Symmetries in Constraint Satisfaction

Weisfeiler-Leman Invariance and Promise Problems

Silvia Butti

TESI DOCTORAL UPF / 2022

Director de la tesi
Dr. Víctor Dalmau Lloret

Department of Information and Communication Technologies

Universitat Pompeu Fabra Barcelona

A Gianna

Acknowledgements

I am well aware that these will be the only pages of this thesis that the vast majority of people will read, and so it only makes sense that I devoted what some people would deem a disproportionate amount of time to writing them. Those who know me well might speculate that this also provided a not-so-needed excuse for doing one of the things that I do best, that is, procrastinating difficult tasks (such as writing the actual thesis). Well, they wouldn't be wrong. Having said that, the last four years have been an unimaginable rollercoaster of emotions that went far beyond the academic process of working towards a PhD, and I sincerely want to thank everybody who has been a part of this journey, whether big or small.

I must start by thanking my supervisor, Víctor Dalmau, for being just the right person to guide me through this PhD. Víctor, thank you for believing in me from the start, for listening to my convoluted ideas and terrible explanations, and powering through many a deadline weekend of late-night zoom calls and last-minute hidden mistakes. Thanks for the constant encouragement and support even when it wasn't clear that things were going to work out.

Secondly, thanks to my collaborator and mentor, Libor Barto, for having me over in Prague for a summer of CSPs and beers. Thank you Libor for sharing your time and knowledge with me and for giving me the opportunity to learn so much in such a short time.

Finally, I want to thank my MSc supervisor, Standa Živný, for introducing me to the *wonderland* of CSPs, and more broadly to the world of theoretical research, during my studies back in Oxford.

Beyond the academic aspect, a lot of people have been working behind the scenes to make the lives of PhD students like me a little easier, be it by helping us navigate bureaucracy and paperwork, organize work trips, or secure funding. Thank you so much to Lydia, Joana, Maria and Ruth from the *Secretaria* and to Elsa, Gisela and Paola from La Caixa Foundation for making us your priority every day.

Despite always feeling a little bit like an outsider in an engineering department, working at DTIC has been an absolute pleasure. Thanks to everyone who made it such a wonderful place to do (and have a break from) research, from the beach volley tournaments to the bouldering evenings to the many nights of bravas and wine around *El Poblenou*. You are too many to mention, but you know who you are. In particular, thank you to the *Volleybolud@s* and to my *Queens & Boomers* for being the best teammates and friends. Thank you to the AI&ML group for making me feel like I belong (and for the coffee, thank you so much for the coffee!). Thank you to the Web Science group for adopting me as the weird theoretical office mate, and particularly to Ana for your guidance and support during my first year at DTIC. Thank you Adriá and Antoine for helping me make this thesis look prettier. Thank you Marina for being the queen of the Queens (and the queen of errors). Thank you Adri for being there for me whenever I needed it. E grazie Francesco, *zí*, for sharing this PhD adventure with me from the beginning to the end, and for making me laugh like no one else can. You are the best PhD brother that I could ask for.

Outside of work, Barcelona immediately felt like home and this is thanks to the warmth of the people I met here. Thanks to the *People Who Answer*, my extra talented and quirky international PhD community: Bradley, Christoph, Enrico, Iffy, Ignasi, Ivan, Iván, Loïc, Luca, Max and Poonam, I had so much fun with you. Thanks for the many dinners, the day trips, and the curfew-proof pijama parties. And thanks to all the *Lacaixitos* for providing a much-needed PhD support group over these strange times.

Thank you Claire, Gui, Natalia and Peter for our *avocado cooking classes*, for bearing with my mathematical models of *The Mind*, and for sharing with me the love for dance. And thank you, Felix, for being such an important part of this journey. Thank you especially for being by my side through the strange year that was 2020, and turning what could have been a really difficult time into a sweet memory. I don't know what I would have done without you.

Balboa was not just a house for the body, it was a home for the soul. I was lucky enough that some of the most important people in this PhD journey have shared this home with me. Thank you Benen, Bruno, Eline, Lorna and Sofi for being the best housemates, and for putting up with my weird bedtimes and my morning grumpiness.

Savvas, thank you for being my twin soul, and for taking care of me day by day. Thank you for being my biggest cheerleader in the final sprint before submission, and for telling me that "we've got this" so many times that I finally started to believe it.

And thank you Aru and Yasmin, my *Barcelonetas*, for being like sisters to me from the very beginning and throughout. Thank you because with you I grew, learned, laughed, cried, and laughed even more.

My time in Prague was special beyond what I learned academically: it made me fall in love with my work all over again after a long year of lonely home office. I must thank the CSP community for being so warm and welcoming and for the many beers, table football matches, and karaoke nights in Prague and beyond. Thank you Albert for the late night talks in the middle of the Slovenian mountains (and for being the absolute best kicker teammate). Thank you Jakub for pushing me to climb higher than I think possible - both physically and metaphorically - and for being there to catch me if I fall. And Kristina, my coauthor and CSP sister, thank you for travelling the world with me, one conference at a time.

Some people were there much before I started the PhD, and were just exceptional at sticking with me despite the distance and my terrible track record for online communication. I am grateful to each and every one of you for not giving up on me, for visiting me in Barcelona or just calling

me every now and then, and I can't wait to see you again.

Thank you Antonio for our math-dance common language, Aryan for infecting me with some of your love for algebra, Bea for our first trip to Barcelona back in 2016, Carlos for the neverending philosophical conversations, Dani for the WhatsApp podcasts, Danny for our improvised *halušky* lunch, Erika for our 23 year old friendship, Eva, Fenix and Jelle for bringing a little bit of London to Barcelona, John for the cat pictures, Josef for the *Ryzlink*-infused dancing, Laura for making me nostalgic of Oxford, Rui for your fluffiness, and Ryan for the books in the mail.

To my *dream team*, Eddy and Réka, thank you for being so talented and inspiring. Having each other as friends and role models to walk together on the not-so-beaten path of being women mathematicians was one of the best gifts that my time at UCL could give me.

And to my *Interrail* gang, Anna, Fillo, Giec, Tia and Ventu: in ten years where everything changed, our friendship has been the one thing that stayed. Thank you for our NYEs turned long weekends across Europe, I can't wait to see where the road will take us next.

Ai miei nonni mi rivolgerò in italiano. Grazie a Nonna Armida per la determinazione della sua scolorina. Grazie a Nonno Walter che a distanza di anni trova sempre nuovi modi di parlarmi e guidarmi. E a Nonna Gianna, che mi ha cresciuta e nutrita di cibo e amore come una seconda mamma, grazie per avermi insegnato l'arte dei veri abbracci.

To my incredible sister Elena, thank you for teaching me how to read and write, add and multiply, and for giving me nothing short of excellence as an example: it is by watching you succeed that I learned about the power of hard work and dedication. And of course, thank you for being my forever best friend, *sore*.

And finally to my wonderful parents, il Ciano e la Dodo, thank you for proudly embracing me and my quirks, and for loving me *for* and not *despite* being a bit of a strange human. Thank you for being my point of reference, my biggest supporters and my safety net, and most importantly, thank you for saving the Sudokus from *Il Corriere della Sera* for me every day, year after year, so that I could start playing with CSPs before I even knew that CSPs existed.

My studies took me from the shores of the river Adige to the Thames, to the Vltava, to the Mediterranean sea, but deep down my heart always belonged in the mountains. *Il Monte Altissimo* since childhood, and more recently *el Massís de Montserrat*, have taught me that the road might be a long and difficult one but that this is precisely why the views from the summit are so beautiful.

In a way, I see this work as my own personal mountain which, with the help of teachers and mentors and the support of friends and family, I was able to discover, explore, give shape to, and ultimately climb. I hope that you will join me at the top and enjoy the view with me.

Abstract

This thesis focuses on the complexity of the fixed-template Constraint Satisfaction Problem (CSP) and its variants. Our contributions are two-fold. On the one hand, we study how closure of the space of CSP instances under an equivalence relation induced by the 1-dimensional Weisfeiler-Leman algorithm correlates with solvability by the Sherali-Adams hierarchy of linear programs, invariance of the template under symmetric operations, and tractability by distributed algorithms. We then extend this analysis to the more general framework of Promise Valued CSPs. On the other hand, we initiate the study of the complexity of the Promise Model Checking Problem (PMC) parametrised by the model for the existential positive and the positive fragments of first-order logic. We lay the foundations for an algebraic approach to these problems, which allows us to fully characterize the complexity of the PMC for the existential positive fragment and to give a number of upper and lower bounds for the positive fragment.

Resumen

Esta tesis se centra en la complejidad del Problema de Satisfacción de Restricciones (*Constraint Satisfaction Problem*, CSP) de plantilla fija y sus variantes. Nuestras aportaciones se dividen en dos grupos. Por un lado, estudiamos cómo la clausura del espacio de instancias del CSP bajo una relación de equivalencia inducida por el algoritmo unidimensional de Weisfeiler-Leman se correlaciona con la resolubilidad por la jerarquía de programas lineales de Sherali-Adams, la invariabilidad de la plantilla bajo operaciones simétricas y la tratabilidad por algoritmos distribuidos. A continuación, extendemos esta análisis al marco más general de los *Promise Valued CSPs*. Por otro lado, iniciamos el estudio de la complejidad del *Promise Model Checking Problem* (PMC) parametrizado por el modelo para los fragmentos existencial positivo y positivo de la lógica de primer orden. Fijamos las bases de un enfoque algébrico para estos problemas, que nos permite caracterizar completamente la complejidad del PMC para el fragmento existencial positivo, y dar una serie de límites superiores e inferiores para el fragmento positivo.

Resum

Aquesta tesi se centra en la complexitat del Problema de Satisfacció de Restriccions (*Constraint Satisfaction Problem*, CSP) de plantilla fixa i les seves variants. Les nostres aportacions es divideixen en dos grups. D'una banda, estudiem com la clausura de l'espai d'instàncies del CSP sota una relació d'equivalència induïda per l'algorisme unidimensional de Weisfeiler-Leman es correlaciona amb la resolubilitat per la jerarquia de programes lineals de Sherali-Adams, la invariabilitat de la plantilla sota operacions simètriques i la tractabilitat per algorismes distribuïts. A continuació, estenem aquesta anàlisi al marc més general dels *Promise Valued CSPs*. D'altra banda, iniciem l'estudi de la complexitat del *Promise Model Checking Problem* (PMC) parametritzat pel model per als fragments existencial positiu i positiu de la lògica de primer ordre. Fixem les bases d'un enfocament algèbric per aquests problemes, que ens permet caracteritzar completament la complexitat del PMC per al fragment existencial positiu i donar una sèrie de límits superiors i inferiors per al fragment positiu.

Sommario

Questa tesi si concentra sulla complessità del Problema di Soddisfacimento di Vincoli (*Constraint Satisfaction Problem*, CSP) a modello fisso e delle sue varianti. Il nostro contributo è duplice. In primo luogo, studiamo come la chiusura dello spazio delle istanze del CSP rispetto a una relazione di equivalenza indotta dall'algoritmo unidimensionale di Weisfeiler-Leman sia correlata alla risolvibilità mediante la gerarchia di programmi lineari di Sherali-Adams, all'invarianza del modello rispetto a operazioni simmetriche e alla trattabilità mediante algoritmi distribuiti. Estendiamo poi questa analisi al quadro più generale del *Promise Valued CSP*. In secondo luogo, avviamo lo studio della complessità del *Promise Model Checking Problem* (PMC) parametrizzato dal modello per i frammenti esistenziale positivo e positivo della logica del primo ordine. Poniamo le basi per un approccio algebrico a questi problemi, che ci permette di caratterizzare completamente la complessità del PMC per il frammento esistenziale positivo e di fornire una serie di limiti superiori e inferiori per il frammento positivo.

Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Manuscript Outline and Contributions 3

2 Background **7**

2.1 The Constraint Satisfaction Problem 7

2.1.1 The algebraic approach 10

2.2 Extensions of the CSP 12

2.2.1 Promise Constraint Satisfaction Problems 13

2.2.2 Valued Constraint Satisfaction Problems 14

2.2.3 Quantified Constraint Satisfaction Problems 16

3 Preliminaries **19**

I Weisfeiler-Leman Invariant and Distributed CSPs **27**

4 Introduction **29**

5 Relaxation techniques **33**

5.1 Combinatorial relaxations of isomorphism 33

5.1.1 Fractional isomorphism of graphs 33

5.1.2 From graphs to relational structures 36

5.2 Linear Programming 40

5.2.1 The Sherali–Adams hierarchy for CSPs 41
5.2.2 Applying the SA method exactly 46
5.3 Local Consistency 49

6 Distributed CSPs **53**
6.1 Introduction 53
6.2 Distributed CSPs 55
6.3 The Structure Theorem 65
6.4 The Complexity of DCSP 70
6.4.1 Intractable Templates 70
6.4.2 Tractable Templates 76
6.4.3 The Search Algorithm 82

7 Weisfeiler-Leman Invariant CSPs **85**
7.1 Introduction 85
7.2 The Decomposition Theorem 87
7.3 Weisfeiler-Leman invariant CSPs 91

8 Weisfeiler-Leman Invariant Promise Valued CSPs **95**
8.1 Introduction 95
8.2 Promise Valued CSPs 97
8.3 Fractional operations 100
8.4 The Decomposition Theorem for valued structures 107
8.5 Weisfeiler-Leman invariant PVCSPs 108

9 A Glimpse on the Higher Levels **115**
9.1 k-WL, Counting Logics, and Treewidth 115
9.2 Sherali-Adams meets Weisfeiler-Leman 119
9.3 Proof of Lemma 9.4 122
9.4 Proof of Theorem 5.4 123
9.5 Proof of Theorem 9.7 136

10 Conclusion **145**

II Promise Model Checking **147**

11 Introduction **149**
11.1 Introduction 149
11.1.1 Model checking problem parametrized by the model 150
11.1.2 Promise model checking problem 152
11.2 Preliminaries 154
11.3 Interesting fragments 159

12 The Complexity of Promise Model Checking **161**
12.1 Existential positive fragment 161
12.1.1 Characterization of templates and p-$\mathscr{L}$-definability . 161
12.1.2 Complexity classification 164
12.2 Positive fragment 165
12.2.1 Witnesses for quantified formulas 165
12.2.2 Characterization of templates and p-$\mathscr{L}$-definability . 166
12.2.3 Membership 168
12.2.4 Hardness 171
12.2.5 Summary 175

13 Conclusion **177**

Bibliography **179**

Glossary of Abbreviations **201**

Funding **203**

List of Figures

1.1 Roadmap of this manuscript. 6

3.1 The component-wise application of a polymorphism. 23

5.1 Iterated degree as infinite tree. 35

5.2 $\equiv_1$-equivalent non-isomorphic graphs. 36

8.1 Diagram of the proof of Theorem 8.7. 113

9.1 $\equiv_1$-equivalent non-isomorphic 3-regular graphs. 116

9.2 Vandermonde Matrix. 129

9.3 A winning strategy closed under odd chains. 142

11.1 Known complexity results for $\mathscr{L}$-MC($\mathbf{A}$). 152

11.2 Complexity results for $\mathscr{L}$-PMC($\mathbf{A}, \mathbf{B}$). 154

13.1 An example of unknown complexity. 178

1 Introduction

1.1 Motivation

What makes a computational problem easy or hard? This question is at the heart of all the research in computational complexity, that is, the science of determining whether or not there exist 'fast' algorithms to solve certain computational problems, and of classifying such problems into broad classes induced by the (asymptotic) speed of these algorithms.

The relevance of studying the speed of algorithms in today's Digital Age could not be overstated. In fact, the most important open problem in computational complexity, which asks whether the complexity classes P and NP coincide – that is, whether for all problems for which a solution can be verified to be correct in polynomial time, a solution can also be found in polynomial time – is one of the seven Millennium Prize Problems in mathematics. A resolution of the P versus NP problem in either direction would have far-reaching consequences in virtually all aspects of life as we know it, with tangible applications in cryptography, artificial intelligence, operations research, and philosophy of science just to name a few [Imp95]. A proof that P = NP would even drastically revolutionise mathematics as it would, in the words of Stephen Cook, "allow a computer to find a formal proof of any theorem that has a proof of reasonable length" [Coo00].

The fact that the P versus NP problem remains wide open indicates that, as a scientific community, we still do not have a satisfactory answer to the question that we opened this thesis with. The study of constraint satisfaction problems can be an incredibly good laboratory to look for

these answers and, hence, contribute to the grand challenge of better understanding the sources of tractability for computational problems. This is the context that we wish to position this thesis in.

The constraint satisfaction problem (CSP) is, informally, the problem of finding an assignment from a set of values to a set of variables that satisfies a series of given constraints on the values that the variables can take. CSPs originated around 40 years ago independently in different communities, and it was only relatively recently that the questions asked by these different communities were brought together under a common framework. The benefits of studying the CSP arise from the fact that it is a general enough framework to express a wide range of computational problems (including various complete problems for the standard complexity classes mentioned above), yet it maintains a rich mathematical structure that allows researchers to draw techniques from fields as diverse as universal algebra, finite model theory, graph theory, combinatorics, and more.

The fundamental observation in constraint satisfaction theory is that the computational complexity of these problems is fully determined by a class of symmetries on the space of solutions of said problems. In particular, more symmetries guarantee the correctness of certain polynomial time algorithms (and hence yield tractable problems), while the lack of symmetries in the solution space of a CSP produces a reduction to 3-SAT (the prototypical NP-complete problem) and hence implies that the CSP is intractable.

Symmetries in the context of the CSP and its extensions will arise in this thesis under different guises: in the classical form of *polymorphisms* giving tractability of CSPs by certain algorithms; in the form of (surjective) *multi-homomorphisms* determining the complexity of an extension of the CSP known as the Model Checking Problem; and in the unusual form of an equivalence relation on the variables of a CSP instance induced by the *Weisfeiler-Leman algorithm*, which bears surprising connections to solvability by linear programs, distributed computing, and even polymorphisms themselves.

1.2 Manuscript Outline and Contributions

This thesis is organized as follows. In Chapter 2 we introduce constraint satisfaction problems in a little more detail and present some of the classical results in CSP theory. Furthermore, we introduce three generalisations of the CSP: Promise, Valued, and Quantified CSPs.

In Chapter 3 we set the notation and provide all the definitions that the reader ought to be familiar with in order to proceed with the reading. The thesis is then divided in two parts.

Part I is dedicated, broadly, to the study of CSPs solvable by linear programming algorithms. In fact, the main contribution of this part of the thesis is perhaps the addition of two novel characterizations of the class of CSPs solvable by a certain linear program: one in terms of solvability by distributed algorithms, and the other in terms of invariance under an equivalence relation induced by the Weisfeiler-Leman algorithm. Chapter 4 serves as a discursive introduction for this first part of the thesis.

In Chapter 5 we present some more technical background on three types of relaxations techniques that will appear again and again in the subsequent chapters: particularly, the Weisfeiler-Leman isomorphism test (whose application in the context of CSPs is new to this work), linear programming relaxations for CSPs, and local consistency methods. While the majority of this chapter consists of background material that may be already known to the reader who is familiar with CSPs, we also introduce some definitions that are novel in this work.

In Chapter 6 we study the complexity of solving CSPs via distributed algorithms. The main result of this chapter is a dichotomy theorem for the Distributed CSP: we show that this problem is solvable in polynomial time if and only if the template is invariant under symmetric polymorphisms of all arities. The results in this chapter have been published in

[BD22] Silvia Butti and Víctor Dalmau. The Complexity of the Distributed Constraint Satisfaction Problem. *Theory of Computing Systems*, 2022.

An extended abstract of this work appeared in

[BD21b] Silvia Butti and Víctor Dalmau. The complexity of the distributed constraint satisfaction problem. In *Proceedings of the 38th International Symposium on Theoretical Aspects of Computer Science* (STACS 2021).

In Chapter 7, drawing inspiration from the methods used to establish the dichotomy for the distributed CSP, we explore the connection between linear programming and the Weisfeiler-Leman method. The main result of this chapter is a *decomposition theorem* for the first level of a hierarchy of linear programs obtained using the Sherali-Adams lift-and-project method. Using the decomposition theorem, we obtain the aforementioned novel characterizations of CSPs solvable by said linear program. These results appeared in[1]

[BD21a] Silvia Butti and Víctor Dalmau. Fractional Homomorphism, Weisfeiler-Leman Invariance, and the Sherali-Adams Hierarchy for the Constraint Satisfaction Problem. In *Proceedings of the 46th International Symposium on Mathematical Foundations of Computer Science* (MFCS 2021).

In Chapter 8 we show that the connection explored in the previous chapter extends to the more general framework of Promise Valued CSPs, establishing an analogous decomposition theorem for valued structures. As above, we use this decomposition theorem to study the characterization of PVCSPs solvable by a certain linear program. Collaterally, we show that two commonly used linear programming relaxations are no longer equivalent in this broader framework. These results appeared in

[BB22] Libor Barto and Silvia Butti. Weisfeiler-Leman Invariant Promise Valued CSPs. In *Proceedings of the 28th International Conference on Principles and Practice of Constraint Programming* (CP 2022).

Finally, in Chapter 9 we lift the connection between the Weisfeiler-Leman method and linear programming to higher dimensions. In partic-

[1] It is worth noting that many of the proofs as we present them here have been substantially rewritten and simplified, drawing inspiration from the follow-up [BB22].

ular, we lift the decomposition theorem studied in the previous chapters to higher levels of the Sherali-Adams hierarchy. Moreover, we define a higher-dimensional notion of Weisfeiler-Leman equivalence for relational structures, and characterize it in terms of homomorphism indistinguishability. These results also appeared in

[BD21a] Silvia Butti and Víctor Dalmau. Fractional Homomorphism, Weisfeiler-Leman Invariance, and the Sherali-Adams Hierarchy for the Constraint Satisfaction Problem. In *Proceedings of the 46th International Symposium on Mathematical Foundations of Computer Science* (MFCS 2021).

In Chapter 10 we conclude Part I by proposing some open questions and directions for future work.

In Part II we take a logical perspective and regard the CSP as the model checking problem over the positive existential conjunctive fragment of first-order logic. Motivated by recent developments in the area, we study a problem that extends the CSP in two directions. First, we consider different fragments of first-order logic which arise from different choices of quantifiers and connectives. Second, we transform this into a promise problem by considering two models, one 'strong' and one 'weak', and only ask to distinguish sentences that are satisfiable in the strong model from those that are not even satisfiable in the weak model.

We call this problem the *Promise Model Checking problem* (PMC) and introduce it in Chapter 11. As our main contributions, in Chapter 12 we give a dichotomy for the PMC over the existential positive equality-free fragment of first-order logic, and we show that the PMC over the positive equality-free fragment includes problems that are complete for at least four standard complexity classes. We conclude in Chapter 13 by discussing some concrete open problems whose complexity we still do not fully understand. These results appeared in

[ABB22] Kristina Asimi, Libor Barto and Silvia Butti. Fixed-Template Promise Model Checking Problems. In *Proceedings of the 28th International Conference on Principles and Practice of Constraint Programming* (CP 2022).

