in \mathcal{K}$ and $q \in \mathbb{N}$ such that $\mathsf{CFI}_{2^q}(G, f) \equiv^{2k+m,k,\{2\}}_{\mathcal{M}} \mathsf{CFI}_{2^q}(G, g)$ for all $f, g\colon E \to \mathbb{Z}_{2^q}$ satisfying $\sum g = \sum f + 2^{q-1}$.*

Proof. Let $\mathcal{K}$ be the class of graphs given by Lemma 7.61 for every $n \in \mathbb{N}$ equipped with some total order. Let $k, m \in \mathbb{N}$ and $G \in \mathcal{K}$ such that it has degree $d \geq d(k, m) > m$, girth at least $2(2r(k+1)+2)+1$, and G is at least $(2k+m+1)$-connected. Let $q = q(k)$, $\mathfrak{e} = \{\mathfrak{u}, \mathfrak{v}\} \in E$, and $g, f\colon E \to \mathbb{Z}_{2^q}$ such that $\sum g = \sum f + 2^{q-1}$. Up to isomorphism of the CFI structures, we can assume that $\mathfrak{e}$ is the only base edge twisted by f and g and that $g(\mathfrak{e}) = f(\mathfrak{e}) + 2^{q-1}$. Let $\mathfrak{A}_f = \mathsf{CFI}_{2^q}(G, f)$ and $\mathfrak{A}_g = \mathsf{CFI}_{2^q}(G, g)$ both with universe A. We show that Duplicator has a winning strategy in the invertible-map game $\mathcal{M}^{2k+m,k,\{2\}}$ played on $\mathfrak{A}_f$ and $\mathfrak{A}_g$.

We consider the case where m many pebble pairs are placed on the structures. Starting with fewer pebbles makes the proof more elaborate without providing any additional insights. Let $\bar{u} \in A^{2k+m}$ and $\bar{v} \in A^{2k+m}$ such that the type of $\bar{u}$ in $\mathfrak{A}_f$ is the same as of $\bar{v}$ in $\mathfrak{A}_g$ and $\mathfrak{e} \not\subseteq \mathsf{orig}(\bar{u})$, i.e., there exists a local isomorphism mapping $\bar{u}$ to $\bar{v}$. Assume we play on $(\mathfrak{A}_f, \bar{u})$ and $(\mathfrak{A}_g, \bar{v})$. Let $P \in \mathsf{orb}_{2k+m}((\mathfrak{A}_f, \bar{u}))$ contain $\bar{u}$ and $Q \in \mathsf{orb}_{2k+m}((\mathfrak{A}_g, \bar{v}))$ contain $\bar{v}$. Because $\bar{u}$ and $\bar{v}$ have the same type, P and Q have the same type. Because $\mathfrak{e} \not\subseteq \mathsf{orig}(\bar{u})$, we have that $\mathfrak{A}_f[\mathsf{orig}(P)] = \mathfrak{A}_g[\mathsf{orig}(Q)]$ and thus $P = Q$. That is, there is an automorphism $\varphi \in \mathsf{Aut}(\mathfrak{A}_g)$ such that $\varphi(\bar{v}) = \bar{u}$ (Corollary 7.11). Up to isomorphism, we can continue the game on $(\mathfrak{A}_f, \bar{u})$ and $\varphi((\mathfrak{A}_g, \bar{v})) = (\mathfrak{A}_g, \bar{u})$. Spoiler picks up $2k$ pebbles on each graph leaving us with the structures $(\mathfrak{A}_f, \bar{w})$ and $(\mathfrak{A}_g, \bar{w})$ for some $\bar{w} \in A^m$, where $\bar{w}$ has the same type in $\mathfrak{A}_f$ and in $\mathfrak{A}_g$.

There is a base vertex $\mathfrak{w} \in V$ such that $\mathsf{dist}_G(\mathsf{orig}(\bar{w}), \mathfrak{w}) > 2r(k+1)$ by Lemma 7.62. We can assume up to exchanging $\mathfrak{u}$ and $\mathfrak{v}$ that $\mathfrak{e} \cap \mathsf{orig}(\bar{w}) \subseteq \{\mathfrak{u}\}$ because $\mathfrak{e} \not\subseteq \mathsf{orig}(\bar{w})$. Because G is $(2k+m+1)$-connected, there is a path $\bar{\mathfrak{s}} = (\mathfrak{u}, \mathfrak{v}, \dots, \mathfrak{w}', \mathfrak{w})$ such that $\bar{\mathfrak{s}}$ and $\mathsf{orig}(\bar{w})$ are disjoint apart from $\mathfrak{u}$. Then we apply the path-isomorphism $\psi := \vec{\pi}[2^{q-1}, \bar{\mathfrak{s}}]$ and w.l.o.g. can continue the game on $(\mathfrak{A}_f, \bar{w})$ and $\psi((\mathfrak{A}_g, \bar{w})) = (\mathfrak{A}_h, \bar{w})$, where f and h only twist the base edge $\{\mathfrak{w}, \mathfrak{w}'\}$. Now between $\mathfrak{A}_f$ and $\mathfrak{A}_h$ the twist is moved sufficiently far away from $\mathsf{orig}(\bar{w})$. Duplicator chooses the partitions $\mathbf{P} := \mathsf{orb}_{2k}((\mathfrak{A}_f, \bar{w}))$ and

$\mathbf{Q} := \mathsf{orb}_{2k}((\mathfrak{A}_h, \bar{w}))$ of $A^k \times A^k$ and the invertible matrix S that k-blurs the twist between $(\mathfrak{A}_f, \bar{w})$ and $(\mathfrak{A}_h, \bar{w})$ given by Lemma 7.55. By construction, the conditions of the lemma are satisfied. The matrix S induces a map $\mathbf{P} \to \mathbf{Q}$ mapping $P \mapsto Q$ if and only if P and Q have the same type in $(\mathfrak{A}_f, \bar{w})$ and $(\mathfrak{A}_h, \bar{w})$, respectively (Definition 7.22). Spoiler chooses orbits $P \in \mathbf{P}$ and $Q \in \mathbf{Q}$ of the same type and $\bar{u}' \in P$ and $\bar{v}' \in Q$. Then $\bar{u}'$ has the same type in $(\mathfrak{A}, \bar{w})$ as $\bar{v}'$ in $(\mathfrak{A}_h, \bar{w})$. So $\bar{w}\bar{u}'$ and $\bar{w}\bar{v}'$ induce a local isomorphism and the next round starts. As we can see, we are in the same situation as before, that is $\bar{w}\bar{u}'$ and $\bar{w}\bar{v}'$ have the same type in $\mathfrak{A}_f$ and $\mathfrak{A}_h$, respectively, and we can apply Duplicator's strategy again. So Duplicator has a winning strategy in the $\mathcal{M}^{2k+m,k,\{2\}}$-game. □

We now show that CPT distinguishes CFI structures over all $\mathbb{Z}_{2^q}$. To do so, we review an extended result of the canonization of bounded and abelian colors (Theorem 4.88) from [103]. The canonization result can be strengthened from bounded color class size to ordered colors. A τ-structure with **ordered colors** is a tuple $(\mathfrak{A}, \mathbb{\Gamma})$, where $\mathfrak{A}$ is a colored relational τ-structure with color classes $C_1, \dots, C_n$ and $\mathbb{\Gamma} = \{(\Gamma_i, \leq_i) \mid i \in [n]\}$ is a family of ordered permutation groups such that Γ_i is a transitive group with domain C_i for every $i \in [n]$. Note that structures with ordered colors are, without further encoding, not relational structures because $\mathbb{\Gamma}$ is a higher-order object. However, we only define, given a relational τ-structure $\mathfrak{A}$, the family $\mathbb{\Gamma}$ of ordered permutation groups in CPT and thus can represent $\mathbb{\Gamma}$ as a hereditarily finite set.

Theorem 7.64 ([103]). *Canonization of τ-structures with ordered abelian colors is* CPT-*definable.*

Theorem 7.65. *There is a class of τ-structures $\mathcal{K}$ such that* IFPC+R $<$ *Ptime on $\mathcal{K}$ and* CPT $=$ *Ptime on $\mathcal{K}$.*

Proof. Let $\mathcal{K}'$ be the graph class from Theorem 7.63, and set $\mathcal{K} := \mathsf{CFI}_{2^\omega}(\mathcal{K}')$ (recall Definition 7.20). We want to show that the CFI query for $\mathcal{K}$ is not IFPC+R-definable. This is the task of deciding whether for a given $\mathsf{CFI}_{2^q}(G, f) \in \mathcal{K}$ it holds that $\sum f = 0$. By Lemma 7.21, it suffices to show that the CFI query is not IFPC+R$_{\{2\}}$-definable. The claim follows from Lemma 7.1 and Theorem 7.63.

