

The Structure of Graphs and New Logics for the Characterization of Polynomial Time

DISSERTATION

zur Erlangung des akademischen Grades

Doctor Rerum Naturalium
im Fach Informatik

eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät II
Humboldt-Universität zu Berlin

von

M.Sc. Bastian Laubner

geboren am 2. März 1982 in Berlin

Gefördert durch die Deutsche Forschungsgemeinschaft im Rahmen des Graduiertenkollegs
'Methods for Discrete Structures' (GRK 1408)

Präsident der Humboldt-Universität zu Berlin:

Prof. Dr. Jan-Hendrik Olbertz

Dekan der Mathematisch-Naturwissenschaftlichen Fakultät II:

Prof. Dr. Peter Frensch

Gutachter:

1. Prof. Dr. Martin Grohe

2. Prof. Dr. Erich Grädel

3. Prof. Dr. Anuj Dawar

eingereicht am: 22. November 2010

Tag der mündlichen Prüfung: 23. Februar 2011

Zusammenfassung

Die deskriptive Komplexitätstheorie ist der Bereich der theoretischen Informatik, der sich mit abstrakten Charakterisierungen von maschinellen Berechnungen durch mathematische Logiken beschäftigt. Ähnlich der gewöhnlichen Komplexitätstheorie ist das Ziel, Berechnungsprobleme anhand der Ressourcen zu klassifizieren, die für ihre Lösung benötigt werden. Anstelle von maschinellen Ressourcen betrachtet die deskriptive Komplexitätstheorie jedoch die logische Ausdrucksstärke, die benötigt wird, um eine Klasse von Problemen zu definieren. Der Zweck besteht darin, alternative Charakterisierungen von Komplexitätsklassen zu finden, mit denen sich die Stärken und Schwächen von maschinellen Berechnungen ohne Bezugnahme auf das konkrete Berechnungsmodell untersuchen lassen.

Diese Arbeit leistet einen Beitrag zu drei Teilgebieten der deskriptiven Komplexitätstheorie. Zunächst betrachten wir ein Missverhältnis zwischen maschinellen Berechnungen, deren Eingabe stets aus geordneten Wörtern besteht, und logischen Definitionen, deren abstrakte Betrachtung von Eingabestrukturen ohne Ordnung auskommt. Wir zeigen, dass ein kombinatorischer Graphkanonisierungsalgorithmus mit einfach exponentieller Laufzeit von Corneil und Goldberg (1984) auf kantengefärbte Graphen ausgeweitet werden kann. Die Repräsentationsinvarianz dieses Algorithmus gestattet uns die Implementierung in der Logik „Choiceless Polynomial Time with Counting“ ($\tilde{C}PT+C$) bei Beschränkung auf logarithmisch große Fragmente solcher Graphen. Auf Strukturen, deren Relationen höchstens Stelligkeit 2 haben, charakterisiert $\tilde{C}PT+C$ damit gerade die Polynomialzeit-Eigenschaften (PTIME) von Fragmenten logarithmischer Größe.

Der zweite Beitrag untersucht die deskriptive Komplexität von PTIME-Berechnungen auf eingeschränkten Graphklassen. Wir stellen eine neuartige Normalform von Intervallgraphen vor, die sich in Fixpunktlogik mit Zählen (FP+C) definieren lässt, was bedeutet, dass FP+C auf dieser Graphklasse PTIME charakterisiert. Die Methoden, die wir dafür entwickeln, machen erstmals die modulare Zerlegung von Graphen nutzbar für eine systematische Ergründung des Verhältnisses zwischen Logiken und Komplexitätsklassen. Wir führen anschließend das Konzept von „Non-Capturing“-Reduktionen ein, das uns zu einer Reihe von Graphklassen führt, auf denen das Einfangen von Komplexitätsklassen genau so schwer ist wie auf der Klasse aller Graphen. Schließlich adaptieren wir unsere Methoden, um einen kanonischen Beschriftungsalgorithmus für Intervallgraphen zu erhalten, der sich mit logarithmischer Platzbeschränkung (LOGSPACE) berechnen lässt. Wir ziehen den Schluss, dass das Intervallgraphen-Isomorphieproblem vollständig ist für die Klasse LOGSPACE. Auf diese Art leisten unsere logischen Methoden einen Beitrag zur klassischen Komplexitätstheorie.

Im dritten Teil dieser Arbeit beschäftigt uns die ungelöste Frage, ob es eine Logik gibt, die alle Polynomialzeit-Berechnungen charakterisiert. Wir führen eine Reihe von Ranglogiken ein, die die Fähigkeit besitzen, den Rang von Matrizen über Primkörpern zu berechnen. Wir zeigen, dass diese Ergänzung um lineare Algebra robuste Logiken hervor bringt, und dass diese die Ausdrucksstärke von FP+C und vielen anderen Logiken übertreffen, die im Zusammenhang mit dieser Frage betrachtet werden. Durch Anpassung einer Konstruktion von Hella (1996) gelingt es uns außerdem zu beweisen, dass Ranglogiken strikt an Ausdrucksstärke gewinnen, wenn wir die Zahl an Variablen erhöhen, die die betrachteten Matrizen indizieren. Wir rechtfertigen unsere spezielle Wahl von Rangberechnungen dadurch, dass die meisten klassischen Probleme der linearen Algebra entweder in Ranglogiken oder bereits in FP+C definierbar sind. Dann bauen wir eine Brücke zur klassischen Komplexitätstheorie, indem wir über geordneten Strukturen eine Reihe von Komplexitätsklassen zwischen LOGSPACE und PTIME durch Ranglogiken charakterisieren. Unsere Arbeit etabliert die stärkste unserer Ranglogiken als Kandidat für die Charakterisierung von PTIME und legt nahe, dass Ranglogiken genauer erforscht werden müssen, um weitere Fortschritte zu erzielen im Hinblick auf eine Logik für Polynomialzeit.

Abstract

Descriptive complexity theory is the area of theoretical computer science that is concerned with the abstract characterization of computations by means of mathematical logics. It shares the approach of complexity theory to classify problems on the basis of the resources that are needed to solve them. Instead of computational resources, however, descriptive complexity theory considers the expressiveness necessary for a logic to define classes of problems. Its primary goal is to provide alternative characterizations of computational complexity that give us a way to reason about the strengths and limitations of computational procedures without reference to the underlying machine model.

This thesis is making contributions to three strands of descriptive complexity theory. First, we consider an incongruence between machine computations, which receive ordered strings as their input, and logics, whose abstract view on input structures omits the ordering. We show that a combinatorial, singly exponential-time graph canonization algorithm of Corneil and Goldberg (1984) can be extended to edge-colored directed graphs. The algorithm's input invariance allows us to implement it in the logic Choiceless Polynomial Time with Counting ($\tilde{C}PT+C$) if we restrict our attention to logarithmic-sized fragments of such graphs. This means that on structures whose relations are of arity at most 2, the polynomial-time (PTIME) properties of logarithmic-sized fragments are precisely characterized by $\tilde{C}PT+C$.

The second contribution investigates the descriptive complexity of PTIME computations on restricted classes of graphs. We present a novel canonical form for the class of interval graphs which is definable in fixed-point logic with counting (FP+C), resulting in FP+C capturing PTIME on this graph class. The methods developed for this result may serve as the foundation for a systematic study of the interrelation of logics and complexity classes on the basis of modular decompositions of graphs. Then we introduce the notion of non-capturing reductions that lead us to a wide variety of graph classes on which computational problems are equally hard to characterize as on the class of all graphs. We finally mold our methods into a canonical labeling algorithm for interval graphs which is computable in logarithmic space (LOGSPACE) and we draw the conclusion that interval graph isomorphism is complete for LOGSPACE. In this way, our methods developed for the logical domain make a contribution to classical complexity theory.

In the final part of this thesis, we take aim at the open question whether there exists a logic which generally captures polynomial-time computations. We introduce a variety of rank logics with the ability to compute the ranks of matrices over (finite) prime fields. We argue that this introduction of linear algebra makes for a robust notion of a logic and that it increases the expressiveness of FP+C and many other logics considered in the context of capturing PTIME. By adapting a construction of Hella (1996), we establish that rank logics strictly gain in expressiveness when we increase the number of variables that index the matrices we consider. Rank computations are the first natural operation for which this property is shown in the presence of recursion. Our particular choice of rank computations is justified by showing that most classic problems in linear algebra can either be defined by rank logics or are already definable in FP+C. Then we establish a direct connection to standard complexity theory by showing that in the presence of orders, a variety of complexity classes between LOGSPACE and PTIME can be characterized by suitably defined rank logics. Our exposition provides evidence that rank logics are a natural object to study and establishes the most expressive of our rank logics as a viable candidate for capturing PTIME, suggesting that rank logics need to be better understood if progress is to be made towards a logic for polynomial time.

Acknowledgements

I am indebted to my supervisor Martin Grohe for his guidance and support throughout the time I spent researching for this thesis. He brought to my attention most of the questions I investigated, worked closely together with me on some of them, and has helped me hone my understanding of all these problems in countless discussions.

I am grateful to my coauthors Anuj Dawar, Bjarki Holm, Johannes Köbler, Sebastian Kuhnert, and Oleg Verbitsky for the experience I gained during these collaborations. I would like to give particular thanks to Anuj Dawar and Bjarki Holm for hosting me in Cambridge on two occasions and for the exciting research we have done together.

I would like to thank the lively group of researchers at Martin Grohe's Lehrstuhl for spirited discussions and being there for all sorts of concerns. Special thanks go to my office mate Holger Dell for keeping me sane.

Thanks go to the Graduiertenkolleg "Methods for Discrete Structures" for the inspirational academic environment it provided and for the financial support.

I would like to express my gratitude to Christoph Berkholz, Holger Dell, Kord Eickmeyer, Berit Grußien, André Hernich, Bjarki Holm, Daniel Kirsten, Siamak Tazari, and Nils Vortmeier for proofreading and pointing out inaccuracies in parts of earlier versions of this thesis. I am particularly indebted to Vivian Ebert for proofreading the entire final draft of this document.

Last but not least, I would like to thank my parents Ellen and Peer for their kind support throughout all this time.

Contents

1	Introduction	1
1.1	Descriptive complexity theory	1
1.2	Contributions of this thesis	6
1.3	About this thesis	8
2	Notation, Definitions, Foundational Results	11
2.1	Graphs	12
2.1.1	Colored graphs	13
2.1.2	Intersection graphs	13
2.1.3	Graph classes	14
2.1.4	Graph reachability	14
2.2	Complexity classes	15
2.3	Logics	16
2.3.1	Capturing complexity classes	17
2.3.2	Plain logics	18
2.3.3	Counting logics	21
2.3.4	Logical reductions	22
2.4	Definable canonization mappings	25
2.4.1	Basic formulas	27
3	Capturing Polynomial Time on Fragments of Logarithmic Size	31
3.1	Isomorphism algorithms for edge-colored graphs	33
3.1.1	Encoding edge-colored directed graphs as undirected graphs	33
3.1.2	Existing isomorphism algorithms	35
3.2	Basic notions	36
3.2.1	Choiceless Polynomial Time	36
3.2.2	Properties of $\tilde{CPT}+C$	40
3.2.3	Stable colorings and Weisfeiler-Lehman color refinement	41
3.2.4	Individualization of vertices	43
3.3	Sections of stable partitions	44
3.3.1	Definition of sections	44
3.3.2	Finding sections	45
3.4	The singly exponential canonization scheme	49
3.5	Bounding the size of the recursion tree	52
3.6	Discussion	53
4	Capturing Results on Classes of Graphs	55
4.1	Interval graphs	57
4.1.1	Definition and basic properties	57
4.1.2	Deciding the classes of interval graphs, chordal graphs, and comparability graphs	59
4.1.3	Modular decomposition trees	60

Contents

4.1.4	Existing algorithms for interval graphs	60
4.2	Logics <u>not</u> capturing complexity classes on graph classes	61
4.2.1	Non-capturing reductions	62
4.2.2	Non-capturing results for FP+C	64
4.3	Capturing PTIME on interval graphs	66
4.3.1	Extracting information about the order of maximal cliques	67
4.3.2	Canonizing when \prec_M is a linear order	71
4.3.3	Canonizing general interval graphs	72
4.4	The quest for a logic capturing LOGSPACE on interval graphs	74
4.4.1	Capturing LOGSPACE on proper interval graphs	74
4.4.2	STC+C does not capture LOGSPACE on <i>all</i> interval graphs	76
4.5	Canonizing interval graphs in LOGSPACE	77
4.5.1	LOGSPACE-hardness of interval graph isomorphism	78
4.5.2	Constructing a modular tree for interval graphs	79
4.5.3	Lindell's (colored) tree isomorphism order	84
4.5.4	Canonical interval representations and orderings of interval graphs in logspace	85
4.6	Discussion	87
5	Stronger Logics for Polynomial Time	89
5.1	Definition of Rank Logics	90
5.2	Basic properties of rank operators	94
5.2.1	Counting	95
5.2.2	Similarity of matrices	95
5.2.3	Linear systems	96
5.2.4	Closure under logical reductions	97
5.2.5	Undirected and deterministic graph reachability in FO+rk _p	98
5.2.6	Counting paths modulo p in directed acyclic graphs	99
5.2.7	Matrix multiplication	100
5.2.8	Directed graph reachability in FO+rk	101
5.3	Linear algebra in FP+C and FO+rk	102
5.3.1	Methods for calculating determinants and characteristic polynomials	103
5.3.2	Binary and Chinese remainder representations	105
5.3.3	Implementing Le Verrier's method in FP+C and FO+rk	107
5.3.4	Determinants, ranks, and matrix inverses	110
5.4	Cai-Fürer-Immerman graphs	111
5.5	Descriptive complexity of rank logics over ordered structures	113
5.5.1	FO+rk _p captures MOD _p L on ordered structures	114
5.5.2	FO+rk captures the logspace counting hierarchy on ordered structures	116
5.6	The arity hierarchy of rank operators	121
5.6.1	Characterizing Lindström quantifiers	122
5.6.2	Embedding FP+rk in infinitary logic with rank quantifiers	124
5.6.3	The construction of separating queries	127
5.6.4	Proving the queries to be undefinable for low quantifier arity	131
5.7	Discussion	133
6	Conclusion	139
	Bibliography	143

1 Introduction

The motivation behind the work in this thesis traces back to the most important problem in complexity theory: the question whether non-deterministic polynomial-time bounded computations can be cast in deterministic polynomial time. This has become famous as the $P \stackrel{?}{=} NP$ question, which could not be resolved despite almost four decades of intensive research. The question is not only baffling from a theoretician's standpoint but it also has far-reaching consequences for practical computational problems. While the class P (or $PTIME$)¹ is commonly accepted as a good theoretic model of what can be solved efficiently on a real-world computer, its non-deterministic counterpart NP contains a vast array of problems of practical importance for which it is unknown whether exact solutions for large instances can be computed efficiently.

In the course of investigating the $P \stackrel{?}{=} NP$ question, a wide variety of approaches was developed to better understand the nature of these complexity classes. Cook's 1972 discovery [Coo72] of problems inside NP that all of NP can be efficiently reduced to greatly enhanced the understanding of the structure of NP and lay the groundwork for the later investigations into $P \stackrel{?}{=} NP$. An efficient algorithm for any of these NP -complete problems would imply that P and NP are the same. A popular theoretical approach in the 1980s was relativization which aims at distinguishing complexity classes by their behavior when they may refer certain computational tasks to oracles of varying power. More recent strands of research include the complexity of computation by circuits, the role of randomness in computation as it is epitomized by the $P \stackrel{?}{=} BPP$ question, and probabilistically checkable proofs which led to the celebrated PCP Theorem's characterization of NP . While these approaches generally emphasize different aspects of computability, they all provide models that enhance our understanding of the composition and dynamics underpinning the complexity of computational problems.

This thesis is dedicated to yet another approach for characterizing complexity classes such as P and NP . Instead of asking what resources are necessary for a machine to compute a given problem, we aim to classify the problem by asking *how difficult it is to describe*. More precisely, we want to know how expressive a logical system has to be for it to be able to define the problem at hand. Any mathematical logic such as first-order logic FO can be associated with the problems which are definable in it. In this way logics describe their own classes of problems. The most interesting cases are those where a problem class defined by a logic coincides with the class defined by a machine model so that the complexity of the problems in such a class is then characterized in two ways: computationally and descriptively. The field which is concerned with finding logical characterizations of complexity classes is aptly named *descriptive complexity theory*.

1.1 Descriptive complexity theory

The starting point of descriptive complexity was Fagin's 1974 finding [Fag74] that the problems in NP are precisely those that can be defined by means of existential second-order logic $\exists SO$, i.e. first-order logic together with existential quantification over relations. In short, the expressive power of $\exists SO$ captures NP 's computational power entirely. Given the importance of the $P \stackrel{?}{=} NP$

¹ $PTIME$ is the class of problems solvable by deterministic polynomial-time Turing machines.

1 Introduction

question, it was only natural when in 1988, Gurevich asked whether a similar logical characterization can be found for the class of problems decidable by deterministic polynomial-time Turing machines [Gur88]. His question for a *logic capturing* PTIME is still open over twenty years later and has become the most important open problem in descriptive complexity and finite model theory.

Gurevich himself conjectured at the time that no such logic for PTIME exists. The precise conditions such a logic must satisfy depend on subtleties which we will illuminate later. In any case, we should not expect a proof of Gurevich's conjecture to come easy, since verification of the non-existence of such a logic in any meaningful sense would distinguish PTIME from NP and therefore give a negative answer to the seminal $P \stackrel{?}{=} NP$ question. The possibility of separating P from NP alone makes logical characterizations of these complexity classes an attractive approach to start with. But even if we do not heed Gurevich's negative hypothesis, the pursuit of a logic for PTIME is a promising method for discovering integral ingredients of polynomial-time computations. In fact, past decades' progress in the field of descriptive complexity has revealed that, at the very least, large fragments of PTIME can be captured with rather natural logical formalisms. Crucially, any of these results tells us that the problems contained in these fragments are built up from only a small selection of simple building blocks.

At this point, it is important to note a conceptual difference between machine computations and logical definitions. While machines decide strings which are written on their input tape, logics define properties of more general structures. The problems of interest in complexity theory are themselves often times properties of structures rather than deciding mere classes of strings, so logics actually provide more direct access to the problems that stand to be decided. Examples of such problems are tests for the connectedness or the 3-colorability of graphs. In order to decide such properties, Turing machines have to be provided with a string encoding the input structure. In the particular case when a logic captures a complexity class, this descriptive characterization is testimony to the machine model's indifference to the detour of string encodings and how the machine's computational power propagates fluently to general structures. In the search for capturing results, however, incorporating the encoding step of structures is an additional stumbling block.

The Immerman-Vardi theorem and capturing on ordered structures

Capitalizing on the additional power coming from a linear order, Immerman [Imm82] and Vardi [Var82] independently gave an important partial answer in the quest for a logic capturing PTIME in 1982, now known as the Immerman-Vardi Theorem: first-order logic with a certain kind of fixed-point operator (FP) defines precisely all polynomial-time queries of structures which carry a linear ordering of their universe. Ordered structures are conceptually close to strings since there is a canonical way to encode ordered structures for the purpose of writing them onto a Turing machine's input tape. In short, their result says that FP captures PTIME *on ordered structures*. The fixed-point operators considered by Immerman and Vardi are a way to provide first-order logic with a means of recursion which is otherwise absent from FO. Fixed-point operators had been considered previously in model theory, but only their application in finite model theory invests them with the additional significance of expressing efficient real-world computations.² The ability to define these fixed-points can be likened to the capability of making *inductive definitions*. We may say that when an ordering is present, polynomial-time computations are entirely composed of the simple rules of first-order predicate calculus and the evaluation of inductive definitions.

Under the requirement that structures are ordered, a variety of further capturing results of complexity classes were discovered using suitably tailored logics. By a result of Abiteboul and

²Here we happily assume that Turing machines are *the right model* for all kinds of computations possible in the real world; a belief commonly known as the *Church-Turing Thesis*.

Vianu [AV89], FO with a different, potentially more powerful fixed-point operator for computing so-called *partial fixed-points* (PFP), captures the class PSPACE of all problems solvable with polynomial workspace on ordered structures (also see [EF99]). As for weaker complexity classes, problems decidable in non-deterministic logarithmic space are precisely characterized on ordered structures by the extension of FO with an operator that defines the transitive closure of relations (cf. [Imm99]). Similarly, deterministic logspace is captured by FO with a deterministic restriction of the transitive closure operator (ibid.). Especially for the capturing of complexity classes which are contained in PTIME, it appears that the presence of an order is of high significance since no capturing result on general structures could yet be found for any of these complexity classes.

Despite the order requirement, the Immerman-Vardi Theorem has direct implications for descriptive complexity theory. For example, it implies that PTIME equals the complexity class PSPACE if and only if least fixed-point logic FP and partial fixed-point logic PFP (see preceding paragraph) are equally expressive on ordered structures. What is more, Abiteboul and Vianu [AV91] could show that the requirement of structures to be ordered can be avoided, so that $\text{PTIME} = \text{PSPACE}$ if and only if FP is as expressive as PFP. Similarly, $\text{P} = \text{NP}$ if and only if FP is just as expressive as $\exists\text{SO}$ on the class of all ordered structures. In this last correspondence, the restriction to ordered structures can unfortunately not be easily dropped. If it could, this would prove $\text{P} \neq \text{NP}$ since there is a simple (unordered) query which is expressible in $\exists\text{SO}$ but not in FP.

Separations on ordered structures

At this point we could contend that the descriptive approach has done its part and we should now concentrate on finding methods to separate FP and $\exists\text{SO}$ on the class of ordered structures. Unfortunately, this task seems to be hardly any easier than working directly with the corresponding machine models. There are working characterizations of a number of small variations of FO which can be used to separate these logics from $\exists\text{SO}$ over ordered structures (cf. [Imm99]). However, for a logic which seems only slightly stronger, FO with so-called majority quantifiers (FO+M), there is no approach known to separate FO+M and $\exists\text{SO}$ on ordered structures. There has certainly been a score of attempts at such a separation since on ordered structures, FO+M captures the circuit complexity class TC^0 [BIS90], which is widely believed but not proven to be strictly contained in NP. Given that FO+M is also believed to be strictly less expressive than FP on ordered structures, it is hard to fathom how an even stronger separation result of FP and $\exists\text{SO}$ could be obtained on ordered structures. Similar remarks hold for separating FO with the above-mentioned transitive-closure operators from $\exists\text{SO}$.

So it seems that FP's recursion mechanism together with an ordering is still very close to the original polynomial-time computation model, suggesting that it is not much easier to prove $\text{FP} \neq \exists\text{SO}$ on ordered structures than showing $\text{P} \neq \text{NP}$ directly. Given that logical characterizations of PTIME on ordered structures have not warranted a breakthrough in the $\text{P} \stackrel{?}{=} \text{NP}$ question, let us next look at what there is to gain from logical characterizations of PTIME on unordered structures.

Model theoretic methods in descriptive complexity

Logical characterizations of complexity classes such as Fagin's open the doors for transferring classical methods from logic and model theory to problems of relating complexity classes. This is particularly true of *game characterizations* of a logic's expressiveness, which are a popular tool in model theory for proving that certain queries cannot be expressed in a given logic. *Ehrenfeucht-Fraïssé style pebble games*, for example, are typically played by two players: spoiler and duplicator. Spoiler's job is to find proof that two given structures are different, while duplicator tries to

1 Introduction

avoid this within the rules of the game. Now if a logic's expressiveness is suitably characterized by such a game, proving that this logic cannot define a certain query amounts to designing a winning strategy for duplicator for the comparison of structures from inside and outside the query. Since these games typically have a simple set of rules, they often have a conceptual advantage over other methods for proving separation results.

Game characterizations have been employed successfully to show that various candidates for capturing PTIME and other complexity classes were actually too weak for these tasks. We use an Ehrenfeucht-Fraïssé style game ourselves in Chapter 5 to show a separation result in the expressiveness of rank operators. If a sensible logic for PTIME were to be discovered, the hope would be that it does not only give insight into the inherent characteristics of polynomial-time computations, but that such logical methods open an inroad for a true separation (or equality) proof of $P \stackrel{?}{=} NP$.

There are many other types of games considered in the context of descriptive complexity. An important family of games are model checking games, which can be used to determine if a specific logical sentence holds in a given structure. Grädel gives a good overview of model checking games in [Grä02]. All the games in finite model theory derive from logics which may only use a bounded number of variables – otherwise these logics would be too expressive. Dawar [Daw99], Grohe [Gro98a], and Otto [Ott97] provide good overviews of bounded-variable logics in finite model theory, as well as their applications and pebble games.

As just mentioned, there are limits to the application of model theoretic methods in complexity theory. Most importantly, classical complexity classes such as P and NP work over strings of finite length. Consequently, they may only decide properties of *finite structures*, whereas general model theory is very much concerned with models of infinite size. Unfortunately, many properties distinguishing logics over infinite structures, such as compactness and the Löwenheim-Skolem property, are void when considered only in the realm of finite models. This difference has had a great impact on the logics considered in descriptive complexity. In classical model theory, first-order logic FO is the classical workhorse, as it is the most expressive logic which is compact and satisfies the Löwenheim-Skolem property (cf. [EFT94]). In finite model theory, however, FO turns out to be quite weak for most practical purposes, which explains the large variety of different logics that have been driving the development of finite model theory in the past twenty years. The methods developed in this context have established descriptive complexity theory as a field in its own right.

The question of Chandra and Harel

There are further reasons to continue the search for a logic which captures PTIME not only on ordered structures, but on all structures. From a practical point of view, the most important reason comes from database theory, where Chandra and Harel [CH82] gave the first formulation of Gurevich's question for a logic capturing PTIME already in 1982. Aho and Ullman [AU79] had previously realized that SQL, the standard query language for relational databases, is incapable to express all polynomial-time queries. Chandra and Harel then asked if there is a *recursive enumeration* of all PTIME queries, which is equivalent to the question whether there is a query language for databases expressing all polynomial-time queries, and which comes along with an evaluator answering each of the language's queries in polynomial time.³ The advantages of such a query language are clear: it would give the database operator the maximal freedom to query her database all the while being certain that the query will be evaluated by and large within a manageable time frame.

³To be precise, the polynomial time bound may well depend on the query. The evaluator must achieve this query's polynomial time bound itself; it is not enough that just any polynomial-time algorithm exists.

The Immerman-Vardi Theorem actually implies that there is such a query language for relational databases, allowing to construct queries from FO and fixed-point operators. It has the drawback, though, that some order on the database's contents has to be assumed and used in defining the queries, otherwise the language falls far short of expressing all possible efficient queries. Certainly, databases in real-world applications do always come equipped with an order deriving from the physical order in which the data is stored. However, relational databases are supposed to abstract away from the physical storage of their data; meanwhile, operators should ideally be able to work with just the database's table representation, unconcerned with its physical underpinning.

Logics with counting

Given these good reasons for finding a true logic for PTIME, the search for it continued. It was soon realized that on structures without an ordering, FP suffers from an inability to define many simple queries that involve counting. For example, it is not possible to define in FP the class of all those structures whose universe has even size (cf. [EF99]). Clearly, the class of even-sized structures is easy to decide by a PTIME machine. Acknowledging this obvious shortcoming, Immerman proposed to *add* to FP the ability to count and ventured that the resulting logic FP+C might capture PTIME [Imm87]. In 1992, however, Immerman himself could show together with Cai and Fürer [CFI92] that FP+C is too weak to capture PTIME on all structures by exhibiting a property of graphs which is decidable in PTIME but not definable in FP+C. It remained open at the time as to what other operators might be sensible additions to FP in order to overcome this weakness of FP+C. In Chapter 5, we propose a new kind of operator to decide the rank of matrices over finite fields and give ample justification why such rank logics are the most sensible extension of counting logics in the quest for a logic capturing PTIME.

Current candidates for capturing polynomial time

In the sequel to Cai, Fürer, and Immerman's 1992 dethronement of FP+C, only a few more logics were introduced as possible candidates for capturing PTIME. Blass, Gurevich, and Shelah [BGS99] proposed a computation model that lacks the ability to make arbitrary choices, called Choiceless Polynomial Time $\tilde{C}PT$. The fundamental idea of $\tilde{C}PT$ is to implement algorithms directly over structures instead of string encodings of them. This gives the logic $\tilde{C}PT$ a rather algorithmic feel. Thus, $\tilde{C}PT$ panders to the idea of operating computers on the basis of a virtual abstraction instead of physical representations of its data, which we alluded to in the context of Chandra and Harel's question. When structures come equipped with an ordering, then $\tilde{C}PT$ captures PTIME just like FP. Though $\tilde{C}PT$ is strictly more expressive than FP on general structures, it suffers from a similar innumeracy as FP. It is an open question whether $\tilde{C}PT+C$, the augmentation of $\tilde{C}PT$ with counting, captures PTIME. We will consider the logic $\tilde{C}PT+C$ ourselves in Chapter 3.

Another logic was introduced by Blass and Gurevich (cp. [Gur88]) and is rather technical in the sense that it tries to define away the lack of an ordering. The logic consists of FP-sentences φ which may make use of an order symbol $<$. An unordered structure \mathcal{A} is then said to model φ if in \mathcal{A} and in all structures \mathcal{B} of size at most that of \mathcal{A} , the truth value of φ does not depend on the way in which $<$ orders the structure's universe (in these structures, φ is called order-invariant). While this logic can certainly express all polynomial-time queries, it is hard to see how a PTIME machine could check the order-invariance property on all structures smaller than the one which is considered. In fact, Gurevich showed that for some formulas, deciding order-invariance on just one given structure is a co-NP-hard problem [Gur88]. Chen and Flum [CF10] recently added significance to the above logic by linking it to proof theory, showing that there is a certain optimal

1 Introduction

proof system for tautologies of propositional logic if and only if this logic effectively captures PTIME.

Yet another approach considers the carefree addition of PTIME properties to FO. The framework for this is provided by generalized quantifiers which are also known as Lindström quantifiers. We also consider Lindström quantifiers in Chapter 5. Certainly, by adding all PTIME properties to FO, we may express all PTIME properties in this logic. However, without a way to decide which generalized quantifiers actually encode order-invariant PTIME-properties on unordered structures, this trick does not provide us with a satisfactory logic for PTIME. Of course, this trouble links back to Chandra and Harel’s question whether the order-invariant polynomial-time properties can actually be recursively enumerated. Dawar could show that such a recursive enumeration exists if and only if there is a single Lindström quantifier from which a logic for PTIME can be obtained by a very uniform vectorization operation [Daw95].

The logics considered in this thesis will largely be of a more natural kind than the last two examples. More naturally formed logics have the advantage that it is easier to separate fundamental abilities of PTIME computations from those which can be derived easily. The discovery of the inherent building blocks of complexity classes is arguably the most important goal of descriptive complexity theory, and all of the results presented here should be seen in this light.

1.2 Contributions of this thesis

The main results of this thesis are presented in Chapters 3, 4, and 5. Chapter 2 provides precise definitions for the objects we investigate, along with a collection of the fundamental results upon which we build.

This thesis’s first contribution in Chapter 3 relates back to the prominent role played by orderings of a structure and is devoted to the question of how difficult it is to obtain an order of any given structure when our logical system does not have access to an auxiliary order to start with. Building on work by Corneil and Goldberg [CG84], we show that ordered canonical forms of edge-colored directed graphs can be obtained in $2^{O(n)}$ time, where n is the number of the graph’s vertices. This is superior to the naïve approach of searching the space of all $n!$ linear orders. The algorithmic procedure has the important feature that it avoids making arbitrary choices, so it can be implemented without reference to an auxiliary order of the graph’s vertices.

Given the exponential run-time of the canonization procedure, it cannot be implemented in any logic whose sentences can be evaluated in polynomial time. However, if we restrict our attention to logarithmic-sized fragments of edge-colored graphs, then the logic Choiceless Polynomial Time with counting ($\tilde{CPT}+C$) is capable of defining the corresponding canonical forms. By an application of the Immerman-Vardi theorem to the ordered canonical form defined in $\tilde{CPT}+C$, it follows that $\tilde{CPT}+C$ captures PTIME on logarithmic-sized fragments of τ -structures whenever τ is a vocabulary with relations of arity at most 2. For these structures, the result improves on a result by Blass, Gurevich, and Shelah [BGS99] that \tilde{CPT} can express all PTIME queries of fragments of size k where $k!$ is of polynomial size with respect to the size of the structure. Let us remark that logically defined canonical forms give rise to isomorphism tests. Thus, we cannot expect a $\tilde{CPT}+C$ -definable canonical form of the whole structure unless we find a polynomial-time isomorphism test, whose existence is in itself a great open problem in complexity theory.

Chapter 4 takes up the question of how PTIME’s computational power propagates to unordered structures by probing into capturing results on restricted graph classes. The restriction to graphs in this context is common and of little significance since all structures can be encoded as undirected graphs, so that a logic which captures PTIME on graphs also captures PTIME on all structures (cf. [EF99]). As such, the domain of graphs with its rich theory of graph classes offers a formidable test bed for structural properties which allow us to capture all of PTIME.

In this setting, we isolate a graph class which exhibits sufficient characteristics to be dealt with in fixed-point logic with counting. We show that the class of interval graphs admits FP+C-definable canonical forms and therefore FP+C captures PTIME on the class of (unordered) interval graphs. Given that there are already quite a few PTIME-capturing results by FP+C on restricted classes of graphs, such as planar graphs or graphs of bounded treewidth, the main contribution of this result are its new techniques for arriving at the canonical form. Unlike most graph classes considered in this context before, interval graphs are not closed under the operation of taking graph minors. Consequently, our methods do not use tree decompositions but instead employ a definable version of modular decompositions of interval graphs.

With a similar approach as for interval graphs, we also manage to show that FO with an operator for undirected graph reachability (STC+C) is sufficient to capture LOGSPACE on the class of proper interval graphs. However, we will argue that STC+C is too weak to capture LOGSPACE on the class of all interval graphs, going back to a fundamental inability of STC+C to handle trees [EI00]. The theme of non-capturing results is broadened by an exploration of graph classes on which FP+C fails to capture PTIME, among them chordal and bipartite graphs. Making reference to the standard notion of reducibility, we introduce the concept of non-capturing reductions which suggest a systematic study of logics' expressiveness in the genealogy of graph classes.

Taking our methods further, we complete Chapter 4 by proving that LOGSPACE possesses enough power to compute canonical forms of interval graphs. This puts the isomorphism problem of interval graphs into LOGSPACE, which we also show to be hard for this complexity class under FO-reductions. In this way, our tools developed for the logical domain make a contribution to pinpointing the complexity of the isomorphism problem of interval graphs.

Finally in Chapter 5, we take direct aim at the question for a logic capturing PTIME by augmenting FP+C with operators for calculating the ranks of matrices over finite fields. This gives rise to a variety of *rank logics*, the most expressive of which is FP+rk. As we argue there, the choice of rank operators is not arbitrary. As it turns out, FP+C's inability to express Cai, Fürer, and Immerman's query mentioned above derives from a more general inability to express queries which relate to linear algebra over finite fields [ABD07, Daw08]. We explore the expressiveness of rank operators by showing that they can define a variety of important basic queries, among them counting, reachability in undirected graphs, and also directed reachability when we allow the characteristics of finite fields to vary. We also show in detail that rank logics can express Cai, Fürer, and Immerman's query which separates FP+C from PTIME.

Taking our investigation further, we show that FP+C can already express ranks of matrices over the field of rationals \mathbb{Q} , which justifies why we introduce rank operators only for finite fields. Additionally, FP+C can compute determinants over \mathbb{Q} and even over finite fields, which means that determinants would not be an interesting addition to FP+C. Interestingly, though, we can show that FO together with all rank operators (FO+rk) can express these queries from linear algebra as well.

We then take the step to relate the expressiveness of rank logics to complexity theory. As with all known capturing results of complexity classes inside PTIME, we only show ours on the class of ordered structures. There, FO with rank operators over finite fields of prime characteristic p (FO+rk _{p}) captures MOD _{p} L, a complexity class based on non-deterministic logspace Turing machines. Furthermore, FO+rk captures the logspace counting hierarchy #LH on the class of ordered structures. In this way, we can show that rank logics have a robust link to established classes in standard complexity theory, which suggests further investigation in the context of descriptive complexity.

At the end, we show that the expressiveness of rank operators strictly increases when their arity is increased. This is true even if they are embedded in bounded-variable infinitary logic $L_{\infty\omega}^{\omega}$, which is very expressive in itself. This is in strong contrast to counting operators, whose

1 Introduction

expressiveness does not increase with growing arity in the presence of fixed-point operators. By showing this property for rank operators, we essentially clear all known hurdles for a logic to be a candidate for capturing PTIME. While we have no immediate reason to believe that $\text{FP}+\text{rk}$ really does capture PTIME, this shows that rank logics cannot be ignored if further progress towards a logic for PTIME is to be made.

1.3 About this thesis

Let us first make a remark about a presentational obstacle which we will repeatedly encounter in capturing results on unordered structures. As both mathematicians and computer scientists are typically used to having orders among all the objects they consider, it is not always immediately accessible why certain things might be hard to define in the absence of orderings. For example, our PTIME capturing result on interval graphs in Chapter 4 led us to the development of completely new methods for this graph class, even though several different polynomial-time algorithms for the required tasks have been around for a while. In this case, these methods could not be employed because they crucially rely on auxiliary orders which are available for PTIME computations. On many occasions we will therefore take out the time to explain why certain objects cannot be defined the *straight-forward way*.

The natural presumption of an ordering is causing comparable trouble in classical mathematics. Note that the concept of auxiliary orderings is closely related to the ability of making arbitrary choices since the one can be obtained from the other.⁴ Arbitrary choices appear equally often in mathematical proofs as in algorithms, typically indicated by phrases such as “fix some element x ” or “for each neighbor do something.” Our quest for a choiceless way to define properties of finite models is in that way analogous to attempts in mathematics to avoid the application of the axiom of choice. The axiom of choice is independent of the remaining axioms of Zermelo-Fraenkel set theory, which is why it is generally desirable to know when it is actually necessary. Still, wide areas of mathematics, such as most of analysis, generally assume the axiom of choice without taking special notice of its application. And even in set theory itself, proofs are often carried out with a pass at the axiom of choice for reasons of simplicity, even if a proof without an appeal to the axiom is possible (Chapter 7F of [Mos09] exemplifies this). In searching for capturing results on unordered structures, we are working directly at the faultlines between the presence and the absence of arbitrary choice in the realm of finite model theory, so we do not have the choice to brush over these differences. For the sake of a succinct exposition, we will make frequent appeals to the intuition of logical definability rather than the more common intuition of machine computability.

At the end of this introduction, the author would like to acknowledge that the research in several parts of this thesis was done in collaboration. Of course, the suitable attributions are also made when these results are introduced later on.

The definition and initial study of rank logics (Chapter 5) originated from collaboration with Bjarki Holm, Anuj Dawar, and Martin Grohe, and was presented at the 24th IEEE Symposium on Logic in Computer Science [DGHL09]. Parts of this work appear in Section 5.1 (definition of rank logics), Section 5.2 (counting, linear systems, deterministic and undirected reachability), Section 5.3 (ranks over \mathbb{Q} and determinants in $\text{FP}+\text{C}$), Section 5.4 (definability of the CFI-query in $\text{FO}+\text{rk}_2$), Section 5.5.1 ($\text{FO}+\text{rk}_p$ captures $\text{MOD}_p \text{L}$ on ordered structures), and Section 5.6 (arity hierarchy of rank operators).

The methods for handling interval graphs in LOGSPACE arose in collaboration with Johannes

⁴Given an ordering, there is a canonical way to make choices by always selecting the least of the elements in question. Conversely, an ordering is obtained from first choosing a smallest element, then a second smallest, and so on.

Köbler, Sebastian Kuhnert, and Oleg Verbitsky, and appeared at the 37th International Colloquium on Automata, Languages and Programming [KKLV10a]. Parts of this work appear in Section 4.4 (LOGSPACE-hardness of interval graph isomorphism, the modular tree for interval graphs, canonical labeling and isomorphism of interval graphs in LOGSPACE).

2 Notation, Definitions, Foundational Results

This chapter has the dual purpose of providing a succinct introduction to the basic notions which are relevant for this work and of serving as a reference point for fundamental concepts which would clutter the later presentation if they were to be explained there. Readers with a background in descriptive complexity may only want to skim through this chapter and otherwise rely on being referred back to here when the respective concepts become relevant.

Numbers. We write \mathbb{N} and \mathbb{N}_0 for the positive and non-negative integers, respectively. For $m, n \in \mathbb{N}_0$, let $[m, n] := \{\ell \in \mathbb{N}_0 \mid m \leq \ell \leq n\}$ be the *closed interval of integers from m to n* , and let $[n] := [1, n]$. If V is a set, then $\binom{V}{n}$ denotes the set of all n -element subsets of V . The set of rational numbers is denoted by \mathbb{Q} and the set of real numbers by \mathbb{R} . For prime $p \in \mathbb{N}$, GF_p denotes the finite field with p elements.

Variable tuples. Tuples of variables (v_1, \dots, v_k) are often denoted by \vec{v} and their length by $|\vec{v}|$. We also use the abbreviations $\vec{u} = \vec{v}$ meaning that $u_i = v_i$ for all $i \in [|\vec{u}|] = [|\vec{v}|]$; $\vec{u} \leq \vec{v}$ meaning that $u_i \leq v_i$ for all $i \in [|\vec{u}|] = [|\vec{v}|]$; and $\vec{u} \leq t$ meaning that $u_i \leq t$ for all $i \in [|\vec{u}|]$. When convenient, we sometimes write tuples like a string $v_1 v_2 \dots v_k$, and we denote the length of a string x by $|x|$.

Equivalence relations. A binary relation \sim on a set X is an *equivalence relation* if it is reflexive, symmetric, and transitive. For $x \in X$, the equivalence class $\{a \in X \mid x \sim a\}$ which contains x is usually denoted by $[x]$. The set of equivalence classes of \sim is denoted by X / \sim . When there is a (k -ary) relation R present on X , then the relation R_\sim on X / \sim defined by $[x_1] \dots [x_k] \in R_\sim \iff \exists y_1 \in [x_1] \dots \exists y_k \in [x_k] y_1 \dots y_k \in R$ is the *relation induced on X / \sim by R* .

Orders. A binary relation $<$ on a set X is a *strict partial order* if it is irreflexive and transitive.

Two elements x, y of a partially ordered set X are called *incomparable* if neither $x < y$ nor $y < x$. We call $<$ a *strict weak order* if it is a strict partial order, and in addition, incomparability is an equivalence relation, i.e., whenever x is incomparable to y and y is incomparable to z , then x and z are also incomparable. If x, y are incomparable with respect to a strict weak order $<$, then $x < z$ implies $y < z$ and $z < x$ implies $z < y$.

$<$ is a (*strict*) *linear order* if it is a strict partial order in which no two elements are incomparable. If $<$ defines a strict weak order on X and \sim is the equivalence relation defined by incomparability, then $<$ induces a strict linear order on X / \sim .

Lexicographic orderings. Given $k \in \mathbb{N}$ and a set X with a linear order $<$ defined on it, the *lexicographic order on k -tuples* from X^k is defined as follows: for $\vec{x}, \vec{y} \in X^k$, let $\vec{x} <_{lex} \vec{y}$ if and only if there is $i \in [k]$ so that $x_i < y_i$ and $x_j = y_j$ for all $j < i$. It is easy to see that $<_{lex}$ defines a linear order on X^k .

Lexicographic orderings are a pervasive concept in this work and appear in a variety of other contexts. First of all, the lexicographic order on tuples carries over to strings of elements from X .¹ Multisets of elements from X are ordered lexicographically by identifying each multiset M with the string which contains the elements of M in ascending order. For the subsets of an ordered

¹For strings, we have to extend the lexicographic order of tuples in order to handle strings of different lengths. Classically, this is done by letting $s = s_1 \dots s_k <_{lex} s' = s'_1 \dots s'_\ell$ if and only if s is a proper prefix of s' or $s_1 \dots s_{\min\{k, \ell\}} <_{lex} s'_1 \dots s'_{\min\{k, \ell\}}$. In most of our applications, it seems more natural to first compare the length of strings and only consider their lexicographic order when their lengths match. The precise definition is not important for us.

2 Notation, Definitions, Foundational Results

set $X = \{x_1, \dots, x_k\}$ we usually find it more intuitive to define their lexicographically order as follows: with each $U \subseteq X$, associate the $\{0, 1\}$ -string s^U of length $|X|$ for which $s_i^U = 1$ if and only if the i^{th} element of X is contained in U . For $U, V \subseteq X$, we then set $U < V$ if and only if $s^U <_{\text{lex}} s^V$, where $<_{\text{lex}}$ is based on the natural ordering $0 < 1$. By way of a similar identification of structures with strings we will define a lexicographic order of ordered structures in Section 2.4.1.

2.1 Graphs

We generally assume graphs to be *finite, simple, and undirected* unless explicitly stated otherwise. Let $G = (V, E)$ be a graph with vertex set V and edge set E . E is usually viewed as a binary relation. Sometimes, we also find it convenient to view undirected edges e as sets containing their two endpoints, as in $e = \{u, v\} \subseteq V$. Out of convenience, we often write $uv \in E$ to mean that u and v are joined by an edge. For *isomorphic graphs* G and H we write $G \cong H$. The *complement of a graph* $G = (V, E)$ is the graph $G^c := (V, \binom{V}{2} \setminus E)$.

For $W \subseteq V$ a set of vertices, $G[W]$ denotes the *induced subgraph* of G on W . The *(closed) neighborhood* of a vertex $v \in V$, denoted $N[v]$, is the set of vertices adjacent to v under E , plus v itself. Two vertices u, v are *connected twins* if $N[u] = N[v]$.

For disjoint $A, B \subseteq V$ we say that A and B are *connected as sets* if there is an edge from a vertex in A to a vertex in B . When we deal with directed graphs and $u, v \in V$, then u is an *incoming neighbor* of v if $uv \in E$. Similarly, u is an *outgoing neighbor* of v if $vu \in E$.

Cliques and independent sets. A subset $W \subseteq V$ is called a *clique* of G if $G[W]$ is a complete graph. $W \subseteq V$ is called an *independent set* if $G[W]$ does not contain any edges. A clique W of G is *maximal* if it is inclusion-maximal as a clique in G , i.e., if no vertex $v \in V \setminus W$ can be added to W so that $W \cup \{v\}$ is a clique of G . Since maximal cliques are central to the constructions in Chapter 4, they will often just be called *max cliques*.

Paths. A *simple path* P in G is a sequence $\overrightarrow{v_1, v_2, \dots, v_k}$ of distinct vertices so that $v_i v_{i+1} \in E$ for all $i \in [k-1]$. The *length of a path* is the number of edges it contains; so P has length $k-1$. Non-simple paths may also contain vertices and edges repeatedly. Unless explicitly stated otherwise, the paths we consider are always simple. A graph is called *connected* if there is a path between any two vertices.

Cycles. A *cycle* C in G is a sequence $\overrightarrow{v_0, v_1, v_2, \dots, v_k}$ of vertices with $k \geq 3$ so that both $\overrightarrow{v_0, \dots, v_{k-1}}$ and $\overrightarrow{v_1, \dots, v_k}$ are simple paths and $v_0 = v_k$. The *length of C* is k , which is both the number of edges and vertices in C . A *chord* of C is an edge between non-consecutive vertices in C .

Graph modouts. If \sim is an equivalence relation on V , then $G/\sim = (V/\sim, E_\sim)$ is the graph on the set of equivalence classes $V/\sim = \{[v] \mid v \in V\}$ for which $[u][v] \in E_\sim$ if and only if $[u] \neq [v]$ and there is an edge between the sets $[u]$ and $[v]$ in G . Thus E_\sim is the irreflexive part of the relation E induces on V/\sim .

Most of the time, we will only consider graph modouts where each pair of equivalence classes $[u]$ and $[v]$ is either completely connected or completely disconnected in G . In this case, each equivalence class $[u]$ is a *module of G* : $M \subseteq V$ is a module of G if each $v \in V \setminus M$ is either completely connected or completely disconnected to all vertices in M . When all equivalence classes are modules, G/\sim does not lose any information about edges between the sets $[u]$ and $[v]$, which can be used to recover G from G/\sim and the induced subgraphs of G on all the equivalence classes. This idea is employed in Chapter 3 and developed in greater detail in Chapter 4.

2.1.1 Colored graphs

A *colored graph* is a graph $G = (V, E)$ together with a map $c : V \rightarrow \mathbb{N}_0$ called a *coloring of G* . For $v \in V$, the value $c(v)$ is called the *color of v* . Occasionally, we will consider colorings $c : V \rightarrow Z$ of G with a range Z different from \mathbb{N}_0 . Importantly, though, we always require *colors to be ordered*, meaning that the range Z has to be linearly ordered. Two colored graphs G, H with colorings c_G and c_H , respectively, are said to be *color isomorphic* if there is a graph isomorphism $\varphi : G \rightarrow H$ which preserves the colorings, i.e. $c_G(v) = c_H(\varphi(v))$ for all vertices v of G .

Unary relations. There is an alternative view of colorings which is based on relational structures as they are introduced in Section 2.3. Let $\tau = (U_1, \dots, U_k, R_1, \dots, R_\ell)$ be a finite relational vocabulary and let U_1, \dots, U_k be the unary relations in τ . The U_i induce a coloring of any τ -structure $\mathcal{A} = (A, (U_i^{\mathcal{A}})_{i \in [k]}, (R_i^{\mathcal{A}})_{i \in [\ell]})$ by setting $c(v) := \sum_{i=1}^k 2^{k-i} \cdot \delta_{U_i v}$ for any $v \in A$, where $\delta_{U_i v} = 1$ if $v \in U_i$ and $\delta_{U_i v} = 0$ if $v \notin U_i$. Notice that the colors in $[0, 2^k - 1]$ correspond to the subsets of $\{U_1, \dots, U_k\}$ in lexicographic order. Bearing in mind this correspondence, we sometimes speak of unary relations as colors.

Ordered partitions. Given a coloring c of a graph $G = (V, E)$, “being of the same color” is an equivalence relation on V . Its (monochromatic) equivalence classes are also called *color classes*. Using the linear order on the colors we obtain a linear order of G ’s color classes P_1, \dots, P_k . In this way, the coloring c induces the *ordered partition* $\mathcal{P} := (P_1, \dots, P_k)$ of V . The sets P_1, \dots, P_k are called *cells of \mathcal{P}* . In many of our applications, it is not important what names colors are given, but only what ordered partitions they induce. This is the case, for example, in the Weisfeiler-Lehman color refinement procedure (see Section 3.2.3). For this reason, we often loosely identify the concepts of colorings and ordered partitions.

Refinement of colorings. Let $G = (V, E)$ be a graph, let c, d be colorings, and let $\mathcal{P}_c = (P_1, \dots, P_k)$ and $\mathcal{Q}_d = (Q_1, \dots, Q_\ell)$ be the corresponding ordered partitions. c is called a *refinement of d* if for each $P_i \in \mathcal{P}_c$ there is a cell $Q_j \in \mathcal{Q}_d$ such that $P_i \subseteq Q_j$. The *refinement of c by d* is the coloring $c \times d : V \rightarrow c(V) \times d(V)$ given by $v \mapsto (c(v), d(v))$. It is clear that $c \times d$ is a refinement both of c and of d . The colors in $c(V) \times d(V)$ are ordered lexicographically. Thus, $c \times d$ and $d \times c$ have the same color classes, but their order is usually different. It is easy to see that the ordered partition induced by $c \times d$ is $(P_1 \cap Q_1, P_1 \cap Q_2, \dots, P_1 \cap Q_\ell, P_2 \cap Q_1, \dots, P_k \cap Q_\ell)$ with the empty sets deleted. Thus, we may consider color refinement purely as a process of ordered partitions.

2.1.2 Intersection graphs

Let $\mathcal{O} = \{o_1, \dots, o_k\}$ be a set of sets. The *intersection graph $G_{\mathcal{O}}$* has \mathcal{O} as its vertex set and edges joining o_i and o_j whenever $i \neq j$ and $o_i \cap o_j \neq \emptyset$. \mathcal{O} is called an *intersection model* for a graph G if $G \cong G_{\mathcal{O}}$. Notice that any finite graph has an intersection model of sets from \mathbb{N} , one of which can be directly inferred from enumerating the graph’s max cliques.

This is not the case, however, if we suitably restrict intersection models so that they may only contain sets of certain types. An *intersection graph class \mathcal{G}* is a class of graphs for which there is a set \mathcal{Z} so that $G \in \mathcal{G}$ if and only if G has an intersection model $\mathcal{O} \subseteq \mathcal{Z}$. For example, line graphs, chordal graphs, and interval graphs are intersection graph classes. For a wealth of intersection graph classes, confer [BLS99].

Lemma 2.1.1. *Intersection graph classes are closed under taking induced subgraphs.*

Proof. If \mathcal{Y} is an intersection graph class, $G = (V, E) \in \mathcal{Y}$, and U is any subset of V , then $G[U]$ is just the intersection graph of the objects in U . Therefore, $G[U] \in \mathcal{Y}$. \square

2.1.3 Graph classes

The graph classes introduced here are all well-established and have been widely investigated. More information about them can be found, for example, in [BLS99] or [Die06]. While bipartite graphs and trees frequently appear throughout this thesis, the other graph classes mainly just play a role in Chapter 4.

Bipartite and split graphs. A graph G is *bipartite* if its vertex set can be partitioned into two sets U and V which are independent sets in G . We write $G = (U \dot{\cup} V, E)$ to emphasize on the partition. G is a *split graph* if its vertex set can be partitioned into two sets U and V so that U is a clique and V is an independent set. As for bipartite graphs, we write $G = (U \dot{\cup} V, E)$ to emphasize on the partition.

Trees. An undirected graph G which does not contain any cycle as a subgraph is called a *forest*. If G is also connected, then it is a *tree*. A tree $T = (V, E)$ together with one of its vertices $r \in V$ is called a *rooted tree* and r is called its *root*. In rooted trees, we usually assume all edges E to be directed away from the root. Any vertex which does not have an outgoing edge is called a *leaf* of T . The *height* of T is the maximal length of a path from the root to a leaf.

Interval graphs. Interval graphs are the intersection graphs of intervals. This means a graph $G = (V, E)$ is an interval graph if there exists a set \mathcal{I} of intervals from \mathbb{N} and a bijection $\rho : V \rightarrow \mathcal{I}$ so that u, v are adjacent in G if and only if $\rho(u) \cap \rho(v) \neq \emptyset$. We refer to Section 4.1 for concepts related to interval graphs and a discussion of their properties and algorithmic aspects.

Chordal graphs. A graph is called *chordal* if all its induced cycles are of length 3. In other words, a graph is chordal if it does not contain a chordless cycle of length ≥ 4 . It is easy to show that every interval graph is chordal. Chordal graphs can alternatively be characterized by the property that its maximal cliques can be arranged in a tree T , so that for every vertex of the graph the set of max cliques containing it is connected in T (cf. [Die06]). Using the concept of *tree decompositions of a graph* (see [Die06]), a graph is chordal if and only if it has a tree decomposition into its max cliques. By rephrasing this characterization once again, we see that the chordal graphs are precisely the intersection graphs of subtrees of a tree.

Comparability graphs. A graph $G = (V, E)$ is called a *comparability graph* if there exists a strict partial ordering $<$ of its vertex set V so that $uv \in E$ if and only if u, v are comparable with respect to $<$. Alternatively, G is a comparability graph if and only if its edges can be oriented in such a way that the corresponding binary relation is transitive. A graph is called a *co-comparability graph* if its complement is a comparability graph. It is a well-known fact that every interval graph is a co-comparability graph. In fact, a graph is an interval graph if and only if it is a chordal co-comparability graph [GH64, Gol04].

2.1.4 Graph reachability

The *graph reachability problem* is to decide whether in a graph G , there is a path connecting given vertices s and t . The problem is also known as the graph accessibility problem (GAP) or s, t -connectivity problem.

In a recent break-through, Reingold showed that undirected graph reachability can be decided in deterministic logarithmic space ([Rei05, Rei08a], also see [AB09, Section 21.4]). Reingold's theorem will an important basis for all our LOGSPACE-related results about interval graphs in Section 4.5.

Reingold's Theorem 2.1.2. Given an undirected graph G and vertices s, t , deciding whether there is a path from s to t is in LOGSPACE. ■

When working with logical frameworks, we take an alternative view on graph reachability

problems. For a (directed or undirected) graph $G = (V, E)$, the *transitive closure of E* is the smallest relation $E^* \subseteq V^2$ with $E \subseteq E^*$ and for all $u, v, w \in V$: $uv, vw \in E^* \Rightarrow uw \in E^*$. E^* can be obtained from E by inductively adding edges wherever transitivity is violated until the relation is transitive. It is easy to see that for $u \neq v$ we have $uv \in E^*$ if and only if there is a path from u to v in G .

2.2 Complexity classes

We assume familiarity with the basic notions of complexity theory as presented in [GJ79]; good recent accounts are [Koz06] and [AB09]. In this section we briefly review the most important concepts that are relevant for this work.

All Turing machines we consider are *total*, which means that they halt on all inputs. For simplicity, we assume $\{0, 1\}$ to be the common alphabet used by all our machines.

PTIME and NP. A *polynomial-time Turing machine M* is a Turing machine for which there is a constant $c \in \mathbb{N}$ so that all computation paths of M terminate within $O(|x|^c)$ steps on all input strings x . A problem (i.e., class of finite strings) is *polynomial-time decidable* if it is decidable by some deterministic polynomial-time Turing machine. We denote the class of all polynomial-time decidable problems by PTIME. Observe that in complexity theory, this class is typically just called P. We choose the longer acronym following an apparent tradition in the field of descriptive complexity, and also in order to avoid confusion with letters abbreviating logics. Classically, though, we write NP for the class of all problems decidable by a non-deterministic polynomial-time Turing machine.

LOGSPACE and NLOGSPACE. A *logspace Turing machine M* is a Turing machine whose input tape is read-only, and which uses at most $O(\log|x|)$ work tape cells for any computation path and for any input string x . LOGSPACE is the class of all problems which are decidable by a deterministic logspace Turing machine. The long name of the class is again non-standard, and in some places we also find it convenient to use its more classical name L. NLOGSPACE is the class of all problems which are decidable by a non-deterministic logspace Turing machine.

Configurations. A *configuration of a Turing machine M* consists of a state, head positions, and worktape contents of M . In particular, the input tape contents of logspace-bounded machines are not part of its configurations, however, the head position on this tape is. Thus, if M is $O(\log n)$ -space bounded, then there are at most n^d configurations for some constant d and for sufficiently large input length n . We may generally assume that the Turing machines we consider have a unique start and a unique accept configuration.

The *configuration graph G_M* of a machine M with respect to an input x is defined over the vertex set C_M of possible configurations of M over inputs of length $|x|$ and has a directed edge from configurations a to b if b is a valid successor configuration of a in the computation of M over input x . By augmenting the machines we consider with *clocks* that terminate the computation after the maximum computation time is exceeded, we make sure that the configuration graph is acyclic everywhere, also in those places which are not reachable from the start configuration.²

Ebbinghaus and Flum [EF99] show how to encode and define the configuration graph of a given logspace Turing machine M over an ordered domain (see Lemma 5.5.7). We will use their result in Section 5.5 to show capturing results on ordered structures.

Transducers. A *logspace transducer T* is a deterministic logspace Turing machine which has, in addition to its read-only input tape and its worktapes, one write-only output tape. Logspace transducers are the machines underlying the classical concept of *logspace many-one reducibility*.

²We will only consider configuration graphs of logspace-bounded Turing machines, which may clock themselves without exceeding their $O(\log n)$ space bound.

Notice that while T 's worktape usage is logarithmically bounded, the string on its output tape may be polynomial in length, where the polynomial restriction follows from the polynomial bound on the number of T 's configurations and our assumption that T is total. It is well-known that logspace transducers can be concatenated, i.e., if one logspace transducer uses the output of another as its input, these two transducers can be combined into a single logspace transducer (cf. [Koz06]).

Duality between decision and computation problems. There is a duality between logspace decision and computation problems which we will use implicitly in some places. If T is a logspace transducer whose output size is bounded by n^d , let M be the Turing machine which takes as input a string x and a number $i \leq n^d$, runs $T(x)$, and accepts if and only if the i^{th} bit of T 's output is 1. As there is hardly any additional overhead, M is also logspace-bounded. Conversely, we may recover a logspace transducer T' from M by simulating $M(x, i)$ for i from 1 up to n^d and outputting each of the results on the output tape. Clearly, T' is equivalent to T . Thus, we may consider logspace decision and computation problems to be equivalent. The same kind of duality is true of polynomial-time decision and computation problems.

2.3 Logics

We assume basic knowledge in logic, particularly of first-order logic FO (cf. [EF99]). This section will introduce a variety of well-known extensions of FO, augment these logics with counting operators, treat logical reductions, and explain what it means for a logic to capture a complexity class. Detailed discussions of all the logics and the other topics mentioned here can be found in [EF99, GKL⁺07, Imm99, Ott97].

Relational structures. A *vocabulary* τ is a finite sequence of relation symbols (R_1, \dots, R_k) in which each R_i has a fixed arity $a_i \in \mathbb{N}$. τ is called a vocabulary of arity at most r if each $R \in \tau$ has arity at most r . A τ -*structure* $\mathcal{A} = (U(\mathcal{A}), R_1^{\mathcal{A}}, \dots, R_k^{\mathcal{A}})$ is a non-empty set $U(\mathcal{A})$, called the *universe* of \mathcal{A} , together with relations $R_i^{\mathcal{A}} \subseteq U(\mathcal{A})^{a_i}$ *interpreting* $R_i \in \tau$ for $1 \leq i \leq k$. We let $|\mathcal{A}| := |U(\mathcal{A})|$.

With two exceptions, we always assume that *all structures are finite*, i.e., $|\mathcal{A}| \in \mathbb{N}$. One exception are the states of $\tilde{\text{CPT}}$ -programs introduced in Section 3.2.1, the other exception are the two-sorted structures \mathcal{A}^+ to which we adjoin the number sort \mathbb{N} .

Formulas. Given a logic L and a vocabulary τ , $L[\tau]$ denotes the set of τ -formulas of L . For a formula $\varphi \in L[\tau]$ we write $\varphi(\vec{x})$ to indicate that all of φ 's free variables are among \vec{x} . Given a τ -structure \mathcal{A} , we write $\mathcal{A} \models_L \varphi[\vec{a}]$ if $\vec{a} \in U(\mathcal{A})^{|\vec{x}|}$ and \mathcal{A} satisfies φ under the assignment of a_i to x_i for every $i \in [|\vec{x}|]$. If it is clear from the context that \vec{a} is an assignment for \vec{x} , then we simply write $\mathcal{A} \models_L \varphi[\vec{a}]$. Also, the logic L is usually clear from the context, in which case we omit the subscript to the satisfaction relation.

Queries and expressiveness. A τ -formula $\varphi(\vec{x})$ with $|\vec{x}| = k$ defines a k -*ary query* that associates with any τ -structure \mathcal{A} the set of k -tuples \vec{a} from $U(\mathcal{A})$ for which $\mathcal{A} \models \varphi[\vec{a}]$. We denote this set by $\varphi^{\mathcal{A}}[\cdot]$. Similarly, for example, $\varphi^{\mathcal{A}}[a_1, \dots, a_{k-1}, \cdot]$ denotes the subset of elements $a_k \in U(\mathcal{A})$ for which $\mathcal{A} \models \varphi[a_1, \dots, a_k]$. A τ -sentence φ defines a *query of τ -structures* which is associated with the set of all τ -structures \mathcal{A} for which $\mathcal{A} \models \varphi$.

We say that a logic L_1 is (*at least*) *as expressive* as a logic L_2 , and write $L_2 \leq L_1$, if every query definable in L_2 is also definable in L_1 . We write $L_1 \equiv L_2$ if $L_1 \leq L_2$ and $L_2 \leq L_1$.

Isomorphism of structures. Two τ -structures \mathcal{A}, \mathcal{B} are *isomorphic*, and we write $\mathcal{A} \cong \mathcal{B}$, if there is a bijection $f : U(\mathcal{A}) \rightarrow U(\mathcal{B})$ which preserves all the relations in τ . That means for all $R \in \tau$, we have $x_1 \dots x_a \in R^{\mathcal{A}}$ if and only if $f(x_1) \dots f(x_a) \in R^{\mathcal{B}}$, where a is the arity of R . We always assume all classes of structures \mathcal{K} to be *closed under isomorphism*, i.e., whenever $\mathcal{A} \in \mathcal{K}$ and $\mathcal{B} \cong \mathcal{A}$, then also $\mathcal{B} \in \mathcal{K}$.

If two τ -structures \mathcal{A}, \mathcal{B} are isomorphic and they are both linearly ordered with respect to some relation $\leq \in \tau$, then we emphasize the presence of these orders by saying that \mathcal{A} and \mathcal{B} are *order isomorphic*. It is very easy to decide whether two ordered structures are order isomorphic since we only have to consider the unique map which preserves the ordering.

2.3.1 Capturing complexity classes

The fundamental motivation for this work is the question for precise logical characterizations of complexity classes. In principle, we want to find logics L and complexity classes K so that any L -definable query is decidable in K and any K -decidable query is definable in L . Then we say that L captures C . However, since the complexity classes we wish to consider are defined in terms of Turing machines, which work over strings, and our logics define queries on general structures, we have to be more precise about what we mean for a logic and a complexity class to define the same queries.

Ordered structures. For a vocabulary τ with $\leq \in \tau$, we call a τ -structure \mathcal{A} an *ordered structure* if \mathcal{A} interprets \leq as a total order of its universe. We often identify \mathcal{A} 's linearly ordered universe with $[0, |\mathcal{A}| - 1] \subset \mathbb{N}_0$.

Encoding of ordered structures. Let $\tau = (R_1, \dots, R_k)$ be a vocabulary with $\leq \in \tau$ and let a_i denote the arity of R_i . We associate with any ordered τ -structure \mathcal{A} the $\{0, 1\}$ -string $\text{enc}(\mathcal{A}) = s^1 s^2 \dots s^k$, the concatenation of strings s^1 through s^k , where $|s^i| = |U(\mathcal{A})|^{a_i}$ and $s_j^i = 1$ if and only if R_i holds for the j^{th} tuple in the lexicographic ordering of a_i -tuples from $U(\mathcal{A})$. We call $\text{enc}(\mathcal{A})$ the *canonical encoding of the ordered structure \mathcal{A}* .³

It is immediately clear from the definition that two ordered τ -structures \mathcal{A}, \mathcal{B} are order isomorphic if and only if $\text{enc}(\mathcal{A}) = \text{enc}(\mathcal{B})$. Of course, our encoding is not the only way to preprocess ordered structures for the use in Turing machines, but $\text{enc}(\mathcal{A})$ conveniently double-acts as the basis for lexicographic comparisons of ordered structures in Section 2.4.1. A detailed discussion of a slightly different encoding can be found in [EF99, Section 7.2.2].

Deciding classes of ordered structures. If K is a class of ordered τ -structures, we say that a Turing machine M decides K if for any ordered τ -structure \mathcal{A} ,

$$M(\text{enc}(\mathcal{A})) \begin{cases} \text{accepts} & \text{if } \mathcal{A} \in K, \\ \text{rejects} & \text{if } \mathcal{A} \notin K. \end{cases}$$

For fixed vocabulary τ , deciding whether a given string $x \in \{0, 1\}^*$ is a valid encoding of a τ -structure is a simple matter of examining the length of x . Thus, M can be turned into a machine that decides $\{\text{enc}(\mathcal{A}) \mid \mathcal{A} \in K\} \subseteq \{0, 1\}^*$. For a complexity class C we write $K \in C$ to mean $\{\text{enc}(\mathcal{A}) \mid \mathcal{A} \in K\} \in C$.

Definition 2.3.1 (Capturing on ordered structures). Given a complexity class C , a logic L *captures C on ordered structures* if for any vocabulary τ with $\leq \in \tau$ and any class K of ordered τ -structures, $K \in C$ if and only if there is a sentence ϕ_K of $L[\tau]$ that defines K .

General capturing. Let τ be any finite vocabulary. If K is a class of τ -structures, then

$$K_{\leq} := \{(\mathcal{A}, \leq) \mid \mathcal{A} \in K, \leq \text{ is a linear order of } U(\mathcal{A})\}$$

is called the *class of ordered representations* of structures in K . Using K_{\leq} , we can now describe what it means to capture a complexity class in full generality. [EF99] refers to this concept as

³It might seem superfluous that we also encode the ordering relation $\leq \in \tau$. We see this as a simple way to ensure that the size of the structure can be inferred from the encoding even when there are no other relations present besides \leq . When there are other relations, though, the encoding of \leq can also be suppressed.

strongly capturing a complexity class.

Definition 2.3.2 (Capturing complexity classes). Let L be a logic and let C be a complexity class. We say that L *captures* C if for any vocabulary τ and any class K of τ -structures, $K_{\leq} \in C$ if and only if K is definable in $L[\tau]$.

Capturing on restricted classes of structures. If \mathcal{K} is a class of structures, then we write $\mathcal{K}[\tau]$ to denote the set of τ -structures in \mathcal{K} . We always assume classes of structures to be *closed under isomorphism*.

Definition 2.3.3 (Capturing on a structure class \mathcal{K}). Let L be a logic, C be a complexity class, and \mathcal{K} be a class of structures. We say that L *captures* C *on* \mathcal{K} if for any vocabulary τ and any class $K \subseteq \mathcal{K}[\tau]$, there is a class $H \in C$ with $K_{\leq} = H \cap \mathcal{K}[\tau]$ if and only if there is a sentence $\varphi \in L[\tau]$ so that for all $\mathcal{A} \in \mathcal{K}[\tau]$ we have $\mathcal{A} \models \varphi \leftrightarrow \mathcal{A} \in K$.

If the class \mathcal{K} is both definable in L and decidable in C , then the condition of Definition 2.3.3 can be simplified: for any vocabulary τ and any $K \subseteq \mathcal{K}[\tau]$, $K \in C$ if and only if K is definable in L . All capturing results on restricted structure classes in this thesis are actually of this simpler type.

The notion of capturing C on ordered structures as in Definition 2.3.1 can essentially be seen through the lens of Definition 2.3.3 as capturing C on \mathcal{O} , where \mathcal{O} is the class of all structures for which some relation defines a linear order of the universe. Similarly, if \mathcal{K} denotes the class of all structures, then Definition 2.3.3 is equivalent to the general concept of capturing complexity classes given in Definition 2.3.2. We sometimes mention the *class of all structures* explicitly to highlight the generality of capturing results.

Abstract notions of capturing. The literature knows further notions of what it means for a complexity class to be captured by a logic. In particular, we have not specified precisely what constitutes a *logic* in Definitions 2.3.1 to 2.3.3. This is not a small matter as there are pathological examples of “logics” that capture PTIME by essentially declaring Turing machines to be logical sentences (cf. [Gur88]). Nash, Remmel, and Vianu [NRV05] discuss minimum requirements on candidate logics for capturing PTIME in a broad sense. For such a logic L , the set of sentences in $L[\tau]$ should be decidable for each vocabulary τ and all $\varphi \in L[\tau]$ should define isomorphism-invariant properties of τ -structures. All the logics we consider satisfy these requirements. Also, following [EF99], we say that a logic L *effectively captures* a complexity class C if, in addition to capturing C in the sense of Definition 2.3.2, there is a recursive procedure which associates with each $\varphi \in L[\tau]$ a Turing machine M and a time bound n^d so that M decides $\{\mathcal{A} \mid \mathcal{A} \models \varphi\}_{\leq}$ in time n^d . All our capturing results are also effective in this sense.

2.3.2 Plain logics

Extending logics. Most logics that we consider are *extensions* of first-order logic, meaning that all terms and formulas of FO are also terms and formulas of these logics. In fact, we define most logics in this way, too. Given a term and formula formation calculus \mathcal{C} for a logic L , we say that L' *is the extension of* L *with a rule* \mathcal{R} if the terms and formulas of L' are obtained through any sequence of rules from $\mathcal{C} \cup \{\mathcal{R}\}$. The semantics of \mathcal{R} must be given separately and the semantics of the rules in \mathcal{C} remain the same. In this process, we generally assume that all rules are *blind towards the logic*. Thus, if any rule \mathcal{S} requires formulas and terms as input, \mathcal{S} may not restrict these terms and formulas to be formed only with respect to a subset of rules from $\mathcal{C} \cup \{\mathcal{R}\}$. In particular, if we extend FO with a new operator o , we are allowed to nest applications of o , apply o to arbitrarily defined relations in this new logic, and use o together with all the first-order connectives. Examples for this are the logics DTC, STC, and TC which are all defined as extensions of FO by operators for deciding certain graph reachability queries. Complete examples

of the extension of first-order calculi with new rules for many of the logics we consider here are given in [EF99].

FO with majority quantifiers FO+M. If \mathcal{A} is a finite structure and φ is a formula in which the variable x possibly occurs free, then $\mathcal{A} \models Mx \varphi$ if and only if φ holds for the majority of $x \in U(\mathcal{A})$, i.e. $|\{a \in U(\mathcal{A}) \mid \mathcal{A} \models \varphi[\frac{a}{x}]\}| > \frac{1}{2}|U(\mathcal{A})|$. M is called *majority quantifier*. The extension of FO with majority quantifiers is denoted by FO+M. FO+M captures the circuit complexity class of FO-uniform TC⁰ by a result of Barrington, Immerman, and Straubing [BIS90].

Transitive Closure Logic TC. Let R be a binary relation on a set A , i.e., $R \subseteq A^2$. We view (A, R) as a directed graph and say that $(x, y) \in A^2$ is in the *transitive closure of R* if there is an R -path of length at least 1 from x to y . Now let τ be any vocabulary and let φ be a formula in which the distinct variables \vec{x} and \vec{y} with $|\vec{x}| = |\vec{y}|$ possibly occur free. Over any τ -structure \mathcal{A} , $\varphi^{\mathcal{A}}$ defines a directed edge relation on $U(\mathcal{A})^{|\vec{x}|}$ with respect to \vec{x} and \vec{y} . We introduce the *transitive closure operator* tc of arity $k := |\vec{x}|$ by setting

$$\mathcal{A} \models (tc_{\vec{x}, \vec{y}} \varphi)(\vec{a}, \vec{b}) \iff (\vec{a}^{\mathcal{A}}, \vec{b}^{\mathcal{A}}) \in U(\mathcal{A})^k \times U(\mathcal{A})^k \text{ is in the transitive closure of } \varphi^{\mathcal{A}}.$$

The free variables of $(tc_{\vec{x}, \vec{y}} \varphi)(\vec{a}, \vec{b})$ are $\{\vec{a}, \vec{b}\} \cup (\text{free}(\varphi) \setminus \{\vec{x}, \vec{y}\})$. *Transitive closure logic* TC is defined as the extension of FO with the tc -operators of all arities.

Theorem 2.3.4 (Immerman [Imm99]). *On ordered structures, TC captures NLOGSPACE.* ■

Symmetric Transitive Closure Logic STC. Let R be a symmetric binary relation on a set A . Viewing (A, R) as an undirected graph we say that $(x, y) \in A^2$ is in the *symmetric transitive closure of R* if there is an R -path of length at least 1 from x to y . If $G = (V, E)$ is a directed graph, then $G_{sym} = (V, E_{sym})$ denotes the *undirected graph underlying G* , which is obtained from G by forgetting about the edge directions. For a τ -formula φ and variables \vec{x}, \vec{y} with $k := |\vec{x}| = |\vec{y}|$, set

$$\mathcal{A} \models (stc_{\vec{x}, \vec{y}} \varphi)(\vec{a}, \vec{b}) \iff (\vec{a}^{\mathcal{A}}, \vec{b}^{\mathcal{A}}) \in U(\mathcal{A})^k \times U(\mathcal{A})^k \text{ is in the symmetric transitive closure of the undirected graph underlying } \varphi^{\mathcal{A}}.$$

The free variables are defined in the same way as for tc . *Symmetric transitive closure logic* STC is the extension of FO with the stc -operators of all arities. The symmetrization of the directed edge relation given by φ is easily definable. Setting $\varphi_{sym} := \varphi \vee \exists \vec{a} = \vec{x}, \vec{b} = \vec{y} \exists \vec{x} = \vec{b}, \vec{y} = \vec{a} \varphi$, we have $(stc_{\vec{x}, \vec{y}} \varphi)(\vec{a}, \vec{b}) \leftrightarrow (tc_{\vec{x}, \vec{y}} \varphi_{sym})(\vec{a}, \vec{b})$, and hence

Lemma 2.3.5. $STC \leq TC$. □

By Reingold's Theorem 2.1.2 the data complexity of STC-sentences is in LOGSPACE.