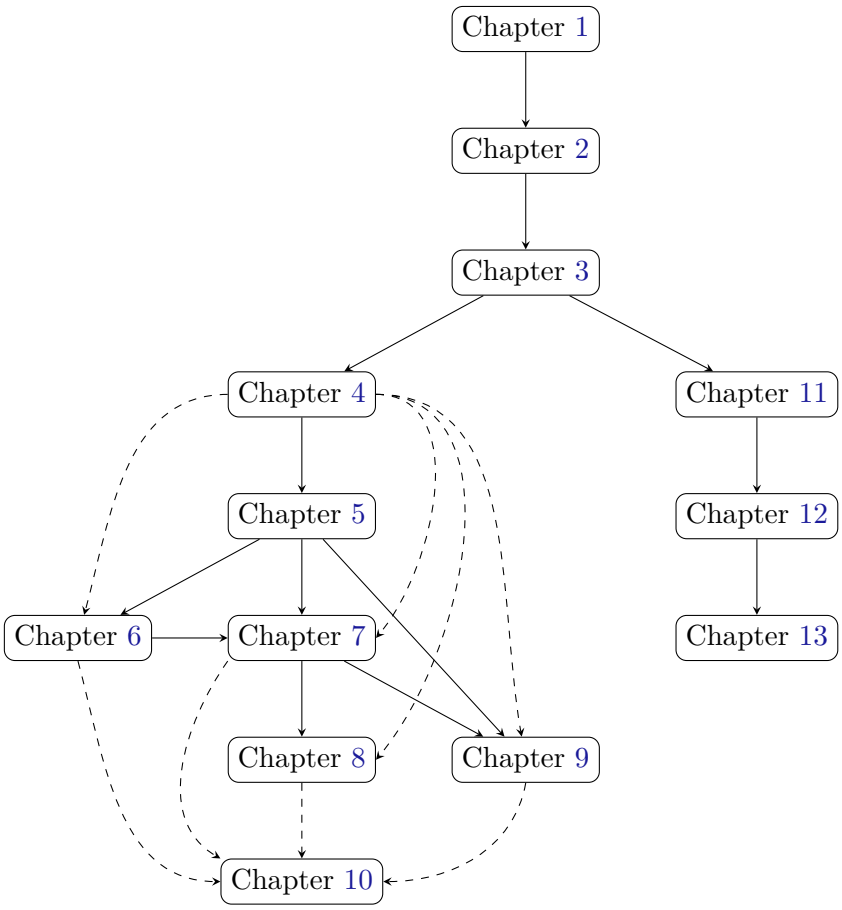

Figure 1.1: Roadmap of this manuscript.

2 Background

In this chapter we introduce constraint satisfaction problems and discuss the main recent results in the field. We then briefly survey three extensions of the classical problem.

2.1 The Constraint Satisfaction Problem

The Constraint Satisfaction Problem (CSP) consists of a collection of variables and a collection of constraints where each constraint specifies the valid combinations of values that can be taken simultaneously by the variables in its scope. The goal is to decide if there exists an assignment of the elements of a domain to the variables that satisfies all constraints. The CSP is a very rich mathematical framework that is general enough to encompass many important computational problems such as various versions of logical satisfiability, graph coloring, and systems of linear equations, as well as applications in areas as diverse as artificial intelligence, optimization, computer algebra, computational biology, computational linguistics, among many others [RBW06, KŽ17b].

Formally, an instance of the CSP is commonly defined as a triple (X, D, C) where X is a finite set of variables, D is a finite domain, and C is a finite set of constraints. Each constraint $c \in C$ is a pair $((x_1, \ldots, x_r), R)$ where $(x_1, \ldots, x_r)$ is a tuple of elements from X and R is an r-ary relation on the domain D. The goal is then to decide if there exists a map $h : X \to D$ such that all the constraints are satisfied – that is, $(h(x_1), \ldots, h(x_r)) \in R$ for each $((x_1, \ldots, x_r), R) \in C$.

As a simple exercise, the reader can verify that if all the constraints are the binary inequality relation, then an instance (X, D, C) of the CSP is the problem of deciding whether a given graph (with vertex set X and edge set defined by C) is $|D|$-colourable.

Aside from the decision problem, which is the primary focus of this thesis, there are various other interesting computational problems that arise from the CSP framework, such as the problem of finding a satisfying assignment (under the assumption that one exists) [BJK05], the problem of finding an assignment that satisfies the maximum number of constraints (where a fully satisfying assignment may not exist) [DJKK08], the problem of counting the number of satisfying assignments [Jer17]. Going back to the graph colouring example, the above problems correspond to the following tasks respectively: given a graph G defined by (X, C) as above, find a proper $|D|$-colouring of G; find a $|D|$-colouring of G that maximizes the number of non-monochromatic edges; count in how many ways G can be $|D|$-coloured with a proper colouring.

There are a number of alternative formulations for the constraint satisfaction problem. The one that is most commonly used by universal algebraists is the homomorphism formulation [FV98, Jea98]: given a pair of relational structures $\mathbf{X}$ and $\mathbf{A}$, the constraint satisfaction problem asks to decide whether $\mathbf{X}$ is homomorphic to $\mathbf{A}$. Going back once again to the example above, the d-coloring problem can be encoded as the problem of deciding whether a given graph is homomorphic to $\mathbf{K}_d$, where $\mathbf{K}_d$ denotes the clique on d vertices.

Alternatively, one can regard the CSP as a model checking problem [MM18]. Given a structure $\mathbf{A}$ (also called a *model*) and a sentence ϕ in a specified logic, the model checking problem asks whether $\mathbf{A} \vDash \phi$, that is, whether ϕ is satisfied in $\mathbf{A}$. If we require that the model is purely relational and we restrict the logic to the positive existential conjunctive fragment of first-order logic, we obtain precisely the constraint satisfaction problem. Notice that this logical formulation of the CSP can also be viewed as the problem of evaluating the truth value of a Boolean conjunctive query over a relational database [KV00].

It is not difficult to see that these three formulations of the CSP are all equivalent. In this thesis, we will predominantly use the homomorphism formulation. However, in Part II, it will be convenient to switch to the

logical formulation.

The CSP in its full generality is NP-complete, as it can encode famously NP-complete problems such as 3-SAT and graph colouring. Consequently, an important research effort has been put into identifying tractable fragments of the problem, in particular by fixing the target structure $\mathbf{A}$, also known as the template. This version of the problem is known as the *fixed-template*[1] CSP and, together with its variants, will be the main focus of this work.[2] In particular, the phrase "a CSP" will mean the CSP over some fixed template.

The first seminal result in the field is Schaefer's dichotomy theorem for the CSP on the Boolean domain: in his influential 1978 paper, Schaefer showed that for every fixed Boolean template $\mathbf{A}$, the CSP over $\mathbf{A}$ is either solvable in polynomial time or it is NP-complete [Sch78]. Subsequently, Hell and Nešetřil [HN90] showed that a parallel dichotomy holds for all the CSPs where the template is a graph: in particular, they showed that for every graph $\mathbf{A}$, CSP($\mathbf{A}$) is in the complexity class P if $\mathbf{A}$ is bipartite, and is NP-complete otherwise.

Motivated by these results, in their foundational 1998 paper [FV98], Feder and Vardi conjectured that this dichotomy holds for all CSPs: that is, for every finite-domain template $\mathbf{A}$, the CSP over $\mathbf{A}$ is either solvable in polynomial time or it is NP-complete. This hypothesis came to be known as the *dichotomy conjecture*, and has been the starting point for an intensive research program in the subsequent 20 years.

The stepping stone that allowed the field to flourish was the observation that the complexity of the fixed-template CSP depends only on the template's invariance under certain operations, now known as its polymorphisms. This idea, which came to be known as *the algebraic approach to constraint satisfaction*, was pioneered in [JCG97, Jea98]. Thanks to the algebraic approach, the dichotomy conjecture was confirmed for the 3-element domain [Bul06] and for smooth digraphs (i.e., directed graphs with no sources and no sinks) [BKN09].

In 2005, Bulatov, Jeavons and Krokhin [BJK05] gave a first formulation

[1] Sometimes this is also referred to as the *non-uniform* CSP.

[2] We remark that considerable work has also been devoted to the study of the CSP with *structural* or *left-hand side* restrictions, see [Gro07].

of the dichotomy conjecture in algebraic terms, and proved the hardness side: informally, that if a template has no non-trivial higher-dimensional symmetries, then the corresponding CSP is NP-complete. It took until 2017 for the tractability direction to be confirmed, with the two independent proofs of Bulatov [Bul17] and Zhuk [Zhu17, Zhu20], who provided polynomial time algorithms for all the CSPs that exhibit said non-trivial symmetry. Recall that, if P $\neq$ NP, then NP contains problems of intermediate complexity [Lad75]. As observed two decades earlier by Feder and Vardi, the results of Bulatov and Zhuk – and hence the positive answer to the dichotomy conjecture – imply that the constraint satisfaction problem is one of the largest classes of computational problems that exhibit a P versus NP-complete dichotomy.

Recently, Barto et al. [BBB+21] proposed a unifying framework for the three approaches that had been deployed to attack the dichotomy conjecture: the aforementioned theories of Bulatov and Zhuk, and absorption theory [BK17], which had proved so fruitful in the characterization of CSPs solvable by local consistency methods (more about this in Section 5.3).

The recent *post-dichotomy* years have also seen an increasing interest in researching "variations on the theme" of the CSP. Many of these will be touched upon in this thesis. We introduce the main extensions of the CSP in Section 2.2, but first, let us discuss the fundamentals of the algebraic approach to the CSP in a little more detail.

2.1.1 The algebraic approach

In this section we discuss some basic concepts that are at the core of what is known as *the algebraic approach to constraint satisfaction*. For a more substantial introduction to the topic, see the surveys [Che09, BKW17]. While not all the concepts introduced in this section will be used directly in this thesis, the idea that the complexity of a computational problem is a product of its symmetries (or the lack thereof) will be a common thread in the entirety of this work.

A *primitive positive formula* (*pp-formula* for short) is a formula in first-order logic which only uses atomic formulas, the equality relation, conjunction, and existential quantification. Let $\mathbf{A}$, $\mathbf{A}'$ be relational struc-

tures on the same domain. We say that $\mathbf{A}$ *pp-defines* $\mathbf{A}'$ if all the relations in $\mathbf{A}'$ can be defined via pp-formulas that only use atomic formulas (i.e., relations) from $\mathbf{A}$. The largest relational structure that can be pp-defined from $\mathbf{A}$ will be denoted $\langle\mathbf{A}\rangle$. Note that $\langle\mathbf{A}\rangle$ potentially has infinitely many relations. The set of relations of $\langle\mathbf{A}\rangle$ will be called a *relational clone* (or simply *co-clone*), the reason for this will become clear later.

An easy gadget reduction gives the following theorem.

Theorem 2.1. *Let* $\mathbf{A}$, $\mathbf{A}'$ *be relational structures on the same domain. If* $\mathbf{A}$ *pp-defines* $\mathbf{A}'$, *then* $\mathrm{CSP}(\mathbf{A}')$ *is log-space reducible to* $\mathrm{CSP}(\mathbf{A})$.

We say that an n-ary operation f *preserves* a relation R if whenever we apply f component-wise to any combination of tuples from R, we obtain another tuple in R. If f preserves all the relations of a relational structure $\mathbf{A}$, then we say that f is a *polymorphism* of $\mathbf{A}$. The set of polymorphisms of $\mathbf{A}$ will be denoted $\mathsf{Pol}(\mathbf{A})$.

A *clone* is a set of finite-arity operations over the same set that contains all the projections and is closed under composition. It turns out that for every relational structure $\mathbf{A}$, $\mathsf{Pol}(\mathbf{A})$ forms a clone. On the other hand, for a set of operations F, we will denote by $\mathsf{Inv}(F)$ the largest relational structure that has all operations in F as polymorphisms.

The following theorem, due to Geiger [Gei68] and Bodnarchuk et al. [BKKR69a, BKKR69b], establishes a Galois connection between clones and relational clones.

Theorem 2.2 ([Gei68, BKKR69a, BKKR69b])**.** *Let* $\mathbf{A}$ *be a finite-domain relational structure. Then,* $\langle\mathbf{A}\rangle = \mathsf{Inv}(\mathsf{Pol}(\mathbf{A}))$.

That is, a relation R is pp-definable from $\mathbf{A}$ if and only if all polymorphisms of $\mathbf{A}$ preserve R. Putting together Theorems 2.1 and 2.2, we obtain that the complexity of $\mathrm{CSP}(\mathbf{A})$ is fully determined by $\mathsf{Pol}(\mathbf{A})$. This is made explicit in the following theorem, due to [Jea98], which is at the heart of the algebraic approach to constraint satisfaction.

Theorem 2.3 ([Jea98])**.** *Let* $\mathbf{A}$, $\mathbf{A}'$ *be relational structures on the same domain. If* $\mathsf{Pol}(\mathbf{A}) \subseteq \mathsf{Pol}(\mathbf{A}')$, *then* $\mathrm{CSP}(\mathbf{A}')$ *is log-space reducible to* $\mathrm{CSP}(\mathbf{A})$.

Theorem 2.3 then justifies making statements about the complexity of CSPs in terms of invariance under polymorphisms. In particular, we can reformulate the dichotomy conjecture – now Dichotomy Theorem – in algebraic terms.

We say that an operation f is *weak near-unanimity* (WNU) if for all x, y in the domain of f, it holds that

$$f(y, x, \ldots, x) = f(x, y, \ldots, x) = \ldots = f(x, x, \ldots, y).$$

It turns out that WNU operations capture the notion of "non-trivial symmetry" mentioned above.[3] Then, the Dichotomy Theorem can be rephrased as follows.

Theorem 2.4 ([Bul17, Zhu20])**.** *Let* $\mathbf{A}$ *be a finite-domain relational structure. If* $\mathbf{A}$ *has a WNU polymorphism of some arity, then* CSP($\mathbf{A}$) *is in* P*. Otherwise,* CSP($\mathbf{A}$) *is* NP*-complete.*

We point out that the algebraic approach has since been developed further, for instance, stronger definability notions have been developed (with an appropriate Galois correspondence in terms of polymorphisms) that allow for reductions between relational structures on different domains, see [BOP18]. Moreover, the meta-problem of the complexity of checking invariance under certain classes of polymorphisms has been studied in [CL17].

2.2 Extensions of the CSP

In this section, we briefly discuss three extensions of the CSP that have received a lot of attention in recent years and that will be relevant to the rest of the thesis. Other extensions, such as the infinite-domain CSP [Bod21], surjective CSP [BKM12], or counting CSP [Jer17] are outside the scope of this thesis.

[3] We point out that WNU operations are not *the only* source of non-trivial higher-dimensional symmetry. In fact, equivalent characterizations for the dichotomy theorem are available in terms of cyclic, Siggers, and Taylor operations [BKW17].

2.2.1 Promise Constraint Satisfaction Problems

Currently, one of the most promising research directions in the field of constraint satisfaction is the recently introduced framework of the Promise Constraint Satisfaction Problem (PCSP). Intuitively, in this setting each constraint comes in two forms, one strong and one weak, and the task is to distinguish whether the input is satisfiable in the strong sense, or it is not satisfiable even in the weak sense. Here, the "promise" is that the input will fall precisely into one of these cases. More formally, a PCSP template is a pair of structures $(\mathbf{A}, \mathbf{B})$ such that there is a homomorphism from $\mathbf{A}$ to $\mathbf{B}$, and the PCSP over $(\mathbf{A}, \mathbf{B})$ is the problem of distinguishing structures homomorphic to $\mathbf{A}$ from those that are not homomorphic to $\mathbf{B}$. In the search version, the input structure is guaranteed to be homomorphic to $\mathbf{A}$, and the task is to find a homomorphism to $\mathbf{B}$. Note that when $\mathbf{A}$ and $\mathbf{B}$ coincide, the PCSP reduces to the CSP.

A well-known family of PCSP examples is the problem of distinguishing c-colourable graphs from those that are not even d-colourable for some fixed $d \geq c$. Formally, this can be encoded as the PCSP over $(\mathbf{K}_c, \mathbf{K}_d)$. Dating back to the 1970s [GJ76], the complexity of the approximate graph colouring problem – though conjectured to be NP-hard for all $d \geq c \geq 3$ – is still not well understood, and prior to recent years the only known result was the hardness of 4-colouring a 3-colourable graph [KLS00].

PCSPs were originally introduced in [AGH17], where the authors defined a natural extension of the notion of polymorphism[4] and used this newly developed theory to establish NP-hardness of a particular version of approximate Boolean satisfiability which they called $(2 + \varepsilon)$-SAT.

Building on [AGH17], in a series of papers [BG19, BG20, BGWŽ20] Brakensiek and Guruswami gave algebraic conditions for tractability of PCSPs by certain algorithms based on convex relaxations, providing further evidence that, much like in the CSP setting, the complexity of promise problems also depends on the higher-level symmetries encapsulated in the notion of polymorphism. Moreover, the authors were able to provide a dichotomy for a family of Boolean PCSPs [BG21], leading to further study in [FKOS19].

An important advancement in the development of an algebraic theory

[4] Called *weak polymorphism* in [AGH17].

of PCSPs was added in [BKO19, BBKO21], where the authors showed that the complexity of a PCSP does not depend on the associated polymorphisms *per se* but rather on the identities satisfied by said polymorphisms. This observation – and the theory that was developed upon it – was the foundation to overcome what is maybe the main obstacle in lifting results from the classical CSP theory to PCSPs, that is, the fact that in the promise world polymorphisms are not closed under composition.

One of the main applications of the new algebraic approach was the NP-hardness of various approximate graph colouring problems, including the hardness of 5-colouring a 3-colourable graph, which represented a major breakthrough. Despite considerable subsequent progress [WŽ20, AD22, CŽ22b], the complexity of approximate graph colouring in its full generality is still an open problem.

While a complete complexity classification for PCSPs seems currently far away, in recent years there have been substantial developments in the understanding of the power of specific algorithms for PCSPs, e.g. of convex relaxations [BGWŽ20, CZ22a], constant levels of the Sherali-Adams hierarchy [CŽ22b], or local consistency methods [AD22].

2.2.2 Valued Constraint Satisfaction Problems

The framework of Valued Constraint Satisfaction Problems (VCSP) is a generalization of the CSP that has an optimisation flavour. For a comprehensive introduction to Valued CSPs, we refer the reader to [KŽ17a].

In the VCSP, instead of relations we consider valued relations (also known as cost functions) – mappings that assign to tuples rational or positive infinite costs. That is, each valued relation R of arity r over a finite domain D is a function from D^r to $\mathbb{Q} \cup \{\infty\}$, and an instance of the VCSP is specified by a set of variables $X = \{x_1, \ldots, x_n\}$ and an objective function of the form

$$R(x_1, x_2) + R(x_3, x_1) + S(x_2, x_4, x_1) + R(x_3, x_3) + \ldots \tag{2.1}$$

where each of the addends is a valued constraint, that is, a formal expression of the form $R(\mathbf{x})$ where R is a valued relation of arity r and $\mathbf{x} \in X^r$.

In the search version of the VCSP, the task is to find an assignment $h : X \to D$ such that $R(h(x_1), h(x_2)) + R(h(x_3), h(x_1)) + \dots$ is minimal. In the decision version, the instance is such a sum together with a rational number τ and the goal is to decide whether the minimum is at most τ.

Notice that (the decision version of) the VCSP indeed generalizes the CSP since relations can be modelled by $\{0, \infty\}$-valued relations.[5] On the other hand, MaxCSP [DJKK08, MM17] – where the aim is to maximize the number of satisfied constraints given a CSP instance – is exactly the VCSP over $\{0, 1\}$-valued relational structures. The VCSP framework also includes many problems of a mixed optimization and combinatorial nature, such as the Vertex Cover Problem (see [KŽ17a]).

The algebraic approach to the Valued CSP was pioneered in [CCC+13], where the authors (based on a series of preliminary results in [CCJ06, Živ09, CCJŽ11, CŽ11]) showed that, just like in the CSP case, the complexity of the VCSP is also determined by invariance under certain operations, which are now commonly referred to as fractional polymorphisms. Using the newly established Galois connection between valued relations and weighted clones (i.e., appropriately closed sets of fractional polymorphisms), the authors were able to rederive a complete complexity classification of the VCSP on the Boolean domain, which was previously known from [CCJK06] using somewhat different techniques.

The algebraic approach was further developed in [KO15], where an analogue of the CSP dichotomy conjecture for VCSPs was formulated and the hardness direction was shown to hold. The tractability part was conformed shortly after in [KKR17], under the assumption that the corresponding CSP classes are tractable, thus reducing the VCSP dichotomy conjecture to the CSP dichotomy conjecture. The Bulatov-Zhuk theorem therefore also implies a P versus NP-complete dichotomy for VCSPs.

We note that substantial results have also been obtained for problems related to the VCSP other than its fixed-template complexity, such as a complexity classification of finite-valued CSPs [TŽ16] and of left-hand side restricted VCSPs [CRŽ22], and results on the power of specific algorithms for VCSPs [TŽ12, KTŽ15, TŽ17].

[5] The term *crisp* [CCJK06] is sometimes used in contrast to *valued* to refer to $\{0, \infty\}$-valued or classical relations.

2.2.3 Quantified Constraint Satisfaction Problems

While in this thesis we will not focus directly on the Quantified Constraint Satisfaction Problem (QCSP), having an intuition of this problem's definition and challenges will be useful in sight of Part II. For a more comprehensive survey on the (non-Boolean) QCSP, see the relatively recent [Mar17].

Recall that the CSP can alternatively be defined as the positive existential conjunctive model checking problem (see Section 2.1), i.e., the inputs are logical sentences in the fragment of first-order logic that only uses atomic formulas, the existential quantifier $\exists$, and the logical connective $\wedge$.

Now, for each of the 2^7 subsets $\mathscr{L} \subseteq \{\exists, \forall, \wedge, \vee, =, \neq, \neg\}$ (and each fixed relational model $\mathbf{A}$), we can define the corresponding model checking problem, which we call the $\mathscr{L}$-Model Checking Problem over $\mathbf{A}$ and denote $\mathscr{L}$-MC($\mathbf{A}$). Then, the Quantified CSP over a fixed template $\mathbf{A}$ corresponds exactly to the $\{\exists, \forall, \wedge\}$-MC($\mathbf{A}$), that is, the problem of deciding whether a given $\{\exists, \forall, \wedge\}$-sentence of first-order logic is satisfiable in $\mathbf{A}$.

The QCSP in its full generality is PSPACE-complete. For each (meaningful) $\mathscr{L} \subseteq \{\exists, \forall, \wedge, \vee, =, \neq, \neg\}$, the complexity of fixed-template $\mathscr{L}$-Model Checking has by now been classified [MM18] except for the QCSP (and its dual), for which a full polychotomy is still a challenging open problem.

The complexity of the fixed-template Boolean QCSP, also known as Quantified Boolean Formula, has been classified independently in [Dal97, CKS01], which gave a P versus PSPACE-complete dichotomy. The algebraic approach to the QCSP was later initiated in the work of Börner et al. [BBJK03, BBC$^+$09], who established a Galois connection between surjective polymorphisms and QCSP templates much like the renowned Inv-Pol Galois connection for the CSP. Harnessing the power of the algebraic approach, the authors were able to show the first non-trivial complexity classification for a class of non-Boolean QCSPs, that is, a trichotomy (P, NP-complete, or PSPACE-complete) for a certain class of binary QCSPs defined by graphs of permutations.

The partial results obtained thus far led researchers to believe that the

fixed-template QCSP might exhibit a P/NP-complete/PSPACE-complete trichotomy, a hypothesis that came to be known as the *Chen conjecture* [Che12] (see also [CMZ17]). In a recent breakthrough, Martin and Zhuk [ZM20] showed that the Chen conjecture fails even for the 3-element domain case, for which they provided a tetrachotomy (P, NP-complete, coNP-complete, or PSPACE-complete). Moreover, they showed that there exist templates for which the QCSP is complete in the complexity classes DP[6] and Θ_2^P.[7]

In recent unpublished work, Zhuk found a 7^{th} complexity class for which there are complete QCSPs[8] and conjectured that a heptachotomy might be the definitive picture capturing the complexity of the fixed-template QCSP.