We now argue that CPT captures Ptime on $\mathcal{K}$. By Theorem 7.64, it suffices to show that $\mathcal{K}$ is a class of structures with abelian and ordered colors (cf. Section 7.1). By Lemma 7.3, the colors are abelian and it remains to define for every color class an ordered and transitive permutation group in CPT. Every gadget forms a color class. As seen in Section 7.3, every gadget forms a 1-orbit and has a regular, so in particular transitive, automorphism group (Lemmas 7.10 and 7.15). Consider a degree d gadget. Then every automorphism φ of the gadget can be identified with a tuple $\bar{a} \in \mathbb{Z}_{2^q}^d$ satisfying $\sum \bar{a} = 0$ such that $\varphi(u) = v$ if and only if $\bar{a}(u) = v$. Indeed, every such tuple represents an automorphism. Hence, the automorphism group of the gadget can be represented by $\{\bar{a} \in \mathbb{Z}_{2^q}^d \mid \sum \bar{a} = 0\}$, which is a CPT-definable set (the tuples, although of unbounded length, can be represented by standard set encoding of tuples). This set can be ordered using the lexicographical order on tuples. Because the base graph is ordered, the automorphism corresponding to the tuple $\bar{a}$ is CPT-definable: The i-th entry defines the action on the i-th outgoing base edge, which is definable using the relation R_I. □

Corollary 7.66. IFPC+R $<$ *Ptime.*

7.9 Linear-Algebraic Logic

In this section, we review the connection between the invertible-map game and **linear-algebraic logic** (LA) [25]. This logic extends infinitary first-order logic by Lindström quantifiers for every linear-algebraic operator over finite fields and as such captures all extensions of IFPC by some linear-algebraic operators over finite fields. We do not give a formal definition of linear-algebraic logic here and refer for details to the work of Dawar, Grädel, and Pakusa [25,26]. Intuitively, a **linear-algebraic operator** over a finite prime field $\mathbb{F}_p$ is a function f that maps tuples $(M_1, \dots, M_m)$ of $\mathbb{F}_p$-linear transformations on an abstract vector space $\mathbb{F}_p^B$ (for some abstract basis B) to some linear-algebraic information $f(M_1, \dots, M_m) \in \mathbb{N}$. To define "linear-algebraic information", it is required that f is invariant under $\mathbb{F}_p$-vector space isomorphisms. That is, $f(M_1, \dots, M_m) = f(N_1, \dots, N_m)$ whenever $M_1, \dots, M_m$ and $N_1, \dots N_m$ are simultaneously similar. To recall, this means that there is an $\mathbb{F}_p$-vector space isomorphism S such that, for all $i \in [m]$, we have $M_i = S \cdot N_i \cdot S^{-1}$ or equivalently $M_i \cdot S = S \cdot N_i$. Here the connection to the invertible-map game becomes apparent. This is the only requirement for f to be a linear-algebraic operator, in particular, f is not required to be computable at all. For every linear-algebraic operator f, we add a family of Lindström quantifiers to the logic. For all $m, t \in \mathbb{N}$, we add the quantifier $\mathcal{Q}_f^{m,t}$: If $\bar{\Theta}$ is an m-tuple of LA-interpretations each defining a linear-transformation, i.e., a binary structure encoding a matrix, then

$$\mathcal{Q}_f^{m,t}.\, \bar{\Theta}$$

is an LA-formula. The formula is satisfied if $\bar{\Theta}$ defines the matrices $M_1, \dots, M_m$ and $f(M_1, \dots, M_m) \geq t$. We denote, for $k \in \mathbb{N}$ and a set of primes Q, by $\mathrm{LA}^k(Q)$ the k-variable fragment of LA that only considers linear-algebraic transformations over finite fields $\mathbb{F}_p$ with $p \in Q$. In particular, all the interpretations in the Lindström quantifiers are at most k-dimensional.

The equivalence relation induced by the invertible-map game precisely corresponds to the one induced by linear-algebraic logic [25] (and hence we do not need a fully formal definition of LA). For two τ-structures $\mathfrak{A}$ and $\mathfrak{B}$, we write $\mathfrak{A} \equiv_{\mathcal{M}}^{k,Q} \mathfrak{B}$ if $\mathfrak{A} \equiv_{\mathcal{M}}^{m,k,Q} \mathfrak{B}$ for every $2m \leq k$.

Theorem 7.67 ([25, Theorem 2]). *For every $k \geq 2$, every set of primes Q, every finite relational structure $\mathfrak{A}$, and every $\bar{u}, \bar{v} \in A^k$, the following are equivalent:*

1. $(\mathfrak{A}, \bar{u}) \equiv_{\mathcal{M}}^{k,Q} (\mathfrak{A}, \bar{v})$.
2. $(\mathfrak{A}, \bar{u}) \equiv_{\mathrm{LA}^k(Q)} (\mathfrak{A}, \bar{v})$.

The theorem easily generalizes to consider two structures $\mathfrak{A}$ and $\mathfrak{B}$ and tuples $\bar{u} \in A^k$ and $\bar{v} \in B^k$. Let $\mathcal{K}$ be a class of ordered base graphs and recall the following steps to separate IFPC+R from PTIME:

1. Every IFPC+R-formula is equivalent to some IFPC+R$_{\{2\}}$-formula over $\mathsf{CFI}_{2^\omega}(\mathcal{K})$ by Lemma 7.21.
2. If the invertible-map game $\mathcal{M}^{m,k,\{2\}}$ does not distinguish all non-isomorphic CFI structures of $\mathsf{CFI}_{2^\omega}(\mathcal{K})$ for some fixed m and k, then IFPC+R$_{\{2\}}$ does not distinguish all non-isomorphic CFI structures of $\mathsf{CFI}_{2^\omega}(\mathcal{K})$ by Lemma 7.1.

3. The invertible-map game $\mathcal{M}^{m,k,\{2\}}$ does not distinguish all non-isomorphic CFI structures by Theorem 7.63 for all k and $2m \leq k$.

To lift this result from rank logic to linear-algebraic logic, we need a similar statement to Lemma 7.21 for linear-algebraic logic that states that over $\mathsf{CFI}_{2^\omega}(\mathcal{K})$, linear-algebraic logic reduces to its characteristic 2 fragment, which only considers linear-algebraic operators over $\mathbb{F}_2$. Such a statement was essentially shown by Dawar, Grädel, and Pakusa [25]. Using generalizations of CFI graphs over prime fields over $\mathbb{F}_p$ similar to our generalization to rings $\mathbb{Z}_{2^i}$, the authors showed that for every k and every set of primes Q with $p \notin Q$, the logic $\mathrm{LA}^k(Q)$ does not distinguish all CFI structures over $\mathbb{F}_p$. In particular, the authors show that $\mathrm{LA}^k(Q)$-formulas are equivalent to IFPC-formulas for a class of CFI structures, whose base graphs satisfy certain conditions. The arguments also translate to our scenario of CFI structures over rings $\mathbb{Z}_{2^i}$ as shown in joint work with Dawar and Grädel [28].

Lemma 7.68 ([28, Lemma 9]). *For every $k \in \mathbb{N}$, there exists a $c \in \mathbb{N}$ such that for every $q \geq 1$, every c-connected base graph $G = (V, E, \leq)$, and every $f, g\colon E \to \mathbb{Z}_{2^q}$, it holds that $\mathsf{CFI}_{2^q}(G, f) \equiv_{\mathcal{M}}^{k,\mathbb{P}\setminus\{2\}} \mathsf{CFI}_{2^q}(G, g)$.*

Again, $\mathbb{P}$ denotes the set of all prime numbers. Transferring the results of [25] to the CFI structures of this chapter has some technical difficulties. This chapter uses a variant only with gadget vertices for base graphs with high connectivity but [25] uses the relational variant only using edge vertices on 3-regular base graphs. These graphs are essentially expander graphs and ensure that the k-orbits of the CFI structures are definable in ℓk-variable counting logic for a fixed ℓ. This property is known as homogeneity. Instead of showing that full linear-algebraic logic reduces to its characteristic 2 fragment on our CFI structures, the following lemma allows for combining indistinguishability results for different primes.