Remark 2.3.6. STC has been investigated considerably less than TC and DTC. Perhaps because of this, it is still an open question whether TC is strictly more expressive than STC, even though the complexity of directed and undirected reachability queries could be separated in other logical frameworks. Unlike undirected reachability, for example, directed reachability is not expressible in symmetric datalog [ELT08]. Also, undirected reachability is expressible in existential monadic second-order logic, but directed reachability is not [AF90]. At least this implies that arity-1 TC is strictly more expressive than arity-1 STC.

Deterministic Transitive Closure Logic DTC. Let $G = (V, E)$ be a directed graph. We call $G_d = (V, E_d)$ the *deterministic part of G* and let $uv \in E_d$ if and only if $uv \in E$ and u has out-degree 1 in G . In other words, G_d is obtained from G by deleting the outgoing edges from all those vertices which have multiple outgoing edges. The reachability problem in G_d of determining whether there is a path between vertices s and t is called the *deterministic graph reachability*

2 Notation, Definitions, Foundational Results

problem. For a τ -formula φ and variables \vec{x}, \vec{y} with $k := |\vec{x}| = |\vec{y}|$, set

$$\mathcal{A} \models (\text{dte}_{\vec{x}, \vec{y}} \varphi)(\vec{a}, \vec{b}) \iff (\vec{a}^A, \vec{b}^A) \in U(\mathcal{A})^k \times U(\mathcal{A})^k \text{ is in the transitive closure of the deterministic part of } \varphi^A.$$

The free variables are defined in the same way as for tc. *Deterministic transitive closure logic* DTC is the extension of FO with the dte-operators of all arities. Of course, the deterministic part of the directed graph defined by φ is easily FO-definable so that $\text{DTC} \leq \text{TC}$. What is more, deterministic reachability can also be reduced to undirected graph reachability. The result is folklore.

Lemma 2.3.7. *Deterministic reachability FO-reduces to symmetric reachability. Therefore, $\text{DTC} \leq \text{STC}$.*

Proof. Let $G = (V, E)$ be a directed graph which has out-degree at most 1 and let s, t be vertices. Let G^* be the *undirected* graph obtained from G by removing any outgoing edge from t and then forgetting about the edge directions. Clearly, if there is a directed path from s to t in G , the same path connects s and t in G^* . Conversely, if P is a simple undirected s, t -path in G^* , following P backwards from t to s we always use edges from G in the reverse direction since all vertices have out-degree at most 1 and t has no outgoing edge. Thus, there is a path from s to t in G if and only if there is an undirected s, t -path in G^* . Of course, the graph G^* is first-order definable over G , so the result follows. \square

Theorem 2.3.8 (Immerman [Imm99]). *On ordered structures, DTC captures LOGSPACE.* \blacksquare

Corollary 2.3.9. *STC captures LOGSPACE on ordered structures.*

Proof. By Lemma 2.3.7 and Reingold's Theorem 2.1.2. \square

FP. *Inflationary fixed-point logic* FP is the extension of FO by a fixed-point operator with inflationary semantics, which is defined as follows. Let \mathcal{A} be a τ -structure, let $X \notin \tau$ be a *relation variable* of arity r , and let \vec{x} be an r -tuple of variables. Let φ be a formula whose free variables may include X as a free relation variable and \vec{x} as free variables. For any set $F \subseteq U(\mathcal{A})^r$, let $\varphi[F]$ denote the set of r -tuples $\vec{v} \in U(\mathcal{A})^r$ for which φ holds when X is interpreted as F and \vec{v} is assigned to \vec{x} . Let the sets F_i be defined inductively by $F_0 = \varphi[\emptyset]$ and $F_{i+1} = F_i \cup \varphi[F_i]$. Since $F_i \subseteq F_{i+1}$ for all $i \in \mathbb{N}_0$, we have $F_k = F_{|U(\mathcal{A})|^r}$ for all $k \geq |U(\mathcal{A})|^r$. We call the r -ary relation $F_{|U(\mathcal{A})|^r}$ the *inflationary fixed-point* of φ and denote it by $(\text{ifp}_{X \leftarrow \vec{x}} \varphi)$. The free variables of the formula $(\text{ifp}_{X \leftarrow \vec{x}} \varphi)(\vec{a})$, $|\vec{a}| = |\vec{x}|$, is $\{\vec{a}\} \cup (\text{free}(\varphi) \setminus \{\vec{x}\})$. FP denotes the extension of FO with the ifp-operator.

In 1982, Immerman [Imm82] and Vardi [Var82] showed that FP characterizes PTIME on classes of ordered structures⁴.

Immerman-Vardi Theorem 2.3.10. *FP captures PTIME on the class of ordered structures.* \blacksquare

Lemma 2.3.11 ([EF99]). $\text{TC} \leq \text{FP}$. \blacksquare

Infinitary logic $L_{\infty\omega}^\omega$. Let Φ_s be an arbitrary, possibly infinite set of formulas so that for each $\varphi \in \Phi_s$, all *free and bound* variables in φ are among x_1, \dots, x_s . *Infinitary first-order logic with s variables* $L_{\infty\omega}^s$ is obtained by extending FO with the rule that, for each set Φ_s as above,

$$\bigvee \Phi_s$$

⁴In fact, Immerman and Vardi showed this capturing result using a different fixed-point operator for *least fixed points*. Inflationary and least fixed points were shown to be equivalent by Gurevich and Shelah [GS85] and Kreutzer [Kre04].

is also a formula. The semantics is clear: $\mathcal{A} \models \bigvee \Phi_s$ if and only if there is $\varphi \in \Phi_s$ such that $\mathcal{A} \models \varphi$. *Finite-variable infinitary logic* $L_{\infty\omega}^\omega$ is now obtained as the union of all the s -variable fragments: $L_{\infty\omega}^\omega := \bigcup_{s \in \mathbb{N}} L_{\infty\omega}^s$. Thus, formulas in $L_{\infty\omega}^\omega$ can be of infinite length, but they may contain only a finite number of different variables.

2.3.3 Counting logics

Two-sorted structures. We equip structures with an additional integer sort. For a τ -structure $\mathcal{A} = (U(\mathcal{A}), (R^A)_{R \in \tau})$ we define \mathcal{A}^+ to be the extension of \mathcal{A} by the standard model of arithmetic. In other words, \mathcal{A}^+ is the two-sorted structure $(U(\mathcal{A}), \mathbb{N}_0, (R^A)_{R \in \tau}, +, \cdot, \leq, 0, 1)$, where $+$ and \cdot are binary functions denoting standard addition and multiplication over the integer sort, \leq is the linear order on the integers, and 0 and 1 are the usual constants from \mathbb{N}_0 .⁵ Notice that none of \mathcal{A} 's relations is defined anywhere on the number sort. Likewise, \leq is defined nowhere on $U(\mathcal{A})$.

For logics over such two-sorted structures, we assume all variables to be typed, so each variable x ranges either over the universe $U(\mathcal{A})$ or over the numbers \mathbb{N}_0 . x is called an *element variable* if it ranges over $U(\mathcal{A})$ and a *number variable* if it ranges over \mathbb{N}_0 . A *numeric term* t is a term in the language of \mathcal{A}^+ that takes values $t^{\mathcal{A}} \in \mathbb{N}_0$.

Bounded quantification. In order to avoid undecidability, quantification over number variables has to be *bounded* (also known as *guarded quantification*). Thus, if x is a number variable, its binding quantifier must appear in the form $\forall x \leq t \varphi$ or $\exists x \leq t \varphi$ where t is a numeric term in which x does not occur free. Let us denote first-order logic over the two-sorted extension of structures with bounded quantification over the numerical sort by FO^+ . Notice that each structure \mathcal{A} is extended to \mathcal{A}^+ in a canonical fashion, therefore we can view FO^+ -sentences as defining properties of plain structures. Consequently, we usually write $\mathcal{A} \models \varphi$ to mean $\mathcal{A}^+ \models_{\text{FO}^+} \varphi$.

Counting operators. The connection between a structure's universe and the number sort is established by *counting operators*. Let φ be an $\text{FO}^+[\tau]$ -formula. We define the *numeric counting term* $\sharp \vec{x} \varphi$ to denote the number of assignments to \vec{x} so that φ holds, i.e. $(\sharp \vec{x} \varphi)^{\mathcal{A}} = |\{\vec{a} \in U(\mathcal{A})^{|\vec{x}|} \mid \mathcal{A} \models \varphi[\frac{\vec{a}}{\vec{x}}]\}|$. We also allow counting operators to bind number variables, but once again we require all their occurrences to be bounded. We write $\sharp \vec{x} \vec{y} \leq \vec{t} \varphi$ to indicate that the number variables y_i are bounded by numeric terms t_i for each i respectively, and that y_i is not free in t_i . We set $\text{free}(\sharp \vec{x} \vec{y} \leq \vec{t} \varphi) = \text{free}(\vec{t}) \cup (\text{free}(\varphi) \setminus \{\vec{x}, \vec{y}\})$. Let FO+C denote the extension of FO^+ with counting operators. Note that $(\sharp x x = x)^{\mathcal{A}} = |\mathcal{A}| \in \mathbb{N}$.

Data complexity of FO+C. It is easy to see that for any numeric τ -term t of FO+C with free variables \vec{x} there is a polynomial p so that $t^{\mathcal{A}} \leq p(|\mathcal{A}|, \vec{x}^{\mathcal{A}})$ for all τ -structures \mathcal{A} . Consequently, if η is an FO+C -term or formula then there is a logspace-bounded Turing machine M_η which, on input $\text{enc}(\mathcal{A})$ and an assignment to η 's free variables, computes $\eta^{\mathcal{A}}$ (cp. [EF99]).

Duality between numeric terms and formulas. If η is a numeric FO+C -term, then $\psi(z) := \eta = z$ is an FO+C -formula which holds precisely when z is equal to $\eta^{\mathcal{A}}$. Conversely, suppose $\psi(z)$ is a formula with a number variable z and t is a term so that $\psi(z)$ holds for precisely one number $z \leq t^{\mathcal{A}}$. Then $\eta := \sharp x \leq t \exists z \leq t x < z \wedge \psi(z)$ is a numeric term defining this value, i.e. $\mathcal{A} \models \psi[\eta^{\mathcal{A}}]$.⁶ Thus, we may switch between terms and formulas defining numbers over \mathbb{N}_0 in a flexible manner. The same is true in all other counting logics.

With this duality between terms and formulas in mind, it is an easy matter to define all sorts of basic arithmetic operations over FO+C 's number sort. For example, if s and t are numeric terms, then $s - t$ is FO+C -definable (given that $s \leq t$), and the same is true of $s \bmod t$, s/t (division

⁵The constants 0 and 1 can also be defined in counting logics. Since they are frequently used in the construction of terms, though, we prefer to include them explicitly.

⁶The $<$ -relation is used because $0 \in \mathbb{N}_0$ counts as a satisfying assignment too.

2 Notation, Definitions, Foundational Results

without remainder), as well as addition, subtraction, multiplication, and division over GF_p when p is prime.

Definition of counting logics. With the exception of infinitary logic, we consider the extension with counting operators of all the logics introduced in Section 2.3.2. We allow all operators to bind number variables, but again, each number variable needs to be bounded.

- DTC+C, STC+C, and TC+C denote the extensions of FO+C with the *dtc*, *stc*, and *tc* operator, respectively. The term bounding number variables is indicated as in $\text{tc}_{\vec{x}_1 \vec{x}_2 \leq t, \vec{y}_1 \vec{y}_2 \leq t} \varphi$. The semantics is clear from the observation that φ defines a directed graph on $U(\mathcal{A})^{|\vec{x}_1|} \times [t^{\mathcal{A}}]^{|\vec{x}_2|}$.
- FP+C is the extension of FO+C with the *ifp* operator. The bound on number variables is indicated as in $\text{ifp}_{X \leftarrow \vec{x} \vec{y} \leq t} \varphi$, and we note that X may now be a relation variable of mixed type over the universe and the number sort. The fixed-point is then formed over the domain $U(\mathcal{A})^{|\vec{x}|} \times [t^{\mathcal{A}}]^{|\vec{y}|}$ which means that X may never contain any tuples which contain an integer larger than $t^{\mathcal{A}}$.

It is clear how to define the free variables in each case. Also, it can be checked that the data complexity of DTC+C and STC+C is in LOGSPACE, the data complexity of TC+C is in NLOGSPACE, and the data complexity of FP+C is in PTIME.

In each of the above definitions we have only allowed for the specification of a single term bounding all number variables simultaneously. This was mainly done out of notational convenience. An individual bound t_i for a number variable y_i can sensibly be incorporated into the bound formula φ by conjoining the condition $y_i \leq t_i$.

Lemma 2.3.12 ([EF99]). $\text{FO+C} \leq \text{DTC+C} \leq \text{STC+C} \leq \text{TC+C} \leq \text{FP+C}$. ■

Correspondence to classical counting logics. In most of the literature on the logics we have introduced here, these logics are defined slightly differently. Instead of adjoining all of \mathbb{N}_0 as the counting sort, they only consider two-sorted structures of the form $(\mathcal{A}, [0, |\mathcal{A}|], <)$. In return, quantification over the number sort does not have to be bounded in these logics. Of course, sentences defined in this model carry over to our definition of counting logics by simply using the bound $|\mathcal{A}|$ for all quantification over the number sort.

Conversely, we can encode terms from our counting logics into the restricted model by encoding numbers larger than $|\mathcal{A}|$ with tuples of variables. This is possible in a uniform way because of the polynomial bound on numeric terms. The arithmetic necessary for the encoding and the simulation of the operations $+$ and \cdot is definable in DTC over the ordered number sort (see [EF99]). Thus, all of our counting logics which are at least as expressive as DTC+C are precisely as expressive as their classically defined counterparts. Note that this is not necessarily true of FO+C, however, our results do not concern the precise expressiveness of FO+C.

In choosing our definition, we had to weigh up the increased flexibility of our version against the need to explicitly specify term bounds all the time. As we will introduce rank logics over matrices which are larger than the base structure in Chapter 5, the definition here makes for the cleaner presentation. We remark that our definition is also more at home with the framework of *metafinite model theory* introduced by Grädel and Gurevich in [GG98].

2.3.4 Logical reductions

Syntactic interpretations. We will be using various notions of logical reductions between classes of structures throughout. All of them will be special cases of the general notion of *syntactic interpretations*. Actually, the literature does not agree on one single definition of what constitutes a syntactic interpretation. Instead, authors usually select those aspects of syntactic interpretations

which suit their purpose and leave out those which would complicate their presentation. We will do the same here and not present the most general definition of syntactic interpretations, but rather an intermediate one. We will discuss generalizations afterwards, though, and in particular we will present further restrictions that are particularly easy to handle and which are sufficient for our work. In this way, we hope to convey the strength of using syntactic interpretations as a generic tool, while at the same time avoiding any unnecessary technical convolutions.

Definition 2.3.13 (Syntactic interpretation/ logical reduction). Let L be a logic and let τ_1, τ_2 be vocabularies. An ℓ -ary L -interpretation from τ_1 to τ_2 is a tuple

$$\Gamma = (\gamma_V(\vec{x}), \gamma_{\approx}(\vec{x}, \vec{y}), (\gamma_R(\vec{y}_R))_{R \in \tau_2}),$$

of $L[\tau_1]$ -formulas where $\vec{x}, \vec{y}, \vec{y}_R$ are tuples of element variables with $|\vec{x}| = |\vec{y}| = \ell$ and $|\vec{y}_R| = k \cdot \ell$ for each k -ary $R \in \tau_2$. An L -interpretation is also called an L -reduction.

Remark 2.3.14. When considering counting logics, it also makes sense to allow syntactic interpretations *to the number sort* by allowing Γ 's formulas to use number variables. In this case, the formula γ_V defining the universe of target structures needs to be accompanied with numeric terms for every number variable, imposing explicit polynomial bounds on the universe of the target structure. In fact, canonization mappings can be considered a special case of such *numeric syntactic interpretations*.

Syntactic interpretations as functions of structures. Let \mathcal{A} be a τ_1 -structure and suppose that Γ is an ℓ -ary L -interpretation from τ_1 to τ_2 so that $\gamma_V^{\mathcal{A}}[\cdot] \neq \emptyset$. If $\gamma_{\approx}^{\mathcal{A}}[\cdot, \cdot]$ defines an equivalence relation \approx on $\gamma_V^{\mathcal{A}}[\cdot]$, then we let $U(\Gamma[\mathcal{A}]) = \gamma_V^{\mathcal{A}}[\cdot] / \approx$. Otherwise, we let $U(\Gamma[\mathcal{A}]) = \gamma_V^{\mathcal{A}}[\cdot]$. We define the τ_2 -structure $\Gamma[\mathcal{A}]$ over the universe $U(\Gamma[\mathcal{A}])$ with each k -ary relation $R \in \tau_2$ induced by $\gamma_R^{\mathcal{A}}[\cdot, \dots, \cdot] \cap (\gamma_V^{\mathcal{A}}[\cdot])^k$, where γ_R is seen as a k -ary relation on ℓ -tuples from $U(\mathcal{A})$. That means that $[\vec{x}_1] \dots [\vec{x}_k] \in R^{\Gamma[\mathcal{A}]}$ whenever there are $\vec{y}_1 \in [\vec{x}_1], \dots, \vec{y}_k \in [\vec{x}_k]$ so that $\mathcal{A} \models \gamma_R^{\mathcal{A}}[\vec{y}_1, \dots, \vec{y}_k]$. In this way, syntactic interpretations transform τ_1 -structures into τ_2 -structures.

Pull back under syntactic interpretations. For us, the most important property of syntactic interpretations is that they allow to *pull back* τ_2 -formulas under certain conditions. For example, it is easy to show the following.

Proposition 2.3.15 ([EFT94, EF99]). Let τ_1 and τ_2 be vocabularies and let Γ be an ℓ -ary FO-interpretation from τ_1 to τ_2 . Then for every FO $[\tau_2]$ -sentence ψ there is an FO $[\tau_1]$ -sentence $\psi^{-\Gamma}$ such that

$$\mathcal{A} \models \psi^{-\Gamma} \quad \Leftrightarrow \quad \Gamma[\mathcal{A}] \models \psi. \quad \blacksquare$$

Thus, if ψ is an FO-formula defining a property of τ_2 -structures, then $\psi^{-\Gamma}$ is an FO-formula which defines the property of τ_1 -structures that ψ holds after applying Γ . In this way, Γ pulls back FO-definable properties of τ_2 -structures to FO-definable properties of τ_1 structures.

Closure under logical reductions. Whenever K_1 is a class of τ_1 -structures and K_2 a class of τ_2 -structures, we also say that Γ is a *first-order reduction from K_1 to K_2* if any τ_1 -structure \mathcal{A} is in K_1 if and only if $\Gamma[\mathcal{A}] \in K_2$. Notice that syntactic interpretations and logical reductions use the exact same formalism.

Definition 2.3.16. Let L and F be logics. We say that L is *closed under F -reductions* if for all (relational) vocabularies τ_1, τ_2 , all $\ell \in \mathbb{N}$, all ℓ -ary F -interpretations Γ from τ_1 to τ_2 , and all $L[\tau_2]$ -sentences ψ there is an $L[\tau_1]$ -sentence $\psi^{-\Gamma}$ such that

$$\mathcal{A} \models \psi^{-\Gamma} \quad \Leftrightarrow \quad \Gamma[\mathcal{A}] \models \psi.$$

2 Notation, Definitions, Foundational Results

for any τ_1 -structure \mathcal{A} . If L is closed under L -reductions, then we also say that L is *closed under logical reductions*.

It is not hard to prove that when L is closed under F -reductions, then $F \leq L$. Whether a logic is closed under logical reductions or not depends on various properties. For example, monadic second-order logic is not closed under FO -reductions since pull-backs of formulas under k -ary interpretations with $k > 1$ would require the ability to quantify over k -ary relations.

Lemma 2.3.17 ([Ott97]). *The logics FO , DTC , STC , TC , FP , $DTC+C$, $STC+C$, $TC+C$, and $FP+C$ are all closed under logical reductions.*

Idea. The proof essentially consists of modifying occurrences of relation symbols $R \in \tau_2$ and quantification in φ with the right versions of γ_V , γ_{\approx} and γ_R . For FO , FP , and $FP+C$, the result is proven by Otto in [Ott97, Lemma 1.49, Proposition 4.8]. The proofs for DTC , STC , TC and their counting variants is analogous. When we prove Lemma 5.2.9, which shows the corresponding statement for rank logics, most of what there is to show for the logics here is also covered there. Notice that it is FO -definable whether γ_{\approx} defines an equivalence relation on $\gamma_V[\cdot]$. Under our definition of counting logics, Lemma 2.4.3 below is needed in order to deal with counting quantifiers in a sensible manner. \square

Remark 2.3.18. The counting logics mentioned in Lemma 2.3.17 are equally closed under numeric syntactical reductions (see Remark 2.3.14).

Other notions of syntactic interpretations. Let us make some remarks about our definition of syntactic interpretations. It is very similar to the definition given in [EF99], except that we also allow formulas γ_{\approx} to define equivalence relations by which we modify the universe of the target structure. Modding out by equivalence relations will give us additional freedom in defining maps between structures and we will use this possibility in several places.

The concept of a syntactic interpretation Γ may be strengthened further by allowing an additional tuple of variables \vec{z} to *parameterize* Γ . In that case, \vec{z} appears free in all the formulas contained in Γ and an additional formula $\gamma_{app}(\vec{z})$ defines which values of \vec{z} are allowed to define a corresponding map $\Gamma(\vec{z})$. Such a definition of syntactic interpretations can be found in [Gro10a]. As an example, we can define in DTC an ordering on the vertices of a cycle if the direction of an edge (z_1, z_2) is given. Thus, there is a DTC -interpretation from cycles to ordered cycles. Similarly, 3-connected planar graphs can be ordered in FP by an appropriate choice of parameters ([Gro98b]).⁷ We will not require this additional feature of syntactic interpretations.

Note also that we are restricting our syntactic interpretations to structures over relational vocabularies. While this is enough for our purposes, syntactic interpretations can also be defined for vocabularies containing constants and function symbols. In that case, we use formulas to define the graphs of functions and constants in τ_2 , and we require in addition that function values and constants are uniquely defined. This approach is taken in [EFT94] for unary FO -interpretations. In this case, being closed under logical reductions further requires a logic to permit the replacement of function and constant symbols with formulas. This is the case for all the logics considered in Lemma 2.3.17, but this will not be important for our work here.

Graph interpretations. Our main use for syntactic interpretations will be to show that logics do not capture certain complexity classes on specific graph classes in Section 4.2. When reducing between graph classes, both the source and the target vocabulary just contain one binary relation. We call this special case of syntactic interpretations *graph interpretations* and re-state their simplified definition to make these results more directly accessible.

⁷If we define canonical forms on the number sort instead of canonical orderings on the structure's universe (see Section 2.4), then the use of additional parameters to define linear orders can be circumvented by defining the syntactic interpretation to map to the lexicographic leader obtained among all parameters (cf. Lemma 2.4.5).

Definition 2.3.19. Let L be a logic. An ℓ -ary L -graph interpretation is a tuple $\Gamma = (\gamma_V(\vec{x}), \gamma_{\approx}(\vec{x}, \vec{y}), \gamma_E(\vec{x}, \vec{y}))$ of L -formulas so that $|\vec{x}| = |\vec{y}| = \ell$ and in any graph, γ_{\approx} defines an equivalence relation \approx on $\gamma_V^G[\cdot]$. If $G = (V, E)$ is a graph, then $\Gamma[G] = (V_\Gamma, E_\Gamma)$ denotes the graph with vertex set $V_\Gamma = \gamma_V^G[\cdot] / \approx$ and edge set E_Γ where $[\vec{u}][\vec{v}] \in E_\Gamma$ if and only if $[\vec{u}] \neq [\vec{v}]$ and there are $\vec{u}' \in [\vec{u}]$ and $\vec{v}' \in [\vec{v}]$ such that $G \models \gamma_E[\vec{u}', \vec{v}']$ or $G \models \gamma_E[\vec{v}', \vec{u}']$.

As a consequence of the closure of FP+C under logical reductions, we point out the following fact for later reference.

Graph Interpretations Lemma 2.3.20. Let Γ be an ℓ -ary FP+C-graph interpretation. Then for any FP+C-sentence φ there is a sentence $\varphi^{-\Gamma}$ with the property that $G \models \varphi^{-\Gamma} \Leftrightarrow \Gamma[G] \models \varphi$. \square

2.4 Definable canonization mappings

When we are trying to prove that a logic L captures PTIME on a class \mathcal{K} , it seems hopeless to try and define the computations of Turing machines M over encodings of structures from \mathcal{K} directly. After all, M still operates just like any other PTIME Turing machine, so that our knowledge about the input of M does not seem to give us an advantage over capturing PTIME outright on the class of all structures.

A more sensible approach is to make use of capturing results on ordered structures such as the Immerman-Vardi Theorem 2.3.10. If we can show that for any $\mathcal{A} \in \mathcal{K}$ we can define an ordered version of \mathcal{A} , then we can apply the result for ordered structures in order to define any PTIME property of \mathcal{K} .

Canonicity and isomorphism invariance. Let \mathcal{K} and \mathcal{Z} be classes of structures. A mapping $f : \mathcal{K} \rightarrow \mathcal{Z}$ is called *invariant under isomorphism* if $\mathcal{A} \cong \mathcal{B} \in \mathcal{K}$ implies that $f(\mathcal{A}) \cong f(\mathcal{B})$. More generally, let Z be any set with an equivalence relation \sim_Z on it. For example, we might consider $Z = \mathbb{N}$ and let \sim_Z be equality on \mathbb{N} . Then a map $f : \mathcal{K} \rightarrow Z$ is invariant under isomorphism if $\mathcal{A} \cong \mathcal{B} \in \mathcal{K}$ implies that $f(\mathcal{A}) \sim_Z f(\mathcal{B})$.

In this setting, a function $f : \mathcal{K} \rightarrow Z$ is said to be *canonical* if it is invariant under isomorphism, the induced map $\mathcal{K} / \cong \rightarrow Z / \sim_Z$ is injective, and \sim_Z is easy to decide. Thus, $\mathcal{A}, \mathcal{B} \in \mathcal{K}$ are isomorphic **if and only if** $f(\mathcal{A}) \sim_Z f(\mathcal{B})$. Since \sim_Z is supposed to be easy to decide, computing f gives us a way to decide isomorphism of structures in \mathcal{K} . We leave the last criterion about \sim_Z fuzzy on purpose. Most of the time, \sim_Z will denote equality on a set or order isomorphism of structures where easy decidability is clear. However, we sometimes define other objects from which isomorphism of structures can be inferred, such as the *modular tree* in Section 4.5, and we also like to call these objects canonical.

Canonical orderings. A canonical ordering on a class of structures \mathcal{K} is a mapping associating with each $\mathcal{A} \in \mathcal{K}$ a linear order $<_{\mathcal{A}}$ of $U(\mathcal{A})$ with the property that for isomorphic $\mathcal{A}, \mathcal{B} \in \mathcal{K}$ the ordered structures $(\mathcal{A}, <_{\mathcal{A}})$ and $(\mathcal{B}, <_{\mathcal{B}})$ are order isomorphic.

Since order isomorphism of two structures is trivial to decide, efficiently computable canonical orderings immediately entail efficient isomorphism tests. If we are able to *define* orderings on \mathcal{K} -structures in a logic such as FP+C, then these orderings are automatically canonical since the properties defined in FP+C are invariant under isomorphism. Furthermore, any object we define using the canonical ordering of a structure \mathcal{A} is invariant under isomorphism, which is not the case when we use arbitrary orders. Therefore, canonical orderings are certainly very desirable.

Unfortunately, canonical orderings are often times impossible to define by logics. For example, if we consider an \emptyset -structure (i.e., bare set) A , then any two linear orderings $<_1$ and $<_2$ of A give us order isomorphic structures $(A, <_1), (A, <_2)$. As FP-formulas define order-invariant properties, any FP-formula defining any order on A would somehow have to define all $n!$ orders on A . In fact,

if FP could define an ordering on A , then it could decide whether $|A|$ is even – since FP cannot define evenness (cf. [EF99]), it therefore cannot define any linear order on A . Turing machines do not suffer from this restriction since they can arbitrarily choose any ordering of A . Section 4.5 considers the more elaborate case of defining canonical orderings of interval graphs in logspace.

Canonization mappings. Let \mathcal{K} be a class of τ -structures. By a *canonization mapping* or *canonical form* for \mathcal{K} we mean a map f which associates with any $\mathcal{A} \in \mathcal{K}$ an ordered structure $f(\mathcal{A}) = (\mathcal{A}_f, <_f)$ so that

- $\mathcal{A} \cong \mathcal{A}_f$ as τ -structures, and
- $\mathcal{A}, \mathcal{B} \in \mathcal{K}$ are isomorphic if and only if $f(\mathcal{A})$ and $f(\mathcal{B})$ are order isomorphic.

Unlike canonical orderings, canonization mappings do not require us to define an ordering on the universe of a given structure \mathcal{A} . Instead, canonizations ask for a *canonical ordered isomorphic copy* of \mathcal{A} . We call this copy $f(\mathcal{A})$ the *canon* of \mathcal{A} .

Definable canonization. Let $\tau = (R_1, \dots, R_k)$ be a vocabulary where R_i has arity a_i and let \mathcal{K} be a class of τ -structures. Let L be a counting logic such as FP+C. By an L -definable canonization mapping for \mathcal{K} we mean a tuple $\Gamma = (t_V, \gamma_1(\vec{y}_1), \dots, \gamma_k(\vec{y}_k))$ where t_V is a numeric term, each \vec{y}_i are number variables with $|\vec{y}_i| = a_i$, and for which the mapping $\mathcal{A} \mapsto \Gamma[\mathcal{A}]$ is a canonization mapping for \mathcal{K} . $\Gamma[\mathcal{A}]$ is the τ -structure $([t_V^{\mathcal{A}}], \gamma_1^{\mathcal{A}}[\cdot], \dots, \gamma_k^{\mathcal{A}}[\cdot])$ which is ordered by the natural order on $[t_V^{\mathcal{A}}] \subset \mathbb{N}$. As FP+C is closed under logical reductions, the Immerman-Vardi Theorem 2.3.10 implies the following lemma.

Lemma 2.4.1. *Let \mathcal{K} be a class of τ -structures and suppose that there is an FP+C-definable canonization mapping for \mathcal{K} . Then FP+C captures PTIME on \mathcal{K} . \square*

Canonization mappings are weaker than canonical orderings, in the sense that canonical forms can easily be defined from any canonical ordering. However, most capturing results of PTIME on a certain class \mathcal{K} only exhibit a logically definable canonization mapping from structures in \mathcal{K} to the number sort. Our capturing results in Chapters 3 and 4 will also be proved in this way. For example, in the latter chapter we show that there is an FP+C-formula $\varepsilon(x, y)$ with numeric variables x and y so that any interval graph $G = (V, E)$ is isomorphic to $([|V|], \varepsilon^G[\cdot, \cdot])$.

Consequences of FP+C-canonization. Cai, Fürer and Immerman have observed in [CFI92] that if a structure class admits FP+C-definable canonization, then there is an integer $k \in \mathbb{N}$ so that a generic method known as the k -dimensional Weisfeiler-Lehman (WL) algorithm can be used to decide isomorphism in polynomial time ($O(n^k \log n)$). We introduce and use 2-dimensional WL ourselves in Section 3.2.3. Of course, FP+C-definable canonical forms provide us with an efficiently computable canonization mapping and isomorphism test anyways. The importance here lies in the fact that a simple combinatorial algorithm decides isomorphism without specifically exploiting these structures' inherent composition. The algorithm is generic in the sense that it also decides isomorphism of planar graphs [Gro98b], graphs of bounded treewidth [GM99], and many others.

Complete invariants. Given a class of structures \mathcal{K} , a complete invariant for \mathcal{K} is a mapping ι from \mathcal{K} to an ordered set D such that $\mathcal{A}, \mathcal{B} \in \mathcal{K}$ are isomorphic if and only if $\iota(\mathcal{A}) = \iota(\mathcal{B})$. Any canonization mapping f can be turned into the complete invariant $\mathcal{A} \mapsto \text{enc}(f(\mathcal{A}))$, so complete invariants are a weaker concept than canonical forms. Still, efficiently computable complete invariants imply efficient isomorphism tests. ι is a *canonical function* in the sense defined above, however, it is an open question whether it is possible to obtain an FP+C-definable canonization mapping from an FP+C-definable complete invariant ι . Interestingly, on the class of ordered structures, Gurevich [Gur97] shows that PTIME-computable complete invariants can be turned into PTIME canonization mappings. PTIME canonization mappings in turn imply the existence

of a logic effectively capturing PTIME in the broad sense of logics described in Section 2.3.1. This will not be relevant for our work, but the interested reader is pointed to the discussion in [EF99, Section 11.1].

2.4.1 Basic formulas

At this point, let us note some basic constructions which will be useful later on. The first one is used in the embedding of counting logics into infinitary logic in Section 5.6.2.

Lemma 2.4.2. *Let \mathcal{A} be a structure with a linear order $<$ defined on its elements. For each $i \in \mathbb{N}$, there is an FO-formula $\rho_i(x)$ using only the variables x, y, z which holds if and only if x is the i^{th} element in the ordering $<$.*

Proof. The formulas are built inductively:

- $\rho_1(x) := \neg \exists y y < x$.
- $\rho_j(x) := \exists y (y = x \wedge \exists x (\rho_{j-1}(x) \wedge \neg \exists z x < z < y))$. □

The remaining constructions in this section are important for defining FP+C canonical forms. They lay the technical foundation that allows us to abstract away from bulky formulas and give more high-level descriptions of the canonization procedures in Chapters 3 and 4. The existence of these formulas is essentially folklore, and variants of them can for example be found in [GKL⁺07]. The first lemma aides us in using auxiliary structures which are defined in terms of modouts by equivalence relations. This is an important strategy in canonization procedures since the lack of an ordering makes it impossible to choose representatives for equivalence classes.

Lemma 2.4.3 (Counting equivalence classes). *Let L be a logic with counting and $\text{DTC}+\text{C} \leq L$. Suppose \sim is an L -definable equivalence relation on k -tuples from A , and let $\varphi(\vec{x})$ be an L -formula with $|\vec{x}| = k$. Then there is an L -counting term giving the number of equivalence classes $[\vec{v}]$ of \sim such that $G \models \varphi[\vec{u}]$ for some $\vec{u} \in [\vec{v}]$.*

Proof. The idea is to construct the sum slicewise for each cardinality of equivalence classes first, which gives us control over the number of classes rather than the number of elements in these classes. Let s, z, a, b be number variables. Define the relation $R(s, \vec{x})$ to hold if \vec{x} is contained in a \sim -equivalence class of size s which contains some element making φ true. Using the dtc-operator, it is then easy to define the sum

$$\sum_{s \in [|A|^k]} \#a \leq |A|^k \exists b \leq |A|^k (a < b \wedge b \cdot s = \# \vec{x} R(s, \vec{x}))$$

from which we can derive the desired counting term. □

Lexicographic order on ordered structures. Let τ be a vocabulary with $\leq \in \tau$. Recall the canonical encoding $\text{enc}(\mathcal{A})$ of ordered τ -structures \mathcal{A} as $\{0, 1\}$ -strings defined in Section 2.3.1. If \mathcal{A}, \mathcal{B} are ordered τ -structures, then we say that \mathcal{A} is *lexicographically smaller than* \mathcal{B} , written $\mathcal{A} <_{\text{lex}} \mathcal{B}$, if $\text{enc}(\mathcal{A})$ is lexicographically smaller than $\text{enc}(\mathcal{B})$. As noted before, two structures \mathcal{A} and \mathcal{B} are order isomorphic if and only if they are incomparable with respect to $<_{\text{lex}}$. Thus, $<_{\text{lex}}$ is a strict linear order of ordered τ -structures. We let \leq_{lex} denote the obvious reflexive version of $<_{\text{lex}}$.

The next step is to argue that this lexicographic order is definable. For the sake of a concise exposition we choose to encode ordered τ -structures as follows. If a is the maximum arity of relations R_1, \dots, R_k in τ , then let $\vec{y} = (y_0, y_1, \dots, y_a)$ be an $a + 1$ tuple of variables. For a structure \mathcal{A} with $|\mathcal{A}| \geq k$, we consider y_0 as an integer and let $\varphi(\vec{y})$ hold if and only if $y_0^{\mathcal{A}} \leq k$ and $y_1 \dots y_{a_{y_0}} \in$

2 Notation, Definitions, Foundational Results

$R_{y_0}^{\mathcal{A}}$, where a_{y_0} is the arity of R_{y_0} . It can be checked that for all ordered τ -structures \mathcal{A}, \mathcal{B} , we have $\mathcal{A} <_{lex} \mathcal{B}$ if and only if $(U(\mathcal{A}), \varphi^{\mathcal{A}}) <_{lex} (U(\mathcal{B}), \varphi^{\mathcal{B}})$. With this encoding it is clear how the following lemma can be used to define the lexicographic order on ordered structures.

Lemma 2.4.4. *Let \vec{y} be number variables bounded by a numeric term t , let \vec{x} be any variables, and let $\varphi(\vec{x}, \vec{y})$ be some formula. There is an FO^+ -formula $\vartheta(\vec{x}, \vec{x}')$ based on φ which holds if and only if $\varphi[\vec{x}, \cdot] \leq_{lex} \varphi[\vec{x}', \cdot]$ on the ordered domain $[t]^{|\vec{y}|}$.*

Proof.

$$\vartheta(\vec{x}, \vec{x}') := \forall \vec{y} \leq t \left((\varphi(\vec{x}, \vec{y}) \wedge \neg \varphi(\vec{x}', \vec{y})) \rightarrow \exists \vec{y}' \leq t (\vec{y}' \leq_{lex} \vec{y} \wedge \varphi(\vec{x}', \vec{y}') \wedge \neg \varphi(\vec{x}, \vec{y}')) \right)$$

where $\vec{y}' \leq_{lex} \vec{y} = y'_1 \leq y_1 \vee (y'_1 = y_1 \wedge y'_2 \leq y_2) \vee \dots \vee (\bigwedge_{i=1}^{|\vec{y}|-1} y'_i = y_i \wedge y'_{|\vec{y}|} \leq y_{|\vec{y}|})$. \square

Lexicographic leaders. In the setting of Lemma 2.4.4, if $\varphi[\vec{x}, \cdot]$ is lexicographically smallest with respect to all tuples \vec{x} , then $\varphi[\vec{x}, \cdot]$ is called the *lexicographic leader*. Observe that such \vec{x} need not be unique. Lexicographic leaders are used to break ties during the inductive definition of a graph's ordered canonical form.

Lemma 2.4.5 (Lexicographic leader). *Let L be a counting logic with $\text{FO+C} \leq L$ and let \mathcal{A} be a structure. Let \vec{y} be a tuple of number variables with bound t and let \vec{x} be a tuple of variables taking values in $U(\mathcal{A})^k \times [s^{\mathcal{A}}]^\ell$. Suppose $\varphi(\vec{x}, \vec{y})$ is an L -formula and \sim is an L -definable equivalence relation on $U(\mathcal{A})^k \times [s^{\mathcal{A}}]^\ell$. Then there is an L -formula $\lambda(\vec{x}, \vec{y})$ so that for any $\vec{v} \in U(\mathcal{A})^k \times [s^{\mathcal{A}}]^\ell$, $\lambda^{\mathcal{A}}[\vec{v}, \cdot]$ is the lexicographic leader among the relations $\{\varphi^{\mathcal{A}}[\vec{u}, \cdot] \mid \vec{u} \sim \vec{v}\}$.*

Proof. Let $\vartheta(\vec{x}, \vec{x}')$ be the formula from Lemma 2.4.4 for which $\vartheta[\vec{u}, \vec{v}]$ holds if and only if $\varphi[\vec{u}, \cdot]$ is lexicographically smaller or equal to $\varphi[\vec{v}, \cdot]$. Now λ is given by

$$\lambda(\vec{x}, \vec{y}) := \exists \vec{z} (\vec{x} \sim \vec{z} \wedge \varphi(\vec{z}, \vec{y}) \wedge \forall \vec{w} (\vec{w} \sim \vec{z} \rightarrow \vartheta(\vec{z}, \vec{w}))) \quad \square$$

Lexicographic disjoint union. We will repeatedly encounter the situation where the disjoint union of given ordered structures has to be defined in a canonical way. If $\mathcal{A}_1 = ([v_1], (R^{\mathcal{A}_1})_{R \in \tau})$, \dots , $\mathcal{A}_k = ([v_k], (R^{\mathcal{A}_k})_{R \in \tau})$ are ordered structures in lexicographically ascending order, then we define their disjoint union $\mathcal{A} = (A, (R^{\mathcal{A}})_{R \in \tau})$ on $A = [\sum_{i \in [k]} v_i]$ so that \mathcal{A} restricted to $[[\sum_{j \in [i-1]} v_j + 1, \sum_{j \in [i]} v_j]]$ is order isomorphic to \mathcal{A}_i for all $i \in [k]$. We call \mathcal{A} the *lexicographic disjoint union* of $\{\mathcal{A}_i\}_{i \in [k]}$. The following lemma says that lexicographic disjoint unions are FP+C -definable, noting that using the above encoding it is sufficient to consider a single relation.

Lemma 2.4.6 (Lexicographic disjoint union). *Suppose \sim is an FP+C -definable equivalence relation on $U(\mathcal{A})^k \times [s^{\mathcal{A}}]^\ell$. Let $v(\vec{x})$ be a numeric term, $\varepsilon(\vec{x}, \vec{z})$ be an FP+C -formula with number variables \vec{z} defining structures $(v^{\mathcal{A}}[\vec{v}], \varepsilon^{\mathcal{A}}[\vec{v}, \cdot])$ on the number sort for each $\vec{v} \in U(\mathcal{A})^k \times [s^{\mathcal{A}}]^\ell$. Then there are FP+C -formulas μ and $\omega(\vec{z})$ defining on \mathbb{N}_0 the lexicographic disjoint union $(\mu^{\mathcal{A}}, \omega^{\mathcal{A}}[\cdot])$ of the lexicographic leaders of \sim 's equivalence classes.*

Proof. We first observe that we may have to compare ordered structures \mathcal{A}, \mathcal{B} whose universes are of different sizes. We let $\mathcal{A} < \mathcal{B}$ if $|\mathcal{A}| < |\mathcal{B}|$ or $|\mathcal{A}| = |\mathcal{B}|$ and $\mathcal{A} <_{lex} \mathcal{B}$.

Let \prec be the strict weak order on $U(\mathcal{A})^k \times [s^{\mathcal{A}}]^\ell$ for which $\vec{x} \prec \vec{y}$ if the lexicographic (v, ε) -leader in \vec{x} 's equivalence class under \sim is lexicographically smaller than the lexicographic leader in \vec{y} 's equivalence class. Using the fixed point-operator, define ω inductively starting with the \prec -least elements, saving in a relation R all those elements $\vec{u} \in U(\mathcal{A})^k \times [s^{\mathcal{A}}]^\ell$ from equivalence classes that have already been considered. In each step, find the \prec -least elements L in $U(\mathcal{A})^k \times [s^{\mathcal{A}}]^\ell$ which are not in R , calculate the number n of equivalence classes contained in L , and then expand

2.4 Definable canonization mappings

ω by n copies of $\lambda[\vec{u}, \cdot]$ (which is the same for any $\vec{u} \in L$). We obtain μ in the same process by summing up the respective $|v^A[\vec{u}]|$. \square

3 Capturing Polynomial Time on Fragments of Logarithmic Size

The fact that FP captures PTIME on ordered structures but fails to do so on general unordered structures, begs one question: how difficult is it to obtain an ordering for any given structure? The answer may, of course, depend on the structures under consideration. For example, very limited means of recursion such as deterministic transitive closure are enough to give an orientation to an undirected path and to extract a linear order from this orientation. The investigation of structure classes and the task of *endowing* them with an ordering based on inherent patterns will be picked up in Chapter 4, where such a result is shown for the class of interval graphs.

Yet, if we ask how hard it is to obtain an ordering *without using the specific composition of the structure at hand*, our approach has to be more generic. Obviously, an ordering is easy to obtain in \exists SO, by simply quantifying existentially over all binary relations and then verifying the first-order axioms of a linear ordering. A naïve implementation on a deterministic Turing machine would take time $O(2^{n^2})$ to cycle through all possible binary relations, where n is the size of the universe. This is not optimal at all, since there are only $n! \leq 2^{n \log n}$ different linear orders on a universe of size n . However, we are not even required to cycle through all possible orders of the universe since we only need *a single linear order* to serve as an aid to operators such as ifp, not *all linear orders*. The first result in this chapter shows that we can really do better on the class of *edge-colored directed graphs*, in that $2^{O(n)}$ time is enough to obtain an ordered copy of any of these graphs.

Theorem 3.1. *There is a combinatorial canonization algorithm for directed graphs with colorings of edges and vertices running in time $2^{O(n)}$, where n is the size of the graph's vertex set.*

Naturally, as we want to answer the question how hard it is to obtain an ordering, we are not even very interested in the algorithmic time complexity on machines which come equipped with implicit orders anyways. Instead, our motivation for such a procedure comes from logical frameworks which do not have an ordering available from the start. Thus, we would like to implement this canonization procedure in such a logical framework. Unfortunately, the algorithm of Theorem 3.1 cannot be implemented in fixed-point logics and the likes. As will be argued below, this is not only due to the exponential run-time, since an implementation is also impossible for logarithmic-sized fragments of the structure. It rather seems that some of the typical algorithmic techniques used in the canonization algorithm cannot be modeled in these FO-based logics.

We can formalize the approach in a different framework, though, invariantly working directly within the structure and without using arbitrary choices in the course of the procedure. To be precise, we show that our canonization method can be cast in *choiceless polynomial time with counting* ($\tilde{\text{CPT}}+\text{C}$) when we restrict ourselves to canonizing a fragment of the structure which is of *logarithmic size*. Using the Immerman-Vardi Theorem 2.3.10, we obtain the main result of this section: that $\tilde{\text{CPT}}+\text{C}$ captures PTIME on such logarithmic fragments.

Theorem 3.2. *Let τ be a finite vocabulary whose relations have arity at most 2 and which contains a unary relation symbol U . Let $c \in \mathbb{N}$. Then $\tilde{\text{CPT}}+\text{C}$ expresses all PTIME-properties of the restriction of \mathcal{A} to U on the class of all τ -structures \mathcal{A} with $|U| \leq c \log |\mathcal{A}|$. Thus, $\tilde{\text{CPT}}+\text{C}$ captures PTIME on logarithmic-sized fragments of τ -structures.*

3 Capturing Polynomial Time on Fragments of Logarithmic Size

Remark. As we are entering the logical world now, we have stated the result for vocabularies of arity at most 2, whose several binary relations can also be viewed as one colored directed edge relation. The difference is that in Theorem 3.2, the vocabulary τ is kept fixed, while the number of edge colors in edge-colored graphs may vary with the size of the graph. As the proof will show, the restriction to fixed vocabularies is not necessary. Given a suitable representation of colored edges and vertices, $\tilde{\text{CPT}}+\text{C}$ also captures PTIME on logarithmic-sized fragments of directed graphs with edge and vertex colorings.

Theorem 3.2 warrants the statement that obtaining an ordering of arity-2 structures without the ability to make arbitrary choices takes singly exponential time. Notice, however, that the proof of the theorem does not provide a canonical ordering of the structure, but instead it defines a *canonical copy* of the restriction of \mathcal{A} to $U^{\mathcal{A}}$ (cp. Section 2.4). Still, for all practical purposes the procedure provides us with a linear ordering of \mathcal{A} without cycling through all possible $|\mathcal{A}|!$ linear orders.

The canonization procedure behind Theorems 3.1 and 3.2 is based on a $2^{O(n)}$ -time canonization algorithm for *undirected graphs* by Corneil and Goldberg [CG84]. Their approach is combinatorial and uses rather basic concepts, which allows us to obtain the necessary generalizations. First of all, Theorem 3.1 allows for an arbitrary amount of unary relations to be present in the structure, i.e., the structures may be colored. Since the algorithm is based on *Weisfeiler-Lehman color refinement* (see Section 3.2.3), the handling of vertex colorings does not cost us any extra effort. Next, we extend Corneil and Goldberg's methods to directed graphs which is not very difficult but was not noted in earlier work. Finally, if we want to handle several directed edge relations at a time, this can be done with just some extra bookkeeping but without changing the conceptual framework.

We note that Theorem 3.1 offers the best-known run-time for edge-colored graphs even though there are faster algorithms available for similar tasks. For example, undirected graphs can be canonized in *moderately exponential time* $2^{n^{\frac{1}{2}+o(1)}}$ by the work of Babai and Luks [BL83]. When we allow at most k colors on edges and vertices, edge-colored directed graphs can be encoded as undirected graphs with no more than $O(k^3 n)$ vertices (see Proposition 3.1.1), so that Babai and Luks's algorithm produces canonical forms in moderately exponential time. The same is true when we consider τ -structures over a fixed vocabulary τ with relations of arity at most 2. We discuss this in greater detail in Section 3.1.

The way in which we apply a singly exponential canonization procedure in order to capture PTIME on logarithmic fragments sheds light on a fundamental weakness of FP and FP+C in comparison with PTIME: the logics are unable to profit from padding. Naturally, if we extend any input string x of length n with 2^n additional symbols, all $2^{O(n)}$ -time (ETIME) properties of x can then be decided in PTIME. Similarly, adjoining to a graph G with n vertices 2^n additional isolated vertices puts into PTIME all ETIME-properties of G .¹ FP, FP+C, and even $\text{C}_{\infty\omega}^{\omega}$, however, do not gain in expressiveness upon indiscriminately *padding our structures* with isolated vertices. To see this, recall that the expressiveness of $\text{C}_{\infty\omega}^{\omega}$ can be precisely described with a pebble game played by the two players *spoiler* and *duplicator* as described in [CFI92, EF99]. It is a simple task to verify that spoiler cannot gain from playing pebbles on the padded vertices and therefore any winning strategy for duplicator on graphs G and H remains intact in the presence of padding. Of course, any logic which is supposed to capture PTIME has to gain in expressiveness from padding in just the same way as time-bounded Turing machines do.

The logic $\tilde{\text{CPT}}+\text{C}$ satisfies this *padding property* essentially by superimposing a time and space restriction which depends on the size of the input structure. $\tilde{\text{CPT}}+\text{C}$ was introduced by Blass,

¹For this, we need to assume that our encoding of graphs does not distinguish between G 's original and padded vertices, so that the padded vertices are also encoded in unary.

Gurevich, and Shelah [BGS99] in 1999 with the idea to take away from PTIME computations the ability to make arbitrary choices or to refer to the ordering which they are implicitly given on the input tape. It is based on Gurevich's more general computational framework of *Abstract State machines*, which gives the logic a rather *algorithmic feel*, as if one was devising imperative programs directly on unordered structures.

Before we give the precise definition of $\tilde{\text{CPT}}+\text{C}$ below in Section 3.2, we want to highlight its main feature that allows for the implementation of our canonization procedure. Unlike logics based on FO, $\tilde{\text{CPT}}+\text{C}$ has the ability to quantify and define relations over objects such as sets, sets of sets, and so on. Objects of this type are called the *hereditarily finite sets* \mathcal{HF} . Of course, quantification over \mathcal{HF} has to be restricted, since otherwise we could easily simulate logics such as $\exists\text{SO}$. Roughly speaking, the number of such objects is the place where $\tilde{\text{CPT}}+\text{C}$ has a polynomial restriction. On fragments U of logarithmic size $m = O(\log n)$ we can therefore explicitly consider all 2^m subsets, so that $\tilde{\text{CPT}}+\text{C}$ trivially subsumes the expressive power of monadic second-order logic on such logarithmic fragments. Similarly, as the central auxiliary structure used in our canonization process, the *recursion tree*, is defined over partitions of U and has at most singly exponential size, it can also be explicitly defined in $\tilde{\text{CPT}}+\text{C}$.

The ability to handle hereditarily finite sets as *first-class objects*, albeit in a limited fashion, appears to be critical to the implementation of the canonization procedure. As argued above, it is impossible to implement it in $\text{C}_{\infty\omega}^\omega$, which rules out the implementation even in partial fixed-point logic. In particular, the operator $\text{pfp}_{\text{PTIME}}$, a partial fixed-point operator with the semantic restriction to converge in a polynomial number of steps (cf. [EF99, Section 8.4.1]), is not sufficient to formalize the canonization procedure – even though it shares the idea with $\tilde{\text{CPT}}+\text{C}$ to impose an *external semantic restriction* in order to guarantee evaluation in PTIME.

The canonization procedure which will prove Theorems 3.1 and 3.2 is based on Weisfeiler-Lehman color refinement, combined with the notion of so-called *sections of graphs* which allow for a divide-and-conquer approach. Sections of undirected graphs were introduced by Goldberg in [Gol83] and our notion of a section is a generalization of Goldberg's definition to edge-colored directed graphs. Our sections will be defined in Section 3.3 after introducing the relevant notions of Choiceless Polynomial Time and color refinement in Section 3.2. Section 3.3 also shows how to find sections in edge-colored graphs, which is used by the canonization algorithm described in Section 3.4. The algorithm follows the approach of graph canonization taken in [CG84], but we already incorporate the necessary modifications to cast the procedure in $\tilde{\text{CPT}}+\text{C}$. The time and space complexity of the procedure hinges on the size of a central data structure called the *recursion tree* \mathcal{T} . Section 3.5 establishes the required singly exponential bound on the size of \mathcal{T} . We keep our exposition generic enough to prove Theorems 3.1 and 3.2 at the same time.

3.1 Isomorphism algorithms for edge-colored graphs

It appears that isomorphism algorithms specifically for edge-colored directed graphs have not been considered before. In this section we therefore discuss how isomorphism algorithms for different types of structures can be applied to edge-colored directed graphs. First, we consider an encoding of edge-colored directed graphs in undirected graphs in Section 3.1.1. In Section 3.1.2 we then discuss isomorphism algorithms for graphs and hypergraphs.

3.1.1 Encoding edge-colored directed graphs as undirected graphs

Let $G = (V, D)$ be a directed graph with an edge coloring $d : D \rightarrow \mathbb{N}$ and a vertex coloring $c : V \rightarrow \mathbb{N}$. Our goal is to construct a canonical undirected and uncolored graph $G^\circ = (V^\circ, E)$ from G . Figure 3.1 illustrates our construction.

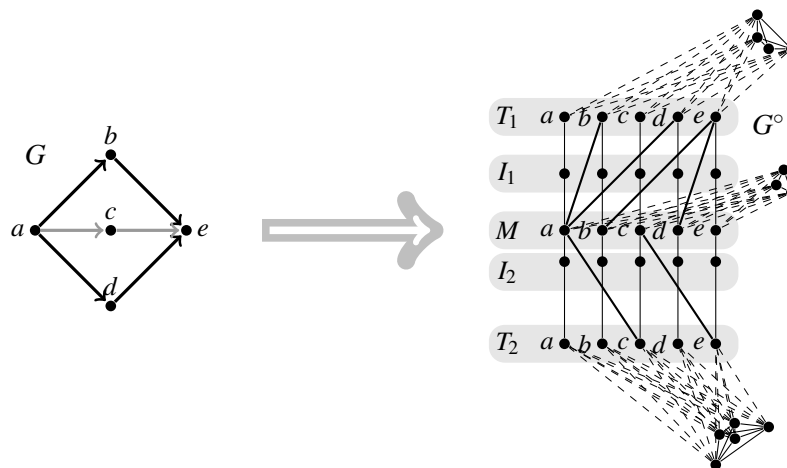


Figure 3.1: The encoding of an edge-colored directed graph G into an undirected graph G° . The sample graph G has 2 edge colors (black and gray) and no vertex coloring. The edges of G° are uncolored; their different styles are only for better legibility.

Proposition 3.1.1. *Given a directed graph G with ℓ edge colors and k vertex colors, we can construct an undirected graph G° with $O(k\ell^2n)$ vertices and with the property that edge-and-vertex-colored directed graphs G and H are isomorphic if and only if $G^\circ \cong H^\circ$.*

Proof. The first step is to eliminate the colors and the orientation of edges. Let D_1, \dots, D_ℓ denote the edge color classes partitioning D . For each edge color class D_i , $i \in [\ell]$, let T_i and I_i be identical copies of V . Let M be another copy of V . Let V^c be the disjoint union of M , all the T_i and all the I_i . We define the *undirected* edge relation E^c on V^c as follows:

- If $uv \in D_i$, then $u \in M$ and $v \in T_i$ are joined by an edge and
- for all $i \in [\ell]$, each $v \in I_i$ is joined with $v \in M$ and $v \in T_i$.

Finally, we color the vertices in V^c . Let \hat{c} be the vertex coloring of V^c which colors all the vertices in M with a new special color and which assigns to all the vertices in T_i the same color that D_i received under d . The vertices in all the I_i remain uncolored.

Separately, let c' be the coloring of V^c which colors the vertices in M in the same way that c colors V . The vertices in all the T_i and I_i remain uncolored by c' . We let the final coloring of V^c be $\hat{c} \times c'$, the refinement of \hat{c} by c' (see Section 2.1.1).

We used two simple ideas in the construction of $G^c = (V^c, E^c)$ from G . Firstly, directed edges are represented by undirected edges from M to T_i (for some $i \in [\ell]$). The sets M and T_i receive different colors so that any automorphism of G^c has to fix them as sets. Secondly, as M and T_i contain distinct copies of each $v \in V$, the set I_i with its edges to M and T_i serves to identify the two copies of each $v \in V$. We note that, in order for this construction to be canonical, the special color assigned to the vertices in M must always be the same, regardless of the specific graph G we consider.

Claim 3.1.2. Two directed graphs G and H with edge and vertex colorings are isomorphic if and only if G^c and H^c are color isomorphic.

Proof. It is easy to see that G^c can be defined in FP+C, where the number sort is used to define the various distinct copies of V . Similarly, it is not hard to verify that G can be reconstructed from G^c in FP+C. As all FP+C-definable relations are order-invariant, the claim follows. \square

Now we construct $G^\circ = (V^\circ, E^\circ)$ from $G^c = (V^c, E^c)$ by eliminating the vertex coloring of G^c . We use gadgets to identify the color classes. Let C_1, \dots, C_k be the color classes of G^c 's vertices. For each $i \in [k]$, let N_i consist of $i + 2$ new vertices. Let V° be the disjoint union of V^c with all the $N_i, i \in [k]$. Let E° be defined as follows. On V^c, E° equals E^c , each N_i forms a clique under E° , and if $v \in C_i$ then E° joins v and all vertices in N_i .

Observe that in V^c , all color class are independent sets and G^c is tripartite. Thus, if none of the color classes $C_i, i \in [k]$ is empty, all maximal cliques of size $i + 3$ contain the set N_i and one of its neighbors. Note that we only used the ordered partition of G^c 's coloring in the construction of G° in order to keep the number of new vertices small. However, if the cardinalities of corresponding color classes in G^c and H^c match, then the above observation implies that G^c and H^c are color isomorphic if and only if $G^\circ \cong H^\circ$.

If G has n vertices, ℓ edge colors, and k vertex colors, then G^c has $(2\ell + 1)n$ vertices and $k + \ell$ vertex colors. Consequently, G° has less than $(k + \ell + 2)^2 + (2\ell + 1)n$ vertices, which implies the rather crude but simple upper bound of $O(k\ell^2n)$. \square

Corollary 3.1.3. *If \mathcal{G} is a class of directed graphs with a bounded number of edge and vertex colors, then for each $G \in \mathcal{G}$, G° has $O(n)$ vertices.* \square

The encoding of the edge-colored directed graph G to G° given in the proof of Proposition 3.1.1 is easily seen to be definable in FP+C or $\tilde{C}PT+C$. The number sort is used to define additional vertices. Thus, it would be enough to show the capturing result in Theorem 3.2 for undirected graphs and the result for τ -structures of arity at most 2 would follow as τ fixes the number of edge and vertex colors. If the number of edge and vertex colors is not bounded, however, then G° may have $O(n^2)$ vertices or more. Under this encoding, no known graph isomorphism algorithm achieves the singly exponential run-time of edge-colored directed graphs provided by Theorem 3.1. This is why we take the approach of canonizing edge-colored directed graphs directly.

3.1.2 Existing isomorphism algorithms

The 1984 algorithm of Corneil and Goldberg [CG84] that we build on in this chapter appears to have received little echo in the graph isomorphism community. At the time, a larger effort was under way to establish group theoretic methods as tools for deciding graph isomorphism, following Luks's publication [Luk82] of a polynomial-time isomorphism procedure for graphs of bounded degree based on group theory. The refinement of these methods led to the fastest graph isomorphism algorithm known today, achieving the moderately exponential run-time of $2^{O(\sqrt{n \log n})}$ (cf. [BL83, BKL83]). The algorithm by Babai and Luks [BL83] also produces canonical forms and canonical orderings of graphs. In the subsequent development, the introduction of more elaborate group theoretic methods and of deep results such as the classification of the finite simple groups led to efficient parallel algorithms for isomorphism of graphs of bounded color classes and many related problems (see [BLS87, LM88]). As shown by Proposition 3.1.1 in the preceding section, directed graphs with a bounded number of edge-colors can be encoded as undirected graphs without increasing the number of vertices by more than a linear factor. Thus, Babai and Luks's algorithm implies a moderately exponential time canonization algorithm for directed graphs with a bounded number of edge and vertex color classes.

Instead of encoding edge-colored directed graphs as undirected graphs, we can also encode them as a structure of higher arity. All edge color classes can be encoded into a single 4-ary R which uses its first two variables to identify the color class.² While there is a large literature on

²On ordered structures, elements from the structure can be used to identify the color classes. This encoding is canonical since the edge colors were ordered. On unordered structures, new vertices from an ordered domain need to be used to identify the color classes.

the undirected graph isomorphism problem, much less work has been done for general structures. The first approach would be, yet again, to encode general structures as undirected graphs. For example, Miller [Mil79] achieves an encoding of τ -structures by introducing a gadget for each membership $\vec{x} \in R$ (for each $R \in \tau$). If R is of arity k , then R may hold for up to n^k tuples, so the resulting graph has $O(n^k)$ vertices. Clearly, such a polynomial increase in the size of the vertex set cannot be completely avoided. Therefore, the run-time of the above-mentioned graph isomorphism algorithms deteriorates with increasing arity of the relations we consider, e.g., under Miller's encoding it is $2^{n^{k/2+o(1)}}$ for encodings of structures of arity k . For edge-colored directed graphs encoded as 4-ary structures as above, this does not provide us with a desirable run-time.

Instead of encoding general structures as unordered graphs, we may use algorithms for hypergraph isomorphism for general structures. In 1999, Luks presented the first $2^{O(n)}$ isomorphism algorithm for n -vertex hypergraphs based on only a few basic concepts from the algorithmic group theory toolbox [Luk99]. An *edge* E of a hypergraph on vertex set V is any non-empty subset of V . Luks's algorithm may not be applied directly to general structures, since relations are sets of *ordered tuples*, while hypergraph edges are unordered sets. Yet, we can encode the order of k -tuples by forming hyperedges over $V \times [k]$ instead and using gadgets to pin down the slices $V \times \{i\}$ and $\{v\} \times [k]$ for $i \in [k]$ and $v \in V$. Given that a vocabulary τ fixes the arity of its relations, Luks's result implies a singly exponential time isomorphism algorithm for general structures in this way.

Just recently, Babai and Codenotti [BC08] gave a moderately exponential time ($2^{O(\sqrt{n} \log^{O(1)} n)}$) algorithm for hypergraphs of bounded rank, i.e., hypergraphs whose edges have cardinality bounded by some fixed $k \in \mathbb{N}$. Babai and Codenotti's algorithm retains its moderately exponential running time under the encoding mentioned above, therefore deciding arity-2 structure isomorphism faster than our algorithm. It is unknown, however, whether the two hypergraph isomorphism procedures can be modified to produce *canonical forms* of hypergraphs, which is a drawback common to many group theory-based isomorphism procedures as they rely on pre-defined orderings for such tasks as finding generators of the graph's automorphism group.

We note that for a relation R of arity k over a set A , there is a simple encoding of R as a directed graph over $\lceil k/2 \rceil$ -tuples from A , using vertex colorings and additional edge relations to identify tuples that share the same vertex in certain places. In this way, our canonization scheme yields a $2^{O(n^{\lceil k/2 \rceil})}$ -time structure canonization algorithm. Of course, for the purpose of deciding structure isomorphism, this procedure has no advantage over directly quantifying over all orders when $k \geq 3$. It is an open problem whether there is a $2^{O(n)}$ -time canonization algorithm for general structures.

3.2 Basic notions

3.2.1 Choiceless Polynomial Time

The logic Choiceless Polynomial Time ($\tilde{\text{CPT}}$) was introduced by Blass, Gurevich, and Shelah in [BGS99] on the basis of the computational model of *abstract state machines* (ASMs), which was chiefly developed by Gurevich (cf. [Gur94, GKOT00], also see [Rei08b]). In fact, calling $\tilde{\text{CPT}}$ a logic affords a broad view of what constitutes a logic, since the ASM framework is rather a computational model than a formal language. Given that a logic for PTIME, if it exists at all, might not be confinable to a sleek formalism, such a broad view seems advisable. Nash, Remmel, and Vianu [NRV05] discuss in detail the possible shapes of languages for PTIME and put down conditions which are desirable to be satisfied by any candidate logic. $\tilde{\text{CPT}}$ (and also $\tilde{\text{CPT}}+\text{C}$) satisfies them all: the $\tilde{\text{CPT}}$ programs are effectively enumerable, they define isomorphism-invariant properties of structures over any finite vocabulary, and there is a Turing machine evaluating $\tilde{\text{CPT}}$ programs such that an explicit polynomial time-bound for the simulation can be extracted effi-

ciently from the $\tilde{\text{CPT}}$ program. Therefore, we feel justified to speak of $\tilde{\text{CPT}}$ and $\tilde{\text{CPT}}+\text{C}$ as logics.

Given a structure \mathcal{A} , the basic idea is that a $\tilde{\text{CPT}}$ program transforms \mathcal{A} into another structure \mathcal{A}' . Repeated application of the program produces a sequence of structures $\mathcal{A}^0, \mathcal{A}^1, \dots, \mathcal{A}^k$, where \mathcal{A}^k is a fixed-point of the program, and the program is restricted in a way so that k is polynomial in $|\mathcal{A}|$. The structures \mathcal{A}^0 to \mathcal{A}^k are also called *states* of the $\tilde{\text{CPT}}$ program. In each step, the program is allowed to manipulate certain *dynamic functions* inside the structure, and the interpretation of these dynamic functions in the final state \mathcal{A}^k is considered the result of the $\tilde{\text{CPT}}$ program. In this way, a $\tilde{\text{CPT}}$ program defines a query on structures \mathcal{Q} by designating a nullary function c (i.e. a constant) so that $\mathcal{A} \in \mathcal{Q}$ if and only if c is *true* in \mathcal{A}^k . Similarly, $\tilde{\text{CPT}}$ programs may define a relation R by means of a function f , letting $\vec{v} \in R \leftrightarrow f(\vec{v}) = \text{true}$.

In the following, we define $\tilde{\text{CPT}}$ by explaining the roles of *hereditarily finite sets*, *static* and *dynamic functions*, *states*, *terms*, *rules*, *programs*, and their semantics. The extension of Choiceless Polynomial Time with counting ($\tilde{\text{CPT}}+\text{C}$) will then be obtained from $\tilde{\text{CPT}}$ by adjoining an additional number sort and counting operators in much the same way as for counting logics. For a thorough discussion of the concepts introduced here, the reader is referred to [BGS99], [BGS02], and [DRR08].

Hereditarily finite sets. Let A be a finite set. We define the set $\mathcal{HF}(A)$ to be the least set such that $A \subseteq \mathcal{HF}(A)$ and every finite subset of $\mathcal{HF}(A)$ is also a member of $\mathcal{HF}(A)$. $\mathcal{HF}(A)$ can also be defined recursively in an equivalent way as the closure of A under the following operation: if elements x_1, \dots, x_n are in the set and $n \in \mathbb{N}_0$, then also put $\{x_1, \dots, x_n\}$ into the set. We refer to the elements in A as *atoms* and call $\mathcal{HF}(A)$ the set of *hereditarily finite sets* built from A .

For example, if $A = \{a, b\}$, then the sets \emptyset , $\{\{a\}\}$, and $\{a, \{a, b\}\}$ are hereditarily finite sets over A . Notice that while $\mathcal{HF}(A)$ is of countably infinite cardinality, all of its members are either atoms or finite sets.

Given a structure $\mathcal{A} = (A, (R^A)_{R \in \tau})$ over some finite vocabulary τ , the first thing we do to prepare \mathcal{A} for the run of a $\tilde{\text{CPT}}$ program is to replace its universe with *all hereditarily finite sets built from A* , writing $\mathcal{A}' = (\mathcal{HF}(A), (R^{\mathcal{A}'})_{R \in \tau})$. The relations $R^{\mathcal{A}'}$ are only defined on the atoms $A \subseteq \mathcal{HF}(A)$ so that $\mathcal{A}'|_A \cong \mathcal{A}$ and they are considered *false* for all tuples with values from $\mathcal{HF}(A) \setminus A$. $\tilde{\text{CPT}}$ programs will later be able to define functions also over sets in $\mathcal{HF}(A)$.

In order to set bounds on what $\tilde{\text{CPT}}$ programs may define over $\mathcal{HF}(A)$, we still need the following notions. A set S is called *transitive* if $x \in y \in S$ implies $x \in S$. If X is any set, then we let $\text{TSC}(X)$ be the least transitive set containing X , and we call $\text{TSC}(X)$ the *transitive subset closure* of X . We caution the reader not to confuse the transitive subset closure of a set defined here with the *transitive closure operator* tc as defined earlier in the context of transitive closure logic.

If A is a finite set and $X \in \mathcal{HF}(A)$, then clearly also $\text{TSC}(X) \in \mathcal{HF}(A)$. $\text{TSC}(X)$ can also be defined recursively from X as the closure under the following operation: if $y \in X$ and $x \in y$, then also put x into X .

Ordered pairs of elements (x, y) are represented as elements from $\mathcal{HF}(A)$ by using the standard Kuratowski coding as $\{x, \{x, y\}\}$. In general, *k-tuples* (x_1, \dots, x_k) are inductively encoded as $\{x_1, \{x_1, (x_2, \dots, x_k)\}\}$. All typical operations on tuples, such as concatenation and element extraction, are $\tilde{\text{CPT}}$ -definable (cp. [DRR08]).

$\tilde{\text{CPT}}$ vocabularies and states. In order to handle hereditarily finite sets and define properties of an input structure \mathcal{A} , a $\tilde{\text{CPT}}$ program Π needs an extended vocabulary π containing both function and relation symbols. Nullary functions are also called constants. Each symbol in π is marked as either *static* or *dynamic*. Unlike static symbols, dynamic functions and relations may be modified by the $\tilde{\text{CPT}}$ program. We give a complete list of the content of π , explaining the semantics where it is not obvious:

3 Capturing Polynomial Time on Fragments of Logarithmic Size

- **Logical symbols:** the equality relation, Boolean constants *true* and *false*, the usual Boolean relations \wedge, \vee, \neg . All logical symbols are static.
- **Set-theoretic symbols:** the binary set inclusion relation \in , the constant \emptyset , and the constant Atoms interpreted as $A \in \mathcal{HF}(A)$. Additionally, the unary union function \bigcup for which $\bigcup X = \{x \mid \exists y \in X \text{ s.t. } x \in y\}$, a unary function ι for which $\iota(X) = x$ if $X = \{x\}$, and $\iota(X) = \emptyset$ otherwise, and a binary set constructor function $\{\cdot, \cdot\}$. Set-theoretic symbols are all static.
- **Dynamic functions:** π always contains the dynamic Boolean constants Halt and Output. Additionally, π may contain a finite number of dynamic functions of any arity.

We implicitly assume that the Boolean domain containing the truth values *true* and *false* is adjoint to the structure. With this in mind, it is convenient to view relations as functions which may only take Boolean values. In this way, dynamic functions can also act as relations and we will only need to describe how $\tilde{\text{CPT}}$ programs manipulate functions.³

Given an input structure $\mathcal{A} = (A, (R^A)_{R \in \tau})$ over a vocabulary τ , a *state* is a $\tau \cup \pi$ -structure $\mathcal{S} = (\mathcal{HF}(A), (R^{\mathcal{S}})_{R \in \tau}, (f^{\mathcal{S}})_{f \in \pi})$ where $R^{\mathcal{S}} = R^A$ is only interpreted over $A \in \mathcal{HF}(A)$, all logical symbols are interpreted over the Boolean domain in the obvious way, and all set-theoretic symbols are interpreted over $\mathcal{HF}(A)$ in the way explained above. The relations in τ are treated as static. The structure $\mathcal{A}^0 = (\mathcal{HF}(A), (R^{A^0})_{R \in \tau}, (f^{A^0})_{f \in \pi})$ is called the *initial state* of the $\tilde{\text{CPT}}$ program Π if in addition to being a state, all dynamic functions are set to \emptyset for all tuples from $\mathcal{HF}(A)$.

Terms and interpretations. As in most definitions of formal languages, every variable is a term. If $f \in \pi$ is a function of arity k and t_1, \dots, t_k are terms, then $f(t_1, \dots, t_k)$ is a term. If f is a relation of the same arity, then $f(t_1, \dots, t_k)$ is a *Boolean term*. The free variables in $f(t_1, \dots, t_k)$ are the union of the free variables in t_1, \dots, t_k .

Additionally, $\tilde{\text{CPT}}$ may use a type of guarded set construction called *comprehension*. If $t(v)$ is a term, r is a term without free occurrences of v , and $g(v)$ is a Boolean term, then

$$\{t(v) : v \in r : g(v)\}$$

is a term. The free variables are $\text{free}(t) \cup \text{free}(r) \cup \text{free}(g) \setminus \{v\}$. The semantics of this term is the set of $t(v)$ for which $v \in r$ and $g(v)$ holds.

If \mathcal{S} is a state and t is a term, then we write $\llbracket t \rrbracket^{\mathcal{S}}$ for the denotation of t in \mathcal{S} . For the term construction above, we would therefore write $\llbracket \{t(v) : v \in r : g(v)\} \rrbracket^{\mathcal{S}} = \{\llbracket t \rrbracket^{\mathcal{S}[a/v]} \mid a \in \llbracket r \rrbracket^{\mathcal{S}} \text{ and } \llbracket g \rrbracket^{\mathcal{S}[a/v]} = \text{true}\}$. The notation with the double braces $\llbracket \cdot \rrbracket$ conforms to common notation for ASMs and emphasizes that we are dealing with the state of a $\tilde{\text{CPT}}$ program instead of just a plain structure.

Rules, programs, and runs. Rules are the *instructions* of a $\tilde{\text{CPT}}$ program. The semantics of a rule R in a state \mathcal{S} is a set $\llbracket R \rrbracket^{\mathcal{S}}$ of *updates* to the dynamic functions in π . The updates determine the interpretations of the dynamic functions in the successor state of \mathcal{S} . When f is a dynamic function, we write $\langle f, \vec{a}, b \rangle$ to denote the update which modifies $f(\vec{a})$ to the effect that $f(\vec{a}) = b$.

$\tilde{\text{CPT}}$ programs allow the following rules:

- Update rule $f(\vec{t}) := t'$, where $f \in \pi$ is a dynamic function, t' is a term, and \vec{t} is a tuple of terms of the same arity as f . Let $\llbracket f(\vec{t}) := t' \rrbracket^{\mathcal{S}} := \{\langle f, \llbracket \vec{t} \rrbracket^{\mathcal{S}}, \llbracket t' \rrbracket^{\mathcal{S}} \rangle\}$

³In fact, it is not important how exactly we deal with Boolean values. Dawar, Richerby, and Rossman use $\emptyset, \{\emptyset\} \in \mathcal{HF}(A)$ in order to represent *false* and *true* [DRR08].

- Conditional rule `if t then R1 else R2`, where t is a term and R_1, R_2 are rules. Its semantics is $\llbracket R_1 \rrbracket^{\mathcal{S}}$ if $\llbracket t \rrbracket^{\mathcal{S}} = \text{true}$ and $\llbracket R_2 \rrbracket^{\mathcal{S}}$ otherwise.
- Parallel execution rule `do forall v ∈ t R`, where v is a variable, t a term in which v is not free, and R is a rule. v occurs bound in this rule. Let $\llbracket \text{do forall } v \in t R \rrbracket^{\mathcal{S}} := \bigcup \{ \llbracket R \rrbracket^{\mathcal{S}[a/v]} \mid a \in \llbracket t \rrbracket^{\mathcal{S}} \}$
- No-operation rule `Skip`. Of course, $\llbracket \text{Skip} \rrbracket^{\mathcal{S}} := \emptyset$

A set of updates *clashes* if it contains conflicting updates $\langle f, \vec{a}, b \rangle$ and $\langle f, \vec{a}, b' \rangle$ with $b \neq b'$. If R is a rule and \mathcal{S} is a state, the *successor state* \mathcal{S}' to \mathcal{S} is obtained by applying all the updates in $\llbracket R \rrbracket^{\mathcal{S}}$ to the dynamic functions in \mathcal{S} , unless $\llbracket R \rrbracket^{\mathcal{S}}$ clashes, in which case $\mathcal{S}' = \mathcal{S}$.