[6] DP consists of the set of all languages that can be expressed as the intersection of a language in NP and a language in coNP. We remark that DP $\neq$ NP $\cap$ coNP, unless NP = coNP. In fact, we have that NP $\cup$ coNP $\subseteq$ DP.

[7] Θ_2^P is the class of languages recognizable by deterministic Turing machines in polynomial time with at most logarithmically many calls to an NP oracle. We have DP $\subseteq \Theta_2^P$.

[8] The class Π_2^{P} from the polynomial hierarchy containing all decision problems solvable in coNP time by a Turing machine augmented by an NP oracle. We have $\Theta_2^P \subseteq \Pi_2^{\text{P}} \subseteq$ PSPACE.

3 Preliminaries

In this chapter we set the notation and introduce all the concepts that are necessary to make this thesis self-contained.

Notation. Let $\mathbb{N}$ be the set of positive integers. For $k \in \mathbb{N}$, we set $[k] = \{1, \ldots, k\}$. We shall denote tuples in boldface. For a set A and a tuple $\mathbf{a} \in A^k$, we use either $\mathbf{a}[i]$ or, where there is no ambiguity, a_i to refer to the i^{th} entry of $\mathbf{a}$. We say that $\mathbf{a}$ has a *repetition* if there exist $i \neq j \in [k]$ such that $a_i = a_j$.

We use double curly brackets $\{\{\ldots\}\}$ to denote multisets. For a non-negative integer k, $k \cdot \{\{\ldots\}\}$ stands for the multiset obtained by multiplying the multiplicity of each element in the original multiset by k. Slightly abusing the notation, the set and the multiset of entries of a tuple $\mathbf{a}$ are denoted by $\{\mathbf{a}\}$ and $\{\{\mathbf{a}\}\}$, respectively.

For a function f on a domain containing $\{\mathbf{a}\}$, we denote by $f(\mathbf{a})$ the coordinate-wise application of f to $\mathbf{a}$, that is, $f(\mathbf{a}) = (f(a_1), \ldots, f(a_k))$.

For every tuple $\mathbf{i} = (i_1, \ldots, i_n) \in [k]^n$ we use $\pi_{\mathbf{i}}\mathbf{a}$ to denote the *projection* of $\mathbf{a}$ to $\mathbf{i}$, i.e, the tuple $(a_{i_1}, \ldots, a_{i_n})$. If $I \subseteq [k]$ we might abuse slightly notation and use $\pi_I\mathbf{a}$ to refer to $\pi_{\mathbf{i}}\mathbf{a}$ where $\mathbf{i}$ is the tuple that contains the elements of I in increasing order.

We use $\mathscr{P}(A)$ to denote the power set of a finite set A, i.e., $\mathscr{P}(A) = \{S \mid S \subseteq A\}$. Moreover, we define $\mathscr{P}_{\neq\emptyset}(A) = \mathscr{P}(A) \setminus \emptyset$.

Relational structures. A *signature* σ is a finite collection of relation symbols, each with an associated arity. We shall use $\mathrm{ar}(R)$ to denote the arity of a relation symbol R. Given a set A and a positive integer r, an r-ary *relation* over A is a subset of A^r. A *(relational) structure* $\mathbf{A}$ over σ, or simply a σ-structure, consists of a set A called the *universe* of $\mathbf{A}$, and a non-empty relation $R^{\mathbf{A}}$ of arity $\mathrm{ar}(R)$ over A for each $R \in \sigma$. In this context, we call $R^{\mathbf{A}}$ the *interpretation* of R in $\mathbf{A}$. We shall use the same boldface and standard capital letter to refer to a structure and its universe, respectively. We will sometimes use the notation $(A; R_1, \ldots, R_k)$ to denote a structure with universe A and relations $R_1, \ldots, R_k$. In this thesis, we will only deal with structures over a finite universe. We denote by $\mathcal{C}_{\mathbf{A}}$ the set of formal expressions $\{R(\mathbf{a}) \mid \mathbf{a} \in R^{\mathbf{A}}, R \in \sigma\}$. Elements of $\mathcal{C}_{\mathbf{A}}$ will be referred to as *constraints*, where $\mathbf{a}$ is also known as the *scope* of the constraint $R(\mathbf{a})$. The arity of a constraint is the arity of the associated relation symbol, and will also be denoted $\mathrm{ar}(c)$ for any $c \in \mathcal{C}_{\mathbf{A}}$. We will sometimes write $a \in R(\mathbf{a})$ to indicate that $a \in \{a_1, \ldots, a_{\mathrm{ar}(R)}\}$.

Two structures are said to be *similar* if they have the same signature.

The *union* $\mathbf{A} \cup \mathbf{B}$ of two σ-structures $\mathbf{A}$ and $\mathbf{B}$ is the structure $\mathbf{C}$ with $C = A \cup B$ and $R^{\mathbf{C}} = R^{\mathbf{A}} \cup R^{\mathbf{B}}$ for every $R \in \sigma$. The *disjoint union* of two structures $\mathbf{A}$ and $\mathbf{B}$ is the structure $\mathbf{A} \cup \mathbf{B}'$ where $\mathbf{B}'$ is obtained from $\mathbf{B}$ by renaming variables wherever necessary so that $A \cap B' = \emptyset$. We say that a structure is *connected* if it cannot be expressed as the disjoint union of two structures. We say that $\mathbf{A}$ is a *substructure* of $\mathbf{B}$ if $\mathbf{A} \cup \mathbf{B} = \mathbf{B}$. If, in addition, $R^{\mathbf{A}} = R^{\mathbf{B}} \cap A^{\mathrm{ar}(R)}$ for every $R \in \sigma$ then $\mathbf{A}$ is the substructure of $\mathbf{B}$ *induced* by A.

A *digraph* is a relational structure whose signature consists of a single binary relation, which we will call the *edge set*. A *graph* (in the standard graph-theoretic sense) can be seen as a digraph where, additionally, the edge relation is symmetric and non-reflexive. We will often represent graph edges as sets rather than ordered pairs to stress that the edge relation is symmetric. A *cycle* in a graph $(V; E)$ is a sequence $v_0, v_1, \ldots, v_k$ with $k \geq 3$ such that $\{v_{i-1}, v_i\} \in E$ for all $i \in [k]$, $v_0 = v_k$, and $\{v_1, \ldots, v_k\}$ are distinct. A *tree* is a graph that contains no cycles. A graph $(V; E)$ is said to be *bipartite* if the vertex set V can be partitioned into two classes V_1, V_2 such that for every edge $e \in E$ and every $i \in [2]$, $|e \cap V_i| = 1$.

The *Factor Graph*[1] [FPY18] $\mathrm{G}_{\mathbf{A}}$ of a structure $\mathbf{A}$ is the undirected labelled bipartite graph with vertex set $A \cup \mathcal{C}_{\mathbf{A}}$ and edge set $\{\{a, R(\mathbf{a})\} \mid a \in R(\mathbf{a})\}$. Each edge in $\mathrm{G}_{\mathbf{A}}$ that is incident to a variable a and a constraint $R(\mathbf{a})$ has a label $\ell_{\{a,R(\mathbf{a})\}} = (S, R)$ for $S = \{i \in [\mathrm{ar}(R)] \mid a_i = a\}$. We denote the set of labels $\{(S, R) \mid S \subseteq [\mathrm{ar}(R)], R \in \sigma\}$ by $\mathcal{L}_\sigma$, or simply $\mathcal{L}$ when the signature is clear from the context.

An alternative definition of connectedness, which will be useful in this thesis, is the following: a structure $\mathbf{A}$ is connected if $\mathrm{G}_{\mathbf{A}}$ is connected in the standard graph-theoretic sense. Note that this definition coincides with the notion of connectedness presented above. Similarly, we say that $\mathbf{A}'$ is a connected component of $\mathbf{A}$ if $\mathrm{G}_{\mathbf{A}'}$ is a connected component of $\mathrm{G}_{\mathbf{A}}$. Moreover, we say that a structure $\mathbf{A}$ is an *ftree* (factor tree) if its (unlabelled) factor graph is a tree. If $\mathbf{A}'$ is a substructure of $\mathbf{A}$ and $\mathbf{A}'$ is an ftree then we say that $\mathbf{A}'$ is a *subftree* of $\mathbf{A}$.

Example 3.1

We present here some examples of relations that will be mentioned at various points in this thesis.

Inequality on $\{0, \dots, k-1\}$	$\neq_k = \{\mathbf{d} \in \{0, \dots, k-1\}^2 \mid d_1 \neq d_2\}$
Inequality on D	$\neq_D = \{\mathbf{d} \in D^2 \mid d_1 \neq d_2\}$
1-in-3-SAT	1-in-3-SAT $= \{(1,0,0),(0,1,0),(0,0,1)\}$
Not-all-equal-SAT	$\mathrm{NAE} = \{0,1\}^3 \setminus \{(0,0,0),(1,1,1)\}$
r-ary not-all-equal on D	$\mathrm{NAE}_D^r = \{\mathbf{d} \in D^r \mid \neg(d_1 = \cdots = d_r)\}$
r-ary rainbow on D	$\mathrm{Rb}_D^r = \{\mathbf{d} \in D^r \mid \{d_1, \dots, d_r\} = D\}$

Notice that $\neq_2$, $\neq_{\{0,1\}}$, $\mathrm{NAE}^2_{\{0,1\}}$ and $\mathrm{Rb}^2_{\{0,1\}}$ coincide, and that NAE, $\mathrm{NAE}^3_{\{0,1\}}$ and $\mathrm{Rb}^3_{\{0,1\}}$ also coincide.

[1] We note that the notion of factor graph, although similar, differs in several ways from the incidence multigraph (see [LLT07]) as the latter allows for parallel edges.

Morphisms. Let $\mathbf{A}$, $\mathbf{B}$ be σ-structures. A *homomorphism* from $\mathbf{A}$ to $\mathbf{B}$ is a map $h : A \to B$ such that for every $R \in \sigma$ and every $\mathbf{a} \in R^{\mathbf{A}}$ it holds that $h(\mathbf{a}) \in R^{\mathbf{B}}$, where h is applied to $\mathbf{a}$ component-wise. If there exists a homomorphism from $\mathbf{A}$ to $\mathbf{B}$ we say that $\mathbf{A}$ is homomorphic to $\mathbf{B}$ and we write $\mathbf{A} \to \mathbf{B}$. We say that two structures are *homomorphically equivalent* if $\mathbf{A} \to \mathbf{B}$ and $\mathbf{B} \to \mathbf{A}$. We shall use $\hom(\mathbf{A}; \mathbf{B})$ to denote the number of homomorphisms from $\mathbf{A}$ to $\mathbf{B}$.

An *isomorphism* from $\mathbf{A}$ to $\mathbf{B}$ is a bijective map $f : A \to B$ such that for every $R \in \sigma$ and every $\mathbf{a} \in A^{\mathrm{ar}(R)}$ it holds that $\mathbf{a} \in R^{\mathbf{A}}$ if and only if $f(\mathbf{a}) \in R^{\mathbf{B}}$.

CSP and PCSP. For a relational structure $\mathbf{A}$, the (fixed-template) *CSP over* $\mathbf{A}$, denoted $\mathrm{CSP}(\mathbf{A})$, is the problem of deciding whether an input structure $\mathbf{X}$ similar to $\mathbf{A}$ is homomorphic to $\mathbf{A}$. $\mathbf{X}$ is also referred to as the *instance* and $\mathbf{A}$ as the *constraint language* or *template* in this context. When the template is clear from the context, if an instance $\mathbf{X}$ is homomorphic to the template, we also say that $\mathbf{X}$ is *satisfiable*.

Beyond the decision problem, we can define the *search problem* for $\mathrm{CSP}(\mathbf{A})$: given an input structure $\mathbf{X}$ which satisfies $\mathbf{X} \to \mathbf{A}$, find a homomorphism from $\mathbf{X}$ to $\mathbf{A}$. It is well-known that the decision and search problem for CSP have the same complexity [BJK05].

Given two σ-structures $\mathbf{A}$ and $\mathbf{B}$, the (fixed-template) *Promise CSP over* $(\mathbf{A}, \mathbf{B})$, denoted $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$, is defined as follows: given a σ-structure $\mathbf{X}$, output Yes if $\mathbf{X}$ is homomorphic to $\mathbf{A}$, and output No if $\mathbf{X}$ is not homomorphic to $\mathbf{B}$. Note that we do not impose any requirements on the algorithm in the case that $\mathbf{X}$ is neither a Yes instance nor a No instance. Alternatively, we are *promised* that the input is either a Yes instance or a No instance. Notice that $\mathrm{PCSP}(\mathbf{A}, \mathbf{A})$ is precisely $\mathrm{CSP}(\mathbf{A})$.

The PCSP is a well-defined problem if and only if the sets of Yes and No instances are disjoint. The pairs of structures $(\mathbf{A}, \mathbf{B})$ that satisfy this condition are called *PCSP templates*. It is an easy observation that this happens exactly when $\mathbf{A}$ is homomorphic to $\mathbf{B}$.

Proposition 3.2. $(\mathbf{A}, \mathbf{B})$ *is a* PCSP *template if and only if* $\mathbf{A} \to \mathbf{B}$.

Proof. For the forward direction, if we assume that $\mathbf{A}$ is not homomorphic

$$f\left(\begin{pmatrix} a_{11} \\ \vdots \\ a_{1k} \end{pmatrix}, \begin{pmatrix} a_{21} \\ \vdots \\ a_{2k} \end{pmatrix}, \ldots, \begin{pmatrix} a_{\ell 1} \\ \vdots \\ a_{\ell k} \end{pmatrix}\right) = \begin{pmatrix} f(a_{11}, \ldots, a_{\ell 1}) \\ \vdots \\ f(a_{1k}, \ldots, a_{\ell k}) \end{pmatrix} \in R$$
$$\begin{matrix} \Cap & \Cap & & \Cap \\ R & R & & R \end{matrix}$$

Figure 3.1: The component-wise application of a polymorphism.

to $\mathbf{B}$, then $\mathbf{A}$ is both a Yes instance and a No instance, leading to a contradiction. For the backwards direction, it is enough to notice that homomorphisms compose. □

Polymorphisms. Let $R \subseteq A^r$ be an r-ary relations. A k-ary *polymorphism* of R is an operation $f : A^k \to A$ such that the coordinate-wise application of f to any list of k tuples from R gives a tuple in R. More formally, f is a polymorphism of R if for any $\mathbf{a}_1, \ldots, \mathbf{a}_k \in R$ we have that $(f(a_{11}, \ldots, a_{\ell 1}), \ldots, f(a_{1k}, \ldots, a_{\ell k})) \in R$, where a_{ij} denotes the j^{th} entry of $\mathbf{a}_i$ (see Figure 3.1).

We say that a function f is a polymorphism of a σ-structure $\mathbf{A}$ if f is a polymorphism of $R^{\mathbf{A}}$ for all $R \in \sigma$. Equivalently, we say that $\mathbf{A}$ is *invariant* under f, or that f *preserves* $\mathbf{A}$. Note that a unary polymorphism of $\mathbf{A}$ is just an endomorphism of $\mathbf{A}$. The set of polymorphisms of $\mathbf{A}$ will be denoted $\mathsf{Pol}(\mathbf{A})$.

Example 3.3 (Power structure).

Given a σ-structure $\mathbf{A}$ and a positive integer k, the k^{th} *power structure* of $\mathbf{A}$ is the σ-structure $\mathbf{A}^k$ with universe A^k and such that for every $R \in \sigma$ of arity r and for all $\mathbf{a}_1, \ldots, \mathbf{a}_k \in R^{\mathbf{A}}$ we have $((a_{11}, \ldots, a_{k1}), \ldots, (a_{1r}, \ldots, a_{kr})) \in R^{\mathbf{A}^k}$, where as above a_{ij} denotes the j^{th} entry of $\mathbf{a}_i$. It follows easily that for every $f : A^r \to A$, f is a homomorphism from $\mathbf{A}^k$ to $\mathbf{A}$ if and only if f is a k-ary polymorphism of $\mathbf{A}$.

A k-ary operation $f : A^k \to A$ is said to be *symmetric* if for all $a_1, \ldots, a_k \in A$ and all permutations ρ on $[k]$ we have that $f(a_1, \ldots, a_k) = f(a_{\rho(1)}, \ldots, a_{\rho(k)})$. An operation k-ary operation $f : A^k \to A$ is said to be *totally symmetric* (also called a *set function*) if its output only depends on the set of variables in the argument, i.e., if there exists a function $g : \mathscr{P}_{\neq\emptyset}(A) \to A$ such that, for each $\mathbf{a} \in A^k$, $f(\mathbf{a}) = g(\{\mathbf{a}\})$. Finally, a k-ary ($k \geq 2$) operation $f : A^k \to A$ is said to be *weak near-unanimity* if for all $x, y \in A$ it holds that

$$f(y, x, \ldots, x) = f(x, y, \ldots, x) = \ldots = f(x, x, \ldots, y).$$

Example 3.4 (Inequality relation).

Consider the inequality relation $\neq_2 = \{(0,1),(1,0)\}$ on the Boolean domain. It is easy to see that the ternary minority operation f given by $f(x, y, z) = x \oplus y \oplus z$ is a polymorphism of $\neq_2$. On the other hand, one can show that R does not have symmetric polymorphisms of arity 2. In particular, let $\mathbf{t}_1 = (0,1)$ and $\mathbf{t}_2 = (1,0)$. Since a symmetric binary operation f needs to satisfy $f(0,1) = f(1,0)$, the coordinate-wise application of f to $\mathbf{t}_1, \mathbf{t}_2$ would yield a reflexive tuple, which cannot possibly belong to $\neq_2$.

Primitive positive definability. Given a σ-structure $\mathbf{A}$ with domain A, a relation $R \subseteq A^k$ is said to be *primitive positive definable*, most commonly shortened to *pp-definable*, from $\mathbf{A}$ if there exists an existential conjunctive first-order formula $\psi(x_1, \ldots, x_k)$ which uses only relation symbols from σ and equality, such that for all $(a_1, \ldots, a_k) \in A^k$, $(a_1, \ldots, a_k) \in R$ if and only if ψ is satisfied when each x_i, $i \in [k]$ is evaluated as a_i, and the relation symbols in ψ are interpreted as in $\mathbf{A}$.

Alternatively, pp-definitions can be seen in the following equivalent way. A relation $R \subseteq A^k$ is pp-definable from a σ-structure $\mathbf{A}$ if there exists a pair $((x_1, \ldots, x_k), \mathbf{X})$ where $\mathbf{X}$ is a $\sigma \cup \{=\}$-structure and $x_1, \ldots, x_k$ are distinct variables in X such that for every tuple $(a_1, \ldots, a_k) \in A^k$, $(a_1, \ldots, a_k) \in R$ if and only if there exists a homomorphism $h : X \to A$ such that $a_i = h(x_i)$ for all $i \in [k]$.

A structure $\mathbf{A}'$ is pp-definable from $\mathbf{A}$ if all the relations in $\mathbf{A}'$ can be pp-defined from $\mathbf{A}$ (notice that $\mathbf{A}$ and $\mathbf{A}'$ do not necessarily have the same signature).

Example 3.5 (pp-definitions).

Consider the inequality relation $\neq_2$ from example 3.4. A pp-definition of $\neq_2$ from $\mathbf{A} = (\{0,1\}; \mathrm{NAE})$ is

$$\neq_2 (x, y) = \exists z[\mathrm{NAE}(x, y, z) \wedge (x = z)].$$

Part I: Weisfeiler-Leman Invariant and Distributed CSPs

This is a major theme in mathematics: things are what you want them to be. You have endless choices; there is no reality to get in your way.

Lockhart's Lament

4 Introduction

The motivation for this work stems from questions that are seemingly distant from the world of constraint satisfaction: we would like to investigate the computational power of distributed algorithms from a theoretical standpoint. Under what conditions is distributed computation guaranteed to converge to a solution in finite time? And in the case that we do have this guarantee, how fast can we expect the computation to converge? These questions, while very natural and relevant to practitioners, seem to have received little attention in the computer science theory literature.

To address these questions, in Chapter 6 we begin our journey with an exploration of the computational complexity of solving constraint satisfaction problems via distributed algorithms. We consider a synchronous, anonymous network of deterministic processes, each of which controls either a variable or a constraint of the CSP, and ask for which templates $\mathbf{A}$ is the distributed CSP over $\mathbf{A}$ tractable. We obtain a surprising dichotomy: there is a (polynomial-time) distributed algorithm that solves a certain CSP if and only if the template is invariant under symmetric polymorphisms of every arities. What is more, this is a *computability* dichotomy rather than a *complexity* dichotomy: in the case that there are no polynomial-time algorithms for the distributed CSP, we can show that there are in fact no algorithms at all.

In fact, the connection between invariance under polymorphisms and solvability by certain algorithms has been studied extensively in the constraint satisfaction community. In particular, the algebraic condition that defines the borderline of tractability for the distributed CSP is precisely

the same condition that corresponds to solvability by a very natural linear program. This connection with linear programming also arises independently in our work on distributed CSP, as the proof of one of the main results relies on some theoretical properties of an approximation method for linear programs.

On the other hand, message passing in distributed networks can be seen as the natural setting in which to run a graph vertex partitioning procedure known as the 1-dimensional Weisfeiler-Leman algorithm. In fact, it is not difficult to show that a necessary condition for the distributed CSP to be tractable is that it is closed under the equivalence induced by this algorithm, in the sense that if two instances satisfy this equivalence condition, it must be the case that they are either both satisfiable or both unsatisfiable. This simple fact allows us to draw the first of the many connections between linear programs and the Weisfeiler-Leman method, which will be the leitmotif of this first part of the thesis.

Inspired by this, in Chapter 7 we explore this correspondence further from a different viewpoint. The Weisfeiler-Leman equivalence relation mentioned above has an equivalent characterization in terms of a well-studied linear programming relaxation for the graph isomorphism problem, known as fractional isomorphism. The starting point for this new side of the correspondence is the observation of a similarity between the algebraic formulation of fractional isomorphism, and that of the aforementioned natural linear program for the CSP. We build on this similarity to obtain a decomposition of the feasibility of the latter linear program in terms of the former, attained by means of a CFI-type construction. This allows us to add two more equivalent conditions to the characterization of CSPs solvable by the linear programming relaxation: in terms of invariance under the Weisfeiler-Leman equivalence, and in terms of solvability by the class of distributed algorithms studied in Chapter 6.

In Chapter 8 we take this connection further and show that it extends to the much more general framework of Promise Valued CSPs. Promise Valued CSPs generalize both Promise CSPs and Valued CSPs, and are broad enough to include all constant factor approximation problem, hence the results in this chapter might be of interest also to researchers outside the strict CSP community. Along the way, we delve into the world of fractional operations, the study of which has been pivotal in the develop-

ment of an algebraic theory of valued constraints. While the connection that is the center of this part of the thesis does – perhaps as expected or at least as desired – extend to the promise valued setting, the study of solvability by linear programs in the context of PVCSPs leads us to a surprising discovery. The linear programming relaxation for the CSP that we already mentioned several times in this chapter has two versions which for all practical purposes are equivalent in terms of solving CSPs, as well as PCSPs and (finite-valued) VCSPs. It turns out that this is no longer the case in the realm of the PVCSP: we can show that one version of the relaxation is strictly stronger than the other. This suggests that the fine difference between how these two versions of the relaxation work might have so far been overlooked in the literature.