Lemma 7.69 ([28, Lemma 10]). *Let P, Q be two sets of primes, $k, q \in \mathbb{N}$, $G = (V, E, \leq)$ be a $(k+3)$-connected base graph, and $f, g\colon E \to \mathbb{F}_{2^q}$. If $\mathsf{CFI}_{2^q}(G, f) \equiv_{\mathcal{M}}^{k+3,P} \mathsf{CFI}_{2^q}(G, g)$ and $\mathsf{CFI}_{2^q}(G, f) \equiv_{\mathcal{M}}^{k+3,Q} \mathsf{CFI}_{2^q}(G, g)$, then $\mathsf{CFI}_{2^q}(G, f) \equiv_{\mathcal{M}}^{k,P\cup Q} \mathsf{CFI}_{2^q}(G, g)$.*

Combining Theorem 7.63 with Lemma 7.69 yields the desired result.

Theorem 7.70 ([28, Theorem 7]). *For every $k \in \mathbb{N}$, there is a base graph $G = (V, E, \leq)$, a $q \in \mathbb{N}$, and $f, g\colon E \to \mathbb{Z}_{2^q}$ with $\sum f = \sum g + 2^{q-1}$ such that $\mathsf{CFI}_{2^q}(G, f) \equiv_{\mathcal{M}}^{k,\mathbb{P}} \mathsf{CFI}_{2^q}(G, g)$.*

This theorem has the following immediate consequences. The first one is that the equivalence induced by the k-ary invertible-map game does not yield an isomorphism test on finite structures for a fixed k.

Corollary 7.71. *There is no fixed $k \in \mathbb{N}$ such that $\equiv_{\mathcal{M}}^{k,\mathbb{P}}$ coincides with isomorphism on finite structures.*

The second consequence is that no extension of fixed-point logic by linear-algebraic operators captures PTIME. Every such extension can be embedded in linear-algebraic logic, which fails to define the isomorphism problem of the CFI structures considered. But this query is CPT-definable and thus in particular polynomial-time decidable.

Corollary 7.72. *No extension of fixed-point logic* IFP *(or of* IFPC*) by linear-algebraic operators over finite fields captures* CPT *or* PTIME.

7.10 Discussion

We showed that rank logic does not capture CPT and in particular not PTIME on the class of CFI structures over rings $\mathbb{Z}_{2^i}$, even if the base graph is totally ordered. To do so, we used combinatorial objects called blurrers and a recursive approach over the arity. The non-locality of k-tuples for $k > 1$ increased the difficulty of k-ary rank operators dramatically compared to the arity 1 case. It was suggested by Grädel and Pakusa [45] that CFI graphs over $\mathbb{Z}_4$ could be a separating example for rank logic and PTIME. Our concepts for blurrers required rings $\mathbb{Z}_{2^i}$ with $i > 2$ for higher arities. Actually, our computer experiments used to check Lemma 7.55 indicate that CFI graphs over $\mathbb{Z}_4$ may be distinguishable in the k-ary invertible-map game for some $k > 1$. It might also be possible that the CFI query over $\mathbb{Z}_4$ is definable in rank logic using rank operators of higher arities, but this remains an open question.

There are various definitions of rank logic, which slightly differ in the way the matrices in the rank operator are defined. In particular, there is an extension, in which rank operators not only bind universe variables, but also numeric variables [45,83,103]. It is not clear whether this extension is more expressive or not. However, for a suitable adaptation of the invertible-map game, which also supports numeric variables to construct matrices, we strongly believe that our arguments work exactly the same. In fact, we think that at least in the invertible-map game numeric variables do not increase the expressiveness. Thus, our arguments directly apply and also cover the versions of rank logic in which numeric variables can be used to define matrices.

A natural question is how rank logic can be extended such that it can define the CFI query. We have shown that it is not sufficient to compute ranks over finite fields. Even more, our construction applies to arbitrary linear-algebraic operators over finite fields. However, it is not clear how rank logic can be extended to rings $\mathbb{Z}_i$. Over rings, there are several non-equivalent notions of the rank of a matrix. For a discussion see [24, 103]. As opposed to rank logic, solvability logic can easily be extended to rings and thus should be able to define the CFI query over all $\mathbb{Z}_i$. Notably, such an extension would also capture PTIME on structures with bounded and abelian colors [103].

Regarding linear-algebraic logic and the invertible-map game, we note that the CFI structures used to prove Theorem 7.70 are super-exponential in the arity k. In particular, we get only a weak lower bound on k compared to the size of non-isomorphic structures that cannot be distinguished. The bound is super-constant but sub-logarithmic. This should be contrasted with the linear lower bound for the dimension of the Weisfeiler-Leman algorithm respective the number variable of counting logic needed to distinguish CFI graphs [20]. It is an interesting question whether the bound for the invertible-map equivalence can be strengthened.

Chapter 8

Conclusion

In this thesis, we investigated the frontier in the search for a logic capturing PTIME. In fact, we examined all current candidates for logics capturing PTIME. The two most promising ones are Choiceless Polynomial Time (CPT) and rank logic. CPT expresses all choice-less polynomial-time computations on relational structures. Rank logic extends fixed-point logic by an operator for ranks over finite fields. A third approach is the extension of fixed-point logic by witnessed symmetric choice, a restricted choice-mechanism that allows for choices from definable orbits. We studied the expressive power of both, CPT and witnessed symmetric choice, and considered the combination of these two different approaches. Moreover, we ruled out rank logic as a candidate for capturing PTIME.

In Chapter 4, we investigated the expressive power of CPT and provided the first canonization result for structures with bounded color class size and non-abelian color classes, namely for structures with dihedral colors. We introduced a normal form in which 2-injective 3-factor subdirect products play a crucial role and characterized these products for dihedral groups. Using this approach, we showed how more group theory can be exploited in logics. The normal form may be of more general interest and it is an interesting open problem whether our method can be extended to other classes of groups. The question whether CPT canonizes all structures with bounded color class size, and thereby captures PTIME on these structures, remains open.

The general approach to canonize structures of bounded color class size in CPT is that there are only polynomially many "small" sets, e.g., orders of the atoms of one color class, that cannot be distinguished. Hence, all such sets can be considered without violating the polynomial bound. In Chapter 5, we took an orthogonal point of view and considered the extension of CPT with witnessed symmetric choice, which allows for choices from definable orbits. Choosing from orbits ensures isomorphism-invariance of the logic. By making choices iteratively, one can choose one object out of exponentially many. This is not possible in CPT and considering all of these objects is not possible, either.

Using witnessed symmetric choice, a definable complete invariant can be turned into a definable canonization in CPT+WSC. Furthermore, by generalizing the DeepWL computation model to feature witnessed symmetric choice, we proved that a CPT+WSC-definable isomorphism test implies a CPT+WSC-definable complete invariant and thus a CPT+WSC-canonization. This is a surprising result and originates from the combination of the – at first sight incompatible – concepts of symmetry-invariant computation in CPT and the arbitrary choices of witnessed symmetric choice. In particular, CPT+WSC-captures PTIME for every class of structures for which CPT+WSC defines the isomorphism problem by the Immerman-Vardi Theorem. Thus, we reduced capturing

Ptime with CPT+WSC to defining isomorphism in CPT+WSC. For a class of structures with a known polynomial-time isomorphism testing algorithm, capturing Ptime with CPT+WSC becomes actually equivalent to defining isomorphism in CPT+WSC.

This raises the intriguing question whether CPT+WSC is strictly more expressive than CPT. A positive answer to this question would, in particular, separate CPT from Ptime. We at least constructed a class of base graphs, for which CPT+WSC easily defines the CFI query, but for which it is not known whether CPT defines it. Notably, the class of hypercubes is currently investigated as a class of base graphs, for which CPT might fail to define the CFI query [102]. Hypercubes are graphs of low degree but with many symmetries. This means that the CFI gadgets are rather small, but because all CPT-definable sets are closed under automorphisms, definable sets become large quickly. However, because hypercubes are graphs with a well-understood structure, it should be possible to distinguish the orbits of hypercubes in CPT. Essentially, this suffices to show that CPT+WSC defines the CFI query over hypercubes. So does CPT+WSC capture Ptime if CPT does not? In some sense, this would be surprising because CPT+WSC needs to define orbits to exploit symmetric choice but, at least algorithmically, computing the orbit-partition is polynomial-time equivalent to the graph isomorphism problem, for which the complexity status is unknown.