A *program* is a rule without free variables. Generically, a *run* of a program Π on an input structure \mathcal{A} is the finite or countable sequence of states $(\mathcal{A}^0, \mathcal{A}^1, \dots, \mathcal{A}^i, \dots)_{i < \kappa}$ where the initial state \mathcal{A}^0 is constructed as explained above, \mathcal{A}^{i+1} is the successor state to \mathcal{A}^i with respect to Π , and $\llbracket \text{Halt} \rrbracket^{\mathcal{A}^i} = \text{true}$ if and only if $i + 1 = \kappa$. To be precise, $\kappa \leq \omega$, where ω is the first infinite ordinal.

A run is called *terminating* if $\kappa < \omega$ so that the above sequence of states is finite. Note that if Π produces clashing updates at any stage then the run is non-terminating. We do not require runs at this point to end in a halting state since we aim at introducing external restrictions on the number of steps a run may have.

The rules described above may seem rather few, but it is easy to show that they suffice to construct many rules known from imperative programming, using auxiliary constants where necessary (see [BGS99, BGS02]). An exception is the classical `while`-loop, which we may not use as a rule, but one of which is simulated by the states evolving throughout the program's run.

Polynomial bounds. Given an input structure \mathcal{A} , we need to bound both the number of steps that a program's run may have and the number of objects it may manipulate throughout its run. For the former we just decree a polynomial bound, for the definition of the latter bound we need to be slightly more sophisticated.

An object $x \in \mathcal{HF}(A)$ is called *critical* at some stage of a run if it is in the range of a dynamic function f or part of a tuple \vec{a} which f maps to a value other than \emptyset . An object is *active* if it is in the transitive closure of some critical object.

Definition 3.2.1 ($\tilde{\text{CPT}}$ program). A $\tilde{\text{CPT}}$ program Π is a program in the sense above together with integer polynomials p and q . The run of the $\tilde{\text{CPT}}$ program Π on an input structure \mathcal{A} is the greatest initial segment of the run of Π on \mathcal{A} containing at most $p(|\mathcal{A}|)$ steps and $q(|\mathcal{A}|)$ active elements in total.

Π is said to *accept* \mathcal{A} if its run terminates with `Halt` and `Output` both set to *true*, and Π *rejects* \mathcal{A} if its run terminates with `Halt true` and `Output false`. Π is called *total* if for all possible input structures \mathcal{A} , Π either accepts or rejects \mathcal{A} . A class of τ -structures \mathcal{S} is $\tilde{\text{CPT}}$ -*definable* if there is a total $\tilde{\text{CPT}}$ program Π which accepts precisely the structures $\mathcal{A} \in \mathcal{S}$.

Choiceless Polynomial Time with counting ($\tilde{\text{CPT}}+\text{C}$). Finally, we augment $\tilde{\text{CPT}}$ with the ability to count. Doing this is sensible since $\tilde{\text{CPT}}$ alone lacks this ability due to its choiceless nature. For example, just like FP, $\tilde{\text{CPT}}$ cannot define the class of sets whose size is even (cf. [BGS99, Theorem 10]). Notice that since we can encode von Neumann ordinals over $\mathcal{HF}(A)$, the problem for $\tilde{\text{CPT}}$ is not the lack of an ordered domain to work with, but rather the transition from formulas to the corresponding number of satisfying assignments.

3 Capturing Polynomial Time on Fragments of Logarithmic Size

Unlike Blass, Gurevich, and Shelah in [BGS02], we do not use von Neumann ordinals to encode numbers, but we follow Dawar, Richerby, and Rossman [DRR08] in adjoining an ordered *number sort* to the structure in much the same way as for FP+C. This is done from the very start, i.e., instead of constructing the initial state \mathcal{A}^0 from the input \mathcal{A} , we construct it from the two-sorted structure $\mathcal{A}^+ = (A, N_A, (R^A)_{R \in \tau, \leq^{N_A}, +^{N_A}, \cdot^{N_A}})$. Here, N_A is an interval $[0, n] \subset \mathbb{N}_0$ whose size is polynomial in $|A|$ and will be specified precisely below. A $\tilde{\text{CPT}}+\text{C}$ program may then use $\mathcal{HF}(A \cup N_A)$ as its domain of hereditarily finite sets, and we observe that the type of atoms from $A \cup N_A$ can be told apart by checking if \leq is defined for them.

The remaining construction is the same as for $\tilde{\text{CPT}}$, except that we add one additional kind of term formation. If t is a term, then $\#t$ is a *counting term* denoting the cardinality of the set $\llbracket t \rrbracket^{\mathcal{S}}$ as an atom from N_A (and 0 if $\llbracket t \rrbracket^{\mathcal{S}}$ is not a set).⁴

Definition 3.2.2 ($\tilde{\text{CPT}}+\text{C}$ program). A $\tilde{\text{CPT}}+\text{C}$ program Π is a program as above together with integer polynomials p and q . Π may use counting terms and works over $\mathcal{HF}(A \cup N_A)$ when given an input structure \mathcal{A} , where $N_A = [0, q(|\mathcal{A}|)]$. As for $\tilde{\text{CPT}}$ programs, the run of Π on input \mathcal{A} is the greatest initial segment of the state sequence Π produces with at most $p(|\mathcal{A}|)$ steps and $q(|\mathcal{A}|)$ active elements in total. Acceptance and rejection are defined in the same way as for $\tilde{\text{CPT}}$ programs.

$\tilde{\text{CPT}}+\text{C}$'s ability to count enables us to add a *clock* to any $\tilde{\text{CPT}}+\text{C}$ program Π which ensures that Π always *terminates*, i.e., it either accepts or rejects any input structure and is therefore total (cf. [BGS02]). Thus, we may regard any $\tilde{\text{CPT}}+\text{C}$ program over a vocabulary τ as defining a query on τ -structures.

3.2.2 Properties of $\tilde{\text{CPT}}+\text{C}$

The name ‘‘choiceless polynomial time’’ suggests that $\tilde{\text{CPT}}$ does not exceed PTIME’s computational power. Indeed, the data complexity of $\tilde{\text{CPT}}+\text{C}$ is in polynomial time.

Theorem 3.2.3 ([BGS99, BGS02]). *Let τ be a vocabulary and let Π be a $\tilde{\text{CPT}}+\text{C}$ program which defines a class K of τ -structures. Then there is a polynomial time Turing machine M_Π which accepts precisely all the encodings of τ -structures from K . ■*

On the other hand, it was noted very early on that this computation model is not too weak. In particular, it includes all inflationary fixed-point operations.

Theorem 3.2.4 ([BGS02]). *Let φ be an FP+C-sentence. Then there is a $\tilde{\text{CPT}}+\text{C}$ program Π_φ which defines the same query as φ . ■*

In fact, $\tilde{\text{CPT}}+\text{C}$ is strictly more expressive than FP+C. Blass, Gurevich, and Shelah [BGS02] showed this by expressing a *padded version of the CFI-query* in $\tilde{\text{CPT}}$ (for the CFI-query, see Section 5.4). As noted in the introduction to this chapter, $\tilde{\text{CPT}}$ does profit from padding, so it was a relevant question whether $\tilde{\text{CPT}}$ is more expressive than FP *just* because of this, or also for other reasons. Dawar, Richerby, and Rossman [DRR08] settled this question by showing that the CFI-query without padding can also be defined in $\tilde{\text{CPT}}$.⁵ Already in [BGS99], it was noted that $\tilde{\text{CPT}}$ suffers from an inability to count in a similar way as FP does. The following theorem sums up these results.

⁴To be on the safe side, we could count modulo $|N_A|$ in order to handle the case that $|\llbracket t \rrbracket^{\mathcal{S}}| > \max N_A$. However, we will choose N_A large enough so that this case cannot arise.

⁵They were using the fact, however, that the CFI-query can be defined on the basis of *ordered graphs* which limits the degrees of freedom of the CFI-structures. Our approach using rank logics in Chapter 5 can define the CFI-query without this restriction.

Theorem 3.2.5. $\text{FP} \preceq \tilde{\text{CPT}} \preceq \tilde{\text{CPT}}+\text{C}$ and $\text{FP}+\text{C} \preceq \tilde{\text{CPT}}+\text{C}$. ■

Our result in this chapter says that by padding an n -vertex edge-colored directed graph G with $2^{O(n)}$ isolated vertices, all polynomial-time properties of G become definable in $\tilde{\text{CPT}}+\text{C}$. Since $\text{FP}+\text{C} \preceq \tilde{\text{CPT}}+\text{C}$ (Theorem 3.2.4), it is enough to $\tilde{\text{CPT}}+\text{C}$ -define a canonical form of G and then employ the Immerman-Vardi Theorem 2.3.10 to define any PTIME query. As noted already by Blass, Gurevich, and Shelah [BGS99], if we just allow for enough padding, we can directly quantify over all the structure's orderings.

Theorem 3.2.6. *Let τ be a finite vocabulary which contains a unary relation symbol U . Let $c \in \mathbb{N}$. Then $\tilde{\text{CPT}}$ expresses all PTIME-properties of the restriction of \mathcal{A} to U on the class of all τ -structures \mathcal{A} with $|U|! \leq |\mathcal{A}|^c$.* ■

The bound on $|U|$ in Theorem 3.2.6 is more restrictive than our bound of $2^{|U|} \leq |\mathcal{A}|^c$ from Theorem 3.2. So our result is an improvement of Theorem 3.2.6 on structures of arity 2 when passing from $\tilde{\text{CPT}}$ to $\tilde{\text{CPT}}+\text{C}$.

3.2.3 Stable colorings and Weisfeiler-Lehman color refinement

The classical method of color refinement is a simple combinatorial approach to distinguishing non-isomorphic graphs. It starts by coloring the vertices of graphs G and H with their respective degrees. Any isomorphism $\varphi : G \rightarrow H$ has to respect these color classes, so if the cardinalities of the color classes in G and H do not match, then G and H cannot be isomorphic. What is more, if φ is an isomorphism and $\varphi(u) = v \in V_H$ for some $u \in V_G$, then u and v must also have equal numbers of neighbors from each color class. Thus, it makes sense to distribute the information of neighboring color classes throughout the graph, splitting up color classes based on the colors of their neighbors. The hope is that, after this *refinement process* has stabilized, the graph G is divided up into singleton color classes, so that H is isomorphic to G if and only if its color refinement process produces the same singleton color classes. Of course, this outcome cannot be guaranteed: regular graphs, for example, fool this particular procedure. Still, classical color refinement is a strong tool on most graphs: Babai, Erdős, and Selkow show in [BES80] that color refinement results in singleton color classes on *almost all graphs*, meaning that for $n \rightarrow \infty$ the ratio of graphs of size n for which this refinement is successful approaches 1.

Classical color refinement is usually attributed to Weisfeiler and Lehman who published a detailed investigation of it in 1968 [WL68] (see also [Wei76]). We refer to [CFI92] for an excellent account of the history of the color refinement method.

Corneil and Goldberg's graph canonization algorithm uses classical color refinement as a subroutine. For our generalization of their method, we need a similar procedure for directed graphs. We could define such color refinement by treating the neighbors at incoming and at outgoing edges separately. Instead, we choose to employ an established procedure known as *2-dimensional Weisfeiler-Lehman color refinement*, abbreviated as 2-dim WL. The goal is to produce *stable colorings* of the graph's vertices. We first define such stable colorings directly by the core property which will be used in the analysis of our canonization. Afterwards, we will define 2-dim WL and show that it computes stable colorings in the sense of our definition. Recall from Section 2.1.1 that we deliberately blur the distinction between colorings and ordered partitions.

Definition 3.2.7 (Stable coloring/ stable partition). Let τ be a finite vocabulary with relations of arity at most 2, let $\mathcal{A} = (A, (R^{\mathcal{A}})_{R \in \tau})$ be a τ -structure, and let $R \in \tau$ be a binary relation. Let \mathcal{P} be a coloring of A which partitions A into sets (P_1, \dots, P_ℓ) . \mathcal{P} is called *R -stable* if for all indices $i, j \in [\ell]$, the numbers

$$n_{ij}^0 := |\{v \in P_j \mid (\hat{u}, v) \in R^{\mathcal{A}}\}| \quad \text{and} \quad n_{ij}^1 := |\{u \in P_i \mid (u, \hat{v}) \in R^{\mathcal{A}}\}|$$

3 Capturing Polynomial Time on Fragments of Logarithmic Size

are invariant under the choice of $\hat{u} \in P_i$ and $\hat{v} \in P_j$. \mathcal{P} is called *stable* if it is R -stable for all binary relations $R \in \tau$ and it is a refinement of the coloring induced by the unary relations in τ .

The intuition behind stable colorings is that the vertices of any given color class look indistinguishable with respect to any relation $R \in \tau$. Notice, however, that this does not mean that the exchange of elements within a color class induces an isomorphism of the structure. In particular, the stable colorings computed by 2-dim WL do not necessarily have this property (cp. [CFI92]). Our canonization procedure overcomes this shortcoming, but only by performing exponentially many steps in the worst case.

Remark 3.2.8. For our canonization procedure it is actually immaterial which kind of color refinement is used as long as it can be implemented in $\tilde{\text{CPT}}+\text{C}$ and it produces stable colorings in the sense of Definition 3.2.7. In particular, we could just as well use k -dimensional Weisfeiler-Lehman color refinement for any $k > 2$ (cf. [CFI92]).

2-dimensional Weisfeiler-Lehman We follow the exposition of [CFI92] in defining the 2-dimensional Weisfeiler-Lehman color refinement method. As the name suggests, the procedure natively deals with *colored graphs* instead of plain graphs. We will use this feature in our canonization procedure when we re-run color refinement after individualizing a vertex (see Section 3.2.4).

When first considering an uncolored directed graph $\mathcal{A} = (A, R)$, we give it an *initial coloring* c by coloring every pair $(u, v) \in A^2$ with its *isomorphism type*. Over directed graphs with a single edge relation R , the isomorphism type consists of the information whether $u = v$ or $u \neq v$ and whether $(u, v) \in R$ or $(u, v) \notin R$. The initial coloring $c = c_R$ then corresponds to the ordered partition

$$\left(\left\{ (u, u) \in A^2 \mid (u, u) \in R \right\}, \left\{ (u, u) \in A^2 \mid (u, u) \notin R \right\}, \right. \\ \left. \left\{ (u, v) \in A^2 \mid u \neq v \text{ and } (u, v) \in R \right\}, \left\{ (u, v) \in A^2 \mid u \neq v \text{ and } (u, v) \notin R \right\} \right).$$

When there are several edge relations or several edge color classes R_1, \dots, R_k , then we let the initial coloring be $c := c_{R_1} \times c_{R_2} \times \dots \times c_{R_k}$ (cf. Section 2.1.1). When the directed graph \mathcal{A} already comes with a vertex coloring $e : A \rightarrow \mathbb{N}$, then $e^2 : (u, v) \mapsto (e(u), e(v))$ is a coloring of A^2 with the property that $e(u) < e(v) \Rightarrow e^2(u, u) <_{\text{lex}} e^2(v, v)$. We initialize \mathcal{A} 's coloring with the refinement of e^2 by c (again, cf. Section 2.1.1).

Now let \mathcal{A} be an edge-colored directed graph whose vertex set A has received its initial coloring as described. Each *color refinement step* consist of the following operation: for any pair $(u, v) \in A^2$, let $c(u, v)$ denote its color class, and let $X(u, v)$ be the $|A|$ -element multiset consisting of pairs of colors

$$(c(x, v), c(u, x)),$$

one for each element $x \in A$. We assign to every (u, v) a new color $(c(u, v), X(u, v))$. Since the tuples' old colors are part of the new color which they are assigned, the new coloring is a refinement of the old coloring. As the old colors were ordered, the new colors are linearly ordered by lexicographical comparison. For the next step, it is enough to remember the ordered partition induced by the new color classes instead of keeping track of the bulky new color names which were introduced in this step. The color refinement step is repeated until no further refinement of color classes occurs.

Theorem 3.2.9 ([Ott97, Lemma 4.14]). *There is an FP+C-formula $\chi(\vec{x}, \vec{y})$ defining the strict weak ordering of the color classes resulting from the Weisfeiler-Lehman method. That means, if*

$\mathcal{W} = (W_1, \dots, W_\ell)$ is the final 2-dim WL coloring of $U(\mathcal{A})^2$, then $\mathcal{A} \models \chi[\vec{a}, \vec{b}]$ if and only if $\vec{a} \in W_i$ and $\vec{b} \in W_j$ with $i < j$. \blacksquare

Note that the color classes can be inferred from the formula χ of Theorem 3.2.9 since \vec{a} and \vec{b} receive the same color if neither $\mathcal{A} \models \chi[\vec{a}, \vec{b}]$ nor $\mathcal{A} \models \chi[\vec{b}, \vec{a}]$. By Theorem 3.2.4, χ is also $\tilde{\text{CPT}}+\text{C}$ -definable. As announced earlier, we now show that the colorings defined by 2-dim WL are stable in the sense of Definition 3.2.7, and therefore suitable for our purposes.

Lemma 3.2.10. *Let τ be a finite vocabulary with relations of arity at most 2 and let $\mathcal{A} = (A, (R^A)_{R \in \tau})$ be a τ -structure. For every pair $(u, v) \in A^2$, let $c(u, v)$ denote the final color that (u, v) receives in the 2-dimensional Weisfeiler-Lehman procedure. For each $u \in A$, we define the color of u as $c(u) := c(u, u)$. This coloring is stable in the sense of Definition 3.2.7.*

Proof. Let P_1, \dots, P_ℓ be the ordered list of colors partitioning A . If $(u, v) \in R^A$ and $(x, y) \notin R^A$ for any binary $R \in \tau$, then these two pairs receive different colors at initialization of the WL procedure. Similarly, if $u \in U^A$ and $v \notin U^A$ for any unary $U \in \tau$, then (u, u) and (v, v) receive different colors at initialization of the WL procedure. Thus, $c(u) \neq c(v)$ as the final coloring given by 2-dim WL is a refinement of the original one. Also, if $c(u) \neq c(x)$ and v, y are any elements in A , then one step of color refinement renders $c(u, v) \neq c(x, y)$ since the right-hand color list of (u, v) does not contain the color $c(x, x)$ (and the right-hand color list of (x, y) does). Similarly, $c(v, u) \neq c(y, x)$.

Suppose that there are a binary relation $R \in \tau$, $i, j \in [\ell]$, and $u, u' \in P_i$ so that u and u' have a different amount of incoming (or outgoing) R -neighbors in P_j . Then the left-hand (respectively right-hand) color lists of (u, u) and (u', u') contain refinements of $(P_j \times P_i) \cap R^A$ (respectively $(P_i \times P_j) \cap R^A$) an unequal number of times. Thus, 2-dim WL assigns different colors to u and u' , which contradicts our assumption that $u, u' \in P_i$. \square

Remark 3.2.11. Usually, the coloring of *pairs* computed by the 2-dim WL color refinement process is itself called a *stable coloring*. Lemma 3.2.10 assures us that every stable outcome of 2-dim WL is stable in the sense of Definition 3.2.7. Conversely, however, it is not necessarily true that any stable partition is immune to further refinement by 2-dim WL. The reason is that Definition 3.2.7 does not distinguish self-loops from edges between distinct vertices, while WL color refinement does. Although we will always refer to the conditions of Definition 3.2.7 when speaking of stable colorings, it is not wrong to think of stable colorings in the stronger sense as the result of 2-dim WL.

3.2.4 Individualization of vertices

Let \mathcal{A} be an edge-colored directed graph and let $\mathcal{P} = (P_1, \dots, P_\ell)$ be a coloring of its vertices. By the *individualization of a vertex x* we mean the process of assigning a new color to x which is not shared by any other vertex. Formally, if $x \in P_i$, the new coloring corresponds to the partition $(P_1, \dots, \{x\}, P_i \setminus \{x\}, \dots, P_\ell)$.

Individualization is a very simple idea to turn 2-dimensional Weisfeiler-Lehman into a reliable graph-isomorphism test: whenever color refinement stops short of totally partitioning the vertex set of G , select one color class C (e.g. the first) and a vertex $x \in C$, individualize x , and re-run the WL algorithm to obtain a stable coloring \mathcal{P}^* properly refining the original one. When testing whether two graphs G and H are isomorphic, we now have to do the same thing for all vertices x in the color class corresponding to C in H and compare the result to G with its coloring \mathcal{P}^* . Clearly, by individualizing at most n vertices, we always obtain a total partition of the vertex set. Unfortunately, there are instances in which we really need to individualize $\Omega(n)$ vertices (cf. [CFI92]). Testing isomorphism by individualizing all the vertices simply amounts to trying out

all possible linear orderings, so this naïve individualization procedure has a worst-case running time of $O(n!)$. In order to avoid this time complexity we combine the individualization strategy with the concept of sections, which are introduced in the upcoming section.

3.3 Sections of stable partitions

From now on, we assume that we are working over τ -structures $\mathcal{U} = (U, (R)_{R \in \tau})$, where τ is any vocabulary of arity at most 2. All the results are equally valid for general edge-colored directed graphs if we simply consider $(R)_{R \in \tau}$ to be the edge color classes listed in the order of the colors. All of the upcoming definability results hold regardless of the number of edge relations in such an ordered list. We choose to focus on τ -structures since this is the relevant framework for casting the canonization procedure in $\tilde{\text{CPT}}+\text{C}$.

3.3.1 Definition of sections

We use ordered partitions to represent colorings of subsets $V \subseteq U$ as described in Section 2.1.1. We will always consider the cells of \mathcal{P} to be ordered by index, i.e. $P_i \leq P_j \Leftrightarrow i \leq j$. A partition is *total* if all of its cells are singletons. A total ordered partition of V induces an ordering on V in the obvious manner. If $\mathcal{P} = (P_1, \dots, P_\ell)$ is a partition of V and $H \subseteq V$, then $H \cap \mathcal{P}$ denotes the partition of H given by $(H \cap P_1, \dots, H \cap P_\ell)$. $H \cap \mathcal{P}$ is called the *restriction of \mathcal{P} to H* . It is critical for our applications not to forget any coloring information when forming the restriction of \mathcal{P} to H . Therefore, we *do not eliminate any empty sets from $(H \cap P_1, \dots, H \cap P_\ell)$* so as the cells of \mathcal{P} faithfully correspond to the cells of $H \cap \mathcal{P}$.

Definition 3.3.1 (Section of a stable coloring). Let \mathcal{U} be a colored directed graph with vertex set U and edge relation R , and let \mathcal{P} be a stable coloring of U . For $H \subseteq U$, let $\sim_{\mathcal{P}}^H$ be the equivalence relation on U with $x \sim_{\mathcal{P}}^H y$ if and only if x, y have the same color and $x \in H \Leftrightarrow y \in H$. H is called a *section of \mathcal{P} in \mathcal{U}* if for all $(u, v), (u', v') \in U^2 \setminus (H^2 \cup (U \setminus H)^2)$ with $u \sim_{\mathcal{P}}^H u'$ and $v \sim_{\mathcal{P}}^H v'$, it holds that $(u, v) \in R^{\mathcal{U}}$ if and only if $(u', v') \in R^{\mathcal{U}}$.

If τ is a vocabulary of arity at most 2, \mathcal{U} is a τ -structure and \mathcal{P} is a stable coloring of U , then $H \subseteq U$ is a section of \mathcal{P} if it is a section with respect to all binary relations $R \in \tau$.

If $H = \emptyset$ or $H = U$ then we call H a *trivial section*. We are only interested in non-trivial sections. We also call a partition $\gamma = \{B_1, \dots, B_k\}$ of U a section if each B_i is a section of \mathcal{P} . The definition of a section is symmetric, in that a set $H \subseteq U$ is a section if and only if $U \setminus H$ is a section. Hence, $\{H, U \setminus H\}$ is also a section in this sense. Figure 3.2 illustrates the occurrence of a section in a graph. Further examples of sections can be found in Figures 3.3 and 3.4.

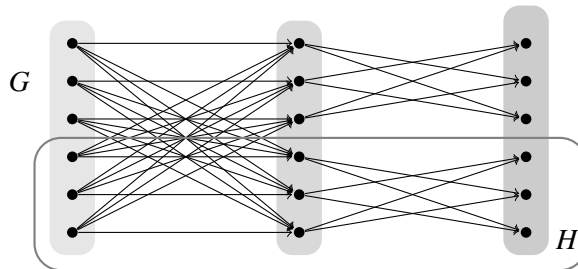


Figure 3.2: A directed graph G with a stable coloring and a section H .

In words, H is a section if for any present binary relation R and for any pair $(u, v) \in U^2$ for which neither $u, v \in H$ nor $u, v \in U \setminus H$, it holds that membership in $R^{\mathcal{U}}$ is completely determined

by the information in which part of $\mathcal{P} \cap H$ or $\mathcal{P} \cap (U \setminus H)$ the elements u, v are contained. Given this property, a section H allows us to decompose the graph into $G[H]$ and $G[U \setminus H]$. Once we have recursively obtained canonical forms for these two induced subgraphs, we can piece them back together and recover the edge relation between them by only looking at the vertices' color classes.

3.3.2 Finding sections

We first show how to find sections in directed graphs with just one binary relation R . Recall that in this situation, a partition $\mathcal{P} = (P_1, \dots, P_\ell)$ is called *stable* if for all indices $i, j \in [\ell]$ the numbers $n_{ij}^0 := |\{v \in P_j \mid (u, v) \in R^A\}|$ and $n_{ij}^1 := |\{u \in P_i \mid (u, v) \in R^A\}|$ do not depend on the choice of $u \in P_i$ and $v \in P_j$, respectively. By counting the edges from P_i to P_j in two different ways, we have

$$|R^A \cap (P_i \times P_j)| = n_{ij}^0 \cdot |P_i| = n_{ij}^1 \cdot |P_j| \quad \text{for all } j \in [k]. \quad (\blacktriangle)$$

Notice that $n_{ij}^0 > \frac{1}{2}|P_j|$ if and only if $n_{ij}^1 > \frac{1}{2}|P_i|$. We use this property to reduce the number of edges in the graph in a canonical way. Instead of $R^A \cap (P_i \times P_j)$, we also write $R|_{P_i \times P_j}$ to denote the restriction of R to $P_i \times P_j$.

Definition 3.3.2 (Switching equivalent graph). Given a stable partition $\mathcal{P} = (P_1, \dots, P_\ell)$ of \mathcal{U} , the *switching equivalent* relation $R_{\mathcal{P}}$ is defined as follows: for all indices $i, j \in [\ell]$, the restriction of $R_{\mathcal{P}}$ to $P_i \times P_j$ is defined as the complement of $R|_{P_i \times P_j}$ if $n_{ij}^0 > \frac{1}{2}|P_j|$; otherwise $R_{\mathcal{P}}|_{P_i \times P_j}$ is defined as $R|_{P_i \times P_j}$. We write $\mathcal{U}_{\mathcal{P}} := (U, R_{\mathcal{P}})$ and call $\mathcal{U}_{\mathcal{P}}$ the *switching equivalent graph* with respect to \mathcal{P} . By $G_{\mathcal{P}}$ we denote the undirected graph obtained from $\mathcal{U}_{\mathcal{P}}$ by forgetting the edges' directions.

Considering the switching equivalent graph and its undirected version will make it particularly easy to find sections. For this to work, we need the following lemma.

Lemma 3.3.3. *A set H is a section of \mathcal{P} in \mathcal{U} if and only if it is a section of the same partition in the switching equivalent graph $\mathcal{U}_{\mathcal{P}}$.*

Proof. For any cells $P_i, P_j \in \mathcal{P}$, $P_i \cap H$ is fully connected or fully disconnected to $P_j \setminus H$ in \mathcal{U} if and only if $P_i \cap H$ is either fully connected or fully disconnected to $P_j \setminus H$ in $\mathcal{U}_{\mathcal{P}}$. \square

Proposition 3.3.4. *Let $\mathcal{U} = (U, R)$ be a directed graph, and let \mathcal{P} be a stable partition of \mathcal{U} . If C is the union of connected components of $G_{\mathcal{P}}$, then C is a section.*

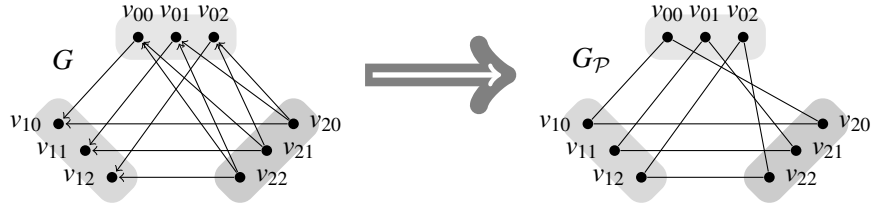
Proof. If C and $U \setminus C$ are fully disconnected in $G_{\mathcal{P}}$, then there is not a single pair from $U^2 \setminus (C^2 \cup (U \setminus C)^2)$ contained in $R_{\mathcal{P}}$. Thus, membership in $R_{\mathcal{P}}$ of pairs from $C \times (U \setminus C)$ and from $(U \setminus C) \times C$ is completely determined, therefore C is a section both in $\mathcal{U}_{\mathcal{P}}$ and in \mathcal{U} . \square

There is a partial converse to Proposition 3.3.4 (Lemma 3.3.6) and a characterization of sections in graphs whose switching equivalent graph is connected (Lemma 3.3.7). Figure 3.3 shows examples of these different types of sections. Before getting to these results, though, we first show how stable colorings often leave little choice in what a section might look like.

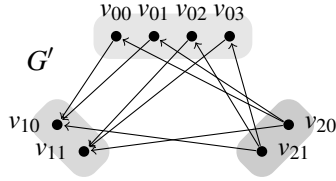
Lemma 3.3.5. *Let $\mathcal{P} = (P_1, \dots, P_\ell)$ be a coloring of U which is stable with respect to a binary relation R and let $i, j \in [\ell]$. Suppose that $n_{ij}^0 \notin \{0, \frac{1}{2}|P_j|, |P_j|\}$ and that H is a section of \mathcal{P} . Then $H \cap P_j = \{v \in P_j \mid \exists u \in H \cap P_i \text{ s.t. } (u, v) \in R_{\mathcal{P}}\}$.*

Proof. By Equation (\blacktriangle) , also $n_{ij}^1 \notin \{0, \frac{1}{2}|P_i|, |P_i|\}$. Let \tilde{n}_{ij}^0 and \tilde{n}_{ij}^1 denote the quantities of outgoing and incoming neighbors in the switching equivalent graph $\mathcal{U}_{\mathcal{P}}$. By our assumptions, $0 < \tilde{n}_{ij}^0 < \frac{1}{2}|P_j|$ and $0 < \tilde{n}_{ij}^1 < \frac{1}{2}|P_i|$. Let N_j be the set of vertices in P_j which have an incoming $R_{\mathcal{P}}$ -neighbor from $P_i \cap H$. We have to show that $H \cap P_j = N_j$.

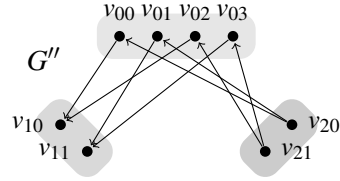
3 Capturing Polynomial Time on Fragments of Logarithmic Size



(a) A graph G and its switching equivalent graph $G_{\mathcal{P}}$. The connected components $(\{v_{00}, v_{10}, v_{20}\}, \{v_{01}, v_{11}, v_{21}\}, \{v_{02}, v_{12}, v_{22}\})$ of $G_{\mathcal{P}}$ form a section in G .



(b) A graph G' for which every $H \in \{(v_{00}, v_{01}), (v_{02}, v_{03})\} \times \{v_{10}, v_{11}\} \times \{v_{20}, v_{21}\}$ yields a section.



(c) A graph G'' without a section. Note the similarity to the CFI construction (cf. Section 5.4).

Figure 3.3: Sample graphs with different types of sections. All displayed colorings are stable.

First, suppose that there is $v \in N_j \setminus (H \cap P_j)$. Let $u \in P_i \cap H$ such that $(u, v) \in R_{\mathcal{P}}$ and recall the equivalence relation $\sim_{\mathcal{P}}^H$ on U whose equivalence classes are $(\mathcal{P} \cap H) \cup (\mathcal{P} \cap (U \setminus H))$. We denote by $[v]$ the equivalence class of $v \in U$ under $\sim_{\mathcal{P}}^H$. Since $(u, v) \in (P_i \cap H) \times (P_j \setminus H)$ and H is a section, we must have $[u] \times [v] \subseteq R_{\mathcal{P}}$. Thus, every vertex in P_i has at least $[v]$ $R_{\mathcal{P}}$ -neighbors, and by the assumptions of the lemma we have

$$|[v]| \cdot |P_i| \leq \tilde{n}_{ij}^0 \cdot |P_i| < \frac{1}{2} |P_j| |P_i|.$$

It follows that $[v] = |P_j \setminus H| \leq \tilde{n}_{ij}^0 < \frac{1}{2} |P_j|$. Consequently, $|P_j \cap H| > \frac{1}{2} |P_j|$ and no element from $P_i \setminus H$ can be $R_{\mathcal{P}}$ -connected to any element in $P_j \cap H$, since otherwise $P_i \setminus H$ and $P_j \cap H$ would have to be fully connected, which contradicts $\tilde{n}_{ij}^0 < \frac{1}{2} |P_j|$. But since \mathcal{P} is a stable coloring and $P_i \cap H$ is fully connected to $P_j \setminus H$, the same must then be true for $P_i \setminus H$. Thus, P_i is fully connected to $P_j \setminus H$, contradicting that $n_{ij}^1 \neq |P_i|$. This shows $N_j \subseteq P_j \cap H$.

Conversely, suppose that there is $v \in (P_j \cap H) \setminus N_j$. Since $\tilde{n}_{ij}^1 \neq 0$ and v does not have any incoming $R_{\mathcal{P}}$ -neighbor from $P_i \cap H$, there must be $u \in P_i \setminus H$ such that $(u, v) \in R_{\mathcal{P}}$. Recall that $U \setminus H$ is also a section of \mathcal{P} , so by what we have just shown above, the vertices in P_j which have an incoming $R_{\mathcal{P}}$ -neighbor from $P_i \cap (U \setminus H) = P_i \setminus H$ must all be contained in $P_j \cap (U \setminus H) = P_j \setminus H$. This contradiction shows that $P_j \cap H \subseteq N_j$, finishing the proof. \square

Lemma 3.3.6. *Let $\mathcal{U} = (U, R)$ be a directed graph, and let $\mathcal{P} = (P_1, \dots, P_\ell)$ be a stable partition of \mathcal{U} . Suppose that for all $i, j \in [\ell]$ we have $n_{ij}^0 \neq \frac{1}{2} |P_j|$. Then every section in \mathcal{U} is also the union of some connected components of $G_{\mathcal{P}}$.*

Proof. This type of section is illustrated in Figure 3.3(a). Let H be a section of \mathcal{P} in \mathcal{U} . By the assumption that $n_{ij}^0 \neq \frac{1}{2} |P_j|$ for all $i, j \in [\ell]$, Lemma 3.3.5 implies that there can be no $R_{\mathcal{P}}$ -edges in either direction between H and $U \setminus H$. \square

For the characterization of sections in the case when $G_{\mathcal{P}}$ is connected, we are going to broaden our view and consider general τ -structures of arity at most 2. For a τ -structure \mathcal{U} , a stable coloring \mathcal{P} , and binary $R \in \tau$, let \mathcal{U}^R be the directed graph with only the relation R . Let $R_{\mathcal{P}}$ be the corresponding relation in the switching equivalent graph $\mathcal{U}_{\mathcal{P}}^R$. For \mathcal{U} , we now let $G_{\mathcal{P}}$ be the undirected graph for which u, v are connected by an edge if there is $R \in \tau$ such that $uv \in R_{\mathcal{P}}$ or $vu \in R_{\mathcal{P}}$. Note that the edge set of $G_{\mathcal{P}}$ is simply the union of the edges of all the graphs $G_{\mathcal{P}}^R$.

Lemma 3.3.7. *Let τ be a vocabulary of arity at most 2, let \mathcal{U} be a τ -structure and let $\mathcal{P} = (P_1, \dots, P_{\ell})$ be a stable partition. Suppose that $G_{\mathcal{P}}$ is connected and that there are indices $i, j \in [\ell]$ such that for some relation $R \in \tau$ we have $n_{ij}^0 = \frac{1}{2}|P_j|$. If H is a non-trivial section of \mathcal{U} , then*

- $|P_i \cap H| = \frac{1}{2}|P_i|$ for all $i \in [\ell]$, and
- for any other non-trivial section Q and all $i \in [\ell]$ we either have $P_i \cap H = P_i \cap Q$ or $P_i \cap H = P_i \cap (U \setminus Q)$.

Proof. This type of section is illustrated in Figure 3.3(b). Again, let \tilde{n}_{xy}^0 denote the number of $\mathcal{U}_{\mathcal{P}}$ -edges going out of any vertex in P_x to P_y . Similarly, let \tilde{n}_{xy}^1 be the number of $\mathcal{U}_{\mathcal{P}}$ -edges coming in to any vertex in P_y from P_x . It holds that $\tilde{n}_{xy}^0 \leq \frac{1}{2}|P_y|$ and $\tilde{n}_{xy}^1 \leq \frac{1}{2}|P_x|$.

The proof is by induction on the indices. We write $h_i := |P_i \cap H|/|P_i|$ for all indices $i \in [\ell]$. Let $i, j \in [\ell]$ be indices such that $n_{ij}^0 = \frac{1}{2}|P_j|$. By Equation (▲), we also have $n_{ij}^1 = \frac{1}{2}|P_i|$ for some $R \in \tau$. Considering the P_j -neighbors of any $u \in P_i$ and the P_i -neighbors of any $v \in P_j$ it is easy to see that $h_i = h_j \in \{0, \frac{1}{2}, 1\}$ if H is really supposed to be a section. Additionally, if $h_i = h_j = \frac{1}{2}$, then $H \cap P_i$ must be either equal to $\{u \in P_i \mid (u, v) \in R\}$ or $P_i \setminus \{u \in P_i \mid (u, v) \in R\}$, and similarly $H \cap P_j$ equals either the vertices in P_j connected to u or those disconnected from u with respect to R . Set $I := \{i, j\}$.

As long as $I \neq [\ell]$, let $x, y \in [\ell]$ be indices such that $x \in I$, $y \notin I$, and P_x and P_y are connected by an edge in $G_{\mathcal{P}}$. Such indices x, y must exist since $G_{\mathcal{P}}$ is connected. Without loss of generality assume that there is an $R_{\mathcal{P}}$ -edge from P_x to P_y so that $n_{xy}^0|P_x| = n_{xy}^1|P_y| \notin \{0, |P_x| \cdot |P_y|\}$ for some $R \in \tau$. If $h_x \in \{0, 1\}$, then it is clear that $h_y = h_x$ by the properties of sections and the fact that P_x is neither fully connected nor fully disconnected to P_y . If $h_x = \frac{1}{2}$, then there are two cases to consider.

Firstly, if $\tilde{n}_{xy}^1 < \frac{1}{2}|P_x|$, then by Lemma 3.3.5 $P_y \cap H = \{v \in P_y \mid \exists u \in P_x \cap H \text{ s.t. } (u, v) \in R_{\mathcal{P}}\}$. Thus, counting $R_{\mathcal{P}}$ -edges, we have that $\tilde{n}_{xy}^0 \cdot |P_x \cap H| = \tilde{n}_{xy}^1 \cdot |P_y \cap H|$ and $\tilde{n}_{xy}^0 \cdot |P_x| = \tilde{n}_{xy}^1 \cdot |P_y|$. It follows that $h_y = |P_y \cap H|/|P_y| = |P_x \cap H|/|P_x| = h_x = \frac{1}{2}$.

The second case is that $\tilde{n}_{xy}^1 = \frac{1}{2}|P_x|$. Then $\tilde{n}_{xy}^0 = \frac{1}{2}|P_y|$, so any $u \in P_x$ is $R_{\mathcal{P}}$ -connected to precisely half of P_y . It follows directly that $h_y = h_x = \frac{1}{2}$, and H either equals the neighborhood of u in P_y or its complement in P_y . The inductive step is concluded by adding y to I .

This shows that $h_1 = h_2 = \dots = h_{\ell} \in \{0, \frac{1}{2}, 1\}$. Hence, H is either a trivial section or we have $|P_i \cap H| = \frac{1}{2}|P_i|$ for all $i \in [\ell]$. Additionally, out of every part P_i of \mathcal{P} , any non-trivial section must contain precisely one of two halves which are completely determined by the underlying structure \mathcal{U} , thus also showing the second statement of the lemma. \square

We are now ready to show how Lemmas 3.3.6 and 3.3.7 can be used to find a section in $\tilde{\text{CPT}}+\text{C}$ if there is any.

Proposition 3.3.8. *Let τ be a vocabulary of arity at most 2, let \mathcal{U} be a τ -structure and let $\mathcal{P} = (P_1, \dots, P_{\ell})$ be a stable coloring of \mathcal{U} . It is $\tilde{\text{CPT}}+\text{C}$ -definable whether \mathcal{U} has a non-trivial section and there is a $\tilde{\text{CPT}}+\text{C}$ -term defining such a section if any exists.*

Proof. Let R^1, \dots, R^k be the binary relations in τ . For each R^i , let \mathcal{U}^{R^i} denote the directed graph with edge relation R^i and let $\mathcal{U}_{\mathcal{P}}^{R^i}$ be the corresponding switching equivalent graph. We write

3 Capturing Polynomial Time on Fragments of Logarithmic Size

$n_{xy}^0(R^i)$ for the number of R^i -edges going out of each vertex in P_x to P_y , $n_{xy}^1(R^i)$ for the number of R^i -edges coming in to each vertex in P_y from P_x , and we write $\tilde{n}_{xy}^0(R^i)$ and $\tilde{n}_{xy}^1(R^i)$ for the respective quantities in $\mathcal{U}_{\mathcal{P}}^{R^i}$.

As above, let $G_{\mathcal{P}}$ be the undirected graph on U which has an edge between $u, v \in U$ whenever there is $i \in [k]$ such that $uv \in R^i$ or $vu \in R^i$. If $G_{\mathcal{P}}$ is not connected, then its connected components $\{H_1, \dots, H_m\}$ form a section of \mathcal{P} in \mathcal{U}^6 by Proposition 3.3.4 and since every H_i is a union of connected components of each $G_{\mathcal{P}}^{R^j}$. The connected components can easily be found in $\tilde{\text{CPT}}+\text{C}$.

If $G_{\mathcal{P}}$ is connected, then we check if there are any $x, y \in [\ell]$ and $i \in [k]$ so that $n_{xy}^0(R^i) = \frac{1}{2}|P_y|$. If not, then there is no non-trivial section of \mathcal{P} in \mathcal{U} . To see this, consider any $H \subset U$. As $G_{\mathcal{P}}$ is connected, there is a relation R^i so that H is not the union of connected components of $G_{\mathcal{P}}^{R^i}$. By Lemma 3.3.6 H is not a section in \mathcal{U}^{R^i} and therefore it is not a section in \mathcal{U} .

So suppose that we find $n_{xy}^0(R^i) = \frac{1}{2}|P_y|$ and let i, x, y be lexicographically minimal with this property. We now search for a section H in \mathcal{U} as indicated in the proof of Lemma 3.3.7. To start, the set

$$\mathcal{H}_I := \{ \{v \in P_y \mid uv \in R^i\} \mid u \in P_x \}$$

is clearly $\tilde{\text{CPT}}+\text{C}$ -definable. \mathcal{H}_I must contain precisely two non-equal sets which partition P_y , otherwise there is no section in \mathcal{U} . Set $I = \{y\}$.

At all points in time, \mathcal{H}_I must satisfy the property that it partitions $\bigcup_{x \in I} P_x$ and that for all $x, y \in I$, each $H \in \mathcal{H}_I$, and every $j \in [k]$, $P_x \cap H$ must be either totally connected or disconnected to $P_y \setminus H$ with respect to R_j and the same must be true for $P_x \setminus H$ and $P_y \cap H$. If this is true, then we call \mathcal{H}_I a *pre-section*. If \mathcal{H}_I is not a pre-section, then it cannot be extended to a section. We maintain the property that if \mathcal{H}_I cannot be extended to a section then there is no section in \mathcal{U} at all. This is the case with $I = \{y\}$ at this point by Lemma 3.3.7.

We now do the following until $I = [\ell]$. If there are $x \in I, y \notin I$ and $j \in [k]$ so that $0 < \tilde{n}_{xy}^0(R^j) < \frac{1}{2}|P_y|$ (or $0 < \tilde{n}_{yx}^1(R^j) < \frac{1}{2}|P_y|$) then let j, x, y be lexicographically minimal with this property. Add to each set $H \in \mathcal{H}_I$ all the elements in $v \in P_y$ so that there is an R^j -edge from $P_x \cap H$ to v (respectively from v to $P_x \cap H$) in order to obtain $\mathcal{H}_{I \cup \{y\}}$. This extension is forced since by Lemma 3.3.5 any possible extension of \mathcal{H}_I to a section in \mathcal{U} must contain the sets in $\mathcal{H}_{I \cup \{y\}}$. We add y to I .

If $I \neq [\ell]$ and there are no $x \in I, y \notin I$ and $j \in [k]$ so that $0 < \tilde{n}_{xy}^0(R^j) < \frac{1}{2}|P_y|$ or $0 < \tilde{n}_{yx}^1(R^j) < \frac{1}{2}|P_y|$, then there must be $x \in I, y \notin I$ and $j \in [k]$ so that $\tilde{n}_{xy}^0(R^j) = \frac{1}{2}|P_y|$ or $\tilde{n}_{yx}^1(R^j) = \frac{1}{2}|P_y|$ since $G_{\mathcal{P}}$ is connected. Let j, x, y be lexicographically minimal with this property. Add to each set $H \in \mathcal{H}_I$ all the elements in $v \in P_y$ so that there is an R^j -edge from $P_x \cap H$ to v (respectively from v to $P_x \cap H$) in order to obtain $\mathcal{H}_{I \cup \{y\}}$.

We claim that if \mathcal{H}_I is extendible to a section, then our choice of $\mathcal{H}_{I \cup \{y\}}$ is a pre-section which is also extendible to a section. To see this, suppose that F is a non-trivial section in \mathcal{U} extending $H \in \mathcal{H}_I$. As $n_{xy}^0(R^j) = \frac{1}{2}|P_y|$ or $n_{yx}^1(R^j) = \frac{1}{2}|P_y|$, we must have $F \cap P_y = H \cap P_y$ for one of the two sets $H \in \mathcal{H}_{I \cup \{y\}}$. Set $F' = H \cup \bigcup_{i \notin I \cup \{y\}} (F \cap P_i)$. Consider any $s \in I$ and $t \notin I$. As \mathcal{H}_I is extendible to a section and $H \setminus P_y \in \mathcal{H}_I$, H contains one of two fixed halves of P_s by Lemma 3.3.7. Similarly, F contains one of the two matching halves of P_t . Given that F is a section and $\tilde{n}_{st}^0(R^i), \tilde{n}_{ts}^1(R^i) \in \{0, \frac{1}{2}|P_t|\}$ for all $i \in [k]$, all edges between P_s and P_t respect these halves. Thus, F' is a section extending H , and therefore $\mathcal{H}_{I \cup \{y\}}$ is an extendible pre-section. Finish the step by adding y to I .

It is readily verified that all the necessary properties and case distinctions are $\tilde{\text{CPT}}+\text{C}$ -definable. Since $G_{\mathcal{P}}$ is connected, the inductive process terminates with a section $\mathcal{H}_{[\ell]}$ unless one of the intermediate steps is found not be a pre-section, in which case we may conclude that there is no section of \mathcal{P} in \mathcal{U} . \square

⁶I.e. each $H_i, i \in [m]$, is a section of \mathcal{P} in \mathcal{U} ; see the discussion after Definition 3.3.1.

3.4 The singly exponential canonization scheme

Let U be a unary relation symbol and let τ be vocabulary of arity at most 2 with $U \in \tau$. Let a τ -structure \mathcal{A} be given with its universe of size n . We are only interested in the values of relations $R \in \tau$ on elements from U^A , and we write \mathcal{U} for the substructure of \mathcal{A} induced on U^A . By a sensible abuse of notation, we denote the universe of \mathcal{U} by U . Throughout this chapter, we assume that $m := |U| = O(\log n)$. The goal is to canonize \mathcal{U} in $\tilde{\text{CPT}}+\text{C}$ (which works over \mathcal{A} as its input).

As in the last section, all definitions and results carry over to edge-colored directed graphs by considering $(R)_{R \in \tau}$ to be the ordered list of edge color classes.

The canonization of \mathcal{U} is constructed in two steps. Firstly, we define a rooted tree \mathcal{T} whose vertices are stable colorings of subsets of U and whose edges stand for applications of sections, individualization, and color refinement. In Corneil and Goldberg's exposition, this tree is implicit in a recursive implementation of their algorithm. Here, we store \mathcal{T} as an actual object that $\tilde{\text{CPT}}+\text{C}$ can work with, but we keep the name and call \mathcal{T} the recursion tree. The leaves of \mathcal{T} will be totally ordered partitions of subsets of U . In a second step, we label the tree vertices with canonical copies of the induced subgraphs that they correspond to, and we show how canonical forms labeling the children of a vertex induce the canonical label of the vertex itself. The label of the root then gives us the canonical copy of \mathcal{U} .

We define \mathcal{T} inductively, using ordered partitions of subsets of U as its vertices. At termination of the inductive definition, all partitions in \mathcal{T} will be stable. Also, the edge relation will then have the property that $(\mathcal{P}, \mathcal{Q}) \in \mathcal{T}$ implies that \mathcal{Q} is either a refinement of \mathcal{P} (in case of individualization) or a refinement of a restriction of \mathcal{P} (in case of sectioning). Figure 3.4 illustrates the definition of the recursion tree for a directed graph G .

Let \mathcal{P}_0 be the partition of U corresponding to the initial coloring of the 2-dimensional Weisfeiler-Lehman method (see Section 3.2.3). At the beginning, \mathcal{T} is initialized with \mathcal{P}_0 as its root. Then the following is repeated until \mathcal{T} does not grow any more.

For each leaf $\mathcal{P} = (A_1, \dots, A_\ell)$ of \mathcal{T} do:

1. If \mathcal{P} is not a stable partition, do color refinement.
2. If \mathcal{P} is a stable partition, use Proposition 3.3.8 to check if there is a section. If so, Proposition 3.3.8 produces a section (H_1, \dots, H_k) . Make $H_i \cap \mathcal{P}$ a child of \mathcal{P} in \mathcal{T} for each $i \in [k]$.
3. If \mathcal{P} is stable, does not admit a section and is not total, let $i \in [\ell]$ be the smallest index such that A_i is of smallest cardinality among all nonsingleton sets in \mathcal{P} . For each $x \in A_i$, make $(A_1, \dots, \{x\}, A_i \setminus \{x\}, \dots, A_\ell)$ a child of \mathcal{P} in \mathcal{T} .

From now on, \mathcal{T} shall only refer to the final result of this inductive definition. One property of \mathcal{T} is immediately obvious.

Observation 3.4.1. *Every leaf of \mathcal{T} is a total ordered partition of a subset of U .* □

Before we show how \mathcal{T} can be used to obtain a canonical form of \mathcal{U} , we first argue that this tree can technically be defined in $\tilde{\text{CPT}}+\text{C}$ – under the provision that its size remains polynomially bounded.

Proposition 3.4.2. *If $c \in \mathbb{N}$ is a fixed constant and $|\mathcal{T}| \leq O(n^c)$ over a structure \mathcal{A} , then \mathcal{T} is $\tilde{\text{CPT}}+\text{C}$ -definable.*

3 Capturing Polynomial Time on Fragments of Logarithmic Size

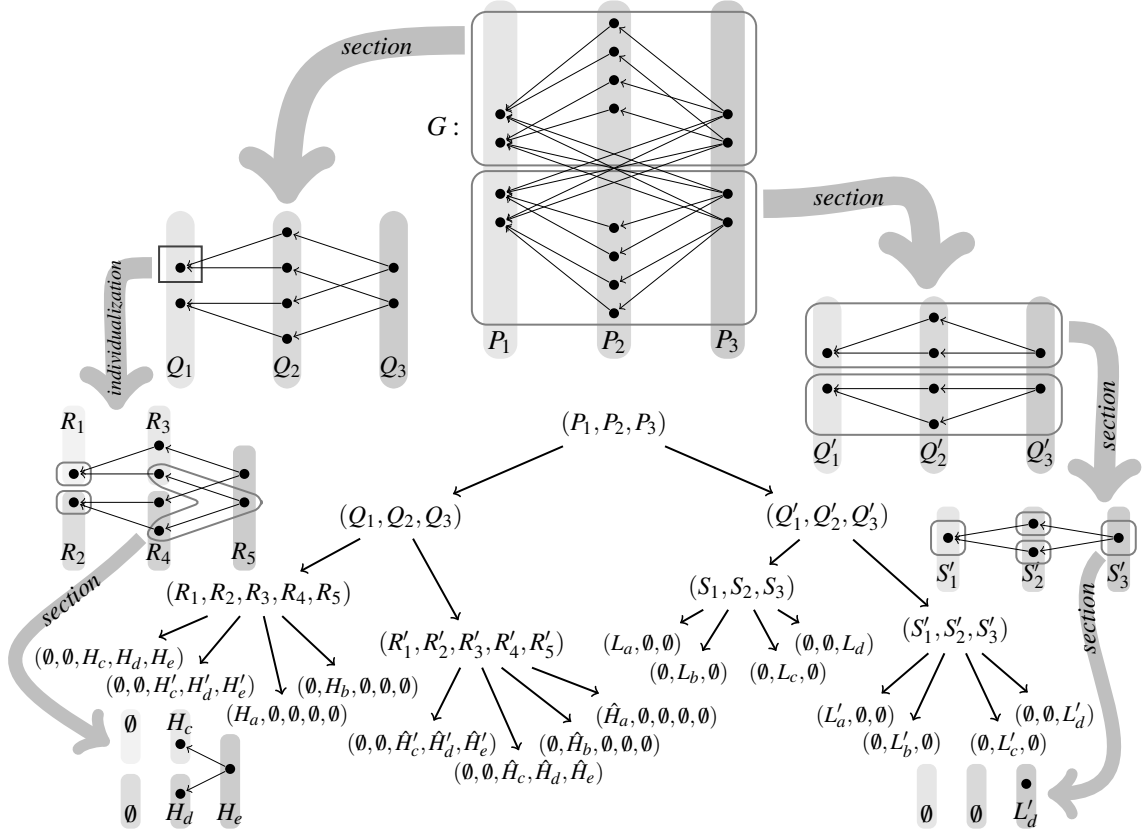


Figure 3.4: A directed graph G and its recursion tree \mathcal{T} , with an illustration of \mathcal{T} 's outer branches. All displayed colorings are stable. Sections are indicated by gray boxes.

Proof. Color refinement of non-stable partitions is definable in $\tilde{\text{CPT}}+\text{C}$ by Theorem 3.2.9. Finding a graph's connected components is in FP and thus in $\tilde{\text{CPT}}+\text{C}$ by Theorem 3.2.4. The inductive definition of \mathcal{T} already suggests how to be cast in $\tilde{\text{CPT}}+\text{C}$. Steps 1 to 3 form the *program* Π (cf. Section 3.2.1) which is repeated until \mathcal{T} is fully defined. The program's control flow is readily seen to be definable in $\tilde{\text{CPT}}+\text{C}$.

The vertices of \mathcal{T} are ordered partitions of subsets of U , which are encoded as sets from $\mathcal{HF}(A)$ (see Section 3.2.1), each of which induces $O(n)$ active elements. As $|\mathcal{T}| \leq O(n^c)$, both the length of Π 's run and the total number of active elements are bounded by fixed polynomials. Therefore, Π is a $\tilde{\text{CPT}}+\text{C}$ program defining \mathcal{T} . \square

We will later show in Section 3.5 that the size of \mathcal{T} is bounded by $2^{O(m)}$, and since $m = |U| = O(\log n)$, this guarantees $|\mathcal{T}| = n^{O(1)}$ and the applicability of Proposition 3.4.2. Once \mathcal{T} has been defined in $\tilde{\text{CPT}}+\text{C}$, it is left to show how to employ \mathcal{T} in the construction of a canonical copy of \mathcal{U} on the number sort of $\tilde{\text{CPT}}+\text{C}$.

Theorem 3.4.3. *Suppose \mathcal{T} is $\tilde{\text{CPT}}+\text{C}$ -definable over \mathcal{U} . Then we can $\tilde{\text{CPT}}+\text{C}$ -define an ordered canonical copy of \mathcal{U} on the number sort of $\tilde{\text{CPT}}+\text{C}$.*

Proof. Let R^1, \dots, R^k be the binary relations in τ . We construct a labeling of \mathcal{T} 's vertices from the leaves inwards so that the root's label contains the desired canonical form. Let $\mathcal{P} = (P_1, \dots, P_\ell)$ be a vertex of \mathcal{T} and let $\mathcal{U}[\cup \mathcal{P}]$ be the corresponding induced substructure. We write $V_{\mathcal{P}} := [|\cup \mathcal{P}|]$. The *label* of \mathcal{P} consists of

- an ordered partition $\mathcal{P}' = (P'_1, \dots, P'_\ell)$ of $V_{\mathcal{P}}$ so that $|P'_i| = |P_i|$ for all $i \in [\ell]$ and

3.4 The singly exponential canonization scheme

- an ordered τ -structure $C_{\mathcal{P}} = (V_{\mathcal{P}}, \varepsilon_1, \dots, \varepsilon_k)$ for which there is an isomorphism $\varphi : \mathcal{U}[\cup \mathcal{P}] \rightarrow C_{\mathcal{P}}$ which maps vertices from P_i to P'_i for all $i \in [\ell]$.

Note that the isomorphism φ is not part of the label; we only used it to illustrate that $C_{\mathcal{P}}$ must be *color isomorphic* to $\mathcal{U}[\cup \mathcal{P}]$ where we identify colors P_i and P'_i . Keeping track of the vertex colors inside the canonical forms is necessary for defining the edge relation when we piece back together the parts of a section.

Since \mathcal{T} 's leaves are total partitions of subsets of U , it is easy to define \mathcal{P}' and $C_{\mathcal{P}}$ for them directly. Next, consider a node \mathcal{P} of \mathcal{T} and suppose all its children $\mathcal{Q}_1, \dots, \mathcal{Q}_q$ are already labeled. There are two cases: Either the children of \mathcal{P} stem from an individualization operation, or they stem from splitting \mathcal{P} along a section. We can tell the two cases apart by checking if $|\cup \mathcal{P}| = |\cup \mathcal{Q}_i|$ for any of the \mathcal{Q}_i .

In the case of individualization, we let $C_{\mathcal{P}}$ be the lexicographic leader among the labels $C_{\mathcal{Q}_1}, \dots, C_{\mathcal{Q}_q}$. Since the $C_{\mathcal{Q}_i}$, $i \in [q]$, are totally ordered, their lexicographic leader is $\tilde{\text{CPT}}+\text{C}$ -definable by Lemma 2.4.5 and Theorem 3.2.4. For any $x \in V_{\mathcal{P}} = [|\cup \mathcal{P}|]$, let $c_{C_{\mathcal{Q}_i}}(x)$ be the color class of x in one of the $C_{\mathcal{Q}_i}$, let $c_{\mathcal{Q}_i}(x)$ be the corresponding color class in \mathcal{Q}_i , and let $c_{\mathcal{P}}(x)$ be the color in \mathcal{P} of any vertex $y \in c_{\mathcal{Q}_i}(x)$. Since all the \mathcal{Q}_i are refinements of \mathcal{P} , $c_{\mathcal{P}}(x)$ is uniquely defined and we obtain \mathcal{P}' from this correspondence.

In the case of sectioning, we need to paste together the structures labeling $\mathcal{Q}_1, \dots, \mathcal{Q}_q$. Let $C' = (V_{\mathcal{P}}, \varepsilon'_1, \dots, \varepsilon'_k)$ be the lexicographic disjoint union of the *colored structures* $C_{\mathcal{Q}_1}, \dots, C_{\mathcal{Q}_q}$ which is $\tilde{\text{CPT}}+\text{C}$ -definable by Lemma 2.4.6 and Theorem 3.2.4. This means that the vertex colors \mathcal{Q}'_i of each $C_{\mathcal{Q}_i}$ are taken into account so that $C_{\mathcal{Q}_i} = C_{\mathcal{Q}_j}$ if and only if the two structures are order isomorphic with respect to their edge relations and $\mathcal{Q}'_i = \mathcal{Q}'_j$.

Next, we define the colors \mathcal{P}' on the vertices $V_{\mathcal{P}}$. For $x \in V_{\mathcal{P}}$ let $C_{\mathcal{Q}_{i_1}}, \dots, C_{\mathcal{Q}_{i_s}}$ be the order and color isomorphic structures which are mapped to an interval $I \subseteq V_{\mathcal{P}}$ containing x when forming the lexicographic disjoint union of all the $C_{\mathcal{Q}_i}$. As above, let $c_{C_{\mathcal{Q}_{i_j}}}(x)$ be the color of x in any of the $C_{\mathcal{Q}_{i_j}}$, let $c_{\mathcal{Q}_{i_j}}(x)$ be the corresponding color class in \mathcal{Q}_{i_j} , and let $c_{\mathcal{P}}(x)$ be the color in \mathcal{P} of any vertex $y \in c_{\mathcal{Q}_{i_j}}(x)$. Each \mathcal{Q}_{i_j} is a refinement of $\mathcal{P} \cap \cup \mathcal{Q}_{i_j}$, and since all the $C_{\mathcal{Q}_{i_j}}$ are color isomorphic the same is true of all the $\mathcal{U}[\cup \mathcal{Q}_{i_j}]$ with respect to the colorings $\mathcal{P} \cap \cup \mathcal{Q}_{i_j}$. Therefore, $c_{\mathcal{P}}(x)$ is again uniquely defined and we can use it to define \mathcal{P}' on $V_{\mathcal{P}}$. Note that at this point, we really need to use the fact that we kept track of empty sets in the *partition* $\mathcal{P} \cap \cup \mathcal{Q}_{i_j}$, otherwise we could not guarantee that $c_{\mathcal{P}}(x)$ is uniquely defined.

Finally, the edge relation between vertices from different $C_{\mathcal{Q}_i}$ is determined from the coloring we just defined. For this, we need to once again refine the ordering of the $C_{\mathcal{Q}_i}$ since, even when $C_{\mathcal{Q}_i} = C_{\mathcal{Q}_j}$, the graphs $\mathcal{U}[\cup \mathcal{Q}_i]$ and $\mathcal{U}[\cup \mathcal{Q}_j]$ might be connected differently inside $\mathcal{U}[\cup \mathcal{P}]$. So when $C_{\mathcal{Q}_{i_1}} = \dots = C_{\mathcal{Q}_{i_s}}$ are a maximal set of order and color isomorphic canons, we can compare them on whether $P_x \cap \cup \mathcal{Q}_{i_j}$ and $P_y \setminus \cup \mathcal{Q}_{i_j}$ are fully connected or totally disconnected with respect to each R^j . Since the cells P_x and the relations R^j are linearly ordered, this induces a strict weak ordering on $C_{\mathcal{Q}_{i_1}}, \dots, C_{\mathcal{Q}_{i_s}}$.

So let $x, y \in V_{\mathcal{P}}$ be in the image of different structures $C_{\mathcal{Q}_j}$ and $C_{\mathcal{Q}_{j'}}$, $j \neq j'$, when forming the lexicographic disjoint union of all the $C_{\mathcal{Q}_i}$ and after refining the lexicographic order as in the preceding paragraph. Then x, y shall be connected by an ε_i -edge if the sets $c_{\mathcal{P}}(x) \cap \cup \mathcal{Q}_j$ and $c_{\mathcal{P}}(y) \cap \cup \mathcal{Q}_{j'}$ are R^i -connected as sets in $\mathcal{U}[\cup \mathcal{P}]$. Since $(\cup \mathcal{Q}_1, \dots, \cup \mathcal{Q}_k)$ is a section of \mathcal{P} in $\mathcal{U}[\cup \mathcal{P}]$, $c_{\mathcal{P}}(x) \cap \cup \mathcal{Q}_j$ and $c_{\mathcal{P}}(y) \cap \cup \mathcal{Q}_{j'}$ are either fully R^i -connected or totally R^i -disconnected. By the refined ordering from the preceding paragraph, each choice for $C_{\mathcal{Q}_j}$ and $C_{\mathcal{Q}_{j'}}$ above leads to the same result of x, y being connected or not. So we let $C_{\mathcal{P}} = (V_{\mathcal{P}}, \varepsilon_1, \dots, \varepsilon_k)$, where each ε_i consists of the edges in ε'_i together with the edges we just described. We conclude that $C_{\mathcal{P}}$ is (color) isomorphic to $\mathcal{U}[\cup \mathcal{P}]$.

This finishes the description of the labeling process of \mathcal{T} , which terminates with a canonical

3 Capturing Polynomial Time on Fragments of Logarithmic Size

copy $C_{\mathcal{P}_0}$ of (U, R^1, \dots, R^k) at the root \mathcal{P}_0 . Any unary relation $X \in \tau$ can be defined on the ordered domain using the correspondence between \mathcal{P}'_0 and \mathcal{P}_0 and the fact that \mathcal{P} is a refinement of the coloring induced by the unary relations in τ .

It should be clear from the description of each labeling step that it can be defined in a $\tilde{\text{CPT}}+\text{C}$ program. The run of this program then repeats the labeling step until all vertices of \mathcal{T} are labeled and the canonical form has been found. The bounds on the number of active elements and length of the run both depend polynomially on the size of \mathcal{T} , which is polynomial by our assumption that \mathcal{T} is $\tilde{\text{CPT}}+\text{C}$ -definable. \square

3.5 Bounding the size of the recursion tree

The analysis in this section relies largely on [CG84]. The key to obtaining a size bound of $2^{O(m)}$ on \mathcal{T} is the following lemma, which also justifies the use of sections.

Lemma 3.5.1. *Let \mathcal{P} be a stable partition of \mathcal{U} which does not have a section, let A be a non-singleton cell of \mathcal{P} and let $x \in A$. If $\mathcal{P}^*(x)$ denotes the stable coloring after individualizing x , then there are cells Y of $\mathcal{P}^*(x)$ and X of \mathcal{P} such that $Y \subset X$ and $1 < |Y| \leq \frac{1}{2}|X|$.*

Proof. Let X_1, \dots, X_a be the non-singleton cells of \mathcal{P} and let Y_1, \dots, Y_b be the non-singleton cells of $\mathcal{P}^*(x)$. Each Y_j is the subset of some X_i and we may assume $a \geq b$, since otherwise some X_i contains at least two Y_j s, making the lemma immediately true. So, after renumbering we have $Y_i \subset X_i$ for $i \in [b]$. We define

$$Z_i := \begin{cases} X_i \setminus Y_i & i \in [b] \\ X_i & i \in \{b+1, \dots, a\} \end{cases}$$

Since $\bigcup_{i \in [a]} Z_i$ must not be a section of \mathcal{P} , there are a binary relation $R \in \tau$ and $(u, v) \in R$ so that w.l.o.g. $u \in Z_i$ and $v \in Y_j$ for some $i \in [a]$, $j \in [b]$, and there is an element $u' \in Z_i$ so that $(u', v) \notin R$. The case where the same conditions hold with $(v, u) \in R$, $(v, u') \notin R$ works symmetrically.

Since both u and u' constitute singleton cells in $\mathcal{P}^*(x)$, u is fully connected to Y_j . To see this notice that otherwise being connected to u would split up the color class Y_j . Similarly, u' is totally disconnected to Y_j . If N and N' denote the sets of outgoing neighbors of u and u' in X_j , respectively, then we have that

$$|Y_j| \leq |N \cap X_j| = |N' \cap X_j| \leq |Z_j| = |X_j \setminus Y_j|,$$

where equality in the middle is warranted by \mathcal{P} being a stable coloring. Therefore, $|Y_j| \leq \frac{1}{2}|X_j|$, completing the proof. \square

The worst-case bound on the size of \mathcal{T} is achieved if the $\tilde{\text{CPT}}+\text{C}$ -algorithm never encounters a section, so we assume that all of \mathcal{T} 's branches stem from individualization. In order to bound the size of \mathcal{T} , repeat the following operation until no more changes can occur: choose an arbitrary node of \mathcal{T} and replace all of its branches by one of its biggest branches.⁷ Suppose \mathcal{T} is of this form now, and write \mathcal{P}_i for the unique partition associated with level i (where 0 is the level of the root and \tilde{h} is the height of \mathcal{T}). Then for each level i , all nodes on this level have the same outdegree d_i , and we have $|\mathcal{T}| = 1 + d_1 + d_1 \cdot d_2 + \dots + d_1 \cdot \dots \cdot d_{\tilde{h}} \leq 2d_1 \cdot \dots \cdot d_{\tilde{h}}$, where the inequality is justified by the fact that each $d_i \geq 2$ as only vertices in non-singleton partition cells become individualized. To get a bound on this product we define $\theta_q := \left| \left\{ i \in [\tilde{h}] \mid d_i > \frac{m}{q} \right\} \right|$, where $m = |\mathcal{U}|$ as before. We aim to show the following:

⁷The order in which we choose to replace branches of \mathcal{T} may lead to different outcomes, however, any result of this process is fine for us.

Proposition 3.5.2. $\theta_q < q$ for all q .

Proof. Fix q and let the partitions among $\mathcal{P}_0, \dots, \mathcal{P}_h$ with associated $d_i > \frac{m}{q}$ be denoted by $\mathcal{Q}_1, \dots, \mathcal{Q}_{\theta_q}$. In each \mathcal{Q}_i , all nonsingleton cells have size larger than $\frac{m}{q}$ since only vertices from the smallest cell are individualized.

Note that for all $i < j$, \mathcal{P}_j is a refinement of \mathcal{P}_i . If A is a cell of some partition \mathcal{P}_i , then we denote by $s(A)$ the number of partitions \mathcal{Q}_j in which A occurs nontrivially split in the sense of Lemma 3.5.1, i.e.,

$$s(A) := \left| \left\{ j \in [\theta_q] \mid \exists B \in \mathcal{Q}_j \text{ s.t. } B \subset A \text{ and } 1 < |B| \leq \frac{1}{2}|A| \right\} \right|.$$

Assume that \mathcal{Q}_{θ_q} is the partition among the \mathcal{Q}_i that occurs farthest away from the root in \mathcal{T} . Let $\mathcal{P}_0, \dots, \mathcal{P}_r$ be the complete list of partitions occurring in \mathcal{T} starting from the root down to $\mathcal{Q}_{\theta_q} = \mathcal{P}_r$.

Claim 3.5.3. For every $i \in [0, r]$ and any nonsingleton cell $A \in \mathcal{P}_i$ we have $|A| > \frac{m}{q}(s(A) + 1)$.

Proof of Claim 3.5.3. We show this by induction on i , starting at $i = r$. For any cell $A \in \mathcal{P}_r = \mathcal{Q}_{\theta_q}$, this is true since $s(A) = 0$ and $|A| > \frac{m}{q}$, as noted above.

Now let $A \in \mathcal{P}_i$ and suppose that in \mathcal{P}_{i+1} , cell A occurs split into nonsingleton cells B_1, \dots, B_k with $|B_j| > \frac{m}{q}(s(B_j) + 1)$. If $k \geq 2$, then it is clear that $|A| \geq \sum_{j \in [k]} |B_j| > \frac{m}{q} \sum_{j \in [k]} (s(B_j) + 1) \geq \frac{m}{q}(s(A) + 1)$ since $s(A) \leq 1 + \sum_{j \in [k]} s(B_j)$. For $k = 1$ there are two cases. Either, $|B_1| > \frac{1}{2}|A|$, in which case $s(B_1) = s(A)$ and the claim for A follows. Or, $|B_1| \leq \frac{1}{2}|A|$, whence $|A| > 2 \frac{m}{q}(s(B_1) + 1) \geq \frac{m}{q}(s(A) + 1)$ since $s(A) \leq s(B_1) + 1$. *Claim 3.5.3* \square

If now P_1, \dots, P_ℓ are the non-singleton cells of the root \mathcal{P}_0 , then by Claim 3.5.3

$$m \geq \sum_{j \in [\ell]} |P_j| > \frac{m}{q} \sum_{j \in [\ell]} (s(P_j) + 1).$$

Invoking Lemma 3.5.1, we know that on each level, there occurs at least one nontrivial split, so in particular this happens when going to $\mathcal{Q}_i \neq \mathcal{P}_0$ from the level above for each $i \in [\theta_q]$. Thus, $\sum_{j \in [\ell]} (s(P_j) + 1) \geq \theta_q$. We conclude that $m > \frac{m}{q} \cdot \theta_q$ and therefore $\theta_q < q$, as required. \square

Proposition 3.5.2 now gives us a bound on the product of degrees $d_1 \cdots d_h$ in \mathcal{T} : there is at most one d_i with $d_i > \frac{m}{2}$, at most two d_i with $d_i > \frac{m}{3}$, and so on. As there are at most m individualizations of vertices, we obtain

$$d_1 \cdots d_h \leq m \cdot \frac{m}{2} \cdot \frac{m}{3} \cdots \frac{m}{m} = \frac{m^m}{m!} < e^m = 2^{O(m)}$$

This proves that $|\mathcal{T}| \leq 2^{O(m)}$ and completes the proofs of Theorems 3.1 and 3.2.

3.6 Discussion

In this chapter we showed how to find canonical forms of arbitrary directed graphs with edge and vertex colors in singly exponential time and space. The canonization procedure avoids making arbitrary choices altogether, which meant that we could implement it in the logic Choiceless Polynomial Time with counting ($\tilde{\text{CPT}}+\text{C}$) when restricting our attention to logarithmic-sized fragments of such graphs. In particular, this means for vocabularies τ of arity at most 2 that on classes of τ -structures where a unary relation U defines a logarithmic-sized subset of the structure, $\tilde{\text{CPT}}+\text{C}$ captures PTIME on the restrictions of these structures to U . Notice that when $|U| \leq c \log n$,

3 Capturing Polynomial Time on Fragments of Logarithmic Size

where n is the size of the structure, then PTIME can in effect express all singly exponential time (ETIME) properties of U , and by the capturing result, so can $\tilde{\text{CPT}}+\text{C}$.

Of course, the most important open question at the end of this chapter is whether this result for structures of arity at most 2 can be extended to general structures.

Open Problem. *Is there a canonical form for general structures which can be computed in a choiceless manner in singly exponential time? Consequently, does $\tilde{\text{CPT}}+\text{C}$ capture PTIME on logarithmic-sized fragments of general structures?*

We note that so far, no singly exponential time canonization algorithm for general structures is known at all, irrespective of whether it may use an auxiliary order or not (cf. Section 3.1). It is an open question whether the singly exponential time isomorphism algorithms of Luks [Luk99] or the faster algorithms of Babai and Codenotti [BC08] can be turned into canonization algorithms. In particular, it is open whether any of the group theoretic methods which they use can be cast in Choiceless Polynomial Time. We will highlight the obstacles for this in Section 5.7 when we discuss the prospects of implementing group theoretic methods in rank logics.

In this situation, it would be interesting to know whether the combinatorial canonization algorithm presented here can be generalized to a canonization algorithm for more general structures. Instead of 2-dimensional Weisfeiler-Lehman color refinement, a higher-dimensional version as presented in [CFI92] could be used to deal with relations of higher arity. The decisive property that we would need to aim for is a more general kind of section which can be found easily and which guarantees that the individualization of vertices splits the color classes in a non-trivial way when there is no section of a stable coloring. However, this might be hard to find: in Section 5.6, we present structures with a relation symbol R of arbitrary arity k in which R encodes the property of k numbers summing to 0 – but without a label or an ordering of these numbers. More precisely, for some $n \in \mathbb{N}$ let R be defined on $[n]^k$ so that $\vec{a} \in R$ if $\sum_{i \in [k]} a_i = 0 \pmod{n}$. This structure has many degrees of freedom: for any $i \neq j \in [k]$, $\vec{a} \mapsto (a_1, \dots, a_i - 1, \dots, a_j + 1, \dots, a_k) \pmod{n}$ defines an automorphism. Additionally, the map $\vec{a} \mapsto -\vec{a} \pmod{n}$ defines an automorphism. As the coloring determined by Weisfeiler-Lehman cannot cut through orbits of the structure's automorphism group, color refinement does not come up with a meaningful refinement. As the tuples in R are rather sparse, it is hard to imagine what a section for this case should look like. And individualizing any single element does not imply any further refinement of the color classes as this does not fix enough automorphisms. In any case, a generalization of this chapter's approach to general structures would have to deal with these kinds of structures.