Finally, in Chapter 9 we lift the by now established connection between the Weisfeiler-Leman method and linear programming to higher dimensions. This is done in the spirit of previous work that shows that the power of higher-dimensional levels of the Weisfeiler-Leman method interleave with the levels of the well-known Sherali-Adams hierarchy (applied to fractional isomorphism) in terms of distinguishing non-isomorphic graphs. Nonetheless, the connection between these two hierarchies in the context of homomorphism of relational structures (i.e., CSPs), is new to this work. In particular, we are able to lift the results of Chapter 7 to higher levels of the Sherali-Adams hierarchy (applied to our natural linear program for CSPs) and give an analogous decomposition in terms of fractional isomorphism.

Moreover, we define a higher-dimensional notion of equivalence for relational structures in the style of Weisfeiler and Leman. This definition might not resemble what one would expect if one were to naively extend the definition of higher-dimensional Weisfeiler-Leman equivalence that is commonly used in the context of graphs. This difference is due to obstacles in dealing with relations of large arity that trivially do not occur when one restricts one's focus to graphs only. However, when the dimension is large enough, our notion of equivalence coincides with the natural extension of the well-studied notion for graphs. In order to characterize it exactly, we touch upon a variety of topics, such as distinguishability in fixed-variable logics with counting and homomorphism counts from structures of bounded treewidth, and discuss their connection to the Weisfeiler-

Leman method.

The next chapter is meant as an introduction to many of the technical topics that we will need in the subsequent chapters. In particular, we will survey three types of relaxations for decision problems, that is, heuristics that exhibit one-sided errors in that they always accept Yes instances, but do not always reject No instances. These are the already mentioned Weisfeiler-Leman method for isomorphism (Section 5.1) and linear programming methods for CSPs (Section 5.2), as well as local consistency methods for CSPs (Section 5.3). While we have not mentioned the latter methods in this introduction, they are also relevant to our work as they lend themselves particularly well to being adapted to the distributed setting.

5 Relaxation techniques

In this chapter we survey three incomplete methods to test for isomorphism (Section 5.1) and homomorphism (Sections 5.2, 5.3) of graphs and relational structures. Each of these methods will find plenty of applications in the remainder of this thesis.

5.1 Combinatorial relaxations of isomorphism

We start by describing a number of relaxations for the graph isomorphism problem in graphs, and then we will see how to extend these methods to arbitrary relational structures. The following definitions will be useful.

A matrix M of non-negative real numbers is said to be *left* (resp. *right*) *stochastic* if all its columns (resp. rows) sum to 1. Note that we do not require M to be square. A *doubly stochastic matrix* is a square matrix that is both left and right stochastic.

5.1.1 Fractional isomorphism of graphs

Recall that two graphs are isomorphic iff there exists an edge-preserving bijection between their vertex sets. Despite an important research effort, it is still an open problem to determine whether the isomorphism problem – that is, problem of deciding whether two given graphs are isomorphic – can be solved in polynomial time. The recent quasipolynomial algorithm for the problem presented by Babai [Bab16] is widely regarded as a major breakthrough in theoretical computer science.

Now, the isomorphism problem for graphs G and H can be reformulated as an integer program which asks whether there exists a permutation matrix P such that $PN_G = N_H P$, where N_G and N_H are the adjacency matrices of G and H respectively. If we relax this condition to only require that P is doubly stochastic, that is, we drop the integrality constraint, we obtain a linear program known as *fractional isomorphism*. It turns out that fractional isomorphism has a number of equivalent characterizations, which we introduce next.

In a graph G, the degree of a vertex v is the number of edges incident to v. The zeroth iterated degree of v is equal to its degree. For $j \geq 1$, the j^{th} iterated degree of v is the multiset of $(j-1)^{\text{th}}$ degrees of v's neighbours in G. Then, the iterated degree[1] of a vertex v is given by the list of its j^{th} iterated degrees for $k \geq 0$, and the iterated degree sequence of a graph G is the multiset of iterated degrees of the vertices of G.[2]

One can think of the j^{th} iterated degree of a vertex as the "local view" of G from v. That is, the j^{th} iterated degree of v can be seen as the rooted tree T_v of depth j constructed inductively in such a way that there is a map γ from the vertices of T_v to the vertices of G which maintains the following properties: the root is mapped by γ to v, and for every t in T_v, the set of children of t is mapped bijectively to the neighbours of $\gamma(t)$. Then, two vertices have the same iterated degree if the corresponding infinite trees constructed via this procedure are isomorphic (see Figure 5.1).

The *colour refinement algorithm*, also known as *1-dimensional Weisfeiler-Leman algorithm* (1-WL) [LW68][3], is the procedure that calculates the iterated degree sequence of a graph (see for instance [GKMS17]). Thanks to the observation that only a linear number of iterations is needed to reach a stable point, 1-WL is often used as a simple heuristic for the graph isomorphism problem: if two graphs are isomorphic then they must

[1] Sometimes referred to as the *ultimate* iterated degree [SU11].

[2] The degree sequence is often defined to be a list. However, when looking at iterated degree it is common [RSU94, SU11] and more practical to use multisets instead of lists, while maintaining the terminology *sequence* to highlight that we are dealing with a generalisation of the classical concept of degree sequence.

[3] To be precise, we point out that the algorithm proposed in the original paper of Weisfeiler and Leman [LW68] is equivalent to what is known as the 2-dimensional Weisfeiler-Leman algorithm. This will be defined in its general (k-dimensional) version in Chapter 9.

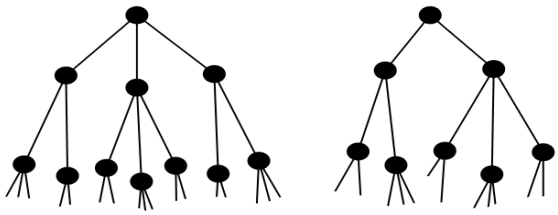

Figure 5.1: The first three levels of the "local view" of the red nodes (left) and blue nodes (right) of the graphs in Figure 5.2.

have the same iterated degree sequence, but the opposite is not true. We say that 1-WL distinguishes two graphs G and H if their iterated degree sequences differ. Famously, 1-WL distinguishes almost all non-isomorphic graphs [BK79, BES80]. However, it also fails on some very simple instances (see for example Figure 5.2). To address these limitations, a hierarchy of higher-dimensional versions of this algorithm (k-WL for $k > 1$) was proposed, see Chapter 9 for details.

An alternative characterization of fractional isomorphism is in terms of the notion of equitable partition. Let $G = (V; E)$ be a graph and $\boldsymbol{\mathcal{P}} = \{\mathcal{P}_i \mid i \in I\}$ be a partition of V. $\boldsymbol{\mathcal{P}}$ is said to be *equitable* if for every $i, j \in I$ there exists an integer $c_{i,j}$ such that for every $v \in \mathcal{P}_i$ we have

$$|N(v) \cap \mathcal{P}_j| = c_{i,j}$$

where $N(v)$ denotes the neighbourhood of v in G. The integers $c_{i,j}$ are called collectively the *parameters* of the partition. Two graphs G, H are said to have a *common equitable partition* if there exist equitable partitions $\{\mathcal{P}_i^G \mid i \in I\}$ of G and $\{\mathcal{P}_i^H \mid i \in I\}$ and H with the same parameters satisfying $|\mathcal{P}_i^G| = |\mathcal{P}_i^H|$ for all $i \in I$.

It turns out that all of the concepts introduced above are equivalent for the purpose of distinguishing non-isomorphic graphs. In fact, much more is true. Let $\mathscr{C}$ denote the first-order logic enriched with counting quantifiers - that is, objects of the form $\exists^n x$ for each positive integer n, where the meaning of a sentence $\exists^n x \varphi(x)$ is "there exist at least n objects x that satisfy φ"; and let $\mathscr{C}^k$ denote the k-variable fragment of $\mathscr{C}$. Then, we have the following characterization of fractional isomorphism:

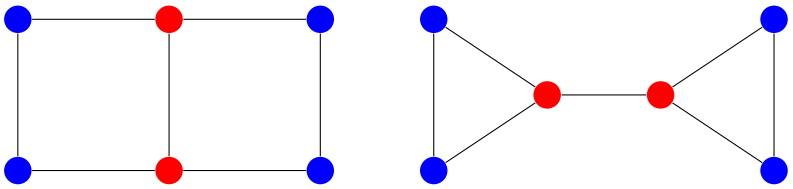

Figure 5.2: It is easy to see that the graphs depicted above have the same iterated degree sequence. In particular, the equivalence classes are given by the distinct colours, red and blue. However, the graphs are clearly not isomorphic, since the left graph is bipartite while the right graph is not.

Theorem 5.1. *Let G, H be graphs. The following are equivalent:*

1. *There exists a doubly stochastic matrix P such that $PN_G = N_H P$;*
2. *G and H have the same iterated degree sequence;*
3. *G and H have a common equitable partition;*
4. *G and H satisfy the same formulae in the logic $\mathscr{C}^2$;*
5. *$\hom(T;G) = \hom(T;H)$ for all trees T.*

The equivalence of (1), (2) and (3) can be traced back to the work of Leighton [Lei82], Tinhofer [Tin86, Tin91], and Ramana, Scheinerman and Ullman [RSU94]. The equivalence of (2) and (4) is due to Cai, Fürer and Immerman [CFI92] (in fact, in the same paper a much stronger result is proved, see Section 9.1). Item (5) was added later independently by Dvorák [Dvo10] and Dell, Grohe and Rattan [DGR18].

5.1.2 From graphs to relational structures

We now extend each of the notions defined above in the context of graphs to arbitrary relational structures. We start by adapting the the 1-dimensional Weisfeiler-Leman algorithm to calculate iterative refinements of a colouring of the universe and constraint set of a relational structure. While there are syntactical differences, when run on graphs this procedure is equivalent for all purposes to 1-WL.

In what follows it will be convenient to allow disconnected instances. Consider the labelled factor graph $\mathrm{G}_{\mathbf{X}}$ of a relational structure $\mathbf{X}$, which we defined in Chapter 3. Let v be a node of $\mathrm{G}_{\mathbf{X}}$ and denote its neighbourhood in the factor graph by $N(v)$. For every $j \geq 0$ and $v \in X \cup \mathcal{C}_{\mathbf{X}}$, we define inductively the j^{th} *iterated degree* $\delta_j^{\mathbf{X}}(v)$ of v on $\mathbf{X}$ as follows. We set $\delta_0^{\mathbf{X}}(v)$ to be one of two arbitrary symbols that distinguish elements of X from elements of $\mathcal{C}_{\mathbf{X}}$. For $j \geq 1$ we set $\delta_j^{\mathbf{X}}(v) = \{\{(\ell_{\{v,w\}}, \delta_{j-1}^{\mathbf{X}}(w)) \mid w \in N(v)\}\}$, and the *iterated degree* of v is defined accordingly as $\delta^{\mathbf{X}}(v) = (\delta_0^{\mathbf{X}}(v), \delta_1^{\mathbf{X}}(v), \delta_2^{\mathbf{X}}(v), \ldots)$. For vertices v and v' we write $v \equiv_1 v'$ if $\delta^{\mathbf{X}}(v) = \delta^{\mathbf{X}}(v')$. In this case, we say that v and v' are $\equiv_1$-*equivalent*.[4] It can be shown (see the following proposition) that as j increases, the partition induced by $\delta_j^{\mathbf{X}}$ gets more refined (or remains unchanged), and indeed it reaches a fixed point for some $j \leq 2n$ where $n = |X|$.

Proposition 5.2. *Let $\mathbf{X}$ be a relational structure and $v, v' \in X \cup \mathcal{C}_{\mathbf{X}}$. Let $j \geq 2n$ where $n = |X|$. Then, $\delta_j^{\mathbf{X}}(v) = \delta_j^{\mathbf{X}}(v')$ implies $v \equiv_1 v'$.*

Proof. We start by showing that for all non-negative integers j, j' with $j \leq j'$, the partition induced by $\delta_{j'}^{\mathbf{X}}$ on $X \cup \mathcal{C}_{\mathbf{X}}$ is at least as refined as the partition induced by $\delta_j^{\mathbf{X}}$. The proof goes by induction. Let v, v' be arbitrary nodes in $\mathrm{G}_{\mathbf{X}}$. Clearly if $\delta_0^{\mathbf{X}}(v) \neq \delta_0^{\mathbf{X}}(v')$, then $\delta_j^{\mathbf{X}}(v) \neq \delta_j^{\mathbf{X}}(v')$ for all $j \in \mathbb{N}$, so in particular $\delta_1^{\mathbf{X}}(v) \neq \delta_1^{\mathbf{X}}(v')$. Now assume that for all $v, v' \in X \cup \mathcal{C}_{\mathbf{X}}$, $\delta_j^{\mathbf{X}}(v) = \delta_j^{\mathbf{X}}(v')$ implies $\delta_{j-1}^{\mathbf{X}}(v) = \delta_{j-1}^{\mathbf{X}}(v')$. Then it is a clear consequence of the definition of $\delta_j^{\mathbf{X}}$ that $\delta_{j+1}^{\mathbf{X}}(v) = \delta_{j+1}^{\mathbf{X}}(v') \Rightarrow \delta_j^{\mathbf{X}}(v) = \delta_j^{\mathbf{X}}(v')$ for all $v, v' \in X \cup \mathcal{C}_{\mathbf{X}}$ too as required.

Now it remains to show that if $\delta_{2n}^{\mathbf{X}}(v) = \delta_{2n}^{\mathbf{X}}(v')$, then $\delta_j^{\mathbf{X}}(v) = \delta_j^{\mathbf{X}}(v')$ for all $j \geq 2n$. The result is immediate if we replace $2n$ by $n + m$. To achieve $2n$ we use the fact that the factor graph is bipartite. Denote by $\boldsymbol{\mathcal{P}}^j$ and $\boldsymbol{\mathcal{Q}}^j$ the partitions induced by $\delta_j^{\mathbf{X}}$ on X and $\mathcal{C}_{\mathbf{X}}$ respectively and note that if $\boldsymbol{\mathcal{P}}^{j-2} = \boldsymbol{\mathcal{P}}^j$ then $(\boldsymbol{\mathcal{P}}^j, \boldsymbol{\mathcal{Q}}^j)$ is a fixed point. We notice that $\boldsymbol{\mathcal{P}}^{j-2} = \boldsymbol{\mathcal{P}}^j$ must occur for some $j \leq 2n$ and we are done. □

The *iterated degree sequence* of a relational structure $\mathbf{X}$ is defined as $\delta(\mathbf{X}) = \{\{\delta^{\mathbf{X}}(v) \mid v \in X \cup \mathcal{C}_{\mathbf{X}}\}\}$; for two σ-structures $\mathbf{X}$, $\mathbf{Y}$, we write $\mathbf{X} \equiv_1 \mathbf{Y}$ if they have the same iterated degree sequence. Notice that in

[4] The "1" refers to "1-dimensional".

order to prove that $\mathbf{X} \equiv_1 \mathbf{Y}$ it is sufficient to show that $\{\{\delta^{\mathbf{X}}(x) \mid x \in X\}\} = \{\{\delta^{\mathbf{Y}}(y) \mid x \in Y\}\}$.

Here we deviate for a moment from the analogy with the presentation for graphs to introduce the notion of weak congruence. We say that two σ-structures $\mathbf{X}$ and $\mathbf{Y}$ are *weakly congruent* if $|Y| \cdot \delta(\mathbf{X}) = |X| \cdot \delta(\mathbf{Y})$. The following observation will be useful in the proofs of the main theorems of Chapters 7 and 8 (respectively, 7.2 and 8.7).

Remark 5.3 (Weakly congruent connected components).

It is a simple consequence of the definition of iterated degree that for any two (possibly disconnected) σ-structures $\mathbf{X}$ and $\mathbf{Y}$ with $\mathbf{X} \equiv_1 \mathbf{Y}$ and any two connected components $\mathbf{X}'$ of $\mathbf{X}$ and $\mathbf{Y}'$ of $\mathbf{Y}$, either the iterated degree sequences $\delta(\mathbf{X}')$ and $\delta(\mathbf{Y}')$ are disjoint, or $\mathbf{X}'$ and $\mathbf{Y}'$ are weakly congruent.

Indeed, for a "variable vertex" $x \in X$ of $\mathrm{G}_{\mathbf{X}}$, a label $\ell \in \mathcal{L}_\sigma$, and a "constraint vertex" $c \in \mathcal{C}_{\mathbf{X}}$ such that x and c are adjacent in $\mathrm{G}_{\mathbf{X}}$, denote $x[\ell, c] = \{c' \mid \delta^{\mathbf{X}}(c') = \delta^{\mathbf{X}}(c), \ell_{\{x,c'\}} = \ell\}$. Observe that if there exists an edge between x and c labeled ℓ, then $\{x'[\ell, c] \mid \delta^{\mathbf{X}}(x') = \delta^{\mathbf{X}}(x)\}$ is a collection of mutually disjoint sets of equal size, which covers $\{c' \mid \delta^{\mathbf{X}}(c) = \delta^{\mathbf{X}}(c')\}$; and, moreover, the same claim holds when x' and c' are restricted to the connected component containing x (or c). The claim easily follows.

Most of the results in Chapters 6, 7 and 8 are phrased in terms of iterated degree, as this is the formulation that better lends itself to be used in the context of distributed algorithms. However, we can lift the equivalent characterizations of the 1-dimensional Weisfeiler-Leman algorithm from graphs to relational structures. In order to do so, we need to define some more concepts.

For a signature σ, let $\mathcal{L}_\sigma := \{(S, R) \mid R \in \sigma, S \subseteq [\mathrm{ar}(R)]\}$ be the set of possible labels for the edges of the factor graph of a σ-structure. We construct the *matrix representation* $M_{\mathbf{X}}$ of a σ-structure $\mathbf{X}$ as follows (it will be convenient to assume that the indices of the rows and columns of a matrix are arbitrary sets). $M_{\mathbf{X}}$ is an $X \times \mathcal{C}_{\mathbf{X}}$ matrix whose entries are elements of $\mathcal{L}_\sigma$. In particular, for all $x \in X$ and $R(\mathbf{x}) \in \mathcal{C}_{\mathbf{X}}$, we have that

$M_{\mathbf{X}}[x, R(\mathbf{x})] = (S, R)$ where $S = \{i \in [\mathrm{ar}(R)] \mid x = \mathbf{x}[i]\}$.

In a nutshell we are lifting from graph isomorphism to matrix isomorphism, where two matrices are isomorphic if they are identical modulo a permutation of the rows and columns. To formalize this, it will be convenient to associate $\mathbf{X}$ with a set of 0-1 incidence matrices. In particular, for every $\ell \in \mathcal{L}_\sigma$ we define $M^\ell_{\mathbf{X}} \in \{0,1\}^{X \times \mathcal{C}_{\mathbf{X}}}$ as follows: $M^\ell_{\mathbf{X}}[x, R(\mathbf{x})] = 1$ if $M_{\mathbf{X}}[x, R(\mathbf{x})] = \ell$ and $M^\ell_{\mathbf{X}}[x, R(\mathbf{x})] = 0$ otherwise.

We are now in the position to present the adaptation of the notion of equitable partition to relational structures. A *partition* of a σ-structure $\mathbf{A}$ is a pair $(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{Q}})$ where $\boldsymbol{\mathcal{P}} = \{\mathcal{P}_i \mid i \in I\}$ is a partition of A and $\boldsymbol{\mathcal{Q}} = \{\mathcal{Q}_j \mid j \in J\}$ is a partition of $\mathcal{C}_{\mathbf{A}}$. We say that $(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{Q}})$ is *equitable* if for every $i \in I$, $j \in J$, and $\ell \in \mathcal{L}_\sigma$, there are integers $c^\ell_{i,j}, d^\ell_{j,i}$, called the *parameters* of the partition, such that for every $i \in I$, every $a \in \mathcal{P}_i$, every $\ell \in \mathcal{L}_\sigma$, and every $j \in J$, we have

$$|\{R(\mathbf{a}) \in \mathcal{Q}_j \mid M_{\mathbf{A}}[a, R(\mathbf{a})] = \ell\}| = c^\ell_{i,j} \tag{P1}$$

and, similarly, for every $j \in J$, every $R(\mathbf{a}) \in \mathcal{Q}_j$, every $\ell \in \mathcal{L}_\sigma$, and every $i \in I$ we have

$$|\{a \in \mathcal{P}_i \mid M_{\mathbf{A}}[a, R(\mathbf{a})] = \ell\}| = d^\ell_{j,i}. \tag{P2}$$

As above, we say that two structures $\mathbf{A}$, $\mathbf{B}$ have a *common equitable partition* if there are equitable partitions $(\{\mathcal{P}^{\mathbf{A}}_i \mid i \in I\}, \{\mathcal{Q}^{\mathbf{A}}_j \mid j \in J\})$ and $(\{\mathcal{P}^{\mathbf{B}}_i \mid i \in I\}, \{\mathcal{Q}^{\mathbf{B}}_j \mid j \in J\})$ of $\mathbf{A}$ and $\mathbf{B}$ with the same parameters satisfying $|\mathcal{P}^{\mathbf{A}}_i| = |\mathcal{P}^{\mathbf{B}}_i|$ for every $i \in I$ and $|\mathcal{Q}^{\mathbf{A}}_j| = |\mathcal{Q}^{\mathbf{B}}_j|$ for every $j \in J$. Here we point out that, if $\mathbf{A}$ and $\mathbf{B}$ are connected, the notion of weak congruence introduced above can be alternatively rephrased as the existence of a partition $(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{Q}})$ that satisfies conditions (P2) and (P1), but where we do not have any requirements on the sizes of the partition classes.

Then, we have the following theorem, which generalizes the equivalence between the algebraic and combinatorial characterizations of fractional isomorphism for graphs from Theorem 5.1 to arbitrary relational structures. The proof is deferred to Chapter 9.

Theorem 5.4. *Let* $\mathbf{A}$, $\mathbf{B}$ *be* σ*-structures. The following are equivalent:*

1. *There exist doubly stochastic matrices* P, Q *such that for every* $\ell \in \mathcal{L}_\sigma$ *it holds that* $PM^\ell_{\mathbf{A}} = M^\ell_{\mathbf{B}}Q$ *and* $M^\ell_{\mathbf{A}}Q^T = P^T M^\ell_{\mathbf{B}}$*;*

2. $\mathbf{A}$ *and* $\mathbf{B}$ *have the same iterated degree sequence;*

3. $\mathbf{A}$ *and* $\mathbf{B}$ *have a common equitable partition;*

4. $\hom(\mathbf{T};\mathbf{A}) = \hom(\mathbf{T};\mathbf{B})$ *for all* σ*-ftrees* $\mathbf{T}$.

If, additionally, $\mathbf{A}$ *and* $\mathbf{B}$ *are graphs, then the following is also equivalent:*

6. *There exists a doubly stochastic matrix* P *such that* $PN_{\mathbf{A}} = N_{\mathbf{B}}P$.

Then, we can write $\mathbf{X} \equiv_1 \mathbf{Y}$ if $\mathbf{X}$ and $\mathbf{Y}$ satisfy any of the conditions of Theorem 5.4. Notice that $\equiv_1$ is clearly an equivalence relation.

Note in particular that condition (6) shows that our definition of $\equiv_1$ coincides with the standard notion of fractional isomorphism when $\mathbf{A}$ and $\mathbf{B}$ are graphs.

We say that a decision problem $\mathcal{D}$ is *invariant* (or *closed*) under an equivalence relation $\sim$ defined on the set of instances of $\mathcal{D}$ if for any two instances I_1 and I_2 of $\mathcal{D}$, if $I_1 \sim I_2$ then I_1 and I_2 are either both Yes instances or both No instances of $\mathcal{D}$.