Because we do not know whether CPT captures Ptime, investigating the power of witnessed symmetric choice in CPT is difficult. Therefore, we considered inflationary fixed-point logic with counting (IFPC) as a base logic for studying witnessed symmetric choice. IFPC has the advantage that most of the known separation results for IFP and symmetric choice do not apply because they are based on counting. However, understanding the power of symmetric choice is not about simulating counting, which can be achieved much more naturally by counting operators.

In Chapter 6, we considered the extension of IFPC by witnessed symmetric choice (IFPC+WSC) and the further extension by the interpretation operator (IFPC+WSC+I), which allows us to evaluate a subformula in the image of an interpretation. This image may have different symmetries, which can be exploited by witnessed symmetric choice. Using the CFI construction and variations of it, we separated IFPC+WSC from IFPC+WSC+I (and thus in particular from Ptime). Additionally, we provided a graph construction based on iteratively applying the CFI construction. We used this construction to show that two nested WSC-fixed-point and interpretation operators are strictly more expressive than a single WSC-fixed-point iterator and a single interpretation operator. It is conceivable that the construction can be used to show a general operator nesting depth hierarchy for IFPC+WSC+I. If this can be done, then IFPC+WSC+I is separated from Ptime. So we also provide a new approach to separate IFPC+WSC+I from Ptime and thereby to show that, at least for IFPC, symmetric choice does not provide a logic capturing Ptime.

But to which extent can these results be transferred to CPT? Can this graph construction be adapted for CPT? As already mentioned, separating CPT from CPT+WSC implies separating CPT from Ptime. If it turns out that CPT fails to capture Ptime, can we then separate CPT+WSC from CPT or separate CPT+WSC from Ptime? More specifically, assume CPT fails to define the CFI query for a class of base graphs, say for example for hypercubes. Can our results for IFPC be adapted to show that under this assumption CPT+WSC fails to define the CFI query, too? Or how does CPT+WSC relate to the further extension of CPT+WSC with the interpretation operator?

Lastly, we considered the approach to capture Ptime by extending IFPC with operators from linear algebra. We showed in Chapter 7 that rank logic, the extension of IFPC by the uniform rank operator over finite fields, fails to capture Ptime. We proved that rank logic does not define the CFI query over the rings $\mathbb{Z}_{2^i}$. To do so, we introduced combinatorial objects called blurrers and combined them with a recursive approach over the arity of rank operators. With this technique, we could not only show that rank logic fails to define the CFI query over $\mathbb{Z}_{2^i}$, but we also showed that the characteristic 2 invertible-map game fails to distinguish these CFI structures. The invertible-map game has a potentially much stronger distinguishing power than rank logic and corresponds to the so-called linear-algebraic logic. This logic extends infinitary first-order logic by all linear-algebraic operators over finite fields. We were able to show that not only does the characteristic 2 invertible-map game fail to distinguish CFI structures over $\mathbb{Z}_{2^i}$, but also that the general invertible-map game and thus linear-algebraic logic fail to do so. Because linear-algebraic logic subsumes every extension of fixed-point logic by linear-algebraic operators over finite fields, we finally showed that using these operators it is impossible to capture Ptime. This of course poses the question of extensions by operators over rings. Extensions using quantifiers for the solvability of linear equation systems over finite rings define the CFI query over $\mathbb{Z}_{2^i}$. Currently, there is no known approach suitable for separating such a logic from Ptime. The CFI construction is inherently limited to abelian groups, but it seems that a non-abelian construction is needed to separate extensions with algebraic operators for finite rings from Ptime.

We also addressed the relationship between rank logic and CPT. Because CPT is able to define the CFI query over $\mathbb{Z}_{2^i}$ for the class of base graphs used to separate rank logic from Ptime, we showed that rank logic cannot be as expressive as CPT (see Figure 1.1). What remains open is the precise relation between these different approaches using linear-algebra and symmetry-invariant computations on hereditarily finite sets. Is CPT strictly more expressive than rank logic or are these two logics incomparable? Equivalently, is there a query definable in rank logic but not in CPT? The already mentioned CFI query over hypergraphs is a candidate for such a query because rank logic defines the CFI query (over $\mathbb{F}_2$) for all base graphs. Notably, it should be straightforward to prove that IFPC+WSC+I defines the CFI query over $\mathbb{Z}_{2^i}$ for ordered base graphs and also for hypercubes as base graphs. That is, IFPC+WSC+I defines the two queries, the one separating rank logic from CPT and the one possibly separating CPT from rank logic. However, our results for iteratively applying the CFI construction indicate that IFPC+WSC+I fails to define the CFI query for all base graphs. If that turns out to the case, then rank logic and IFPC+WSC+I are incomparable, too. This leaves us with the question whether there is candidate for a query separating CPT and IFPC+WSC+I.

If all the mentioned separations can be proved, then all the logics CPT, rank logic, and IFPC+WSC+I would be ruled out as candidates. What would this mean for the quest for a logic capturing Ptime? On the one hand, we can consider extensions circumventing the separating queries, for example, CPT+WSC or even CPT+WSC+I and the extension of IFPC by linear-algebraic operators for finite rings. On the other hand, we may need to think of completely new approaches. However, it is not clear at all what such approaches should be. In any case, the search for a logic capturing Ptime will go on and, hopefully, lead to interesting research results.

Bibliography

[1] Serge Abiteboul and Victor Vianu. Datalog extensions for database queries and updates. *J. Comput. Syst. Sci.*, 43(1):62–124, 1991. doi:10.1016/0022-0000(91)90032-Z.

[2] A. Adrian Albert. *Modern higher algebra.* Chicago University Press, Chicago, Ill., 1937.

[3] Markus Anders and Pascal Schweitzer. Parallel computation of combinatorial symmetries. In *29th Annual European Symposium on Algorithms, ESA 2021, September 6-8, 2021, Lisbon, Portugal (Virtual Conference)*, volume 204 of *LIPIcs*, pages 6:1–6:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ESA.2021.6.

[4] Markus Anders and Pascal Schweitzer. Search problems in trees with symmetries: Near optimal traversal strategies for individualization-refinement algorithms. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, pages 16:1–16:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ICALP.2021.16.

[5] Vikraman Arvind and Somenath Biswas. Expressibility of first order logic with a nondeterministic inductive operator. In *4th Annual Symposium on Theoretical Aspects of Computer Science, STACS 1987, Passau, Germany, February 19-21, 1987*, volume 247 of *Lecture Notes in Computer Science*, pages 323–335. Springer, 1987. doi:10.1007/BFb0039616.

[6] Albert Atserias, Andrei A. Bulatov, and Anuj Dawar. Affine systems of equations and counting infinitary logic. *Theor. Comput. Sci.*, 410(18):1666–1683, 2009. doi:10.1016/j.tcs.2008.12.049.

[7] Albert Atserias and Elitza N. Maneva. Sherali-Adams relaxations and indistinguishability in counting logics. *SIAM J. Comput.*, 42(1):112–137, 2013. doi:10.1137/120867834.

[8] László Babai. Monte carlo algorithms in graph isomorphism testing. Technical Report D.M.S. No. 79-10, Université de Montréal, 1979.

[9] László Babai. Graph isomorphism in quasipolynomial time [extended abstract]. In *48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 684–697. ACM, 2016. doi:10.1145/2897518.2897542.

[10] László Babai and Eugene M. Luks. Canonical labeling of graphs. In *15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA*, pages 171–183. ACM, 1983. doi:10.1145/800061.808746.

[11] Camino Balbuena and Julián Salas. A new bound for the connectivity of cages. *Appl. Math. Lett.*, 25(11):1676–1680, 2012. doi:10.1016/j.aml.2012.01.036.

[12] Christoph Berkholz. The propagation depth of local consistency. In *Principles and Practice of Constraint Programming - 20th International Conference, CP 2014, Lyon, France, September 8-12, 2014. Proceedings*, pages 158–173, 2014. doi:10.1007/978-3-319-10428-7_14.

[13] Christoph Berkholz and Martin Grohe. Linear diophantine equations, group CSPs, and graph isomorphism. In *28th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 327–339. SIAM, 2017. doi:10.1137/1.9781611974782.21.