4 Capturing Results on Classes of Graphs

In the absence of a viable candidate logic for capturing PTIME on all structures, this chapter presents properties of graphs that may prevent or enable the precise logical characterization of complexity classes. On the negative side, we may ask how much structural richness is enough for a complexity class to elude being captured on a graph class by a certain logic. We report on a variety of graph classes on which fixed-point logic with counting FP+C does not capture PTIME. On the positive side, it is worthwhile to exhibit restrictions to structural richness that allow our logics to get a full grip and work out capturing results on these restricted graph classes. We show here that the class of interval graphs enjoys properties that enable the capturing of all PTIME properties on it. In both cases, we gain insight into the interplay between the expressiveness of logical sentences and the structures they are evaluated in.

The approach of capturing PTIME on restricted graph classes has been very fruitful in the realm of graph classes defined by lists of forbidden minors. A graph H is a minor of G if it can be obtained from G by a sequence of edge deletions, vertex deletions, and edge contractions, which corresponds to identifying the ends of an edge by an equivalence relation and then modding out by this relation (cf. Section 2.1). Most of these results show that PTIME is captured by FP+C when restricting ourselves to one such class, such as planar graphs [Gro98b], graphs of bounded tree-width [GM99], or K_5 -free graphs [Gro08]. In fact, Grohe has recently shown that FP+C captures PTIME on any graph class which is defined by a list of forbidden minors [Gro10b].

Given such deep results for classes of minor-free graphs, it is natural to ask if similar results can be obtained for graph classes which are defined by a (finite or infinite) list of forbidden induced subgraphs. Much less is known here. For starters, it is shown in [Gro10a] that FP+C does not capture PTIME on the class of chordal graphs. Hence, a general capturing result analogous to Grohe's is not possible for FP+C on induced subgraph-free graph classes. We replicate this proof in Section 4.2 and also show that FP+C does not capture PTIME on graphs whose complements are comparability graphs of partial orders. This is particularly interesting in light of this section's main result that PTIME is captured by FP+C on interval graphs, which have chordal and co-comparability graphs as natural superclasses. Thus, these two classes are shown to be a ceiling on the structural richness of graph classes on which capturing PTIME requires less effort than for general graphs. Along the way, we prove such non-capturing results for a variety of other graph classes as well.

Theorem 4.1. *FP+C does not capture PTIME on the classes of bipartite graphs, chordal graphs, split graphs, comparability, or co-comparability graphs.*

This result may not be surprising if we take into account that the concept of graph classes with forbidden induced subgraphs significantly extends the concept of graph classes with forbidden minors. Since an induced subgraph of a graph G is also a minor of G , any graph class defined by a list of forbidden minors may also be defined by a (albeit longer) list of forbidden induced subgraphs. Methods for capturing PTIME on all induced subgraph-free graph classes would therefore have to apply simultaneously to the classes that Grohe has considered, and to a large variety of further classes such as intersection graphs of high-dimensional boxes, line graphs, and perfect graphs, including bipartite, chordal, and comparability graphs. Theorem 4.1 says that the blow-up in structural variety when going from minor-free to induced subgraph-free graphs

4 Capturing Results on Classes of Graphs

overburdens FP+C's expressive power and makes it fall behind the expressiveness of PTIME. In fact, we go further and show in Section 4.2 that the graph classes mentioned in Theorem 4.1 are in some sense *complete* for the problem of capturing PTIME, showing that any *reasonable* logic captures PTIME on these restricted classes if and only if it captures PTIME on the class of all graphs.

This does not rule out the possibility, however, of finding induced subgraph-free graph classes which are not minor-free and on which FP+C does suffice to capture PTIME. Indeed, the main result in this chapter is a positive one affirming that FP+C captures PTIME on the class of interval graphs, the class of intersection graphs of intervals. Since this graph class is itself FP+C-definable (see Corollary 4.1.11), this means that a set \mathcal{K} of interval graphs is decidable in PTIME if and only if there is a sentence of FP+C defining \mathcal{K} .

Theorem 4.2. *FP+C captures PTIME on the class of interval graphs.*

The result is shown by describing an FP+C-definable canonization procedure for interval graphs, which for any interval graph constructs an isomorphic copy on an ordered domain. The capturing result then follows from the Immerman-Vardi Theorem 2.3.10. Both Theorems 4.1 and 4.2 have been published by the author of this thesis in [Lau10].

In order to make interval graphs logically tractable, we had to develop new methods that do not make use of an auxiliary order of the graph. The methods are conceptually rather simple and rely on a guess-and-verify sort of approach to find the vertices that have to come first in the graph's interval representation. As a by-product of these methods, we obtain two further results. The first one is an immediate consequence of our approach.

Theorem 4.3. *STC+C captures LOGSPACE on the class of proper interval graphs.*

Theorem 4.3 is proven in Section 4.4, where it is also shown that STC+C does not capture LOGSPACE on general interval graphs. This is really a shortcoming of the logic STC+C, though, since interval graphs are actually quite manageable with logarithmic workspace. This is the second, more intriguing result following from our methods.

Theorem 4.4. *Computing a canonical interval representation of an interval graph can be done in logspace. Deciding whether two interval graphs are isomorphic is LOGSPACE-complete.*

This improves the previously known AC^2 -upper bound for interval graph isomorphism [Kle96]. By also showing hardness for LOGSPACE of these problems under FO-reductions, we precisely pin down their complexity. We remark that our methods heavily rely on Reingold's Theorem 2.1.2 that undirected reachability is decidable in LOGSPACE. It is hard to fathom what an implementation without reachability queries as a subroutine might look like. The results in Theorem 4.4 have been published jointly by Johannes Köbler, Sebastian Kuhnert, Oleg Verbitsky, and the author of this thesis in [KKLV10a].

This chapter starts with introducing and discussing interval graphs in Section 4.1, where we also explain how the class of interval graphs can be defined in STC and thus recognized in LOGSPACE. Section 4.2 then shows the negative *non-capturing results* based on a notion of *non-capturing reductions* between graph classes. In Section 4.3, we turn to the proof of our principal Theorem 4.2 and show that FP+C captures PTIME on interval graphs. The methods developed there will then be applied to show that STC+C captures LOGSPACE on proper interval graphs in Section 4.4. The section also shows that STC+C is not expressive enough to capture LOGSPACE on all interval graphs. Finally, Section 4.5 shows the results about dealing with interval graphs in logarithmic space.

4.1 Interval graphs

4.1.1 Definition and basic properties

The properties of interval graphs mentioned here are based on [GH64] and [Möh84].

Definition 4.1.1 (Interval graph). Let \mathcal{I} be a finite collection of closed intervals $I_i = [a_i, b_i] \subset \mathbb{N}$. The graph $G_{\mathcal{I}} = (V, E)$ defined by \mathcal{I} has vertex set $V = \mathcal{I}$ and edge relation $I_i I_j \in E \Leftrightarrow I_i \cap I_j \neq \emptyset$. \mathcal{I} is called an *interval representation* of a graph G if $G \cong G_{\mathcal{I}}$. A graph G is an *interval graph* if there is a collection of closed intervals \mathcal{I} which is an interval representation of G .

We remark that usually, interval representations of interval graphs may contain general intervals over \mathbb{R} . It is easy to see, though, that any finite collection of intervals over \mathbb{R} can be transformed into intervals over \mathbb{N} without changing the intersection graph. Restricting ourselves to intervals over \mathbb{N} therefore gives rise to the same graph class. Also, there is nothing special about \mathbb{N} and we may have used any other countably infinite linearly ordered set in its place.

If $v \in V$, then I_v denotes the interval corresponding to vertex v in \mathcal{I} . An interval representation \mathcal{I} for an interval graph G is called *minimal* if the set $\bigcup \mathcal{I} \subset \mathbb{N}$ is of minimum size among all interval representations of G . Let us show that any interval representation \mathcal{I} can be converted into a minimal interval representation as follows. Choose $m \in \mathbb{N}$ such that all intervals of G are contained in $[m]$. For $k \in [m]$, let $M(k) = \{v \mid k \in I_v\}$. Clearly, each $M(k)$ forms a clique (if $M(k)$ is non-empty). Let $D \subseteq [m]$ be obtained by deleting all those points in $[m]$ for which $M(k)$ is not a *maximal* clique. We want to argue now that if we restrict \mathcal{I} to D , then we do not lose any information about the intersection of intervals and \mathcal{I} continues to be an interval representation for G . For this, we need the following fundamental lemma.

Lemma 4.1.2 (Helly property of interval graphs). *Let $G = (V, E)$ be an interval graph with interval representation \mathcal{I} and let $M \subseteq V$ be a clique of G . Then $\bigcap_{v \in M} I_v$ is a non-empty interval.*

Proof. Let \mathcal{E} be any set of (closed) intervals which intersect pairwise. Let $a = \max_{I \in \mathcal{E}} \min I$ be the *largest beginning point* of an interval in \mathcal{E} (in the order of the underpinning set). Similarly, let $b = \min_{I \in \mathcal{E}} \max I$ be the *least end point* of an interval in \mathcal{E} . If $b < a$, then there would be two intervals that do not intersect. Hence, it must hold that $a \leq b$ and it is immediately clear that $\bigcap \mathcal{E} = [a, b]$, so the intersection of all intervals in \mathcal{E} is a non-empty interval (possibly containing only one point). Now let M be a clique of G . As the intervals corresponding to M intersect pairwise, they also have an interval as their common intersection. \square

Lemma 4.1.2 shows that intervals have the *Helly property* (cf. [BLS99]). Generally, a class of objects has the Helly property if pairwise intersection implies a non-empty common intersection for each set of such objects.

Let us return to the set D we have constructed above. If u, v are neighboring vertices in G , then they are contained in some maximal clique $M_{u,v}$ of G . By Lemma 4.1.2 and the maximality of $M_{u,v}$, there is at least one point $k \in [m]$ so that $M(k) = M_{u,v}$. Thus, $k \in D$ and I_u and I_v intersect at k in D .

For any maximal clique N of G , let $k_N \in D$ be some point in D such that $M(k_N) = N$. Now let $D' = \{k_N \mid N \text{ is a maximal clique of } G\} \subseteq D$. Again, no information is lost when we restrict \mathcal{I} to D' . If we consider D' as an initial segment of \mathbb{N} , then \mathcal{I} restricted to D' is a minimal interval representation of G . To see this, notice that by Lemma 4.1.2, any maximal clique M of G corresponds to some interval $I_M := \bigcap_{v \in M} I_v$. If N, M are two distinct maximal cliques of G , then we must have that $I_N \cap I_M = \emptyset$, as otherwise these two cliques would form a bigger clique together, contradicting their maximality. Thus, the number of distinct maximal cliques is a lower bound on the number of points any interval representation of G must contain. Observe that \mathcal{I} over D' achieves this lower bound.

4 Capturing Results on Classes of Graphs

For our purposes here, let us sum up the main insight from the conversion procedure above.

Lemma 4.1.3. *Let G be an interval graph and \mathcal{I} be a minimal interval representation for G . For each $k \in \bigcup \mathcal{I}$, the set $M(k) = \{v \mid k \in I_v\}$ is a maximal clique of G . For any maximal clique M of G , $\bigcap_{v \in M} I_v = \{k\}$ for some $k \in \bigcup \mathcal{I}$. \square*

By this lemma, if \mathcal{I} is a minimal interval representation for G , then \mathcal{I} induces a strict linear order on the maximal cliques of G which has the property that each vertex is contained in *consecutive* maximal cliques. Conversely, suppose we can bring G 's maximal cliques into a linear order so that G 's vertices are contained in consecutive maximal cliques. Then we can define an interval representation for G by identifying the ordered set of maximal cliques with an initial segment of \mathbb{N} , and by assigning to each vertex v the interval of maximal cliques that v is contained in. This argument actually shows how to characterize interval graphs in a different way.

Theorem 4.1.4 ([GH64, Möh84]). *A graph G is an interval graph if and only if its maximal cliques can be brought into a linear order, so that each vertex of G is contained in consecutive maximal cliques. In other words, interval graphs are precisely those graphs which admit a path decomposition into their maximal cliques. \square*

We have seen that maximal cliques arguably play an important role for the structure of interval graphs. Indeed, maximal cliques will be central to the PTIME-capturing result on interval graphs in Section 4.3. For the sake of brevity, we will often just call them *max cliques* and denote the set of a graph's max cliques by \mathcal{M} . By the *span of a vertex v* , denoted $\text{span}(v)$, we mean the number of max cliques that v is contained in. The linear order induced on \mathcal{M} by a minimal interval representation \mathcal{I} will be denoted by $\prec_{\mathcal{I}}$. We call a max clique C a possible *end* of G if there is a minimal interval representation \mathcal{I} of G so that C is $\prec_{\mathcal{I}}$ -minimal.

Let us observe two properties of interval graphs here.

Lemma 4.1.5. *Every interval graph is chordal.*

Proof. This is seen directly from considering any interval representation. \square

Lemma 4.1.6. *Every interval graph is a co-comparability graph.*

Proof. For an interval graph G , we have to show that the edges in its complement G^c can be oriented transitively. Let \mathcal{I} be an interval representation of G . Whenever uv is an edge of G^c , then the intervals I_u and I_v do not intersect. We orient the edge uv from u to v if for all elements $x \in I_u, y \in I_v$ we have $x < y$. It is easy to see that this yields a transitive orientation of G^c 's edges. \square

In fact, a result by Gilmore and Hoffman shows that these two containments give rise to yet another characterization of interval graphs.¹

Theorem 4.1.7 (Gilmore and Hoffman [GH64]). *A graph is an interval graph if and only if it is chordal and its complement is a comparability graph. \blacksquare*

Gilmore and Hoffman's theorem is the reason why we are going to focus on chordal and co-comparability graphs when proving non-capturing results in Section 4.2. By their theorem, the two graph classes are arguably very natural superclasses of interval graphs.

Interval graphs are a classical example of an *intersection graph class* (see Section 2.1.2). Thus, interval graphs are closed under taking induced subgraphs, as it is the case for any intersection graph class (Lemma 2.1.1). Any graph class \mathcal{G} that is closed under taking induced subgraphs can

¹Gilmore and Hoffman actually show that interval graphs are precisely those co-comparability graphs that do not contain an induced 4-cycle. In fact, it is not hard to see that complements of comparability graphs cannot contain induced k -cycles for $k \geq 5$.

also be defined by a possibly infinite list of *forbidden induced subgraphs*, for example all those graphs not in \mathcal{G} that are minimal with respect to the relation of being an induced subgraph. A complete infinite family of forbidden induced subgraphs defining the class of interval graphs is given by Lekkerkerker and Boland in [LB62].

4.1.2 Deciding the classes of interval graphs, chordal graphs, and comparability graphs

By the work of Reif, it is known that the problems of recognizing the classes of chordal graphs, comparability graphs, and interval graphs can all be logspace-reduced to the undirected graph non-reachability problem [Rei84, Theorem 5.6]. Consequently, by Reingold's Theorem 2.1.2, all these graph classes can be recognized in LOGSPACE. Reif uses well-known characterization theorems for the above graph classes in order to show his reduction. However, his method depends on the construction of a minimum spanning tree, which can be done in logspace but requires an ordering of the graph's edges to produce suitable edge weights. Thus, his results do not easily extend to our logical frameworks where we have no auxiliary order available on the graph's vertices or edges.

It turns out that the above classes can in fact be defined without any ordering on the graph in STC. This insight slightly generalizes Reif's theorem and shows logical definability for all the logics relevant in this chapter. Even though these results are not hard to establish, it seems that they have not been taken note of previously, so we prove them in detail here.

Lemma 4.1.8. *The class of comparability graphs is definable in STC.*

The proof of Lemma 4.1.8 relies on the characterization theorem of comparability graphs by Gilmore and Hoffman, which is also used by Reif. If C is a cycle in a graph G and x, y are vertices on C which are not neighbors in C , then an edge between x and y is called a *chord* of C . If x and y are separated by only one vertex in C , then an edge between them is called a *triangular chord*.

Theorem 4.1.9 (Gilmore and Hoffman [GH64]). *A graph G is a comparability graph if and only if each odd cycle has at least one triangular chord.* ■

Proof of Lemma 4.1.8. Let $G = (V, E)$ be an undirected graph which we want to test for being a comparability graph. Consider the graph $G' = (V', E')$ defined on the vertex set $V' = V^2$ and $E' := E'_1 \cup E'_2$ where

$$\begin{aligned} (u, v)(u, w) \in E'_1 & :\Leftrightarrow v \neq w, \quad uv, uw \in E, \text{ and } vw \notin E, \\ (v, u)(w, u) \in E'_2 & :\Leftrightarrow v \neq w, \quad uv, uw \in E, \text{ and } vw \notin E. \end{aligned}$$

As E' is symmetric, G' is an undirected graph. The pairs (x, y) reachable from an edge (u, v) are called the *implication class of (u, v)* . Implication classes of edges were first introduced by Gallai in [Gal67] in the context of modular decomposition for finding transitive orientations of comparability graphs. The ordered version of implication classes which we are using here was first outlined by Kozen, Vazirani, and Vazirani in [KVV85], implicitly comprising this proof.

The significance of implication classes in a comparability graph is the following: If \mathcal{D} is a transitive orientation of the edges of G and $(u, v)(u, w) \in E'$, say, then uv and uw must both be directed towards u or away from u , as otherwise \mathcal{D} would not be transitive. Arguing inductively, if (u, v) and (x, y) are in the same implication class and uv is directed from u to v , then the edge xy must be directed from x to y . Thus, if there is an edge uv in G so that (v, u) is reachable from (u, v) in G' , then G cannot be a comparability graph.

Conversely, assume that G is not a comparability graph. We could now argue directly that some (u, v) and (v, u) must be in the same implication class by essentially invoking the relevant

4 Capturing Results on Classes of Graphs

theorems about modular decomposition (cf. [Gal67]), yet Gilmore and Hoffman's Theorem 4.1.9 offers a handy shortcut. By their criterion, there is an odd cycle C in G without triangular chords. If uv is any edge of C , then the absence of triangular chords and the odd length of C immediately imply that (v, u) is reachable from (u, v) in G' . Since it is easily definable in STC whether there is an edge uv of G for which (v, u) is reachable from (u, v) in G' , Lemma 4.1.8 follows. \square

The corresponding result for chordal graphs is even easier to show.

Lemma 4.1.10. *The class of chordal graphs is STC-definable.*

Proof. Simply check that for any induced simple path \overrightarrow{abcd} of length 3, there is no path from a to d which avoids b, c , and all their neighbors (except a and d). \square

By Gilmore and Hoffman's Theorem 4.1.7, the class of interval graphs is equal to the class of chordal co-comparability graphs. Thus, checking G for chordality and G^c for being a comparability graph is sufficient to determine if G is an interval graph.

Corollary 4.1.11. *The class of interval graphs is STC-definable.* \square

4.1.3 Modular decomposition trees

The canonization of interval graphs in Section 4.3 will implicitly rely on a specific decomposition of graphs known as the *modular decomposition*. This decomposition was first introduced by Gallai [Gal67] and used there not for algorithmic purposes, but for proving a characterization theorem of comparability graphs (which is alluded to in our proof of Lemma 4.1.8). It is based on the central notion of a *module* in a graph $G = (V, E)$: a set of vertices M is a module if for all vertices $v \in V \setminus M$ either $\{v\} \times M \subseteq E$ or $\{v\} \times M \cap E = \emptyset$. The vertex set V and all vertex sets of size 1 are modules by this definition, called *trivial modules*.

If G is not connected, then its connected components M_1, \dots, M_k are clearly modules (each of which is non-trivial if it contains more than one vertex). Similarly, if G^c is not connected, then the connected components M_1, \dots, M_k of G^c are modules in G . Gallai [Gal67] shows that if both G and G^c are connected, then the set of maximal proper modules M_1, \dots, M_k of G is a partition of G 's vertex set. In all three cases, the properties of modules imply that each pair of modules $M_i, M_j, i \neq j$, is either completely connected or completely disconnected. So if \sim is the equivalence relation corresponding to the partition (M_1, \dots, M_k) , then G is completely determined by G/\sim and all the graphs $G[M_1], \dots, G[M_k]$. By decomposing the $G[M_i], i \in [k]$, inductively until we arrive at singular sets, we obtain G 's modular decomposition tree. This decomposition is uniquely determined for every graph G [Gal67].

Modular decomposition will appear as a natural by-product in our canonization procedure of interval graphs in Section 4.3. We will not be defining the modular decomposition tree explicitly there, but it is not hard to see how to recover it from our construction. The main difference between our decomposition and Gallai's is that we do not bother to decompose sets of connected twins any further since we can handle them perfectly well with our methods. In fact, it is not hard to see that the modular decomposition tree of any graph is FP+C-definable with Gallai's methods ([Gal67], also see the proof of Lemma 4.1.8).

4.1.4 Existing algorithms for interval graphs

In the past decades, there has been persistent interest in the algorithmic aspects of interval graphs, spurred on by their practical applicability to such areas as DNA sequencing (cf. [ZSF⁺94]) or scheduling problems (cf. [Möh84]). In 1976, Booth and Lueker presented the first recognition algorithm for interval graphs [BL76], running in time linear in the number of vertices and edges, which they followed up by a linear-time interval graph isomorphism algorithm [LB79]. These

4.2 Logics not capturing complexity classes on graph classes

algorithms are based on a special data structure called *PQ-trees*. By pre-processing the graph's modular decomposition tree, Hsu and Ma [HM99] later presented a simpler linear-time recognition algorithm that avoids the use of PQ-trees. Habib et al. [HMPV00] achieve the same time bound employing the lexicographic breadth-first search of Rose, Tarjan, and Lueker [RTL76] in combination with smart pivoting. A parallel AC^2 algorithm was given by Klein in [Kle96].

All of the above algorithms have in common that they compute a *perfect elimination ordering* (peo) of the graph's vertices. This ordering has the property that for every vertex, its neighborhood among its successors in the ordering forms a clique. Rose [Ros70] shows that a graph has a peo if and only if it is chordal (see also [FG65]), and the above methods determine whether a graph is an interval graph in linear time once a peo is known.

Unfortunately, the construction of a peo by lexicographic breadth-first search needs to examine the children of a vertex in some fixed order. Such an ordering is not available, though, when defining properties of the bare unordered graph structure by means of logic. Klein's parallel construction of a peo in [Kle96] is based on partition refinement and might guide the way to defining a peo in fixed-point logic, but it seems out of reach to implement this in STC+C or LOGSPACE. This would be necessary to recover the full spectrum of our results here.

Another type of interval graph algorithms is based on the computation of a transitive orientation of the graph's non-edges using modular decomposition (cf. [Gal67]). This makes sense since interval graphs are co-comparability graphs. Kozen, Vazirani, and Vazirani use this approach in [KVV85] to design an NC parallel algorithm for finding an interval representation of an interval graph. They do not give a precise bound on the depth of the circuits they use. It seems, though, as if their modular decomposition and transitive orientation algorithm can be implemented in LOGSPACE using Reingold's Theorem 2.1.2, thus putting all of their computation of an interval representation into logspace. However, it remains elusive whether their method can be modified to yield an interval graph isomorphism decision procedure and a canonical form for interval graphs.

We note that an algorithmic implementation of our method would be inferior to the existing linear-time algorithms for interval graphs. Given that our method must rely entirely on the inherent structure of interval graphs and not on an additional ordering of the vertices, we reckon that is the price to pay for the disorderliness of the graph structure.

The main commonality of existing interval graph algorithms and the canonical form developed here is the construction of a *modular decomposition* of the graph. Modules are subgraphs which interact with the rest of the graph in a uniform way, and they play an important algorithmic role in the construction of modular decomposition trees (cf. [BLS99]). As a by-product of our approach here, we obtain a specific modular decomposition tree that is FP+C-definable (in fact even STC-definable). Such modular decompositions are fundamentally different from *tree decompositions*, which are the ubiquitous tool of FP+C-canonization proofs for the aforementioned minor-free graph classes (cf. [Gro08] for a survey of tree decompositions in this context). Since tree decompositions do not appear to be very useful for defining canonical forms on subgraph-free graph classes, showing the definability of modular decompositions may be a foundation for the systematic study of capturing results on these graph classes.

4.2 Logics not capturing complexity classes on graph classes

This section contains negative results of FP+C not capturing PTIME on a number of graph classes, proving Theorem 4.1 along the way. In order to aid in structuring these proofs, Section 4.2.1 first establishes a general theorem on how to transfer the failure to capture PTIME from one graph class to another. Section 4.2.2 then contains all of our non-capturing results. In particular, such a result will be shown for bipartite graphs (Theorem 4.2.9) using a simple construction and the machinery of graph interpretations (see Definition 2.3.19). For the classes of comparability and

co-comparability graphs the result will then follow. Changing our proof only slightly, we also show that FP+C does not capture PTIME on the classes of split and chordal graphs.

4.2.1 Non-capturing reductions

We first turn to a general-purpose theorem that will help us show the results in the next section. The main end of Theorem 4.2.2 is not so much the statement in itself but its role of encapsulating the reasoning behind showing how non-capturing results carry over to new graph classes. All the same, we will discuss the concept of such reductions between graph classes afterwards and compare it to the notion of graph classes being isomorphism-complete.

Before stating Theorem 4.2.2, we need some definitions. Let K be a complexity class and let g be some function. We say that K is *closed under composition with g* if for all languages $A \in K$, the language $g^{-1}(A)$ is decidable in K . $g^{-1}(A)$ denotes the *pre-image* of A under g and deciding it amounts to first executing g and then running the machine deciding A on the output of g . For example, if g is a polynomial-time computable function, then PTIME is closed under composition with g .

For the upcoming definitions and results recall that we always assume all graph classes to be *closed under isomorphism*. Also, we always assume functions γ from graphs to graphs to be *isomorphism-invariant*. γ is an *injective function on graphs* if for all graphs G, H we have $G \cong H$ if and only if $\gamma(G) \cong \gamma(H)$.

Definition 4.2.1. Let K be a complexity class, let \mathcal{G} and \mathcal{H} be graph classes and let $\gamma : \mathcal{G} \rightarrow \mathcal{H}$ be an injective function on graphs. We say that γ is *K-reversible* if the following conditions are satisfied:

- there is a function δ mapping graphs to graphs which is a left inverse to γ , meaning that for all $G \in \mathcal{G}$ we have $\delta(\gamma(G)) \cong G$, and
- K is closed under composition with δ .²

Recall that by Lemma 2.3.17, the logics FO, DTC, FP+C, and virtually all other logics we are considering in this work are closed under logical reductions (see Definition 2.3.16). If K is a complexity class and \mathcal{G} is a graph class, then we say that a property $Q \subset \mathcal{G}$ is *K-decidable on \mathcal{G}* if there is $Q' \in K$ so that $Q = Q' \cap \mathcal{G}$. Similarly, for a logic L , $Q \subset \mathcal{G}$ is *L-definable on \mathcal{G}* if there is a sentence φ of L so that Q is precisely the set of φ 's models from \mathcal{G} . Figure 4.1 illustrates the setup of non-capturings as described by the following theorem.

Theorem 4.2.2 (Non-capturing reductions). *Let L and F be logics so that L is closed under F -reductions. Let \mathcal{G} and \mathcal{H} be classes of graphs and let K be a complexity class. Suppose there is a query on \mathcal{G} which is decidable in K , but not definable by L on \mathcal{G} . If there is a K -reversible F -graph interpretation Γ from \mathcal{G} to \mathcal{H} , then there is a K -decidable query on \mathcal{H} which is not L -definable, so L does not capture K on \mathcal{H} .*

Proof. Suppose for contradiction that L does capture K on \mathcal{H} . Let $Q \subset \mathcal{G}$ be the K -decidable query on \mathcal{G} which is not L -definable and let $Q' \in K$ such that $Q = Q' \cap \mathcal{G}$. We denote the K -computable inverse of Γ by Δ .

Let $\mathcal{R} := \Delta^{-1}(Q') \cap \mathcal{H}$ and observe that \mathcal{R} is K -decidable on \mathcal{H} since K is closed under composition with Δ . Since L captures K on \mathcal{H} , there is a sentence φ of L defining \mathcal{R} on \mathcal{H} . As L is closed under F -reductions, there is an L -sentence $\varphi^{-\Gamma}$ defining $\mathcal{R}^{-\Gamma} := \{G \in \mathcal{G} \mid \Gamma[G] \in \mathcal{R}\}$ on

²I.e. K is closed under composition with the function corresponding to δ which maps encodings of graphs to encodings of graphs.

\mathcal{G} . Then $G \cong \Delta(\Gamma[G]) \in \mathcal{Q}'$ for all $G \in \mathcal{R}^{-\Gamma}$ and $\varphi^{-\Gamma}$ defines $\mathcal{R}^{-\Gamma} = \mathcal{Q}$ on \mathcal{G} , contradicting our initial assumption. \square

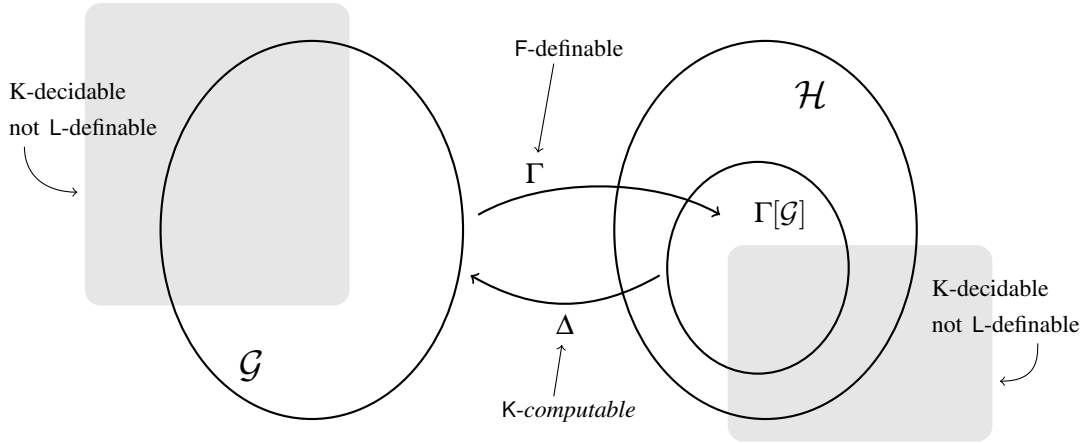


Figure 4.1: Non-capturing reduction.

Remark 4.2.3. Since we are only considering logics here which are closed under logical reductions, we could also have formulated Theorem 4.2.2 without reference to another logic F defining the reduction Γ . However, this differentiation enables us to keep track of the complexity of reductions. In particular, when there is a very simple graph interpretation Γ , then Γ establishes non-capturing reductions for all those logics L which are closed under Γ at the same time. In that way, we may also consider non-capturing reductions for logics which are not closed under general logical reductions, but closed under reductions with respect to weaker logics. An example would be $\exists\text{SO}$, which is not closed under $\exists\text{SO}$ -reductions, but under FO -reductions.

Remark 4.2.4. We may also have formulated Theorem 4.2.2 in a positive way: if there is a K -reversible L -graph interpretation from \mathcal{G} to \mathcal{H} and L captures K on \mathcal{H} , then L also captures K on \mathcal{G} . As we will mainly be concerned with extending non-capturing results to further graph classes, we have chosen the negative formulation.

Theorem 4.2.2 allows us to think of graph interpretations as reductions between graph classes which preserve the property that a logic is not capturing some complexity class. The decisive criterion above for \mathcal{G} to *reduce* to \mathcal{H} is that there must be an *injective* graph interpretation whose inverse can additionally *be computed in* K . We may say that \mathcal{H} needs to be structurally at least as rich as \mathcal{G} in order for this to be possible.

The notion of *non-capturing reductions* to a graph class is similar to the notion of a graph class being *isomorphism complete*. A class of graphs \mathcal{G} is said to be graph isomorphism complete if the isomorphism problem on general graphs reduces to the isomorphism problem of graphs from \mathcal{G} (cf. [CK80]).³ Showing that a graph class \mathcal{G} is isomorphism-complete typically involves the design of an injective PTIME-encoding of general graphs into graphs from \mathcal{G} . In our non-capturing reductions above, we also require injective encodings, but usually require these encodings to be definable by logics which are less expressive than PTIME. And while for isomorphism-completeness it is sufficient that the reduction preserves graph isomorphism, our notion requires the reduction itself to be efficiently reversible. Clearly, whenever there is any injective FP+C -graph interpretation from the class of all graphs to \mathcal{G} , then \mathcal{G} is also graph isomorphism-complete. Thus, we may view non-capturing reductions to a graph class as a refinement of the notion of isomorphism-completeness.

³In other words, we call \mathcal{G} graph isomorphism-complete if the isomorphism problem on \mathcal{G} is complete for the complexity class GI.

4 Capturing Results on Classes of Graphs

For later use, we note a couple of simple corollaries to Theorem 4.2.2.

Corollary 4.2.5. *Let $\mathcal{G} \subset \mathcal{H}$ be graph classes and let K be a complexity class. If L does not capture K on \mathcal{G} , then it does not capture K on \mathcal{H} either.*

Proof. Choose $\Gamma : \mathcal{G} \rightarrow \mathcal{H}$ to be the identity. Notice that any logic L is closed under Γ . □

Corollary 4.2.6. *Suppose $\mathcal{H} \subset \mathcal{G}$ are graph classes, K is a complexity class, and there is a K -reversible F -graph interpretation from \mathcal{G} to \mathcal{H} . If L is closed under F -reductions, then L captures K on \mathcal{G} if and only if L captures K on \mathcal{H} .*

Proof. One direction is taken care of by Corollary 4.2.5. If L does not capture K on \mathcal{G} , then there is a K -decidable query on \mathcal{G} which is not L -definable. The result follows from applying Theorem 4.2.2. □

Corollary 4.2.7. *Let \mathcal{G} be a graph class and let \mathcal{G}^c denote the class of complements of graphs in \mathcal{G} . Let K be a complexity class which is closed under composition with complementing the edge relation and let L be a logic which is closed under unary quantifier-free FO-reductions. Then L captures K on \mathcal{G} if and only if L captures K on \mathcal{G}^c .*

Proof. Suppose L does not capture K on \mathcal{G} , then there is a K -decidable query $Q \subset \mathcal{G}$ which is not L -definable. Let Γ be the obvious unary quantifier-free graph interpretation which complements the edge relation of graphs in \mathcal{G} to obtain graphs in \mathcal{G}^c . By Theorem 4.2.2, L does not capture K on \mathcal{G}^c either. Observe that Γ is equal to its own inverse as a function on graphs. By symmetry, the reverse direction is proved in the same way. □

4.2.2 Non-capturing results for FP+C

The results in this section are all based on the following theorem due to Cai, Fürer, and Immerman [CFI92].

Fact 4.2.8 (Cai, Fürer, and Immerman [CFI92]). *There is a PTIME-decidable property \mathcal{P}_{CFI} of graphs of degree 3 which is not FP+C-definable.*

The actual definition of their graph property will be given later in Chapter 5 (Definition 5.4.1), where it is shown that rank logics can define this property. The non-capturing results in this section, therefore, do not necessarily extend to rank logics. We begin by establishing a non-capturing result on bipartite graphs.

Theorem 4.2.9. *FP+C does not capture PTIME on the class of bipartite graphs.*

Proof. We apply Theorem 4.2.2. First, we exhibit an FO-graph interpretation from general graphs to the class of bipartite graphs.

For any graph $G = (V, E)$, the *incidence graph* G^I has $V \cup E$ as its vertices and $v \in V, e \in E$ are connected in G^I if $v \in e$. The sets V and E are independent sets of G^I , so it is bipartite. This transformation is definable by a binary FO-graph interpretation as follows. γ_V defines the vertex set $\{(u, u) \in V^2\} \cup \{(u, v) \in E\}$. We only want one vertex per edge in G , so we let γ_{\approx} declare all (u, v) and (v, u) to be equivalent. The edge relation γ_E (symmetrically) joins vertices (u, u) and $(u, v) \in E$. All these formulas are easily definable by (quantifier-free) FO-formulas. Clearly, $\Gamma[G] \cong G^I$ for any graph G , so Γ is a binary FO-graph interpretation from general graphs to bipartite graphs.

Now given any graph $H = (V_H, E_H)$, we can find a 2-coloring of it in linear time, if one exists, using a depth-first search (e.g. [Koz92]). Assume that H is connected. There must be a color class $W \subset V_H$ so that all vertices in W have degree 2 in H . Otherwise, it is clear that there is no graph

G such that $G^I \cong H$. If there is such a color class W , though, let G have vertices $V_G = V_H \setminus W$ and connect $u, v \in V_G$ whenever they have a common neighbor in H . It follows straight away that $G^I \cong H$.

For connected graphs H , the preceding paragraph describes a PTIME-procedure to check if H is in the image of Γ and, if so, reconstruct a preimage $\Gamma^{-1}[H]$. Note that there are at most two possible preimages, and this only happens in case both color classes consist entirely of degree-2 vertices. But in that case, H is an even-length cycle, so both preimages under Γ are cycles of half that length, thus isomorphic. For general graphs, the reconstruction has to be done for all connected components of H independently. This shows that Γ is injective and PTIME-reversible. The theorem now follows from applying Fact 4.2.8 and Theorem 4.2.2. \square

Theorem 4.1 is now a simple corollary of the following Lemma.

Lemma 4.2.10. *Every bipartite graph $G = (U \dot{\cup} V, E)$ is a comparability graph.*

Proof. A suitable partial order $<$ on $U \dot{\cup} V$ is defined by letting $u < v$ if and only if $u \in U, v \in V$, and $uv \in E$. \square

Corollary 4.2.11. *FP+C does not capture PTIME on the classes of comparability and co-comparability graphs.*

Proof. Immediate from Corollaries 4.2.5 and 4.2.7. \square

This tells us that being a comparability or co-comparability graph alone is not sufficient for a graph G to be uniformly FP+C-canonizable. Section 4.3, however, is going to show that this is possible if G is a co-comparability graph which is also chordal (i.e. an interval graph). This result is not simply a corollary of a capturing result on chordal graphs either, as we will argue now. The construction is due to Grohe [Gro10a]. As for co-comparability graphs, we will go through a smaller graph class to obtain the non-capturing result for chordal graphs. Recall that a graph is *split* if its vertices can be partitioned into a clique U and an independent set V (cp. Section 2.1.3).

Theorem 4.2.12. *FP+C does not capture PTIME on split graphs.*

Proof. We first give a binary graph interpretation into the class of split graphs. For any graph $G = (V, E)$, the *split incidence graph* G^S has vertices $V \cup E$, edges between $v \in V$ and $e \in E$ if $v \in e$, and all edges between vertices from V . Notice that G^S differs from the incidence graph G^I used in the proof of Theorem 4.2.9 only by the fact that V forms a clique in G^S . Obviously, G^S is a split graph.

Given the similarity of G^S and G^I , the analysis for split incidence graphs is analogous to the one for incidence graphs above. It is immediately clear that the transformation from G to G^S is definable by a binary quantifier-free graph interpretation Γ .

Unlike the case of incidence graphs, the reconstruction of a graph G from its split incidence graph G^S is not always uniquely possible. More concretely, the graph consisting of just one edge and the graph consisting of three isolated vertices have the same split incidence graph. Γ is injective, though, when applied only to graphs with at least 4 vertices. We show this next, and treat Γ as a graph interpretation from the class of graphs with at least 4 vertices to the class of split graphs.

So let $H = (V_H, E_H)$ be a graph and suppose that there is some other graph $G = (V_G, E_G)$ with $|G| \geq 4$ such that $G^S \cong H$. Let $U \subseteq V_H$ be those vertices of H of degree ≥ 3 . Since $V_H = V_G \cup E_G$ and all vertices from E_G have degree 2, we must have $U = V_G$ and U must be a clique. If now all vertices from $V_H \setminus U$ really have degree 2 and U is a clique, it is clear how to reconstruct G . Thus, we can decide the image of Γ and reverse it in PTIME (or even in FO).

4 Capturing Results on Classes of Graphs

Given that there are only finitely many graphs with at most 3 vertices, the query from Fact 4.2.8 is not FP+C-definable on the class of graphs with at least 4 vertices either.⁴ The theorem follows from applying Theorem 4.2.2. \square

The non-capturing result for chordal graphs now follows from the following lemma. This result is due to Földes and Hammer who first studied split graphs, and who actually proved that split graphs are precisely those chordal graphs whose complement is also chordal.

Lemma 4.2.13 ([FH77]). *Every split graph $G = (K \cup V, E)$ is chordal.*

Proof. Suppose $v_1, \dots, v_k \in K \cup V$ form a chordless cycle in G . Since K is a clique, at most 2 of the v_i can be from K , and they have to occur consecutively. Since V is an independent set, at most half of the v_i can be from V , and they can never occur as neighbors. It follows immediately that $k \leq 3$. \square

Corollary 4.2.14 (Grohe [Gro10a]). *FP+C does not capture PTIME on the class of chordal graphs.* \square

Remark 4.2.15. The proofs of Lemmas 4.2.9 and 4.2.12 admit even stronger conclusions. Since the graph interpretations used in both of them are quantifier-free, we can apply Corollary 4.2.6 to get: any logic closed under quantifier-free FO-reductions captures PTIME on the class of bipartite graphs (respectively comparability/ co-comparability graphs, split graphs, chordal graphs) if and only if it captures PTIME on the class of all graphs. Of course, requiring a logic for PTIME to be closed under quantifier-free reductions is a very natural condition since PTIME itself is closed under such reductions.

Let us conclude this section by noting some non-capturing results for further intersection graph classes. A t -interval graph is the intersection graph of sets which are the union of t intervals. By a result of Griggs and West [GW79], any graph of maximum degree 3 is a 2-interval graph, so Fact 4.2.8 directly implies that FP+C does not capture PTIME on t -interval graphs for $t \geq 2$.

A graph is called a *grid intersection graph* if it is the intersection graph of axis-parallel line segments in the plane. Thus, grid intersection graphs are an extension of the class of interval graphs. In [Ueh08], Uehara shows that this class is graph isomorphism-complete. For that, he gives an encoding of general graphs into grid intersection graphs that can be directly phrased as an FO-graph interpretation and which is also easily reversible in PTIME. It follows that FP+C does not capture PTIME on the class of grid intersection graphs.

Uehara's result also shows a non-capturing result for graphs of higher *boxicity*. A graph has boxicity d if it is the intersection graph of axis-parallel boxes in \mathbb{R}^d (cf. [BLS99]). Boxicity-1 graphs are simply the interval graphs. It is clear that the class of boxicity- $(d+1)$ graphs contains all boxicity- d graphs for all $d \in \mathbb{N}$. Now boxicity-2 graphs contain the class of grid intersection graphs, which can be seen by slightly thickening the line segments of any intersection model for a grid intersection graph. Hence, FP+C does not capture PTIME on the class of boxicity- d graphs for all $d \geq 2$.

4.3 Capturing PTIME on interval graphs

The goal of this section is to prove Theorem 4.2 by canonization (cf. Section 2.4). We will exhibit a numeric FP+C-formula $\varepsilon(x, y)$ so that for any interval graph $G = (V, E)$, $([|V|], \varepsilon^G[\cdot])$ defines a graph on the number sort of FP+C which is isomorphic to G . The canonization essentially consists of finding the lexicographic leader among all possible interval representations of G . For this, as discussed in Section 4.1.1, it is enough to bring the maximal cliques of G in the right linear order.

⁴Or notice that any non-empty 3-regular graph has at least 4 vertices, anyways.

The first lemma shows that the maximal cliques of G are FO-definable. Recall that $N[v]$ denotes the closed neighborhood of a vertex v , i.e. the set containing v and all vertices adjacent to v .

Lemma 4.3.1. *Let $G = (V, E)$ be an interval graph and let M be a maximal clique of G . Then there are vertices $u, v \in M$, not necessarily distinct, such that $M = N[u] \cap N[v]$.*

This is fairly intuitive. Consider the minimal interval representation of a graph G depicted in Figure 4.2. Max cliques 1 and 3 are precisely the neighborhoods of vertices a and e , respectively. The vertex pairs (a, a) , (a, b) , and (a, c) all define max clique 1, and similarly three different vertex pairs define max clique 3. Max clique 2 is not the neighborhood of any single vertex but it is uniquely defined by $N[b] \cap N[d]$.

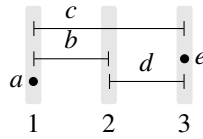


Figure 4.2: An interval graph.

Proof of Lemma 4.3.1. Let \mathcal{I} be a minimal interval representation of G , and let M be a maximal clique of G . By Lemma 4.1.3, $\bigcap_{v \in M} I_v = \{k\}$ for some $k \in \mathbb{N}$. Thus, there have to be vertices $v \in M$ with $\min I_v = k$ and $u \in M$ with $\max I_u = k$ (u, v do not need to be distinct). Since $I_v \cap I_u = \{k\}$, any vertex x adjacent to both u and v must satisfy $k \in I_x$. But then x is adjacent to all vertices in M , so by M 's maximality we have $x \in M$. Hence, $N[u] \cap N[v] \subseteq M$, and since $M \subseteq N[u] \cap N[v]$ is true for any $u, v \in M$, the lemma is proven. \square

Now, whether or not a vertex pair $(u, v) \in V^2$ defines a max clique is easily definable in FO, as is the equivalence relation on V^2 of vertex pairs defining the same max clique. Lemma 4.3.1 tells us that *all* max cliques can be defined by such vertex pairs. Recall that the span of a vertex $v \in V$ is the number of max cliques of G that v is contained in. Since equivalence classes can be counted by Lemma 2.4.3, $\text{span}(x)$ is FP+C-definable on the class of interval graphs by a counting term with x as a free vertex variable.

Generally representing max cliques by pairs of variables $(x, y) \in V^2$ allows us to treat max cliques as first-class objects that can be quantified over. For reasons of conceptual simplicity, the syntactic overhead which is necessary for working with this representation will not be made explicit in the remainder of this section.

4.3.1 Extracting information about the order of maximal cliques

Now that we are able to handle maximal cliques, we would like to simply pick an end of the interval graph G and work with the order which this choice induces on the rest of the maximal cliques. Of course, the choice of an end does not necessarily impose a linear order on the maximal cliques. We will deal with this later. The following recursive procedure turns out to recover all the information about the order of the max cliques induced by choosing an end of G .

Let \mathcal{M} be the set of maximal cliques of an interval graph $G = (V, E)$ and let $M \in \mathcal{M}$. The binary relation \prec_M is defined recursively on the elements of \mathcal{M} as follows:

$$\begin{aligned} \text{Initialization: } & M \prec_M C \text{ for all } C \in \mathcal{M} \setminus \{M\} \\ C \prec_M D & \text{ if } \begin{cases} \exists E \in \mathcal{M} \text{ with } E \prec_M D \text{ and } (E \cap C) \setminus D \neq \emptyset & \text{or} \\ \exists E \in \mathcal{M} \text{ with } C \prec_M E \text{ and } (E \cap D) \setminus C \neq \emptyset. \end{cases} \quad (\star) \end{aligned}$$

4 Capturing Results on Classes of Graphs

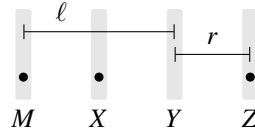


Figure 4.3: An interval graph illustrating (\star) .

The interval representation in Figure 4.3 of a graph G illustrates this definition. Suppose we have picked max clique M , then $X \prec_M Z$ and $Y \prec_M Z$ follow since $\ell \in X \cap Y \cap M \setminus Z$ and $M \prec_M Z$ by the initialization step. In a second step, it is determined that $X \prec_M Y$ since $r \in Y \cap Z \setminus X$ and $X \prec_M Z$. So in this example, \prec_M actually turns out to be a strict linear order on the max cliques of G . This is not the case in general, but \prec_M will still be useful when M is a possible end of G .

It is readily seen how to define \prec_M using the inflationary fixed-point operator, where maximal cliques are defined by pairs of vertices from G . In fact, by exploiting the definition's symmetry, we can show that \prec_M can even be defined through a reachability query in an undirected graph. This will be used in Section 4.4 to show that interval graph canonization can also be done in LOGSPACE.

Lemma 4.3.2. *Let G be an interval graph. The relation \prec_M is STC-definable over G for any max clique M of G .*

Proof. Consider the graph O_M which has pairs of max cliques from \mathcal{M} as its vertices, and in which two vertices (A, B) and (C, D) are connected by an edge whenever $A \prec_M B$ implies $C \prec_M D$ with one application of (\star) . Precisely speaking, $((A, B), (C, D)) \in E$ if and only if $B = D$ and $(A \cap C) \setminus B \neq \emptyset$ or $A = C$ and $(B \cap D) \setminus A \neq \emptyset$. Notice the symmetry of this definition, which implies that $((A, B), (C, D)) \in E$ if and only if $((C, D), (A, B)) \in E$. Thus, we may view E as a simple undirected graph.

Now we have that $A \prec_M B$ if and only if (A, B) is reachable in O_M from (M, X) for some max clique $X \neq M$. O_M 's vertex set and edge relation are FO-definable since G 's max cliques are FO-definable and, hence, there is a formula of STC deciding whether $A \prec_M B$ for given max cliques A, B, M . \square

Let us now turn to a series of lemmas proving important properties of \prec_M . We say that a binary relation R on a set A is *asymmetric* if $ab \in R$ implies $ba \notin R$ for all $a, b \in A$. In particular, asymmetric relations are irreflexive.

Lemma 4.3.3. *If \prec_M is asymmetric, then it is transitive. Thus, if \prec_M is asymmetric, then it is a strict partial order.*

Proof. By a derivation chain of length k we mean a finite sequence $X_0 \prec_M Y_0, X_1 \prec_M Y_1, \dots, X_k \prec_M Y_k$ such that $X_0 = M$ and for each $i \in [k]$, the relation $X_i \prec_M Y_i$ follows from $X_{i-1} \prec_M Y_{i-1}$ by one application of (\star) . Clearly, whenever it holds that $X \prec_M Y$ there is a derivation chain that has $X \prec_M Y$ as its last element.

So assume that \prec_M is asymmetric. Suppose $A \prec_M B \prec_M C$ and let a derivation chain (L_0, \dots, L_k) of length k be given for $A \prec_M B$. The proof is by induction on k . If $k = 0$, then $A = M$ and $A \prec_M C$ holds. For the inductive step, suppose we have shown the statement for derivation chains of length $k - 1$ and consider the second to last element L_{k-1} in the derivation chain. There are two cases:

- $L_{k-1} = (X \prec_M B)$ and there is a vertex $v \in (X \cap A) \setminus B$: By induction it holds that $X \prec_M C$. Now if we had $v \in C$, the fact that $A \prec_M B$ would imply $C \prec_M B$, which contradicts asymmetry of \prec_M . Hence, $v \notin C$ and one more application of (\star) yields $A \prec_M C$.

- $L_{k-1} = (A \prec_M X)$ and there is a vertex $v \in (X \cap B) \setminus A$: If $v \in C$, then we immediately get $A \prec_M C$. If $v \notin C$, then $X \prec_M C$. Thus, we can derive $A \prec_M X \prec_M C$ where the left derivation chain has length $k - 1$. By induction, $A \prec_M C$ follows. \square

Lemma 4.3.4. *Let $\mathcal{C} \subset \mathcal{M}$ be a set of max cliques with $M \notin \mathcal{C}$. Suppose that for all $A \in \mathcal{M} \setminus \mathcal{C}$ and any $C, C' \in \mathcal{C}$ it holds that $A \cap C = A \cap C'$. Then the max cliques in \mathcal{C} are mutually incomparable with respect to \prec_M .*

Proof. Suppose for contradiction that there are $C, C' \in \mathcal{C}$ with $C \prec_M C'$. Let $M \prec_M Y_0, X_1 \prec_M Y_1, \dots, X_k \prec_M Y_k$ be a derivation chain for $C \prec_M C'$ as in the proof of Lemma 4.3.3. Since $X_k = C, Y_k = C'$, and $M \notin \mathcal{C}$, there is a largest index i so that either X_i or Y_i is not contained in \mathcal{C} .

If $X_i \notin \mathcal{C}$, then $X_{i+1} \in \mathcal{C}$ and $Y_i = Y_{i+1} \in \mathcal{C}$ and it holds that $X_i \cap X_{i+1} \setminus Y_{i+1} \neq \emptyset$. Consequently, $X_i \cap X_{i+1} \neq X_i \cap Y_{i+1}$, contradicting the assumption of the lemma. Similarly, if $Y_i \notin \mathcal{C}$, then $Y_{i+1} \in \mathcal{C}$ and $X_i = X_{i+1} \in \mathcal{C}$ and it holds that $Y_i \cap Y_{i+1} \setminus X_{i+1} \neq \emptyset$. Thus, $Y_i \cap Y_{i+1} \neq Y_i \cap X_{i+1}$, again a contradiction. \square

In fact, there is a converse to Lemma 4.3.4 when the set of \prec_M -incomparable max cliques is maximal.

Lemma 4.3.5. *Suppose M is a max clique of G and \mathcal{C} is a maximal set of \prec_M -incomparable max cliques. Let $D \in \mathcal{M} \setminus \mathcal{C}$. Then $D \cap C = D \cap C'$ for all $C, C' \in \mathcal{C}$.*

Proof. We say that a max clique A splits a set of max cliques \mathcal{X} if there are $X, Y \in \mathcal{X}$ so that $A \cap X \neq A \cap Y$. If in addition to splitting \mathcal{X} , A is also \prec_M -comparable to all the elements in \mathcal{X} , then either $A \cap X \setminus Y \neq \emptyset$ or $A \cap Y \setminus X \neq \emptyset$ and one application of (\star) implies that X and Y are comparable.

Suppose for contradiction that there is $X_1 \in \mathcal{M} \setminus \mathcal{C}$ splitting \mathcal{C} . We greedily grow a list of max cliques X_i with the property that $X_i \in \mathcal{M} \setminus (\mathcal{C} \cup \{X_1, \dots, X_{i-1}\})$ splits the set $\mathcal{X}_{i-1} := \mathcal{C} \cup \{X_1, \dots, X_{i-1}\}$. The list X_1, \dots, X_k is complete when no further max clique splits the set \mathcal{X}_k .

Suppose that $M \notin \mathcal{X}_k$. For any $D \in \mathcal{M} \setminus \mathcal{X}_k$ we have $D \cap X = D \cap X'$ for all $X, X' \in \mathcal{X}_k$, so Lemma 4.3.4 implies that the max cliques in \mathcal{X}_k are \prec_M -incomparable. However, this is impossible since we assumed $\mathcal{C} \subsetneq \mathcal{X}_k$ to be maximal. Therefore, $M \in \mathcal{X}_k$.

Now let Y_1, \dots, Y_ℓ be a shortest list of max cliques from \mathcal{X}_k so that $Y_\ell = M$ and each Y_j splits $\mathcal{Y}_{j-1} := \mathcal{C} \cup \{Y_1, \dots, Y_{j-1}\}$.

Claim 4.3.6. For all $j \in [2, \ell]$, $Y_j \cap Y_{j-1} \neq Y_j \cap A$ for all $A \in \mathcal{Y}_{j-2}$.

Proof of Claim 4.3.6. Consider $j = \ell$ and suppose that there is $A \in \mathcal{Y}_{\ell-2}$ with $Y_\ell \cap A = Y_\ell \cap Y_{\ell-1}$. As Y_ℓ splits $\mathcal{Y}_{\ell-1}$, there must be some $B \in \mathcal{Y}_{\ell-2}$ such that $Y_\ell \cap B \neq Y_\ell \cap Y_{\ell-1}$. But then Y_ℓ already splits $\mathcal{Y}_{\ell-2}$, so by eliminating $Y_{\ell-1}$ we could make the list shorter.

Inductively, suppose that the claim holds for all $i > j$, but not for j . Then there are $A, B \in \mathcal{Y}_{j-2}$ such that $Y_j \cap B \neq Y_j \cap Y_{j-1} = Y_j \cap A$, so Y_j already splits \mathcal{Y}_{j-2} . Removing Y_{j-1} from the list gives us a shorter list in which Y_i still splits \mathcal{Y}_{i-1} for all $i > j$ because of our inductive assumption. As we assumed our list to be shortest, this concludes the inductive step. *Claim 4.3.6* \square

We now argue once again inductively backwards down the list Y_1, \dots, Y_ℓ with the goal of showing that Y_j is \prec_M -comparable to all max cliques in \mathcal{Y}_{j-1} . Certainly, this is true for $Y_\ell = M$ and $\mathcal{Y}_{\ell-1}$. Assume that Y_j is comparable to all max cliques in \mathcal{Y}_{j-1} for $j \in [2, \ell]$. Since $Y_j \cap Y_{j-1} \neq Y_j \cap A$ for all $A \in \mathcal{Y}_{j-2}$ by Claim 4.3.6, it follows that Y_{j-1} is comparable to all max cliques in \mathcal{Y}_{j-2} .

Now Y_1 is comparable to all max cliques in \mathcal{C} . Since Y_1 splits \mathcal{C} , there are $C, C' \in \mathcal{C}$ so that $C \prec_M C'$, contradicting our assumption that the max cliques in \mathcal{C} are \prec_M -incomparable. Therefore we conclude that there is no $D \in \mathcal{M} \setminus \mathcal{C}$ splitting \mathcal{C} . \square

4 Capturing Results on Classes of Graphs

Lemma 4.3.5 says that incomparable max cliques interact with the rest of \mathcal{M} in a uniform way. Let us make this notion more precise. A *module* of $G = (V, E)$ is a set $S \subseteq V$ so that for any vertex $x \in V \setminus S$, S is either completely connected or completely disconnected to x . In other words, for all $u, v \in S$ and all $x \in V \setminus S$ it holds that $ux \in E \leftrightarrow vx \in E$. Figure 4.4 illustrates the occurrence of a module in an interval graph. Using \prec_M we gain access to such modules.

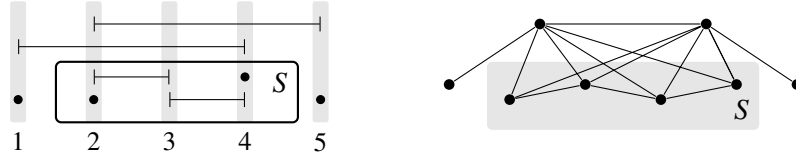


Figure 4.4: A typical module in an interval graph.

Corollary 4.3.7. *Suppose M is a max clique of G and \mathcal{C} is a maximal set of \prec_M -incomparable max cliques. Then*

- $S_{\mathcal{C}} := \bigcup_{C \in \mathcal{C}} C \setminus \bigcup_{D \in \mathcal{M} \setminus \mathcal{C}} D$ is a module of G , and
- $S_{\mathcal{C}} = \{v \in \bigcup \mathcal{C} \mid \text{span}(v) \leq |\mathcal{C}|\}$.

Proof. Let $u, v \in S_{\mathcal{C}}$ and $x \in V \setminus S_{\mathcal{C}}$ and suppose that $ux \in E$, but $vx \notin E$. There is a max clique $C \in \mathcal{M}$ with $u, x \in C$, but $v \notin C$, and since $u \in S_{\mathcal{C}}$ we must have $C \in \mathcal{C}$. By the definition of $S_{\mathcal{C}}$, x is also contained in some max clique $D \in \mathcal{M} \setminus \mathcal{C}$. Let C' be some max clique in \mathcal{C} containing v , so $x \notin C'$. Then $D \cap C \neq D \cap C'$, contradicting Lemma 4.3.5.

For the second statement, let $v \in \bigcup \mathcal{C}$. If $v \in S_{\mathcal{C}}$, then clearly $\text{span}(v) \leq |\mathcal{C}|$. But if $v \notin S_{\mathcal{C}}$, then it is contained in some $D \in \mathcal{M} \setminus \mathcal{C}$, and by Lemma 4.3.5 v must also be contained in all max cliques in \mathcal{C} . Thus, $\text{span}(v) > |\mathcal{C}|$, proving the statement. \square

Corollary 4.3.7 gives us a good characterization of the modules which emerge incidentally from the definition of \prec_M . This will be central in our canonization procedure of G . There is another result following from Lemma 4.3.5 which proves that \prec_M has a particularly nice structure. Intuitively, max cliques A and B are incomparable with respect to \prec_M because they interact with the rest of the graph *in the same uniform manner*. The same can be said if max cliques B and C are also incomparable. By this reasoning, A and C are then incomparable as well because they interact *in the same uniform manner*. So incomparability with respect to \prec_M appears to be an equivalence relation, which promotes \prec_M from a strict partial order to a strict weak order. Corollary 4.3.8 confirms this intuition.

Corollary 4.3.8. *If M is a max clique of G so that \prec_M is a strict partial order, then \prec_M is a strict weak order.*

Proof. We need to prove that \prec_M -incomparability is a transitive relation of G 's max cliques. So let (A, B) and (B, C) be incomparable pairs with respect to \prec_M . Let \mathcal{C}_{AB} and \mathcal{C}_{BC} be maximal sets of incomparables containing $\{A, B\}$ and $\{B, C\}$, respectively. By Lemma 4.3.5, we have $D \cap X = D \cap B = D \cap Y$ for every $X, Y \in \mathcal{C}_{AB} \cup \mathcal{C}_{BC}$ and $D \in \mathcal{M} \setminus (\mathcal{C}_{AB} \cup \mathcal{C}_{BC})$. As $M \notin \mathcal{C}_{AB} \cup \mathcal{C}_{BC}$ Lemma 4.3.4 implies that the max cliques in $\mathcal{C}_{AB} \cup \mathcal{C}_{BC}$ are \prec_M -incomparable, so in particular A and C are incomparable with respect to \prec_M . \square

At this point, let us put the pieces together and show that picking an arbitrary max clique M as an end of G and defining \prec_M is a useful way to obtain information about the structure of G .

Lemma 4.3.9. *Let M be a max clique of an interval graph G . If \prec_M is a strict weak order, then there is a minimal interval representation \mathcal{I} so that $\triangleleft_{\mathcal{I}}$ extends \prec_M . Also, \prec_M is a strict weak order if and only if M is a possible end of G .*

Proof. If M is a possible end of G , then let \mathcal{I} be a minimal interval representation of G which has M as its first clique. Let $\triangleleft_{\mathcal{I}}$ be the linear order \mathcal{I} induces on the max cliques of G . In order to show asymmetry of \prec_M it is enough to observe that as relations, we have $\prec_M \subseteq \triangleleft_{\mathcal{I}}$. It is readily verified that this holds true of the initialization step in the recursive definition of \prec_M , and that whenever max cliques C, D satisfy (\star) with \prec_M replaced by $\triangleleft_{\mathcal{I}}$, then it must hold that $C \triangleleft_{\mathcal{I}} D$. This shows asymmetry, and by Lemma 4.3.3 and Corollary 4.3.8 \prec_M is a strict weak order.

Conversely, suppose \prec_M is a strict weak order. The first aim is to turn \prec_M into a linear order. Let \mathcal{C} be a maximal set of \prec_M -incomparable max cliques, and recall the set $S_{\mathcal{C}} = \bigcup_{C \in \mathcal{C}} C \setminus \bigcup_{D \in \mathcal{M} \setminus \mathcal{C}} D$. Since $G[S_{\mathcal{C}}]$ is an interval graph, we can pick an interval representation $\mathcal{I}_{S_{\mathcal{C}}}$ for $G[S_{\mathcal{C}}]$. Since $S_{\mathcal{C}}$ is a module, $C \cap S_{\mathcal{C}} \not\subseteq C' \cap S_{\mathcal{C}}$ for any $C \neq C'$ from \mathcal{C} and the set of max cliques of $G[S_{\mathcal{C}}]$ is given by $\{C \cap S_{\mathcal{C}} \mid C \in \mathcal{C}\}$. Thus, $\mathcal{I}_{S_{\mathcal{C}}}$ induces a linear order $\triangleleft_{\mathcal{C}}$ on the elements of \mathcal{C} . Now let $C \triangleleft_M D$ if and only if $C \prec_M D$, or $C, D \in \mathcal{C}$ for some maximal set of \prec_M -incomparables \mathcal{C} and $C \triangleleft_{\mathcal{C}} D$. This is a strict linear order since \prec_M is a strict weak order. Also, \triangleleft_M extends \prec_M . We claim that \triangleleft_M is an ordering of the max cliques which is isomorphic to the linear order induced by some interval representation of G . This will also imply that M is a possible end of G .

In order to prove the claim, it is enough to show that each vertex $v \in V$ is contained in consecutive max cliques. Suppose for contradiction that there are max cliques $A \triangleleft_M B \triangleleft_M C$ and v is contained in A and C , but not in B . Certainly, this cannot be the case if A, B, C are incomparable with respect to \prec_M , so assume without loss of generality that $A \prec_M B$. Now, since $v \in (A \cap C) \setminus B$, (\star) implies that $C \prec_M B$, which contradicts the asymmetry of \triangleleft_M . \square

Remark 4.3.10. The recursive definition of \prec_M and Lemmas 4.3.3 through 4.3.8 do not depend on G being an interval graph. However, the proof of Lemma 4.3.9 shows that \prec_M only turns out to be a partial order if the max cliques can be brought into a linear order, modulo the occurrence of modules. In particular, defining \prec_M in a general chordal graph does not yield any useful information if the graph's tree decomposition into max cliques requires a tree vertex of degree 3 or more, which is the case for all chordal graphs that are not interval graphs.

Since \prec_M is STC-definable for any max clique M and since asymmetry of \prec_M is FO-definable, Lemma 4.3.9 gives us a way to define possible ends of interval graphs in FP. Moreover, the proof of Lemma 4.3.9 and Lemma 4.3.5 show that by picking a possible end M of G , \prec_M linearly orders all max cliques except for those which interact with the rest of the graph in a uniform way and which may be arranged in more than one way when extending \prec_M to a linear order. Thus we may say that \prec_M contains precisely the ordering imposed on G 's max cliques by the choice of M as the first clique.

4.3.2 Canonizing when \prec_M is a linear order

We now turn to defining a canonical copy of G from information about the order of max cliques. Let $G = (V, E)$ be an interval graph and let us assume here that \prec is a linear order on the max cliques which is induced by an interval representation of G . Define the binary relation $<^G$ on the vertices of G as follows. For $x \in V$, let A_x denote the \prec -least max clique of G containing x . Then let

$$x <^G y \quad :\Leftrightarrow \quad \begin{cases} A_x \prec A_y, \text{ or} \\ A_x = A_y \text{ and } \text{span}(x) < \text{span}(y). \end{cases}$$

It is readily verified that $<^G$ is a strict weak order on V , and if x, y are incomparable, then $N[x] = N[y]$. Now it is easy to canonize G : if $[v]$ denotes the equivalence class of vertices incomparable to

4 Capturing Results on Classes of Graphs

v , then $[v]$ is represented by the numbers from the interval $[a + 1, a + |[v]|]$, where a is the number of vertices which are strictly $<^G$ -smaller than v . Since all vertices in $[v]$ have precisely the same neighbors in $G \setminus [v]$ and $[v]$ forms a clique, it is also clear how to define the edge relation on the number sort. We record this result for later reference.

Lemma 4.3.11. *Let G be an interval graph, let M be a max clique of G , and suppose \prec_M is a linear order on G 's max cliques. Then $<^G$ as defined above is a strict weak order on G 's vertices with the property that vertices are incomparable if and only if they are connected twins. Consequently, there is an STC+C-formula ε defining an ordered copy of G on the number sort.* \square

4.3.3 Canonizing general interval graphs

If G is any interval graph and M is a possible end, we can adapt the ordering defined in the preceding section to define an ordering for those vertices that are not contained in a module. Let \sim_M^G be the equivalence relation on V for which $x \sim_M^G y$ if and only if $x = y$ or there is a nonsingular maximal set of incomparables \mathcal{C} with respect to \prec_M so that $x, y \in S_{\mathcal{C}}$. Denote the equivalence class of $x \in V$ under \sim_M^G by $[x]$, and let $G_M = (V / \sim_M^G, E_M) := G / \sim_M^G$ (recall from Section 2.1 that $[u][v] \in E_M \Leftrightarrow \exists x \in [u], y \in [v]$ s.t. $xy \in E$).

It follows directly from the definition of \sim_M^G that if A is a max clique which is \prec_M -comparable to all other max cliques in G , then all $v \in A$ are in singleton equivalence classes $[v] = \{v\}$. If \mathcal{C} is a nonsingular maximal set of \prec_M -incomparables in G , then $[v] = S_{\mathcal{C}}$ as sets for every $v \in S_{\mathcal{C}}$. So in G_M , each of the modules $S_{\mathcal{C}}$ is contracted into a single vertex.

Lemma 4.3.12. *Let G be an interval graph and let M be a possible end of G . Then $G_M = G / \sim_M^G$ is an interval graph. If A is a max clique of G , then $\{[v] \mid v \in A\}$ is a max clique of G_M . All max cliques of G_M are of this form and \prec_M induces a strict linear order on G_M 's max cliques.*

Proof. Let \mathcal{I} be a minimal interval representation of G for which $\triangleleft_{\mathcal{I}}$ extends \prec_M (c.f. Lemma 4.3.9). For each $[x] \in V / \sim_M^G$ let $I_{[x]} := \left[\min \bigcup_{v \in [x]} I_v, \max \bigcup_{v \in [x]} I_v \right]$. We claim that $\mathcal{I}_M := \{I_{[x]} \mid [x] \in V / \sim_M^G\}$ is an interval representation for G_M . If $[x]$ is a singleton equivalence class, then clearly $I_x = I_{[x]}$. Otherwise, $[x]$ equals some module $S_{\mathcal{C}}$ coming from a maximal set of \prec_M -incomparable max cliques \mathcal{C} . As $\triangleleft_{\mathcal{I}}$ extends \prec_M , any interval I_v with $v \in V \setminus S_{\mathcal{C}}$ intersects $I_{[x]}$ if and only if it intersects some I_u with $u \in S_{\mathcal{C}}$. This corresponds precisely to the neighborhood of $[x]$ in G_M , so \mathcal{I}_M is an interval representation of G_M .

By Lemma 4.1.2, all max cliques of G_M can be obtained from \mathcal{I}_M as sets of intervals containing some $k \in \mathbb{N}$. By construction, there are two kinds of max cliques:

- Max cliques that are already present in G . Let A be a max clique of G which is \prec_M -comparable to all other max cliques. Then $\bigcap_{v \in A} I_v = \{k\}$ for some $k \in \mathbb{N}$, and $\{[v] \mid v \in A\} = \{[v] \mid k \in I_{[v]} \in \mathcal{I}_M\}$ is a max clique of G_M .
- Max cliques that are localized in \mathcal{I}_M in places where modules used to be in \mathcal{I} . If $S_{\mathcal{C}}$ is such a module and $k \in I_{S_{\mathcal{C}}} = \left[\min \bigcup_{v \in S_{\mathcal{C}}} I_v, \max \bigcup_{v \in S_{\mathcal{C}}} I_v \right]$, then $A_k = \{[v] \mid k \in I_{[v]} \in \mathcal{I}_M\}$ is a max clique of G_M . By construction of \mathcal{I}_M , we have $A_k = \{[v] \mid v \in A\}$ for each choice of k above.

Thus, any maximal set of \prec_M -incomparable max cliques in G corresponds to exactly one max clique in G_M . As \prec_M is a strict weak order of G 's max cliques, it induces a linear order on the max cliques of G_M . \square

Remark 4.3.13. The max clique M of G is also a max clique of G_M since it is \prec_M -comparable to all other max cliques of G . Let us write $M' := \{[v] \mid v \in M\}$. It is easy to see that defining $\prec_{M'}$ on G_M gives rise to the same linear order on max cliques as the one induced by \prec_M .

It is immediate from the definition of \sim_M^G that G_M is FP+C-definable.⁵ We obtain the following corollary by combining Lemmas 4.3.11 and 4.3.12.

Corollary 4.3.14. *Let G be an interval graph and M be one of its max cliques. If \prec_M is a strict weak ordering of G 's max cliques, then*

- $G_M = G / \sim_M^G$ is FP+C-definable,
- \prec_M induces a linear ordering on the max cliques of G_M , and
- a strict weak ordering $\prec_M^{G_M}$ of G_M 's vertices is FP+C-definable and has the property that incomparable vertices are connected twins. \square

What is left is to deal with the contents of modules S_C coming from maximal sets of \prec_M -incomparables. Let $P' = \{(M, n) \mid M \in \mathcal{M}, n \in [|V|]\}$. For each $(M, n) \in P'$ define $V_{M,n}$ as the set of vertices of the connected component of $G[\{v \in V \mid \text{span}(v) \leq n\}]$ which intersects M (if non-empty). Let P be the set of those $(M, n) \in P'$ for which

- $M_n := M \cap V_{M,n}$ is a max clique of $G[V_{M,n}]$ and
- defining \prec_{M_n} over $G[V_{M,n}]$ yields a strict weak order of $G[V_{M,n}]$'s max cliques.

It is immediate from Corollary 4.3.7 that for any maximal set of \prec_M -incomparable max cliques \mathcal{C} , $S_{\mathcal{C}} = \bigcup_{C \in \mathcal{C}} V_{C,|C|}$. In this situation, for any $C \in \mathcal{C}$, the set $V_{C,|C|}$ defines a component of $S_{\mathcal{C}}$, and $(C, |C|) \in P$ if and only if $C \cap S_{\mathcal{C}}$ is a possible end of (one of the components of) $G[S_{\mathcal{C}}]$. This gives us enough structure to perform canonization.

Proof of Theorem 4.2. We define the relation $\varepsilon(M, n, x, y)$ inductively, where $(M, n) \in P$ and x, y are number variables. Each $([V_{M,n}], \varepsilon^G[M, n, \cdot, \cdot])$ will be an isomorphic copy of $G[V_{M,n}]$ on the numeric sort. To this end, start defining ε for all $(M, 1) \in P$, then for all $(M, 2) \in P$, and so on up to all $(M, |V|) \in P$.

Suppose we want to define ε for $(M, n) \in P$, then first compute the strict weak order \prec_{M_n} on the interval graph $G[V_{M,n}]$. Consider any nonsingular maximal set of \prec_{M_n} -incomparables \mathcal{C} and let $m := |\mathcal{C}|$. Let H_1, \dots, H_h be a list of the components of $G[S_{\mathcal{C}}]$ and let H_i be such a component. By the above remarks, there exists at least one $C \in \mathcal{C}$ so that $V_{C,m} = H_i$ and $(C, m) \in P$.

Notice that by the definitions of P and \prec_{M_n} , we have $m < n$ and therefore all $\varepsilon^G[C, m, \cdot, \cdot]$ with $C \in \mathcal{C}$ have already been defined. Let \equiv be the equivalence relation on $P \cap (\mathcal{C} \times \{m\})$ defined by $(C, m) \equiv (C', m) \Leftrightarrow V_{C,m} = V_{C',m}$. Using Lemma 2.4.6, we obtain the lexicographic disjoint union $\omega_{\mathcal{C}}(x, y)$ of the lexicographic leaders of \equiv 's equivalence classes.

Finally, recall the equivalence relation $\sim_{M_n}^{G[V_{M,n}]}$ on $V_{M,n}$ which identifies vertices that occur in the same module S_C . Let $\prec_{M_n}^{G[V_{M,n}]}$ be the strict weak order on $V_{M,n} / \sim_{M_n}^{G[V_{M,n}]}$ given by Corollary 4.3.14. Let c_1, \dots, c_k be the list of non-singular equivalence classes of $\sim_{M_n}^{G[V_{M,n}]}$. Each c_i is associated with a unique maximal set of incomparables \mathcal{C}_i , and $c_i = S_{\mathcal{C}_i}$ as sets. We are going to canonize $G[V_{M,n}]$ using $\prec_{M_n}^{G[V_{M,n}]}$, inserting the graph defined by $\omega_{\mathcal{C}_i}(x, y)$ in place of each c_i . Here is how: we may deal with connected twins in $G[V_{M,n}] / \sim_{M_n}^{G[V_{M,n}]}$ in the same way as in Lemma 4.3.11, so let us assume that $G[V_{M,n}] / \sim_{M_n}^{G[V_{M,n}]}$ is free of connected twins and $\prec_{M_n}^{G[V_{M,n}]}$ is a linear order on

⁵As we may not choose representatives for the equivalence classes $[x] \in V / \sim_M^G$, defining G_M in FP+C simply amounts to defining \sim_M^G and treating all vertices in $[x]$ as representatives of the class.

4 Capturing Results on Classes of Graphs

$G[V_{M,n}] / \sim_{M_n}^{G[V_{M,n}]}$. Each $[v] \in V_{M,n} / \sim_{M_n}^{G[V_{M,n}]}$ is represented by the interval $[a+1, a+|[v]|]$, where a is the number of vertices in equivalence classes strictly $<_{M_n}^{G[V_{M,n}]}$ -less than $[v]$. Since all vertices in $[v]$ have the same neighbors in all of $G[V_{M,n}] \setminus [v]$, it is clear how to define the edge relation between $[v]$ and $G[V_{M,n}] \setminus [v]$. If $[v]$ is not a singleton set, then $c_i = [v]$ for some i and the edge relation on $[a+1, a+|[v]|]$ is given by $\omega_{c_i}(x, y)$.