Then in particular, we have that for any σ-structure $\mathbf{A}$, CSP($\mathbf{A}$) is invariant under $\equiv_1$ iff for any two σ-structures $\mathbf{X}_1$ and $\mathbf{X}_2$, $\mathbf{X}_1 \equiv_1 \mathbf{X}_2$ implies that $\mathbf{X}_1 \to \mathbf{A}$ if and only if $\mathbf{X}_2 \to \mathbf{A}$. We shall call these families of CSPs *Weisfeiler-Leman invariant* to stress the connection of $\equiv_1$ with the Weisfeiler-Leman method.

5.2 Linear Programming

Let us now turn our attention to linear programming relaxations of the homomorphism problem. These have been used intensively in the more general setting of the Constraint Satisfaction Problem and, most usually, in the approximation of its optimization versions such as MaxCSP, among other variants. For instance, a simple linear programming relaxation yields a 2-approximation algorithm for the Vertex Cover problem, and no better polynomial time approximation algorithm is known [KR08, KMTV09].

Given a pair of σ-structures $\mathbf{X}$ and $\mathbf{A}$, there is an LP relaxation known as the *basic LP relaxation* (see for example [KOT+12]), denoted BLP($\mathbf{X}, \mathbf{A}$), which is defined as follows. It has a variable $p_x(a)$ for every

$x \in X$ and $a \in A$, and a variable $p_{R(\mathbf{x})}(\mathbf{a})$ for every $R(\mathbf{x}) \in \mathcal{C}_{\mathbf{X}}$ and every $\mathbf{a} \in A^{\mathrm{ar}(R)}$. All variables must take values in the range $[0, 1]$. The value of $p_x(a)$ is interpreted as the probability that x is assigned to a, and similarly, the value of $p_{R(\mathbf{x})}(\mathbf{a})$ is interpreted as the probability that the $\mathbf{x}$ is assigned component-wise to $\mathbf{a}$. The variables are restricted by the following equations:

$$\sum_{a \in A} p_x(a) = 1 \qquad x \in X \tag{BLP1}$$

$$p_x(a) = \sum_{\mathbf{a} \in A^{\mathrm{ar}(R)},\ a_i = a} p_{R(\mathbf{x})}(\mathbf{a}) \qquad a \in A, R(\mathbf{x}) \in \mathcal{C}_{\mathbf{X}},\ i \in [\mathrm{ar}(R)] \text{ such that } x_i = x \tag{BLP2}$$

$$p_{R(\mathbf{x})}(\mathbf{a}) = 0 \qquad R(\mathbf{x}) \in \mathcal{C}_{\mathbf{X}},\ \mathbf{a} \in A^{\mathrm{ar}(R)} \text{ such that } R(\mathbf{a}) \notin \mathcal{C}_{\mathbf{A}} \tag{BLP3}$$

Intuitively, equations (BLP1)–(BLP3) ensure that, for each $R(\mathbf{x}) \in \mathcal{C}_{\mathbf{X}}$, the values of $p_{R(\mathbf{x})}(\mathbf{a})$ form a probability distribution on $R^{\mathbf{A}}$ (which is additionally consistent with $p_x(a)$'s).

5.2.1 The Sherali–Adams hierarchy for CSPs

Similarly to the Weisfeiler-Leman hierarchy for fractional isomorphism, any LP relaxation of an integer program can be further strengthened by sequentially applying so-called lift-and-project methods from mathematical programming in order to obtain a hierarchy of increasingly tighter relaxations which can still be solved efficiently. The main idea behind these is to add auxiliary variables and valid inequalities to an initial relaxation of a 0/1 integer program. These methods, which include Lovász-Schrijver [LS91] and Sherali-Adams [SA90], have been used to study classical problems in combinatorial optimization such as Max-Cut, Vertex Cover, Maximal Matching, among many others.

We consider relaxations arising from the application of the Sherali-Adams (SA) method. In this presentation we will follow [AM13]. Let $\mathbb{P} \subseteq [0, 1]^n$ be a polytope $\{\mathbf{y} \in \mathbb{R}^n : M\mathbf{y} \geq \mathbf{b}, 0 \leq \mathbf{y} \leq 1\}$ for a matrix $M \in \mathbb{R}^{m \times n}$, and a column vector $\mathbf{b} \in \mathbb{R}^m$. We denote the convex hull of the $\{0, 1\}$-vectors in $\mathbb{P}$ by $\mathbb{P}^{\mathbb{Z}}$. The sequence of Sherali-Adams relaxations

of $\mathbb{P}^{\mathbb{Z}}$ is the sequence of polytopes $\mathbb{P} = \mathbb{P}^1 \supseteq \mathbb{P}^2 \supseteq \cdots$ where $\mathbb{P}^k$ is defined in the following way.

Each inequality in $M\mathbf{y} \geq \mathbf{b}$ is multiplied by all possible terms of the form $\Pi_{i \in I} y_i \Pi_{j \in J}(1 - y_j)$ where $I, J \subseteq [n]$ satisfy $|I \cup J| \leq k - 1$ and $I \cap J = \emptyset$. This leaves a system of polynomial inequalities, each of degree at most k. Then, this system is *linearized* and hence relaxed in the following way: each square y_i^2 is replaced by y_i and each resulting monomial $\Pi_{i \in K} y_i$ is replaced by a variable z_K. In this way we obtain a polytope $\mathbb{P}_L^k$. Finally, $\mathbb{P}_L^k$ is projected back to n dimensions by defining

$$\mathbb{P}^k := \{\mathbf{y} \in \mathbb{R}^n : \text{there exists } \mathbf{z} \in \mathbb{P}_L^k \text{ such that } \mathbf{z}_{\{i\}} = \mathbf{y}_i \text{ for each } i \in [n]\}.$$

We note here that $\mathbb{P}^{\mathbb{Z}} \subseteq \mathbb{P}^k$ for every $k \geq 1$.

In order to apply the SA method to the homomorphism problem there are different possible choices for the polytope $\mathbb{P}$ (encoding a relaxation of homomorphism) to start with, each one then yielding a different hierarchy.

Here we shall adapt a SA-based family of relaxations commonly used in optimization variants of CSP [GMT09, CLRS13, YZ14, TŽ17, CŽ22b] which we transform into a relaxation of (plain) CSP by just turning the objective function into a set of new restrictions. Hence, the resulting system of inequalities is not, strictly speaking, obtained using the SA method. Nonetheless, we shall abuse slightly notation and still use SA^k to refer to our system of inequalities.

In fact, giving an explicit description of the inequalities obtained using the SA method for any natural polytope $\mathbb{P}$ encoding the LP relaxation for a *general* CSP in our setting is a bit cumbersome (because the constraints of the CSP are encoded in the polytope-defining inequalities rather than in the objective function as in the optimization variants).[5]. Hence, it seems sensible to settle for a good approximation as SA^k. Indeed, as we will show in Section 5.2.2, the sequence of relaxations SA^k is tightly interleaved with the sequence $\mathbb{P}^k$ obtained by the SA method, in stricto sensu, for a natural choice of initial polytope $\mathbb{P}$.

Given a pair of σ-structures $\mathbf{X}$ and $\mathbf{A}$, the system of inequalities $\mathrm{SA}^k(\mathbf{X}, \mathbf{A})$ for the homomorphism problem over $(\mathbf{X}, \mathbf{A})$ contains a variable $p_V(f)$ for every $V \subseteq X$ with $1 \leq |V| \leq k$ and every $f : V \to A$, and a

[5] A precise description of these is given in [AD22]

variable $p_{R(\mathbf{x})}(f)$ for every $R(\mathbf{x}) \in \mathcal{C}_\mathbf{A}$ and every $f : \{\mathbf{x}\} \to A$. Each variable must take a value in the range $[0, 1]$. The variables are constrained by the following conditions:

$$\sum_{f:V\to A} p_V(f) = 1 \qquad V \subseteq X \text{ s.t. } |V| \leq k \tag{SA1}$$

$$p_U(f) = \sum_{g:V\to A, g|_U = f} p_V(g) \qquad U \subseteq V \subseteq X \text{ s.t. } |V| \leq k,\ f : U \to A \tag{SA2}$$

$$p_U(f) = \sum_{g:V\to A, g|_U = f} p_{R(\mathbf{x})}(g) \qquad R(\mathbf{x}) \in \mathcal{C}_\mathbf{X},\ U \subseteq \{\mathbf{x}\} = V \text{ s.t. } |U| \leq k, f : U \to A \tag{SA3}$$

$$p_{R(\mathbf{x})}(f) = 0 \qquad R(\mathbf{x}) \in \mathcal{C}_\mathbf{X},\ f : \{\mathbf{x}\} \to A \text{ s.t. } f(\mathbf{x}) \notin R^\mathbf{A} \tag{SA4}$$

For the particular case of $k = 1$ we shall use the simplified notation $p_x(f(x))$ to denote the variable $p_V(f)$ for a singleton set $V = \{x\}$ and a function $f : V \to A$.

It is easy to see that $\mathrm{SA}^1(\mathbf{X}, \mathbf{A})$ is equivalent to $\mathrm{BLP}(\mathbf{X}, \mathbf{A})$, subject to the following additional constraint:

$$p_{R(\mathbf{x})}(\mathbf{a}) = 0 \qquad R(\mathbf{x}) \in \mathcal{C}_\mathbf{X},\ \mathbf{a} \in A^{\mathrm{ar}(R)} \colon \exists i, j \in [\mathrm{ar}(R)] \text{ such that } x_i = x_j \text{ and } a_i \neq a_j \tag{↺}$$

That is, BLP and SA^1 differ only in the way that they deal with repeated entries in a tuple.[6] An easy example to show the difference between BLP and SA^1 is the following.

Example 5.5

Let $\mathbf{X} = (\{0\}; =_1)$ be the equality relation on the 1-element domain and $\mathbf{A} = (\{0, 1\}, \neq_2)$ be the Boolean inequality relation. Let R be the unique binary relation symbol in the signature of $\mathbf{X}$ and $\mathbf{A}$. The equations in (BLP3) together with (↺) enforce that $p_{R(0,0)}(\mathbf{a}) = 0$ for all $\mathbf{a} \in \{0, 1\}^2$, so SA^1 correctly detects that $\mathbf{X} \not\to \mathbf{A}$. On the other

[6] We remark that in the literature the difference between the two relaxations is sometimes neglected, which occasionally leads to unjustified or slightly incorrect claims.

hand, a feasible solution of $\mathrm{BLP}(\mathbf{X}, \mathbf{A})$ is given by $p_0(0) = p_0(1) = 1/2$ and $p_{R(0,0)}(0,1) = p_{R(0,0)}(1,0) = 1/2$.

For a linear program $\mathrm{L} \in \{\mathrm{BLP}, \mathrm{SA}^1\}$ we say that $\mathrm{L}(\mathbf{X}, \mathbf{A})$ is *feasible* if there exists a rational solution to the system $\mathrm{L}(\mathbf{X}, \mathbf{A})$. We say that L *decides* $\mathrm{CSP}(\mathbf{A})$ if, for every input structure $\mathbf{X}$, we have that $\mathrm{L}(\mathbf{X}, \mathbf{A})$ is feasible implies $\mathbf{X} \to \mathbf{A}$ (notice that the opposite implication holds trivially).

While Example 5.5 shows that there are specific instances for which SA^1 is strictly stronger than BLP, it turns out that in general these two relaxations have the same expressive power in terms of solving fixed-template CSPs. In particular, the class of CSPs that are decided by BLP and SA^1 have a precise algebraic characterization in terms of polymorphisms, as the following well-known result shows.

Theorem 5.6 ([BD21a, KOT+12])**.** *Let* $\mathbf{A}$ *be a fixed finite* σ*-structure. The following are equivalent:*

1. SA^1 *decides* $\mathrm{CSP}(\mathbf{A})$*;*
2. BLP *decides* $\mathrm{CSP}(\mathbf{A})$*;*
3. $\mathbf{A}$ *has symmetric polymorphisms of all arities.*

Proof. The proof of this theorem is folklore in the CSP community. A version of the proof of (2) $\Leftrightarrow$ (3) can be found for example in [KOT+12], but we are not aware of a proof of (2) $\Leftrightarrow$ (1) other than the one in [BD21a]. For completeness, we provide the entire argument below.

For the proof of (1) $\Leftrightarrow$ (2), we will use the following fact which follows from the Sparse Incomparability Lemma [NR89]: for every σ-structure $\mathbf{X}$, there exists a structure $\mathbf{X}'$ with no loops such that $\mathbf{X}' \to \mathbf{X}$ and $\mathbf{X} \to \mathbf{A}$ iff $\mathbf{X}' \to \mathbf{A}$. Now, assume that BLP does not solve $\mathrm{CSP}(\mathbf{A})$. This means that there exists a structure $\mathbf{X}$ not homomorphic to $\mathbf{A}$ and such that $\mathrm{BLP}(\mathbf{X}, \mathbf{A})$ is feasible. Now, let $\mathbf{X}'$ be the structure given by the Sparse Incomparability Lemma. Since $\mathbf{X}' \to \mathbf{X}$ it follows that $\mathrm{BLP}(\mathbf{X}', \mathbf{A})$ is feasible, and, since $\mathbf{X}'$ has no loops, $\mathrm{SA}^1(\mathbf{X}', \mathbf{A})$ is feasible as well. Since $\mathbf{X}'$ is not homomorphic to $\mathbf{A}$ it follows that SA^1 does not solve $\mathrm{CSP}(\mathbf{A})$. The same argument can be used to show the converse although it is not

necessary as it also follows immediately by comparing the inequalities of SA^1 and BLP.

On the other hand, the equivalence of (2) and (3) follows immediately if we assume the following result:

Lemma 5.7. *Let* $\mathbf{A}$ *be a* σ*-structure. There exists a set of structures denoted* $\mathrm{LP}^m(\mathbf{A})$*, one for each* $m \in \mathbb{N}$*, which satisfies the following properties:*

1. *For all* $m \in \mathbb{N}$, $\mathrm{LP}^m(\mathbf{A}) \to \mathbf{A}$ *if and only if* $\mathbf{A}$ *has an* m*-ary symmetric polymorphism.*
2. *For all* σ*-structures* $\mathbf{X}$, $\mathrm{BLP}(\mathbf{X}, \mathbf{A})$ *is feasible if and only if there exists some* $m \in \mathbb{N}$ *such that* $\mathbf{X} \to \mathrm{LP}^m(\mathbf{A})$*;*

Then, for (2) $\Rightarrow$ (3) suppose that $\mathbf{A}$ does not have a symmetric polymorphism of some arity m. Then, there $\mathrm{LP}^m(\mathbf{A})$ is not homomorphic to $\mathbf{A}$. It follows that $\mathrm{LP}^m(\mathbf{A})$ is a witness that BLP does not decide $\mathrm{CSP}(\mathbf{A})$, since $\mathrm{BLP}(\mathrm{LP}^m(\mathbf{A}), \mathbf{A})$ must be feasible from (2). Conversely, for (3) $\Rightarrow$ (2), assume that $\mathbf{A}$ has symmetric polymorphisms of all arities. Let $m \in \mathbb{N}$. We know that $\mathrm{LP}^m(\mathbf{A})$ is homomorphic to $\mathbf{A}$ and therefore for all σ-structures $\mathbf{X}$, $\mathrm{BLP}(\mathbf{X}, \mathbf{A})$ is feasible if and only if $\mathbf{X} \to \mathbf{A}$. Hence, BLP decides $\mathrm{CSP}(\mathbf{A})$. □

So it only remains to prove that Lemma 5.7 holds. Variants of a set of structures with this properties have been defined in the literature both for CSP [KOT+12] and for VCSP [TŽ12, VŽ21]. For completeness, we present the proof here.

Proof of Lemma 5.7. The universe of $\mathrm{LP}^m(\mathbf{A})$ is the set $\binom{\binom{A}{m}}{}$ of all A-multisets of size m. For every $R \in \sigma$ of arity r and for every $s_1, \ldots, s_r \in \binom{\binom{A}{m}}{}$, we have that

$$(s_1, \ldots, s_r) \in R^{\mathrm{LP}^m(\mathbf{A})} \Leftrightarrow \exists \mathbf{a}_1, \ldots, \mathbf{a}_m \in R^{\mathbf{A}} \text{ such that } s_i = \{\{\mathbf{a}_1[i], \ldots, \mathbf{a}_m[i]\}\}. \tag{5.1}$$

Item (1) follows easily from equation (5.1) and the obvious correspondence between maps from $\binom{\binom{A}{m}}{}$ to A and symmetric m-ary operations on A.

As for item (2), for the backwards implication let $m \in \mathbb{N}$ be such that $\mathbf{X} \to \mathrm{LP}^m(\mathbf{A})$, and let $h : X \to \left(\!\!\binom{A}{m}\!\!\right)$ be such a homomorphism. We define a feasible solution to $\mathrm{BLP}(\mathbf{X}, \mathbf{A})$ as follows. For $s \in \left(\!\!\binom{A}{m}\!\!\right)$ and $a \in A$, let $n_s(a)$ be the multiplicity of a in s. Then, for each $x \in X$ and $a \in A$ we define $p_x(a) = n_{h(x)}(a)/m$. Similarly, let $R \in \sigma$, $\mathbf{x} \in R^{\mathbf{X}}$, and $\mathbf{s} = (s_1, \ldots, s_{\mathrm{ar}(R)}) = h(\mathbf{x})$ be the image of $\mathbf{x}$ under h. Let $S = \{\{\mathbf{a}_1, \ldots, \mathbf{a}_m\}\}$ be a multiset of tuples in $R^{\mathbf{A}}$ such that $s_i = \{\{\mathbf{a}_1[i], \ldots, \mathbf{a}_m[i]\}\}$, notice that S must exist because of (5.1). Let $n_S(\mathbf{a})$ be the multiplicity of $\mathbf{a}$ in S. Then, we define $p_{R(\mathbf{x})}(\mathbf{a}) = n_S(\mathbf{a})/m$. It is easy to see that these functions satisfy equations (BLP1), (BLP2) and (BLP3) of $\mathrm{BLP}(\mathbf{X}, \mathbf{A})$.

On the other hand, for the forward implication let $p_x(a)$, $p_{R(\mathbf{x})}(\mathbf{a})$ define a feasible solution to the linear system $\mathrm{BLP}(\mathbf{X}, \mathbf{A})$, and let m be the least common multiple of the denominators of this solution. We claim that there exists a homomorphism from $\mathbf{X}$ to $\mathrm{LP}^m(\mathbf{A})$.

For every $x \in X$, we define $h(x)$ to be the A-multiset such that each $a \in A$ appears exactly $m \cdot p_x(a)$ times in $h(x)$ (in particular, by (BLP1) $h(x)$ has size m). Now for some $R \in \sigma$ and $\mathbf{x} \in R^{\mathbf{X}}$, let $S = \{\{\mathbf{a}_1, \ldots, \mathbf{a}_m\}\}$ be such that each $\mathbf{a} \in A^{\mathrm{ar}(R)}$ appears in S exactly $p_{R(\mathbf{x})}(\mathbf{a})$ times. By equation (BLP2), we have that for every $j = 1, \ldots, \mathrm{ar}(R)$, $h(\mathbf{x}[j]) = \{\{\mathbf{a}_1[j], \ldots, \mathbf{a}_m[j]\}\}$ and so $h(\mathbf{x}) \in R^{\mathrm{LP}^m(\mathbf{A})}$ as required. □

The following is an immediate consequence of the proofs of Theorem 5.6 and Lemma 5.7.

Corollary 5.8. *Let $\mathbf{A}$ be a fixed finite σ-structure such that $\mathsf{Pol}(\mathbf{A})$ contains symmetric operations of all arities. Then, for all σ-structures $\mathbf{X}$ where $\mathrm{BLP}(\mathbf{X}, \mathbf{A})$ is feasible, there exists a homomorphism h from $\mathbf{X}$ to $\mathbf{A}$ such that for all $x, x' \in X$ with $p_x(a) = p_{x'}(a)$ for all $a \in A$, we have $h(x) = h(x')$.*

5.2.2 Applying the SA method exactly

Here we shall show that the family of relaxations SA^k considered in the present paper is closely interleaved with the system of relaxations obtained by applying the SA method to a natural choice of initial polytope $\mathbb{P}$.

Let $\mathbf{X}$ and $\mathbf{A}$ be σ-structures. We define polytope $\mathbb{P} = \mathbb{P}(\mathbf{X}, \mathbf{A})$ using a system of inequalities. The variables of the system are $y_{x,a}$ for each $x \in X$ and $a \in A$. Each variable must take a value in the range $[0, 1]$. We remark that by fixing some arbitrary ordering on the variables in $y_{x,a}$ we can represent any assignment on the variables $y_{x,a}$ with a tuple $\mathbf{y} \in \mathbb{R}^n$ with $n = |X| \cdot |A|$. Therefore we shall abuse notation and use $\mathbf{y}_{x,a}$ to refer to the value in $\mathbf{y}$ corresponding to variable $y_{x,a}$.

The variables are subject to the following inequalities.

$$\sum_{a \in A} y_{x,a} = 1 \qquad x \in X, \tag{5.2}$$

$$\sum_{x \in \{\mathbf{x}\}} y_{x,f(x)} \leq |\{\mathbf{x}\}| - 1 \qquad \text{for each } R \in \sigma,\ \mathbf{x} \in R^{\mathbf{X}},\ \text{and } f : \{\mathbf{x}\} \to A \text{ with } f(\mathbf{x}) \notin R^{\mathbf{A}}. \tag{5.3}$$

Note that if h is a homomorphism from $\mathbf{X}$ to $\mathbf{A}$ then the assignment setting $y_{x,h(x)} = 1$ for every $x \in X$ and the rest of variables to zero is feasible.

Now let $\mathbb{P}^k$, $k \geq 1$ be the sequence of polytopes obtained using the SA method. The next lemma shows that the sequence of relaxations defined by SA^k and $\mathbb{P}^k$ are interleaved.

Lemma 5.9. *Let $k \geq 1$ and let r be the maximum arity of a relation in σ. Then*

1. *If $\mathbb{P}^k \neq \emptyset$ and $r \leq k$ then SA^k is feasible.*
2. *If $\mathrm{SA}^{k+r-1}(\mathbf{X}, \mathbf{A})$ is feasible then $\mathbb{P}^k \neq \emptyset$.*

Proof. (1). Assume that $\mathbb{P}^k \neq \emptyset$ and let $\mathbf{z}$ be a feasible solution of $\mathbb{P}^k_L$. We shall construct a feasible solution of $\mathrm{SA}^k(\mathbf{X}, \mathbf{A})$. First, set every variable of the form $p_V(f)$ to z_K where $K = \{(x, f(x)) \mid x \in V\}$. We first observe that this assignment satisfies (SA1) and (SA2). Indeed, let $U \subseteq X$ with $|U| < k$, let $f : U \to A$, and let $I = \{(u, f(u)) : u \in U\}$. Then, multiplying the equality (5.2) with $x \in X \setminus U$ by $\Pi_{i \in I} y_i$ and linearizing we obtain equality (SA2) for U, f, and $V = U \cup \{x\}$. In this way we can obtain all equalities in (SA2) for $|U| + 1 = |V|$. We note here that the rest of equalities in (SA2) along all equalities in (SA1) can be obtained as a linear combination.

Secondly, let us set the rest of variables. For every $(\mathbf{x}, R) \in \mathcal{C}_{\mathbf{X}}$ and $f : \{\mathbf{x}\} \to A$, set $p_{(\mathbf{x},R)}(f)$ to be z_K where $K = \{(x, f(x)) \mid x \in \{\mathbf{x}\}\}$ (note that we are using implicitly the fact that $r \leq k$). Then, (SA3) follows directly from (SA2). Finally, it only remains to show that (SA4) is also satisfied. Indeed, for every $f(\mathbf{x}) \notin R^{\mathbf{A}}$ we obtain equality $p_{(\mathbf{x},R)}(f) = 0$ if we multiply (5.3) by the term $\Pi_{i \in K}$ and linearize.