[14] Christoph Berkholz and Jakob Nordström. Near-optimal lower bounds on quantifier depth and Weisfeiler-Leman refinement steps. In *31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2016, New York, NY, USA, July 5-8, 2016*, pages 267–276, 2016. doi:10.1145/2933575.2934560.

[15] Samuil D. Berman. On the theory of group codes. *Kibernetika*, 3(1):31–39, 1967. doi:10.1007/BF01072842.

[16] Andreas Blass and Yuri Gurevich. The logic of choice. *J. Symb. Log.*, 65(3):1264–1310, 2000. doi:10.2307/2586700.

[17] 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". 12 2005. accessed 05 Dec 2022. URL: https://www.microsoft.com/en-us/research/publication/150a-quick-update-open-problems-blass-gurevich-shelahs-article-polynomial-time-computations-unordered-structures/.

[18] Andreas Blass, Yuri Gurevich, and Saharon Shelah. Choiceless Polynomial Time. *Ann. Pure Appl. Log.*, 100(1-3):141–187, 1999. doi:10.1016/S0168-0072(99)00005-6.

[19] Andreas Blass, Yuri Gurevich, and Saharon Shelah. On polynomial time computation over unordered structures. *J. Symb. Log.*, 67(3):1093–1125, 2002. doi:10.2178/jsl/1190150152.

[20] Jin-yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992. doi:10.1007/BF01305232.

[21] Peter J. Cameron. *Combinatorics: Topics, Techniques, Algorithms.* Cambridge University Press, Cambridge, 1994. doi:10.1017/CBO9780511803888.

[22] Ashok K. Chandra and David Harel. Structure and complexity of relational queries. *J. Comput. Syst. Sci.*, 25(1):99–128, 1982. doi:10.1016/0022-0000(82)90012-5.

[23] Gang Chen and Ilia Ponomarenko. *Lectures on coherent configurations*. Central China Normal University Press, Wuhan, 2019. A draft is available at www.pdmi.ras.ru/~inp/ccNOTES.pdf.

[24] Anuj Dawar, Erich Grädel, Bjarki Holm, Eryk Kopczynski, and Wied Pakusa. Definability of linear equation systems over groups and rings. *Log. Methods Comput. Sci.*, 9(4), 2013. doi:10.2168/LMCS-9(4:12)2013.

[25] 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, July 9-12, 2019, Patras, Greece*, pages 112:1–112:14, 2019. doi:10.4230/LIPIcs.ICALP.2019.112.

[26] Anuj Dawar, Erich Grädel, and Wied Pakusa. Approximations of isomorphism and logics with linear-algebraic operators. *arXiv preprint*, abs/1902.06648, 2019. arXiv:1902.06648.

[27] Anuj Dawar, Martin Grohe, Bjarki Holm, and Bastian Laubner. Logics with rank operators. In *24th Annual IEEE Symposium on Logic in Computer Science, LICS 2009, 11-14 August 2009, Los Angeles, CA, USA*, pages 113–122. IEEE Computer Society, 2009. doi:10.1109/LICS.2009.24.

[28] Anuj Dawar, Erich Grädel, and Moritz Lichter. Limitations of the invertible-map equivalences. *J. Log. Comput.*, 09 2022. doi:10.1093/logcom/exac058.

[29] Anuj Dawar and Bjarki Holm. Pebble games with algebraic rules. In *39th International Colloquium on Automata, Languages, and Programming, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part II*, volume 7392 of *Lecture Notes in Computer Science*, pages 251–262. Springer, 2012. doi:10.1007/978-3-642-31585-5_25.

[30] Anuj Dawar and Kashif Khan. Constructing hard examples for graph isomorphism. *J. Graph Algorithms Appl.*, 23(2):293–316, 2019. doi:10.7155/jgaa.00492.

[31] Anuj Dawar and David Richerby. A fixed-point logic with symmetric choice. In *Computer Science Logic, 17th International Workshop, CSL 2003, 12th Annual Conference of the EACSL, and 8th Kurt Gödel Colloquium, KGC 2003, Vienna, Austria, August 25-30, 2003, Proceedings*, volume 2803 of *Lecture Notes in Computer Science*, pages 169–182. Springer, 2003. doi:10.1007/978-3-540-45220-1_16.

[32] Anuj Dawar and David Richerby. Fixed-point logics with nondeterministic choice. *J. Log. Comput.*, 13(4):503–530, 2003. doi:10.1093/logcom/13.4.503.

[33] Anuj Dawar and David Richerby. The power of counting logics on restricted classes of finite structures. In *Computer Science Logic, 21st International Workshop, CSL 2007, 16th Annual Conference of the EACSL, Lausanne, Switzerland, September*

11-15, 2007, Proceedings, volume 4646 of *Lecture Notes in Computer Science*, pages 84–98. Springer, 2007. doi:10.1007/978-3-540-74915-8_10.

[34] Anuj Dawar, David Richerby, and Benjamin Rossman. Choiceless Polynomial Time, counting and the Cai-Fürer-Immerman graphs. *Ann. Pure Appl. Log.*, 152(1-3):31–50, 2008. doi:10.1016/j.apal.2007.11.011.

[35] Holger Dell, Martin Grohe, and Gaurav Rattan. Lovász meets Weisfeiler and Leman. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 40:1–40:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.40.

[36] Jan Van den Bussche and Dirk Van Gucht. Semi-determinism. In *11th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 2-4, 1992, San Diego, California, USA*, pages 191–201. ACM, 1992. doi:10.1145/137097.137866.

[37] Heinz-Dieter Ebbinghaus. Extended logics: the general framework. In *Model-Theoretic Logics*, Perspectives in Mathematical Logic, pages 25–76. Association for Symbolic Logic, 1985.

[38] Heinz-Dieter Ebbinghaus and Jörg Flum. *Finite model theory.* Perspectives in Mathematical Logic. Springer, Berlin, Heidelberg, 1995. doi:10.1007/978-3-662-03182-7.

[39] Ronald Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In *Complexity of Computation, SIAM-AMS Proceedings*, pages 43–73. SIAM–AMS Proc., Vol. VII, 1974.

[40] Martin Fürer. Weisfeiler-Lehman refinement requires at least a linear number of iterations. In *28th International Colloquium on Automata, Languages, and Programming, ICALP 2001, Crete, Greece, July 8-12, 2001, Proceedings*, volume 2076 of *Lecture Notes in Computer Science*, pages 322–333. Springer, 2001. doi:10.1007/3-540-48224-5_27.

[41] Merrick L. Furst, John E. Hopcroft, and Eugene M. Luks. Polynomial-time algorithms for permutation groups. In *21st Annual Symposium on Foundations of Computer Science, Syracuse, New York, USA, 13-15 October 1980*, pages 36–41. IEEE Computer Society, 1980. doi:10.1109/SFCS.1980.34.

[42] Françoise Gire and H. Khanh Hoang. An extension of fixpoint logic with a symmetry-based choice construct. *Inf. Comput.*, 144(1):40–65, 1998. doi:10.1006/inco.1998.2712.

[43] Erich Grädel and Martin Grohe. Is polynomial time choiceless? In *Fields of Logic and Computation II - Essays Dedicated to Yuri Gurevich on the Occasion of His 75th Birthday*, volume 9300 of *Lecture Notes in Computer Science*, pages 193–209. Springer, 2015. doi:10.1007/978-3-319-23534-9_11.

[44] Erich Grädel, Phokion G. Kolaitis, Leonid Libkin, Maarten Marx, Joel Spencer, Moshe Y. Vardi, Yde Venema, and Scott Weinstein. *Finite Model Theory and Its Applications*. Texts in Theoretical Computer Science. An EATCS Series. Springer, Berlin, Heidelberg, 2007. doi:10.1007/3-540-68804-8.

[45] Erich Grädel and Wied Pakusa. Rank logic is dead, long live rank logic! *J. Symb. Log.*, 84(1):54–87, 2019. doi:10.1017/jsl.2018.33.

[46] Erich Grädel, Wied Pakusa, Svenja Schalthöfer, and Lukasz Kaiser. Characterising Choiceless Polynomial Time with first-order interpretations. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015*, pages 677–688. IEEE Computer Society, 2015. doi:10.1109/LICS.2015.68.

[47] Erich Grädel and Svenja Schalthöfer. Choiceless Logarithmic Space. In *44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019, August 26-30, 2019, Aachen, Germany*, volume 138 of *LIPIcs*, pages 31:1–31:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.MFCS.2019.31.