It is clear from the construction that $([|V_{m,n}|], \varepsilon^G[M, n, \cdot, \cdot]) \cong G[V_{M,n}]$. Also, $\varepsilon(M, n, x, y)$ can be defined in FP+C for all $(M, n) \in P$ using a fixed point-operator iterating n from 1 to $|V|$. Finally, let $\varepsilon(x, y)$ be the lexicographic disjoint union of the lexicographic leaders canonizing the components of G , each of which is defined by some $(M, |V|) \in P$. Then $([|V|], \varepsilon^G[\cdot, \cdot]) \cong G$, which concludes the canonization of G . \square

4.4 The quest for a logic capturing LOGSPACE on interval graphs

The methods of Section 4.3 are not restricted to FP+C-canonization of interval graphs but can be adapted to yield a number of related results. The basis for the results in this section and the next is the observation that the inductive ordering procedure (\star) from Section 4.3.1 can be modeled as a reachability query in an undirected graph. Section 4.4.1 applies this insight to show that the logic STC+C captures LOGSPACE on proper interval graphs. Proper interval graphs have an interval representation where no interval completely contains any other. The result may raise hopes to find a logic capturing LOGSPACE on all interval graphs; however, we show in Section 4.4.2 that STC+C is not sufficient for this.

4.4.1 Capturing LOGSPACE on proper interval graphs

Recall from Section 4.3.1 the relation \prec_M defined on the maximal cliques of an interval graph G . As explained there, \prec_M contains precisely the ordering enforced on the max cliques of G by choosing M as the end of an interval representation of G . Not only are we able to define G 's max cliques in FO (Lemma 4.3.1), but \prec_M is even STC-definable by Lemma 4.3.2.

Unfortunately, being able to define \prec_M in STC does not directly give us a capturing result for STC+C on interval graphs, since we used the expressive power of the fixed-point operator in order to deal with the modular decomposition tree in Section 4.3.3. However, if we restrict ourselves to interval graphs in which no such modules occur, defining \prec_M is enough to obtain canonical forms. The class of *proper interval graphs*, which has been studied thoroughly in the literature (see [BLS99]), turns out to possess this property. Even though proper interval graphs are a proper subclass of all interval graphs, they still contain graphs with fairly complicated structure. For example, *paths of cliques* are proper interval graphs.

Definition 4.4.1. A set of intervals (over \mathbb{N}) is called *proper* if no interval is a subset of any other interval. A graph G is called a *proper interval graph* if it has a proper interval representation.

Lemma 4.4.2. *Let G be a connected proper interval graph, let M be a max clique of G , and suppose that \prec_M is a strict weak order. Then \prec_M is a linear ordering of G 's max cliques.*

Proof. Suppose there are max cliques A, B which are incomparable by \prec_M . As G is connected A and B must be connected to the rest of the graph, so w.l.o.g. there is a vertex $v \in A$ which is also contained in some max clique D for which $D \prec_M A, B$. Since A and B are incomparable, $v \in A \cap B$ (see Lemma 4.3.5).

As D is a maximal clique and by invoking Lemma 4.3.5 again, there is $d \in D \setminus (A \cup B) = D \setminus A \neq \emptyset$. Also, there are $a \in A \setminus B$ and $b \in B \setminus A$ as all these are distinct max cliques. Notice that $a, b \notin D$ since otherwise this would imply $A \prec_M B$ or $B \prec_M A$. Now consider any interval

4.4 The quest for a logic capturing LOGSPACE on interval graphs

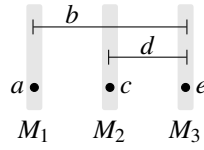


Figure 4.5: A non-proper interval graph linearly ordered by \prec_{M_2} and \prec_{M_3} .

representation \mathcal{I} of G and let $k_A \in \bigcap_{x \in A} I_x$, $k_B \in \bigcap_{x \in B} I_x$, and $k_D \in \bigcap_{x \in D} I_x$. Then $k_A, k_B, k_D \in I_v$, but the intervals I_a, I_b, I_d only contain the points k_A, k_B, k_D , respectively. Therefore at least one of the intervals I_a, I_b, I_d must be completely contained in I_v and G does not have a proper interval representation, contradicting our assumptions and proving the lemma. \square

We remark that not all interval graphs whose max cliques get linearly ordered by \prec_M for some $M \in \mathcal{M}$ are also proper interval graphs. Figure 4.5 shows a simple example of a non-proper interval graph which is linearly ordered by the right choice of an end. Of course, the LOGSPACE capturing result from Theorem 4.3 extends to the class of all interval graphs which become linearly ordered by the choice of one of their max cliques as an end. However, there does not seem to exist an independent characterization of this class nor any mention of it in the literature, so the result is only stated for the restricted but more common case.

Theorem (4.3). *There is an STC+C-definable canonical form of proper interval graphs. Thus, STC+C captures LOGSPACE on the class of proper interval graphs.*

Proof. We first show that proper interval graphs are STC+C-canonizable. Let $G = (V, E)$ be a proper interval graph. The connected components of G are STC+C-definable. We only need to show how to produce a canon for each of G 's connected components. The canon for G is then obtained as the lexicographic disjoint union of the components' canons (cf. Lemma 2.4.6).

So let us assume that G is connected. By Lemma 4.3.2, the relation on max cliques \prec_M is STC-definable for any given max clique M . By Lemma 4.3.11, an ordered copy ε_M of the graph is STC+C-definable from \prec_M in those cases where \prec_M is a linear order on the graph's max cliques. Our desired canonical form is now obtained as the lexicographic leader of all those ε_M for which \prec_M is a linear ordering on the max cliques. By Lemmas 4.3.9 and 4.4.2, this is the case for each possible end of G , thus for at least one max clique M .⁶ Since the lexicographic leader is FO+C-definable from the ε_M by Lemma 2.4.5, this canonical copy of G is STC+C-definable.

As for the capturing result, first observe that undirected reachability queries can be evaluated with only logarithmic space by Reingold's Theorem 2.1.2 [Rei08a]. Therefore, the data complexity of evaluating STC+C-formulas is in LOGSPACE. Conversely, DTC captures LOGSPACE on ordered structures (cf. Theorem 2.3.8). Deterministic transitive closure queries can be easily reduced to symmetric transitive closure queries, whence $\text{DTC} \leq \text{STC}$ (we show this in a different context in Chapter 5, see Corollary 5.2.13). Thus, all LOGSPACE properties of proper interval graphs can be decided in STC on the ordered canonical forms of proper interval graphs, and the capturing result follows. \square

Remark 4.4.3. Notice that the canonical form in the proof above is the same as the one for general interval graphs presented in Section 4.3. In particular, the order computed on the max cliques gives rise to a canonical (minimal) interval representation of the proper interval graph, but not necessarily to a *canonical proper interval representation*. It is not hard to see that such a proper interval representation can be computed in logspace from our interval representation here by

⁶In fact, it is not hard to see that connected proper interval graphs which are not themselves a clique have precisely two possible ends.

4 Capturing Results on Classes of Graphs

perturbing the intervals representing connected twins. By the capturing result, a proper interval representation can also be defined in STC+C.

Arguably, Theorem 4.3 is quite a trivial by-product of the analysis for general interval graphs in Section 4.3, which there led to the PTIME capturing result of Theorem 4.2. Still, it is noteworthy that Theorem 4.3 is one of the very few capturing results of LOGSPACE on non-trivial graph classes, and it appears to be the first for a non-trivial graph class containing all cliques.

Of course, it is harder to obtain capturing results for LOGSPACE since the logics that may be employed have lower expressiveness than the ones used in capturing PTIME. Specifically the logics DTC and STC, though, suffer from an expressiveness mismatch on unordered structures in comparison to LOGSPACE computations. This will emerge in the next section when we try to extend the STC-canonicalization result from proper interval graphs to general interval graphs.

4.4.2 STC+C does not capture LOGSPACE on *all* interval graphs

In [EI00], Etessami and Immerman show the following:

Theorem 4.4.4 (Etessami and Immerman [EI00]). *Tree isomorphism is not expressible in TC+C.*

By encoding trees as interval graphs, we want to show here that any canonization procedure for interval graphs also implies a canonical form of trees, which can then be used to decide tree isomorphism. The goal is the following proposition:

Proposition 4.4.5. *There is a LOGSPACE-definable query on interval graphs which is not TC+C-definable.*

Proof. We aim to apply Theorem 4.2.2 on non-capturing reductions and start by describing a unary DTC-graph interpretation from rooted trees to interval graphs. Let T be a rooted tree in which all edges are directed away from the root. Let T' be the undirected version of T 's transitive closure graph, which means T' has the same vertex set as T and an edge connects vertices u, v whenever there is a path from u to v or from v to u in T . Figure 4.6 illustrates the definition of T' from T .

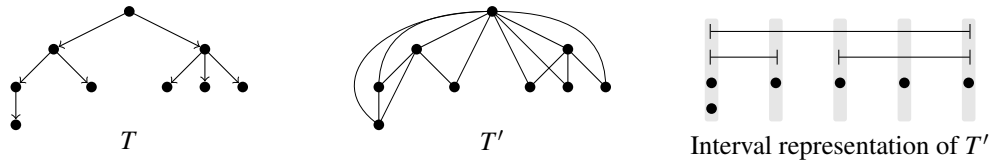


Figure 4.6: Conversion of a tree to an *interval tree*.

T' is really an interval graph: for any node v of T , consider some ordering on v 's children. These orderings induce a total order on the leaves of T by comparing their paths from the root lexicographically. This order has the property that for any node v of T , the descendant leaves of v are an interval within this ordering. It is readily verified that assigning the interval of descendant leaves to every node of T gives an interval representation of T' . Let us call an interval graph that can be constructed from a tree in this way an *interval tree*.

The construction of T' from T can easily be reversed. If all vertices of T' have different closed neighborhoods, then T is simply the graph which has an edge from u to v whenever $N[v] \subset N[u]$ and there is no vertex w with $N[v] \subset N[w] \subset N[u]$. Otherwise, let \sim denote the equivalence relation of vertices in T' which have the same closed neighborhood. Then T' / \sim can be converted as above, and upon replacing each equivalence class $[v]$ with a path of length $|[v]|$, we recover T . It is easy to see that T' can be defined from T in DTC, and that reconstructing T from T' can

be done in logarithmic space. Note that LOGSPACE is closed under composition with functions computable by logspace transducers.

Let us remark that our assumption that T be rooted and directed is inessential. Given an undirected and unrooted tree G_T , we can *pick* a root by existentially quantifying over the vertices. The direction of an edge uv away from the root can be defined in STC by verifying that the root is not reachable from v in G_T after deleting the edge uv .

By a result of Lindell [Lin92], there is a logspace transducer which canonizes trees (also see Theorem 4.5.8), hence tree isomorphism is in LOGSPACE. By Etessami and Immerman's Theorem 4.4.4, this query is not definable in TC+C. We may view tree isomorphism as a query on graphs by considering as instances graphs containing two connected components, each of which is a tree.⁷ The result now follows by applying Theorem 4.2.2. \square

Remark 4.4.6. Interval trees can be characterized independently from the construction given in the above proof. A connected interval graph is an interval tree if and only if for any two neighbors u, v we have $N[u] \subseteq N[v]$ or $N[v] \subseteq N[u]$. It is clear that the interval trees constructed above satisfy this property and the reversion procedure given in the proof of Proposition 4.4.5 produces a suitable tree for any interval graph with this property.

Corollary 4.4.7. *STC+C does not capture LOGSPACE on the class of interval graphs.*

Proof. The LOGSPACE-definable query on interval graphs from Proposition 4.4.5 is not definable in STC+C, either, since $\text{STC+C} \leq \text{TC+C}$ (Lemma 2.3.12). \square

Corollary 4.4.8. *TC+C does not capture NLOGSPACE on the class of interval graphs.* \square

4.5 Canonizing interval graphs in LOGSPACE

In this section, we apply the methods developed for interval graphs in Section 4.3 to the design of a logspace-computable canonical form for interval graphs. Consequently, we place interval graph isomorphism in LOGSPACE, and even a canonical labeling of interval graphs can be computed within logarithmic space. Previous research in this direction has focused primarily on parallelizing the techniques used in the fast linear interval graph algorithms discussed in Section 4.1.4, such as finding perfect elimination orderings and computing PQ-trees. The best previously known upper bound following from these efforts was AC^2 . Since $\text{LOGSPACE} \subseteq \text{NLOGSPACE} \subseteq AC^1$, our result is a significant improvement over this. As we also show that interval graph isomorphism is hard for LOGSPACE, we settle the problem's complexity status. We remark, though, that our procedure makes ample use of undirected graph reachability queries which were not known to be in LOGSPACE at the time when most previous results were obtained.

Unlike the existing linear-time algorithms for interval graphs we reviewed in Section 4.1.4, our presentation aims to optimize for space complexity. Finding logspace algorithms for the graph isomorphism problem of restricted graph classes has received much attention of late. Lindell's canonization algorithm for trees ([Lin92], see Theorem 4.5.8) may count as an early contribution to this area and forms the basis for most recent results. Köbler and Kuhnert have shown how to decide isomorphism of k -trees in LOGSPACE [KK09]. Datta, Limaye, Nimbhorkar, Thierauf and Wagner showed the corresponding result for planar graphs [DLN⁺09]. In both cases, the isomorphism problem also turns out to be LOGSPACE-complete. The graph classes considered in these results have in common that their clique size is bounded by a constant. To the best of our knowledge, the logspace completeness result for interval graph isomorphism presented here is the first for a natural class of graphs containing cliques of arbitrary size.

⁷A more robust formulation of the graph isomorphism problem is given by considering graphs with one vertex color marking one of the two graphs to be compared.

4 Capturing Results on Classes of Graphs

The extension of the FP+C-methods for interval graphs to LOGSPACE was done and published jointly with Johannes Köbler, Sebastian Kuhnert, and Oleg Verbitsky [KKLV10a] (an extended version is available in [KKLV10b]). The development there is self-contained and geared towards the algorithmic aspects of the methods. For example, it uses *interval hypergraphs* as an intermediate device to improve the method's plasticity with regard to data structures. [KKLV10a] also proves LOGSPACE-hardness of the interval graph isomorphism problem, which we show in the upcoming Section 4.5.1.

In our development, we marry the algorithmic techniques from [KKLV10a] with the logical definability framework established here and in [Lau10]. Basing our development on the insights gained in Section 4.3 shortens the space necessary for the presentation of the results and it exonerates us from reproving results obtained earlier in a different framework. It also enables us to frequently argue with *definability in STC+C* instead of *decidability in LOGSPACE*. Definability guarantees invariance under isomorphism of the objects we define and therefore aides in showing their canonicity.

4.5.1 LOGSPACE-hardness of interval graph isomorphism

We start by showing some LOGSPACE-hardness results for interval graph problems. In particular, we will show that interval graph isomorphism is hard for LOGSPACE, thereby proving one half of the completeness result announced by Theorem 4.4.

To prove the hardness results, we reduce from the problem ORD introduced by Etessami in [Ete97] and shown to be LOGSPACE-hard under quantifier-free projections. Quantifier-free projections are a particularly restricted version of FO-reductions (Definition 2.3.13; see [Ete97] or [IL95] for discussions of quantifier-free projections). The problem can be stated as follows:

Definition 4.5.1. Let a directed path P be given and let s, t be vertices. ORD is the problem of deciding whether s comes before t in P .

Theorem 4.5.2 (Etessami [Ete97]). *ORD is LOGSPACE-complete under quantifier-free projections.*

Theorem 4.5.3. *Proper interval graph isomorphism is hard for LOGSPACE under FO-reductions.*

We note that FO-reductions are also DLOGTIME-uniform AC^0 reductions, which are considered by some the most natural kind of reductions for showing LOGSPACE-hardness results (see [BIS90]).

Proof of Theorem 4.5.3. Figure 4.7 illustrates the simple idea of the reduction from ORD. Let an instance of ORD be given with a directed path $P = (V_P, E_P)$ and vertices $s, t \in V_P$. We need to give FO-formulas defining proper interval graphs G_1 and G_2 which are isomorphic if and only if s comes before t in P . We can check directly whether $s = t$ or s, t have distance ≤ 1 , in which case we can define trivial yes- or no-instance (e.g. using cliques and graphs without edges), depending on the finding. So let us assume that s and t have distance at least 2 in P .

$G_1 = (V_1, E_1)$ and G_2 are defined in almost identical ways. Let V_1 consist of pairs of vertices from V_P of which the first component is s or t :

$$\phi_{V_1}(x, y) := (x = s \vee x = t)$$

In Figure 4.7, we have written v instead of (s, v) and v' instead of (t, v) . The figure also shows how to define the edge relation. Let us denote the successor of a vertex u in P by $u + 1$ and its predecessor by $u - 1$. Essentially, (x, u) is adjacent to (x, v) if $u = v + 1$ or $v = u + 1$, except

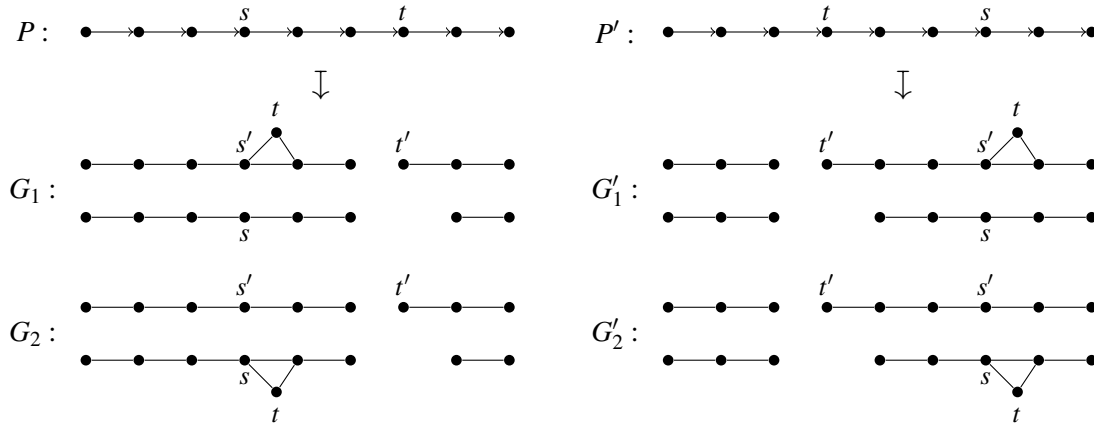


Figure 4.7: Reduction of ORD-instances to proper interval graph isomorphism.

for those cases where u or v is equal to t . (t, t) is only adjacent to $(t, t + 1)$, not to $(t, t - 1)$ (if they exist). And (s, t) is only adjacent to (t, s) and $(t, s + 1)$ in G_1 (respectively, only adjacent to (s, s) and $(s, s + 1)$ in G_2), if it exists. It is clear that this edge relation can be defined by a (quantifier-free) FO-formula using E_P and the constants s and t .

Both G_1 and G_2 are proper interval graphs since paths are proper interval graphs and the vertex (s, t) is represented by an interval overlapping two consecutive intervals. Any isomorphism between G_1 and G_2 has to map the degree-3 vertices to each other. But when s comes after t in P , the connected components containing these degree-3 vertices have different cardinalities, whence G_1 and G_2 are not isomorphic. From Figure 4.7, it should then be clear that G_1 and G_2 are isomorphic if and only if s comes before t in P . The special cases where either s or t appear at one of the ends of P can also be verified.⁸ \square

Corollary 4.5.4. *Interval graph isomorphism is LOGSPACE-hard under FO-reductions.* \square

4.5.2 Constructing a modular tree for interval graphs

We now turn to the task of canonizing interval graphs in logspace. The general strategy for this is simple. The relations \prec_M on the max cliques of an interval graph can be defined through undirected graph reachability queries (cf. Lemma 4.3.2) and are therefore in LOGSPACE by Reingold's Theorem 2.1.2. The main thing left to deal with then is to handle the interval graph's modular decomposition tree as in Section 4.3.3, only that we need to replace the direct aggregation of modules with a more sophisticated approach using Lindell's logspace tree canonization algorithm.

Let $G = (V, E)$ be an interval graph and recall the situation in Section 4.3.3: for each pair $(M, n) \in \mathcal{M} \times [|V|]$, we considered the vertices $V_{M,n}$ in the connected component of $G[\{v \mid \text{span}(v) \leq n\}]$ which contains vertices from M (if any). On the max cliques of each such component $V_{M,n}$, we then defined \prec_M , and whenever \prec_M turned out to be a strict weak ordering, we used it to define an ordered copy of $G[V_{M,n}]$, filling in for each encountered module the lexicographic leader among its ordered representations. Since we were iterating n from 1 to $|V|$, the ordered copies of these smaller modules had already been defined. This procedure cannot be implemented directly in LOGSPACE since it requires to *save* the intermediate canonical forms of all the modules contained in G . Furthermore, this scheme does not necessarily have the shape of a tree since the

⁸Alternatively, we could also check whether s or t appear at the ends of P and treat these special cases separately.

4 Capturing Results on Classes of Graphs

canon of a smaller module S is typically required for inclusion in all ordered copies of a larger module containing S , of which there are one for each possible end of the larger module.

Instead, we need to abstract away from representing sets $V_{M,n}$ together with a distinguished max clique M . Suppose that M is a max clique of G so that \prec_M is a strict weak order on G 's max cliques. Recall the equivalence relation \sim_M^G on the vertices of G , for which $x \sim_M^G y$ whenever $x = y$ or there is a nonsingular maximal set of \prec_M -incomparable max cliques \mathcal{C} so that x and y are contained in the module $S_{\mathcal{C}}$ associated with \mathcal{C} . As shown in Lemma 4.3.12, $G_M = G / \sim_M^G$ is an interval graph and contains precisely one max clique per maximal set of \prec_M -incomparables in G , so that \prec_M induces a total linear order on the max cliques of G_M .

Let Z_M be the max clique which is \prec_M -maximal in G_M . Now we forget about \prec_M and consider \prec_{Z_M} on G_M . We write $L_M := G_M / \sim_{Z_M}^{G_M}$. Lemma 4.3.12 implies again that \prec_{Z_M} induces a linear order on the max cliques of L_M , and by Corollary 4.3.14 we may use \prec_{Z_M} to define a strict weak order \prec^{L_M} on the vertices of L_M . In fact, \prec_{Z_M} determines the order of all vertices of L_M except for connected twins. As we are now working in LOGSPACE, we may break these ties arbitrarily using the underlying graph order, so that \prec^{L_M} is actually a linear order on the vertices of L_M . We will use \prec_{Z_M} to mean both the order on max cliques and the order \prec^{L_M} it induces. The following lemma says that in essence, L_M is the abstraction we are looking for.

Lemma 4.5.5. *Let G be a connected interval graph that does not contain any vertex which is adjacent to all other vertices, and let M_1, \dots, M_k be its possible ends. Then all of the graphs L_{M_k} , $k \in [k]$, are isomorphic and we may partition $[k]$ into at most two sets Q, Q' so that $(L_{M_i}, \prec_{Z_{M_i}})$ and $(L_{M_j}, \prec_{Z_{M_j}})$ are order isomorphic whenever $i, j \in Q$ or $i, j \in Q'$.*

Proof. By our assumptions, G contains more than one max cliques. Let \mathfrak{P} be the set of all maximal proper subsets \mathcal{C} of \mathcal{M} with the property that for any $D \in \mathcal{M} \setminus \mathcal{C}$ we have $D \cap \mathcal{C} = D \cap \mathcal{C}'$ for all $\mathcal{C}, \mathcal{C}' \in \mathfrak{P}$. We must have $|\mathfrak{P}| \geq 3$ since G is connected and no vertex may be included in all max cliques. Furthermore, if $\mathcal{C}, \mathcal{C}' \in \mathfrak{P}$ and $\mathcal{C} \neq \mathcal{C}'$, then $\mathcal{C} \cap \mathcal{C}' = \emptyset$. To see this, suppose that $B \in \mathcal{C} \cap \mathcal{C}'$. Then $D \cap A = D \cap B = D \cap \mathcal{C}$ for all $A, C \in \mathcal{C}$ and $D \notin \mathcal{C} \cup \mathcal{C}'$. So as $|\mathfrak{P}| \geq 3$, $\mathcal{C} \cup \mathcal{C}'$ is a proper subset of \mathcal{M} satisfying the above property, which contradicts the maximality of \mathcal{C} and \mathcal{C}' . We conclude that \mathfrak{P} is a partition of \mathcal{M} .

For each $\mathcal{C} \in \mathfrak{P}$ we define $S_{\mathcal{C}} = \bigcup \mathcal{C} \setminus \bigcup (\mathcal{M} \setminus \mathcal{C})$. The correspondence in names to the modules $S_{\mathcal{C}}$ is intended, of course, and makes sense since the sets $\mathcal{C} \in \mathfrak{P}$ enjoy the same interaction properties with the rest of the graph as maximal sets of \prec_M -incomparable max cliques (compare Lemma 4.3.5). Consequently, let $\sim_{\mathfrak{P}}$ be the equivalence relation on the vertices of G for which $x \sim_{\mathfrak{P}} y$ if and only if $x = y$ or there is a non-singular $\mathcal{C} \in \mathfrak{P}$ so that $x, y \in S_{\mathcal{C}}$. $\sim_{\mathfrak{P}}$ does the same as \sim_M^G , only that it is based on \mathfrak{P} instead of the (finer) partition of max cliques induced by a strict weak ordering \prec_M .

Our goal is to show that each L_M with $M \in \{M_1, \dots, M_k\}$ is isomorphic to $G / \sim_{\mathfrak{P}}$. For this it is enough to show that the concatenation of equivalence relations \sim_M^G with $\sim_{Z_M}^{G_M}$ is equal to $\sim_{\mathfrak{P}}$. Whenever $\mathcal{C} \in \mathfrak{P}$ and $M \notin \mathcal{C}$, Lemma 4.3.4 implies that the max cliques in \mathcal{C} are \prec_M -incomparable. As the sets in \mathfrak{P} were chosen to be maximal, \mathcal{C} is also a maximal set of \prec_M -incomparables. It follows that $\sim_{\mathfrak{P}}$ is equal to \sim_M^G on $\bigcup_{M \notin \mathcal{C} \in \mathfrak{P}} \mathcal{C}$.

When forming $G_M = G / \sim_M^G$, each maximal set of \prec_M -incomparable max cliques \mathcal{C} is replaced by the max clique $M_{\mathcal{C}} = \{[v] \mid v \in \bigcup \mathcal{C}\}$ (by Lemma 4.3.12). Note that this is also true when \mathcal{C} consists of just one max clique. As a result, \mathfrak{P} induces a partition \mathfrak{P}_M of the max cliques of G_M . Also, if \mathcal{C}_M is the cell of \mathfrak{P}_M which contains M , then \mathcal{C}_M is the only cell of \mathfrak{P}_M which is possibly non-singular. As $|\mathfrak{P}_M| \geq 3$, $Z_M \notin \mathcal{C}_M$.

The final step is to show that $\sim_{\mathfrak{P}_M}$ equals $\sim_{Z_M}^{G_M}$ on G_M . If $[v]$ is a vertex of G_M and $[v]$ is a non-singular equivalence class of \sim_M^G , then $[v]$ is only contained in one max clique of G_M . Hence, \mathfrak{P}_M inherits from \mathfrak{P} the property that it partitions the max cliques \mathcal{M}_M of G_M into maximal sets

\mathcal{C} so that for any $D \in \mathcal{M}_M \setminus \mathcal{C}$ we have $D \cap \mathcal{C} = D \cap \mathcal{C}'$ for all $\mathcal{C}, \mathcal{C}' \in \mathcal{C}$. Arguing analogously as above, it follows that $\sim_{\mathfrak{P}_M}$ equals $\sim_{Z_M}^{G_M}$. Therefore, $\sim_{\mathfrak{P}}$ is equal to the concatenation of \sim_M^G with $\sim_{Z_M}^{G_M}$ and L_M is isomorphic to $G / \sim_{\mathfrak{P}}$. This proves the first part of the lemma.

To see the second part, observe that \prec_{Z_M} induces a linear order on L_M 's max cliques. This is true for all $M \in \{M_1, \dots, M_k\}$, so whenever N is a possible end of L_M , then \prec_N linearly orders the max cliques of L_M . Thus, L_M has two possible ends which correspondingly induce two orders on the max cliques and vertices of $G / \sim_{\mathfrak{P}}$. \square

Let us call a vertex of G which is adjacent to all other vertices an *apex* of G . We use Lemma 4.5.5 as follows. If $G[V_{M,n}]$ is connected and does not contain an apex, let N_1, \dots, N_k be its possible ends. Consider the list of ordered graphs $(L_{N_1}, \prec_{Z_{N_1}}), \dots, (L_{N_k}, \prec_{Z_{N_k}})$. We associate with $V_{M,n}$ the lexicographic leader $L_{V_{M,n}}$ in this list. By Lemma 4.5.5, the $(L_{N_i}, \prec_{Z_{N_i}})$ induce at most two orders on the max cliques of $L_{V_{M,n}}$. If these two orders give rise to order isomorphic graphs, then we call them $\prec_{V_{M,n}}$ and $\prec'_{V_{M,n}}$. In this case, $L_{V_{M,n}}$ is symmetric in the sense that its two possible minimal interval representations are mirror symmetric. Otherwise, one of these two orders is excluded by choice of the lexicographic leader. In that case, we just denote the remaining order by $\prec_{V_{M,n}}$.

Next, suppose that $G[V_{M,n}]$ contains an apex. We want to be consistent in our treatment with the case where $G[V_{M,n}]$ has no apex, so we consider all max cliques as one *set of incomparable max cliques*. Figure 4.8 illustrates this. Here is the formal definition. The set $A \subseteq V_{M,n}$ of apices of $G[V_{M,n}]$ is FO-definable and $V_{M,n} \setminus A$ is clearly a module of $G[V_{M,n}]$ (if nonempty). We let \prec be the strict weak order which simply declares all of $G[V_{M,n}]$'s max cliques incomparable. $\sim^{G[V_{M,n}]}$ is the corresponding equivalence relation for which $x \sim^{G[V_{M,n}]} y$ if and only if $x = y$ or $x, y \in V_{M,n} \setminus A$. We set $L_{V_{M,n}} := G[V_{M,n}] / \sim^{G[V_{M,n}]}$ and observe that $L_{V_{M,n}}$ is simply a clique.⁹ Of course, the order \prec induced on $L_{V_{M,n}}$'s max cliques is trivial and we denote it by $\prec_{V_{M,n}}$ in accordance with the naming convention above.

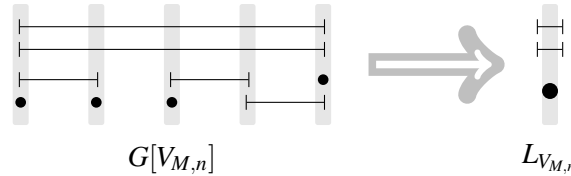


Figure 4.8: Handling modules which contain an apex.

Now recall that $L_{V_{M,n}}$ is actually defined on equivalence classes of $V_{M,n}$ and a similar statement is true of the orders $\prec_{V_{M,n}}$ and $\prec'_{V_{M,n}}$. An equivalence class $[v]$ contains more than one vertex from $V_{M,n}$ only if it represents a module. We call such vertices of $L_{V_{M,n}}$ *slot vertices*. Observe that if $[v]$ is a slot vertex of $L_{V_{M,n}}$, then $N[[v]]$ is a max clique of $L_{V_{M,n}}$ by Lemma 4.3.12.

Remark 4.5.6. The graphs $L_{V_{M,n}}$ resemble the concept of *overlap components* used in [KKLV10a] for the definition of a similar kind of modular tree. Overlap components are connected components of the subgraph of G in which only those edges are present for which the neighborhood of neither endpoint is contained in the neighborhood of the other (intuitively, their intervals *overlap*). The connection is that \prec_M also needs such paths of *properly overlapping* intervals in order to distinguish the graph's max cliques, and $L_{V_{M,n}}$ contains all vertices that can be distinguished by virtue of any of the \prec_M . It can be checked that overlap components and graphs $L_{V_{M,n}}$ only differ in the way they treat vertices that are contained in just one max clique: overlap components treat them as further modules (which they trivially are), the $L_{V_{M,n}}$ graphs directly put them into their unambiguous places.

⁹If $G[V_{M,n}]$ is a clique itself, then $L_{V_{M,n}}$ is identical to $G[V_{M,n}]$, otherwise $L_{V_{M,n}}$ is a clique of $|A| + 1$ vertices.

4 Capturing Results on Classes of Graphs

We are now ready to define the colored *modular tree* $\mathcal{T} = (V_{\mathcal{T}}, E_{\mathcal{T}})$. This time, we really obtain a true modular decomposition tree structure \mathcal{T} which will make its canonization tractable in logarithmic space. The definition of \mathcal{T} is necessarily a bit technical. Most of the concept can be grasped, though, from the illustration in Figure 4.9.

As in Section 4.3.3, let P be the set of those (M, n) for which defining \prec_{M_n} over $G[V_{M,n}]$ is a strict weak ordering (where $M_n = M \cap V_{M,n}$). For uniqueness, we also require that when $(M, n) \in P$, then n is maximal among those n' with the property that $V_{M,n'} = V_{M,n}$. Let $V_{\mathcal{T}}$ be the union of the following sets:

- the set \mathcal{V} of *component vertices* $v_{V_{M,n}}$, one for each set $V_{M,n}$ with $(M, n) \in P$,
- the set \mathcal{A} of *arrangement vertices* $a_{Q, V_{M,n}}$ where $Q \in \left\{ \{ \prec_{V_{M,n}}, \prec'_{V_{M,n}} \}, \{ \prec_{V_{M,n}} \}, \{ \prec'_{V_{M,n}} \} \right\}$ if $L_{V_{M,n}}$ is order isomorphic under the two orderings $\prec_{V_{M,n}}$ and $\prec'_{V_{M,n}}$, and $Q = \{ \prec_{V_{M,n}} \}$ otherwise,
- the set \mathcal{S} of *module vertices* $s_{[v], V_{M,n}}$ for which $[v]$ is a slot vertex in $L_{V_{M,n}}$, and
- $\{s_V\}$, where s_V is a special vertex acting as the root of \mathcal{T} .

We color the vertices in \mathcal{V} by assigning to each $v_{V_{M,n}} \in \mathcal{V}$ the ordered graph $L_{V_{M,n}}$. These colors are ordered by lexicographic comparison of the $L_{V_{M,n}}$. The vertices in \mathcal{A} remain uncolored and may therefore be exchanged by an automorphism of \mathcal{T} whenever their subtrees are isomorphic. Each $s_{[v], V_{M,n}} \in \mathcal{S}$ is colored with the multiset of integers corresponding to the positions that the max clique $N[[v]]$ takes in the orders of $L_{V_{M,n}}$. Observe that different slot vertices are in distinct positions under any single one of the orders on $L_{V_{M,n}}$, but their colors may be equal if they are in symmetric positions with respect to the two orders on $L_{V_{M,n}}$, if defined.

The edge relation $E_{\mathcal{T}}$ of \mathcal{T} is now defined in a straight-forward manner, with all edges directed away from the root s_V .

- s_V is connected to all those $v_{V_{M,n}} \in \mathcal{V}$ for which $V_{M,n}$ is a connected component of G .
- Each $v_{V_{M,n}} \in \mathcal{V}$ is connected to all vertices in \mathcal{A} of the form $a_{Q, V_{M,n}}$.
- Each $a_{Q, V_{M,n}} \in \mathcal{A}$ is connected to all those $s_{[v], V_{M,n}} \in \mathcal{S}$ so that Q is the set of orders of $L_{V_{M,n}}$ under which $[v] \in L_{V_{M,n}}$ attains its minimal position.
- Every $s_{[v], V_{M,n}} \in \mathcal{S}$ is connected to those $v_{V_{M',n'}} \in \mathcal{V}$ for which $V_{M',n'}$ is a connected component of the module $[v]$.

It is readily seen that \mathcal{T} is indeed a tree. The point of the arrangement vertices \mathcal{A} is to ensure that the order of submodules is properly accounted for. If our modular tree did not have such a safeguard, exchanging modules in symmetric positions might give rise to a non-isomorphic graph, but it would not change the tree, so \mathcal{T} would be useless for the task of distinguishing between these two graphs. In fact, we will show in Lemma 4.5.7 below that our modular trees are a complete invariant of interval graphs, so modular trees can really be used to tell whether two interval graphs are isomorphic.

It is straight-forward to verify that \mathcal{T} is STC+C-definable with one caveat. As there is no way in STC+C to pick a canonical representative of elements $(M, n) \in P$ which give rise to the same set $V_{M,n}$, \mathcal{V} has to be defined as a set of equivalence classes of vertices rather than individual vertices. Similar remarks hold true for \mathcal{A} and \mathcal{S} and for the colors of \mathcal{V} . This problem can be remedied in LOGSPACE since we have an ordering on G available and we can arbitrarily

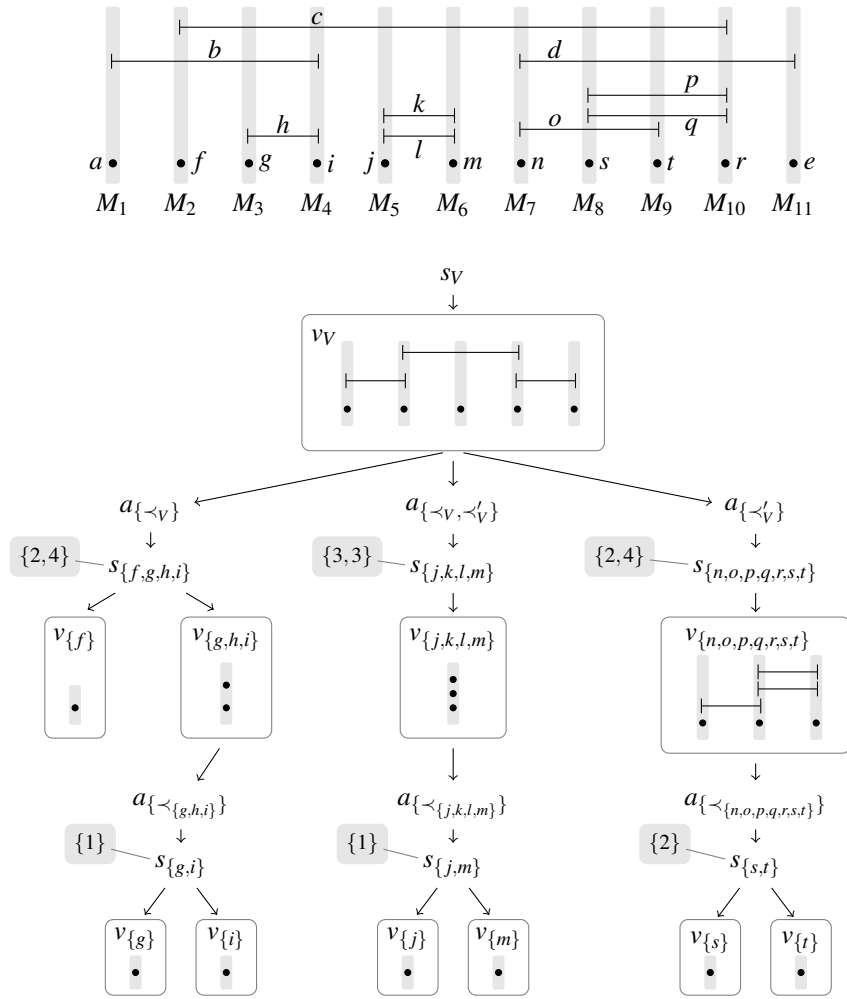


Figure 4.9: An interval graph and its modular tree. Component vertices v_U are represented together with the interval graph L_U labeling them. The colors of module vertices are indicated in the gray fields next to them.

4 Capturing Results on Classes of Graphs

pick one element from each such equivalence class as a representative. In any case, \mathcal{T} can be computed from G by a logspace transducer and therefore be used in any logspace computation by compositionality of logspace machines. Furthermore, STC+C-definability implies invariance under isomorphism, thus if two interval graphs are isomorphic, then their corresponding modular trees are also isomorphic.

Lemma 4.5.7. *Let G and H be interval graphs. If their modular trees are isomorphic, then so are G and H .*

Proof. We show this by induction on the height of the modular tree, by which we mean the number of layers of component vertices. Let \mathcal{T}_G and \mathcal{T}_H be the graphs' modular trees and let ρ be a color-preserving isomorphism between them. If the modular trees are of height 1, then all connected components of G and H become linearly ordered by the choice of an end. Thus if U is a connected component of G , then L_U is isomorphic to $G[U]$, and similarly for H . Since ρ is color-preserving, it induces isomorphisms between the connected components of G and H which it matches up, so $G \cong H$.

Now assuming the claim for up to height x , consider the case $x+1$. The argument for collecting connected components is the same as in the base case, so let us assume that $G = (V_G, E_G)$ and $H = (V_H, E_H)$ are connected. ρ now matches up the vertices of the form $s_{[v],V_G}$ and $s_{[v],V_H}$ in \mathcal{T}_G and \mathcal{T}_H , and we write $\rho(s_{[v],V_G}) = s_{\rho([v]),V_H}$. By our inductive hypothesis the graphs $G[[v]]$ and $H[\rho([v])]$ are isomorphic for all the modules $[v]$ in L_{V_G} . Also, the graphs L_{V_G} and L_{V_H} are order isomorphic.

Since ρ preserves the color of the slot vertices, $s_{[v],V_G}$, we can be sure that the positions of modules in L_{V_G} and L_{V_H} match. To see this we need to consider two cases. If there is only one ordering \prec_{V_G} present on L_{V_G} , then the same is true of L_{V_H} since the numbers of arrangement vertices on this level of \mathcal{T}_G and \mathcal{T}_H must match. In this case each slot vertex $s_{[v],V_G}$ is labeled with just one number which indicates the position of $[v]$ in L_{V_G} with respect to \prec_{V_G} (and the same is true for H , of course). Thus, L_{V_G} and L_{V_H} have isomorphic modules occurring in the exact same positions, so G and H are isomorphic.

The other case is when there are two orderings \prec_{V_G} and \prec'_{V_G} defined on L_{V_G} (and the same is true of L_{V_H}). All slot vertices are labeled with two numbers corresponding to the positions in the two orderings of L_{V_G} (respectively L_{V_H}). To each slot vertex $s_{[v],V_G}$, assign one of its two labels depending on its parent in \mathcal{T}_G :

- the lesser label if its parent is $a_{\prec_{V_G},V_G}$,
- the larger label if its parent is $a_{\prec'_{V_G},V_G}$, and
- the only present label if its parent is $a_{\{\prec_{V_G},\prec'_{V_G}\},V_G}$.

\mathcal{T}_G was constructed so that each slot vertex has the arrangement vertex as its parent which corresponds to the order under which it receives its lesser label. Therefore, G is isomorphic to L_{V_G} with all modules $G[[v]]$ substituted into the \prec_{V_G} -positions assigned to $s_{[v],V_G}$ above. The same is true for the corresponding construction for H with respect to $\rho(a_{\prec_{V_G},V_G})$. By the isomorphism ρ , L_{V_G} and L_{V_H} have isomorphic modules occurring in the exact same positions, and we conclude that G and H are isomorphic. \square

4.5.3 Lindell's (colored) tree isomorphism order

The final step is to canonize \mathcal{T} , or rather to compute a canonical ordering on its vertices. For this, we use a result of Lindell that such a canonical ordering can be computed by a logspace

transducer [Lin92]. In fact, we need an extension of Lindell's algorithm here which can handle colored trees.

Lemma 4.5.8 (Canonical colored tree ordering, based on Lindell [Lin92]). *Given a directed, rooted, and colored tree T , there is a logspace algorithm which computes a canonical ordering of the vertices of T .*

Proof sketch: We shall not repeat the whole of Lindell's proof here but only point out the small modifications necessary to make his proof carry over. Lindell does not directly define an ordering on T 's vertices but he defines a linear order on the isomorphism classes of trees called *tree isomorphism order*, which means that two trees are incomparable if and only if they are isomorphic. It is inductively defined as follows: $S < T$ if

1. $|S| < |T|$, or
2. $|S| = |T|$ and the root of S has less children than the root of T , or
3. $|S| = |T|$, the roots of S and T have the same number of children inducing subtrees (S_1, \dots, S_k) and (T_1, \dots, T_k) , respectively, and the multiset $\{S_1, \dots, S_k\}$ is lexicographically smaller than the multiset $\{T_1, \dots, T_k\}$.

Let us denote the color of a vertex v of T by $c(v)$. In order to handle colors within the tree isomorphism order, we add the following rule to the ones above:

0. $c(s) < c(t)$, where s and t are the roots of S and T , respectively,

and we require $c(s) = c(t)$ for all the above conditions 1 – 3. It is easy to see that this set of conditions is a total order on colored tree isomorphism classes. Let us appropriately call this order the *colored tree isomorphism order*.

Lindell's algorithm now remains virtually unchanged. His procedure compares trees recursively, always comparing 1 and 2 first before calling recursively on the comparison of the subtrees rooted at their children, as required by condition 3. At the point where conditions 1 and 2 are evaluated, we now simply add the evaluation of condition 0. Notice that the order of the vertices' colors must be computable in logspace. Other than that, the algorithm does not change and its space complexity remains in logspace.

Finally, we obtain a canonical ordering of the vertices of T by traversing the tree starting from the root in a depth-first manner, always visiting the children of each node in the order given by the colored tree isomorphism order of the subtrees which these children induce. This traversal procedure is described by Lindell in greater detail. Whenever there are several choices of equal subtrees, these colored trees are isomorphic and the tie may be broken arbitrarily without losing canonicity. So in that case, we use the underlying input order on T to break these ties. The order in which the depth-first traversal visits the tree's vertices gives us the desired canonical ordering. \square

4.5.4 Canonical interval representations and orderings of interval graphs in logspace

We are now ready to prove the main results of this Section. Let $G = (V, E)$ be an interval graph and let $<_{\mathcal{T}}$ be the canonical ordering from Theorem 4.5.8 on G 's modular tree \mathcal{T} . By Lemma 4.5.7, the ordered modular tree $(\mathcal{T}, <_{\mathcal{T}})$ is not only an invariant of isomorphism classes of interval graphs, but it is also *canonical* in the sense that non-isomorphic interval graphs have non-isomorphic ordered modular trees. In short, the idea for canonization is now to choose at each

4 Capturing Results on Classes of Graphs

component vertex v_U the ordering $\prec \in \{\prec_U, \prec'_U\}$ of L_U which is associated with the $\prec_{\mathcal{T}}$ -lesser arrangement vertex $a_{\prec, U}$. This defines canonical orderings on all modules which we just have to *collect* together. We start by restating and proving the theorem from the introduction of this chapter and then use it to derive that canonical orderings are also logspace computable for interval graphs.

Theorem 4.5.9 (Restatement of Theorem 4.4). *Let $G = (V, E)$ be an interval graph. A canonical interval labeling of G can be computed in logspace. That means, there is a logspace-computable function which assigns to each vertex $v \in V$ an interval $I_v \subset \mathbb{N}$ with the following properties:*

- $\{I_v \mid v \in V\}$ is a minimal interval representation of G with interval I_v corresponding to v .
- If G and H are isomorphic interval graphs, then their corresponding sets of intervals are equal. Thus, any pairing of equal intervals from G and H induces an isomorphism from G to H .

Proof. The key to the canonical labeling is the reconstruction of a canonical ordering \prec^* of the maximal cliques of $G = (V, E)$ from $(\mathcal{T}, \prec_{\mathcal{T}})$. Essentially, we pick one of the (at most) two orderings of L_U for each component vertex v_U . Recall that all these orderings were based on max clique orders \prec_M , so for each of them there are minimal interval representations of L_U which define the same order on the max cliques of L_U . As in Section 4.3.3, we can piece these orders together arbitrarily since the order of max cliques in modules is independent of the order of the remaining max cliques in the graph. All we need to do is to prove that \prec^* is definable.

Let M, N be two max cliques of G and let v_U be the unique component vertex so that $M \cap U \neq \emptyset$, $N \cap U \neq \emptyset$, and v_U is farthest away from the root of \mathcal{T} . Consider the label $L_U = (V_U, E_U)$ and recall that V_U consists of equivalence classes of the vertices in $G[U]$. By Lemma 4.3.12, $M' := \{[v] \in V_U \mid v \in M\}$ and $N' := \{[v] \in V_U \mid v \in N\}$ are max cliques of L_U . Let $\prec \in \{\prec_U, \prec'_U\}$ be the ordering defined on L_U for which the arrangement vertex $a_{\prec, U}$ is $\prec_{\mathcal{T}}$ -lesser than the one corresponding to the other order, if defined. If $M' \neq N'$, then we let $M \prec^* N$ if $M' \prec N'$. If $M' = N'$, then M and N are located in the same module S in $G[U]$, but they are contained in different connected components of $G[S]$. Thus, there is a unique module vertex $s_{S, U}$ with $M \cap U' \neq \emptyset \neq N \cap U'$ and unique children $v_{U^M} \neq v_{U^N}$ of $s_{S, U}$ such that $M \cap U^M \neq \emptyset$ and $N \cap U^N \neq \emptyset$. We let $M \prec^* N$ if $v_{U^M} \prec_{\mathcal{T}} v_{U^N}$. From this, it is easy to see that \prec^* is DTC-definable from G, \mathcal{T} , and $\prec_{\mathcal{T}}$.

Now it is readily verified that \prec^* is a linear order on the max cliques \mathcal{M} of G and there is a minimal interval representation \mathcal{I} of G so that $\prec_{\mathcal{I}}$ is the same as \prec^* . Using \prec^* , we identify \mathcal{M} with the initial segment $[[\mathcal{M}]]$ of \mathbb{N} . Finally, we assign to each vertex v of G the interval $I_v = \{M \in \mathcal{M} \mid v \in M\}$, giving us a minimal interval representation \mathcal{I} of G .¹⁰ \mathcal{I} was defined canonically, thus proving the theorem. \square

Using Theorem 4.4, we can now easily define a canonical ordering on the vertices of an interval graph. We remark that this is better even than the FP+C-definable canonical form we obtained in Section 4.3, which forgets about the relationship between the vertices of the input graph and the vertices of the defined canon. However, since we are unable to define even a canonical ordering on cliques in FP+C, we could not have hoped for better. With the auxiliary order available in LOGSPACE, though, we could keep track of the ties between the input graph and its modular tree \mathcal{T} , enabling the following result.

Corollary 4.5.10. *There exists a logspace-computable canonical ordering of interval graphs.*

Proof. Let $G = (V, E)$ be an interval graph and let I_v be the (canonical) interval label for $v \in V$ which logspace computable by Theorem 4.4. We let $u \prec^G v$ if I_u is lexicographically smaller than

¹⁰The name for this interval representation is appropriate since it is essentially the same as \mathcal{I} mentioned before.

I_v as sets of integers. This defines a strict weak order on V in which two vertices are incomparable if and only if their intervals are equal, so they are connected twins. We break ties between connected twins arbitrarily to turn $<^G$ into a linear order on V . This order has the property that interval graphs G and H are isomorphic if and only if they are equal under the orderings $<^G$ and $<^H$, so it is canonical. \square

We finish this section noting that interval graph isomorphism is LOGSPACE-decidable, which is remarkable in itself.

Corollary 4.5.11. *Interval graph isomorphism is decidable in LOGSPACE.* \square

4.6 Discussion

In this chapter, we have presented results about the expressive power of FP+C on various graph classes. In particular, we proved that interval graphs admit FP+C-definable canonical forms. Hence FP+C captures PTIME on the class of interval graphs. The result is interesting as it is one of the first capturing results for a graph class which is not definable by a list of forbidden minors. As such, the techniques we established are different from those used for minor-free graph classes in that they closely relate to modular decompositions of graphs.

We then applied our methods to two similar problems. Firstly, we showed that STC+C captures LOGSPACE on the class of proper interval graphs. But while such a capturing result does not hold on the class of all interval graphs, we could prove, secondly, that interval graphs can be canonized in LOGSPACE. This improves on the previously known AC^2 -upper bound and together with the corresponding hardness result, settles the complexity of the interval graph isomorphism problem.

Additionally, we exhibited limits to our approach by showing that FP+C does not capture PTIME on a variety of graph classes. Among them are bipartite graphs, split graphs, as well as chordal graphs and co-comparability graphs, which are natural superclasses of interval graphs.

There are various ways in which the work of this chapter could be extended. First of all, there is still a large array of superclasses of interval graphs that may be amenable to FP+C-canonization. This might be possible, for example, for circular-arc graphs, which are the intersection graphs of arcs on a circle. Despite the close resemblance between interval and circular-arc graphs, and despite the existence of efficient algorithms for circular-arc graph isomorphism, FP+C-reducing circular-arc graph canonization to interval graph canonization has eluded the attempts of the author so far.

Another interesting superclass of interval graphs are unit-box or unit-disk graphs. Unlike boxicity-2 graphs, on which capturing PTIME is just as hard as on the class of all graphs, these graph classes are not known to be graph isomorphism complete. However, their recognition is NP-complete [BK98], so that any FP+C canonization procedure would probably have to avoid producing a certificate of whether the considered graph is actually part of the targeted graph class. This might be hard to get around: for example, the interval graph canonization procedure presented here can be used to decide whether a graph is really an interval graph (see [Lau10, Corollary I.3]).

We remark that chordal graphs, even though they do not admit FP+C-canonization themselves, often seem to be tractable in fixed-point logic as soon as additional properties are satisfied, such as being a line graph [Gro10a] or a co-comparability graph. It would be instructive to unify these properties. In this context, we also point to Grohe's conjecture [Gro10a] that FP+C captures PTIME on the class of claw-free chordal graphs.

We hope that the methods exhibited in this chapter can be a foundation for further work into the expressiveness of logics on restricted graph classes. In particular, modular decomposition of graphs appears to be a promising tool for the canonization of graphs. It may be very beneficial to

4 *Capturing Results on Classes of Graphs*

develop more generic methods on the basis of modular decomposition as a large group of further graph classes might become accessible to capturing results in this way.

5 Stronger Logics for Polynomial Time

This chapter is devoted to the most direct approach to finding a logic for PTIME: creating one. This is arguably not an easy task, so it is sensible to base our quest on known frameworks that are in some sense *close* to PTIME. Given that FP captures PTIME on ordered structures, fixed-points appear to be an inherent ingredient of polynomial time computations. This begs the question of what has to be added to FP in order to achieve sufficient expressiveness without outgrowing polynomial time complexity. As discussed in Chapter 1, it is natural to add counting operators to FP to remedy its inability to define queries such as the parity of the structure's size. The resulting counting logic FP+C was intensively studied, among others leading to the wealth of capturing results on restricted graph classes mentioned in Chapter 4. In 1992, however, Cai, Fürer, and Immerman's separation of FP+C from PTIME dashed all hopes that FP+C might be expressive enough to capture PTIME [CFI92]. There has not been a comparable effort into investigating candidate logics for PTIME since.

It subsequently became clear to researchers that it was well worth taking a closer look at the nature of the query that dethroned FP+C. The CFI query's main feature is an ingenious encoding of an amorphous parity property into graphs which seemed to require fixing an order of the graph for it to be read out. Given that FP+C had been designed to express properties such as parity, the CFI query demonstrated that there was more to counting when leaving the simple number domain and going to general graphs. This idea could be generalized. Hella, for example, showcased the power of the idea when he used a similar amorphous parity technique to separate arbitrary Lindström quantifiers of different arities [Hel96]. Torán [Tor04] generalized the CFI construction from parities in GF_2 to residue classes in arbitrary finite fields of prime characteristic. Finally, Atserias, Bulatov, and Dawar pinned down the evasive nature of the CFI query when they showed that FP+C is unable to define the solvability of certain basic systems of linear equations, and that the CFI query is essentially an encoding of such a linear system in the language of graphs [ABD07].

At this point, the natural idea is to add to FP+C the ability to deal with linear algebra in order to close the gap exposed by the CFI query. This is what we are doing here in introducing *logics with rank operators*. Even though it sounds simple to remedy FP+C's shortcoming in linear algebra by *adding linear algebra to it*, this matter has much more distinctly chiseled faultlines than FP's innumeracy. For starters, FP+C is well capable of doing all kinds of linear algebra over the field of rationals \mathbb{Q} .¹ We discuss this in Section 5.3 where we also show that FP+C can calculate the determinants of matrices not only over \mathbb{Q} , but over all finite fields of prime characteristic. It is actually a baffling experience to try and separate two non-isomorphic CFI-graphs with GF_2 -determinants of matrices defined from these graphs, since all apparent differences in such matrices cancel out when calculating the determinant.

In this situation, the distinctive qualities FP+C is lacking seem to be the solvability of linear systems and computing the rank of matrices over finite fields. We have chosen to augment FP+C with *rank operators over finite fields* for several reasons. For one, we can define solvability of linear systems with rank operators (Section 5.2.3), but in the converse direction we only know

¹As in the world of computation, working over the reals \mathbb{R} or complex numbers \mathbb{C} is not so much a matter of suitable methods, but one of suitable representations of these numbers. Computations over the reals may often be described more adequately by computational models which work directly over \mathbb{R} . There is also a branch of descriptive complexity theory concerned with these models of computation (cf. [GM95]).

how to use linear systems to check whether an *unordered* matrix has full rank.² Furthermore, though, rank operators also allow for a conceptually smooth extension of counting logics. FO+C already comes equipped with a number sort which we can use to represent matrices' ranks. Also, rank operators can themselves be used to count the solutions of formulas (see Section 5.2.1), so we may consider rank computations as a more sophisticated version of counting quantifiers. There are several further nice properties of including rank computations into our logical frameworks, which will be described in the following. We will pick up the discussion of operators for linear systems at the end of the chapter.

This chapter explores the properties of rank logics from two different angles. On the one hand, we are obviously interested in how good a candidate these logics are for capturing PTIME. We cannot resolve the question of capturing positively or negatively, but we establish rank logics as a possible candidate for PTIME by showing that they satisfy all necessary properties known to date. We do not have any compelling reason to believe that rank logics really do capture polynomial time, but this means that separating rank logics from PTIME will require new techniques that are yet to be discovered. On the other hand, we want to know how expressive these logics are with respect to sample problems and existing complexity classes. Extending FO+C with rank operators, we arrive at capturing results on ordered structures for complexity classes between LOGSPACE and PTIME, like $\oplus L$ and the logspace counting hierarchy. Research into these complexity classes mainly dates to the same decade as the CFI query and characterizes the complexity of various problems in linear algebra. Apart from relating these two strands of research, we also provide a descriptive framework whose formal rigor provides a suitable alternative for classifying the complexity of problems in linear algebra.

We start by defining rank logics in Section 5.1 and present a variety of basic queries which they can express in Section 5.2. In Section 5.3 we explore which queries from linear algebra can already be expressed in FP+C, and in how far these queries can also be computed only with rank operators instead of the fixed-point operator. Section 5.4 will then justify the purpose of rank operators by showing that the CFI query *is* definable using rank operators. Afterwards, we prove the capturing results involving rank logics on ordered structures in Section 5.5. Finally, we show that the expressiveness of rank operators strictly increases when we allow more variables to be used to define the matrices they work over. In this way, Section 5.6 exhibits a major difference between rank and counting operators and strongly suggests further investigation of their expressive power. On this note, we also point to the discussion in Section 5.7 of intriguing problems which are left open in our exposition.

Many of the results in this chapter have been published jointly with Anuj Dawar, Martin Grohe, and Bjarki Holm in [DGHL09]. Some results have also been obtained independently by Bjarki Holm and are documented in his forthcoming Thesis [Hol10]. We will make the appropriate attributions when discussing the individual results.

5.1 Definition of Rank Logics

Let us begin by explaining how matrices can be defined in our logics. The framework for this are the two-sorted counting logics such as FO+C and FP+C (recall Section 2.3.3). Consider a numeric term η and universe variables \vec{x} and \vec{y} , which are possibly free in η . Given a structure \mathcal{A} , define $m_{\vec{a}\vec{b}} := \eta^{\mathcal{A}}[\vec{a}, \vec{b}]$ for tuples \vec{a}, \vec{b} from $U(\mathcal{A})$ interpreting \vec{x} and \vec{y} respectively. In addition to numeric terms, we also allow formulas to define matrices. If φ is a formula with universe variables \vec{x}, \vec{y}

²The situation is different on ordered structures where it can be seen that FO with linear systems operators captures #LH just like FO+rk.

possibly occurring free in φ , then define for $\vec{a} \in U(\mathcal{A})^{|\vec{x}|}$, $\vec{b} \in U(\mathcal{A})^{|\vec{y}|}$

$$m_{\vec{a}\vec{b}} := \begin{cases} 1 & \text{if } \mathcal{A} \models \varphi[\frac{\vec{a}\vec{b}}{\vec{x}\vec{y}}] \\ 0 & \text{if } \mathcal{A} \not\models \varphi[\frac{\vec{a}\vec{b}}{\vec{x}\vec{y}}]. \end{cases}$$

Now let $M := (m_{\vec{a}\vec{b}})$ be the integer matrix whose rows are indexed by $|\vec{x}|$ -tuples and whose columns are indexed by $|\vec{y}|$ -tuples ranging over all of $U(\mathcal{A})$. The size of M is $|\mathcal{A}|^{|\vec{x}|} \times |\mathcal{A}|^{|\vec{y}|}$, so it depends on the size of the universe of \mathcal{A} .

For prime p let M_p denote the matrix of the residue classes of M 's matrix entries modulo p . We view M_p as a matrix over GF_p . Figure 5.1 illustrates the definition of a matrix from a term. Notice that our construction of M_p does *not* provide an ordering of M_p 's rows and columns, but M_p 's rank is still well-defined since it does *not* depend on such an ordering. In general, we also allow matrices to be indexed by number variables, but they need to be bounded again so that we obtain finite matrices.

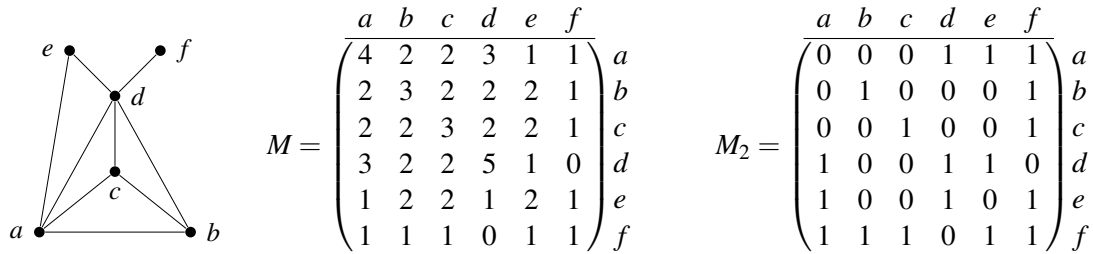


Figure 5.1: Sample definition of a GF_2 -matrix in a graph $G = (V, E)$ given by $\eta(x, y) = \#z Exz \wedge Eyz$.

Definition 5.1.1 (Rank operators). Given a τ -structure \mathcal{A} and a formula or numeric term η , let \vec{x}_1, \vec{y}_1 be all distinct element variables, \vec{x}_2, \vec{y}_2 be all distinct number variables and t be a numeric term bounding all of the variables \vec{x}_2 and \vec{y}_2 (none of which may occur free in t). Let ϖ be a numeric term. Then

$$\text{rk}(\varpi; x_1 x_2 \leq t, y_1 y_2 \leq t) \eta$$

is a numeric term denoting the rank of

$$M = \left(\eta^{\mathcal{A}} \left[\frac{\vec{a}_1 \vec{a}_2}{\vec{x}_1 \vec{x}_2}, \frac{\vec{b}_1 \vec{b}_2}{\vec{y}_1 \vec{y}_2} \right] \text{ mod } \varpi^{\mathcal{A}} \right)_{\substack{\vec{a}_1 \in U(\mathcal{A})^{|\vec{x}_1|}, \vec{b}_1 \in U(\mathcal{A})^{|\vec{y}_1|}, \\ \vec{a}_2 \in [0, t^{\mathcal{A}}]^{|\vec{x}_2|}, \vec{b}_2 \in [0, t^{\mathcal{A}}]^{|\vec{y}_2|}}}$$

over $\text{GF}_{\varpi^{\mathcal{A}}}$ if $\varpi^{\mathcal{A}}$ is prime, and 0 otherwise. Its free variables are $\text{free}(\varpi) \cup \text{free}(t) \cup (\text{free}(\eta) \setminus \{\vec{x}_1, \vec{x}_2, \vec{y}_1, \vec{y}_2\})$. The integer $|\vec{x}_1| + |\vec{x}_2| + |\vec{y}_1| + |\vec{y}_2|$ is called the *arity* of the rank operator $\text{rk}(\varpi; \vec{x}_1 \vec{x}_2 \leq t, \vec{y}_1 \vec{y}_2 \leq t) \eta$. If $p \in \mathbb{N}$ is prime and $\varpi = \underbrace{1 + \dots + 1}_{p \text{ times}}$, then we also write $\text{rk}_p(\vec{x}, \vec{y}) \eta$

instead of $\text{rk}(\varpi; \vec{x}, \vec{y}) \eta$. As in this case, we sometimes omit the emphasis of number variables and their bounding term when it is not needed.

Example 5.1.2. If \vec{x}, \vec{y} are all distinct variables with $|\vec{x}| = |\vec{y}|$, then the formula $\text{rk}_2(\vec{x}, \vec{y}) \eta = \text{rk}_2(\vec{x}, \vec{y}) \vec{x} = \vec{y}$ holds in a structure \mathcal{A} if and only if the matrix defined by $\eta^{\mathcal{A}}$ with respect to row indices \vec{x} and column indices \vec{y} has *full rank* over GF_2 .

5 Stronger Logics for Polynomial Time

Similarly, the formula $\forall \vec{w} \leq (rk_2(x, y)_{x=y}) (rk(\vec{w}; \vec{x}, \vec{y}) \eta = rk(\vec{w}; \vec{x}, \vec{y}) \vec{x} = \vec{y})$ holds in a structure \mathcal{A} if and only if $\eta^{\mathcal{A}}$ defines a matrix of full rank over GF_p with respect to indices \vec{x} and \vec{y} for all primes $p \leq |\mathcal{A}|$.

Remark 5.1.3. Out of notational convenience, we have allowed only one numeric term as a bound for all of a rank operator's numeric variables. This uniform bound is not a restriction on the operator's expressiveness: by conjoining the condition $x \leq t_x$ to the formula η in the scope of the operator we may give each number variable x its own individual bound. If η is a term, the same can be done using multiplication by a term which is 1 when $x \leq t_x$ and 0 when $x > t_x$; the easy construction of such terms is given at the end of this section. In this way we create zero rows or columns in those places where x exceeds its supposed bound so that these rows or columns do not contribute to the rank of the matrix – just as if they did not exist.

It might seem odd at first that rank operators are supposed to work over both formulas and numeric terms. There are good justifications for both, though. Numeric terms give us an easy way to define matrices containing general integers instead of just 0, 1-matrices. It is an open problem whether rank operators over 0, 1-matrices are as expressive as rank operators over general matrices. Except over GF_2 , this does not seem to be the case using any trivial sort of reduction. Of course, it is possible to come up with formalizations of general matrices using just formulas, and we will do that when considering *rank quantifiers* over infinitary logics in Section 5.6, where we do not have a counting sort available. Until then, using numeric terms makes for the cleanest and most intuitive definition of matrices.

Applying rank operators to formulas, on the other hand, is necessary to create a bridge between a structure's universe and the adjoined number sort. In the logics FO+C and FP+C counting terms act as such a bridge. It will be shown that counting can equally be done with rank operators (Proposition 5.2.2), so adding counting quantifiers would be an unnecessary complication.³ In any case, we will frequently employ the adjacency matrices of graphs as the basis for rank operations, where the immediate use of the respective formulas is the most intuitive way to get the desired results.

Recall that the extension of a logic by an operator corresponds to adding the operator's term formation rule to the logic's own term formation rules.

Definition 5.1.4 (Rank logics). Let p be prime.

- $\text{FO}+\text{rk}_p$ denotes the extension of FO^+ with the rank operator rk_p ,
- $\text{FP}+\text{rk}_p$ denotes the extension of $\text{FO}+\text{rk}_p$ with the fixed-point operator ifp ,
- $\text{FO}+\text{rk}$ denotes the extension of FO^+ with the general rank operator, and
- $\text{FP}+\text{rk}$ denotes the extension of $\text{FO}+\text{rk}$ with the fixed-point operator ifp .

Trivially, we have

Observation 5.1.5. $\text{FO}+\text{rk}_p \leq \text{FO}+\text{rk} \leq \text{FP}+\text{rk}$ and $\text{FO}+\text{rk}_p \leq \text{FP}+\text{rk}_p \leq \text{FP}+\text{rk}$ for every prime p . \square

Remark 5.1.6. Given the right representation, one can also define rank operators over arbitrary finite fields GF_{p^k} (p prime, $k \in \mathbb{N}$). Bjarki Holm discusses suitable representations in his thesis [Hol10] and shows that the solvability of linear systems over GF_{p^k} can already be defined in $\text{FP}+\text{rk}_p$.

³All the same, we *will* add counting terms on top of rank logics in Section 5.5.2 because we will need to distinguish more thoroughly between cases that call for rank operations and those for which counting suffices.

Note that all of the rank logics allow to *nest* rank operators, so the ranks of matrices defined by one formula may appear as entries in another matrix. It may not seem immediately clear what this is good for, but aside from some rather technical applications requiring this, Section 5.5.2 will give a formidable justification of this feature by relating it to complexity theory. For that result we will need to measure how deeply rank operators have been nested in a specific formula or term.

Definition 5.1.7 (Nesting depth of rank operators). Let η be an FO+rk formula or term.⁴ The *nesting depth (of rank operators)* $nd(\eta)$ is defined inductively as follows:

- atomic formulas and terms have nesting depth 0,
- if $\eta = \text{rk}(\overline{\omega}; \vec{x}, \vec{y})\psi$, then $nd(\eta) = \max\{nd(\overline{\omega}), nd(\psi)\} + 1$,
- if $\eta = \varphi \star \psi$ with $\star \in \{\vee, \wedge, +, \cdot, =, \leq\}$, then $nd(\eta) = \max\{nd(\varphi), nd(\psi)\}$, and
- if $\eta = \exists x \varphi$, $\eta = \forall x \varphi$, or $\eta = \neg \varphi$, then $nd(\eta) = nd(\varphi)$.

Let us justify the definition of rank logics in two ways. Firstly, *projecting from \mathbb{Z} into finite fields* GF_p is a sensible model for working within the finite fields. This is because for prime p , GF_p is isomorphic to $\mathbb{Z}/p\mathbb{Z}$, where $p\mathbb{Z}$ is the kernel of the ring homomorphism $x \mapsto x \bmod p$ (e.g. see [Lan93, II.§1]). In particular, this means that addition and multiplication of the equivalence classes in $\mathbb{Z}/p\mathbb{Z}$ is well-defined and the isomorphic correspondence gives us $[x] +_{\text{GF}_p} [y] = [x +_{\mathbb{Z}} y]$ and $[x] \cdot_{\text{GF}_p} [y] = [x \cdot_{\mathbb{Z}} y]$. Thus, we may *work over* GF_p by treating elements from GF_p as integers, performing the calculations over \mathbb{Z} instead of GF_p , and then projecting the result back into GF_p . Therefore, it also makes sense for our rank operators to have *the projection into GF_p built in*. Similarly, of course, we may work over the polynomial ring $\mathbb{Z}[x]$ instead of $\text{GF}_p[x]$.

As a second justification, let us show inductively that the data complexity of rank logics is in PTIME. To begin, since we allow bounded quantification over number variables of the form $\exists \mu \leq t \varphi$, we need to prove a polynomial bound on the size of terms for such queries to be solvable in polynomial time.

Lemma 5.1.8. *Let τ be a vocabulary and let t be a numeric τ -term of FP+rk with free number variables μ_1, \dots, μ_k . There is a polynomial $p_t(x, x_1, \dots, x_k)$ so that $t^{\mathcal{A}} \leq p_t(|\mathcal{A}|, \mu_1^{\mathcal{A}}, \dots, \mu_k^{\mathcal{A}})$ in any τ -structure \mathcal{A} . In particular, if t has no free variables, then $t^{\mathcal{A}} \leq p_t(|\mathcal{A}|)$ for all τ -structures \mathcal{A} .*

Proof. This is easy to see using induction over terms. In FO^+ , numeric terms can only be constructed using the constants 0, 1, number variables, and functions \cdot and $+$. The polynomials bounding constants and number variables are obvious. Suppose s and t are terms with free variables $\vec{\mu}$ and $\vec{\nu}$, which are bounded by polynomials p_s and p_t . Then $s + t$ is bounded by $p_s + p_t$ and $s \cdot t$ is bounded by $p_s \cdot p_t$, which are both polynomials. Finally, let us turn to the extension by rank operators. If s is of the form $\text{rk}(\overline{\omega}; \vec{x}_1 \vec{x}_2 \leq t, \vec{y}_1 \vec{y}_2 \leq t) \eta$ and t is bounded by p_t , then s is bounded by the polynomial $x^{\max\{|\vec{x}_1|, |\vec{y}_1|\}} (1 + p_t)^{\max\{|\vec{x}_2|, |\vec{y}_2|\}}$ since the rank of any matrix is bounded by its dimension, regardless of the characteristic of the field determined by $\overline{\omega}$. \square

Now that we have a polynomial bound on the size of terms and given that FP+C-sentences can be evaluated in PTIME, the only thing left to show is that terms of the form $\text{rk}(\overline{\omega}; \vec{x}, \vec{y})\eta$ can actually be evaluated in polynomial time. Given the polynomial bound, checking whether $\overline{\omega}^{\mathcal{A}}$ is prime can simply be done by verifying that there is no integer in $[2, \overline{\omega}^{\mathcal{A}} - 1]$ dividing $\overline{\omega}^{\mathcal{A}}$. The matrix defined by η is of polynomial size and can therefore be written down on the machine's tape completely. Computing the rank of this matrix over $\text{GF}_{\overline{\omega}^{\mathcal{A}}}$ can be done in polynomial time

⁴It should be clear how to define the nesting depth of rank operators for FP+rk. It will not be needed.

5 Stronger Logics for Polynomial Time

using Gaussian elimination (see Section 5.3.1 or any linear algebra textbook). We have shown the following lemma.

Lemma 5.1.9. *Let τ be a vocabulary and let $\eta(\vec{x})$ be a τ -formula or τ -term of FP+rk. Then there is a polynomial-time Turing machine which, given the encoding of a τ -structure \mathcal{A} and an assignment \vec{a} to the free variables \vec{x} , computes $\eta[\vec{a}]$. In particular, if η is a sentence, then deciding the class of structures it defines is in PTIME. \square*

Observe that the Gaussian elimination algorithm we use for calculating the rank of matrices over finite fields in PTIME crucially depends on the use of an auxiliary order. When we consider unordered structures, then Section 5.3 explains how an alternative approach can be used to define matrix rank over \mathbb{Q} in FP+C. Over finite fields, however, it follows from the work of Atserias, Bulatov, and Dawar [ABD07] that the rank of matrices is not definable in FP+C. Consequently, $\text{FO+C} \not\leq \text{FO+rk}_p$ and $\text{FP+C} \not\leq \text{FP+rk}_p$ for all primes p . We will not retrace the approach of Atserias et al.; instead, we obtain these separation results as a corollary to the arity hierarchy theorem of rank logics in Section 5.6 (see Corollary 5.6.2). Over GF_2 , these separations already follow from the expression of the CFI query in Section 5.4.

Before moving on to the basic things that can be expressed in rank logics, we make one convention about the use of terms and formulas. If φ is any formula and x, y are not among its free variables, then

$$\text{rk}_p(x, y)\varphi = \begin{cases} 1 & \text{if } \mathcal{A} \models \varphi, \\ 0 & \text{else} \end{cases}$$

for any prime p . Thus, we may also consider formulas as terms taking values 1 and 0, depending on whether they hold or not. Using this convention, we will freely use formulas in the construction of terms as well.

Remark 5.1.10. The same can be achieved with the counting term $\#z \leq 0 \varphi$ where z is not free in φ .

5.2 Basic properties of rank operators

In this section we present a variety of simple techniques for rank operators that pave the way for the more intricate results in later sections. We also show a number of consequences of these techniques where they have direct implications.

At many points, we will consider a problem with respect to each of the fields GF_p . In all cases where we show that such a problem is definable in FO+rk_p for each prime p , the defining formula is actually the same for all primes and only depends on p as a variable. Thus, we may reuse this formula in FO+rk in a context where p is quantified over. We make the following convention in order to underscore the uniformity of such definitions.