(2). Assume that $\mathrm{SA}^{k+r-1}(\mathbf{X}, \mathbf{A})$ is feasible. We construct a feasible solution $\mathbf{z}$ of $\mathbb{P}^k_L$ as follows. For every $K \subseteq X \times A$ which satisfies $K = \{(x, f(x)) \mid x \in U\}$ for some $U \subseteq X$ with $|U| \leq k$ and $f : U \to A$, we set $z_K := p_U(f)$. Otherwise, we set z_K to zero.

Let us show that this assignment satisfies all inequalities in $\mathbb{P}^k_L$. Let

$$\mathbf{c}^T \mathbf{z} \leq \mathbf{d} \tag{5.4}$$

be any inequality defining $\mathbb{P}^k_L$. Since (5.4) is obtained by multiplying an inequality which contains at most r variables by a term of at most $k-1$ variables, there exists a set $V \subseteq X$ with $|V| \leq r + k - 1$ such that for every variable z_K appearing in (5.4), V satisfies $K \subseteq V \times A$. Note that, by (SA1), variables $p_V(g)$, $g : V \to A$ define a probability distribution. For every $g : V \to A$ in the support of this distribution, consider the assignment $\mathbf{y}^g$ that sets $\mathbf{y}^g_{x,a} = 1$ if $x \in V$ and $b = g(x)$ and $y_{x,b} = 0$ otherwise.

Inequality (5.4) has been obtained by multiplying an inequality from (5.2) or (5.3) by a term and linearizing. We claim that in both cases, the inequality that has generated (5.4) is satisfied by $\mathbf{y}^g$. If the inequality generating (5.4) is $\sum_{a \in A} y_{x,a} = 1$ for some $x \in X$ this follows simply from the fact that $x \in V$. Assume now that (5.4) has been generated by inequality $\sum_{x \in \{\mathbf{x}\}} y_{x,f(x)} \leq |\{\mathbf{x}\}| - 1$. In this case note that $\{\mathbf{x}\} \subseteq V$ and then the claim follows from (SA3) and (SA4). This finalizes the proof of the claim.

Consequently, since $\mathbf{y}^g$ is integral it follows that the assignment $\mathbf{z}^g$ defined as $\mathbf{z}^g_K = \Pi_{i \in K} \mathbf{y}^g_i$ satisfies (5.4). Finally, note that if we set $\alpha^g = p_V(g)$, then for every $K \subseteq V \times A$, $\mathbf{z}_K$ is precisely given by the convex combination $\sum_g \alpha^g \mathbf{z}^g_K$. □

5.3 Local Consistency

Local Consistency methods are widely used in constraint satisfaction as a heuristic to discard unsatisfiable instances at a low computational cost. The basic idea is to run an (efficient) algorithm that checks consistency in sets of bounded size, and rejects if an inconsistency is found. While these methods are only guaranteed to accept Yes instances (yet they do not always reject No instances), it has been shown that for certain classes of CSPs, local consistency methods are sufficient to decide satisfiability.

Let $1 \leq k \leq l$. Intuitively, an instance $\mathbf{X}$ of CSP($\mathbf{A}$) is said to be (k,l)-*consistent* if the set of all substructures of $\mathbf{X}$ with at most l elements admits a system of homomorphisms into $\mathbf{A}$ that are consistent on all sets of variables of size at most k. For each fixed (k,l), there is a polynomial-time algorithm that checks this condition, which is known as the (k,l)-consistency algorithm. We say that $\mathbf{A}$ has *width* (k,l) if the (k,l)-consistency algorithm decides CSP($\mathbf{A}$): that is, if (k,l)-consistency accepts $\mathbf{X}$, then $\mathbf{X} \to \mathbf{A}$. $\mathbf{A}$ is said to have *bounded width* if it has width (k,l) for some fixed k and l. CSPs of bounded width have a number of equivalent characterizations in terms of solvability in DATALOG, pebble games, bounded treewidth duality, and definability in infinitary logics (see e.g. Theorem 19 in [BKL08]).

One of the simplest notions of consistency is called (generalized) arc consistency. The basic idea is, given an instance $\mathbf{X}$ of CSP($\mathbf{A}$), to maintain a set $S_x \subseteq A$ for each $x \in X$ which, at all times, satisfies that for any homomorphism h from $\mathbf{X}$ to $\mathbf{A}$, $h(x) \in S_x$. Arc consistency can be enforced in polynomial time via a simple algorithm which initially sets $S_x = A$ for each $x \in X$ and then iteratively prunes each S_x in a natural way until a fixed point is reached (see e.g. [CDG13] for details).

Let r denote the maximum arity of a relation in the signature. The families of CSPs that are solvable by arc consistency are precisely those that have width $(1,r)$. One of the many examples of the power of the algebraic approach to CSPs can be found in [DP99] (see also [FV98]), where the CSPs of width $(1,r)$ are characterized in terms of invariance under certain polymorphisms.

Theorem 5.10 ([DP99])**.** *Let* $\mathbf{A}$ *be a finite relational structure.* CSP($\mathbf{A}$)

has width 1 if and only if **A** *is invariant under totally symmetric polymorphisms of all arities.*

It was later claimed in [KOT+12] that the condition of Theorem 5.10 would also be equivalent to **A** being invariant under symmetric polymorphisms of all arities, and as a consequence, that arc consistency would have the same power as BLP (see Theorem 5.6). However, this turned out not to be the case, as the following example illustrates [KS16].

Example 5.11

Let **B** be the relational structure on the 3-element domain $B = \{-1, 0, 1\}$ with two ternary relations defined by

$$R^+ = \{(b_1, b_2, b_3) \in B^3 : b_1 + b_2 + b_3 \geq 1\},$$

$$R^- = \{(b_1, b_2, b_3) \in B^3 : b_1 + b_2 + b_3 \leq -1\}.$$

It is easy to show, e.g. by contradiction, that for every $m \geq 1$, the function $f : B^m \to B$ defined as

$$f(b_1, \ldots, b_m) = \begin{cases} 1 & \text{if } \frac{1}{m}\sum_{i=1}^m b_i \geq \frac{1}{3} \\ 0 & \text{if } |\frac{1}{m}\sum_{i=1}^m b_i| < \frac{1}{3} \\ -1 & \text{if } \frac{1}{m}\sum_{i=1}^m b_i \leq -\frac{1}{3} \end{cases}$$

is a polymorphism of **B**.

On the other hand, suppose that **B** has a totally symmetric polymorphism f of arity 3. Then, there exists some $x \in B$ such that $f(\{-1, 1\}) = x$, and in particular

$$f\left(\begin{pmatrix}-1\\1\\1\end{pmatrix}, \begin{pmatrix}1\\-1\\1\end{pmatrix}, \begin{pmatrix}1\\1\\-1\end{pmatrix}\right) = \begin{pmatrix}x\\x\\x\end{pmatrix} = f\left(\begin{pmatrix}1\\-1\\-1\end{pmatrix}, \begin{pmatrix}-1\\1\\-1\end{pmatrix}, \begin{pmatrix}-1\\-1\\1\end{pmatrix}\right)$$

where f is applied component-wise. It follows that $(x, x, x) \in R^+$ (from the first equality) and $(x, x, x) \in R^-$ (from the second equality), which is a contradiction.

CSPs of bounded width were characterized in [Bul09] and independently in [BK09, BK14], answering a conjecture of Feder and Vardi [FV98]. In particular, it follows from the so-called Barto-Kozik theorem [BK14] and [Bar16] that for any $\mathbf{A}$, if $\mathrm{CSP}(\mathbf{A})$ has bounded width, then it has width $(2, \max\{3, r\})$, where r is the maximum arity of a relation in $\mathbf{A}$.

In a recent line of work, Kozik presented a series of progressively weaker consistency notions that strictly improve upon $(2, \max\{3, r\})$-consistency in terms of efficiency, yet still solve all the CSPs of bounded width: Singleton Arc Consistency (SAC) [Koz16], Singleton Linear Arc Consistency (SLAC) [Koz16], and most recently *jpq*-consistency [Koz21]. The *jpq*-consistency algorithm will find a surprising application in this thesis in the context of solving CSPs via distributed algorithms, see Section 6.4.2.

On a related note, the notion of bounded width also finds applicability in the realm of linear programming algorithms. In fact, it was shown in [TŽ17] that the CSPs of bounded width are precisely those that can be solved by the third level of the Sherali-Adams hierarchy (see Section 5.2). All together, we have the following (non-exhaustive) characterization of the CSPs of bounded width:

Theorem 5.12 ([Bul09, BK14, Bar16, Koz16, Koz21, TŽ17])**.** *Let* $\mathbf{A}$ *be a finite* σ*-structure. The following are equivalent:*

1. $\mathbf{A}$ *has a* k*-ary WNU polymorphism for every* $k \geq 3$*;*
2. $\mathbf{A}$ *has bounded width;*
3. $\mathbf{A}$ *has width* $(2, \max\{3, r\})$*, where* $r = \max_{R \in \sigma} \mathrm{ar}(R)$*;*
4. $\mathrm{CSP}(\mathbf{A})$ *is solvable by* SAC*;*
5. $\mathrm{CSP}(\mathbf{A})$ *is solvable by jpq-consistency;*
6. SA^3 *decides* $\mathrm{CSP}(\mathbf{A})$*.*

While not specifically relevant to this thesis, the paper [AD22] provides an interesting contribution that motivates the study of these relaxations for the CSP and its extensions. In particular, the authors show that the implication (2) $\Rightarrow$ (1) in Theorem 5.12 can be lifted to PCSPs, yet the converse implication breaks. In particular, in the Promise setting even the

second level of the Sherali-Adams hierarchy is strictly stronger than local consistency as the former solves, for instance, $\mathrm{PCSP}(\text{1-in-3-SAT}, \mathrm{NAE})$ while the latter does not.

6 Distributed CSPs

In this chapter we study the complexity of the Distributed Constraint Satisfaction Problem (DCSP) on a synchronous, anonymous network from a theoretical standpoint. We show a dichotomy theorem for both the decision and the search version of DCSP in terms of invariance under symmetric polymorphisms.

6.1 Introduction

In this chapter we study the computational complexity of the distributed counterpart of CSP, which is known as DCSP. This was introduced by Yokoo et al. [YIDK92] as a formal framework for the study of cooperative distributed problem solving. In particular, we consider a deterministic, synchronous, anonymous network of agents controlling variables and constraints, and we study the complexity of solving constraint satisfaction problems by using message passing on this network. A number of practical applications can be encoded in the DCSP model, for instance resource allocation tasks in wireless networks, routing, networking, and mobile technologies (see for instance [BKGS01, DBL13]).

We notice that this framework is general enough to encompass some simple Graph Neural Network (GNN) architectures (see for example [Gro20, MRF+19]). In particular, when training a GNN to classify graphs, it is customary that the GNN network ignores the node label when updating its feature vector. This is, in fact, essential as otherwise there would be no way to apply the network trained on a given graph to another one.

However, whereas in all variants of GNNs the computation is limited to a reduced number of operations over feature vectors, in the DCSP model the computation at each node is governed by an arbitrary algorithm. GNNs have a wide range of applications including molecule and image classification (see [BHB+18] for example). Recently, GNNs have been deployed to solve CSPs [TRWG20].

While there are a variety of well-performing distributed algorithms for constraint satisfaction and optimisation (see for instance [YH00, Mei08, FPY18]), the theoretical aspects of distributed complexity are to date not well understood. In this chapter we initiate the study of the complexity of DCSP parametrized by the constraint language, obtaining a complete characterization of its tractable classes. More specifically, building on the connection between the CSP and algebra, we show that for any finite constraint language $\mathbf{A}$, the decision problem for DCSP($\mathbf{A}$) is tractable whenever $\mathbf{A}$ is invariant under symmetric polymorphisms of all arities. Otherwise, there are no message passing algorithms that solve DCSP($\mathbf{A}$). Collaterally, we show that the same holds for the search problem for DCSP.

Our work begins with the identification of a connection between the inherent nature of message passing algorithms in distributed networks and the 1-dimensional Weisfeiler-Leman algorithm. It turns out that, due to the network's anonymity, in every distributed algorithm all the agents that are equivalent with respect to 1-WL must necessarily behave identically at each round. A similar phenomenon has been observed independently in the context of GNNs [MRF+19, XHLJ19] (see also [BKM+20, MLM+21]).

We use this fact to show that, under the absence of symmetric polymorphisms of any arity in $\mathbf{A}$, it is always possible to construct two instances of DCSP($\mathbf{A}$), one satisfiable and the other unsatisfiable, that cannot be distinguished by any message passing algorithm in an anonymous network.

On the other hand, as we saw in Section 5.2, if $\mathbf{A}$ has symmetric polymorphisms of all arities then the basic linear programming relaxation decides CSP($\mathbf{A}$). While it is not clear how to directly use this fact to obtain a distributed algorithm for DCSP($\mathbf{A}$), it can be applied to establish a structure theorem that unveils a simple yet surprising structure in the solution space of every satisfiable instance of CSP($\mathbf{A}$): it must contain a solution that assigns the same value to all variables that are $\equiv_1$-equivalent. The proof of this structure theorem uses the weighted majority algorithm,

a weight update method that is widely used in optimisation and machine learning applications (see [AHK12]). The structure theorem is key in the proof of the positive results as it allows to run an adapted variant of the jpq-consistency algorithm [Koz21] that overcomes the absence of unique identifiers for the variables, by using instead their iterated degree.

This chapter is organised as follows. In Section 6.2 we introduce some definitions and technical concepts about the DCSP model. In Section 6.3 we show the connection between iterated degree and the basic LP relaxation for CSP, culminating in the structure theorem (Theorem 6.12). Section 6.4 is dedicated to the proof of the dichotomy theorem for the complexity of DCSP, with the hardness results in Section 6.4.1, the details of the distributed algorithm for tractable languages in Section 6.4.2, and its extension to the search problem in Section 6.4.3.

6.2 Distributed CSPs

For consistency with the rest of the thesis, we shall deviate from the notation in [BD22] (and of [FPY18]) and regard DCSP as a distributed homomorphism problem.

The Distributed Model. We consider the DCSP model of [YIDK92] with some small modifications. The basic idea is to assign the task of solving a constraint satisfaction problem to a multi-agent system whose organisation depends on the input structure. In the original model, which assumes that all constraints are binary [YDIK98, YH00], the premise is that each variable in the input structure is controlled by an agent, and two agents can communicate with one another if and only if they share a constraint. Here we deviate slightly from the original model to allow for non-binary constraints and we assume that both variables and constraints of the input structure are controlled by distributed agents in the network. An instance of the *Distributed Constraint Satisfaction Problem* over a finite-domain relational structure $\mathbf{A}$, denoted DCSP($\mathbf{A}$), is a triple $(\mathbf{X}, \Psi, \psi)$, where $\mathbf{X}$ is a relational structure similar to $\mathbf{A}$, Ψ is a finite set of agents, and $\psi : X \cup \mathcal{C}_{\mathbf{X}} \to \Psi$ is a surjective function which assigns the control of each variable $x \in X$ and each constraint $c \in \mathcal{C}_{\mathbf{X}}$ to an agent

$\psi(x)$, $\psi(c)$ respectively. Unless explicitly stated otherwise, in this chapter we will adhere to the following assumptions. All instances are assumed to be connected. The number of variables and constraints of the input instance will be denoted by n and m respectively. We shall assume that there are exactly $n + m$ agents, and therefore each agent controls exactly one variable or one constraint. Under this assumption, there is a one-to-one correspondence between instances of CSP($\mathbf{A}$) and of DCSP($\mathbf{A}$), and thus we shall switch freely between them, while maintaining the distinction between agents and their controlled variables and constraints for clarity when discussing distributed algorithms.

Distributed Networks and Message Passing. We now present some fundamental concepts relating to the message-passing paradigm for distributed networks. For a general introduction to distributed algorithms, we refer the reader to [Fok13]. A *distributed system* consists of a finite set of agents or processes, which are connected through communication channels to form a network. Any process in the network can perform events of three kinds: *send*, *receive* and *internal*. Send and receive events are self-explanatory, as they denote the sending or receiving of a message over a communication channel. Any kind of local computation performed at the process level, as well as state changes and decisions, are classified as internal events.

We assume a fully *synchronous* communication model, meaning that the send event at a process p and the corresponding receive event at a process p' can be considered *de facto* as a unique event, with no time delay. As a whole, a synchronous system proceeds in rounds, where at each round a process can perform some internal computation and then send messages to and receive messages from its neighbours. A round needs to terminate at every process before the next round begins. Note that while for simplicity we assume a synchronous network, all our algorithms can be adapted to asynchronous systems by applying a simple synchronizer. Nonetheless, we point out that our negative results rely on the network operating in synchronous rounds.

We make the fundamental assumption that the network is *anonymous*, meaning that variables, constraints and agents do not have IDs. For practical purposes, we still refer to variables and constraints with names (such

as x_i, c_i), however these cannot be communicated through the channels. The assumption of anonymity can have various practical justifications: the processes may actually lack the hardware to have an ID, or they may be unable to reveal their ID due to security or privacy concerns. For instance, the basic architecture of GNNs is anonymous. This is a very desirable property as it allows to deploy GNNs in different networks than those in which they were trained.

We assume that all the processes run locally the same deterministic algorithm, therefore IDs cannot be created and deadlocks cannot be broken by for instance flipping a random coin. Hence, the lack of IDs makes the processes essentially indistinguishable from one another - except, as we will see later, for the structure of their neighbourhood in the network.

Leader election is a procedure by which the processes in a network select a single process to be the leader in a distributed way. If a leader is elected, then she can, for instance, dictate the output to the other processes. Moreover, all the information about the instance can be gathered to the leader, which if the network had unique IDs, would be sufficient to solve the CSP locally at the leader. It is a well-known result that there does not exist a terminating deterministic algorithm to elect a leader in an anonymous ring [Ang80]. Therefore, the assumptions of anonymity and determinism ensure that the DCSP model is intrinsically different from the (centralised) CSP framework, and open up the way for establishing novel, non-trivial complexity results. We remark that while considerable effort has been put into characterizing under what conditions an anonymous network is able to elect a leader [BSV+96, YK88] or compute relations [BV01], our work focuses on characterizing the complexity of the DCSP as parametrised by the template. Therefore, all of our algorithms work regardless of the topology of the network, and hence regardless of whether or not a leader can be elected.

The encoding of a DCSP instance into the message passing framework is straightforward. The processes correspond to the agents of the network, and there is a labelled communication channel between a variable agent $\psi(x)$ and a constraint agent $\psi(c)$ if and only if $x \in c$. More formally, let $\mathrm{G}_{\mathbf{X}}$ be the factor graph of an input structure $\mathbf{X}$. Then, the message passing network corresponds to the factor graph where every node (variable or constraint) is replaced by their associated agent and every edge

by a communication channel of the same label. Note that between any two agents there is at most one channel. If privacy is a concern, we point out that labeling channels does not reveal any more information about the processes than what is strictly necessary for the problem instance to be well defined. It is easy to prove (see Remark 6.1) that in the case that all relations are binary, the original model where only variables are controlled by agents is equivalent to our model.

At the start of an algorithm, a process only has access to very limited information. All processes know the template, the total number n of variables and m of constraints in the CSP instance, the labels of the communication channels that they are incident to in the network, and naturally whether they are controlling a variable or a constraint. During a run of the algorithm a process can acquire further knowledge from the messages that it receives from its neighbours. We assume that at any time each process is in one of a set of states, a subset of which are terminating states. When it enters a terminating state, a process performs no more send or internal events, and all receive events are disregarded. The local algorithm is then a deterministic function which governs the process's next state and the messages it will send to its neighbours. The output of such function only depends on the process's current knowledge (including the messages it has received so far) and on its state. We allow processes to send different messages through different channels. However, since processes can only distinguish the channels based on their labels, identical messages must be sent through channels with identical labels. Note that the power of the model would not decrease if only one message was allowed to be passed through all the channels, since a process can simulate sending a separate message through each channel by tagging each message with the label of the desired channel and concatenating them in a unique string. This, however, comes at the cost of increased message size. Moreover, if a process needs to broadcast multiple messages, these can be concatenated into one. We say that an algorithm terminates when all processes are in a terminating state. For a precise formalisation of the definition of distributed algorithm, see Remark 6.2.

We say that a distributed algorithm solves an instance $(\mathbf{X}, \Psi, \psi)$ of DCSP($\mathbf{A}$) if the algorithm terminates and at termination every process in Ψ correctly outputs Yes if $\mathbf{X}$ is homomorphic to $\mathbf{A}$, and No other-

wise. Moreover, we consider the search version of DCSP($\mathbf{A}$), denoted DCSP-Search($\mathbf{A}$). In the search version, if $\mathbf{X}$ is homomorphic to $\mathbf{A}$, at termination every variable process $\psi(x)$ must additionally specify a value $h(x) \in A$ such that the function $h : X \to A$ is a homomorphism. We say that a distributed algorithm solves DCSP($\mathbf{A}$) (respectively DCSP-Search($\mathbf{A}$)) if it solves all connected instances of DCSP($\mathbf{A}$) (respectively DCSP-Search($\mathbf{A}$)).

In terms of algorithmic complexity, there are a number of measures that can be of interest. Time complexity, which is our primary concern, corresponds to the total amount of time required for the algorithm to terminate, including the time needed for internal events. This is closely related to the number of rounds of the algorithm, which is another measure that we are concerned with. Message complexity and bit complexity measure the total number of messages and bits exchanged respectively. These can be bounded easily from the maximum size of a message.

Remark 6.1 (Binary structures).

Throughout this chapter, we assume that both variables and constraints are controlled by agents in a distributed network (in this section, we will refer to this as model 1). However, when the signature only contains binary relation symbols it is also valid and, indeed, more common to assume that only variables are controlled by agents, and there is a communication channel between any two variable agents $\psi(x)$ and $\psi(x')$ whenever x and x' share a constraint (model 2) which is labelled with the relation and the direction of the constraint.

It is very easy to see that in the case of a binary signature both models are equivalent. Indeed, for every binary template $\mathbf{A}$ and every instance $\mathbf{X}$ of CSP($\mathbf{A}$), let $(\mathbf{X}, \Psi_1, \psi_1)$ and $(\mathbf{X}, \Psi_2, \psi_2)$ be the associated instances of DCSP($\mathbf{A}$) in model 1 and 2 respectively. It is easy to see that every algorithm in model 2 can be easily simulated by an algorithm in model 1. In particular, it is only necessary that at round $2j$ every variable agent $\psi_1(x)$ replicates the j^{th} round of $\psi_2(x)$ (while every constraint agent $\psi_1(c)$ remains idle). Then, round $2j+1$

is used to replicate the messages sent at round j. That is, whenever $\psi_2(x)$ sends a message to a neighbour $\psi_2(x')$ at round j, $\psi_1(x)$ sends a message to $\psi_1(c)$ at round $2j$, where c is the constraint shared by x and x'. At round $2j+1$ then $\psi_1(c)$ forwards the message to $\psi_1(x')$.

Similarly, any algorithm in model 1 can be replicated in model 2. In this case, at a given round j, every agent $\psi_2(x)$ simulates the internal computation done at round j by $\psi_1(x)$ and all its neighbours.

Remark 6.2 (The distributed formalism).