[48] Martin Grohe. Finite variable logics in descriptive complexity theory. *Bull. Symb. Log.*, 4(4):345–398, 1998. doi:10.2307/420954.

[49] Martin Grohe. Fixed-point logics on planar graphs. In *13th Annual IEEE Symposium on Logic in Computer Science, LICS 1998, Indianapolis, Indiana, USA, June 21-24, 1998*, pages 6–15. IEEE Computer Society, 1998. doi:10.1109/LICS.1998.705639.

[50] Martin Grohe. The quest for a logic capturing PTIME. In *23rd Annual IEEE Symposium on Logic in Computer Science, LICS 2008, 24-27 June 2008, Pittsburgh, PA, USA*, pages 267–271. IEEE Computer Society, 2008. doi:10.1109/LICS.2008.11.

[51] Martin Grohe. Fixed-point definability and polynomial time on graphs with excluded minors. In *25th Annual IEEE Symposium on Logic in Computer Science, LICS 2010, 11-14 July 2010, Edinburgh, United Kingdom*, pages 179–188. IEEE Computer Society, 2010. doi:10.1109/LICS.2010.22.

[52] Martin Grohe. *Descriptive Complexity, Canonization, and Definable Graph Structure Theory*. Cambridge University Press, Cambridge, 2017. doi:10.1017/9781139028868.

[53] Martin Grohe. The logic of graph neural networks. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*, pages 1–17. IEEE Computer Socieyt, 2021. doi:10.1109/LICS52264.2021.9470677.

[54] Martin Grohe and Sandra Kiefer. Logarithmic Weisfeiler-Leman identifies all planar graphs. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*,

volume 198 of *LIPIcs*, pages 134:1–134:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ICALP.2021.134.

[55] Martin Grohe, Moritz Lichter, and Daniel Neuen. The iteration number of the Weisfeiler-Leman algorithm. *arXiv preprint*, abs/2301.13317, 2023. to appear at LICS 2023. doi:10.48550/arXiv.2301.13317.

[56] Martin Grohe and Julian Mariño. Definability and descriptive complexity on databases of bounded tree-width. In *Database Theory - ICDT 1999, 7th International Conference, Jerusalem, Israel, January 10-12, 1999, Proceedings*, volume 1540 of *Lecture Notes in Computer Science*, pages 70–82. Springer, 1999. doi:10.1007/3-540-49257-7_6.

[57] Martin Grohe and Daniel Neuen. Canonisation and definability for graphs of bounded rank width. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*, pages 1–13. IEEE Computer Society, 2019. doi:10.1109/LICS.2019.8785682.

[58] Martin Grohe and Martin Otto. Pebble games and linear equations. *J. Symb. Log.*, 80(3):797–844, 2015. doi:10.1017/jsl.2015.28.

[59] Martin Grohe, Pascal Schweitzer, and Daniel Wiebking. Deep Weisfeiler Leman. In *31st Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 2600–2614. SIAM, 2021. doi:10.1137/1.9781611976465.154.

[60] The GAP group. GAP – groups, algorithms, and programming. Version 4.10.1, 2019.

[61] Yuri Gurevich. Logic and the challenge of computer science. In *Current Trends in Theoretical Computer Science*, pages 1–57. Computer Science Press, 1988.

[62] Yuri Gurevich. From invariants to canonization. *Bull. EATCS*, 63, 1997.

[63] Yuri Gurevich and Saharon Shelah. Fixed-point extensions of first-order logic. *Ann. Pure Appl. Log.*, 32:265–280, 1986. doi:10.1016/0168-0072(86)90055-2.

[64] Yuri Gurevich and Saharon Shelah. On finite rigid structures. *J. Symb. Log.*, 61(2):549–562, 1996. doi:10.2307/2275675.

[65] David Harel and David Peleg. On static logics, dynamic logics, and complexity classes. *Inf. Control.*, 60(1-3):86–102, 1984. doi:10.1016/S0019-9958(84)80023-6.

[66] Lauri Hella. Logical hierarchies in PTIME. *Information and Computation*, 129(1):1–19, 1996. doi:10.1006/inco.1996.0070.

[67] D. G. Higman. Coherent algebras. *Linear Algebra and its Applications*, 93:209–239, 1987. doi:10.1016/S0024-3795(87)90326-0.

[68] Bjarki Holm. *Descriptive complexity of linear algebra.* PhD thesis, University of Cambridge, 2011.

[69] Neil Immerman. Relational queries computable in polynomial time. *Inf. Control.*, 68(1-3):86–104, 1986. doi:10.1016/S0019-9958(86)80029-8.

[70] Neil Immerman. Expressibility as a complexity measure: results and directions. In *2nd Annual Conference on Structure in Complexity Theory, Cornell University, Ithaca, New York, USA, June 16-19, 1987.* IEEE Computer Society, 1987.

[71] Neil Immerman. Languages that capture complexity classes. *SIAM J. Comput.*, 16(4):760–778, 1987. doi:10.1137/0216051.

[72] Neil Immerman. *Descriptive complexity.* Graduate texts in computer science. Springer, New York, 1999. doi:10.1007/978-1-4612-0539-5.

[73] Neil Immerman and Eric Lander. Describing graphs: a first-order approach to graph canonization. In *Complexity theory retrospective*, pages 59–81. Springer, New York, 1990. doi:10.1007/978-1-4612-4478-3_5.

[74] Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972. doi:10.1007/978-1-4684-2001-2_9.

[75] Sandra Kiefer and Brendan D. McKay. The iteration number of colour refinement. In *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 73:1–73:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.73.

[76] Sandra Kiefer and Daniel Neuen. The power of the Weisfeiler-Leman algorithm to decompose graphs. *SIAM J. Discret. Math.*, 36(1):252–298, 2022. doi:10.1137/20m1314987.

[77] Sandra Kiefer and Daniel Neuen. A study of Weisfeiler-Leman colorings on planar graphs. In *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 81:1–81:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.81.

[78] Sandra Kiefer, Ilia Ponomarenko, and Pascal Schweitzer. The weisfeiler-leman dimension of planar graphs is at most 3. *J. ACM*, 66(6):44:1–44:31, 2019. doi:10.1145/3333003.

[79] Sandra Kiefer and Pascal Schweitzer. Upper bounds on the quantifier depth for graph differentiation in first-order logic. *Log. Methods Comput. Sci.*, 15(2), 2019. doi:10.23638/LMCS-15(2:19)2019.

[80] Sandra Kiefer, Pascal Schweitzer, and Erkal Selman. Graphs identified by logics with counting. In *40th International Symposium on Mathematical Foundations of Computer Science, MFCS 2015, Milan, Italy, August 24-28, 2015, Proceedings, Part I*, volume 9234 of *Lecture Notes in Computer Science*, pages 319–330. Springer, 2015. doi:10.1007/978-3-662-48057-1_25.

[81] Stephan Kreutzer. Expressive equivalence of least and inflationary fixed-point logic. *Ann. Pure Appl. Log.*, 130(1-3):61–78, 2004. doi:10.1016/j.apal.2004.02.001.

[82] Bastian Laubner. Capturing polynomial time on interval graphs. In *25th Annual IEEE Symposium on Logic in Computer Science, LICS 2010, 11-14 July 2010, Edinburgh, United Kingdom*, pages 199–208. IEEE Computer Society, 2010. doi:10.1109/LICS.2010.42.

[83] Bastian Laubner. *The structure of graphs and new logics for the characterization of Polynomial Time.* PhD thesis, Humboldt University of Berlin, 2011. doi:10.18452/16335.

[84] Moritz Lichter. Separating rank logic from polynomial time. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*, pages 1–13. IEEE Computer Society, 2021. doi:10.1109/LICS52264.2021.9470598.

[85] Moritz Lichter. Separating rank logic from polynomial time. *J. ACM*, 70(2), 03 2023. doi:10.1145/3572918.

[86] Moritz Lichter. Witnessed symmetric choice and interpretations in fixed-point logic with counting. In *50th International Colloquium on Automata, Languages, and Programming (ICALP 2023)*, volume 261 of *LIPIcs*, pages 133:1–133:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ICALP.2023.133.

[87] Moritz Lichter, Ilia Ponomarenko, and Pascal Schweitzer. Walk refinement, walk logic, and the iteration number of the Weisfeiler-Leman algorithm. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*, pages 1–13. IEEE Computer Society, 2019. doi:10.1109/LICS.2019.8785694.