Definition 5.2.1. Suppose that $\eta(p; \vec{z})$ is an FO+rk -formula such that for any fixed prime $p = 1 + \dots + 1$, $\eta(p; \vec{z})$ is a formula of FO+rk_p . Then we say that the property defined by η is *uniformly definable in FO+rk_p* by η . We use the same terminology for numeric terms η .

We start by showing that we may use rank operators for *counting*, for deciding *matrix similarity*, and for checking whether a *linear system of equations is solvable*. The first result is simple and suggests that rank operators may be considered as a more powerful sort of counting operator. Instead of the number of a formula's solutions they count the dimensions of a *definable* vector space. The second result highlights the expressive power of rank operators. The third result makes rank operators particularly useful to us, since many of our applications are more easily cast as a linear system than a direct rank computation. Then we argue that rank logics are closed under

logical reductions, which is convenient for showing many of our results and shows that rank logics make for a robust notion of a logic.

After introducing these basic techniques, we apply them to express *undirected graph reachability* with rank operators, thereby showing $\text{DTC+C} \leq \text{STC+C} \leq \text{FO+rk}_p$ for all primes p . Actually, the last inequality here is strict, which will be proven in Section 5.6 (Corollary 5.6.3).

We then move on to another useful technique and show that rk_p -operators can be used to count paths modulo p in directed graphs when p is prime. This will directly be used to prove that the multiplication of two GF_p -matrices is definable in FO+rk_p and, in fact, we can define the multiplication of polynomially many GF_p -matrices in FO+rk_p . Finally, we show a further application of path counting and use it to express *directed graph reachability* in FO+rk . This result is also a good example for how we may use the multitude of primes available in FO+rk , which will play an even bigger role in expressing determinants (Section 5.3) and capturing the logspace counting hierarchy (Section 5.5.2).

5.2.1 Counting

As the following proposition shows, we can easily use rank operators to count the number of satisfying assignments to a formula.

Proposition 5.2.2 (Counting with rank operators). *Let φ be a formula in which y does not occur free and let p be any prime. Then $\text{rk}_p(x, y)(x = y \wedge \varphi)$ yields the number of $a \in U(\mathcal{A})$ for which $\mathcal{A} \models \varphi[\frac{a}{x}]$.*

Proof. $(x = y \wedge \varphi)$ defines the diagonal matrix $M = (m_{ab})$ which has zeros everywhere except for those m_{aa} for which φ holds when x is interpreted as a . The set of columns containing a 1 is linearly independent over any finite field. \square

Recall that *counting operators* $\#x \varphi$, when interpreted over a structure \mathcal{A} , yield the number of $a \in U(\mathcal{A})$ for which $\mathcal{A} \models \varphi[\frac{a}{x}]$. Proposition 5.2.2 implies that $(\#x \varphi)^{\mathcal{A}} = (\text{rk}_p(x, y)(x = y \wedge \varphi))^{\mathcal{A}}$ whenever p is prime and y does not occur free in φ .

Corollary 5.2.3. $\text{FO+C} \leq \text{FO+rk}_p$ and $\text{FP+C} \leq \text{FP+rk}_p$ for all primes p . \square

5.2.2 Similarity of matrices

Two square matrices $A, B \in \text{GF}_p^{n \times n}$ are called *similar* if there is an invertible matrix $P \in \text{GF}_p^{n \times n}$ so that $A = P^{-1}BP$. The following theorem is ascribed to Dixon in [AW92, Section 5.5] and provides us with a *rank criterion for similarity*.

Theorem 5.2.4 (Dixon). *$A, B \in \text{GF}_p^{n \times n}$ are similar if and only if*

$$\left(\text{rank}(A \otimes I_n - I_n \otimes B^t)\right)^2 = \left(\text{rank}(A \otimes I_n - I_n \otimes A^t)\right) \left(\text{rank}(B \otimes I_n - I_n \otimes B^t)\right). \quad \blacksquare$$

In Theorem 5.2.4, I_n denotes the identity matrix of dimension $n \times n$ and X^t denotes the transpose of the matrix X . $X \otimes Y$ is the *Kronecker product* of matrices $X = (x_{ij})_{i,j \in [n]} \in \text{GF}_p^{n \times n}$ and $Y \in \text{GF}_p^{n \times n}$, which is the $n^2 \times n^2$ matrix

$$X \otimes Y = \begin{pmatrix} x_{11} \cdot Y & x_{12} \cdot Y & \dots & x_{1n} \cdot Y \\ x_{21} \cdot Y & x_{22} \cdot Y & \dots & x_{2n} \cdot Y \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} \cdot Y & x_{n2} \cdot Y & \dots & x_{nm} \cdot Y \end{pmatrix}.$$

5 Stronger Logics for Polynomial Time

This construction is easy to define in FO. For example, if η and ϑ define square matrices A_η and B_ϑ with respect to indices x, y , then $A_\eta \otimes I_n - I_n \otimes B'_\vartheta$ is defined with respect to variables xx', yy' by $\eta(x, y) \cdot (x' = y') - (x = y) \cdot \vartheta(y', x')$. It is not hard to see that $\vartheta(y', x')$ can actually be defined without renaming the variables x, y . This shows the following lemma.

Lemma 5.2.5. *It is uniformly FO+rk_p-definable if matrices defined by numeric terms or formulas η and ϑ are similar over GF_p (p prime). \square*

In the following, we do not actually have any further applications of this lemma. However, it demonstrates the central role that rank operators play for adding *linear algebra* to logics and it obviates the need to consider separate operators for deciding matrix similarity.

5.2.3 Linear systems

Let a τ -structure \mathcal{A} be given along with numeric τ -terms $\psi(\vec{x}, \vec{y})$ and $\beta(\vec{x})$. Considering $\psi^{\mathcal{A}}[\vec{a}, \vec{b}]$ as a matrix A_ψ and $\beta^{\mathcal{A}}[\vec{a}]$ as a vector \mathbf{b}_β over GF_p for some prime p , ψ and β describe the system of linear equations $A_\psi \mathbf{x} = \mathbf{b}_\beta$ on \mathcal{A} .

Such a system is solvable if and only if \mathbf{b}_β is contained in the span of the column vectors of A_ψ , or in other words, if and only if adding \mathbf{b}_β as a new column to A_ψ does not increase the rank of the matrix. Based on this, it is easy to see that the following formula of FO+rk_p defines solvability of the system.

$$\forall z \text{ rk}_p(\vec{x}, \vec{y}y') ((y' \neq z) \cdot \beta + (y' = z) \cdot \psi) \leq \text{rk}_p(\vec{x}, \vec{y}) \psi$$

In the above formula, by our convention, the formulas $y' = z$ and $y' \neq z$ take on truth values from $\{0, 1\} \subset \mathbb{N}_0$. Note that the matrix defined on the left-hand side of the inequality will contain multiple copies of the column vector \mathbf{b}_β , which of course does not alter the rank of that particular matrix. As mentioned in the discussion at the beginning of this chapter the solvability of these linear systems cannot be defined in FP+C in general by a result of Atserias et al. [ABD07]

Proposition 5.2.6. *Let $\psi(\vec{x}, \vec{y})$ and $\beta(\vec{x})$ be formulas or numeric terms and p be prime. There is a uniform formula of FO+rk_p which holds precisely when the system of linear equations $A_\psi \mathbf{x} = \mathbf{b}_\beta$ is solvable over GF_p . The arity of rank operators used for this is $|\vec{x}| + |\vec{y}| + 1$. \square*

The formula defining solvability of the system given by $\psi(\vec{x}, \vec{y})$ and $\beta(\vec{x})$ uses one variable in addition to the variables over which ψ and β are defined, thereby increasing the arity of the rank operator we use. Yet, in Section 5.6 we will need a more parsimonious way to define the solvability of a system which does not increase the arity of the rank operator beyond $|\vec{x}| + |\vec{y}|$. It is unclear whether this is possible in general but we can show an algebraic property which allows us to do this whenever the columns of A_ψ are not all linearly independent.

Lemma 5.2.7. *Let A be a matrix that does not have full column rank. Then the linear system $A\mathbf{x} = \mathbf{b}$ is solvable if and only if for all columns \mathbf{c} of A , adding \mathbf{b} to \mathbf{c} does not increase the rank of A .*

Proof. By a column basis of A we mean a set of column vectors of A that is also a basis for the vector space spanned by A 's column vectors. For the one direction, if $A\mathbf{x} = \mathbf{b}$ is solvable, then \mathbf{b} is in the span of the columns \mathbf{c}_i of A , which means that there are $a_i \in \text{GF}_p$ such that $\sum a_i \mathbf{c}_i = \mathbf{b}$. Fix any column \mathbf{c}_j and denote by A' the matrix where \mathbf{b} has been added to \mathbf{c}_j . First, assume there is a column basis B of A that does not contain \mathbf{c}_j . Then B is also a column basis of A' since $\mathbf{c}_j + \mathbf{b}$ is in the span of B , and hence $\text{rk} A = \text{rk} A'$. Next, assume that all column bases of A contain \mathbf{c}_j and let B be such a column basis. Let B' be obtained from B by exchanging \mathbf{c}_j with $\mathbf{c}_j + \mathbf{b}$ and suppose there is a column \mathbf{c}'_i of A' that is not in the span of B' . Then $B' - \{\mathbf{c}_j + \mathbf{b}\} \cup \{\mathbf{c}'_i\}$ is a linearly independent set of columns from A with the same cardinality as B , hence a column basis

of A not containing \mathbf{c}_j . This contradicts our assumption, and therefore B' spans the column vector space of A' . Since column bases always exist, these two cases are exhaustive and we conclude that $\text{rk}A' \leq \text{rk}A$.

For the converse direction, suppose that $\mathbf{A}\mathbf{x} = \mathbf{b}$ is not solvable, so \mathbf{b} is not in the column span of A . By our assumption, A does not have full column rank. So let B be a column basis of A and let \mathbf{c}_j be a column which is not in B . Then $\mathbf{c}_j + \mathbf{b}$ is not in the span of B since \mathbf{c}_j is, but \mathbf{b} is not, and therefore $\text{rk}A' = \text{rk}A + 1 > \text{rk}A$. \square

In Section 5.6, we will apply Lemma 5.2.7 directly by showing that the linear system's matrix never has full column rank. We will need to show this explicitly since the matrix there has far fewer columns than rows. However, in those cases where the number of columns is not less than the number of rows, we can always apply Lemma 5.2.7, as shown by the following proposition.

Proposition 5.2.8. *Let $\psi(\vec{x}, \vec{y})$ and $\beta(\vec{x})$ be formulas or numeric terms and p be prime. If $|\vec{y}| \geq |\vec{x}|$, then there is a uniform formula of $\text{FO}+\text{rk}_p$ which holds precisely when the system of linear equations $A_\psi \mathbf{x} = \mathbf{b}_\beta$ is solvable over GF_p , and the arity of rank operators used for this is $|\vec{x}| + |\vec{y}|$.* \square

Proof. Structures with just one element are trivial to deal with, so assume that the structure at hand has at least 2 elements. If $|\vec{y}| > |\vec{x}|$, then there are more columns than rows, so the columns cannot be linearly independent. In that case, Lemma 5.2.7 guarantees that the following formula defines the solvability of the system $A_\psi \mathbf{x} = \mathbf{b}_\beta$:

$$\zeta := \forall \vec{z} \text{rk}_p(\vec{x}, \vec{y}) ((\vec{y} = \vec{z}) \cdot \beta + \psi) \leq \text{rk}_p(\vec{x}, \vec{y}) \psi$$

If $|\vec{y}| = |\vec{x}|$, then A_ψ is a square matrix. Either the rank of A_ψ is full, in which case A_ψ is invertible and the system is solvable. Or A_ψ does not have full rank, in which case also the column rank of A_ψ is not full and Lemma 5.2.7 can be applied. Then the formula $\zeta \vee (\text{rk}_p(\vec{x}, \vec{y}) \psi = \text{rk}_p(\vec{x}, \vec{y}) (\vec{x} = \vec{y}))$ defines solvability of the system. \square

5.2.4 Closure under logical reductions

Recall that a logic L is called *closed under logical reductions* if for all finite vocabularies σ, τ , for any syntactic L -interpretation Γ from σ to τ , and any $L[\tau]$ -sentence φ there is an $L[\sigma]$ -sentence $\varphi^{-\Gamma}$ holding in a σ -structure \mathcal{A} precisely when φ holds in $\Gamma[\mathcal{A}]$ (see Definition 2.3.16). The following lemma is implicitly used in most of our definability results.

Lemma 5.2.9. *The logics $\text{FO}+\text{rk}_p$, $\text{FP}+\text{rk}_p$ (p prime), $\text{FO}+\text{rk}$, and $\text{FP}+\text{rk}$ are all closed under logical reductions.*

Proof sketch. Let $\Gamma = (\gamma_V(\vec{x}), \gamma_\approx(\vec{x}, \vec{y}), (\gamma_R)_{R \in \tau})$ be a k -ary $\text{FP}+\text{rk}$ -interpretation from σ to τ . For the other logics mentioned above, the statement is shown in a very similar manner. Let φ be a τ -sentence.

The construction of $\varphi^{-\Gamma}$ is by induction over the formation rules of terms and formulas. For terms t , the inductive assumption is that $(t^{-\Gamma})^{\mathcal{A}} = t^{\Gamma[\mathcal{A}]}$ and for formulas φ it is $\mathcal{A} \models \varphi^{-\Gamma} \Leftrightarrow \Gamma[\mathcal{A}] \models \varphi$. Since our argument is inductive, we implicitly need to deal with the assignments to free variables of numeric terms and formulas. There we also show the inductive invariant that replacing the assignment to free variables \vec{x} with \approx -equivalent elements does not alter the value of $(t^{-\Gamma})^{\mathcal{A}}$ nor the truth value of $(\varphi^{-\Gamma})^{\mathcal{A}}$.

First, element variables y are replaced by a vector \vec{y} of k distinct variables, while number variables μ remain unchanged. We conveniently suppress the mention of number variables in the rest of this proof. We write \vec{x}' for the $k \cdot |\vec{x}|$ -tuple of variables that a tuple \vec{x} is replaced with.

5 Stronger Logics for Polynomial Time

Γ induces a correspondence between the elements of $U(\Gamma[\mathcal{A}])$ and $U(\mathcal{A})^k / \approx$, which we use for the assignment of free variables at induction basis. Equality $x = y$ is replaced with $\gamma_{\approx}(\vec{x}, \vec{y})$. Relations $\varphi = R\vec{x}$ are replaced by $\exists \vec{y}' \gamma_{\approx}(\vec{x}', \vec{y}') \gamma_R(\vec{y}')$. The replacements for \neg , \wedge , \vee and quantification are straight-forward. If $\varphi = (\text{ifp}_{X \leftarrow \vec{x}} \Psi) \vec{x}$, then $\varphi^{-\Gamma} := (\text{ifp}_{X \leftarrow \vec{x}'} \Psi^{-\Gamma}) \vec{x}'$. It is easy to see in all these cases that the inductive assumption carries over.

Since Γ does not change the interpretation of the numeric predicates $+$, \cdot and constants $0, 1$, there is nothing to do here. We are left to consider rank operators. The argument here is particularly elegant, much more so than for counting operators, for instance. If $t = \text{rk}(\varpi; \vec{x}, \vec{y}) \eta(\vec{x}, \vec{y})$, then we let $t^{-\Gamma} = \text{rk}(\varpi^{-\Gamma}; \vec{x}', \vec{y}') \eta^{-\Gamma}(\vec{x}', \vec{y}')$, where $|\vec{x}'| = k \cdot |\vec{x}|$ and $|\vec{y}'| = k \cdot |\vec{y}|$. When $\vec{a} \approx \vec{a}'$ are equivalent row indices, then the rows they index are equal by our inductive assumption, and the same is true for equivalent column indices. It follows that $(t^{-\Gamma})^A = t^{\Gamma[A]}$. \square

Remark 5.2.10. It is not hard to see that the proof of Lemma 5.2.9 works equally well when we allow syntactic interpretations using number variables (together with polynomial term bounds on them – also confer Remark 2.3.14).

5.2.5 Undirected and deterministic graph reachability in FO+rk_p

Recall the *undirected graph reachability* or also *symmetric reachability* problem of determining, given an undirected graph G with distinguished vertices s and t , whether there is a path from s to t in G . We show that undirected reachability can be defined in FO+rk_p for all primes p .

Let $G = (V, E)$ be an undirected graph and let s and t be two vertices in V . For a prime p , let $\mathfrak{S}_{G,s,t}$ be the system of linear equations over GF_p with variables x_v for all $v \in V$ and equations:

- for every edge $e = (u, v) \in E$: $x_u - x_v = 0$,
- $x_s = 1$; $x_t = 0$.

Lemma 5.2.11. *The linear system $\mathfrak{S}_{G,s,t}$ is solvable over GF_p if and only if there is no path between s and t in the graph G .*

Proof. The edge equations of $\mathfrak{S}_{G,s,t}$ force all variables x_u and x_v to take the same value whenever u and v are in the same connected component of G . Since x_s and x_t are set to different values, the system is solvable if and only if s and t are not contained in the same connected component of G . \square

The matrix of the system $\mathfrak{S}_{G,s,t}$ is easily definable in first-order logic by a numeric term $\eta(x_1 x_2, y)$ holding the value 1 at (ss, s) , (tt, t) , and (uv, u) , as well as -1 at (uv, v) for edges $(u, v) \in E$. The value $-1 \in \text{GF}_p$ can be obtained by summing $p - 1$ times the constant 1. Note that every edge equation is stated twice in equivalent ways. The system's solution vector is also easy to define. Thus, we have successfully reduced undirected reachability to the problem of deciding the solvability of a linear system.

Corollary 5.2.12. *Undirected graph reachability is uniformly definable in FO+rk_p for all primes p . Thus, $\text{STC} \leq \text{STC+C} \leq \text{FO+rk}_p$ for all primes p .* \square

The last inclusion in Corollary 5.2.12 will later be shown to be strict in Corollary 5.6.3. The above method for defining reachability fails in general when applied to directed graphs. We can, however, consider an important special case: the class of all directed graphs whose vertices have out-degree at most 1. Since *deterministic graph reachability* easily reduces to undirected graph reachability by Lemma 2.3.7, we obtain another corollary.

Corollary 5.2.13. *Deterministic reachability is uniformly definable in FO+rk_p. Therefore, $\text{DTC} \leq \text{DTC+C} \leq \text{FO+rk}_p$ for all primes p .* \square

5.2.6 Counting paths modulo p in directed acyclic graphs

Rank operators are actually quite a lot more expressive than simply deciding reachability in undirected graphs. For the applications coming up next, such as directed reachability, it is convenient to introduce another tool besides linear systems. We show that we can define terms in $\text{FO}+\text{rk}_p$, which count the number of paths modulo p between two vertices s and t in a directed acyclic graph.

Let $G = (V, E)$ be a directed acyclic graph with n vertices and integer edge weights $w(e)$ for $e \in E$. We assume that edges not present in E are assigned weight 0. Let A denote the matrix of edge weights, i.e., $A(u, v) = w(uv)$. If P is a path in G , then we call the product of all the edge weights of edges in P the *path weight of P* and denote it by $\prod P$. We define paths of length 0 to have weight 1.⁵

We will show that we can define the *sum of all path weights modulo p* of paths between two specified vertices in G for prime p . Notice that if the weights of all present edges are 1, then all paths have weight 1, and hence the sum of path weights of paths between s and t in G is just the number of paths between s and t . Thus, we are doing something more powerful than *just* counting paths, which falls naturally out of the techniques we are using. First, we state and prove a well-known connection between the matrix powers of A and the path weights between vertices. Notice that here A and its powers are considered integer matrices, not matrices over some finite field.

Lemma 5.2.14. *For each $m \in \mathbb{N}$, $A^m(s, t)$ equals the sum of path weights of all paths from s to t of length exactly m .*

Proof. By induction. Let $p_{s,t,m}$ denote the sum of path weights of paths from s to t in G of length exactly m . The statement is clear for $m = 0$ and $m = 1$ (recall that A^0 is the identity matrix). Assume the statement has been shown for m and consider the value $m + 1$. For $s, t \in V$ let N be the set of vertices $u \in V$ so that $ut \in E$. Then

$$p_{s,t,m+1} = \sum_{u \in N} p_{s,u,m} \cdot w(ut) = \sum_{v \in V} A^m(s, v) \cdot A(v, t) = A^{m+1}(s, t)$$

by the inductive assumption. □

If we now consider a directed acyclic graph G in which all edges have weight 1, then Lemma 5.2.14 immediately implies that the powers of G 's adjacency matrix count paths.

Corollary 5.2.15. *Let A be the adjacency matrix of a directed acyclic graph and $m \in \mathbb{N}$. $A^m(s, t)$ equals the number of paths from s to t of length exactly m .* □

Since in a cycle-free directed graph all paths have length at most $n - 1$, Lemma 5.2.14 implies that $A^n = 0$. Let I be the identity matrix of the same size as A and consider the matrix $I - A$. By multiplying out, it is immediately clear that $I - A$ is non-singular and its inverse is explicitly given by $(I - A)^{-1} = I + A + A^2 + \dots + A^{n-1}$. For any $s, t \in V$, the entry $(I - A)^{-1}(s, t)$ therefore equals the total sum of path weights of all paths from s to t in G .

This is true when considering A over the field of rationals, and it is equally true when considering A and all arithmetic involved over GF_p for any prime p , whence we obtain the number of paths modulo p as the entries of $(I - A)^{-1}$.

The adjugate rule (cf. [Jän94] or any other linear algebra textbook) says that for any invertible matrix B , the entries $b_{ij}^{(-1)}$ of its inverse B^{-1} are given by

$$b_{ij}^{(-1)} = (-1)^{i+j} \det B_{-ji} / \det B,$$

⁵Recall that the length of a path is the number of edges it contains (cf. Section 2.1).

5 Stronger Logics for Polynomial Time

where B_{-ji} is B with the j^{th} row and the i^{th} column deleted. We can therefore check if $(I - A)^{-1}(s, t) \neq 0$ by simply testing if $(I - A)_{-ts}$ has full rank. With this, we can now define terms counting the number of paths from s to t in G modulo a prime p .

Path Weight Lemma 5.2.16. Let $\omega(x, y)$ be a term defining a weighted directed acyclic graph G and let p be prime. There is a term $\sigma(p; s, t)$ which is uniformly $\text{FO}+\text{rk}_p$ -definable from ω holding the sum of the path weights of all paths from s to t in G modulo p .

Proof. If $s = t$, then the result is 1. If $(I - A_\omega)_{-ts}$ does not have full rank, then the sum of all path weights is 0 mod p . Otherwise, we know that there is a path from s to t in G and by adding more st -paths, we do not violate acyclicity of G . We use this to find out the precise value of the sum of path weights as follows.

Let $\zeta(s, t, q)$ denote the formula determining if $(I - (A_\omega + q\delta_{st}))_{-ts}$ has full rank over GF_p , where δ_{st} is the matrix which has value 1 in position (s, t) , and 0 everywhere else. That means,

$$\begin{aligned} \zeta(s, t, q) &:= \text{rk}_p(x, y) (x = y \neq s) \\ &= \text{rk}_p(x, y) (x \neq t \wedge y \neq s) \cdot ((x = y) - (\omega(x, y) + q \cdot (x = s \wedge y = t))) \end{aligned}$$

By Lemma 5.2.14, $(I - (A_\omega + q\delta_{st}))^{-1}(s, t) = (I - A_\omega)^{-1}(s, t) + q$ since in the former term we have added weight q to precisely one st -path. Now $\zeta(s, t, q_0)$ is false for precisely one value $q_0 \in [p]$, so the sum of path weights from s to t is $-q_0 \text{ mod } p$. The required term $\sigma(p; s, t)$ is defined as this value using a counting operator. Observe that all involved formulas only depend on the p as a variable, so σ is uniform with respect to p . \square

Corollary 5.2.17. Let G be a directed acyclic graph and let p be prime. The number of paths modulo p from s to t in G is uniformly $\text{FO}+\text{rk}_p$ -definable by a term $\sigma(p; s, t)$. \square

Remark 5.2.18. For Lemma 5.2.16 and Corollary 5.2.17 the only necessary property from linear algebra was to check whether certain matrices have full rank over GF_p . This can also be done with operators determining the determinant modulo p or with operators deciding whether a linear system is solvable modulo p .

5.2.7 Matrix multiplication

We directly apply our ability to count path weights back to defining iterated matrix products. This combinatorial detour might seem odd given that we started with the classical linear algebraic operation of determining a matrix's rank. It appears to be, however, the most direct way to get the following general result.

Proposition 5.2.19 (Iterated matrix product in $\text{FO}+\text{rk}_p$). Let μ be a number variable bounded by some numeric term t and let $\pi(\mu; \vec{x}, \vec{y})$ be a term defining square GF_p -matrices $M_{\pi(\mu)}$ over a structure \mathcal{A} for every $\mu \leq t^A$. The matrix $M_\eta = M_{\pi(0)} \cdot M_{\pi(1)} \cdots M_{\pi(t^A)}$ is uniformly $\text{FO}+\text{rk}_p$ -definable from π by a term $\eta(p; \vec{x}, \vec{y})$.

Proof. We reduce the long matrix product to counting path weights in a graph. For this, we define the weighted directed graph G_π over the vertex set $V_\pi = \mathcal{A}^{|\vec{x}|} \times [0, t^A + 1]$ and edge weights $w(e)$ as follows:

$$w\left(\left(\vec{a}, i\right)\left(\vec{b}, j\right)\right) = \begin{cases} \pi(i; \vec{a}, \vec{b}) & \text{if } j = i + 1 \\ 0 & \text{otherwise.} \end{cases}$$

It is clear that G_π is acyclic. We show by induction on the bound t^A that $M_\eta(\vec{a}, \vec{b})$ is equal to the sum of path weights of all paths from $(\vec{a}, 0)$ to $(\vec{b}, t^A + 1)$. The statement is clear for $t^A = 0$. Now

suppose that the entries (\vec{a}, \vec{b}) of $M^* = M_{\pi(0)} \cdot M_{\pi(1)} \cdots M_{\pi(t^A-1)}$ correspond to the sum of path weights $p_{\vec{a}, \vec{b}}^*$ from $(\vec{a}, 0)$ to (\vec{b}, t^A) . We have

$$(M^* \cdot M_{\pi(t^A)})(\vec{a}, \vec{b}) = \sum_{\vec{c} \in \mathcal{A}^{|\vec{a}|}} M^*(\vec{a}, \vec{c}) \cdot M_{\pi(t^A)}(\vec{c}, \vec{b}) = \sum_{\vec{c} \in \mathcal{A}^{|\vec{a}|}} p_{\vec{a}, \vec{c}}^* \cdot w\left((\vec{c}, t^A)(\vec{b}, t^A + 1)\right),$$

where the right hand side is equal to the sum of path weights from $(\vec{a}, 0)$ to $(\vec{b}, t^A + 1)$ by $w\left((\vec{c}, t^A)(\vec{b}, t^A + 1)\right)$ distributing over the sum of path weights contained in $p_{\vec{a}, \vec{c}}^*$. The term defining M_η over GF_p is now given by the Path Weight Lemma 5.2.16. \square

Remark 5.2.20. In Proposition 5.2.19 we required all of the input matrices to be square and of the same size so that we could be sure that their product is well-defined. This is not really a restriction since matrices of arbitrary size can be embedded into larger matrices by setting superfluous rows and columns to zero.

Of course, we also obtain the ordinary product of two matrices using the same construction.

Corollary 5.2.21. *Let η and ξ be terms defining matrices A_η and A_ξ indexed by (\vec{x}, \vec{y}) and (\vec{y}, \vec{z}) , respectively. For every prime p , the matrix product $A_\eta A_\xi$ over GF_p is uniformly FO+rk_p -definable from η and ξ .* \square

With this technique, we also note that we can define *large sums* modulo prime numbers.

Corollary 5.2.22. *Let $\eta(\vec{x})$ be a numerical term. There is a term ρ which is uniformly FO+rk_p -definable from η holding the value $\sum_{\vec{a} \in \mathcal{A}^{|\vec{x}|}} \eta^A[\vec{a}] \bmod p$ over any structure \mathcal{A} .*

Proof. $\sum_{\vec{a} \in \mathcal{A}^{|\vec{x}|}} \eta^A[\vec{a}]$ is the matrix product between the $1 \times |\mathcal{A}|^{|\vec{x}|}$ -matrix defined by $\eta(\vec{x})$ and the $|\mathcal{A}|^{|\vec{x}|} \times 1$ -matrix defined by the term $\kappa(\vec{x}) = 1$. \square

5.2.8 Directed graph reachability in FO+rk

We now apply the capability of counting paths modulo p to deciding whether a directed graph G contains a path between vertices s and t at all. The crux of the matter is to find a prime p so that the number of paths from s to t is not $0 \bmod p$. The existence of enough primes is guaranteed by the Prime Number Theorem of which there are many different proofs (e.g. Newman gives a short analytic proof in [New80]).

Prime Number Theorem 5.2.23. For $n \in \mathbb{N}$, let $\pi(n)$ be the number of primes less than n . $\pi(n)$ has the asymptotic limit $n/\ln n$, i.e., $\lim_{n \rightarrow \infty} \frac{\pi(n)}{n/\ln n} = 1$. \blacksquare

Let \mathcal{P}_n denote the set of all primes less than n . By the Prime Number Theorem, there is an integer n_0 so that for all $n \geq n_0$ we have $\prod \mathcal{P}_{n^2} > 2^{\pi(n^2)} \geq 2^n$.⁶ Hence

$$\prod \mathcal{P}_{(n_0 n)^2} > 2^{\pi((n_0 n)^2)} \geq 2^{n_0 n} \geq 2^n \quad \text{for all } n \in \mathbb{N}. \quad (\spadesuit)$$

We first consider a *directed acyclic graph* $G = (V, E)$ and denote by $q(s, t)$ the number of paths from s to t in G . Suppose $q(s, t) \neq 0$. As G is acyclic, the number of paths in G is bounded by the number of subsets of V , thus $q(s, t) \leq 2^n$. The inequality (\spadesuit) implies that there must be a prime $p \leq (n_0 n)^2$ which does not divide $q(s, t)$, so $q(s, t) \not\equiv 0 \pmod p$. If $q(s, t) = 0$, however, then $q(s, t) \equiv 0 \pmod p$ for all $p \leq (n_0 n)^2$. Therefore, if we let $\zeta(p, s, t)$ be the FO+rk -term from

⁶This inequality is quite coarse, and it seems plausible that n_0 is actually equal to 1. Yet a proof of this seems to require methods which come close to proving the entire Prime Number Theorem.

Corollary 5.2.17 which counts the number of paths modulo p from s to t whenever p is prime, then

$$\psi(s,t) = \exists p \leq (n_0 \cdot |G|)^2 \text{ (“}p \text{ is prime” } \wedge \zeta(p,s,t) \neq 0)$$

holds in G if and only if there is a path from s to t in G .⁷ Now suppose that $G = (V, E)$ is any directed graph. We define a directed acyclic graph $G_s = (V_s, E_s)$ as a *stratified version of G* which retains the connectivity properties of G . We set $V_s = V \times [n]$ and let $(u, m_1)(v, m_2) \in E_s$ if and only if $uv \in E$ and $m_1 < m_2$. G_s is acyclic since the number in the second component increases whenever we go along an edge of G_s . If there is a path from s to t in G , then there is such a (simple) path of length at most $n - 1$. It follows that t is reachable from s in G if and only if there is a path from $(s, 1)$ to (t, n) in G_s . We may modify $\psi(s, t)$ to work over G_s accordingly and obtain a formula $\psi'(s, t)$ which defines directed reachability in directed graphs (formally speaking, ψ' is obtained from ψ as the pull-back under the FO+C-interpretation mapping G to G_s , see Lemma 2.3.17 and Remark 2.3.18).

Proposition 5.2.24. *There is a formula $\eta(s, t)$ of FO+rk which holds for a directed graph G if and only if there is a path from s to t in G . Hence $\text{TC} \leq \text{FO+rk}$. \square*

5.3 Linear algebra in FP+C and FO+rk

In this section we discuss the computation of various classical objects and operations in linear algebra, such as determinants, matrix inverses, characteristic polynomials, and ranks. Along the way, we will give justification for why we have been focusing on computing the *rank over finite fields* so far: determinants in general and ranks of matrices over the rational numbers \mathbb{Q} may already be computed in FP+C.

The key to our results here is the implementation of Le Verrier’s method for computing the characteristic polynomial $\chi_M(x)$ of a matrix M in FP+C (Section 5.3.3). It was first observed by Holm in [Hol07] that this is possible (see also [DGHL09]). We show that, in addition, Le Verrier’s method can be cast in FO+rk and may also be used to obtain the coefficients of $\chi_M(x)$ over GF_p for arbitrary prime p . Since the polynomial $\chi_M(x)$ contains $\det(M)$ as its constant coefficient, we directly obtain the determinant in this way.

Theorem 5.3.1. *Let M be a definable square integer matrix. The binary representations of the coefficients of the characteristic polynomial $\chi_M(x) \in \mathbb{Q}[x]$ are definable both in FP+C and in FO+rk.*

The input matrix M does not necessarily have to be defined by a simple term, but its entries may be given in *binary representation* for Theorem 5.3.1 to hold. Effectively representing and working with large numbers such as the determinant is actually not simple in FO where we have no sort of recursion mechanism available. Many problems can be avoided in FO+C by passing to the so-called *Chinese remainder representation (CRR)* of large numbers. By a result of [HAB02], CRRs can efficiently be converted into binary representations, thus enabling Theorem 5.3.1. In addition, we will see that CRRs are actually a very natural tool for working with large numbers in FO+rk. CRRs will be introduced thoroughly in Section 5.3.2 since they will be used again in Section 5.5.2 to show that FO+rk captures the logspace counting hierarchy on ordered structures.

Due to the use of CRRs, we require rank operators of many different characteristics. This is necessary as Le Verrier’s method does not generally work over finite fields (it does work, though, to project the result from working over \mathbb{Q} back into GF_p). Thus, we are in the interesting situation of not knowing any way how to define the GF_p -determinant of a matrix M in FO+rk_p alone.⁸

⁷“ p is prime” corresponds to the FO+C-formula $\neg \exists x, y \leq p \cdot x \cdot y = p$.

⁸Over GF_2 , of course, we easily get $\det(M)$ in FO+rk_2 since $\det(M) = 1$ if and only if M has full rank.

In fact, there are many other algorithms around for computing the determinant which we briefly review in Section 5.3.1. We will see there that all of these methods make inherent use of an auxiliary linear order of the matrix's rows and columns, with Le Verrier's method being the only exception. It therefore remains open whether GF_p -determinants can be defined in FO+rk_p for prime $p > 2$.

Finally, we will show in Section 5.3.4 a variety of consequences of Theorem 5.3.1.

Theorem 5.3.2. *Let M be a definable integer matrix. The rank of M over \mathbb{Q} is definable both in FP+C and in FO+rk.*

Theorem 5.3.3. *Let \mathbb{F} be the field \mathbb{Q} or any finite field GF_p with p prime. Let M be a definable square integer matrix. $\det(M) \in \mathbb{F}$ is definable both in FP+C and in FO+rk. If M is invertible over \mathbb{F} , then M^{-1} can also be defined in either FP+C or FO+rk.*

In the statement of Theorem 5.3.3 we have to be careful because the inverse of an integer matrix has entries in \mathbb{Q} but is not necessarily an integer matrix. For our purposes here, we may simply represent the numerators and denominators of M^{-1} 's entries separately. The last two theorems together explain why we chose to consider operators for the rank over finite fields rather than any other linear algebraic operator to extend our logics with. Given that FP+C cannot define the rank of matrices over finite fields, this operation is arguably among the few basic operations in linear algebra that may not be fully specified through inductive definitions alone.

5.3.1 Methods for calculating determinants and characteristic polynomials

Most linear algebra textbooks define the determinant of a square matrix $A = (a_{ij})_{i,j \in [k]}$ through the Leibniz formula

$$\det(A) = \sum_{\sigma \in \mathcal{S}_k} \text{sgn}(\sigma) \prod_{i \in [k]} a_{i\sigma(i)},$$

where \mathcal{S}_k is the set of all permutations of $[k]$ and $\text{sgn}(\sigma) \in \{-1, 1\}$ denotes the *sign* of σ (see, for example, [Jän94]). Of course, the Leibniz formula is never really used for calculating $\det(A)$ since its direct implementation would require summing over $k!$ terms. Most textbooks then go on to note that $\det(A)$ can be calculated using *Gaussian elimination* of the following kind: let r_i denote the i^{th} row $a_{i1} \dots a_{ik}$ of A . We may either exchange rows r_i and r_j . Or we may perform row reductions, i.e. replace row r_j by $r_j - x \cdot r_i$, where x is any element of the field we are working in. Exchanging rows actually changes the sign of the determinant but row reductions leave the determinant unchanged. Using these two operations, we can reduce A to an upper triangular matrix, whose determinant is simply the product of all the elements on the diagonal. A 's determinant is then easily obtained by keeping track of the number of row exchanges we used.

This type of Gaussian elimination can obviously be implemented in cubic time, and it may also be used to infer the rank of A by counting the number of zero rows obtained in this way. However, the algorithm clearly uses an ordering on the rows and columns of the matrix which it uses to decide the order of rows to pivot (after all, the term "upper triangular matrix" only makes sense in the presence of an order). This makes it impossible to implement Gaussian elimination in FP+C over unordered structures.

Luckily, many other methods have been developed for as central an object as the determinant. Most of these methods rather target the characteristic polynomial $\chi_A(x)$ of A since it may be used to compute the eigenvalues of A . Of course, none of these methods were developed with the specific goal in mind to avoid using the ordering of rows and columns of A , which comes natural when writing A down on paper. Let us therefore briefly review these methods' merits for being cast in a logical framework.

Krylov's method. Let A be a square matrix of dimension $n \times n$. A vector v is called *cyclic for A* if the vectors $A^0v, Av, A^2v, \dots, A^{n-1}v$ are linearly independent. Clearly, $A^n v$ can always be written as a linear combination of these vectors since $\chi_A(x)$ has degree n and $\chi_A(A) = 0$ by the Cayley-Hamilton Theorem [Lan93, Theorem XIV.3.1].

Krylov's method (cf. [FF63]) relies on the fact that for *most* matrices there are cyclic vectors. Suppose v is a cyclic vector of A and write $b_i = A^i v$ for $i \in [0, n]$. Krylov's method now consists of solving the linear system $b_{n-1}c_{n-1} + \dots + b_0c_0 = -b_n$ with the indeterminates c_0, \dots, c_{n-1} . We have $\chi_A(x) = x^n + c_{n-1}x^{n-1} + \dots + c_1x + c_0$ since, in this case, the characteristic polynomial is equal to the minimal polynomial of A . Without a method for solving linear systems, this transformation is of course of no advantage for us. Using rank operators and upon supply of a suitable cyclic vector, this system may however be used to determine all coefficients of χ_A by exploiting that it is uniquely determined.

For us, this method has two major drawbacks: unlike a human calculator, we cannot make an educated guess for a cyclic vector of A . Most importantly, though, cyclic vectors often times do not exist, and this is particularly true of logically defined matrices. To see this, note that cyclic vectors exist if and only if $\chi_A(x)$ is equal to A 's minimal polynomial $\mu_A(x)$ (see [NP95]). Let \vec{x}, \vec{y} be element variables and let $\eta(\vec{x}, \vec{y})$ define matrices over sets, i.e., over the empty vocabulary $\tau = \emptyset$. By looking at GF_p -matrix powers of the matrices M_n defined by η for different sizes n of the universe, it can be seen that the different M_n , $n \in N$, only have a finite number of minimal polynomials. Thus, almost all such logically defined matrices M_n do not have a cyclic vector since the degree of μ_{M_n} is strictly less than that of χ_{M_n} . This is a major difference to *random matrices* over GF_p , which have a cyclic vector almost surely ([NP95]). Krylov's method is therefore particularly unsuitable for logically defined matrices.

We remark that another method attributed to *Hessenberg* is also based on the existence of a cyclic vector (see [FF63]). In addition, this method relies on an ordering to determine the characteristic polynomial in an iterative fashion.

Mahajan and Vinay's method. Unlike the other methods presented here, this method is of fairly recent origin and gives an entirely combinatorial characterization of the determinant [MV97]. Mahajan and Vinay generalize the Leibniz formula by considering so-called *clow sequences* instead of permutations of $[k]$. We shall not dwell on the nature of clow sequences here since their final algorithm does not mention these any more. They show that in a specifically constructed weighted graph H_A with special vertices s , t_+ , and t_- , $\det(A)$ corresponds precisely to the difference between the sum of path weights from s to t_+ and the sum of path weights from s to t_- .

This sounds promising given our ability to calculate sums of path weights as in Section 5.2.6. The caveat, however, lies in the construction of H_A which makes use of an order on the domain we are working over. In this respect, their method is close in spirit to Samuelson's and Chistov's methods discussed below, a connection pointed out by Mahajan and Vinay themselves in [MV99]. The ordering enters the definition of the graph H_A in an essential way so that it is unclear whether this method can be cast in a choiceless way.

Csanky's method. In 1976, Csanky used Le Verrier's method to give an NC-algorithm for the inversion of matrices [Csa76]. We will describe Le Verrier's method in detail in Section 5.3.3. It may be used to determine the coefficients of the characteristic polynomial over any commutative ring of characteristic zero. Note that this excludes all finite fields, which is the reason why we cannot cast this method in $\text{FO}+\text{rk}_p$ for fixed prime p .

Finding an appropriate name for this method is not easy. Apparently, it was already known to Sir Isaac Newton (†1727) [Koz92]. Le Verrier used it in 1846 to predict the existence of

planet Neptune based on observed perturbations of Uranus's orbit [Enc10]. The method is also sometimes called the Faddeev-Leverrier method due to a popular generalization of the original approach by Faddeev [FF63]. As we will use neither Faddeev's generalization nor Csanky's analysis of its NC implementation, we will stick with calling it Le Verrier's method from here on.

The methods of Samuelson-Berkowitz and of Chistov. Given that Csanky's method does not work over general fields, it was an open problem for a while whether this can be done in NC. In 1984, Berkowitz gave an affirmative answer by presenting a parallel implementation of Samuelson's method [Sam42] for determining the characteristic polynomial [Ber84]. One year later, Chistov also published a different method for the same task ([Chi85], also see [Koz92]).

Both methods have in common that they relate the characteristic polynomials of a matrix $A = (a_{ij})_{i,j \in [n]}$ to a_{11} , the vectors (a_{12}, \dots, a_{1n}) and (a_{21}, \dots, a_{n1}) , and the submatrix $A_{-11} = (a_{ij})_{i,j \in [2,n]}$. By recursively decomposing A in this way and collecting the relations from all the decomposition steps, they obtain parallelizable methods. Berkowitz's approach effectively yields a reduction to computing the *iterated matrix product* of n matrices. Chistov exhibits relations between the polynomials in question considered as *formal power series*, which may be handled efficiently since we can restrict ourselves to computations modulo x^{n+1} . Both of these methods suffer the drawback that they make essential use of an auxiliary order on the matrix which determines the order of decompositions.

Mulmuley's method. In 1986, Mulmuley gave the first NC-algorithm for calculating the rank of a matrix over any field [Mul86] (also see [Koz92]). Before, it was only known that this is possible over the field of rationals \mathbb{Q} , since there the rank may be inferred from the characteristic polynomial. Why this is so and why this is not true over finite fields will become clear in Section 5.3.4. Mulmuley avoids this problem by essentially multiplying the entries of the i^{th} row of A with the monomial y^{i-1} and computing the characteristic polynomial χ , whose coefficients are now from the polynomial ring $\text{GF}_p[y]$.⁹ He proves that the GF_p -rank of A may be now be inferred from χ .

Again, this algorithm requires an auxiliary order on the matrix in order to determine which power of y each row shall be multiplied with. Recall that the rank of matrices over finite fields cannot be defined in FP+C. Thus, basic recursion and counting alone is insufficient to implement Mulmuley's method in the absence of an auxiliary order, and it is an open problem whether the method can be modified and cast in another choiceless setting such as Choiceless Polynomial Time.

5.3.2 Binary and Chinese remainder representations

The determinant of a matrix A may be exponentially larger than the entries of A . For example, when $E = 2 \cdot I_n$ is the $n \times n$ diagonal matrix with all diagonal entries equal to 2, then $\det(E) = 2^n$. All the terms which we can define in FP+rk, however, are polynomially bounded with respect to the structure's size (Lemma 5.1.8). We call such numbers *small* in order to distinguish them from those numbers whose magnitude is too large to be represented by terms. In order to *calculate* the determinant of a matrix within our logical frameworks, we need to represent this number in a different way than by a plain term.

Let us first describe the classical *binary representation*, which is also the most common way to represent large numbers in Turing machine computations. For $x \in \mathbb{N}$, let $k = \lfloor \log x \rfloor$ and write

⁹For technical reasons, A has to be a symmetric matrix here. The algorithm applies to arbitrary matrices A by considering the symmetric matrix $\begin{pmatrix} 0 & A^t \\ A & 0 \end{pmatrix}$ instead.

5 Stronger Logics for Polynomial Time

$x = \sum_{i \in [0, k]} x_i \cdot 2^i$, where the $x_i \in \{0, 1\}$ are uniquely determined. The digits x_i are called *bits* of x 's binary representation (x_0, \dots, x_k) . We represent the bits of x 's binary representation by a formula $b_x(\mu)$ with a single free number variable μ , where $b_x(i)$ holds if and only if $x_i = 1$.¹⁰ As before, we have to associate a polynomial bound $p_\mu(|\mathcal{A}|)$ with μ depending on the size of the underlying structure \mathcal{A} . Then the term b_x can represent numbers up to $2^{p_\mu(|\mathcal{A}|)+1} - 1$ in binary.

We will also need to represent negative integers. It is not important how exactly we do this since all PTIME-conversions between such representations (cf. [Knu81]) can be performed in FP+C by the Immerman-Vardi Theorem 2.3.10. Let us simply assume that the sign is encoded by some designated bit as part of a number's binary representation ("sign-and-magnitude representation").

In this section, we will actually need to work with matrices all of whose entries are defined in binary representation. We assume these to be given by formulas $\varphi(\vec{x}, \vec{y}; \mu)$ so that for all fixed elements \vec{a}, \vec{b} , $\varphi[\vec{a}, \vec{b}; \cdot]$ is the binary representation of the matrix entry in position (\vec{a}, \vec{b}) . In [BGS02], Blass et al. describe various matrix properties and operations that are definable in FP+C. These constructions can be adapted to work for matrices over \mathbb{Z} in binary representation.

- *Matrix product:* There is a formula of FP+C that defines the matrix product MN for any definable matrices M, N .
- *Matrix powers:* There is a formula of FP+C that defines the matrix M^k for any definable $n \times n$ matrix M and $k \in [n]$.
- *Trace:* There is a formula of FP+C that defines $\text{tr}(M)$ for any definable matrix M .

Treating the field over which to calculate the ranks of matrices as a variable gives us access to an alternative way of handling large numbers. Instead of storing the binary representation of a number x , we store its residues x_p modulo a set of primes $p \in \mathcal{P}$, where $\prod \mathcal{P} > x$. By the Chinese Remainder Theorem [Lan93, Theorem II.2.1], x is the unique integer in $[0, \prod \mathcal{P} - 1]$ with residues x_p . Consequently, we call $(x_p)_{p \in \mathcal{P}}$ the *Chinese Remainder Representation* (CRR) of x with respect to the set of primes \mathcal{P} .

Instead of using formulas to define the CRR of a number x , we use numeric terms since they are much more natural to work with in this context. Let $c_x(\mu)$ be a numeric term with one free number variable μ and let $p_\mu(|\mathcal{A}|)$ be a polynomial bound on μ . $c_x(\mu)$ represents x in Chinese remainder representation if $x \leq \prod_{\text{prime } p \leq p_\mu(|\mathcal{A}|)} p$ and $c_x(p) = x_p = x \bmod p$ for all primes $p \leq p_\mu(|\mathcal{A}|)$. We may require in addition that $c_x(i) = 0$ when i is not prime, yet this is not essential for how we use c_x .

Using the CRR instead of the binary representation has some advantages in itself. For example, addition and multiplication of two numbers x and y given in CRR with respect to the same set \mathcal{P} is simply done by adding or multiplying the residues x_p and y_p componentwise, as long as the result $x + y$ or xy does not exceed $\prod \mathcal{P}$. Another advantage is a very simple representation of negative integers. Observe that our choice of representing the numbers in $[0, \prod \mathcal{P} - 1]$ was somewhat arbitrary, so with the same set of primes we might as well choose to represent the numbers $[o, o + \prod \mathcal{P} - 1]$ for any offset $o \in \mathbb{Z}$. When we cannot be sure that the numbers we are working with are non-negative, we will consider this interval to be centered around 0.

To make CRRs useful for us, we need a variety of results due to Hesse, Allender, and Barrington [HAB02] telling us that we may freely convert between CRRs and binary representations. Recall that FO+M is FO together with majority quantifiers, and that $\text{FO+M} \leq \text{FO+C}$. Let \mathcal{P}_n denote the set of all primes at most n .

¹⁰We may also have chosen a number term to encode the bits of a binary representation, as we will for Chinese remainder representations. Of course, both kinds of representations are easily definable from each other. Since binary representations need to be modified using fixed-points for virtually all algebraic operations on them, using formulas for the encoding is the less cumbersome choice.

Theorem 5.3.4 ([HAB02]). *Let $(x_p)_{p \in \mathcal{P}}$ be the CRR of $x \in [-\prod \mathcal{P}/2, \prod \mathcal{P}/2 - 1]$, defined by a term $c_x(\mu)$. There is a formula $b_x(\mu)$ definable in FO+M from c_x encoding the binary representation of x . ■*

Lemma 5.3.5 ([HAB02]). *Let $b_x(\mu)$ be a formula defining the binary representation of a number x and suppose $\prod \mathcal{P}_n/2 > |x|$. There is a term $c_x(\mu)$ definable in FO+C from b_x encoding the CRR of x with respect to \mathcal{P}_n . ■*

Lemma 5.3.6 ([HAB02]). *Let $c_x(\mu)$ encode the CRR of a number x and let p be a small prime (which is not necessarily used in the representation of x). There is a term t_x which is FO+C-definable from c_x holding the value $x \bmod p$. ■*

We remark that by Equation (♠) from Section 5.2.6, a number whose binary representation requires k bits is uniquely represented by a CRR with respect to the primes in $[k^2]$. This is also true for negative integers given that we use one dedicated bit for the representation of the sign of the binary number. With regard to our polynomial bound on number variables, a number is representable by a binary representation b_x in our logics if and only if it is representable by a CRR c_x .

5.3.3 Implementing Le Verrier's method in FP+C and FO+rk

In this section, we prove Theorem 5.3.1 by showing how Le Verrier's method for computing the characteristic polynomial can be cast both in FP+C and in FO+rk. It has been observed by Rossman that Csanky's algorithm is expressible in Choiceless Polynomial Time with counting $\tilde{\text{CPT}}+\text{C}$ (see Section 3.2.1). Blass and Gurevich [BG05] used this observation to show that $\tilde{\text{CPT}}+\text{C}$ can also express determinants over finite fields. The result that Le Verrier's method can actually be implemented in FP+C was first pointed out by Holm [Hol07] (also see [DGHL09]).

Let M be a square matrix over \mathbb{Q} of dimension $n \times n$ and let $\chi_M(x) = \det(xI - M) = x^n + a_1x^{n-1} + a_2x^{n-2} + \dots + a_n$ denote its characteristic polynomial. Let us define $s_k := \text{tr}(M^k)$ for $k \in [0, n-1]$, i.e. s_k is the sum of the diagonal elements of M^k . Le Verrier's method holds that the coefficients a_1, \dots, a_n are the solution of the following linear system:

$$\begin{pmatrix} n & s_1 & s_2 & \dots & s_{n-2} & s_{n-1} \\ 0 & n-1 & s_1 & \dots & s_{n-3} & s_{n-2} \\ 0 & 0 & n-2 & \dots & s_{n-4} & s_{n-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 2 & s_1 \\ 0 & 0 & 0 & \dots & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} a_n \\ a_{n-1} \\ a_{n-2} \\ \vdots \\ a_2 \\ a_1 \end{pmatrix} = \begin{pmatrix} -s_n \\ -s_{n-1} \\ -s_{n-2} \\ \vdots \\ -s_2 \\ -s_1 \end{pmatrix} \quad (\diamond)$$

Let us justify this by following [Koz92]. Let $\lambda_1, \lambda_2, \dots, \lambda_n$ denote the complex eigenvalues of M , counted with multiplicities. That means

$$\chi_M(x) = x^n + a_1x^{n-1} + a_2x^{n-2} + \dots + a_n = \prod_{i=1}^n (x - \lambda_i).$$

By collecting terms from multiplying out the right hand side, we see that the coefficients a_k of the characteristic polynomial can be written in terms of the eigenvalues as follows:

$$a_k = (-1)^k \sum_{1 \leq i_1 < \dots < i_k \leq n} \prod_{j=1}^k \lambda_{i_j} \quad (1 \leq k \leq n)^{11}$$

¹¹The polynomials on the right hand side are classical algebraic objects called *elementary symmetric polynomials*.

5 Stronger Logics for Polynomial Time

In particular, we have $a_n = (-1)^n \prod_{i=1}^n \lambda_i = (-1)^n \det(M)$ and $a_1 = -\sum_{i=1}^n \lambda_i = -\text{tr}(M)$. From this, we now derive the linear recurrence for the coefficients a_k that corresponds to the system (\diamond) and which no longer makes mention of the eigenvalues.

The matrix M^k has eigenvalues $\lambda_1^k, \dots, \lambda_n^k$ (cf. [Lan93, Theorem XIV.3.10]). Thus

$$s_k = \text{tr}(M^k) = \sum_{i=1}^n \lambda_i^k$$

We define for all $k, m \geq 1$,

$$f_k^m := \sum_{\substack{1 \leq i_1 < \dots < i_k \leq n \\ l \notin \{i_1, \dots, i_k\}}} \left(\prod_{j=1}^k \lambda_{i_j} \right) (\lambda_l)^m.$$

Multiply together a_k and s_m and simplify to obtain:

$$\begin{aligned} a_k s_m &= (-1)^k \left(\sum_{1 \leq i_1 < \dots < i_k \leq n} \prod_{j=1}^k \lambda_{i_j} \right) \left(\sum_{i=1}^n \lambda_i^m \right) \\ &= (-1)^k (f_{k-1}^{m+1} + f_k^m). \end{aligned}$$

Consider the telescoping series

$$\begin{aligned} &a_k s_0 + a_{k-1} s_1 + \dots + a_1 s_{k-1} + s_k \\ &= (-1)^k \left((f_k^0 + f_{k-1}^1) - (f_{k-1}^1 + f_{k-2}^2) + \dots \pm (f_1^{k-1} + f_0^k) \mp f_0^k \right) \\ &= (-1)^k f_k^0 \\ &= (n-k)a_k = (s_0 - k)a_k. \end{aligned}$$

The last equation gives us the desired linear recurrence for a_k in terms of a_1, \dots, a_{k-1} which corresponds to the system (\diamond):

$$ka_k + a_{k-1}s_1 + a_{k-2}s_2 + \dots + a_1s_{k-1} = -s_k.$$

We are now ready to prove Theorem 5.3.1. We restate it here in detail for matrices containing numbers in binary. Since the binary representation of the small number held by a numeric term is definable in FO+C (cf. [Imm99, Theorem 1.17]), this result is also true for the classical matrices we have been using so far.

(Theorem 5.3.1). Let $\eta(\vec{x}, \vec{y}; \mu)$ be a numeric term with $|\vec{x}| = |\vec{y}|$ defining the entries of the matrix $M \in \mathbb{Z}^{|\vec{x}| \times |\vec{y}|}$ in binary representation. The binary representations of the coefficients of the characteristic polynomial $\chi_M(x) \in \mathbb{Z}[x]$ are definable both in FP+C and in FO+rk.

Proof. Let us fix the structure \mathcal{A} we are working over and let $T \in \mathbb{N}$ be the bound on the number variable μ in \mathcal{A} . We first need to define the coefficients $s_k = \text{tr}(M^k)$. In FP+C we can define the binary representation $b_{s_k}(\mu)$ of s_k since the powers M^k can be calculated using the fixed-point operator. If m is a bound on the absolute value of the entries of M , then the largest eigenvalue of M has absolute value at most nm . Therefore, $s_k \leq (nm)^{k+1}$ for any $k \in [n]$ and we may use a bound of $\lceil \log((nm)^{n+1}) \rceil \leq (n+1) \lceil \log(n) + \log(m) \rceil \leq (n+T)^2$ on μ to represent all of the s_k .

In FO+rk we use the CRR $c_{s_k}(\mu)$ of s_k . Since we will need this later, though, we do not employ any primes from the interval $[n]$ for representing s_k . Using the coarse estimate that $\prod \mathcal{P}_n < n^n = 2^{n \log n}$ and Equation (\spadesuit) from Section 5.2.6, it is not hard to see that the primes \mathcal{P} in $[n+1, (2n+T+1)^4]$ are sufficient to represent all the s_k in CRR (for large enough n). By Lemmas 5.3.5 and

5.3.6, we may define the matrices $M \bmod p$ for every $p \in \mathcal{P}$. Now by Proposition 5.2.19, there is a term $\pi(p, k; \vec{x}, \vec{y})$ defining M^k over GF_p for each of our primes p . Using Corollary 5.2.22, we obtain a term $\text{tr}(p, k)$ defining the trace of M^k over GF_p .

It should now be clear that the System (\diamond) is definable both in FP+C and in FO+rk with the entries encoded in binary and in CRR, respectively. Observe that we define this system over the number sort with the rows and columns corresponding to the numbers s_k . Since the matrix and the solution vector are ordered, determining the solution of the system can now be done in FP+C by invoking the Immerman-Vardi Theorem 2.3.10. As the system's matrix is triangular, however, it is also not hard to see how to define a fixed-point iteration which directly computes the entries a_k of the solution vector step by step. Note that we need to divide at each step by the element on the diagonal, which can also be done in FP+C using any naive sort of division algorithm for binary numbers. As we know that the a_k are coefficients of $\chi_M(x) = \det(xI - M)$, they are integers just like the entries of M , so we do not need to worry about representing rational numbers here.

In FO+rk, we could similarly use the fact that the system is ordered and invoke that FO+rk captures the #L hierarchy on ordered structures (this is coming up in Section 5.5.2, Theorem 5.5.11). Solving linear systems is contained in $L^{\#L}$ (for instance, see [ABO99]). In this case, however, let us argue directly how to solve (\diamond) , since the CRR makes this particularly easy. The system's matrix has full rank over GF_p for every prime $p > n$, where $n \in \mathbb{N}$ is the dimension of the matrix. This follows immediately from the observation that p does not divide any of the entries on the diagonal. But then the system is uniquely solvable over GF_p for each of the primes p which we are using for encoding the entries of the system. Notice that the values $1, 2, \dots, n$ on the system's diagonal are precisely the reason why we cannot cast Le Verrier's method in GF_p for $p \leq n$. For each fixed $p \in \mathcal{P}$, we obtain the value for a_n by modifying the system as follows.

$$\begin{pmatrix} 1 & 0 & \dots & 0 & 0 \\ n & s_1 & \dots & s_{n-2} & s_{n-1} \\ 0 & n-1 & \dots & s_{n-3} & s_{n-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 2 & s_1 \\ 0 & 0 & \dots & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} a_n \\ a_{n-1} \\ \vdots \\ a_2 \\ a_1 \end{pmatrix} = \begin{pmatrix} v \\ -s_n \\ -s_{n-1} \\ \vdots \\ -s_2 \\ -s_1 \end{pmatrix}$$

Here v denotes a new and free number variable. This system is solvable over GF_p if and only if $v = a_n \bmod p$, so we obtain $a_n \bmod p$ using a counting quantifier. Computing this quantity for all primes of our CRR representation, we obtain in this way the CRR representation of a_n . In a similar fashion we may also extract the CRRs of all the other coefficients of the characteristic polynomial. Finally, we use Theorem 5.3.4 to convert the CRR of all these coefficients to their binary representations in FO+rk. \square

Remark 5.3.7. The proof of Theorem 5.3.1 may also be used to show that the characteristic polynomial of a matrix M with entries from \mathbb{Q} can be defined in FP+C. The representation of rational numbers requires an additional formula for the binary representation of the fractions' denominators. The same is possible in FO+rk, only that here we first have to factor out the fractional parts of M 's entries since in FO+rk we only know how to define matrix powers for integer matrices.

We have refrained from formulating Theorem 5.3.1 in this generality since it would shift the focus away from the definability of Le Verrier's method and towards the definability of algebraic manipulations of large numbers, which we need for all this to be meaningful. These manipulations are possible in both logics, though, which can be seen by invoking the Immerman-Vardi Theorem 2.3.10 and the result that FO+rk captures the logspace counting hierarchy on ordered structures (Theorem 5.5.11). Similar remarks are true for Theorems 5.3.2 and 5.3.3.

Remark 5.3.8. Our framework for representing finite matrices by definable terms and formulas

allows us to handle matrices in any of the relational vocabularies for prime fields defined by Blass et al. in [BGS02]. Bjarki Holm shows in his thesis [Hol10] that the characteristic polynomial of a matrix over any finite field, not necessarily prime, can be defined in FP+C.

5.3.4 Determinants, ranks, and matrix inverses

As immediate corollaries to Theorem 5.3.1 we obtain one part of Theorem 5.3.3.

Corollary 5.3.9. *Let M be a definable matrix. The binary representation of $\det(M)$ is definable in both FP+C and FO+rk.*

Proof. Let $\chi_M = x^n + a_1x^{n-1} + \dots + a_{n-1}x + a_n$ be M 's characteristic polynomial. As we have seen before, $\det(M) = (-1)^n a_n$. The binary representation of $\det(M)$ is therefore obtained from Theorem 5.3.1 by possibly modifying the sign of a_n . \square

Corollary 5.3.10. *Let M be a definable matrix over GF_p (p prime). Then $\det(M) \in \text{GF}_p$ is definable in both FP+C and FO+rk. The same is true of all coefficients of the characteristic polynomial of M over GF_p .*

Proof. Even though Le Verrier's method cannot generally be carried out in GF_p , we may regard M as an integer matrix M' and consider $\chi_{M'}(x) = \det(xI - M') \in \mathbb{Z}[x]$. By considering the Leibniz formula for computing the determinant, we see that $\chi_{M'}$ is arrived at by only using arithmetic in the polynomial ring $\mathbb{Z}[x]$. It follows that $\chi_M(x) = \chi_{M'}(x) \bmod p$. Thus, we may simply define χ_M 's coefficients by reducing $\chi_{M'}$'s coefficients modulo p , which is possible both in FP+C and in FO+rk by Lemma 5.3.6. \square

Using Theorem 5.3.1, we may also define the inverse of an invertible matrix M . By the Cayley-Hamilton Theorem (e.g., [Lan93, Theorem XIV.3.1]), $\chi_M(M) = 0$. Multiplying by M^{-1} on both sides and rearranging terms, we get

$$M^{-1} = -\frac{1}{a_n} (a_{n-1}I + a_{n-2}M + \dots + a_1M^{n-2} + M^{n-1})$$

In FP+C, the sum on the right is defined using a fixed-point, regardless of whether we are summing over binary representations of numbers or elements from GF_p . In FO+rk we sum up the Chinese remainder representations, which is made possible by Corollary 5.2.22. Now the division by a_n does not necessarily yield integer terms for all entries of M^{-1} when we are working over \mathbb{Q} . For example, the inverse of $\begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$ is $\begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix}$. Thus, when working over \mathbb{Q} , we may only represent the numerators and denominators of M^{-1} 's entries separately. Of course, when we are working over a finite field GF_p , division by $a_n \bmod p$ is definable and we may define M^{-1} by a numeric term. Theorem 5.3.3 now follows.

Finally, let us show that the rank of matrices over \mathbb{Q} can be defined in FP+C and FO+rk (Theorem 5.3.2). Let M be a matrix over \mathbb{Q} , not necessarily square, and let $M^* = M^t M$, where M^t denotes the transpose of M . The following lemmas are proven, for example, in [Koz92].

Lemma 5.3.11. $\text{rank } M = \text{rank } M^* = \text{rank } (M^*)^2$ for any matrix M over \mathbb{R} . \blacksquare

Lemma 5.3.12. *Let M be an $n \times n$ matrix over any field. If $\text{rank } M = \text{rank } M^2$, then $\text{rank } M = n - k$, where x^k is the highest power of x that divides the characteristic polynomial $\chi_M(x)$.* \blacksquare

As M^* is square, the rank of M^* over \mathbb{Q} is now simply the number i so that in $\chi_A(x) = x^n + a_1x^{n-1} + a_2x^{n-2} + \dots + a_n$, $a_i \neq 0$ and $a_j = 0$ for all $j > i$. This proves Theorem 5.3.2.

5.4 Cai-Fürer-Immerman graphs

In 1992, Cai, Fürer and Immerman [CFI92] proved that FP+C does not capture PTIME on the class of all finite structures, thereby settling what had been an important open problem in descriptive complexity theory at the time. For the proof, they constructed a query on a class of graphs that can be defined by a polynomial time computation but not by any sentence of FP+C . In this section we show that this query can be expressed in the logic FO+rk_2 , by considering a first-order definable system of linear equations over GF_2 . This result was published jointly with Anuj Dawar, Martin Grohe, and Bjarki Holm in [DGHL09].

We first introduce the class of Cai-Fürer-Immerman (CFI) graphs on which the separating query is defined. The following presentation of the graphs is adapted from [CFI92] and [DRR08]. Note that unlike the presentation of Dawar, Richerby, and Rossman [DRR08], who show that the CFI query can be expressed in the logic of Choiceless Polynomial Time, we do not require an ordering on the underlying graphs G .

Definition 5.4.1 (Cai-Fürer-Immerman graphs). Let $G = (V, E)$ be a connected undirected graph with minimum degree at least 2. We denote the set of edges incident to $v \in V$ by $E(v)$. Let $T \subseteq V$. For each $v \in T$, let

$$I_v := \{v_Z \mid Z \subseteq E(v), |Z| = 1 \pmod{2}\},$$

and for each $v \in V \setminus T$, let

$$I_v := \{v_Z \mid Z \subseteq E(v), |Z| = 0 \pmod{2}\}.$$

Let $\hat{V} := \bigcup_{v \in V} I_v$, $\hat{E} := \{e_0, e_1 \mid e \in E\}$, $\hat{C} := \{e_c \mid e \in E\}$, and $U^* := \hat{V} \cup \hat{E} \cup \hat{C}$. Define an edge relation E^* on U^* by

$$E^* := \{\{v_Z, e_1\} : e \in Z\} \cup \{\{v_Z, e_0\} : e \in E(v) \setminus Z\} \cup \{\{e_i, e_c\} : e \in E, i \in \{0, 1\}\},$$

and a unary relation $C^* := \hat{C} \subset U^*$. We set $\mathcal{G}^T := (U^*, E^*, C^*)$ and call \mathcal{G}^T the CFI-graph defined from G and T .

We refer to the sets of vertices \hat{C} , \hat{E} and \hat{V} as the *color nodes*, *outer nodes* and *inner nodes* of \mathcal{G}^T , respectively. The color nodes in C^* serve as an identification of vertex pairs e_0, e_1 and make sure that each automorphism of \mathcal{G}^T respects these pairings. We note that instead of singling out the color nodes by actually coloring them, we might also have used graph gadgets, which would have turned \mathcal{G}^T into a proper uncolored graph. The coloring simplifies this task, though.

The *parity* of a CFI graph \mathcal{G}^T is the parity of $|T|$, i.e. $|T| \pmod{2}$. In [CFI92], Cai et al. show the following:

Theorem 5.4.2 (Cai, Fürer, Immerman [CFI92]). *For a connected graph G where every vertex has degree at least two, and for all $T, S \subseteq V(G)$, the graphs \mathcal{G}^T and \mathcal{G}^S are isomorphic if and only if they have the same parity. There is a PTIME algorithm that can distinguish between the odd and even CFI graphs of any graph G .*

Thus, there are exactly two non-isomorphic structures \mathcal{G}^T for G which we aptly call the *even* and the *odd CFI graph*. Furthermore, Cai et al. show that FP+C cannot distinguish between even and odd CFI graphs in general. Thus, FP+C does not capture PTIME on the class of all graphs.

Theorem 5.4.3 (Cai, Fürer, Immerman [CFI92]). *There exists a sequence T_n of regular graphs of degree 3 so that there is no sentence of FP+C which is true on all even and false on all odd CFI graphs constructed from T_n .*

5 Stronger Logics for Polynomial Time

In this section, we show that there is a specific polynomial time algorithm for distinguishing between even and odd CFI graphs which only relies on checking for the solvability of a linear system over GF_2 . In particular, this system is first-order definable from the given CFI graph, so that we obtain the main theorem of this section.

Theorem 5.4.4. *There is a sentence φ_{CFI} of $\text{FO}+\text{rk}_2$ that holds in \mathcal{G}^T if $|T|$ is even but which does not hold if $|T|$ is odd. Therefore, $\text{FO}+\text{rk}_2 \not\equiv \text{FP}+\text{C}$ and $\text{FP}+\text{rk}_2$ is strictly more expressive than $\text{FP}+\text{C}$.*

Remark 5.4.5. It can be shown that the class of all CFI graphs is $\text{FO}+\text{rk}_2$ -definable, so Theorem 5.4.4 implies that the classes of *even* and *odd CFI graphs* are both definable in $\text{FO}+\text{rk}_2$.

Remark 5.4.6. Theorem 5.4.4 is mainly interesting because the CFI graphs have become a yardstick for how expressive a logic must be if it is to be considered a candidate for capturing PTIME . By formulating the query separating even and odd CFI graphs as a linear system over GF_2 , we see that adding suitable operators from linear algebra, like our rank operators, can be considered a minimal uniform way of extending $\text{FP}+\text{C}$ to also cover this query. We note that not only $\text{FP}+\text{rk}_2$ is strictly more expressive than $\text{FP}+\text{C}$, but in fact $\text{FP}+\text{C} \not\equiv \text{FP}+\text{rk}_p$ for all primes p , which we prove in Section 5.6.

Proof of Theorem 5.4.4. Let $G = (V, E)$ be a connected graph where every vertex has degree at least two and let \mathcal{G}^T be a CFI graph constructed from G . Let $\mathfrak{S}_{\mathcal{G}^T}$ be the system of linear equations over GF_2 with variables x_{e_i} for all $e_i \in \hat{E}$ and x_{v_Z} for all $v_Z \in \hat{V}$, and equations:

- $\sum_{v_Z \in \hat{V}} x_{v_Z} = 0$,
- for all $e_i \in \hat{E}$: $x_{e_i} + x_{e_{1-i}} = 1$,
- for all $v_Z \in \hat{V}$: $\sum_{e \in Z} x_{e_1} + \sum_{e \in E(v) \setminus Z} x_{e_0} = \sum_{v_Y \in I_v} x_{v_Y}$.

Note that for any $v \in V$, the variables x_{v_Y} never occur alone in $\mathfrak{S}_{\mathcal{G}^T}$, but always as part of the sum $\sum_{v_Y \in I_v} x_{v_Y}$. We could arguably simplify the system by replacing each of these sums with a new variable without influencing the system's solvability. The advantage of the system's above statement is its direct accessibility to a first-order definition over \mathcal{G}^T as shown by the following lemma.

Lemma 5.4.7. *$\mathfrak{S}_{\mathcal{G}^T}$ is first-order definable over \mathcal{G}^T .*

Proof. Using the coloring of the color nodes, it is easy to construct first-order formulas $\varphi_c(x)$, $\varphi_o(x)$ and $\varphi_i(x)$ that define the sets \hat{C} , \hat{E} and \hat{V} , respectively. Similarly, there is a first-order formula $\theta_p(x, y)$ that says that x and y are distinct outer nodes derived from the same edge $e \in E$, and a first-order formula $\theta_i(x, y)$ that says that x and y are inner nodes derived from the same vertex $v \in V$.

The system $\mathfrak{S}_{\mathcal{G}^T}$ can now be defined by formulas $\varphi(x, y)$ and $\beta(x)$ over \mathcal{G}^T in the following way. Using θ_p , the equations for the outer nodes e_i are easily defined at row indices a for which $\varphi_o[a]$ holds. Similarly, the equations for inner nodes v_Z are defined at row indices a for which $\varphi_i[a]$ holds: we put 1s in all columns y for which either $\theta_i[a, y]$ holds or y is a neighboring outer node of a in \mathcal{G}^T and observe that these are precisely the nodes e_1 for $e \in Z$ and e_0 for $e \in E(v) \setminus Z$. Finally, the equation that sums up all the x_{v_Z} can be defined at row indices a for which $\varphi_c[a] = 1$. There will be multiple copies of this equation, which of course does not affect the solvability of the system. \square

Lemma 5.4.8. *The system $\mathfrak{S}_{\mathcal{G}^T}$ is solvable over GF_2 if and only if \mathcal{G}^T is even.*

5.5 Descriptive complexity of rank logics over ordered structures

Proof. We show that the system is solvable when $|T| = 0$ and not solvable when $|T| = 1$. Since $\mathfrak{S}_{\mathcal{G}^T}$ is definable by a first-order sentence $\varphi(x, y)$ by Lemma 5.4.7 and φ is invariant under isomorphism, it then follows by Theorem 5.4.2 that $\mathfrak{S}_{\mathcal{G}^T}$ is solvable if and only if $|T|$ is even.

First suppose $T = \emptyset$. In this case it is readily verified that the assignment that puts $x_{e_i} = i$ for all $e_i \in \hat{E}$ and $\sum_{v_Y \in I_v} x_{v_Y} = 0$ for all $v \in V$ is a solution to $\mathfrak{S}_{\mathcal{G}^T}$.

Next suppose $T = \{u\}$ where u is an arbitrary vertex in V . Fix one edge $f \in E(u)$ and consider the following equations from $\mathfrak{S}_{\mathcal{G}^T}$:

- for every $v \in V \setminus \{u\}$: $\sum_{e \in E(v)} x_{e_0} = \sum_{v_Y \in I_v} x_{v_Y}$,
- for u : $(\sum_{e \in E(u) \setminus \{f\}} x_{e_0}) + x_{f_1} = \sum_{u_Y \in I_u} x_{u_Y}$.

In this subsystem there is exactly one equation for each $v \in V$. It follows that for all $e \in E \setminus \{f\}$, the variable x_{e_0} occurs exactly twice on the left-hand side of the system, as each edge is connected to two vertices $v \in V$. However, for the edge f , we get both x_{f_0} and x_{f_1} on the left-hand side. Summing up all the above equations we therefore get

$$\sum_{v_Z \in \hat{V}} x_{v_Z} = x_{f_0} + x_{f_1} = 1,$$

where the last equality comes from the corresponding edge equation in $\mathfrak{S}_{\mathcal{G}^T}$. But this is inconsistent with the first equation in $\mathfrak{S}_{\mathcal{G}^T}$. Hence $\mathfrak{S}_{\mathcal{G}^T}$ has no solution. \square

Now by Proposition 5.2.6 and Lemma 5.4.7, there is a sentence π of $\text{FO}+\text{rk}_2$ testing for solvability of the system $\mathfrak{S}_{\mathcal{G}^T}$ over the universe \mathcal{G}^T . By Lemma 5.4.8, π holds in all even CFI graphs and is false in all odd CFI graphs, which finishes the proof of Theorem 5.4.4. \square

Remark 5.4.9. After Cai et al. published their result in 1992, other constructions that expose the limitations of $\text{FP}+\text{C}$ have been found.

- One such construction by Hella [Hel96] will be dealt with in Section 5.6, showing that it too can be defined by rank logics but not by $\text{FP}+\text{C}$. In fact, there we also show that $\text{FP}+\text{rk}_p$ is more expressive than $\text{FP}+\text{C}$ for all primes p , not just for $p = 2$.
- The CFI graph construction has been extended by Torán [Tor04] for general integer moduli, not just even and odd graphs. By using rank operators of corresponding characteristics, these graphs can be distinguished in the same way as the CFI graphs here.
- Gurevich and Shelah [GS96] define a class of rigid structures called *multipedes*, and consider the problem of uniformly defining a linear order over this class. They show that this problem, while computable in polynomial time, is not definable by any fixed formula of $\text{FP}+\text{C}$. Bjarki Holm's thesis [Hol10] shows that also this problem is definable in $\text{FO}+\text{rk}_2$.