We give a formalization of the definition of the distributed model which we presented above. The impatient reader can skip this remark as the informal definitions given above will be sufficient for all the results of this chapter. Note that for simplicity, here we describe a distributed system which only allows message broadcast (as opposed to allowing different messages to be sent through channels with different labels). Nonetheless, as pointed out above, this does not decrease the expressive power of the system.

Let $\mathbb{N}$ denote the set of positive integers and $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$. For a (possibly infinite) set S, we use $\mathbb{N}_0^S$ (as an extension of the power set notation 2^S) to denote the set of all multisets containing just elements from S. Let Σ be a finite alphabet. A *message* is a finite string of symbols from Σ, that is, an element of Σ^*. A *process* (also called an *agent*) is a 7-tuple $(\Sigma, \Lambda, \mathcal{L}, Q, q_o, Q_f, F)$ where Λ is a finite set known as the *memory alphabet*, $\mathcal{L}$ is a finite set of *labels*, Q is a finite set of *states*, $q_0 \in Q$ is the *initial state*, $Q_f \subseteq Q$ denotes the subset of *terminal states*, and F is a Turing-computable function

$$F : Q \times \Lambda^* \times \mathbb{N}_0^{\mathcal{L} \times \Sigma^*} \to Q \times \Lambda^* \times \Sigma^*$$

which takes as input the current state, the memory tape, and the multiset of labelled messages received at the previous round, and outputs a new state, an updated memory tape, and a message to be broadcast to the process's neighbours. We require that terminal states are fixed points of F in the sense that for all $q \in Q_f$, $\lambda \in \Lambda^*$,

and $M \in \mathbb{N}_0^{\mathcal{L}\times\Sigma^*}$ it holds that

$$F(q, \lambda, M) = (q, \lambda, \varepsilon)$$

where $\varepsilon \in \Sigma^*$ denotes the empty string. An output of the function F – that is, a triplet $\gamma = (q, \lambda, s) \in Q\times\Lambda^*\times\Sigma^*$ – is called a *configuration* of the process.

In this thesis, a *message-passing network* is an edge-labelled graph whose vertex set is a set of identical processes and where the label set of the network corresponds to the label set $\mathcal{L}$ of the processes. The neighbourhood of a process i in the network is denoted $N(i)$. Given a message-passing network G with vertex set $[p]$, a *run* of F over G with initial memory $(\lambda_0^1, \ldots, \lambda_0^p)$ is a collection of p sequences of configurations $\gamma_0^i, \gamma_1^i, \gamma_2^i \ldots$, $i \in [p]$ where $\gamma_t^i = (q_t^i, \lambda_t^i, s_t^i)$, such that $q_0^i = q_0$ is the initial state, $s_0^i = \varepsilon$ is the empty string, and for all $t \geq 0$ and all $i \in [p]$, $(q_{t+1}^i, \lambda_{t+1}^i, s_{t+1}^i) = F(q_t^i, \lambda_t^i, M_t^i)$, where $M_t^i = \{\{(\ell_{\{i,j\}}, s_t^j) \mid j \in N(i)\}\}$ and $\ell_{\{i,j\}} \in \mathcal{L}$ denotes the label of edge $\{i, j\}$. A run is said to be *terminating* if there exists some integer T such that, for each $i \in [p]$, $q_T^i \in Q_f$ (and hence, by definition of F, $q_t^i = q_T^i$ for all $t > T$). The *output* of the run is the set of terminal memory tapes λ_T^i, $i \in [p]$.

We point out that, while for convenience we define runs to be infinite sequences of configurations, for all practical purposes one can assume that a process effectively halts as soon as it reaches a terminating state.

Notice that, given p identical processes, a run of the algorithm is fully determined by the connectivity and labels of the network, and by the choice of initial memory λ_0^i for each of the p processes. This is generally fairly limited as each process acquires new information via message passing. It could, for instance, contain the processes' unique identifiers if these were available (this is clearly not the case for anonymous networks like the one that we address in this thesis). In the case of the DCSP message passing network, the network is bipartite and each edge with its corresponding label describes how a variable participates in a constraint. The initial memory is limited to

the numbers n and m of variables and constraints respectively in the DCSP instance, and to a bit which specifies if the process is controlling a variable or a constraint. In particular, it contains no identifier nor any other information about the input instance. Moreover, in the case of DCSP the function F can depend on the template, since this is fixed, but not on the specifics of the input instance. Finally, we remark that while F does not depend explicitly on the round t, this can be stored and updated in the process's memory, and the messages can even be tagged with a time stamp if this is needed for synchronization purposes, for example, due to hardware issues.

Iterated Degree and $\equiv_1$-equivalence. We say that two processes in the distributed network given by $\mathrm{G}_{\mathbf{X}}$ are $\equiv_1$-equivalent if the variables (resp. constraints) that they control are $\equiv_1$-equivalent in $\mathbf{X}$. The notion of iterated degree is strikingly relevant in our work as it captures what it means for two processes in a network to be indistinguishable. This implies that no distributed algorithm can differentiate between two $\equiv_1$-equivalent nodes, as we illustrate in the following result.

Proposition 6.3. *Let $(\mathbf{X}, \Psi, \psi)$ be a not necessarily connected instance of* DCSP($\mathbf{A}$) *and consider two nodes v, v' in $\mathrm{G}_{\mathbf{X}}$. Then, $v \equiv_1 v'$ if and only if any terminating decision algorithm over $\mathbf{X}$ outputs the same decision at $\psi(v)$ and $\psi(v')$. Furthermore, if $v, v' \in X$ and $\mathbf{X} \to \mathbf{A}$, then any terminating search algorithm outputs the same values $h(v) = h(v')$ at $\psi(v)$ and $\psi(v')$.*

Proof. ($\Rightarrow$). At the beginning of the algorithm, all processes are in the same state. Let v be a node in the factor graph of $\mathbf{X}$, and denote by s_t^v the message broadcast at time t by $\psi(v)$ to its neighbours. For any two nodes v, v', $\delta_1^{\mathbf{X}}(v) = \delta_1^{\mathbf{X}}(v')$ is equivalent to v and v' having the same knowledge at the start of the algorithm. This means that the first internal and send events are the same at $\psi(v)$ and at $\psi(v')$, hence $s_1^v = s_1^{v'}$. Then, it is easy to see by induction that $\delta_t^{\mathbf{X}}(v) = \delta_t^{\mathbf{X}}(v') \Rightarrow s_t^v = s_t^{v'}$, which in turn implies that

$$v \equiv_1 v' \Rightarrow s_t^v = s_t^{v'} \quad \text{at all times } t = 1, 2, \ldots$$

This implies that at any time t, $\psi(v)$ and $\psi(v')$ send and receive the same messages, so they have the same knowledge and hence the internal events

at $\psi(v)$ and $\psi(v')$ are the same at all time. In particular, if the algorithm terminates, then the terminating state is the same at $\psi(v)$ and $\psi(v')$, and therefore the decision and, in case of search, the value of h at $\psi(v)$ and $\psi(v')$ are the same.

($\Leftarrow$). Consider the algorithm that calculates the iterated degree at each node (we detail the procedure in the proof of Theorem 6.21). If $v \not\equiv_1 v'$, then we can devise an algorithm that on the basis of the iterated degree gives different outputs at $\psi(v)$ and $\psi(v')$. □

Example 6.4 (Distributed 2-coloring – search).

Let $\mathbf{A} = (\{0,1\}; \neq_2)$ where $\neq_2$ is the inequality relation on the Boolean domain. Let $\mathbf{X}$ be an instance of DCSP($\mathbf{A}$) with variables $\{x_1, \ldots, x_n\}$ and constraints $((x_1, x_2), \neq_2)$, $((x_2, x_3), \neq_2)$, ... , $((x_n, x_1), \neq_2)$; that is, $\mathbf{X}$ can be seen as a directed n-cycle. If n is even, then clearly $\mathbf{X}$ is homomorphic to $\mathbf{A}$. Therefore, if there was a distributed algorithm that solved DCSP-Search($\mathbf{A}$), any two agents controlling two consecutive variables x_i, x_{i+1} should output at termination a partial map h such that $h(x_i) \neq h(x_{i+1})$. But it is easy to see that all variable agents in $\mathbf{X}$ have the same iterated degree, and so by Proposition 6.3, such an algorithm cannot exist.

The following is a direct consequence of Proposition 6.3.

Corollary 6.5. *Let* $(\mathbf{X}, \Psi, \psi)$, $(\mathbf{X}', \Psi', \psi')$ *be (connected) instances of* DCSP($\mathbf{A}$) *such that* $\mathbf{X} \equiv_1 \mathbf{X}'$. *Then with both inputs any terminating decision algorithm for* DCSP($\mathbf{A}$) *will report the same decision.*

Example 6.6 (Distributed 2-coloring).

Let $\mathbf{A} = (\{0,1\}; \neq_2)$ be as in Example 6.4 and consider the two instances pictured in Figure 9.1 (where nodes represent variables and edges represent constraints). By Corollary 6.5, any terminating decision algorithm will report the same decision on both instances. How-

ever, one of the graphs is 2-colorable and the other is not, so there cannot be a distributed algorithm that solves DCSP($\mathbf{A}$).

Equality-free pp-definability. Recall from Section 2.1.1 that pp-definability provides one of the most basic tools for complexity reductions between CSPs (see Theorem 2.1). We would like to harness the power of pp-definitions to obtain analogous reductions between DCSPs. However, in the distributed setting, allowing equality in the pp-definitions introduces a few technical difficulties. Fortunately, this obstacle can be overcome by considering a more restricted notion of pp-definability which, following [JLNZ17], we shall call *equality-free primitive positive definability* (efpp-definability for short) where equality is not allowed. More precisely, we shall say that a relation R is efpp-definable from a σ-structure $\mathbf{A}$ if it is pp-definable from $\mathbf{A}$ and, in addition, the pp-formula defining R only uses relations from σ. That is, we are not allowed to use the equality relation in the formula, unless, of course, it belongs already to $\mathbf{A}$. Then, we have:

Proposition 6.7. *Let $\mathbf{A}$, $\mathbf{A}'$ be finite-domain relational structures, and assume that all the relations of $\mathbf{A}'$ are efpp-definable from $\mathbf{A}$. If* DCSP($\mathbf{A}$) *is solvable in polynomial time (resp. finite time) then so is* DCSP($\mathbf{A}'$).

Proof. Given an algorithm Alg that solves DCSP($\mathbf{A}$) we can design a new algorithm Alg$'$ for DCSP($\mathbf{A}'$) that, given an instance $(\mathbf{X}', \Psi', \psi')$ of DCSP($\mathbf{A}'$) simulates the execution of Alg with the instance $(\mathbf{X}, \Psi, \psi)$ of DCSP($\mathbf{A}$) defined as follows. For every constraint $R(\mathbf{x})$ in $\mathcal{C}_{\mathbf{X}'}$, consider the pair $((y_1, \ldots, y_{\mathrm{ar}(R)}), \mathbf{Y}_R)$ defining $R^{\mathbf{A}'}$ over $\mathbf{A}$, and define $\mathbf{Y}_{R(\mathbf{x})}$ to be the structure obtained from $\mathbf{Y}_R$ by renaming the variables so that $y_i = x_i$ for every $1 \leq i \leq \mathrm{ar}(R)$ and the rest of variables in $\mathbf{Y}_{R(\mathbf{x})}$ are fresh. Note that $\mathbf{Y}_R$ (and hence $\mathbf{Y}_{R(\mathbf{x})}$) is similar to $\mathbf{A}$. Then we define $\mathbf{X}$ to be the union of $\mathbf{X}'$ with all the structures $\mathbf{Y}_{R(\mathbf{x})}$ for $R(\mathbf{x}) \in \mathcal{C}_{\mathbf{X}'}$. Ψ and ψ are defined such that ψ agrees with ψ' over X' and, as usual, every variable and constraint in $\mathbf{X}$ is controlled by a different agent.

The simulation is as follows. At each round, for every $x \in X'$, $\psi'(x)$ simulates the execution of $\psi(x)$ as in Alg, and for every $c \in \mathcal{C}_{\mathbf{X}'}$, $\psi'(c)$ simulates the execution of all constraints and fresh variables in $\mathbf{Y}_c$. We note that no new communication channels need to be created as this simulation

is done internally by $\psi'(R(\mathbf{x}))$. The transmission of messages can be also easily simulated for every pair of neighbours $\psi(x)$ and $\psi(c)$ in Ψ. In fact, if $x \notin X'$, then both $\psi(x)$ and $\psi(c)$ are simulated by the same agent $\psi'(c)$ in Ψ' (and, hence, no communication is required). Otherwise, if $x \in X'$, $\psi(x)$ is simulated by $\psi'(x)$ and $\psi(c)$ is simulated by some neighbour $\psi'(c')$ of $\psi(x)$. □

Remark 6.8 (Power structure as efpp-definition).

We note here that for every $k \geq 1$, the k^{th} power structure of $\mathbf{A}$ constitutes an efpp-definition of the $|A|^k$-ary relation U encoding the set of all polymorphisms of $\mathbf{A}$ of arity k. It then follows from Proposition 6.7 that if DCSP($\mathbf{A}$) is solvable in polynomial time (resp. finite time) then so is DCSP($\mathbf{U}$), where $\mathbf{U} = (A^k; U)$.

6.3 The Structure Theorem

Our work unveils a novel structure in the space of solutions of a CSP instance that is deeply connected to the symmetry of its polymorphisms. In particular, $\mathsf{Pol}(\mathbf{A})$ containing symmetric operations of all arities is equivalent to the existence of a homomorphism for every satisfiable instance of CSP($\mathbf{A}$) that preserves the partition induced by $\equiv_1$. This is the main result of this section. To build up to it we will need the following result.

Theorem 6.9. *Let* $\mathbf{X}$ *and* $\mathbf{A}$ *be* σ*-structures such that* BLP($\mathbf{X}, \mathbf{A}$) *is feasible. Then,* BLP($\mathbf{X}, \mathbf{A}$) *has a feasible solution such that for every* $x, x' \in X$ *with* $x \equiv_1 x'$ *and every* $a \in A$, $p_x(a) = p_{x'}(a)$.

Proof. We start by rewriting BLP($\mathbf{X}, \mathbf{A}$) in the form

$$\exists ? \mathbf{v} \in [0,1]^V \quad B\mathbf{v} \geq \mathbf{b}. \tag{6.1}$$

by replacing every equality $a = b$ by the inequalities $a \geq b$ and $-a \geq -b$.

It will be convenient to index the rows and columns of B not using positive integers. Let us start with the columns. Each column is associated to a variable of BLP($\mathbf{X}, \mathbf{A}$), i.e, a variable of the form $p_x(a)$, $x \in X$, $a \in A$

or $p_c(\mathbf{a})$, $c \in \mathcal{C}_{\mathbf{X}}$, $\mathbf{a} \in A^{\mathrm{ar}(c)}$.In the first case, we index the corresponding column with the pair (x, a) whereas in the second case we index it with the pair $(c, \mathbf{a})$, and we denote by V the set of all such indices.

Now, let us turn our attention to the rows. Every equation in (BLP1) gives rise to two rows that we shall index with $(x, +)$ and $(x, -)$. Similarly, every equation in (BLP2) also gives rise to two rows that we shall index with $(c, i, a, +)$ and $(c, i, a, -)$. Let us denote by W the set of all indexes for rows.

We shall see later how to define an oracle which, given a probability W-vector $\mathbf{p}$ (i.e, a vector $\mathbf{p}$ with non-negative entries such that the sum of all its entries is 1), outputs a vector $\mathbf{v}$ which is a solution to the weaker problem

$$\exists ?\mathbf{v} \in [0,1]^V \quad \mathbf{p}^T B\mathbf{v} \geq \mathbf{p}^T\mathbf{b} \tag{6.2}$$

if one exists, or correctly states that no such vectors exist otherwise. Note that if a solution exists to (6.1), then it is necessarily also a solution to (6.2), while the opposite is not true in general.

For every $w \in W$, let us denote by B_w the row corresponding to w. If $w = (x, +)$ then, since the vector returned by the oracle satisfies $\mathbf{v} \in [0,1]^V$ it follows easily that $B_w\mathbf{v} - \mathbf{b}[w] \in [-1, |A|]$. Similarly, if $w = (c, i, a, +)$ then $B_w\mathbf{v} - \mathbf{b}[w] \in [-1, \max_{R\in\sigma} |R^{\mathbf{A}}|]$. It follows that by setting $\ell = 1$ and $\rho = \max\{|A|, \max_{R\in\sigma} |R^{\mathbf{A}}|\}$ any such oracle-given vector $\mathbf{v}$ satisfies the following condition: there is a fixed subset $J \subseteq W$ (consisting precisely of the positive rows) such that

$$B_w\mathbf{v} - \mathbf{b}[w] \in [-\ell, \rho] \quad \forall w \in J,$$
$$B_w\mathbf{v} - \mathbf{b}[w] \in [-\rho, \ell] \quad \forall w \notin J.$$

Such an oracle is known as an (ℓ, ρ)-*bounded oracle*. Then we have:

Theorem 6.10 ([AHK12])**.** *Let $\varepsilon > 0$ be an arbitrary error parameter. Suppose that there exists an (ℓ, ρ)-bounded oracle for the feasibility problem (6.2). Assume that $\ell \geq \frac{\varepsilon}{2}$. Then there exists an algorithm which either finds $\mathbf{v}$ such that $B\mathbf{v} \geq \mathbf{b} - \varepsilon$ whenever such $\mathbf{v}$ exists, or correctly concludes that no such $\mathbf{v}$ exists otherwise. Such algorithm makes $\mathcal{O}(\ell\rho \log(|W|)/\varepsilon^2)$ calls to the oracle.*

The algorithm that Theorem 6.10 refers to is Multiplicative Weight Update (MWU), a well-known technique that is widely used in optimisation

Algorithm 1: Multiplicative Weight Update

Initialisation: Fix $\eta \in [0, \frac{1}{2}]$ and let $\mathbf{w}^{(1)}$ be a W-vector, whose entries, called weights, are initially set to 1.

for $t = 1, \ldots, T$ **do**

Compute the probability vector $\mathbf{p}^{(t)} = \frac{1}{\Phi(t)}\mathbf{w}^{(t)}$, where $\Phi(t) = \sum_{j=1}^{|W|} \mathbf{w}^{(t)}[j]$

Let $\mathbf{v}^{(t)}$ be a solution satisfying $(\mathbf{p}^{(t)})^T B\mathbf{v}^{(t)} \geq (\mathbf{p}^{(t)})^T\mathbf{b}$ given by oracle O

Compute the losses $\boldsymbol{\ell}^{(t)} = \frac{1}{\rho}(B\mathbf{v}^{(t)} - \mathbf{b})$

Compute the new weights $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} \odot (1 - \eta\boldsymbol{\ell}^{(t)})$ (where $\odot$ denotes the element-wise product)

end

return $\mathbf{v} := \frac{1}{T}\sum_{t=1}^{T} \mathbf{v}^{(t)}$

and machine learning. MWU was discovered independently by researchers of different communities; for a survey of its different variants we refer the reader to [AHK12]. The version that we are interested in is described in Algorithm 1. Recall that the algorithm assumes that $\mathrm{BLP}(\mathbf{X}, \mathbf{A})$ is feasible.

We shall see that if we choose the oracle O wisely then for every $x, x' \in X$ with $x \equiv_1 x'$ and every $a \in A$, the solution returned by the MWU algorithm assigns the same value to $p_x(a)$ and $p_{x'}(a)$.

To see this we need some more notation. We note that $\equiv_1$ induces in a natural way an equivalence relation $\sim_V$ on V. In particular, we have that $v, v' \in V$ are $\sim_V$-related if $v = (x, a)$ and $v' = (x', a)$ where $x \equiv_1 x'$ and $a \in A$, or $v = (c, \mathbf{a})$ and $v' = (c', \mathbf{a})$ where $c \equiv_1 c'$ and $\mathbf{a} \in A^{\mathrm{ar}(c)}$. Similarly $\equiv_1$ induces an equivalence relation, denoted $\sim_W$, on W. More specifically, we have that $w, w' \in W$ are $\sim_W$-related if $w = (x, s)$ and $w' = (x', s)$ where $x \equiv_1 x'$ and $s \in \{+, -\}$ or $w = (c, i, a, s)$ and $w' = (c', i, a, s)$ where $c \equiv_1 c'$, $i \in [\mathrm{ar}(c)]$, $a \in A$, and $s \in \{+, -\}$.

Now, we say that a V-vector $\mathbf{v}$ is $\sim_V$-*preserving* if $\mathbf{v}[v] = \mathbf{v}[v']$ whenever $v \sim_V v'$ and we similarly define $\sim_W$-preserving W-vectors. So it is enough to show that there exists some oracle O that guarantees that at each iteration t of the MWU algorithm, $\mathbf{v}^{(t)}$ is $\sim_V$-preserving. To this end we

need the following easy properties.

Claim 6.11. For all $\sim_V$-preserving V-vectors $\mathbf{v}$ and for all $\sim_W$-preserving W-vectors $\mathbf{w}$, we have that

1. $B\mathbf{v}$ is $\sim_W$-preserving;
2. $\mathbf{w}^T B$ is $\sim_V$-preserving.

Proof of Claim 6.11. We include only the proof of (2) as the proof of (1) is analogous and, indeed, simpler. Let $\mathbf{v} := \mathbf{w}^T B$. An easy computation shows that

$$\mathbf{v}(x,a) = \mathbf{w}(x,+) - \mathbf{w}(x,-) - \sum_{R(\mathbf{x})\in\mathcal{C}_x} \sum_{\substack{1\le i\le \mathrm{ar}(R)\\ \mathbf{x}[i]=x}} \big(\mathbf{w}(R(\mathbf{x}),i,a,+) - \mathbf{w}(R(\mathbf{x}),i,a,-)\big)$$

where we write $\mathcal{C}_x$ to denote the set of all constraints in $\mathcal{C}_\mathbf{X}$ where x appears in the scope, and

$$\mathbf{v}(R(\mathbf{x}),\mathbf{a}) = \sum_{\substack{1\le i\le \mathrm{ar}(R)\\ \mathbf{a}[i]=a}} \big(\mathbf{w}(R(\mathbf{x}),i,a,+) - \mathbf{w}(R(\mathbf{x}),i,a,-)\big)$$

It is immediate to see that, if $\mathbf{w}$ is $\sim_W$-preserving, then $\mathbf{v}(c,\mathbf{a}) = \mathbf{v}(c',\mathbf{a})$ whenever $c \equiv_1 c'$. Let us show that $\mathbf{v}(x,a) = \mathbf{v}(x',a)$ whenever $x \equiv_1 x'$. Since $\mathbf{w}$ is $\sim_W$-preserving we have that $\mathbf{w}(x,s) = \mathbf{w}(x',s)$ for $s \in \{+,-\}$ and hence we only need to show that $\varphi_x(\mathcal{C}_x) = \varphi_{x'}(\mathcal{C}_x)$ where $\varphi_x(\mathcal{C}_x)$ is a shorthand for

$$\sum_{R(\mathbf{x})\in\mathcal{C}_x} \sum_{\substack{1\le i\le \mathrm{ar}(R)\\ \mathbf{x}[i]=x}} \big(\mathbf{w}(R(\mathbf{x}),a,i,+) - \mathbf{w}(R(\mathbf{x}),a,i,-)\big)$$

and $\varphi_{x'}(\mathcal{C}_x)$ is defined analogously.