[88] 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, January 25-28, 2021, Ljubljana, Slovenia (Virtual Conference)*, volume 183 of *LIPIcs*, pages 31:1–31:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.CSL.2021.31.

[89] Moritz Lichter and Pascal Schweitzer. Choiceless Polynomial Time with witnessed symmetric choice. In *37th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2022, Haifa, Israel, August 2 - 5, 2022*, pages 30:1–30:13. ACM, 2022. doi:10.1145/3531130.3533348.

[90] Eugene M. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *J. Comput. Syst. Sci.*, 25(1):42–65, 1982.

[91] Rudolf Mathon. A note on the graph isomorphism counting problem. *Inf. Process. Lett.*, 8(3):131–132, 1979. doi:10.1016/0020-0190(79)90004-8.

[92] Brendan D. McKay and Adolfo Piperno. Practical graph isomorphism, II. *J. Symb. Comput.*, 60:94–112, 2014. doi:10.1016/j.jsc.2013.09.003.

[93] Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and Leman go neural: Higher-order graph neural networks. In *33rd AAAI Conference on Artificial Intelligence, AAAI 2019, and 31st Innovative Applications of Artificial Intelligence Conference, IAAI 2019, and 9th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 4602–4609. AAAI Press, 2019. doi:10.1609/aaai.v33i01.33014602.

[94] Alan Nash, Jeffrey B. Remmel, and Victor Vianu. PTIME queries revisited. In *Database Theory - ICDT 2005, 10th International Conference, Edinburgh, UK, January 5-7, 2005, Proceedings*, volume 3363 of *Lecture Notes in Computer Science*, pages 274–288. Springer, 2005. doi:10.1007/978-3-540-30570-5_19.

[95] Daniel Neuen and Pascal Schweitzer. Benchmark graphs for practical graph isomorphism. In *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria*, volume 87 of *LIPIcs*, pages 60:1–60:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ESA.2017.60.

[96] Daniel Neuen and Pascal Schweitzer. An exponential lower bound for individualization-refinement algorithms for graph isomorphism. In *50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 138–150. ACM, 2018. doi:10.1145/3188745.3188900.

[97] Daniel Neuen and Pascal Schweitzer. Subgroups of 3-factor direct products. *Tatra Mt. Math. Publ.*, 73:19–38, 2019. doi:10.2478/tmmp-2019-0003.

[98] Martin Otto. *Bounded variable logics and counting - a study in finite models*, volume 9 of *Lecture Notes in Logic*. Springer, Berlin, Heidelberg, 1997.

[99] Martin Otto. Epsilon-logic is more expressive than first-order logic over finite structures. *J. Symb. Log.*, 65(4):1749–1757, 2000. doi:10.2307/2695073.

[100] Benedikt Pago. Choiceless computation and symmetry: Limitations of definability. In *29th EACSL Annual Conference on Computer Science Logic, CSL 2021, January 25-28, 2021, Ljubljana, Slovenia (Virtual Conference)*, volume 183 of *LIPIcs*, pages 33:1–33:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.CSL.2021.33.

[101] 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, February 13-16, 2023, Warsaw, Poland*, volume 252 of *LIPIcs*, pages 31:1–31:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.CSL.2023.31.

[102] Benedikt Pago. Lower bounds for choiceless polynomial time via symmetric xor-circuits. *arXiv prepint*, abs/2302.05426, 2023. doi:10.48550/arXiv.2302.05426.

[103] Wied Pakusa. *Linear Equation Systems and the Search for a Logical Characterisation of Polynomial Time.* PhD thesis, RWTH Aachen University, 2015.

[104] Wied Pakusa, Svenja Schalthöfer, and Erkal Selman. Definability of Cai-Fürer-Immerman problems in Choiceless Polynomial Time. In *25th EACSL Annual Conference on Computer Science Logic, CSL 2016, August 29 - September 1, 2016, Marseille, France*, volume 62 of *LIPIcs*, pages 19:1–19:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.CSL.2016.19.

[105] David Richerby. *Fixed-point logics with choice.* PhD thesis, University of Cambridge, 2004.

[106] Benjamin Rossman. Choiceless computation and symmetry. In *Fields of Logic and Computation, Essays Dedicated to Yuri Gurevich on the Occasion of His 70th Birthday*, volume 6300 of *Lecture Notes in Computer Science*, pages 565–580. Springer, 2010. doi:10.1007/978-3-642-15025-8_28.

[107] Horst Sachs. Regular graphs with given girth and restricted circuits. *J. London Math. Soc.*, 38(1):423–429, 1963. doi:10.1112/jlms/s1-38.1.423.

[108] Pascal Schweitzer and Constantin Seebach. Resolution with symmetry rule applied to linear equations. In *38th International Symposium on Theoretical Aspects of Computer Science, STACS 2021, March 16-19, 2021, Saarbrücken, Germany (Virtual Conference)*, volume 187 of *LIPIcs*, pages 58:1–58:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.STACS.2021.58.

[109] Pascal Schweitzer and Daniel Wiebking. A unifying method for the design of algorithms canonizing combinatorial objects. In *51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1247–1258. ACM, 2019. doi:10.1145/3313276.3316338.

[110] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. Weisfeiler-Lehman graph kernels. *J. Mach. Learn. Res.*, 12:2539–2561, 2011.

[111] Yaroslav Shitov. An improved bound for the lengths of matrix algebras. *Algebra & Number Theory*, 13(6):1501 – 1507, 2019. doi:10.2140/ant.2019.13.1501.

[112] Jacobo Torán. On the resolution complexity of graph non-isomorphism. In *16th International Conference on Theory and Applications of Satisfiability Testing, SAT 2013, Helsinki, Finland, July 8-12, 2013. Proceedings*, volume 7962 of *Lecture*

Notes in Computer Science, pages 52–66. Springer, 2013. doi:10.1007/978-3-642-39071-5_6.

[113] Jacobo Torán and Florian Wörz. Number of variables for graph differentiation and the resolution of GI formulas. In *30th EACSL Annual Conference on Computer Science Logic, CSL 2022, February 14-19, 2022, Göttingen, Germany (Virtual Conference)*, volume 216 of *LIPIcs*, pages 36:1–36:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.CSL.2022.36.

[114] Moshe Y. Vardi. The complexity of relational query languages (extended abstract). In *14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA*, pages 137–146. ACM, 1982. doi:10.1145/800070.802186.

[115] Boris Weisfeiler. *On construction and identification of graphs.* Lecture Notes in Mathematics, Vol. 558. Springer, Berlin, Heidelberg, 1976.

[116] Boris Weisfeiler and Andrei Leman. The reduction of a graph to canonical form and the algebra which appears therein. *NTI, Series 2*, 1968. English translation by Grigory Ryabov available at https://www.iti.zcu.cz/wl2018/pdf/wl_paper_translation.pdf.

[117] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[118] Faried Abu Zaid, Erich Grädel, Martin Grohe, and Wied Pakusa. Choiceless Polynomial Time on structures with small abelian colour classes. In *39th International Symposium on Mathematical Foundations of Computer Science, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part I*, volume 8634 of *Lecture Notes in Computer Science*, pages 50–62. Springer, 2014. doi:10.1007/978-3-662-44522-8_5.

[119] Alexander Zimmermann. *Representation Theory: A Homological Algebra Point of View.* Algebra and Applications. Springer International Publishing, 2014. doi:10.1007/978-3-319-07968-4.

List of Figures

List of Symbols

The following symbols and notations are used throughout this thesis. Specific symbols and notations are introduced in the corresponding chapters.