5.5 Descriptive complexity of rank logics over ordered structures

Among the most classical results in descriptive complexity theory are capturing results on ordered structures. For instance, as noted in Theorems 2.3.4 and 2.3.10, TC captures nondeterministic logspace and FP captures PTIME in the presence of an order. In this section, we show such natural correspondences for first-order logic with rank operators, namely that $\text{FO}+\text{rk}$ captures the logspace counting hierarchy $\#\text{LH}$ on ordered structures and that for each prime p , $\text{FO}+\text{rk}_p$ captures the complexity class MOD_pL on ordered structures.

For both capturing results in this section we will need to consider *logspace-bounded non-deterministic oracle Turing machines*. The notion of non-deterministic logspace-bounded Turing

machines making oracle queries is not straightforward, since we have to prevent such machines from consulting their oracle a superpolynomial number of times – this would boost the machine’s computational power beyond recognition [RST84]. The model which is most commonly used in the literature, and which we also use here throughout, is due to Ruzzo, Simon, and Tompa (ibid.).

Definition 5.5.1. A *non-deterministic logspace-bounded oracle Turing machine* M is a non-deterministic logspace machine with an additional write-only *oracle query tape* which is not subject to any space bound. In order to query its oracle, M enters a special *query state*, upon which the oracle query tape is erased and the oracle’s answer appears on a special read-only *oracle answer tape* (answers are yes, no, or a computed function value, depending on the oracle). M has the restriction that whenever it starts writing a symbol on the oracle query tape, it must behave deterministically until it enters the oracle query state for the next time.

If K and O are complexity classes and K is defined based on non-deterministic logspace machines, then K^O is the class of languages L for which there is a non-deterministic logspace oracle machine M and $A \in O$ so that M with A as its oracle, denoted M^A , decides L in accordance with K ’s acceptance condition.

Ruzzo, Simon, and Tompa also show that this notion of oracle machines coincides with oracle machines that may only be asked queries of logarithmic length, but which in turn receive themselves access to the original machine’s input. The latter notion is particularly close to logical definability where subformulas always work over the same structure as their parent formula.

Lemma 5.5.2 (Ruzzo, Simon, Tompa [RST84]). *For any languages L and A , there exists a non-deterministic logspace oracle machine M so that $L = M^A$ if and only if there exists a set B and a non-deterministic logspace oracle machine N such that*

- $L = N^B$,
- all of N ’s oracle queries are strings of the form $x\$y$ where x is N ’s input and the length of y is logarithmically bounded, and
- B is logspace many-one reducible to A .

The intuition behind this lemma is simple: if M must write its oracle queries in a deterministic manner, then each of its queries is completely determined by M ’s configuration when it begins writing on the oracle query tape and by the content of its input tape. But then M could simply provide its input tape and configuration to the oracle and the oracle can figure out by itself what the query was supposed to be, all the while obeying the same space bound as M .

5.5.1 $\text{FO}+\text{rk}_p$ captures MOD_pL on ordered structures

Definition 5.5.3. For a non-deterministic Turing machine M , let $|M(x)|$ denote the number of accepting computation paths on input string x . Let $n \in \mathbb{N}$. A MOD_nL -Turing machine M is a non-deterministic Turing machine with logarithmic workspace which is said to accept an input x whenever $|M(x)| \not\equiv 0 \pmod n$.

The complexity class MOD_nL consists of all problems $P \subseteq \{0, 1\}^*$ for which there is a MOD_nL -Turing machine M deciding P .

MOD_2L is better known under the name *parity logspace*, usually denoted by $\oplus\text{L}$. The following result has been published previously in [DGHL09].

Theorem 5.5.4. *Let p be prime. $\text{FO}+\text{rk}_p$ captures MOD_pL on ordered structures.*

Proof. The proof consists of two parts. Firstly, we have to show that for any sentence $\varphi \in \text{FO}+\text{rk}_p$, there is a MOD_pL -Turing machine M_φ that, given the encoding of a structure \mathcal{A} , decides $\mathcal{A} \models \varphi$. Secondly, given a MOD_pL -Turing machine M , we construct a sentence φ_M that holds in a structure \mathcal{A} if and only if M accepts $\text{enc}(\mathcal{A})$.

For the one direction, assume that τ is a vocabulary with $\leq \in \tau$ and that φ is an $\text{FO}+\text{rk}_p[\tau]$ -sentence. In order to deal with rank operators occurring in φ , we need two results on MOD_pL -machines. The first one says that the rank of a matrix over GF_p can be “computed” by a MOD_pL -machine.

Lemma 5.5.5 (Buntrock et al. [BDHM92]). *Let p be prime. There is a MOD_pL machine M_{rk} which takes as input an integer $r \in \mathbb{N}_0$ and a matrix $A \in \text{GF}_p^{m \times n}$ and decides if $\text{rk}A = r$. ■*

The second result states that MOD_pL -machines that make oracle queries to a MOD_pL problem can be simulated by a MOD_pL -Turing machine without oracle queries.

Lemma 5.5.6 (Hertrampf et al. [HRV00]). *$\text{MOD}_p\text{L}^{\text{MOD}_p\text{L}} = \text{MOD}_p\text{L}$ for all prime p . ■*

We show that there is a MOD_pL -machine M_φ that decides φ by induction over the construction of φ . The value of terms will be calculated deterministically, while the truth value of subformulas is decided in accordance with MOD_pL 's acceptance condition. Since existential quantifiers can be expressed using rank operators, it is enough to show the following cases. We use Lemma 5.5.6 wherever possible to shorten our presentation.

- Addition and multiplication can be done deterministically in logspace by Lemma 5.1.8. Term comparisons $t_1 = t_2$ and $t_1 \leq t_2$ are also decided deterministically. The values for t_1 and t_2 are obtained by querying their respective oracles.¹² Atomic formulas $R\vec{x}$ and $x = y$ can be decided deterministically by lookup in $\text{enc}(\mathcal{A})$ on the input tape.
- If $\varphi = \neg\psi$, then M_φ makes an oracle query to M_ψ and accepts if and only if M_ψ rejects. If $\varphi = \psi_1 \wedge \psi_2$, then M_φ makes oracle queries to M_{ψ_1} and M_{ψ_2} and accepts if both queries succeed, otherwise rejects.
- For a term $\rho := \text{rk}_p(\vec{x}_1 \vec{x}_2 \leq t, \vec{y}_1 \vec{y}_2 \leq t) \eta$ let M_t and M_η be MOD_pL -machines for t and η , which exist by our inductive assumption. The machine M_ρ first computes $t^{\mathcal{A}}$ by querying M_t and saves this value, which is possible since $t^{\mathcal{A}}$ is polynomially bounded (Lemma 5.1.8). Then M_ρ behaves as the MOD_pL -machine from Lemma 5.5.5 to compute ρ , where it queries M_η whenever it needs the value of some matrix entry. For the oracle queries, M_ρ provides $\text{enc}(\mathcal{A})$ and assignments for the variables $\vec{x}_1, \vec{x}_2, \vec{y}_1, \vec{y}_2$, which is of course written on the oracle query tape in a deterministic manner.

For any $\text{FO}+\text{rk}_p$ -sentence φ there is therefore, by repeatedly applying Lemma 5.5.6, a MOD_pL -Turing machine M_φ that decides φ .

For the other direction, consider a MOD_pL -Turing machine M with space bound $d \cdot \log n$ that decides a class of τ -structures $K \in \text{MOD}_p\text{L}$. Without loss of generality we assume that M has only one accepting configuration. We construct a formula φ_M that defines K . We may restrict ourselves to structures \mathcal{A} with $|\mathcal{A}| > \max\{d \log |\mathcal{A}|, q\}$ where q is the number of states of M and choose d' large enough so that all configurations of M can be encoded by d' -tuples of elements from \mathcal{A} . All smaller structures in K can be identified by a large yet finite first-order formula.

Consider the configuration graph of M , that is, the directed graph G_M with vertices $\vec{a} \in U(\mathcal{A})^{d'}$ and edges (\vec{a}, \vec{b}) whenever \vec{b} is a successor configuration of \vec{a} under M 's transition relation (cf.

¹²The two different oracles may be combined since MOD_pL is closed under disjoint union. Frequent switches between decision and function computation problems from MOD_pL are necessary.

5 Stronger Logics for Polynomial Time

Section 2.2). If s and t denote the unique start and accept configurations, respectively, M accepting \mathcal{A} is equivalent to the condition that the number of paths from s to t in G_M is $\not\equiv 0 \pmod p$. Note that G_M can be assumed to be free of cycles (see Section 2.2). Since $\text{DTC} \leq \text{FO+rk}_p$ (Corollary 5.2.13), the following result shows that G_M can be defined in FO+rk_p .

Lemma 5.5.7 (Ebbinghaus and Flum, Lemma 7.3.7 of [EF99]). *There are DTC-formulas $\chi_{\text{start}}(\vec{x})$, $\chi_{\text{accept}}(\vec{x})$, and $\chi_{\text{succ}}(\vec{x}, \vec{y})$ such that for all ordered τ -structures \mathcal{A} with $|\mathcal{A}| > \max\{d \log |\mathcal{A}|, q\}$ and $\vec{a} \in U(\mathcal{A})^{d'}$,*

- $\mathcal{A} \models \chi_{\text{start}}[\vec{a}]$ ($\mathcal{A} \models \chi_{\text{accept}}[\vec{a}]$) if and only if \vec{a} is the encoding of the start (accept) configuration of M ,
- $\mathcal{A} \models \chi_{\text{succ}}[\vec{a}, \vec{b}]$ if and only if \vec{b} is a valid successor configuration of \vec{a} .

Remark 5.5.8. An examination of Ebbinghaus and Flum's proof of Lemma 5.5.7 reveals that the deterministic transitive closure operator is solely used for arithmetic purposes that can equally well be defined using addition, multiplication, and a BIT-predicate. The BIT-predicate is discussed by Immerman in [Imm99, Section 1.2], where he also shows that this predicate is FO-expressible using addition and multiplication. Thus, G_M can actually be defined in FO+C over ordered structures.

$\chi_{\text{succ}}(\vec{x}, \vec{y})$ defines the adjacency matrix A of G_M . By Corollary 5.2.17, there is an FO+rk_p term $\eta_M(\vec{x}, \vec{y})$ counting the number of paths in G_M from \vec{x} to \vec{y} modulo p . Thus, for any ordered τ -structure \mathcal{A} with $|\mathcal{A}| > \max\{d \log |\mathcal{A}|, q\}$,

$$\mathcal{A} \models \forall \vec{s} \forall \vec{t} \chi_{\text{start}}(\vec{s}) \wedge \chi_{\text{accept}}(\vec{t}) \rightarrow \eta_M(\vec{s}, \vec{t}) \neq 0$$

if and only if M accepts \mathcal{A} . This completes the proof of Theorem 5.5.4. □

5.5.2 FO+rk captures the logspace counting hierarchy on ordered structures

Just as for the logics FO+rk_p , we can find a complexity class whose computational power closely matches the expressiveness of FO+rk on ordered structures and which is well-established in the literature. It turns out that FO+rk captures the logspace counting hierarchy when an order is present and the nesting depth of rank operators is closely related to the levels of this hierarchy. We start by giving definitions for the relevant complexity classes (see [AO96]).

Definition 5.5.9. $\#L$ is the class of all functions $f : \{0, 1\}^* \rightarrow \mathbb{N}$ for which there is a nondeterministic logspace machine M with $f(x) = |M(x)|$ for all $x \in \{0, 1\}^*$.

Definition 5.5.10 (Logspace counting hierarchy). Define $\#LH_1$ to be $\#L$, and let $\#LH_{i+1}$ be the class of functions f for which there exists a logspace bounded non-deterministic oracle Turing machine M (in the Ruzzo-Simon-Tompa model, see Definition 5.5.1) and some function $g \in \#LH_i$ such that $f = |M^g|$. $\bigcup_i \#LH_i$ is called the $\#L$ hierarchy and denoted simply by $\#LH$.

We also briefly consider $\text{AC}^0(\#L)$, the class of languages decided by a family of logspace-uniform constant-depth polynomial-size circuits which may include oracle gates for a fixed function $f \in \#L$. As we will not work with the details of this complexity class, the reader is referred to [AO96] for the precise definition of $\#L$ oracle gates for circuits.

The main result in this section is the following:

Theorem 5.5.11. FO+rk captures the class $\#LH$ on ordered structures.

The statement of this theorem needs some explanation since #LH is defined as a class of functions rather than languages. In order to make sense of this, we could also say that FO+rk captures the *decision class* $L^{\#LH}$, which has no greater computational power than #LH since #LH is already the entire logspace counting hierarchy. It will become clear that FO+rk *does* capture #LH in this sense. However, the counting logics we are considering are not restricted to defining classes of structures, either – their terms can just as well be used to associate integral quantities with the underlying structures. As it turns out, every function in #LH can also be *represented* by a term from FO+rk in a way explained below, and vice-versa. Thus, the notions of *deciding and counting* in the machine world correspond in this case to the notions of *defining formulas and terms* in the logical world. The statement of Theorem 5.5.11 is warranted by showing that FO+rk corresponds to #LH in this twofold sense.

One property that makes #L apparently harder a complexity class than LOGSPACE is that it contains functions that grow exponentially with the size of the input. For example, it is easy to define a #L machine that counts the number of paths between two vertices in a directed graph, which can easily become exponential. In logarithmic space, however, only numbers of at most polynomial size can be represented on the worktape.¹³ Likewise, the terms in our rank logics are polynomially bounded (Lemma 5.1.8). As before, we therefore have to choose a representation for handling large numbers in FO+rk. Since we have rank operators over prime finite fields available, Chinese remainder representations (x_p) with respect to a set of primes \mathcal{P} are particularly suitable for this task. We refer to Section 5.3.2 for a discussion of CRRs and recall that we can freely convert between binary and Chinese remainder representations in FO+C.

We will show Theorem 5.5.11 by proving a more technical statement in Lemma 5.5.12. It relates the levels of #LH to the *nesting depth of rank operators* inside other rank operators (Definition 5.1.7). In order to obtain the precise correspondence between nesting depth of rank operators and levels of #LH, we have to ensure not to *wastefully apply* rank operators where less expressive operators suffice.

For this purpose, we augment FO+rk with explicit counting quantifiers **and** with an additional deterministic transitive closure operator. Neither augmentation increases the expressiveness of FO+rk since both operations are FO+rk-definable (see Proposition 5.2.2 and Corollary 5.2.13), so we also continue to denote the logic by FO+rk. In fact, the additional dtc-operator is not really necessary since it is only used for Ebbinghaus and Flum’s result on the definability of configuration graphs (Lemma 5.5.7). As noted in Remark 5.5.8, this result may also be shown using counting operators to transfer the ordered structure over to the number sort and using the arithmetic available there. A proper proof of this result, however, would require a tedious and largely uninteresting recast of Ebbinghaus and Flum’s proof. Since it would only change our result insignificantly underneath the surface, we refrain from doing so and leave it to the reader to select her favorite augmented version of FO+rk.

Theorem 5.5.11 now follows immediately from the following lemma.

Lemma 5.5.12 (Nesting Level Correspondence). *Let $\text{FO+rk}\langle k \rangle$ consist of those FO+rk formulas and terms for which the nesting depth of rank operators is at most k . Then $\#LH_k \subseteq \text{FO+rk}\langle k \rangle \subseteq L^{\#LH_k}$ on ordered structures.*

Lemma 5.5.12 also establishes the following corollary.

Corollary 5.5.13. *The #L hierarchy collapses if and only if the nesting hierarchy of rank operators of FO+rk collapses over ordered structures.* \square

¹³This does not separate these complexity classes, though, as in principle it may be possible that all #L functions are computable by logspace transducers, whose polynomial-length output suffices to represent exponential values. In general, none of the inclusions in $\text{LOGSPACE} \subseteq \text{NLOGSPACE} \subseteq L^{\#L} \subseteq L^{\#LH} \subseteq \text{PTIME}$ are known to be strict.

5 Stronger Logics for Polynomial Time

Compare this result with Lemma 5.5.6 in Section 5.5.1, which implies that the nesting hierarchy of rank operators over a fixed field GF_p does collapse over ordered structures. Whether the #L hierarchy collapses is an open question. Corollary 5.5.13 says this happens unless nesting rank operators for different prime fields increases the logic's expressiveness.

In [AO96], Allender and Ogihara show that #LH coincides with logspace-uniform $\text{AC}^0(\#L)$, which gives us the following corollary to Theorem 5.5.11.

Corollary 5.5.14. *FO+rk captures logspace-uniform $\text{AC}^0(\#L)$ on ordered structures.*

Given the close relationship between AC^0 and FO, it should not be surprising that the last part of the proof of Lemma 5.5.12 uses the same technique as the proof of Allender and Ogihara's result. In fact, the main result in this section could also be proved by showing Corollary 5.5.14 directly. However, this would not make the proof significantly easier and showing the correspondence to #LH enables us to explicitly tie the levels of the logspace counting hierarchy to the nesting depth of rank operators in FO+rk.

Most work in the proof of Lemma 5.5.12 goes into showing that on the one hand, single rank operators rk_p can be evaluated in $L^{\#L}$ uniformly for all small primes p (Lemma 5.5.15) and on the other hand, for any #L-function f there is a corresponding FO+rk-term that *equals* f over any structure (Lemma 5.5.16). The ability to quantify over the prime characteristic p of the base field for which the rank is calculated will be crucial to recover the full power of the logspace counting class. The proof is then completed by showing that the two lemmas also work in the presence of oracles, which allows for an inductive argument revealing the close connection between the levels #LH _{i} of the #L-hierarchy and the nesting depth of rank operators in formulas from FO+rk. We first prove the mentioned lemmas.

Lemma 5.5.15. *Let $\eta(\vec{x}, \vec{y})$ be an FO+C-term, and let $\rho(p) = \text{rk}_p \eta(\vec{x}, \vec{y})$. For \mathcal{A} and a small prime p as input, calculating $\rho^{\mathcal{A}}(p)$ is in $L^{\#L}$.*

Proof. The central observation in this proof is that the rank of a matrix A over GF_p can be inferred from the number of accepting and rejecting paths of a certain nondeterministic logspace Turing machine R . The techniques to show the existence of R are well established and the result is essentially implicit in the works of Toda [Tod91]; Allender, Beals, and Ogihara [ABO99]; Buntrock, Damm, Hertrampf, and Meinel [BDHM92]. However, since none of the authors considers the case of the modulus p given as input, we retrace the relevant steps of the argument here and show that R can be constructed uniformly. The reader may want to skip this development at first reading and go on to the end of the proof.

The idea is to reduce the computation of the rank of a matrix A to computing its characteristic polynomial and then reduce this problem further to the computation of an iterated matrix product. Following Mulmuley's algorithm for computing the rank ([Mul86], see also Section 5.3.1 and [Koz92]), first consider the symmetric matrix $A' = \begin{pmatrix} 0 & A \\ A^t & 0 \end{pmatrix}$ and let $B = YA'$ where $Y = (y_{ij})$ is the diagonal matrix of the same dimension as A' with $y_{ii} = x^{i-1} \in \text{GF}_p[x]$. B is now a matrix over the polynomial ring $\text{GF}_p[x]$, which is a principal ideal domain, so that the rank of B is well-defined (see [AW92] for detailed background on matrices over PIDs). Mulmuley shows that $\text{rk} B = \text{rk} A' = 2\text{rk} A$ and, in fact, that $\text{rk} A = k$ if and only if precisely the first $2n - 2k$ coefficients of B 's characteristic polynomial are all zero modulo x^{4n^2} (the argument uses Lemma 5.3.12).

By a result of Berkowitz [Ber84], we can compute from B in logspace a sequence of matrices D_i so that the coefficients c_0, c_1, \dots, c_r of the characteristic polynomial of B appear in positions $(1, r+1), (1, r), \dots, (1, 1)$ of the matrix $\prod_i D_i$. Observe that the coefficients c_j of B 's characteristic polynomial are themselves elements of $\text{GF}_p[x]$, and the same holds for the elements in the matrices D_i . However, since we only need to calculate the coefficients modulo x^{4n^2} , we can faithfully

represent elements from $\text{GF}_p[x]/x^{4n^2}$ by upper-triangular Toeplitz matrices. A matrix is *Toeplitz* if on each diagonal the entries are all equal. For example,

$$\begin{pmatrix} 0 & 1 & 2 \\ 3 & 0 & 1 \\ 4 & 3 & 0 \end{pmatrix}$$

is a Toeplitz matrix. The representation is as follows: if $s(x) = s_0 + s_1x + s_2x^2 + \dots + s_{4n^2-1}x^{4n^2-1}$ is a polynomial with coefficients $s_i \in \text{GF}_p$ and d_i denotes the value of a $4n^2 \times 4n^2$ Toeplitz matrix S on the i^{th} diagonal above the main diagonal (d_0 being the value of S on the main diagonal), then S corresponds to $s(x)$ if $d_i = s_i$ for all $i \in [0, 4n^2 - 1]$, and S is zero everywhere below the diagonal. It is easy to verify that this is a ring isomorphism from $\text{GF}_p[x]/x^{4n^2}$ with addition and multiplication of polynomials to the space of upper-triangular Toeplitz matrices endowed with addition and matrix multiplication.

Now let D'_i be the matrix D_i where each $\text{GF}_p[x]$ entry is replaced with the corresponding $4n^2 \times 4n^2$ upper-triangular Toeplitz matrix. Clearly, the D'_i can be computed from the D_i in logspace. Now $\text{rk}A = k$ if and only if precisely all the entries $(1, (r+2) \cdot 4n^2 - 1), \dots, (1, (r+2-2k) \cdot 4n^2)$ are zero in GF_p . Finally, by a result which is attributed to Valiant in [ABO99], computing the entries of a matrix which is the result of iterated matrix multiplication is contained in GapL, i.e., there is a nondeterministic logspace machine M so that upon input matrices D'_i and numbers a, b , the $(a, b)^{\text{th}}$ entry of $\prod_i D'_i$ is equal to the difference between the numbers of accepting and rejecting paths of M . Unlike in the reductions above, this last step does not directly produce an element of GF_p , but instead an integer which still has to be reduced modulo p .

Summing up the above discussion, there is a non-deterministic logspace machine R which takes as input a matrix A , a prime p and an index i , returning an integer $\Delta R(A, p, i)$ as the difference between the number of accepting and rejecting paths, with the property that $\text{rk}A$ over GF_p can be easily inferred from the indices i for which $\Delta R(A, p, i) = 0 \pmod p$. Now let T be the deterministic oracle Turing machine that does the following: T cycles through all possible values of \vec{x} and \vec{y} over the input structure \mathcal{A} and outputs the matrix $\eta^{\mathcal{A}}(\vec{x}, \vec{y})$ on its oracle tape. Then T queries the GapL-function corresponding to R repeatedly in order to determine $\text{rk}A \leq \max\{|\vec{x}|, |\vec{y}|\}$ and outputs that value when it has been found. The proof is finished using the observation that $\text{L}^{\text{GapL}} = \text{L}^{\#\text{L}}$ (cf. [AO96]). \square

Lemma 5.5.16. *Let M be a non-deterministic logspace Turing machine. Then there is a formula $\eta(x)$ of FO+rk whose rank operators have nesting depth 1 such that for any ordered structure \mathcal{A} , $\eta^{\mathcal{A}}(x)$ is the binary representation of the integer $|M(\text{enc}(\mathcal{A}))|$.*

Proof. As in Section 5.5.1, we consider the configuration graph G_M of M (cf. Section 2.2). Since we want to avoid using rank operators in the definition of G_M , we now have to use that we augmented FO+rk by explicit counting and dtc operators. By Lemma 5.5.7, G_M is DTC-definable. By Remark 5.5.8, G_M can even be defined in FO+C, so just adding counting quantifiers to FO+rk suffices for defining G_M . The point here is not to use rank operators in the definition of G_M since this would spoil the nesting level correspondence we are aiming for. From a conceptual viewpoint, counting and deterministic transitive closure are classical LOGSPACE-operations and can be applied ubiquitously without changing the level in the logspace counting hierarchy – rank operations, however, do change the level and shall only be used to *count* the number of the machine’s accepting paths.

Of course, G_M and the start configuration of M depend on the input x , but in this proof we ignore this dependence for notational convenience and write $|M|$ instead of $|M(x)|$. Since M has at most polynomially many configurations, we have $|M| \leq 2^{n^d}$ for some fixed d , where $n = |\mathcal{A}|$.

5 Stronger Logics for Polynomial Time

We are going to represent $|M|$ in CRR, so we need a finite set of primes \mathcal{P} so that $\prod \mathcal{P} \geq 2^{n^d}$. As in Section 5.2.8, we use the Prime Number Theorem 5.2.23, by which the number $\pi(n)$ of primes less than n has the asymptotic limit $n/\ln n$, i.e., $\lim_{n \rightarrow \infty} \frac{\pi(n)}{n/\ln n} = 1$. So if \mathcal{P}_n denotes the set of all primes less than n , then

$$\prod \mathcal{P}_{n^{d+1}} > 2^{\pi(n^{d+1})} \geq 2^{n^d}$$

for large enough n . As usual, the cases where n is not large enough can be dealt with separately using a case distinction to define the desired binary representation directly for this finite number of cases. The set $\mathcal{P}_{n^{d+1}}$ is easily defined in FO+C since all the primes are small.

Let A be the adjacency matrix of G_M and let \vec{s}, \vec{t} be the indices corresponding to the unique start and accept configuration, respectively. Let $\sigma_M(p)$ be the FO+rk-term of Corollary 5.2.17 which contains the number of paths from \vec{s} to \vec{t} in G_M modulo p .¹⁴ Inspecting the proof of Corollary 5.2.17 (or rather of Lemma 5.2.16) we see that σ_M has nesting depth 1 of its rank operators. Now $\sigma_M(p)$ and $\mathcal{P}_{n^{d+1}}$ together define the CRR of $|M|$.

By Theorem 5.3.4, we may convert $\sigma_M(x)$ in FO+C to a formula $\eta(x)$ defining the binary representation of $|M|$. On the whole, the only place where rank operators were used was in the term σ_M counting the number of M 's accepting paths in G_M . While the counting has to be done *in parallel* for polynomially many prime field characteristics p , rank quantifiers are not being nested, bounding the nesting depth at 1. \square

Before we can complete the proof of Lemma 5.5.12 and Theorem 5.5.11, we need one more result stating that each level of the #L hierarchy is closed under disjoint union. This small observation does not seem to be documented in the literature yet.

Lemma 5.5.17 (#LH_k closed under disjoint union). *Let g_1, \dots, g_ℓ be a finite list of #LH_k functions. Then the function g which takes as its first input an index i and returns the value g_i computed on the rest of the input is also in #LH_k.*

Proof. The proof is by a simple induction on the level k . If $g_1, \dots, g_\ell \in \#L$, it is clear that $g \in \#L$. If $g_1, \dots, g_\ell \in \#LH_k$, let o_1, \dots, o_ℓ be their #LH_{k-1} oracles. By induction, the function o which computes value o_i on input i is also contained in #LH_{k-1}. Now on input i , g simulates g_i and whenever g_i wants to query its oracle o_i , g queries o instead, handing on the index i . Thus, $g \in \#LH_k$, completing the inductive step. \square

Proof of Lemma 5.5.12. We use induction on the nesting depth of rank operators and the levels #LH_i of the logspace counting hierarchy to show that any term of nesting depth k can be computed in L^{#LH_k}, and that any function in #LH_k is definable by a term from FO+rk of nesting depth at most k . The base cases $k = 1$ both follow directly from Lemmas 5.5.15 and 5.5.16, respectively.

Let η be an FO+rk-term whose rank operators have nesting depth k . As observed in Section 2.3.3, other than the rank operators all logical and arithmetic operations can be evaluated in L. Let $Q = \{\theta_i = \text{rk}_p \psi_i\}$ be a list of all the outermost occurrences of rank operators in η . The nesting depth of each ψ_i is at most $k - 1$, so by induction there are functions $f_i \in L^{\#LH_{k-1}}$ carrying the value of the respective ψ_i when given as input $\text{enc}(\mathcal{A})$ along with an assignment to the free variables of ψ_i . Let g_i be the #LH_{k-1}-oracle used by f_i , and let g be the function which takes as its first input an index i and then computes the g_i on the rest of the input. By Lemma 5.5.17, $g \in \#LH_{k-1}$.

From this, we can now construct a #LH_k-function f from which we can infer all the values in Q and which works analogously to the #L-machine used as the oracle in the proof of Lemma 5.5.15. f takes as input an index i , an assignment to the free variables of θ_i , an index j and the encoding of the input structure \mathcal{A} . f then uses f_i and its access to \mathcal{A} to compute the entries of the matrix

¹⁴We can define σ over G_M because FO+rk is closed under logical reductions (Lemma 5.2.9).

defined by ψ_i as needed, querying g with argument i whenever f_i wants to query its oracle g_i . In the same way as in the proof of Lemma 5.5.15, $\text{rk}_p \psi_i$ can be directly inferred in logspace from the indices j for which f returns an integer equivalent to zero modulo p . Thus, $f \in \#\text{LH}_k$ and η can be evaluated in $L^{\#\text{LH}_k}$.

For the converse direction of the proof let f be a $\#\text{LH}_k$ function. As in the proof of Lemma 5.5.16, we want to define the configuration graph G_M of the non-deterministic logspace oracle Turing machine M for f and use the rank operators to gain access to the CRR of $|M|$. By Lemma 5.5.2, we may assume that M 's query tape is logarithmically bounded, while M 's oracle always obtains full access to M 's input tape. By the conditions of Lemma 5.5.2, this assumption does not change the complexity class that oracles may be chosen from. Since the oracle query tape is logarithmically bounded now, its contents may be included in the encoding of M 's configurations by finitely many variables.

Let $g \in \#\text{LH}_{k-1}$ be M 's oracle. By induction, there is an FO+rk-term γ defining g whose rank operators have nesting depth at most $k-1$. γ directly encodes the binary string which corresponds precisely to g 's answer on M 's oracle tape. Thus, we may use γ directly in the definition of G_M 's edge relation whenever M enters its oracle query state. Arguing entirely analogously as in the proof of Lemma 5.5.16, we obtain the CRR of $|M|$ as a term $\eta(x)$ with nesting depth of rank operators at most k . By employing Theorem 5.3.4 again, we obtain a formula encoding the binary representation of f . \square

5.6 The arity hierarchy of rank operators

Our aim for this section is to show that increasing the number of variables used for defining matrices also increases the expressiveness of the rank operators evaluated over these matrices. By a result of Hella [Hel96], if an *operator* does not satisfy this condition its addition to FP+C also falls short of capturing PTIME. As this is the most recent and arguably the deepest technique known for separating FP+C from PTIME, our result here establishes the logic FP+rk as a *candidate for capturing PTIME* on the class of all structures. This means that proving FP+rk to be too weak to capture PTIME will require completely new techniques. Along the way, we also show in this section that FP+rk_p is more expressive than FP+C for any prime p .

In order to obtain our result, we need to transfer our rank logics into the more generic framework of *infinitary logic with generalized quantifiers*. Generalized quantifiers were introduced by Lindström in [Lin66] and have been studied as a way to increase the expressiveness of FO by a prescribed query (for example [EF99], [KV95], [Daw95, DH95], [Hel89, HLV96], [DG10]). Generally, if σ is a vocabulary and K is any class of σ -structures, then

$$\mathcal{A} \models Q_K \vec{x}_1 \dots \vec{x}_k (\psi_1(\vec{x}_1), \dots, \psi_k(\vec{x}_k))$$

if $(U(\mathcal{A}), \psi_1^A[\cdot], \dots, \psi_k^A[\cdot]) \in K$ (also see Definition 5.6.5). The arity of Q_K is $\max\{|\vec{x}_1|, \dots, |\vec{x}_k|\}$. If \mathcal{Q} is a set of Lindström quantifiers, then $L(\mathcal{Q})$ denotes the extension of a logic L by all the quantifiers in \mathcal{Q} .

In 1996, Hella [Hel96] proved that for any $n \in \mathbb{N}$, augmenting bounded-variable infinitary first-order logic with all Lindström quantifiers of arity at most n , denoted $L_{\infty\omega}^\omega(\mathcal{Q}_n)$, is not expressive enough to define all PTIME queries over the class of all structures (not necessarily ordered). As Grädel and Otto [GO92] showed that $\text{FP+C} \leq L_{\infty\omega}^\omega(\mathcal{Q}_1)$, this result extends Cai, Fürer, and Immerman's result discussed in Section 5.4.

Our goal now is to show that the arities of rank operators yield a strict hierarchy. Rank operators are not themselves Lindström quantifiers, but they can be translated into such quantifiers. In particular, rank operators which do not bind number variables can be transformed into quantifiers

5 Stronger Logics for Polynomial Time

of the form $\text{rk}_p^{\overline{r}} \overline{x} \overline{y} \overline{\psi}$ whose semantics are easy: $\text{rk}_p^{\overline{r}}$ is considered *true* if the matrix defined by the formulas $\overline{\psi}$ with respect to row variables \overline{x} and column variables \overline{y} has rank $r \in \mathbb{N}$ over GF_p . Similarly to the previously used *rank operators*, the *arity* of this quantifier is $|\overline{x}| + |\overline{y}|$. A precise definition of these quantifiers will be given in the upcoming Section 5.6.1.

Let us write $\text{FP}+\text{rk}^{[n]}$ for the set of all those $\text{FP}+\text{rk}$ -formulas in which all occurrences of rk -operators are of arity at most n . It will be shown in Section 5.6.1 that formulas of $\text{FP}+\text{rk}_p^{[n]}$ can be translated into $L_{\infty\omega}^{\omega}(\mathcal{R}_p^n)$ for some (infinite) set of quantifiers $\mathcal{R}_p^n \subset \mathcal{Q}_n$, so in particular $\text{FP}+\text{rk}^{[n]} \leq L_{\infty\omega}^{\omega}(\mathcal{Q}_n)$ for all $n \in \mathbb{N}$. Using this connection between rank operators, Lindström quantifiers, and the techniques of Hella, we will be able to prove the following theorem, which the remainder of this section is devoted to.

Theorem 5.6.1. *For any $n \in \mathbb{N}$ and any prime p there is an $\text{FO}+\text{rk}_p^{[n+1]}$ query that is not definable in $L_{\infty\omega}^{\omega}(\mathcal{Q}_n)$. Thus, for any $n \in \mathbb{N}$ and prime p*

- $\text{FP}+\text{rk}_p^{[n]} \not\leq \text{FP}+\text{rk}_p^{[n+1]}$ and $\text{FO}+\text{rk}_p^{[n]} \not\leq \text{FO}+\text{rk}_p^{[n+1]}$, and
- $\text{FP}+\text{rk}^{[n]} \not\leq \text{FP}+\text{rk}^{[n+1]}$ and $\text{FO}+\text{rk}^{[n]} \not\leq \text{FO}+\text{rk}^{[n+1]}$.

We prove this theorem by generalizing Hella's construction from [Hel96] which separates $L_{\infty\omega}^{\omega}(\mathcal{Q}_n)$ from PTIME for every $n \in \mathbb{N}$. The queries we define depend on a prime p and the arity n and still cannot be defined in $L_{\infty\omega}^{\omega}(\mathcal{Q}_n)$, yet they can be expressed using a linear system over GF_p of arity $n+1$.

As mentioned above, $\text{FP}+\text{C} \leq L_{\infty\omega}^{\omega}(\mathcal{Q}_1)$ [GO92]. Therefore, we finally obtain separations between counting logics and rank logics for all prime field characteristics, not just for GF_2 as in Section 5.4.

Corollary 5.6.2. $\text{FO}+\text{C} \not\leq \text{FO}+\text{rk}_p$ and $\text{FP}+\text{C} \not\leq \text{FP}+\text{rk}_p$ for every prime p . □

Corollary 5.6.3. $\text{STC}+\text{C} \not\leq \text{FO}+\text{rk}_p$ for all primes p . □

Corollary 5.6.4. $\text{TC}+\text{C} \not\leq \text{FO}+\text{rk}$. □

5.6.1 Characterizing Lindström quantifiers

We begin by defining generalized quantifiers.

Definition 5.6.5 (Lindström quantifiers). Let $\sigma = (R_1, \dots, R_k)$ be a vocabulary with relations R_i of arity n_i . For any class of σ -structures K which is closed under isomorphism, let Q_K denote the *Lindström quantifier* associated with K . The *arity* of Q_K is the maximum value among the n_i . Now for any vocabulary τ , a τ -structure \mathcal{A} satisfies the formula $Q_K \overline{x}_1 \dots \overline{x}_k (\psi_1(\overline{x}_1), \dots, \psi_k(\overline{x}_k))$ if $(U(\mathcal{A}), \psi_1^{\mathcal{A}}, \dots, \psi_k^{\mathcal{A}}) \in K$ as a σ -structure.

Let us illustrate this definition by relating it to our rank operators and showing how to frame them as generalized quantifiers. We point out that only *rank operators without number variables* are directly related to the rank quantifiers we define now; the general case will be dealt with in Section 5.6.2. We define quantifiers $\text{rk}_p^{\overline{r}}$ for each $r \in \mathbb{N}$ and prime p , which bind $p-1$ formulas $\psi_1, \dots, \psi_{p-1}$. The intended meaning of the formulas $\psi_1, \dots, \psi_{p-1}$ is to encode the values $a_{\overline{x}\overline{y}} \in \text{GF}_p$ that the underlying matrix $A = (a_{\overline{x}\overline{y}})$ takes. For a given position $(\overline{x}, \overline{y})$, we let $a_{\overline{x}\overline{y}} = i$ if $\psi_i[\overline{x}, \overline{y}]$ is true and $\psi_j[\overline{x}, \overline{y}]$ is false for all $j < i$. If all formulas are false in position $\overline{x}\overline{y}$ we let $a_{\overline{x}\overline{y}} = 0$.¹⁵

Notice that the row and column dimensions $|\overline{x}|$ and $|\overline{y}|$ are implicitly part of the definition of $\text{rk}_p^{\overline{r}}$. We only make this dependence explicit in the way we separate variables indexing rows and

¹⁵Observe that this definition of $\text{rk}_p^{\overline{r}}$ is of course not the only way to cast rank operators as Lindström quantifiers. Instead of multiple formulas, one may also have used just one formula with an indicator variable for the field element; however, such a definition would not be as robust in the absence of an ordering of the structure.

columns. In any case, we will always consider all rank quantifiers of a fixed arity $n = |\vec{x}| + |\vec{y}|$ together. The quantifiers $\text{rk}_p^{\vec{r}}$ are indeed Lindström quantifiers, where the underlying class of structures \mathcal{A} consists of all $|\mathcal{A}|^{|\vec{x}|} \times |\mathcal{A}|^{|\vec{y}|}$ -matrices of rank r over GF_p which are encoded by $p - 1$ relations of arity n in the above way. By renaming variables, we can always ensure that all formulas ψ_i use the same variables \vec{x}, \vec{y} for defining the matrix, and we write $\text{rk}_p^{\vec{r}} \vec{x}\vec{y}\vec{\psi}$ as a shorthand for the occurrence of rank quantifiers.

Let us briefly restrict our attention to square matrices and consider a rank quantifier $\text{rk}_p^{\vec{r}} xy\vec{\psi}$ of arity 2 and its corresponding structure class C as their simplest case. Clearly, rank quantifiers $\text{rk}_p^{\vec{r}} \vec{x}\vec{y}\vec{\psi}$ with $|\vec{x}| = |\vec{y}| = k$ serve the same purpose of defining the rank of square matrices as their arity-2 counterpart, with the only difference that the matrix dimension is $n^k \times n^k$ instead of $n \times n$. Apparently, these arity- $2k$ rank quantifiers could also be defined from C in a fairly uniform way and indeed, these quantifiers are what Dawar [Daw95] calls a *uniformly generated sequence of quantifiers* with respect to C . This process of defining higher-arity Lindström quantifiers from a given quantifier Q is a general method for *vectorizing* quantifiers (compare [Hel96]). By a result of Dawar [Daw95], there is a set of Lindström quantifiers \mathcal{Q} so that $\text{FO}(\mathcal{Q})$ captures PTIME if and only if there is a uniformly generated sequence of quantifiers with this property. Since it follows from the proof of Theorem 5.6.1 that the arity hierarchy of rank quantifiers is strict, these quantifiers also pass this formal uniformity criterion.

We remark that for the tight version of the arity hierarchy result in this section, we need to consider non-square matrices. By embedding non-square matrices into square matrices of higher arity, our proof also shows a strict arity hierarchy for rank operators which are restricted to square matrices. Considering rectangular matrices as well gives us a more detailed picture, though.

Hella's n -bijjective k -pebble game characterizing $\text{L}_{\infty\omega}^k(\mathcal{Q}_n)$ In [Hel96], Hella gives a precise characterization of $\text{L}_{\infty\omega}^k(\mathcal{Q}_n)$ in terms of an Ehrenfeucht-Fraïssé game, which will be one of the building blocks for the proof of Theorem 5.6.1. We simply state how the game works here; later, it will be adapted to the particular structures that come up in our construction.

The *n -bijjective k -pebble game* is played by two players, *spoiler* and *duplicator*, on two structures \mathcal{A} and \mathcal{B} . A maximum of k pairs of pebbles can be placed on elements from $U(\mathcal{A})$ and $U(\mathcal{B})$, respectively. We write $\langle \mathcal{A}, \vec{a} \rangle$ and $\langle \mathcal{B}, \vec{b} \rangle$ in order to emphasize that the i^{th} pebbles in \mathcal{A} and \mathcal{B} are placed on $a_i \in U(\mathcal{A})$ and $b_i \in U(\mathcal{B})$, respectively. Spoiler wins the game if at any point in time, the map defined by $a_i \mapsto b_i$ is not a partial isomorphism from \mathcal{A} to \mathcal{B} . Duplicator wins infinite games in which spoiler never succeeds to force a pebble configuration that is not a partial isomorphism.

The game proceeds in rounds, in each of which the following happens. Suppose $\langle \mathcal{A}, \vec{a} \rangle$ and $\langle \mathcal{B}, \vec{b} \rangle$ with $|\vec{a}| = |\vec{b}| \leq k$ are the pebble configurations before the round. First, duplicator gives a bijection $f : U(\mathcal{A}) \rightarrow U(\mathcal{B})$. Then spoiler picks up an arbitrary number of pebble pairs (a_i, b_i) . Out of the at most k pebble pairs which are not placed on the structures, spoiler can now place up to n pebbles on elements of \mathcal{A} , and for each $a \in U(\mathcal{A})$ which gets newly marked with some pebble i , he also places \mathcal{B} 's corresponding i^{th} pebble on $f(a)$. This completes the round.

In this game, duplicator's job seems rather hard since she has to settle on a bijection at the beginning of each round. Spoiler's scope for attacking the presumption that this bijection is an isomorphism is only limited by the number n of pebbles that he can newly place, and the maximum number k of pebbles that he is allowed to use overall. Hella showed that this game precisely captures the expressiveness of $\text{L}_{\infty\omega}^k(\mathcal{Q}_n)$.

Theorem 5.6.6 (Hella [Hel96]). *An ℓ -ary query q on σ -structures is definable in $\text{L}_{\infty\omega}^k(\mathcal{Q}_n)$ if and only if spoiler has a winning strategy in the n -bijjective k -pebble game with starting configurations $\langle \mathcal{A}, \vec{a} \rangle, \langle \mathcal{B}, \vec{b} \rangle$ whenever $\vec{a} \in q(\mathcal{A})$, $\vec{b} \notin q(\mathcal{B})$. Similarly, a Boolean query q on σ -structures is definable in $\text{L}_{\infty\omega}^k(\mathcal{Q}_n)$ if and only if spoiler has a winning strategy in the n -bijjective k -pebble*

game with starting configurations $\langle \mathcal{A}, \emptyset \rangle, \langle \mathcal{B}, \emptyset \rangle$ whenever $q(\mathcal{A}) \neq q(\mathcal{B})$. ■

5.6.2 Embedding FP+rk in infinitary logic with rank quantifiers

As a second basic building block in proving Theorem 5.6.1, we need to show that $\text{FP+rk}_p^{[n]}$ is in fact contained in $L_{\infty\omega}^\omega(Q_n)$. We will later construct queries which are not definable in $L_{\infty\omega}^\omega(Q_n)$ and therefore not definable in $\text{FP+rk}_p^{[n]}$. The translation of $\text{FP+rk}_p^{[n]}$ -sentences to $L_{\infty\omega}^\omega(Q_n)$ -sentences is easy but technical. It largely follows the outline of Ebbinghaus and Flum's proof that FP+C is contained in $L_{\infty\omega}^\omega(\mathcal{C})$, where \mathcal{C} is the set of all (Lindström) counting quantifiers of the form $\exists^{\geq r} x \varphi$ with $r \in \mathbb{N}$ [EF99, Proposition 8.4.18].

Actually, when embedding $\text{FP+rk}_p^{[n]}$ into $L_{\infty\omega}^\omega(Q_n)$, we do not require *all* Lindström quantifiers from Q_n to be present. When the rank operators occurring in a sentence $\varphi \in \text{FP+rk}_p^{[n]}$ only bind element variables, then there is a sentence φ' of $L_{\infty\omega}^\omega$ with rank quantifiers which is equivalent to φ . We do not know, however, how to express rank operators with simple rank quantifiers in case they also bind number variables, since $L_{\infty\omega}^\omega$ does not have an ordered domain available on which to define such matrices. Yet, we can express such mixed type rank operators by using different kinds of Lindström quantifiers from Q_n . As these quantifiers seem both unwieldy and specific, though, and since proving Theorem 5.6.1 does not require a tighter embedding result than into $L_{\infty\omega}^\omega(Q_n)$, we refrain from placing any more emphasis on these quantifiers and only describe them during the proof of the embedding result.

Compare this to the situation of embedding FP+C into $L_{\infty\omega}^\omega(\mathcal{C})$. There, the fixed-point operator can be used to break apart counting operators which bind several variables [GO92]. In the resulting formulas, counting operators only bind one variable each, and those operators binding number variables can be replaced by a disjunction over all possible values. For rank operators, however, Theorem 5.6.1 implies that fixed-points cannot generally be used to break down rank operators into operators of smaller arity. Still, this does not rule out the possibility that in the special case of number variables, the order on the number domain could be used to reduce general rank operators to rank operators only binding element variables of the same (or smaller) arity. We leave open the question of whether this is really possible and remark that this is of interest in the development of game characterizations for rank logics.

Proposition 5.6.7. $\text{FP+rk}^{[n]} \leq L_{\infty\omega}^\omega(Q_n)$ for all $n \geq 2$.

Proof. The proof is an extension of the proof of [EF99, Proposition 8.4.18]. Given a sentence φ of $\text{FP+rk}^{[n]}$, we show that it is equivalent to a sentence of the form

$$\bigvee_{m \geq 1} ((\text{rk}_p^{\leq m}(x, y) x = y) \wedge \varphi_m)$$

where p may be any prime and φ_m is a sentence of $\text{FO}(Q_n)$ that captures the meaning of φ in models of cardinality m . Let us fix $m \in \mathbb{N}$ and assume now that we are working over a structure of size m while constructing φ_m . The translation is done in several steps:

1. replace term bounds of the form $\mu \leq t$ with fixed bounds such as $\mu \leq z$ with $z \in \mathbb{N}$
2. eliminate non-atomic numeric terms
3. replace mixed-type rank operators with quantifiers from Q_n
4. eliminate fixed-point operators
5. eliminate numeric terms

1. Replace term bounds with fixed bounds. Recall that by Lemma 5.1.8, each term $t(\mu_1, \dots, \mu_k)$ is associated with a polynomial $p_t(x, x_1, \dots, x_k)$ so that $t^A \leq p_t(|\mathcal{A}|, \mu_1^A, \dots, \mu_k^A)$ on all structures \mathcal{A} . Now since φ is a sentence, each free variable μ_i of t occurs bound by some term $t_{\mu_i}(\mu'_1, \dots, \mu'_{k'})$, for example as $\exists \mu_i \leq t_{\mu_i} \psi$ or $\text{rk}_p(x, \mu_i \leq t_{\mu_i}) \psi$. Thus, μ_i is itself bounded by $p_{t_{\mu_i}}(|\mathcal{A}|, \mu'_1, \dots, \mu'_{k'})$, and t is bounded by the polynomial $p_t(x, p_{t_{\mu_1}}, \dots, p_{t_{\mu_k}})$. We continue inductively replacing the polynomial's variables corresponding to free term variables. Since φ is a finite sentence, this process ends and produces a polynomial $q_t(x)$ with the property that $t^A \leq q_t(|\mathcal{A}|)$ on all structures \mathcal{A} .

Using these polynomials we can now replace the terms bounding number variables with absolute numbers in each subformula ψ of φ . While the semantics of the resulting formulas should be clear, we do not elaborate any further on the nature of these integers in our formulas since they themselves will be eliminated in the subsequent steps.

- $\psi = \exists \mu \leq t \chi$ becomes $\exists \mu \leq q_t(m) (\mu \leq t \wedge \chi)$
- $\psi = (\text{ifp}_{X \leftarrow \vec{x}\vec{\mu} \leq t} \chi)$ becomes $(\text{ifp}_{X \leftarrow \vec{x}\vec{\mu} \leq q_t(m)} \vec{\mu} \leq t \wedge \chi)$
- $\psi = \text{rk}(\vec{\omega}; \vec{x}\vec{\mu} \leq t, \vec{y}\vec{v} \leq t) \chi$ becomes $\text{rk}(\vec{\omega}; \vec{x}\vec{\mu} \leq q_t(m), \vec{y}\vec{v} \leq q_t(m)) \vec{\mu} \leq t \wedge \vec{v} \leq t \wedge \chi$

It is readily verified that the modifications produce equivalent subformulas. In particular, the rank of the matrices considered is not increased through our addition of zero rows and vertices. Let us denote the sentence resulting from modifying φ by $\varphi_m^{(1)}$.

2. Eliminate non-atomic numeric terms. The goal of this step is to transform $\varphi_m^{(1)}$ into an equivalent formula $\varphi_m^{(2)}$ in which $+$ and \cdot do not occur. At the same time, we also replace rank operators with rank quantifiers which may, however, still bind number variables. We do this by inductively replacing all terms t in $\varphi_m^{(1)}$ with formulas $\psi_t(\mu)$ in one number variable with the property that $k = t^A$ if and only if $\psi_t(k)$ holds. As a result, the only numeric terms in $\varphi_m^{(2)}$ will be number variables and the constants 0, 1, and all of these only occur at quantification or in atomic terms of the form $v = v$ or $v < v$.

The atomic numeric terms 0, 1, and variables v become formulas $\psi_0(\mu) := (\mu = 0)$, $\psi_1(\mu) := (\mu = 1)$, and $\psi_v(\mu) := (\mu = v)$. By [EF99, Lemma 7.3.11], there are DTC-formulas (and therefore FP-formulas) $\varphi_+(x, y, z)$ and $\varphi_\bullet(x, y, z)$ holding on ordered structures if and only if $x + y = z$, respectively $x \cdot y = z$, with respect to the positions of x, y, z in the structure's ordering. We replace a term $t = s_1 + s_2$ with $\psi_t(\mu) := \exists v \leq q_{s_1} \exists v \leq q_{s_2} \psi_{s_1}(v) \wedge \psi_{s_2}(v) \wedge \varphi_+(v, v, \mu)$, where φ_+ of course uses the ordering over the number sort. Using φ_\bullet instead of φ_+ gives the formula ψ_t for terms of the form $t = s_1 \cdot s_2$.

Terms t of the form $\text{rk}(\vec{\omega}; \vec{x}\vec{\mu} \leq z, \vec{y}\vec{v} \leq z) \eta$ become

$$\psi_t(\mu) := \bigvee_{i \leq q_t(m)} \bigvee_{j \leq q_\omega(m)} (\mu = i \wedge \psi_\omega(j) \wedge \text{rk}_j^{\vec{\omega}}(\vec{x}\vec{\mu} \leq z, \vec{y}\vec{v} \leq z) [\eta = 1 \bmod j, \dots, \eta = j - 1 \bmod j]),$$

where $\mu = i$ is short for the FO^3 -formula $\rho_{i+1}(\mu)$ from Lemma 2.4.2 encoding that μ is the $(i+1)^{\text{th}}$ element of $<$,¹⁶ $\psi_\omega(j)$ is short for $\exists y \leq q_\omega(m) y = j \wedge \psi_\omega(y)$, and the formulas $\eta = a \bmod j$ are short for DTC-formulas encoding that there is a number $z \leq q_\eta(m)$ so that $\psi_\eta(j \cdot z + a)$ holds. Unlike the rank quantifiers we defined above, the quantifiers used here may still bind number variables. The semantics should be clear. The number variables will be removed in the next step.

¹⁶The “+1” takes care of the first element on the number sort being 0, not 1. The formulas $\rho_i(\mu)$ use quantification abundantly which has to be bounded by the appropriate bound for μ .

5 Stronger Logics for Polynomial Time

Finally, consider formulas built from numeric terms. Formulas of the form $t_1 = t_2$ are replaced with $\exists \mu \leq q_{t_1}(m) \psi_{t_1}(\mu) \wedge \psi_{t_2}(\mu)$. Formulas of the form $t_1 < t_2$ are replaced with $\exists \mu \leq q_{t_1}(m) \exists v \leq q_{t_2}(m) \mu < v \wedge \psi_{t_1}(\mu) \wedge \psi_{t_2}(v)$. This concludes step 2 and the construction of $\varphi_m^{(2)}$.

3. Replace mixed-type rank operators. In this step we replace subformulas of $\varphi_m^{(2)}$ of the form $\text{rk}_p^{=k}(\vec{x}\vec{\mu} \leq z, \vec{y}\vec{v} \leq z) \vec{\psi}$ for which $|\vec{\mu}| \neq 0$ or $|\vec{v}| \neq 0$. We make use of the fact that the number variables $\vec{\mu}$ and \vec{v} only depend on the size m of the structure so that we can directly encode their contribution into the Lindström quantifier.

For $\vec{a} \in \mathbb{N}_0^{|\vec{\mu}|}$ and $\vec{b} \in \mathbb{N}_0^{|\vec{v}|}$, let us denote by $\vec{\psi}_{\vec{a}, \vec{b}}$ the formulas $\psi \in \vec{\psi}$ each replaced by

$$\exists \vec{\mu} \vec{v} \leq z \left(\bigwedge_{a_i \in \vec{a}} \mu_i = a_i \wedge \bigwedge_{b_i \in \vec{b}} v_i = b_i \quad \wedge \quad \psi \right),$$

where as before the formulas $v = c$ stand for the FO-formulas $\rho_{c+1}(v)$ from Lemma 2.4.2 encoding that v is the $(c+1)^{\text{th}}$ element in the ordering on the number sort. Let $M_{\vec{\psi}}$ denote the matrix defined by $\vec{\psi}$ with respect to $\vec{x}\vec{\mu}$ and $\vec{y}\vec{v}$. For each $\vec{a}, \vec{b} \leq z$, $\vec{\psi}_{\vec{a}, \vec{b}}$ defines the submatrix of $M_{\vec{\psi}}$ with respect to \vec{x} and \vec{y} where $\vec{\mu} = \vec{a}$ and $\vec{v} = \vec{b}$ are kept fixed. The Lindström quantifiers we define will simply be supplied with all these submatrices.

Recall that the bound z on the number variables is ensured by the first step to be an absolute number which depends polynomially on m . Let $\vec{a}_1, \vec{a}_2, \dots, \vec{a}_k$ and $\vec{b}_1, \vec{b}_2, \dots, \vec{b}_\ell$ be the lexicographically ordered lists of the elements in $[0, z]^{|\vec{\mu}|}$ and $[0, z]^{|\vec{v}|}$, respectively. Let the Lindström quantifier $R_p^{=k; z: |\vec{\mu}|, |\vec{v}|}$ bind $p \cdot k \cdot \ell = p(z+1)^{|\vec{\mu}|+|\vec{v}|}$ formulas. The semantics of

$$R_p^{=k; z: |\vec{\mu}|, |\vec{v}|}(\vec{x}, \vec{y}) \quad \vec{\psi}_{\vec{a}_1, \vec{b}_1} \vec{\psi}_{\vec{a}_2, \vec{b}_2} \cdots \vec{\psi}_{\vec{a}_1, \vec{b}_\ell} \vec{\psi}_{\vec{a}_2, \vec{b}_1} \cdots \vec{\psi}_{\vec{a}_k, \vec{b}_\ell} \quad (\clubsuit)$$

is defined to be the same as $\text{rk}_p^{=k}(\vec{x}\vec{\mu} \leq z, \vec{y}\vec{v} \leq z) \vec{\psi}$. In (\clubsuit) , we have refrained from writing out the variables bound by $R_p^{=k; z: |\vec{\mu}|, |\vec{v}|}$ for every single formula. Instead, by writing (\vec{x}, \vec{y}) we indicate that the variables bound in every formula are all the same. This is not only natural since $\vec{\psi}$ defines $M_{\vec{\psi}}$ with respect to these variables, but it is also necessary since we may not make the number of variables depend on the size of the structure.

We note that $R_p^{=k; z: |\vec{\mu}|, |\vec{v}|}$ is a well-defined Lindström quantifier of arity $|\vec{x}| + |\vec{y}| < n$ since there is no limit on the number of formulas such a quantifier may bind, so $R_p^{=k; z: |\vec{\mu}|, |\vec{v}|} \in \mathcal{Q}_n$. Let us denote the result of replacing in $\varphi_m^{(2)}$ all mixed-type rank quantifiers with the respective quantifier as in (\clubsuit) by $\varphi_m^{(3)}$.

4. Eliminate fixed-point operators. In [EF99, Proposition 8.4.1] it is shown that for any formula $\psi(\vec{x}, X) \in \text{FO}^k$ with free second-order variable X of arity $|\vec{x}|$, there are formulas $\psi^s(\vec{x}) \in \text{FO}^{k+|\vec{x}|}$ defining the s^{th} stage of the fixed-point iteration of $\text{ifp}_{X \leftarrow \vec{x}} \psi$ for each $s \in \mathbb{N}$. Note that the number of variables remains constant irrespective of the stage s we want to define. One glance at the proof of [EF99, Proposition 8.4.1] makes it clear that such formulas ψ^s can be equally well defined when ψ contains arbitrary Lindström quantifiers.

In $\varphi_m^{(3)}$, replace each subformula of the form $(\text{ifp}_{X \leftarrow \vec{x}\vec{\mu} \leq z} \psi) \vec{y}$ with $\psi^s(\vec{y})$, where $s = m^{|\vec{x}|} z^{|\vec{\mu}|}$. In structures of size m , the inflationary fixed-point is always reached after s steps; therefore, the two formulas are equivalent for structures of size m . Let us denote the resulting formula by $\varphi_m^{(4)}$.

5. Eliminate numeric terms. The final step is to eliminate atomic formulas containing number variables and constants 0, 1. Let $\psi(\vec{x}, \vec{\mu})$ be a subformula of $\varphi_m^{(4)}$ with free number variables $\vec{\mu}$. Each $\mu_i \in \vec{\mu}$ is bounded by an absolute value m_i in the place where it is quantified over. For each $\vec{s} \leq \vec{m}$, we inductively construct a formula $\psi_{\vec{s}}(\vec{x})$ equivalent to $\psi(\vec{x}, \vec{s})$ on structures of size m . The relevant cases are the following:

- If $\psi(\vec{x}, \vec{\mu}) = \mu_i < \mu_j$ then $\psi_s := \begin{cases} \text{true} & \text{if } s_i < s_j \\ \text{false} & \text{otherwise.} \end{cases}$

Likewise for $\psi(\vec{x}, \vec{\mu}) = (\mu_i = \mu_j)$.

- If $\psi(\vec{x}, \vec{\mu}) = \exists v \leq z \chi(\vec{x}, \vec{\mu} v)$ then $\psi_s(\vec{x}) := \bigvee_{s \leq z} \chi_{ss}(\vec{x})$.

Since $\varphi_m^{(4)}$ is a sentence, we obtain the sentence $\varphi_m := (\varphi_n^{(4)})_\emptyset$ after inductively modifying all subformulas in this way. φ_m does not contain numeric terms or fixed-point operators, and all Lindström quantifiers are from \mathcal{Q}_n . Thus, $\varphi_m \in \text{FO}^k(\mathcal{Q}_n)$ and it is clear from the proof that k does not depend on m . As φ_m is equivalent to φ on structures of size m , the transformation is complete. \square

Remark 5.6.8. Step 4 in the proof of Proposition 5.6.7 is not limited to inflationary fixed-point operators, but works similarly for partial fixed-point operators with an exponential bound on the stage we need to consider and a check that the iteration really converges. It follows that partial fixed-point logic with rank operators of arity at most n also embeds into $L_{\infty\omega}^\omega(\mathcal{Q}_n)$.

Remark 5.6.9. The transformation presented here can be considered prototypical for embedding fixed-point logics with *general numerical operators of arity $\leq n$* into $L_{\infty\omega}^\omega(\mathcal{Q}_n)$. An example for a numerical operator other than our rank operators would be an operator returning the chromatic number of a graph. Such operators can be turned into Lindström quantifiers in the same way as we did with rank operators here. In general, these operators may not allow the “extension by zeros” as we did for our rank operators in Step 1. This can be circumvented, however, by combining Steps 1 and 2 and using a disjunction in order get the precise value of the number variables’ bounds right.

5.6.3 The construction of separating queries

In this section we lay the groundwork for proving Theorem 5.6.1 by constructing structures that can easily be recognized using rank quantifiers of a certain arity but which cannot be recognized using any Lindström quantifiers of lower arities. For every prime p , this will show that increasing the arity of rk_p always increases its expressiveness. In other words, our rank operators have a strict arity hierarchy. The construction given here is a generalization of a construction by Hella [Hel96] and for $p = 2$, our construction is the same as Hella’s. The generalization is based on an idea of Martin Grohe.

The construction of $D_n^p(G, S)$ Fix $n \geq 3$ and a prime p . Let $G = (V, E^G, <^G)$ be a finite undirected graph which is connected, regular of degree $n \geq 3$, and $<^G$ is a strict linear order of G ’s vertices. For any labeling of the vertices $S : V \rightarrow \text{GF}_p$, we define the structure $D_n^p(G, S)$ over vocabulary $\tau_p = (R_0, \dots, R_{p-1}, E, <)$ with universe $C_G^p = \{(u, e, i) \in V \times E^G \times \text{GF}_p \mid u \in e\}$. Figure 5.2 illustrates the definition. Let $\prec^{D_n^p(G, S)}$ be the strict weak order on C_G^p defined by

$$(u, e, i) \prec (v, f, j) \iff u <^G v \quad \text{or} \quad u = v \text{ and } e <_{\text{lex}} f,$$

where $<_{\text{lex}}$ is the lexicographic order on sets induced by $<^G$. Notice that elements (u, e, i) and (v, f, j) are incomparable if and only if $u = v$ and $e = f$. Hence, $\prec^{D_n^p(G, S)}$ has width p . $E^{D_n^p(G, S)}$ is the simple undirected edge relation on C_G^p given by

$$E^{D_n^p(G, S)} := \{((u, e, i), (v, e, -i)) \mid e = uv \text{ and } i \in \text{GF}_p\}.$$

5 Stronger Logics for Polynomial Time

Finally, the relation $R_j^{D_n^p(G,S)}$ shall consist of precisely those n -tuples $((u, e_1, i_1), \dots, (u, e_n, i_n))$ which are in strict linear order with respect to $\prec^{D_n^p(G,S)}$ and for which

$$\sum_{k=1}^n i_k = j + S(u) \pmod{p}.$$

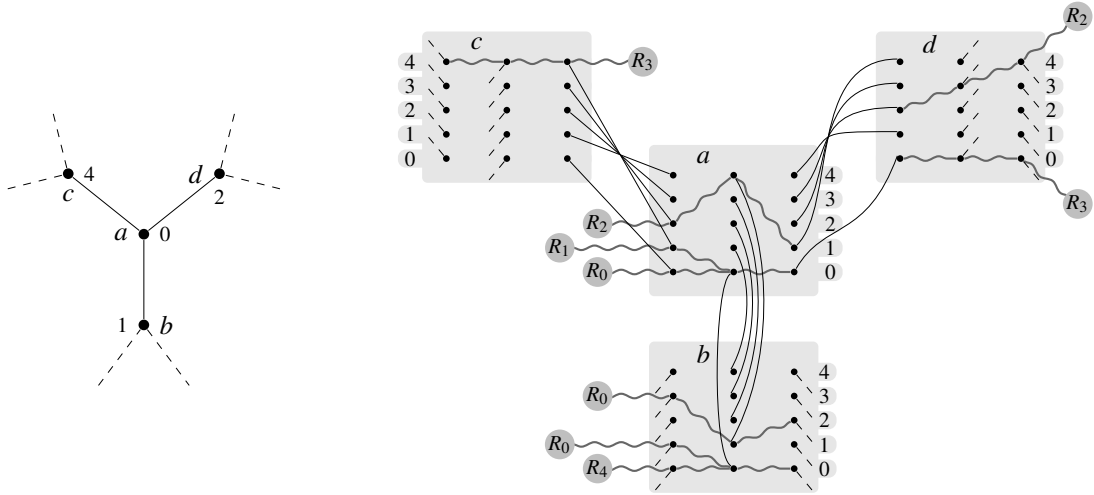


Figure 5.2: Part of a 3-regular graph G with label $S : V \rightarrow \text{GF}_5$ and corresponding structure $D_3(G, S)$ with exemplary illustrations of relations $R_i, i \in [0, 4]$.

Shifting vertex labels Unlike in Cai, Fürer, and Immerman’s original construction of the CFI graphs [CFI92], this construction does not “twist” the actual edges. Instead, the twists are encoded in the relations $R_j^{D_n^p(G,S)}$ for $j \in \text{GF}_p$. Just as in the case of the CFI graphs, however, it does not matter where in the graph such twists occur, since they can be moved freely along edges of G . More precisely, the following results show that the isomorphism class of $D_n^p(G, S)$ does not depend on the precise definition of S , but only on the sum $\sum_{u \in V} S(u)$.

Lemma 5.6.10. *Suppose $uv \in E^G$ and $S, S' : V \rightarrow \text{GF}_p$ are defined such that*

$$S(x) - S'(x) = \begin{cases} 1 & \text{for } x = u, \\ -1 & \text{for } x = v, \\ 0 & \text{else.} \end{cases}$$

Then $D_n^p(G, S) \cong D_n^p(G, S')$.

Proof. Let $e = uv \in E^G$. Define $\psi : D_n^p(G, S) \rightarrow D_n^p(G, S')$ as $(u, e, i) \mapsto (u, e, i - 1)$ and $(v, e, i) \mapsto (v, e, i + 1)$ for all $i \in \text{GF}_p$, and as the identity everywhere else. We claim that ψ is an isomorphism between $D_n^p(G, S)$ and $D_n^p(G, S')$. It is clear that ψ preserves \prec , and it is immediately seen that ψ also preserves E .

Now suppose that $((u, e_1, i_1), \dots, (u, e_\ell, i_\ell), \dots, (u, e_n, i_n)) \in R_j^{D_n^p(G,S)}$. Then

$$\begin{aligned} i_1 + \dots + i_\ell + \dots + i_n &= j + S(u) \pmod{p} \\ \iff i_1 + \dots + (i_\ell - 1) + \dots + i_n &= j + S'(u) \pmod{p}, \end{aligned}$$

and hence the above is equivalent to $(\Psi(u, e_1, i_1), \dots, \Psi(u, e, i_\ell), \dots, \Psi(u, e_n, i_n)) \in R_j^{D_n^p(G, S')}$. Similarly, $((v, e_1, i_1), \dots, (v, e, i_m), \dots, (v, e_n, i_n)) \in R_j^{D_n^p(G, S)}$ holds if and only if $(\Psi(v, e_1, i_1), \dots, \Psi(v, e, i_m), \dots, \Psi(v, e_n, i_n)) \in R_j^{D_n^p(G, S')}$ since

$$i_1 + \dots + i_m + \dots + i_n = j + S(v) = j + S'(v) - 1 \pmod{p}.$$

Thus, Ψ is an isomorphism. \square

Using Lemma 5.6.10, we can now *push* the vertex values given by a labeling S from one vertex to the next. In particular, if we pick out a single vertex $u \in V$, we can *shift* all labels to u : let v be any other vertex and let P be any path from v to u (recall that G is connected). By pushing the value $S(v)$ from vertex to vertex along P , it becomes clear that $D_n^p(G, S)$ is in fact isomorphic to $D_n^p(G, S')$, where $S'(u) = S(u) + S(v)$, $S'(v) = 0$, and $S'(x) = S(x)$ for all other vertices x . Now let S_u be the vertex labeling with $S_u(u) = \sum_{x \in V} S(x)$ and $S_u(x) = 0$ for all other x . By repeating the above argument, we obtain that $D_n^p(G, S) \cong D_n^p(G, S_u)$. What is more, we could have picked any other vertex u' in place of u and shifted all values there instead, showing that $D_n^p(G, S_u) \cong D_n^p(G, S_{u'})$ for any two vertices $u, u' \in V$. For any $k \in \text{GF}_p$, we can therefore write $D_n^p(G, k)$ to identify the graph $D_n^p(G, S_u^k)$ up to isomorphism, where u is any vertex of G , $S_u^k(u) = k$, and $S_u^k(x) = 0$ for all vertices $x \neq u$.

A first linear system for $D_n^p(G, S)$ In order to be useful, we also need a converse assertion to Lemma 5.6.10, namely that when $\sum_{u \in V} S(u) \neq \sum_{u \in V} S'(u)$, then $D_n^p(G, S)$ is not isomorphic to $D_n^p(G, S')$. We show this by directly constructing an isomorphism invariant linear system of equations \mathfrak{S}_n from a structure $D_n^p(G, S)$ with the property that \mathfrak{S}_n is solvable over some structure $D_n^p(G, S')$ if and only if $\sum_{u \in V} S(u) = \sum_{u \in V} S'(u)$. Apart from showing that $D_n^p(G, k)$ and $D_n^p(G, k')$ are only isomorphic if $k = k'$, the system will also make it clear that these structures can be distinguished using rank operators.

Let some structure $D_n^p(G, S)$ with universe C_G^p and some $k \in \text{GF}_p$ be given, let $u \in C_G^p$, and let the system $\mathfrak{S}_n^{u, k}$ consist of variables x_a for each element $a \in C_G^p$ and the following equations

- *edge equations*: for each edge $e = ab \in E^{D_n^p(G, S)}$: $x_a + x_b = 0$
- for any $i \in \text{GF}_p$ and each n -tuple $a_1 \dots a_n \in R_i$:
 - if some a_j is incomparable to u : $x_{a_1} + \dots + x_{a_n} = i + k$
 - if all a_j are comparable to u : $x_{a_1} + \dots + x_{a_n} = i$

Observe at this point that $\mathfrak{S}_n^{u, k}$ is directly first-order definable over $D_n^p(G, S)$ with u and k as free variables. The system is not yet optimal in the sense that its first-order definition would only yield a matrix of arity $n + 1$ (n row variables and one column variable), and defining solvability of the system as in Proposition 5.2.6 would increment the matrix's arity yet again. In order to obtain the strict hierarchy in Theorem 5.6.1, however, we must limit our matrices to arity n . We will show later on how to obtain an equivalent system which can be defined in this way.

Lemma 5.6.11. *The system $\mathfrak{S}_n^{u, k}$ is solvable over GF_p for any element u in the universe of $D_n^p(G, S)$ if and only if $\sum_{v \in V} S(v) = k \pmod{p}$.*

Proof. To see that the system is unsolvable for the wrong values of k , add up all equations of the form $x_{(v, e_1, 0)} + \dots + x_{(v, e_n, 0)} = i$. Then cancel all of the variables on the left hand side using

5 Stronger Logics for Polynomial Time

the edge equations of the form $x_{(u,uv,0)} + x_{(v,uv,0)} = 0$. By the construction of $D_n^p(G, S)$, this linear combination of equations leaves us with

$$0 = k - \sum_{v \in V} S(v),$$

which shows that $\mathfrak{S}_n^{u,k}$ is not solvable for $k \neq \sum_{v \in V} S(v) \pmod p$.

For $k = \sum_{v \in V} S(v)$, however, consider the following assignment to the variables of the system: for $(v, e, i) \in C_G^p$ set $x_{(v,e,i)} = i$. Instead of $D_n^p(G, S)$, let us consider the isomorphic structure $D_n^p(G, S_{u'}^k)$, where we choose $u' \in V$ to be the projection of $u \in C_G^p$ onto its first component. We will show that the above assignment solves $\mathfrak{S}_n^{u,k}$ over $D_n^p(G, S_{u'}^k)$. Since $\mathfrak{S}_n^{u,k}$ is first-order definable from $D_n^p(G, S) \cong D_n^p(G, S_{u'}^k)$ and $u \in C_G^p$, it follows that $\mathfrak{S}_n^{u,k}$ is also solvable over $D_n^p(G, S)$, and this is true for any $u \in C_G^p$.

Certainly, the edge equations are satisfied by the above assignment. If $a_1 \dots a_n \in R_i$ and all a_j are comparable to u (so they are of the form $a_j = (w, \dots)$ with $w \neq u'$), then it is immediately clear from the construction of $D_n^p(G, S_{u'}^k)$ that the equation $x_{a_1} + \dots + x_{a_n} = i$ is satisfied. Lastly, if $(u', e_1, i_1) \dots (u', e_n, i_n) \in R_i$, then by the construction of $D_n^p(G, S_{u'}^k)$, we have $\sum_{j \in [n]} i_j = i + k$, showing that the assignment also satisfies the equation $x_{(u',e_1,i_1)} + \dots + x_{(u',e_n,i_n)} = i + k$. \square

Corollary 5.6.12. *If $k \neq k'$, then $D_n^p(G, k) \not\cong D_n^p(G, k')$.*

Proof. Since $\mathfrak{S}_n^{u,k}$ is first-order definable, it is definable in FO+rk whether a structure $D_n^p(G, k')$ solves $\mathfrak{S}_n^{u,k}$ for all elements $u \in C_G^p$. By Lemma 5.6.11, this is true if and only if $k = k'$. Since FO+rk-sentences are invariant under isomorphism, $D_n^p(G, k)$ and $D_n^p(G, k')$ cannot be isomorphic when $k \neq k'$. \square

Defining $D_n^p(G, 0)$ with n variables Before we show that non-isomorphic structures $D_n^p(G, 0)$ and $D_n^p(G, 1)$ cannot be distinguished in $L_{\infty\omega}^{\omega}(\mathcal{Q}_{n-1})$ in general, we first present a proof that this can be done in FO+rk $_p^{[n]}$. For this, we modify the system $\mathfrak{S}_n^{u,0}$ into an equivalent system \mathfrak{T}_n with the same variables as $\mathfrak{S}_n^{u,0}$ and equations:

- for each edge $e = ab \in E^{D_n^p(G, S)}$: $x_a + x_b = 0$,
- for each n -tuple $a_1 \dots a_n \in R_0$: $x_{a_1} + \dots + x_{a_n} = 0$,
- for incomparable elements $(v, e, i), (v, e, i+1) \in C_G^p$: $x_{(v,e,i)} - x_{(v,e,i+1)} = 1$.

The idea is that the equations that correspond to tuples from R_0 can be defined on rows indexed by only $n-1$ variables, while the new type of equations encodes a cyclic ordering of variable assignments in each block of incomparable elements. At this point, even though $D_n^p(G, S)$ was constructed over elements from $C_G^p = V \times E \times \text{GF}_p$, it is not clear that the new type of equations can be defined in FO from the structure $D_n^p(G, S)$, since for elements $c \in C_G^p$ we do not have access to the GF_p -component of c . However, elements x, y of the form $x = (v, e, i), y = (v, e, i+1)$ can be identified from the relations R_i as follows. Consider any tuple $\vec{a} = a_1 \dots a_n \in R_i, i \in \text{GF}_p$, which contains x , i.e. with $a_j = x$ for some j . Let \vec{b} be the tuple \vec{a} with the j^{th} element replaced with y . Then x and y are of the above form if and only if $\vec{b} \in R_{i+1}$. Hence, the system \mathfrak{T}_n is first-order definable over any structure $D_n^p(G, S)$.

Lemma 5.6.13. *Over any structure $D_n^p(G, S)$, $\mathfrak{S}_n^{u,0}$ is solvable if and only if \mathfrak{T}_n is solvable.*

Proof. In fact, it is easy to see that the solution spaces are equal. Notice that the first two types of equations in \mathfrak{T}_n are also part of $\mathfrak{S}_n^{u,0}$, so we only have to argue why $\mathfrak{S}_n^{u,0}$'s equations for R_i with $i \neq 0$ can be replaced with the third equation type in \mathfrak{T}_n . But this is simply a consequence of the argument in the preceding paragraph and the fact that $\mathfrak{S}_n^{u,0}$'s equations mimic the construction of $D_n^p(G, S)$. \square

\mathfrak{T}_n is definable with only $n - 1$ row variables and 1 column variable. To see this, notice that whenever $\vec{a} = a_1 \dots a_n$ is a tuple in R_0 and $x \in C_G^p$ is any element, then $a_1 \dots a_{n-1}x \in R_0$ implies that $x = a_n$. Thus, the first $n - 1$ elements suffice to identify the rows that should carry the equations of \mathfrak{T}_n 's second sort. The equations of the first and third sort (edge equations and equations for incomparable elements) can then be defined on rows of the form $xy \dots y$ without interfering with each other. Finally, one column variable suffices since \mathfrak{T}_n has only one variable per element from C_G^p , and it is clear how to define the matrix and solution vector corresponding to \mathfrak{T}_n in $\text{FO}+\text{rk}_p^{[n]}$.