Basics

$\mathbb{N}$	set of natural numbers
$\mathbb{Z}$	set of integers
i, j, k, ℓ, m, n	natural numbers or integers
$\mathbb{F}_p$	finite prime field of characteristic p
$\mathbb{Z}_j$	ring of integers modulo j
$[k]$	the set $\{1, \dots, k\}$
$N \uplus M$	disjoint union of two sets N and M
$\{\{a_1, \dots, a_k\}\}$	multiset containing the elements $a_1, \dots, a_k$
$\mathbb{1}$, $\mathbb{0}$	identity matrix and zero matrix

Tuples

N^I	I-indexed tuples over N
$\bar{t}$, $\bar{s}$	tuples
$\bar{t}(i)$	entry for index $i \in I$
$\bar{t}\|_J$	restriction of a tuple to $J \subseteq I$
$T\|_J$	restriction of a set of tuples $T \subseteq N^I$ to J
$T\|^I$	extension of a set of tuples $T \subseteq N^J$ to $I \supseteq J$
N^k	set of tuples of length k over N
t_i	i-th entry of $\bar{t} \in N^k$
$N^{\leq k}$	set of tuples of length at most k over N
N^*	set of all tuples of finite length over N
$\bar{t}\bar{s}$	concatenation of two tuples

Permutation Groups

Γ, Δ	groups
$\mathsf{Sym}(\Omega)$	symmetric group with domain Ω
σ, τ	permutations (if not conflicting with signatures)
$\mathsf{ord}(\sigma)$	order of σ
O	orbit
$\mathsf{orb}_\Gamma(u)$	Γ-orbit of $u \in \Omega$
$\mathsf{orb}(\Gamma)$	set of 1-orbits of Γ
$\mathsf{orb}_k(\Gamma)$	set of k-orbits of Γ

Relational Structures

$\mathfrak{A}$, $\mathfrak{B}$, $\mathfrak{H}$	finite relational structures
A, B, H	universe of the structures
u, v, w	atoms of structures
$\bar{u}$, $\bar{v}$, $\bar{w}$	tuples of atoms
τ, σ	relational signatures
R, S	relation symbols
$R^{\mathfrak{A}}$, $S^{\mathfrak{A}}$	relation symbols interpreted in $\mathfrak{A}$
$\mathrm{ar}(R)$	arity of a relation R
$\preceq$	relation symbol for a total preorder
$\mathcal{K}$, $\mathcal{J}$	classes of structures
$\mathfrak{A}[A']$	substructure of $\mathfrak{A}$ induced by $A' \subseteq A$
$\mathfrak{A}[\bar{u}]$	substructure induced by the set of atoms in $\bar{u}$
$\mathfrak{A} \upharpoonright \sigma$	reduct of a τ-structure $\mathfrak{A}$ to $\sigma \subseteq \tau$
$\mathfrak{A} \cong \mathfrak{B}$	isomorphic structures
φ, ψ	isomorphisms or automorphisms of structures
$\mathsf{Aut}(\mathfrak{A})$	automorphism group of $\mathfrak{A}$
$\mathsf{Aut}((\mathfrak{A}, \bar{u}))$	automorphism group of $\mathfrak{A}$ stabilizing $\bar{u} \in A^*$
$\mathsf{orb}(\mathfrak{A})$	set of 1-orbits of $\mathfrak{A}$
$\mathsf{orb}_k(\mathfrak{A})$	set of k-orbits of $\mathfrak{A}$

Graphs

G, H	graphs
V	vertex set of a graph
E	edge relation of a graph
$G - V'$, $G - E'$	graph obtained by deleting the vertices $V' \subseteq V$ or edges $E' \subseteq E$ from G
$N_G(u)$	neighbors of the vertex u in the graph G
$N_G^k(u)$	k-neighborhood of u in G
$\mathsf{dist}_G(u, v)$	distance of u and v in G
$\mathsf{dist}_G(X, Y)$	distance of two vertex sets X and Y in G
$G[W]$	subgraph of G induced by $W \subseteq V$

Names of Logics

FO	first-order logic
IFP	(inflationary) fixed-point logic
IFPC	fixed-point logic with counting
$\mathcal{C}_k$	k-variable counting logic
$\mathcal{W}_k$	k-walk counting logic
CPT	Choiceless Polynomial Time
CPT+WSC	CPT with witnessed symmetric choice
IFP+WSC	IFP with witnessed symmetric choice
IFPC+WSC	IFPC with witnessed symmetric choice
IFPC+WSC+I	IFPC+WSC with the interpretation operator
IFP+SC	IFP with symmetric choice

IFPC+R	rank logic with the uniform rank operator rk
IFPC+R$_\Omega$	rank logic with the fixed characteristic rank operator rk_p for all $p \in \Omega$
IFPC+S$_\Omega$	solvability logic with solvability quantifiers for all $\mathbb{F}_p$ with $p \in \Omega$
LA	linear-algebraic logic

Logical Objects

L	some logic
$L[\tau]$	L-formulas and terms for the signature τ
$L[\tau, \sigma]$	L-interpretation from τ-structures to σ-structures
$L \leq L'$	L' is at least as expressive as L
$\equiv_L$	equivalence on finite structures induced by L
$\equiv_{\mathcal{C}}^k$	equivalence of $\mathcal{C}_k$
Φ, Ψ	formulas
s, t, r	terms
x, y, z	variables, in particular element variables for IFPC and extensions
ν, μ	numeric variables for IFPC and extensions
$\Phi(\bar{x})$	the free variables of Φ are among the variables in $\bar{x}$
$\Phi^{\mathfrak{A}}$	set of all tuples (to interpret the free variables of Φ) satisfying Φ in $\mathfrak{A}$
$s^{\mathfrak{A}}$	function mapping tuples to the object defined by s in $\mathfrak{A}$
Θ, Π, Υ	logical interpretations
$\mathfrak{A}^{\#}$, $\tau^{\#}$	extension of the structure $\mathfrak{A}$ or the signature τ with counting
λ	bijection in pebble games
$\mathsf{HF}(A)$	hereditarily finite sets over the set of atoms A
a, b, c	hereditarily finite sets in the context of CPT
a, b, c	elements of fields or rings in the context of CFI graphs
$\mathsf{TC}(a)$	transitive closure of a hereditarily finite set
$\mathsf{HF}(\mathfrak{A})$, τ^{HF}	extension of $\mathfrak{A}$ or τ with hereditarily finite sets

CFI Graphs and Structures

$\mathsf{CFI}_{\mathrm{ge}}(G, f)$	CFI graph with gadget and edge vertices over G
$\mathsf{CFI}_{\mathrm{g}}(G, f)$	CFI graph with gadget vertices over G
$\mathsf{CFI}_{\mathrm{e}}(G, f)$	CFI graph with edge vertices over G
$\mathsf{CFI}_{\mathrm{ge}}(\mathcal{K})$	class of CFI graphs over the class of base graphs $\mathcal{K}$
$\mathfrak{u}$, $\mathfrak{v}$, $\mathfrak{w}$	base vertices of the base graph G
$\mathfrak{e}$	base edge of the base graph G
$\bar{\mathfrak{s}}$	path in the base graph G
$\mathsf{orig}(u)$	origin of a vertex u of a CFI graph
$\mathsf{CFI}_{2^q}(G, f)$	(generalized) CFI structure over $\mathbb{Z}_{2^q}$
$\mathsf{CFI}_{2^q}(\mathcal{K})$	class of CFI structure over $\mathbb{Z}_{2^q}$ and over the base graph class $\mathcal{K}$
$\mathsf{CFI}_{2^\omega}(\mathcal{K})$	class of CFI structure over all $\mathbb{Z}_{2^q}$ over the base graph class $\mathcal{K}$
$\mathsf{CFI}_{2^q}(G, g)[W]$	origin-induced subgraph of a CFI structure by a set of base vertices W
$\vec{\pi}[c, \bar{\mathfrak{s}}]$	path-isomorphism moving a twist with value c along the path $\bar{\mathfrak{s}}$
$\pi^*[\bar{c}, \bar{\mathfrak{s}}_1, \dots, \bar{\mathfrak{s}}_\ell]$	star-isomorphism moving twists with values $\bar{c}$ along the star $(\bar{\mathfrak{s}}_1, \dots, \bar{\mathfrak{s}}_\ell)$

In the individual chapters, we may focus on a single variant of the CFI construction and use $\mathsf{CFI}(G, f)$ for $\mathsf{CFI}_{\mathrm{ge}}(G, f)$, $\mathsf{CFI}_{\mathrm{g}}(G, f)$, or $\mathsf{CFI}_{\mathrm{e}}(G, f)$.

Index

Academic Curriculum Vitae

2012-2015	B.Sc. Computer Science *Technical University Darmstadt*
2015-2017	M.Sc. Computer Science *Saarland University*
2018-2021	PhD student & research assistant Supervisor: Prof. Dr. Pascal Schweitzer *TU Kaiserslautern*
2021-2023	PhD student & research assistant Supervisor: Prof. Dr. Pascal Schweitzer *Technical University Darmstadt*