Finally, we need to show how to define solvability of \mathfrak{T}_n with only n variables. For this, we cannot use the forthright approach of Proposition 5.2.6 since it needs an additional variable in order to make space for the solution vector inside the matrix, thereby requiring rank operators of arity $n + 1$. We cannot apply the more specific Proposition 5.2.8 either, since it requires the number of row variables to be bounded by the number of column variables. As mentioned above, the matrix of \mathfrak{T}_n is defined with $n - 1$ row variables and only one column variable. Instead, we show that the system's matrix does not have full column rank - despite the number of rows vastly exceeding the number of columns. By Lemma 5.2.7, it then suffices to check that adding \mathfrak{T}_n 's solution vector to each of the matrix's columns never increases the matrix's rank, thereby only requiring rank quantifiers of arity n .

Lemma 5.6.14. *Let G be an n -regular graph with $n \geq 2$ and let A be the matrix corresponding to the system \mathfrak{T}_n defined above. The columns of A are not linearly independent.*

Proof. Since G is a finite n -regular with $n \geq 2$, G contains a cycle. Let H be such a cycle and consider any of the two possible cyclic orientations of H . Let I be the set consisting of all elements $(u, e, i) \in C_G^p$ for which u and e lie in H and where e is coming in to u ; and similarly, let O contain all elements $(u, e, i) \in C_G^p$ for which u, e lie in H and where e is going out of u . It is now straightforward to verify that on every row of A , the sum of the entries in columns indexed by I is equal to the sum of the entries in columns indexed by O . Thus, A does not have full column rank. \square

Lemma 5.6.15. *There is an $\text{FO}+\text{rk}_p^{[n]}$ -sentence which holds for all structures $D_n^p(G, 0)$ and which is false for all structures $D_n^p(G, k)$ with $k \neq 0$.* \square

5.6.4 Proving the queries to be undefinable for low quantifier arity

A cops-and-robber game for G Our next aim is to establish that $L_{\infty\omega}^\omega(Q_{n-1})$ cannot define a query separating non-isomorphic structures $D_n^p(G, k)$ and $D_n^p(G, k')$ in general. Since the structures can be distinguished using rank operators of higher arity, this particular isomorphism query reveals a fundamental weakness of Lindström operators of bounded arity. In order to do this, we adapt the $(n - 1)$ -bijective k -pebble game from Section 5.6.1 to the specific setting of the structures $D_n^p(G, S)$. In analogy to Hella [Hel96], this cops-and-robber type game is played on the source graph G and will capture the expressiveness of bounded variable infinitary logic with bounded-arity Lindström quantifiers on structures $D_n^p(G, S)$. Interestingly, even though the structures $D_n^p(G, S)$ derived from G are real generalizations of the structures considered by Hella, the cops-and-robber game is precisely the same. It thus appears that the intrinsic reason why Lindström quantifiers of bounded arity are of limited expressiveness is not related to any specific prime field characteristic, but this limitation can be encoded in structures $D_n^p(G, S)$ over any arbitrary prime. The intrinsic limitation will surface in the proof of Lemma 5.6.16 below as the inability

5 Stronger Logics for Polynomial Time

of $L_{\infty}^{\omega}(\mathcal{Q}_{n-1})$ to discover so-called *twists*, which are the points at which bijections between non-isomorphic structures $D_n^p(G, k)$ and $D_n^p(G, k')$ fail to be an isomorphism, and which can be moved about the graph freely, independent of the prime p that the structures $D_n^p(G, S)$ were constructed for (see Lemma 5.6.10).

The cops-and-robber game $CR_n^k(G)$ is played by two players: spoiler, who controls k cops (pebbles) which he can place on the edges of the graph G , and duplicator, who controls one robber (also a pebble) which is always placed on some vertex of G . In the beginning of the game no cops are present in G and the robber is placed on some vertex of G . In each round, duplicator starts by moving the robber along some path in G to a new vertex, and this path must not pass through an edge with a cop on it. Spoiler then answers by picking up at most n of his cops and placing them on arbitrary edges of G . Spoiler wins if after some round, his cops block all the edges adjacent to the location of the robber. Duplicator wins infinite plays. The following lemma and its proof are practically identical to Lemma 8.5 in [Hel96]; however, as the construction of the structures $D_n^p(G, k)$ is more general, we restate the proof here as well.

Lemma 5.6.16. *Let p be prime. If duplicator has a winning strategy in the game $CR_{n-1}^k(G)$, then she has a winning strategy in the $(n-1)$ -bijjective k -pebble game played on $\langle D_n^p(G, x), \emptyset \rangle$ and $\langle D_n^p(G, y), \emptyset \rangle$, where G is regular of degree $n \geq 3$, the structures D are constructed with respect to p , and x, y are arbitrary values from GF_p .*

Proof. We need to describe a winning strategy for duplicator in the $(n-1)$ -bijjective k -pebble game which is based on a winning strategy in the corresponding cops-and-robber game. Let $\mathcal{X} = D_n^p(G, x)$ and $\mathcal{Y} = D_n^p(G, y)$, and suppose $x \neq y$, since otherwise $\mathcal{X} \cong \mathcal{Y}$ and duplicator has a very obvious winning strategy. Duplicator's strategy will be to always maintain a bijection $f : U(\mathcal{X}) \rightarrow U(\mathcal{Y})$ which is an isomorphism with respect to \prec and E and which is an isomorphism with respect to the relations R_i everywhere, except for a block $\{u\} \times E(u) \times \text{GF}_p$ for some vertex $u \in V$.¹⁷ In order to see that such bijections always exist for any vertex u , recall that \mathcal{X} and \mathcal{Y} are isomorphic to $D_n^p(G, S_u^x)$ and $D_n^p(G, S_u^y)$, respectively, where S_u^x and S_u^y are value labelings of G that only place non-zero values x and y on vertex u . $D_n^p(G, S_u^x)$ and $D_n^p(G, S_u^y)$ are constructed over the exact same universe, and the identity map between $D_n^p(G, S_u^x)$ and $D_n^p(G, S_u^y)$ gives us the desired bijection which preserves all relations except for the relations R_i at vertex u . We say that such a bijection f *hides the twist at u* .

Next, we observe that if f hides the twist at u and P is a path from u to some other vertex v in G , then concatenating f with the isomorphisms described in the proof of Lemma 5.6.10 in a suitable way yields a bijection f' which hides the twist at v and which is identical to f on all vertices $(w, e, i) \in C_G^p$ for which e does not lie on P . Duplicator's winning strategy for the $(n-1)$ -bijjective k -pebble game on $\langle D_n^p(G, x), \emptyset \rangle$ and $\langle D_n^p(G, y), \emptyset \rangle$ is now as follows: Duplicator considers the game $CR_{n-1}^k(G)$ in parallel, where every pebble on a vertex $(w, e, i) \in C_G^p$ corresponds to a cop on edge e in G . Whenever the robber is placed on a vertex $u \in V$, the bijection f maintained by duplicator hides the twist at u . At the beginning of a round, duplicator considers the path along which the robber moves in the cops-and-robber game to a (possibly new) vertex u and updates her bijection f accordingly. Then she reveals her bijection f to spoiler, whose pebbles do not detect any change from the last given bijection since the robber's path avoided all the cops. Also notice that u does not have a cop on any of its n adjacent edges, since otherwise spoiler would be able to trap the robber by moving $n-1$ cops to all the remaining adjacent edges. Now spoiler moves at most $n-1$ pebbles, and this is too little to catch f 's twist at u in one move. Duplicator updates the position of the cops in the parallel game according to spoiler's pebble placement. This completes the round and by using her winning strategy for the cops-and-robber game, duplicator will be able to move f 's twist again to a safe vertex in the next round, continuing indefinitely in this way. \square

¹⁷Recall that $E(u)$ denotes the set of edges adjacent to u .

Finishing up the proof We are now ready to finish the proof of Theorem 5.6.1.

Proof of Theorem 5.6.1. Given that the cops-and-robber game above is identical to Hella's in [Hel96], we can now use the same graphs for constructing our separating query. Hella shows that for every $n \in \mathbb{N}$ and every $k > n$, there exists a finite connected graph G_{n+1}^k of degree $n+1$ so that duplicator has a winning strategy in the game $CR_n^k(G)$ [Hel96, Theorem 8.6]. For any fixed prime p , by Lemma 5.6.15, there is an $\text{FO}+\text{rk}_p^{[n+1]}$ -sentence φ which holds for all graphs $D_{n+1}(G_{n+1}^k, 0)$ constructed with respect to p , but not for any $D_{n+1}(G_{n+1}^k, i)$ for $i \neq 0$. By Lemma 5.6.16 and Theorem 5.6.6, the query φ is not $L_{\infty\omega}^\omega(Q_n)$ -definable. \square

5.7 Discussion

In this chapter, we have introduced the rank logics $\text{FO}+\text{rk}$ and $\text{FP}+\text{rk}$ with rank operators over prime fields and their restrictions $\text{FO}+\text{rk}_p$, $\text{FP}+\text{rk}_p$ for fixed primes $p \in \mathbb{N}$. We justified the purpose of these logics by showing that $\text{FO}+\text{rk}_2$ already expresses the query Cai, Fürer, and Immerman used in [CFI92] to separate $\text{FP}+\text{C}$ from PTIME , establishing that $\text{FP}+\text{rk}$ is strictly more expressive than $\text{FP}+\text{C}$. We related the expressiveness of rank logics to many established logics in finite model theory, showing that $\text{FP}+\text{C} \preceq \text{FP}+\text{rk}_p$ and $\text{STC}+\text{C} \preceq \text{FO}+\text{rk}_p$ for all primes p as well as $\text{TC}+\text{C} \preceq \text{FO}+\text{rk}$. Furthermore, we showed that most basic operations in linear algebra other than matrix rank may already be defined in $\text{FP}+\text{C}$. By proving that $\text{FO}+\text{rk}_p$ captures $\text{MOD}_p L$ for prime p and that $\text{FO}+\text{rk}$ captures $\#\text{LH}$ on ordered structures, we also uncovered faithful correspondences between rank logics and established classes in complexity theory. Finally, our theorem that the expressiveness of rank operators increases when their arity is increased is a sharp contrast to simple counting operators. It means that rank operators satisfy the formal requirement that an operator needs to satisfy if it aspires to solve the PTIME -capturing problem. This suggests that rank operators are not merely a quick fix to $\text{FP}+\text{C}$'s shortcomings, but that a deeper understanding of these operators is a necessary precondition for further progress towards the capturing of polynomial time.

By this point, the sceptical mind may wonder what it takes to separate $\text{FP}+\text{rk}$ from PTIME , while the optimist may ask what steps could lead to a positive answer to this question. We believe that both directions of investigation are important as they will both increase the understanding of the power of rank operators and thereby of the native building blocks of PTIME computations.

Let us begin with the negative approach. Separating $\text{FP}+\text{rk}$ from PTIME or any other complexity class requires a good characterization of the limits of $\text{FP}+\text{rk}$'s expressiveness. We have outlined one possibility in Section 5.6.2 by embedding rank logics with arity- n rank operators into $L_{\infty\omega}^\omega(Q_n)$, infinitary logic with all Lindström quantifiers of arity at most n . This logic has a good game characterization (Hella's n -bijective k -pebble game), which enabled us to prove the strict arity hierarchy of rank operators. For separating $\text{FP}+\text{rk}$ from complexity classes, though, $L_{\infty\omega}^\omega(Q_n)$ is too coarse a tool since all queries, in particular all PTIME -queries, on structures of arity $\leq n$ are trivially definable in it.

The situation looks different if we only augment $L_{\infty\omega}^\omega$ with those generalized quantifiers \mathcal{R} which are really necessary for embedding $\text{FP}+\text{rk}$ (see Section 5.6.2). In the case of $\text{FP}+\text{C}$, the logic can be embedded into $L_{\infty\omega}^\omega$ augmented with (unary) counting quantifiers \mathcal{C} . Cai, Fürer, and Immerman's separation of $\text{FP}+\text{C}$ and PTIME is actually deduced from separating $L_{\infty\omega}^\omega(\mathcal{C})$ and PTIME by means of an Ehrenfeucht-Fraïssé style game characterization of $L_{\infty\omega}^\omega(\mathcal{C})$. Thus, it is a good idea to develop a similar game characterization for $L_{\infty\omega}^\omega(\mathcal{R})$.

One approach is provided by the work of Kolaitis and Väänänen who describe general game characterizations for $L_{\infty\omega}^\omega(Q)$ in [KV95], where Q is any set of generalized quantifiers. Their games are direct extensions of the so-called k -pebble game characterizing the logic $L_{\infty\omega}^k$ (cf.

[CFI92] or [EF99]). Similarly as in Hella's n -bijjective k -pebble game (Section 5.6.1), the general idea is that one player (spoiler) places pebbles on the elements of two structures \mathcal{A} and \mathcal{B} , which another player (duplicator) has to respond to, all the while maintaining a partial isomorphism between \mathcal{A} and \mathcal{B} with respect to the pebble placements. The Lindström quantifiers \mathcal{Q} are accounted for by special *quantifier moves* in which spoiler chooses a quantifier $Q \in \mathcal{Q}$ and defines relations $\{R_i\}$ so that $(U(\mathcal{A}), \{R_i\})$ is in the structure class which defines Q . Duplicator has to respond with relations $\{R'_i\}$ on $U(\mathcal{B})$ that also result in a structure for Q , and spoiler may then probe duplicator's response with pebble placements.

While this game sounds like a fair and feasible method, it only works for so-called *monotone* Lindström quantifiers in this simple form. Counting quantifiers such as $\exists^{\geq k} x \varphi$ are monotone since, once the formula is satisfied, an increase in the number of satisfying assignments of $\varphi(x)$ does not change the truth value of the quantifier. The rank quantifiers in \mathcal{R} , however, are not monotone and can also not be cast as monotone quantifiers since, roughly speaking, replacing zeros by ones in a matrix might both increase or decrease the rank. Kolaitis and Väänänen also describe games for non-monotone quantifiers, but these have the drawback that each of the relations defined by spoiler in his quantifier moves must be definable in $L_{\infty\omega}^{\omega}(\mathcal{R})$. This leaves us with a sort of hen-and-egg problem, given that our purpose of using the game was to distinguish between $L_{\infty\omega}^{\omega}(\mathcal{R})$ -definable and undefinable queries. Because of this restriction, it is unclear how this game characterization could be applied to separating FP+rk from PTIME.

A more manageable game characterization called *partition games* is presented by Dawar and Holm in [DH10]. Once again, spoiler and duplicator place pebbles on structures \mathcal{A} and \mathcal{B} , but this time their tasks are divided up differently. Duplicator starts each round by giving partitions \mathcal{P} and \mathcal{P}' of $U(\mathcal{A})^{k \cdot \ell}$ and $U(\mathcal{B})^{k \cdot \ell}$, respectively, as well as a bijection $f : \mathcal{P} \rightarrow \mathcal{P}'$. $k \times \ell$ is the arity of rank operators considered. The partitions need to satisfy the additional property that for any assignment $\gamma : \mathcal{P} \rightarrow \text{GF}_p$ the matrices defined by γ over $U(\mathcal{A})^k \times U(\mathcal{A})^\ell$ and by $\gamma \circ f^{-1}$ over $U(\mathcal{B})^k \times U(\mathcal{B})^\ell$ have the same rank. Now spoiler picks any part $P \in \mathcal{P}$ and places corresponding pebbles on *any* vertices in P and $f(P)$. Unlike before, spoiler gets to place all the pebbles here, giving him every possibility to unveil any inconsistency duplicator was unable to hide. This game characterizes infinitary logic with $k \times \ell$ -ary rank quantifiers.

The advantage of partition games is that the sets which play the role of matrices do not have to be definable and, in fact, do not depend on the underlying structures at all. This leads Dawar and Holm to a number of separation results for arity-2 rank operators over different prime fields, e.g., $\text{FP+rk}_2^{[2]} \neq \text{FP+rk}_3^{[2]}$ [DH10]. This points in the direction of a positive answer to the following open problem.

Open Problem. *Is it true that for all primes $p \neq q$, the expressive powers of FO+rk_p and FO+rk_q are incomparable?*

The difficulty in applying this game really appears in higher arities, though. Separating queries require us to show a winning strategy for duplicator and it appears to be hard to devise such a strategy that respects the wealth of matrices definable through higher-arity partitions. As rank operators have a strict arity hierarchy, there appears to be no way around such considerations. For this purpose it would be useful to gain a deeper understanding of matrices which can be defined by partitions in the above way. We remark, however, that the partition games described here only cover the straight-forward rank quantifiers we defined in Section 5.6.1. It is not clear at this point how the expressiveness of rank operators binding number variables relates to partition games.¹⁸ In any case, the method is a promising approach that may yet prove suitable for separating FP+rk and PTIME.

¹⁸This is no limitation to the separation results mentioned above since rank operators of arity 2 which bind number variables may be embedded into $L_{\infty\omega}^{\omega}$ without using any generalized quantifiers.

There might be a more direct way that does not require games to show that the rank logics $\text{FP}+\text{rk}_p$ have incomparable expressiveness for different primes p . When considering these logics over bare sets (\emptyset -structures), rank operators that only bind element variables can only define very restricted kinds of matrices based on the equality type of row and column indices \vec{x}, \vec{y} . It can be shown that if $\varphi(\vec{x}, \vec{y})$ defines square GF_p -matrices $A_\varphi^{|\mathcal{A}|}$ over sets \mathcal{A} , then $A_\varphi^{|\mathcal{A}|}$ and $A_\varphi^{|\mathcal{A}'|}$ have the same minimal polynomial whenever $|\mathcal{A}| = |\mathcal{A}'| \pmod p$ (for large enough sets $\mathcal{A}, \mathcal{A}'$). Thus, $A_\varphi^{|\mathcal{A}|}$ and $A_\varphi^{|\mathcal{A}'|}$ have the same sets of elementary divisors and their ranks only depend on the way in which the elementary divisors combine to form the matrices' invariant factors (cf. [AW92]).

It is not hard to show that we can employ rank operators over GF_p to define the class of sets with $|\mathcal{A}| = 0 \pmod{p^k}$ for any $k \in \mathbb{N}$.¹⁹ Thus, the combination of elementary divisors into invariant factors mentioned above cannot be completely uniform at all times. Still, based on experimental data, we feel secure to make the following conjecture. If it were true, it would separate the expressiveness of rank operators for different prime fields over the class of sets.

Quasi-Polynomial Rank Conjecture 5.7.1. If η defines a matrix with respect to element variables $|\vec{x}| = |\vec{y}| = k$ over the class of sets, then for each prime p there are polynomials $q_0(x), \dots, q_{p^k-1}(x)$ so that $\text{rk}_p^{\mathcal{A}}(\vec{x}, \vec{y})\eta = q_i(|\mathcal{A}|)$ whenever $|\mathcal{A}| = i \pmod{p^k}$ (for large enough sets \mathcal{A}).

With these precautions in mind, **let us now turn to the positive case** for the strength of rank logics. Unfortunately, it is not clear at this point how exactly rank operators might help in capturing polynomial time. As with all unconditional capturing results, rank logics would have to make up somehow for PTIME 's ability to use auxiliary orderings in the definition of order-invariant queries. The only well-established method for this is definable canonization, but of course we run into trouble when attempting this on the class of all structures (or all graphs). As explained in Section 2.4, any canonization mapping also gives rise to a complete invariant for the isomorphism problem. Given that the logics we consider have data complexity in PTIME , this would mean that φ yields a polynomial-time graph isomorphism algorithm. It would be a bit of a stretch to assume that such a breakthrough result should be easier to obtain inside a specific logical framework rather than the whole class of PTIME , if it exists at all.

Committing ourselves to more modest goals, there are still plenty of loose ends at which rank logics could demonstrate their strength. Since $\text{FP}+\text{rk}$'s expressiveness exceeds that of $\text{FP}+\text{C}$, it would for example make sense to showcase this by capturing PTIME on a class of structures on which $\text{FP}+\text{C}$ fails to do so. A suitable challenge in this area would be the class of bounded-degree graphs or even just the class of graphs of degree at most 3.²⁰ By Cai, Fürer, and Immerman's Theorem 5.4.3, $\text{FP}+\text{C}$ does not capture PTIME on the class of degree- k bounded graphs, for each $k \geq 3$. And by a seminal result of Luks [Luk82], isomorphism of bounded-degree graphs can be tested in polynomial time (for every fixed bound k). Babai and Luks [BL83] and Fürer, Schnyder, and Specker [FSS83] independently gave canonization algorithms for graphs of bounded degree which heavily rely on Luks's techniques.

Naturally, the question arises whether these methods can be cast in rank logics, thereby enabling capturing results. This is not simple, though, since Luks's methods are based on techniques for permutation groups. As discussed in Chapter 3, these methods are generally difficult to implement in a choiceless manner given that tasks as simple as deciding permutation group membership typically make inherent use of auxiliary orderings (cp. [FHL80b]). As usual, it might be a good idea to look into parallel implementations of these algorithms since parallel algorithms, just as our logics, cannot afford to *iterate* one-by-one over the structure's elements (they still do come

¹⁹Of course, this can already be defined using the arithmetic on the number sort. The point here is the implications on the progression of the ranks as the size of sets grows.

²⁰Graphs of degree at most 2 are disjoint unions of cycles and paths, which are easily canonized in $\text{FP}+\text{C}$.

equipped with auxiliary orders, though, and they may make use of them in different ways, as for example in Mulmuley’s algorithm discussed in Section 5.3.1). Indeed, parallel algorithms for many of the underlying permutation group problems have been developed [Luk86, BLS87, LM88], however, they are not easy to adapt since they largely depend on huge case distinctions inspired by and making ample use of the classification of finite simple groups. Also, despite the comprehensive effort to parallelize permutation group techniques in the 1980s, no parallel algorithm for bounded-degree graph isomorphism could yet be found.

The picture is more hopeful for the class of colored graphs each of whose color classes has bounded size (also known as *graphs of bounded color multiplicity*). Just like bounded-degree graphs, this class enjoys the property that Cai, Fürer, and Immerman’s query shows that FP+C does not capture PTIME here and that there is a polynomial-time isomorphism algorithm (cf. [FHL80a]). The isomorphism test once again depends on permutation group methods, but these are sufficiently simple for them to be parallelizable [Luk86]. In fact, Arvind, Kurur, and Vijayaraghavan could recently show that bounded color multiplicity graph isomorphism is contained in the #L hierarchy [AKV05]. So by Theorem 5.5.11 we already know that in the presence of orders, isomorphism for graphs of bounded color multiplicity can be defined in FO+rk. Yet another graph class which possesses an efficient group-theoretic isomorphism algorithm are the *graphs of bounded eigenvalue multiplicity* [BGM82]. It is well worth investigating whether all these methods can be re-cast in a choiceless manner and pave the way for a PTIME-capturing result.

Let us digress and put Section 5.5’s capturing results on ordered structures back into focus. As mentioned in the run-up to Corollary 5.5.14, our result that FO+rk captures the logspace counting hierarchy on ordered structures implies that FO+rk also gives a precise characterization of uniform $AC^0(\#L)$. This connects our rank logics to circuit complexity, where an ongoing effort is under way to compare the powers of Boolean and arithmetic circuits ([All04] gives a good overview). Given that rank logics comprise the capability to define arithmetic functions, tying them to arithmetic circuits might provide us with new techniques for determining the expressive power of rank logics on ordered structures. Conversely, arithmetic comes as a rather natural by-product of rank logics, so that the correspondence to rank logics may serve as a guideline of how natural the arithmetic counterparts of complexity classes really are. On this note, we also point to a proposition by Allender [All04] that the #L hierarchy collapses if $AC^0(\#L) = NC^1(\#L)$.²¹ The characterization of $AC^0(\#L)$ by FO+rk might be useful in showing the latter statement, which would have the interesting consequence that on ordered structures, the nesting of rank operators does not increase the expressive power inside FO+rk.

Let us return to the role of techniques for permutation groups. Typically, efficient isomorphism algorithms which use group-theoretic methods proceed by decomposing the graph’s automorphism group into abelian permutation groups (which are considered easy to handle) and suitably restricted instances of the non-abelian case. The abelian permutation group case is interesting for yet another reason. Besides its status as a natural special case of permutation groups, the problems considered here come particularly close to what our rank operators can do. Every finite abelian group is isomorphic to the *direct sum of p -groups*, each of which is the *product of groups $\mathbb{Z}/p^k\mathbb{Z}$* with $k \in \mathbb{N}$ [Lan93, I, §8]. Using this decomposition, abelian group membership directly reduces to solving systems of linear equations over the rings $\mathbb{Z}/p^k\mathbb{Z}$. In the presence of an ordering, McKenzie and Cook reduce this further to computing *bases of solution spaces of linear systems over GF_p* [MC87]. Since we originally introduced rank operators to handle such applications of linear algebra, this link is both motivation and justification for investigating whether linear systems over $\mathbb{Z}/p^k\mathbb{Z}$ can be solved in FO+rk _{p} .

²¹This statement is not as obvious as it might appear at first. A proof needs to address the specific power coming from #L-oracle gates in logdepth circuits.

Open Problem. Let η, β be numeric terms defining a system of linear equations. Is it definable in $\text{FO}+\text{rk}_p$ (respectively $\text{FP}+\text{rk}_p$, $\text{FO}+\text{rk}$, $\text{FP}+\text{rk}$) if this system has a solution over $\mathbb{Z}/p^k\mathbb{Z}$ for all $k \in \mathbb{N}$?

The procedure McKenzie and Cook use for reducing systems over $\mathbb{Z}/p^k\mathbb{Z}$ to systems over $\mathbb{Z}/p\mathbb{Z}$ is reminiscent of *Hensel liftings* (e.g. [Coh93]). In every step, solving a system modulo p^k is reduced to solving a larger one modulo p^{k-1} , where the additional equations come from a spanning set of the solution space modulo p^{k-1} . Since the computation of bases typically requires making a choice among all possible bases, this cannot directly be implemented in our logics. It may be possible, however, to integrate the recursion into one large matrix. The logics $\text{FO}+\text{rk}$ and $\text{FP}+\text{rk}$ might be needed if Chinese remainder representations have to be used for the faithful representation of operations over \mathbb{Z} .

Going from solving linear systems over prime fields to solving systems modulo prime powers, the next natural step is to solve linear systems over the integers \mathbb{Z} . Such linear systems are usually called *linear diophantine systems* and integer solutions to them can be computed in polynomial time (cf. [GLS93]). This is not quite so obvious as it is for systems over \mathbb{Q} , since Gaussian elimination yields correct solutions, but often produces values in intermediate steps which are simply too large to represent (ibid.). If we are only interested in whether a linear system has an integer solution, we may use a different approach. It is not hard to show using [NZM91] and [AW92] that an integer solution exists if and only if the system is solvable over \mathbb{Q} **and** modulo all prime powers $p^k \leq (2n)^4$, where n is the (sufficiently large) dimension of the matrix. Therefore, if the solvability of linear systems modulo prime powers is definable in $\text{FO}+\text{rk}$, then so is the solvability of linear diophantine equations.

Conversely, suppose $\text{lin}_{\mathbb{Z}}$ is an operator defining the solvability of linear systems over \mathbb{Z} . $\text{lin}_{\mathbb{Z}}$ can be used to express the solvability of linear systems modulo *any* integer $n \in \mathbb{N}$, since we can encode into the system over \mathbb{Z} that arbitrary multiples of n may be added to each equation independently. Therefore, this one operator subsumes the expressiveness of solving linear equations over GF_p for any prime p . Its additional ability to handle systems modulo prime powers puts it into a good position for investigating problems related to abelian permutation groups. In fact, instead of $\text{lin}_{\mathbb{Z}}$, we could also consider a *rank operator* $\text{rk}_{\mathbb{Z}}$ over the principal ideal domain \mathbb{Z} from which $\text{lin}_{\mathbb{Z}}$ could be defined in the same way as the solvability of systems over GF_p is defined from rk_p -operators (for a detailed development see [AW92]). It is not clear, however, whether the expressiveness of $\text{rk}_{\mathbb{Z}}$ or any other *natural* module theoretic operator subsumes the prime field rank operators rk_p .

We hope that the exposition in this chapter will spawn further investigation into the role of linear algebra within the complexity class PTIME. The results here may serve as a basis for such research by combining rank operations and logical frameworks in an arguably sensible manner, demonstrating that they nicely coexist and reinforce each other's powers. We note that earlier inquiries into the logical expressibility of linear algebraic methods as in [ABD07] or [BGS99] used to define matrices and linear systems by sporadic encodings over specific structures. Considering matrices to be given directly by formulas and terms makes linear algebraic methods ubiquitously applicable inside first-order logic and may be helpful in revealing their true power. We believe this is a reasonable step towards settling whether there is more to polynomial time computations than inductive definitions and linear algebra. The question of whether PTIME is captured by $\text{FP}+\text{rk}$ may serve as a guide towards this goal.

6 Conclusion

This thesis is making contributions to three areas of descriptive complexity theory. Chapter 3 was devoted to the question of how difficult it is to obtain a linear ordering on any given graph. We showed that a combinatorial $2^{O(n)}$ -time graph canonization algorithm of Corneil and Goldberg [CG84] can be extended to edge-colored directed graphs. We were particularly interested in the *choiceless nature* of this algorithm and exhibited that the canonization procedure can be implemented in Choiceless Polynomial Time with counting ($\tilde{C}PT+C$) if we restrict our attention to logarithmic-sized fragments of such graphs. This means that on structures over vocabularies whose relations are of arity at most 2, all PTIME properties of logarithmic-sized fragments are expressible in $\tilde{C}PT+C$. The canonical forms obtained through this procedure are ordered, so we may say that on edge-colored directed graphs singly exponential time is sufficient to obtain an ordering without the help of an auxiliary order.

Chapter 4 made contributions to the investigation of PTIME's characteristics on restricted classes of graphs. We presented an FP+C-definable canonical form for the class of interval graphs, resulting in FP+C capturing PTIME on this graph class. Unlike most graph classes for which such capturing results are known, interval graphs are not closed under taking minors. Hence, the methods developed for this capturing result are novel and they pave the way for a systematic study of the interrelation of logics and PTIME on the basis of modular decompositions of graphs. As a corollary to the techniques developed for FP+C on interval graphs, we proved that STC+C captures LOGSPACE on the class of proper interval graphs. On the class of all interval graphs, however, STC+C was shown to fall short of capturing LOGSPACE. Against the backdrop notion of non-capturing reductions, we also established that FP+C is too weak to capture PTIME on a variety of graph classes, among them bipartite graphs and chordal graphs. We finally molded our methods for interval graphs into a canonical labeling algorithm for interval graphs which is computable in logarithmic space. From this, we drew the conclusion that interval graph isomorphism is LOGSPACE-complete. In this way, our methods developed for the logical domain made a direct contribution to classical complexity theory.

Chapter 5 picked up on the loose thread of natural polynomial-time properties that make for a sensible addition to FP+C towards a logic for PTIME. After FP+C had been shown to be strictly less expressive than PTIME almost two decades ago [CFI92], no further augmentation has been proposed because of a lack of understanding of the nature of FP+C's shortcomings. Building on the findings of Atserias, Bulatov, and Dawar [ABD07] that these relate to FP+C's inaptitude to express certain linear algebraic queries, we introduced a variety of rank logics to remedy this weakness with their ability to compute the ranks of matrices over prime fields. Arguing that the strongest of these logics, FP+rk, can express Cai, Fürer, and Immerman's query which separates FP+C from PTIME, we could show that FP+rk is strictly more expressive than FP+C. Moreover, we cast the computation of matrix ranks over \mathbb{Q} and the evaluation of determinants over \mathbb{Q} and over prime fields in FP+C, thereby establishing rank computations over finite fields as a *minimal way* to strictly increase FP+C's expressiveness. In the ensuing investigation of rank operators, we could show that FO+rk already subsumes the expressiveness of TC+C, while the restrictions of this logic to computing ranks only over the field of prime characteristic p , FO+rk $_p$, all subsume the expressiveness of both DTC+C and STC+C. For these rank logics we also demonstrated a robust connection to classical complexity theory by showing that for each prime p , FO+rk $_p$ captures the

6 Conclusion

complexity class MOD_pL on ordered structures, and that $\text{FO}+\text{rk}$ captures the logspace counting hierarchy on ordered structures. This is a contribution to the study of complexity classes between LOGSPACE and PTIME and provides evidence that rank operators are really a natural object to study. In a final step, we adapted a construction of Hella [Hel96] to show that the expressiveness of rank operators strictly increases if we allow the arity of their matrices to increase. By exhibiting this property, we exhausted the currently known methods for separating logics from polynomial time and thereby established $\text{FP}+\text{rk}$ as a viable candidate for capturing PTIME . Even though it seems bold to venture that $\text{FP}+\text{rk}$ is actually expressive enough for this, it shows that rank logics need to be better understood if further progress is to be made towards a logic for PTIME .

At this point, let us shine a light on this work's implications for future research. The discussion sections of Chapters 3, 4, and 5 already point towards open problems raised by our individual results. However, we believe that the quest for a logic capturing PTIME can only be resolved by interconnecting the insights from all such approaches, which is what we shall do now.

To start, let us point out that it is unknown how the logics $\text{FP}+\text{rk}$ and $\tilde{\text{CPT}}+\text{C}$ relate to each other. Both logics currently have to be considered possible candidates for capturing PTIME . On the face of it, $\tilde{\text{CPT}}+\text{C}$ has the advantage to work over a supply of hereditarily finite sets, while $\text{FP}+\text{rk}$'s rank operators are capable of expressing a fairly broad range of advanced queries, as laid out in Chapter 5. Of course, the most immediate question is whether one of the two logics subsumes the other, putting the stronger logic ahead in the race for capturing PTIME . A proof that $\tilde{\text{CPT}}+\text{C}$ can express all $\text{FP}+\text{rk}$ queries would have to do one straightforward thing: show that the ranks of matrices over finite fields are computable in $\tilde{\text{CPT}}+\text{C}$. The understanding of this problem is essentially laid out in Section 5.3. Just like $\text{FP}+\text{C}$, $\tilde{\text{CPT}}+\text{C}$ can express rank computations over \mathbb{Q} and the evaluation of determinants over \mathbb{Q} and over finite fields. For the computation of ranks over finite fields, the only established method besides Gaussian elimination is Mulmuley's algorithm, yet both methods intrinsically rely on the presence of auxiliary orders (see Section 5.3.1). Note that Dawar, Richerby, and Rossman's approach [DRR08] for defining the CFI query in $\tilde{\text{CPT}}$ does not give any immediate hints as to how this could be done, even though the CFI query is inherently related to linear algebra over GF_2 (cf. [ABD07], Section 5.4). It would still be worthwhile to clarify whether similar methods might be able to compute such ranks, especially as we already know that a choiceless approach to calculating ranks over finite fields cannot be expressed with the means available in $\text{FP}+\text{C}$. The discovery of such methods would certainly be a breakthrough in computational linear algebra and would also help to clarify on the matter that finite fields appear to be less tractable than infinite fields, which runs counter to the intuition of most mathematicians and computer scientists.

The program for establishing the converse inclusion that $\text{FP}+\text{rk}$ includes $\tilde{\text{CPT}}+\text{C}$ is mapped out less clearly. The reason for this is that general hereditarily finite sets are non-trivial to represent inside FO -based logics, such as $\text{FP}+\text{rk}$ (cf. [BGS99]). One reasonable approach would be a tree-representation of such nested sets but in order to access this tree we would need the ability to index arbitrary paths which is unclear how to do over unordered domains. Since $\tilde{\text{CPT}}+\text{C}$ is strictly more expressive than $\text{FP}+\text{C}$, any approach to simulating $\tilde{\text{CPT}}+\text{C}$'s hereditarily finite sets in $\text{FP}+\text{rk}$ is forced to use rank computations at some point. While this seems an instructive question to ponder on, it might be more accessible to consider how certain $\tilde{\text{CPT}}+\text{C}$ queries might be expressed in $\text{FP}+\text{rk}$. As pointed out in Chapter 3, $\tilde{\text{CPT}}+\text{C}$ benefits from padding in some situations in much the same way as PTIME . This is not true, however, for $\text{FP}+\text{C}$ on general structures.¹ Thereby, it is an important open question whether rank operators can take advantage of padding. This prospect may seem easy to dismiss, but take notice that rank operators do have the ability to express *global properties* of structures such as the CFI query and that even matrices defined over

¹Observe that on ordered structures, $\text{FP}+\text{C}$'s expressiveness does increase through padding as a consequence of the Immerman-Vardi Theorem 2.3.10.

bare sets exhibit a fairly complicated structure already. It may be possible to harness this property in order to address the padded parts of structures. It would be equally important, though, to find out that rank logics do not profit from padding since this would separate $FP+rk$ from $PTIME$.

Let us move on from the direct confrontation of $FP+rk$ and $\tilde{C}PT+C$ to the rivalry over queries which they can express. This builds a bridge to the study of capturing results on graph classes since the rich knowledge in graph theory offers a good test bed for these two logic's relations to complexity classes. Let us observe here that there are no capturing results of natural graph classes exclusively known for $\tilde{C}PT+C$ or $FP+rk$, even though they are both strictly more expressive than $FP+C$. It would certainly help our understanding of what exactly makes these logics stronger if their abilities could be showcased in relation to the structural properties of graph classes. In Section 5.7 we already pointed out that the most promising graph classes for this task are those which possess polynomial-time isomorphism algorithms based on methods for permutation groups, and identified their reliance on auxiliary orders as the primary obstacle. When considering $\tilde{C}PT+C$ instead of $FP+rk$, it is equally unclear at first how this obstacle may be overcome. Following Dawar, Richerby, and Rossman's approach [DRR08] for defining the CFI query in $\tilde{C}PT+C$, however, it might be possible to overcome these ordering requirements by devising suitable data structures on the basis of the logic's supply of hereditarily finite sets. Especially for abelian permutation group problems, it is essentially sufficient to define the solvability of linear systems modulo prime powers. Thereby we would not even have to show that $\tilde{C}PT+C$ can compute ranks over finite fields in order to make progress on this question.

The status of $FP+rk$ and $\tilde{C}PT+C$ is not the only problem that would benefit from an improved understanding on restricted graph classes. In this thesis we have primarily focused on Gurevich's question for a logic capturing $PTIME$, but second to that open problem is the question for a logic capturing $LOGSPACE$ on all structures. Of course, the auxiliary orderings used by logspace computations pose even more of an obstacle here than they do for capturing $PTIME$ since the logics we may employ likely have to be less expressive than $FP+C$. The prize for an unconditional capturing result for $LOGSPACE$ is equally rewarding as for $PTIME$, though, as it might open the door to separating $LOGSPACE$ from NP or even from $PTIME$.

In this quest, the logics which are typically considered for $LOGSPACE$ appear to be particularly unsuitable over unordered structures. The reason for this is that neither $DTC+C$, $STC+C$, nor $TC+C$ can express tree isomorphism [EI00], which is a fairly central ability of $LOGSPACE$. Our proof in Section 4.4.2 that $STC+C$ fails to capture $LOGSPACE$ on interval graphs hints at the fact that this renders $STC+C$ too weak for $LOGSPACE$ on essentially all graph classes which allow for the encoding of trees. Arguably, this excludes most interesting graph classes. In order to close this gap, it is therefore worthwhile to investigate what may sensibly empower a logic to handle trees without going beyond the scope of logspace computations. Given that arithmetic trees with addition and multiplication can be evaluated in logspace [BCG⁺90, BoC92],² a sensible approach might be to replace our accustomed transitive closure operators with *tree closure operators* which come equipped this capability. Another approach would be to consider logics which are closer to the choiceless computation paradigm. For example, Grädel and Spielmann [GS99] introduce a version of *choiceless logspace* based on suitable restrictions of the hereditarily finite sets that may be used. It is an open question whether tree isomorphism is expressible in this logic.

Our exposition demonstrates that the tools of descriptive complexity theory reveal many intriguing results and open problems. We hope that the results in this thesis lay the groundwork for many more rewarding insights by interconnecting knowledge of the composition of graph classes and the power of novel logics towards a characterization of polynomial time.

²This means that the arithmetic expression corresponding to the tree can be evaluated in logspace given all the inputs.

Bibliography

- [AB09] Arora, S.; Barak, B.: *Computational Complexity – A Modern Approach*. Cambridge University Press, 2009.
- [ABD07] Atserias, A.; Bulatov, A.; Dawar, A.: Affine systems of equations and counting infinitary logic. In: *ICALP '07*, volume 4596 of *LNCS*, pp. 558–570. Springer, 2007.
- [ABO99] Allender, E.; Beals, R.; Ogihara, M.: The complexity of matrix rank and feasible systems of linear equations. In: *Comput. Complex.*, volume 8(2):pp. 99–126, 1999.
- [AF90] Ajtai, M.; Fagin, R.: Reachability is harder for directed than for undirected finite graphs. In: *J. Symb. Log.*, volume 55(1):pp. 113–150, 1990.
- [AKV05] Arvind, V.; Kurur, P. P.; Vijayaraghavan, T. C.: Bounded color multiplicity graph isomorphism is in the #L hierarchy. In: *IEEE Conference on Computational Complexity*, pp. 13–27. IEEE Computer Society, 2005.
- [All04] Allender, E.: Arithmetic circuits and counting complexity classes. In: Krajiček, J., editor, *Complexity of Computations and Proofs*, volume 13 of *Quaderni di Matematica*, pp. 33–72. Seconda Università di Napoli, 2004.
- [AO96] Allender, E.; Ogihara, M.: Relationships among PL, #L, and the determinant. In: *ITA*, volume 30(1):pp. 1–21, 1996.
- [AU79] Aho, A. V.; Ullman, J. D.: The universality of data retrieval languages. In: *POPL*, pp. 110–120. 1979.
- [AV89] Abiteboul, S.; Vianu, V.: Fixpoint extensions of first-order logic and datalog-like languages. In: *LICS*, pp. 71–79. IEEE Computer Society, 1989.
- [AV91] Abiteboul, S.; Vianu, V.: Generic computation and its complexity. In: *STOC*, pp. 209–219. ACM, 1991.
- [AW92] A., William A.; W., Steven H.: *Algebra: An Approach via Module Theory*. Number 136 in Graduate Texts in Mathematics. Springer, 1992.
- [BC08] Babai, L.; Codenotti, P.: Isomorphism of hypergraphs of low rank in moderately exponential time. In: *FOCS*, pp. 667–676. IEEE Computer Society, 2008.
- [BCG⁺90] Buss, S.; Cook, S.; Gupta, A.; Ramachandran, V.; Ac, O.: An optimal parallel algorithm for formula evaluation. In: *SIAM J. Comput.*, volume 21:pp. 755–780, 1990.
- [BDHM92] Buntrock, G.; Damm, C.; Hertrampf, U.; Meinel, C.: Structure and importance of logspace-MOD classes. In: *Math. Syst. Theory*, volume 25:pp. 223–237, 1992.
- [Ber84] Berkowitz, S. J.: On computing the determinant in small parallel time using a small number of processors. In: *Inf. Process. Lett.*, volume 18(3):pp. 147–150, 1984.

Bibliography

- [BES80] Babai, L.; Erdős, P.; Selkow, S. M.: Random graph isomorphism. In: *SIAM J. Comput.*, volume 9(3):pp. 628–635, 1980.
- [BG05] Blass, A.; Gurevich, Y.: A quick update on open problems in Blass-Gurevich-Shelah's article 'On polynomial time computations over unordered structures', 2005.
- [BGM82] Babai, L.; Grigoryev, D. Y.; Mount, D. M.: Isomorphism of graphs with bounded eigenvalue multiplicity. In: *STOC*, pp. 310–324. ACM, 1982.
- [BGS99] Blass, A.; Gurevich, Y.; Shelah, S.: Choiceless polynomial time. In: *Ann. Pure Appl. Logic*, volume 100(1-3):pp. 141–187, 1999.
- [BGS02] Blass, A.; Gurevich, Y.; Shelah, S.: On polynomial time computation over unordered structures. In: *J. Symb. Log.*, volume 67(3):pp. 1093–1125, 2002.
- [BIS90] Barrington, D. A. Mix; Immerman, N.; Straubing, H.: On uniformity within NC^1 . In: *J. Comput. Syst. Sci.*, volume 41(3):pp. 274–306, 1990.
- [BK98] Breu, H.; Kirkpatrick, D. G.: Unit disk graph recognition is NP-hard. In: *Comput. Geom. Theory Appl.*, volume 9(1-2):pp. 3–24, 1998.
- [BKL83] Babai, L.; Kantor, W. M.; Luks, E. M.: Computational complexity and the classification of finite simple groups. In: *FOCS*, pp. 162–171. IEEE, 1983.
- [BL76] Booth, K. S.; Lueker, G. S.: Testing for the consecutive ones property, interval graphs, and graph planarity using pq-tree algorithms. In: *J. Comput. Syst. Sci.*, volume 13(3):pp. 335–379, 1976.
- [BL83] Babai, L.; Luks, E. M.: Canonical labeling of graphs. In: *STOC*, pp. 171–183. 1983.
- [BLS87] Babai, L.; Luks, E. M.; Seress, Á.: Permutation groups in NC. In: *STOC*, pp. 409–420. 1987.
- [BLS99] Brandstädt, A.; Le, V. B.; Spinrad, J. P.: *Graph Classes: a Survey*. Monographs on Discrete Mathematics and Applications. SIAM, 1999.
- [BoC92] Ben-or, M.; Cleve, R.: Computing algebraic formulas using a constant number of registers. In: *SIAM J. Comput.*, volume 21:pp. 54–58, 1992.
- [CF10] Chen, Y.; Flum, J.: On p -optimal proof systems and logics for PTIME. In: *ICALP (2)*, volume 6199 of *Lecture Notes in Computer Science*, pp. 321–332. Springer, 2010.
- [CFI92] Cai, J.; Fürer, M.; Immerman, N.: An optimal lower bound on the number of variables for graph identification. In: *Combinatorica*, volume 12(4):pp. 389–410, 1992.
- [CG84] Corneil, D. G.; Goldberg, M. K.: A non-factorial algorithm for canonical numbering of a graph. In: *J. Algorithms*, volume 5(3):pp. 345–362, 1984.
- [CH82] Chandra, A.; Harel, D.: Structure and complexity of relational queries. In: *J. Comput. Syst. Sci.*, volume 25:pp. 99–128, 1982.
- [Chi85] Chistov, A. L.: Fast parallel calculation of the rank of matrices over a field of arbitrary characteristic. In: *FCT*, volume 199 of *Lecture Notes in Computer Science*, pp. 63–69. Springer, 1985.

- [CK80] Corneil, D.G.; Kirkpatrick, D.G.: A theoretical analysis of various heuristics for the graph isomorphism problem. In: *SIAM J. Comput.*, volume 9(2):pp. 281–297, 1980.
- [Coh93] Cohen, H.: *A Course in Computational Algebraic Number Theory*. Number 138 in Graduate Texts in Mathematics. Springer, 1993.
- [Coo72] Cook, S. A.: A hierarchy for nondeterministic time complexity. In: *STOC*, pp. 187–192. ACM, 1972.
- [Csa76] Csanky, L.: Fast parallel matrix inversion algorithms. In: *SIAM J. Comput.*, volume 5:pp. 618–623, 1976.
- [Daw95] Dawar, A.: Generalized quantifiers and logical reducibilities. In: *J. Log. Comput.*, volume 5(2):pp. 213–226, 1995.
- [Daw99] Dawar, A.: Finite models and finitely many variables. In: Niwinski, D.; Maron, R., editors, *Logic, Algebra and Computer Science*, volume 46 of *Banach Center Publications*. Polish Academy of Sciences, 1999.
- [Daw08] Dawar, A.: On the descriptive complexity of linear algebra. In: *WoLLIC '08*, volume 5110 of *LNCS*, pp. 17–25. Springer, 2008.
- [DG10] Dawar, A.; Grädel, E.: Properties of almost all graphs and generalized quantifiers. In: *Fundam. Inform.*, volume 98(4):pp. 351–372, 2010.
- [DGHL09] Dawar, A.; Grohe, M.; Holm, B.; Laubner, B.: Logics with rank operators. In: *LICS*, pp. 113–122. 2009.
- [DH95] Dawar, A.; Hella, L.: The expressive power of finitely many generalized quantifiers. In: *Inf. Comput.*, volume 123(2):pp. 172–184, 1995.
- [DH10] Dawar, A.; Holm, B.: Pebble games for rank logics. In: *Logical Approaches to Barriers in Computing and Complexity*. 2010.
- [Die06] Diestel, R.: *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer, 3rd edition, 2006.
- [DLN⁺09] Datta, S.; Limaye, N.; Nimbhorkar, P.; Thierauf, T.; Wagner, F.: Planar graph isomorphism is in log-space. In: *CCC*, pp. 203–214. 2009.
- [DRR08] Dawar, A.; Richerby, D.; Rossman, B.: Choiceless polynomial time, counting and the Cai-Fürer-Immerman graphs. In: *Ann. Pure Appl. Logic*, volume 152(1-3):pp. 31–50, 2008.
- [EF99] Ebbinghaus, H.-D.; Flum, J.: *Finite model theory*. Springer-Verlag, 2nd edition, 1999.
- [EFT94] Ebbinghaus, H.-D.; Flum, J.; Thomas, W.: *Mathematical Logic*. Springer-Verlag, 2nd edition, 1994.
- [EI00] Etessami, K.; Immerman, N.: Tree canonization and transitive closure. In: *Inf. Comput.*, volume 157(1-2):pp. 2–24, 2000.
- [ELT08] Egri, L.; Larose, B.; Tesson, P.: Directed st-connectivity is not expressible in symmetric datalog. In: *ICALP (2)*, pp. 172–183. Springer, 2008.

Bibliography

- [Enc10] Urbain-Jean-Joseph Le Verrier. *Encyclopædia Britannica*, 2010.
- [Ete97] Etessami, K.: Counting quantifiers, successor relations, and logarithmic space. In: *J. Comput. Syst. Sci.*, volume 54(3):pp. 400–411, 1997.
- [Fag74] Fagin, R.: Generalized first-order spectra and polynomial-time recognizable sets. In: Karp, R., editor, *Complexity of Computation*, pp. 43–73. SIAM-AMS Proceedings, 1974.
- [FF63] Faddeev, D.K.; Faddeeva, W.N.: *Computational Methods in Linear Algebra*. Freeman, San Francisco, CA, 1963.
- [FG65] Fulkerson, D.R.; Gross, O.A.: Incidence matrices and interval graphs. In: *Pacific Journal of Mathematics*, volume 15(3):pp. 835–855, 1965.
- [FH77] Földes, S.; Hammer, P. L.: Split graphs. In: *Proceedings of the Eighth Southeastern Conference on Combinatorics, Graph Theory and Computing*, pp. 311–315. Congressus Numerantium, No. XIX. Utilitas Math., 1977.
- [FHL80a] Furst, M.; Hopcroft, J.; Luks, E. M.: A subexponential algorithm for trivalent graph isomorphism. Technical Report 80-426, Computer Science, Cornell University, 1980.
- [FHL80b] Furst, M. L.; Hopcroft, J. E.; Luks, E. M.: Polynomial-time algorithms for permutation groups. In: *FOCS*, pp. 36–41. IEEE, 1980.
- [FSS83] Fürer, M.; Schnyder, W.; Specker, E.: Normal forms for trivalent graphs and graphs of bounded valence. In: *STOC*, pp. 161–170. 1983.
- [Gal67] Gallai, T.: Transitiv orientierbare Graphen. In: *Acta Mathematica Hungarica*, volume 18(1-2):pp. 25–66, 1967.
- [GG98] Grädel, E.; Gurevich, Y.: Metafinite model theory. In: *Inf. Comput.*, volume 140(1):pp. 26–81, 1998.
- [GH64] Gilmore, P. C.; Hoffman, A. J.: A characterization of comparability graphs and of interval graphs. In: *Canad. J. Math.*, volume 16:pp. 539–548, 1964.
- [GJ79] Garey, M. R.; Johnson, D. S.: *Computers and Intractability: a Guide to the Theory of NP-Completeness*. W.H. Freeman, 1979.
- [GKL⁺07] Grädel, E.; Kolaitis, Ph.G.; Libkin, L.; Marx, M.; Spencer, J.; Vardi, M.Y.; Venema, Y.; Weinstein, S.: *Finite Model Theory and Its Applications*. Texts in Theoretical Computer Science. Springer, Berlin, Germany, 2007.
- [GKOT00] Gurevich, Y.; Kutter, P. W.; Odersky, M.; Thiele, L., editors: *Abstract State Machines, Theory and Applications, International Workshop, ASM 2000, Proceedings*, volume 1912 of *Lecture Notes in Computer Science*. Springer, 2000.
- [GLS93] Grötschel, M.; Lovász, L.; Schrijver, A.: *Geometric Algorithms and Combinatorial Optimization*. Number 2 in Algorithms and Combinatorics. Springer, 2nd edition, 1993.
- [GM95] Grädel, E.; Meer, K.: Descriptive complexity theory over the real numbers. In: *STOC*, pp. 315–324. ACM, 1995.

- [GM99] Grohe, M.; Mariño, J.: Definability and descriptive complexity on databases of bounded tree-width. In: *ICDT '99*, volume 1540 of *LNCS*, pp. 70–82. Springer, 1999.
- [GO92] Grädel, E.; Otto, M.: Inductive definability with counting on finite structures. In: *CSL*, pp. 231–247. 1992.
- [Gol83] Goldberg, M. K.: A nonfactorial algorithm for testing isomorphism of two graphs. In: *Discrete Applied Mathematics*, volume 6(3):pp. 229 – 236, 1983.
- [Gol04] Golumbic, M. C.: *Algorithmic Graph Theory and Perfect Graphs (Annals of Discrete Mathematics, Vol 57)*. North-Holland Publishing Co., Amsterdam, 2004.
- [Grä02] Grädel, Erich: Model checking games. In: *Electr. Notes Theor. Comput. Sci.*, volume 67, 2002.
- [Gro98a] Grohe, M.: Finite variable logics in descriptive complexity theory. In: *Bulletin of Symbolic Logic*, volume 4(4):pp. 345–398, 1998.
- [Gro98b] Grohe, M.: Fixed-point logics on planar graphs. In: *LICS*, pp. 6–15. IEEE Computer Society, 1998.
- [Gro08] Grohe, M.: Definable tree decompositions. In: *LICS '08*, pp. 406–417. IEEE Computer Society, 2008.
- [Gro10a] Grohe, M.: Fixed-point definability and polynomial time on chordal graphs and line graphs. In: Blass, A.; Dershowitz, N.; Reisig, W., editors, *Fields of Logic and Computation*, volume 6300 of *Lecture Notes in Computer Science*, pp. 328–353. Springer, 2010.
- [Gro10b] Grohe, M.: Fixed-point definability and polynomial time on graphs with excluded minors. In: *LICS*, pp. 179–188. IEEE Computer Society, 2010.
- [GS85] Gurevich, Y.; Shelah, S.: Fixed-point extensions of first-order logic. In: *26th Annual Symposium on Foundations of Computer Science*, pp. 346–353. 1985.
- [GS96] Gurevich, Y.; Shelah, S.: On finite rigid structures. In: *J. Symb. Log.*, volume 61:pp. 61–549, 1996.
- [GS99] Grädel, E.; Spielmann, M.: Logspace reducibility via abstract state machines. In: Wing, J. M.; Woodcock, J.; Davies, J., editors, *World Congress on Formal Methods*, volume 1709 of *Lecture Notes in Computer Science*, pp. 1738–1757. Springer, 1999.
- [Gur88] Gurevich, Y.: Logic and the challenge of computer science. In: Börger, E., editor, *Current trends in theoretical computer science*, pp. 1–57. Computer Science Press, 1988.
- [Gur94] Gurevich, Y.: Evolving algebras. In: *IFIP Congress (1)*, pp. 423–427. 1994.
- [Gur97] Gurevich, Y.: From invariants to canonization. In: *Bulletin of the EATCS*, volume 63, 1997.
- [GW79] Griggs, J.R.; West, D.B.: Extremal values of the interval number of a graph. In: *SIAM J. Alg. Discrete Methods*, volume 1:pp. 1–7, 1979.

Bibliography

- [HAB02] Hesse, W.; Allender, E.; Barrington, D. A. Mix: Uniform constant-depth threshold circuits for division and iterated multiplication. In: *J. Comput. Syst. Sci.*, volume 65(4):pp. 695–716, 2002.
- [Hel89] Hella, L.: Definability hierarchies of generalized quantifiers. In: *Ann. Pure Appl. Logic*, volume 43(3):pp. 235–271, 1989.
- [Hel96] Hella, L.: Logical hierarchies in PTIME. In: *Inf. Comput.*, volume 129(1):pp. 1–19, 1996.
- [HLV96] Hella, L.; Luosto, K.; Väänänen, J. A.: The hierarchy theorem for generalized quantifiers. In: *J. Symb. Log.*, volume 61(3):pp. 802–817, 1996.
- [HM99] Hsu, W.-L.; Ma, T.-H.: Fast and simple algorithms for recognizing chordal comparability graphs and interval graphs. In: *SIAM J. Comput.*, volume 28(3):pp. 1004–1020, 1999.
- [HMPV00] Habib, M.; McConnell, R. M.; Paul, C.; Viennot, L.: Lex-BFS and partition refinement. In: *Theor. Comput. Sci.*, volume 234(1-2):pp. 59–84, 2000.
- [Hol07] Holm, B.: Deciding determinant in fixed-point logic with counting, 2007. Personal communication.
- [Hol10] Holm, B.: *Descriptive Complexity of Linear Algebra*. Ph.D. thesis, University of Cambridge, Magdalene College, 2010.
- [HRV00] Hertrampf, U.; Reith, S.; Vollmer, H.: A note on closure properties of logspace MOD classes. In: *Inf. Process. Lett.*, volume 75:pp. 91–93, 2000.
- [IL95] Immerman, N.; Landau, S.: The complexity of iterated multiplication. In: *Inf. Comput.*, volume 116(1):pp. 103–116, 1995.
- [Imm82] Immerman, N.: Upper and lower bounds for first-order expressibility. In: *Journal of Computer and System Sciences*, volume 25:pp. 76–98, 1982.
- [Imm87] Immerman, N.: Expressibility as a complexity measure: Results and directions. In: *Second Structure in Complexity Theory Conf.*, pp. 194–202. 1987.
- [Imm99] Immerman, N.: *Descriptive Complexity*. Springer, 1999.
- [Jän94] Jänich, K.: *Linear Algebra*. Undergraduate Texts in Mathematics. Springer, 1994.
- [KK09] Köbler, J.; Kuhnert, S.: The isomorphism problem for k-trees is complete for logspace. In: *MFCS*, pp. 537–548. 2009.
- [KKLV10a] Köbler, J.; Kuhnert, S.; Laubner, B.; Verbitsky, O.: Interval graphs: Canonical representation in logspace. In: *ICALP (1)*, pp. 384–395. 2010.
- [KKLV10b] Köbler, J.; Kuhnert, S.; Laubner, B.; Verbitsky, O.: Interval graphs: Canonical representation in logspace. In: *Electronic Colloquium on Computational Complexity (ECCC)*, volume 17:p. 43, 2010.
- [Kle96] Klein, P.N.: Efficient parallel algorithms for chordal graphs. In: *SIAM J. Comput.*, volume 25(4):pp. 797–827, 1996.

- [Knu81] Knuth, D. E.: *The Art of Computer Programming*. Addison-Wesley, 2nd edition, 1981.
- [Koz92] Kozen, D. C.: *The Design and Analysis of Algorithms*. Springer, 1992.
- [Koz06] Kozen, D. C.: *Theory of Computation*. Springer, 2006.
- [Kre04] Kreutzer, S.: Expressive equivalence of least and inflationary fixed-point logic. In: *Annals of Pure and Applied Logic*, volume 130(1-3):pp. 61–78, 2004.
- [KV95] Kolaitis, Ph. G.; Väänänen, J. A.: Generalized quantifiers and pebble games on finite structures. In: *Ann. Pure Appl. Logic*, volume 74(1):pp. 23–75, 1995.
- [KVV85] Kozen, D. C.; Vazirani, U. V.; Vazirani, V. V.: NC algorithms for comparability graphs, interval graphs, and testing for unique perfect matching. In: *FSTTCS*, pp. 496–503. 1985.
- [Lan93] Lang, S.: *Algebra*. Addison-Wesley, 3rd edition, 1993.
- [Lau10] Laubner, B.: Capturing polynomial time on interval graphs. In: *LICS*, pp. 199–208. IEEE Computer Society, 2010.
- [LB62] Lekkerkerker, C. G.; Boland, J. Ch.: Representation of a finite graph by a set of intervals on the real line. In: *Fundamenta Mathematicae*, volume 51:pp. 45–64, 1962.
- [LB79] Lueker, G. S.; Booth, K. S.: A linear time algorithm for deciding interval graph isomorphism. In: *J. ACM*, volume 26(2):pp. 183–195, 1979.
- [Lin66] Lindström, P.: First order predicate logic with generalized quantifiers. In: *Theoria*, volume 32:pp. 186–195, 1966.
- [Lin92] Lindell, S.: A logspace algorithm for tree canonization (extended abstract). In: *STOC*, pp. 400–404. ACM, 1992.
- [LM88] Luks, E. M.; McKenzie, P.: Parallel algorithms for solvable permutation groups. In: *J. Comput. Syst. Sci.*, volume 37(1):pp. 39–62, 1988.
- [Luk82] Luks, E. M.: Isomorphism of graphs of bounded valence can be tested in polynomial time. In: *J. Comput. Syst. Sci.*, volume 25:pp. 42–65, 1982.
- [Luk86] Luks, E. M.: Parallel algorithms for permutation groups and graph isomorphism. In: *FOCS*, pp. 292–302. IEEE, 1986.
- [Luk99] Luks, E. M.: Hypergraph isomorphism and structural equivalence of boolean functions. In: *STOC*, pp. 652–658. 1999.
- [MC87] McKenzie, P.; Cook, S. A.: The parallel complexity of abelian permutation group problems. In: *SIAM J. Comput.*, volume 16(5):pp. 880–909, 1987.
- [Mil79] Miller, G. L.: Graph isomorphism, general remarks. In: *Journal of Computer and System Sciences*, volume 18(2):pp. 128 – 142, 1979.
- [Möh84] Möhring, R. H.: Algorithmic aspects of comparability graphs and interval graphs. In: Rival, Ivan, editor, *Graphs and Order*, volume 147 of *NATO ASI Series C, Mathematical and Physical Sciences*, pp. 41–102. D. Reidel, 1984.

Bibliography

- [Mos09] Moschovakis, Y.N.: *Descriptive Set Theory*, volume 155 of *Mathematical Surveys and Monographs*. American Mathematical Society, 2nd edition, 2009.
- [Mul86] Mulmuley, K.: A fast parallel algorithm to compute the rank of a matrix over an arbitrary field. In: *STOC*, pp. 338–339. ACM Press, 1986.
- [MV97] Mahajan, M.; Vinay, V.: Determinant: Combinatorics, algorithms, and complexity. In: *Chicago J. Theor. Comput. Sci.*, volume 1997, 1997.
- [MV99] Mahajan, M.; Vinay, V.: Determinant: Old algorithms, new insights. In: *SIAM J. Discrete Math.*, volume 12(4):pp. 474–490, 1999.
- [New80] Newman, D.J.: Simple analytic proof of the prime number theorem. In: *Amer. Math. Monthly*, volume 87(9):pp. 693–696, 1980.
- [NP95] Neumann, P. M.; Praeger, C. E.: Cyclic matrices over finite fields. In: *J. London Math. Soc.*, volume 52(2):pp. 263–284, 1995.
- [NRV05] Nash, A.; Remmel, J. B.; Vianu, V.: PTIME queries revisited. In: Eiter, T.; Libkin, L., editors, *ICDT*, volume 3363 of *Lecture Notes in Computer Science*, pp. 274–288. Springer, 2005.
- [NZM91] Niven, I.; Zuckerman, H. S.; Montgomery, H. L.: *An Introduction to the Theory of Numbers*. John Wiley & Sons, 5th edition, 1991.
- [Ott97] Otto, M.: *Bounded Variable Logics and Counting: A Study in Finite Models*. Number 9 in *Lecture Notes in Logic*. Springer, 1997.
- [Rei84] Reif, J.H.: Symmetric complementation. In: *J. ACM*, volume 31(2):pp. 401–421, 1984.
- [Rei05] Reingold, O.: Undirected st-connectivity in log-space. In: Gabow, H. N.; Fagin, R., editors, *STOC*, pp. 376–385. ACM, 2005. ISBN 1-58113-960-8.
- [Rei08a] Reingold, O.: Undirected connectivity in log-space. In: *J. ACM*, volume 55(4), 2008.
- [Rei08b] Reisig, W.: Abstract State Machines for the Classroom - The Basics. In: *Logics of Specification Languages*, volume XXII:pp. 15–46, 2008.
- [Ros70] Rose, D. J.: Triangulated graphs and the elimination process. In: *Journal of Mathematical Analysis and Applications*, volume 32(3):pp. 597 – 609, 1970.
- [RST84] Ruzzo, W. L.; Simon, J.; Tompa, M.: Space-bounded hierarchies and probabilistic computations. In: *J. Comput. Syst. Sci.*, volume 28(2):pp. 216–230, 1984.
- [RTL76] Rose, D.J.; Tarjan, R.E.; Lueker, G.S.: Algorithmic aspects of vertex elimination on graphs. In: *SIAM J. Comput.*, volume 5(2):pp. 266–283, 1976.
- [Sam42] Samuelson, P.A.: A method for determining explicitly the coefficients of the characteristic equation. In: *Ann Math. Statist.*, volume 13:pp. 424–429, 1942.
- [Tod91] Toda, S.: Counting problems computationally equivalent to computing the determinant. Technical Report CSIM 91-07, Department of Computer Science, University of Electro-Communications, Tokyo, Japan, 1991.

- [Tor04] Torán, J.: On the hardness of graph isomorphism. In: *SIAM J. Comput.*, volume 33(5):pp. 1093–1108, 2004.
- [Ueh08] Uehara, R.: Simple geometrical intersection graphs. In: *WALCOM*, pp. 25–33. 2008.
- [Var82] Vardi, M. Y.: The complexity of relational query languages. In: *STOC '82*, pp. 137–146. 1982.
- [Wei76] Weisfeiler, B.: *On Construction and Identification of Graphs*, volume 558 of *Lecture Notes in Computer Science*. Springer, 1976.
- [WL68] Weisfeiler, B.; Lehman, A. A.: A reduction of a graph to a canonical form and an algebra arising during this reduction. In: *Nauchno-Technicheskaya Informatsia*, volume 2(9):pp. 12–16, 1968. In Russian.
- [ZSF⁺94] Zhang, P.; Schon, E. A.; Fischer, S. G.; Cayanis, E.; Weiss, J.; Kistler, S.; Bourne, P. E.: An algorithm based on graph theory for the assembly of contigs in physical mapping of DNA. In: *Bioinformatics*, volume 10(3):pp. 309–317, 1994.

Index

- M^A , 114
- X / \sim , *see* modout
- $[m, n]$, 11
- $[n]$, 11
- $\binom{V}{n}$, 11
- $\chi_M(x)$, *see* characteristic polynomial
- \cong , *see* isomorphism
- $\triangleleft_{\mathcal{L}}$, 58
- $\varphi[\frac{a}{x}]$, 16
- \prec_M , 67

- $AC^0(\#L)$, 116
- active object, 39
- apex of a graph, 81
- arity
 - of Lindström quantifiers, 122
 - of rank operators, 91
- arrangement vertex, 82
- asymmetric relation, 68
- atom, 37

- binary representation of numbers, 105
- bipartite graph, 14
- Boolean term, 38
- bounded-variable infinitary logic, *see* infinitary logic
- boxicity- d graph, 66

- Cai-Fürer-Immerman graphs, *see* CFI graphs
- canonical function, 25
- canonical interval labeling, 86
- canonical ordering, 25
- canonization, **26**, 25–27
- capturing, **17**, 17–18
 - on a class of structures, 18
 - on ordered structures, 17
- cell of an ordered partition, 13
- CFI graphs, 64, **111**
- characteristic polynomial, 107
- Chinese remainder representation of numbers, 105

- Choiceless Polynomial Time, **36–39**
 - with counting, 39
- chord, 12
- chordal graph, 14
- clique, 12
- closed neighborhood, 12
- closed under composition, 62
- closed under logical reductions, 23
 - rank logics, 97
- co-comparability graph, 14
- color isomorphic, 13
- color refinement, **13**, 41
- coloring, 13
- comparability graph, 14
- complete invariant, 26
- complexity class, 15–16
- component vertex, 82
- configuration, 15
- configuration graph, 15, 116
- counting logics, 21–22
- counting operator, 21
- counting term, 40
- $\tilde{C}PT+C$ program, 40
- $\tilde{C}PT$, *see* Choiceless Polynomial Time
- $\tilde{C}PT$ program, 39
- $\tilde{C}PT+C$, *see* Choiceless Polynomial Time with counting
- critical object, 39
- CRR, *see* Chinese remainder representation
- cycle, 12
- cyclic vector of a matrix, 104

- deterministic transitive closure logic, 19
- directed reachability, *see* reachability
- $D_n^p(G, S)$, 127
- DTC, *see* deterministic transitive closure logic
- dtc, *see* deterministic transitive closure logic
- DTC+C, 22
- duplicator, 123
- dynamic symbol, 37

Index

- element variable, 21
- $\text{enc}(\mathcal{A})$, *see* encoding of ordered structures
- encoding of ordered structures, 17
- end of an interval graph, 58
- equivalence class, **11**, 27
- equivalence relation, 11
- $E(v)$, 111
- expressiveness of logics, 16
- extending logics, 18

- First-order logic with majority quantifiers, 19
- fixed-point logic, 20
- FO^+ , 21
- $\text{FO}+\text{C}$, 21
- $\text{FO}+\text{M}$, *see* First-order logic with majority quantifiers
- $\text{FO}+\text{rk}$, *see* rank logic
- $\text{FO}+\text{rk}\langle k \rangle$, 117
- $\text{FO}+\text{rk}_p$, *see* rank logic
- formula, 16
- FP, *see* fixed-point logic
- $\text{FP}+\text{C}$, 22
- $\text{FP}+\text{rk}$, *see* rank logic
- $\text{FP}+\text{rk}^{[n]}$, 122
- $\text{FP}+\text{rk}_p$, *see* rank logic

- $G[W]$, *see* induced subgraph
- game characterization of $L_{\infty\omega}^k(Q_n)$, 123
- Gaussian elimination, 103
- generalized quantifier, *see* Lindström quantifier
- GF_p , 11
- graph, **12**
 - colored, 13
 - complement, 12
 - connected, 12
 - deterministic part, 19
 - underlying a directed graph, 19
- graph interpretation, 24
- Graph Interpretations Lemma, 25
- graph isomorphism completeness, 63
- grid intersection graph, 66

- height of a tree, 14
- Helly property of interval graphs, 57
- hereditarily finite sets, 33, **37**
- \mathcal{HF} , *see* hereditarily finite sets

- I_v , 57
- ifp, *see* fixed-point logic

- Immerman-Vardi Theorem, 20
- implication classes of edges, 59
- independent set, 12
- individualization of a vertex, 43
- induced relation, 11
- induced subgraph, 12
- infinitary logic, 20
- inflationary fixed-point logic, *see* fixed-point logic
- injective function on graphs, 62
- intersection graph, **13**, 58
- intersection model, 13
- interval graph, 14, **57**, 57–59
- interval representation, 57
 - minimal, 57
- interval tree, 76, 77
- invariance under isomorphism, 25
- isomorphic, *see* isomorphism
- isomorphism, **16**

- K-reversible function, 62

- $L_{\infty\omega}^\omega$, *see* infinitary logic
- $L_{\infty\omega}^s$, *see* infinitary logic
- L, *see* LOGSPACE
- $\oplus L$, *see* $\text{MOD}_p L$, 114
- $\#L$, 116
- $\#L$ hierarchy, *see* logspace counting hierarchy

- Le Verrier’s method, 107–108
- leaf of a tree, 14
- Leibniz formula, 103
- length
 - of a cycle, 12
 - of a path, 12
 - of a tuple, 11
- lexicographic disjoint union, 28
- lexicographic leader, 28
- lexicographic order, 11
 - of multisets, 11
 - of ordered structures, 27
 - of strings, 11
 - of subsets, 11
- $\#LH$, *see* logspace counting hierarchy
- Lindström quantifier, 122
- logical reduction, 22–25, *see* syntactic interpretation
- LOGSPACE, 15
- logspace, *see* LOGSPACE
- logspace counting hierarchy, 116

- $L(Q)$, 121
 \mathcal{M} , 67
 majority quantifier, 19
 max clique, *see* maximal clique
 maximal clique, 12
 modout
 graph, 12
 set, *see* equivalence class
 MOD_pL , 114
 modular decomposition tree, 60
 modular tree, 82
 module, **12**, 60, 70
 module vertex, 82

 $N[v]$, *see* closed neighborhood
 \mathbb{N} , 11
 n -bijective k -pebble game, *see* game characterization of $L_{\infty\omega}^k(Q_n)$
 \mathbb{N}_0 , 11
 neighbor
 incoming, 12
 outgoing, 12
 nesting depth of rank operators, 93
 NLOGSPACE , 15
 non-capturing reduction, 62
 non-deterministic logspace oracle Turing machine, 114
 NP , 15
 number variable, 21
 numeric syntactic interpretation, 23
 numeric term, 21

 ORD , 78
 order
 linear, 11
 strict partial, 11
 strict weak, 11
 order isomorphic, 17
 ordered partition, **13**
 ordered structure, 17
 ordering, *see* order

 P , *see* PTIME
 padding, 32
 parity of a CFI graph, 111
 path, 12
 path weight, 99
 Path Weight Lemma, 100
 polynomial-time, *see* PTIME

 Prime Number Theorem, 101
 proper interval graph, 74
 PTIME , 15

 \mathbb{Q} , 11
 Quasi-Polynomial Rank Conjecture, 135
 query, 16

 \mathbb{R} , 11
 rank logic, **92**, 90–94
 rank operator, 91
 reachability, 14
 recursion tree, 49
 Reingold’s Theorem, 14
 relation, 16
 restriction of an ordered partition, 44
 rk , *see* rank operator
 rk_p , *see* rank operator
 $\text{rk}_p^{\leq r}$, 122
 root of a tree, 14
 rule of a $\tilde{\text{CPT}}$ program, 38
 run of a $\tilde{\text{CPT}}$ program, 39
 Ruzzo-Simon-Tompa model, *see* non-deterministic logspace oracle Turing machine

 section of a stable coloring, **44**, 44–48
 small number, 105
 span, 58
 split graph, 14
 spoiler, 123
 stable coloring, 41
 stable partition, *see* stable coloring
 state of a $\tilde{\text{CPT}}$ program, 38
 static symbol, 37
 STC , *see* symmetric transitive closure logic
 stc , *see* symmetric transitive closure logic
 STC+C , 22
 structure, 16
 two-sorted, 21
 switching equivalent graph, 45
 symmetric reachability, *see* undirected reachability
 symmetric transitive closure logic, 19
 syntactic interpretation, **23**, *see* logical reduction

 TC , *see* transitive closure logic
 tc , *see* transitive closure logic
 TC+C , 22

Index

- total ordered partition, 44
- transducer, 15
- transitive closure, 15
 - of a set, *see* transitive subset closure
- transitive closure logic, 19
- transitive subset closure, 37
- tree, 14
- TSC, *see* transitive subset closure
- twin, 12

- undirected reachability, *see* reachability
- uniformly definable in $\text{FO}+\text{rk}_p$, 94
- universe, 16

- vocabulary, 16

- Weisfeiler-Lehman, 13, 26, **41–43**
 - 2-dimensional, 42

Erklärung

Ich erkläre hiermit, dass

- ich die vorliegende Dissertationsschrift “The Structure of Graphs and New Logics for the Characterization of Polynomial Time” selbständig, ohne unerlaubte Hilfe und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe,
- ich mich nicht anderwärtig um einen Doktorgrad beworben habe oder einen solchen besitze und
- mir die Promotionsordnung der Mathematisch-Naturwissenschaftlichen Fakultät II der Humboldt-Universität zu Berlin bekannt ist, gemäß Amtl. Mitteilungsblatt Nr. 34/2006.

Berlin, den 22. November 2010

Bastian Laubner