Now for every $R \in \sigma$, every $S \subseteq [\mathrm{ar}(R)]$, and every class $[c]_{\equiv_1}$ of equivalent constraints, let $\mathcal{C}_{x,R,S,[c]_{\equiv_1}}$ be the set of constraints $R(\mathbf{x})$ in $\mathcal{C}_x \cap [c]_{\equiv_1}$ satisfying that $S = \{i \in [\mathrm{ar}(R)] \mid \mathbf{x}[i] = x\}$. Note that since $\mathcal{C}_x$ and $\mathcal{C}_{x'}$ can be partitioned as the union of sets of this form it is only necessary to show that $\varphi_x(\mathcal{C}_{x,R,S,[c]_{\equiv_1}}) = \varphi_{x'}(\mathcal{C}_{x',R,S,[c]_{\equiv_1}})$ for every choice of R, S,

and $[c]_{\equiv_1}$. To see this it is enough to note that $|\mathcal{C}_{x,R,S,[c]_{\equiv_1}}| = |\mathcal{C}_{x',R,S,[c]_{\equiv_1}}|$ (because $x \equiv_1 x'$) and that, since $\mathbf{w}$ is $\sim_W$-preserving, for every constraint $c' \in [c]_{\equiv_1}$ and every choice of a, i, and s, we have $\mathbf{w}(c', a, i, s) = \mathbf{w}(c, a, i, s)$. ■

Now, consider the oracle $\mathbf{O}$ that, given a W-vector $\mathbf{p}$, returns the V-vector $\mathbf{v}$ defined as $\mathbf{v}[v] = 1$ if $\mathbf{p}^T B[v]$ is positive and $\mathbf{v}[v] = 0$ otherwise. Since $\mathbf{v}$ maximizes $\mathbf{p}^T B\mathbf{v}$ under the restriction $\mathbf{v} \in [0,1]^V$ it follows that $\mathbf{v}$ satisfies (6.2). Furthermore, it is easy to see that if $\mathbf{p}$ is $\sim_W$-preserving then $\mathbf{v}$ is $\sim_V$-preserving.

Now, note that by definition both $\mathbf{w}^{(1)}$ - which is an all-ones W-vector - and $\mathbf{b}$ are $\sim_W$-preserving. It follows easily by induction that for each t, $\mathbf{v}^{(t)}$ is $\sim_V$-preserving and $\mathbf{w}^{(t)}$ is $\sim_W$-preserving. Hence, if we call algorithm 1 iteratively with $T \to \infty$ we obtain in the limit a feasible solution satisfying the conditions of the statement. We note here that, although we have not included explicitly any inequalities requiring that all the variables in $\mathrm{BLP}(\mathbf{X}, \mathbf{A})$ take values in the range $[0,1]$, this is guaranteed by the fact that all the entries of the vector returned by $\mathbf{O}$ are in the range $[0,1]$. This concludes the proof of Theorem 6.9. □

We conclude the section by presenting the theorem on the structure of the solution space of CSP instances.

Theorem 6.12. *Let* $\mathbf{A}$ *be a finite relational structure. The following are equivalent:*

1. $\mathbf{A}$ *has symmetric polymorphisms of all arities.*
2. *For all satisfiable instances* $\mathbf{X}$ *of* $\mathrm{CSP}(\mathbf{A})$ *there exists a homomorphism* $h : X \to A$ *such that for all pairs of variables* $x, x' \in X$*, if* $x \equiv_1 x'$ *then* $h(x) = h(x')$*.*

Proof. (1) ⇒ (2). Let $\mathbf{X}$ be a satisfiable instance of $\mathrm{CSP}(\mathbf{A})$, where $\mathbf{A}$ has symmetric polymorphisms of all arities. Consider the solution of $\mathrm{BLP}(\mathbf{X}, \mathbf{A})$ given by Theorem 6.9 and note that it satisfies $p_x(a) = p_{x'}(a)$ for all $x \equiv_1 x'$ and all $a \in A$. Then, by Corollary 5.8, there is a homomorphism h from $\mathbf{X}$ to $\mathbf{A}$ which satisfies $h(x) = h(x')$ for all $x \equiv_1 x'$.

(2) ⇒ (1). Let $\mathbf{A}$ satisfy (2) and let $k \geq 1$. We shall prove that $\mathbf{A}$ has a symmetric polymorphism of arity k. Let $\mathbf{A}^k$ be the k^{th} power structure

of $\mathbf{A}$. Recall that every homomorphism from $\mathbf{A}^k$ to $\mathbf{A}$ corresponds to a k-ary polymorphism of $\mathbf{A}$, and hence for any $k \geq 1$, the k^{th} power structure is satisfiable since for instance the projection to the first coordinate is a polymorphism of $\mathbf{A}$. Let h be a homomorphism from $\mathbf{A}^k$ to $\mathbf{A}$ which satisfies condition (2). It is easy to show by induction that for every tuple $(a_1, \ldots, a_k) \in A^k$ and every permutation τ of $[k]$, $(a_1, \ldots, a_k) \equiv_1 (a_{\tau(1)}, \ldots, a_{\tau(k)})$. Therefore, $h(a_1, \ldots, a_k) = h(a_{\tau(1)}, \ldots, a_{\tau(k)})$ and h is symmetric as required. □

6.4 The Complexity of DCSP

The primary goal of this section is to prove the main theorem of this chapter, namely, the dichotomy theorem for tractability of DCSP($\mathbf{A}$), which we now state.

Theorem 6.13. DCSP($\mathbf{A}$) *is solvable in polynomial time if and only if* $\mathbf{A}$ *has symmetric polymorphisms of all arities. Otherwise,* DCSP($\mathbf{A}$) *cannot be solved in finite time.*

We show hardness for templates that do not have symmetric polymorphisms of all arities in Section 6.4.1 and tractability of the remaining templates in Section 6.4.2. Moreover, in Section 6.4.3 we extend the decision algorithm so that, additionally, it also solves the search problem. Hence we have:

Theorem 6.14. DCSP-Search($\mathbf{A}$) *is solvable in polynomial time if and only if* $\mathbf{A}$ *has symmetric polymorphisms of all arities. Otherwise,* DCSP-Search($\mathbf{A}$) *cannot be solved in finite time.*

6.4.1 Intractable Templates

In this section we focus on intractable templates, that is, the hardness part of Theorem 6.13.

Theorem 6.15. *Let* $\mathbf{A}$ *be a finite* σ*-structure. If* $\mathsf{Pol}(\mathbf{A})$ *does not contain symmetric operations of all arities, then there is no algorithm that solves* DCSP($\mathbf{A}$) *in finite time.*

Schematically, the proof goes as follows. Assume that $\mathbf{A}$ does not have symmetric polymorphisms of some arity r. Consider the relation U defined by the set of homomorphisms from the the r^{th} power structure $\mathbf{A}^r$ into $\mathbf{A}$, and let $\mathbf{U} = (A^r; U)$. It can be shown (see Remark 6.8) that if DCSP($\mathbf{A}$) is solvable in polynomial (or finite) time then so is DCSP($\mathbf{U}$). Then, we show that there always exist two connected instances of DCSP($\mathbf{U}$), of which one is satisfiable and the other one is not, that are $\equiv_1$-equivalent. Therefore, by Corollary 6.5, any algorithm will return the same output on both instances, meaning that one of these outputs is wrong. Before embarking on the proof we state the following useful combinatorial lemma.

Lemma 6.16. *Let $0 < k < d$ be positive integers. If n is a large enough multiple of k, then there exists a collection $\mathbb{S}$ of n^k k-element subsets of $\{0, 1, \ldots, kn-1\}$ satisfying the following properties:*

(a) *$\mathbb{S}$ contains every k-element subset of $\{0, \ldots, d-1\}$*

(b) *Every element of $\{0, 1, \ldots, kn-1\}$ appears in the same number of sets of $\mathbb{S}$.*

(c) *If $k \geq 2$, the k-uniform hypergraph with vertex set given by $\{0, 1, \ldots, kn-1\}$ and edge set given by $\mathbb{S}$ is connected.*

Proof. If $k = 1$ we can just define $\mathbb{S}$ to be the set containing all singletons in $\{0, 1, \ldots, kn-1\}$ so we can assume that $k \geq 2$. Pick some n that is a multiple of k and consider the subsets of $\{0, 1, \ldots, kn-1\}$. We say that one such set is bad if $S = S + i \pmod{kn}$ for some $i \neq 0$, and good otherwise where the right-hand side of the equation is a shorthand for the set $\{s + i \pmod{kn} \mid s \in S\}$. The following facts hold.

Claim 6.17. If $n > \frac{2(d-1)}{k}$, then all subsets of $\{0, \ldots, d-1\}$ are good.

Proof of Claim 6.17. Let $S \subseteq \{0, \ldots, d-1\}$ and assume that S is bad. Then, there exists i such that $S = S + i \pmod{kn}$. Denote by s_m the smallest element of S. Then, we need $s_m + i \in S$, which implies that $i \leq d-1$. On the other hand, $kn + s_m = x + i$ for some $x \in S$, implying that $kn + s_m \leq 2(d-1) < kn$, a contradiction. ■

Claim 6.18. There are at least n^k good sets.

Proof of Claim 6.18. We say that a bad set (not necessarily of size k) is *canonical* if we can write $S = \{s + c \cdot i \pmod{kn} \mid c = 0, \dots, k-1\}$ for some $s \in S$ and $0 \leq i < kn$. Notice that in this case $|S| \geq 2$. Now, every bad set of size k is a disjoint union of canonical bad sets, and in particular it is the disjoint union of a canonical bad set S_1 of size j for some $j \in \{2, \dots, k\}$, and another bad set S_2 of size $k - j$. Then, to get a loose upper bound on the number of bad sets we notice that there are at most $k(k-2)n$ choices for S_1 (since we have kn choices for the first element and at most $k-2$ choices for the number of elements in S_1), and at most $\binom{kn-j}{k-j} = O(n^{k-2})$ choices for S_2, which leaves us with at most $O(n^{k-1})$ bad sets. This implies that there are at least $\binom{kn}{k} - O(n^{k-1})$ good sets, which, since $k \geq 2$, is at least n^k for n large enough. ■

Therefore, consider the collection of good k-element subsets of $\{0, 1, \dots, kn-1\}$. We say that two sets S, S' are *related* if $S = S' + i \pmod{kn}$ for some $i \neq 0$. Note that, since we are only considering good sets, every class of related sets has exactly kn members and, hence, there are at least n^k/kn many classes. Also it is immediate that every class of related sets satisfies condition (b).

Hence, to construct $\mathbb{S}$ we just need to remove some of the classes of good sets so that we end up having exactly n^k/kn classes, which corresponds to n^k sets. We have to keep all the classes containing one of the sets of condition (b), which is always possible if we pick n large enough so that $\binom{d}{k} \leq n^k$.

It only remains to show that the hypergraph with vertex set $\{0, 1, \dots, kn-1\}$ and edge set $\mathbb{S}$ is connected. To do this it is sufficient to show that each $i \in \{0, 1, \dots, kn-1\}$ is connected to, say, vertex 0 by a walk. This is immediate to see as, for instance, $S_0 = \{0, 1, \dots, k-1\} \in \mathbb{S}$ (because $S_0 \subseteq \{0, 1, \dots, d-1\}$) and therefore $S_i := S_0 + i \in \mathbb{S}$ for all $i \in \{0, 1, \dots, kn-1\}$, hence, the sets $S_0, S_1, \dots, S_{i-k+1}$ witness that 0 and i are connected. □

Proof of Theorem 6.15. Assume that $\mathsf{Pol}(\mathbf{A})$ does not contain symmetric operations of arity $r \geq 2$. Fix any arbitrary order $\mathbf{t}_1, \dots, \mathbf{t}_{|A|^r}$ on the tuples of A^r and consider the relation U defined as

$$\{(f(\mathbf{t}_1), \dots, f(\mathbf{t}_{|A|^r})) \mid f \text{ is a polymorphism of } \mathbf{A} \text{ of arity } r\}$$

This is, U encodes the set of homomorphisms from $\mathbf{A}^r$ to $\mathbf{A}$. Let $\mathbf{U} = (A^r; U)$. Given that U is efpp-definable from $\mathbf{A}$, it follows from Proposition 6.7 and in particular from Remark 6.8 that if DCSP($\mathbf{U}$) is not solvable in finite time then neither is DCSP($\mathbf{A}$).

Partition A^r into equivalence classes where two tuples $\mathbf{t}, \mathbf{t}' \in A^r$ are related, denoted $\mathbf{t} \sim \mathbf{t}'$, if there exists some permutation τ on $\{1, \ldots, r\}$ such that $\mathbf{t}'[i] = \mathbf{t}[\tau(i)]$ for every $i \in \{1, \ldots, r\}$. We shall use $A^r_\sim$ to refer to the collection of classes and $[\mathbf{t}]_\sim$ to refer to the class of tuple $\mathbf{t}$. For every $\mathbf{t} \in A^r$, define $k_{[\mathbf{t}]_\sim}$ to be the number of tuples in $[\mathbf{t}]_\sim$. Then we can choose an integer n large enough such that for every $\mathbf{t} \in A^r$, n is a multiple of $k_{[\mathbf{t}]_\sim}$, and n satisfies Lemma 6.16 for $k = k_{[\mathbf{t}]_\sim}$ and $d = k_{[\mathbf{t}]_\sim} \cdot |A|$. Notice that since $\mathsf{Pol}(\mathbf{A})$ does not contain symmetric operations of all arities, we must have that $|A| \geq 2$, and hence there is some $[\mathbf{t}]_\sim \in A^r_\sim$ such that $k_{[\mathbf{t}]_\sim} \geq 2$.

We are now ready to construct two instances $\mathbf{I}_1$ and $\mathbf{I}_2$ of DCSP($\mathbf{U}$), which are indistinguishable with respect to their iterated degree sequence, but differ with regards to satisfiability. The two instances have the same set of variables, defined to be $\bigcup_{[\mathbf{t}]_\sim \in A^r_\sim} V_{[\mathbf{t}]_\sim}$ where $V_{[\mathbf{t}]_\sim} = \{v^i_{[\mathbf{t}]_\sim} \mid 0 \leq i < k_{[\mathbf{t}]_\sim} n\}$ is a set of $k_{[\mathbf{t}]_\sim} n$ distinct variables. Let R be the unique $|A|^r$-ary relation symbol in the signature of $\mathbf{U}$ which satisfies $R^{\mathbf{U}} = U$.

We start by constructing the constraints of the unsatisfiable instance $\mathbf{I}_1$, which we will do in two stages. First, for every class $[\mathbf{t}]_\sim$, let $\mathbb{S}_{[\mathbf{t}]_\sim}$ be the collection of $n^{k_{[\mathbf{t}]_\sim}}$ sets of cardinality $k_{[\mathbf{t}]_\sim}$ given by Lemma 6.16, as before with $d = k_{[\mathbf{t}]_\sim} \cdot |A|$ and $k = k_{[\mathbf{t}]_\sim}$. Note that each set in $\mathbb{S}_{[\mathbf{t}]_\sim}$ defines naturally a subset of $V_{[\mathbf{t}]_\sim}$ so we shall abuse notation and assume that $\mathbb{S}_{[\mathbf{t}]_\sim}$ is a collection of subsets of $V_{[\mathbf{t}]_\sim}$.

To simplify notation it will be convenient to use $\mathbb{S}$ as a shorthand for the indexed family $\{\mathbb{S}_{[\mathbf{t}]_\sim} \mid [\mathbf{t}]_\sim \in A^r_\sim\}$. Now let S be $\{S_{[\mathbf{t}]_\sim} \mid [\mathbf{t}]_\sim \in A^r_\sim\}$ satisfying $S_{[\mathbf{t}]_\sim} \in \mathbb{S}_{[\mathbf{t}]_\sim}$ for every $[\mathbf{t}]_\sim \in A^r_\sim$. We associate to S the constraint $R(\mathbf{s})$ where the scope $\mathbf{s}$ is constructed as follows. Before defining $\mathbf{s}$ we need some preparation. Recall that when considering the interpretation of R in $\mathbf{U}$ every coordinate of R, and hence of $\mathbf{s}$, is associated to a tuple $\mathbf{t} \in A^r$, so we can talk of the class $[\mathbf{t}]_\sim$ to which each coordinate belongs. In particular, there are $k_{[\mathbf{t}]_\sim}$ coordinates in $\mathbf{s}$ of class $[\mathbf{t}]_\sim$. Hence, by fixing some arbitrary ordering we can use $\mathbf{s}^i_{[\mathbf{t}]_\sim}$, $i = 1, \ldots, k_{[\mathbf{t}]_\sim}$ to refer

to the coordinates in $\mathbf{s}$ of class $[\mathbf{t}]_\sim$. Then, informally, $S_{[\mathbf{t}]_\sim}$ describes which variables from $v^0_{[\mathbf{t}]_\sim}, \ldots, v^{k_{[\mathbf{t}]_\sim} n-1}_{[\mathbf{t}]_\sim}$ to use in order to fill coordinates $\mathbf{s}^i_{[\mathbf{t}]_\sim}$, $i = 1, \ldots, k_{[\mathbf{t}]_\sim}$. Formally, for every $[\mathbf{t}]_\sim \in A^r_\sim$ and each $i = 1, \ldots, k_{[\mathbf{t}]_\sim}$, $\mathbf{s}^i_{[\mathbf{t}]_\sim}$ is assigned to the i^{th} element in $S_{[\mathbf{t}]_\sim}$ in increasing order.

We add such a constraint for each of the $\Pi_{[\mathbf{t}]_\sim \in A^r_\sim} n^{k_{[\mathbf{t}]_\sim}} = n^{(|A|^r)}$ possible choices for S. Therefore, after the first stage we have exactly $n^{(|A|^r)}$ constraints in $\mathcal{C}_{\mathbf{I}_1}$.

In the second stage we add more constraints which will yield the particular symmetry of $\mathbf{I}_1$. Note that every permutation τ on $\{1, \ldots, r\}$ induces a permutation τ' on the coordinates of R in a natural way. Specifically, if coordinate i of R is associated to tuple $\mathbf{t}_i$, then $\tau'(i) = j$ where $\mathbf{t}_j = (\mathbf{t}_i[\tau(1)], \ldots, \mathbf{t}_i[\tau(r)])$. Then, in the second stage, for each permutation τ on $\{1, \ldots, r\}$ and for every constraint $R(\mathbf{s})$ added in the first stage we add the constraint $R(\mathbf{s}')$ where for every $1 \leq i \leq |A|^r$, $\mathbf{s}'[i] = \mathbf{s}[\tau'(i)]$. Therefore, after the second stage we have a total of $m = r! \cdot n^{(|A|^r)}$ constraints as needed.

We now turn to $\mathbf{I}_2$. The constraints are constructed in a similar way, but instead of using the family $\mathbb{S}$ in the first stage, we use a different family $\mathbb{S}'$. In particular, for each class $[\mathbf{t}]_\sim$, $\mathbb{S}'_{[\mathbf{t}]_\sim}$ is obtained by partitioning $V_{[\mathbf{t}]_\sim}$ in $k_{[\mathbf{t}]_\sim}$ blocks of consecutive elements, so that each block has exactly n elements. Then, $\mathbb{S}'_{[\mathbf{t}]_\sim}$ contains the $n^{k_{[\mathbf{t}]_\sim}}$ sets that can be obtained by selecting one element from each block. The second stage is done exactly as in $\mathbf{I}_1$.

We need to show that $\mathbf{I}_1$ and $\mathbf{I}_2$ are connected. Notice that in both instances each constraint $R(\mathbf{s})$ spans all subsets $V_{[\mathbf{t}]_\sim}$, so in both cases it is sufficient to prove that the subgraph of the factor graph induced by the constraint set together with one of these subsets $V_{[\mathbf{t}]_\sim}$ is connected. Pick any $\mathbf{t} \in A^r$ with $k_{[\mathbf{t}]_\sim} \geq 2$. Then, from part (c) of Lemma 6.16, the hypergraph $(V_{[\mathbf{t}]_\sim}; \mathbb{S}_{[\mathbf{t}]_\sim})$ is connected, and since for every $S_{[\mathbf{t}]_\sim} \in \mathbb{S}_{[\mathbf{t}]_\sim}$ there is some constraint in $\mathbf{I}_1$ whose scope contains all the variables in $S_{[\mathbf{t}]_\sim}$, $V_{[\mathbf{t}]_\sim}$ satisfies the aforementioned connectedness condition in $\mathbf{I}_1$, meaning that $\mathbf{I}_1$ is connected. As for $\mathbf{I}_2$, the reasoning is the same except that we are left to show that whenever $k_{[\mathbf{t}]_\sim} \geq 2$, the hypergraph with vertex set $\{0, 1, \ldots, k_{[\mathbf{t}]_\sim} n - 1\}$ and edge set $\mathbb{S}'$ is connected. This is immediate as by construction of $\mathbb{S}'$, any two vertices in $\{0, 1, \ldots, k_{[\mathbf{t}]_\sim} n - 1\}$ either belong

to two separate blocks, in which case they share an edge, or they belong to the same block, in which case they both share an edge with any other vertex in any other block.

Claim 6.19. $\mathbf{I}_1$ and $\mathbf{I}_2$ have the same iterated degree sequence.

Proof of Claim 6.19. Let $[\mathbf{t}]_\sim \in A^r_\sim$. First, we observe that in both instances after the first stage, every variable of $V_{[\mathbf{t}]_\sim}$ appears in the same number of constraints. More specifically, every variable in $V_{[\mathbf{t}]_\sim}$ appears in an n-fraction of the constraints added in the first stage. In the case of instance $\mathbf{I}_1$ this is due to the fact that $\mathbb{S}_{[\mathbf{t}]_\sim}$ satisfies condition (b) in Lemma 6.16 and in instance $\mathbf{I}_2$ this follows from the fact that $\mathbb{S}'_{[\mathbf{t}]_\sim}$ contains all possible sets obtained by choosing an element within each one of the blocks of size n. After the second stage (in both $\mathbf{I}_1$ and $\mathbf{I}_2$ since the second stage is common) every variable in $V_{[\mathbf{t}]_\sim}$ still participates in an n-fraction of the total number of constraints. In addition, it follows easily that the positions of the scope in which a variable in $V_{[\mathbf{t}]_\sim}$ participates distribute evenly among the $k_{[\mathbf{t}]_\sim}$ positions associated to $\mathbf{t}$. That is, in both instances, we have that for every $[\mathbf{t}]_\sim \in A^r_\sim$, every variable $x \in V_{[\mathbf{t}]_\sim}$, and every position i associated to $[\mathbf{t}]_\sim$ there are exactly $m/(nk_{[\mathbf{t}]_\sim})$ constraints in which x appears at position i of the scope, where $m = r! \cdot n^{|A|^r}$. Using this fact it is very easy to prove that $\mathbf{I}_1$ and $\mathbf{I}_2$ have the same iterated degree sequence. Formally, one can show by induction on j that for every $[\mathbf{t}]_\sim \in A^r_\sim$ and $x_1, x_2 \in V_{[\mathbf{t}]_\sim}$, $\delta_j^{\mathbf{I}_1}(x_1) = \delta_j^{\mathbf{I}_2}(x_2)$ and that for any two constraints c_1, c_2 in $\mathbf{I}_1$ and $\mathbf{I}_2$ respectively $\delta_j^{\mathbf{I}_1}(c_1) = \delta_j^{\mathbf{I}_2}(c_2)$. ■

Claim 6.20. $\mathbf{I}_1$ is not satisfiable while $\mathbf{I}_2$ is satisfiable.

Proof of Claim 6.20. We start by showing that $\mathbf{I}_1$ is not satisfiable. Assume by contradiction that there is a homomorphism h from $\mathbf{I}_1$ to $\mathbf{U}$. For each class $[\mathbf{t}]_\sim$, consider the values given by h to the first d variables $v_0, \ldots, v_{d-1}$ in $V_{[\mathbf{t}]_\sim}$. Since $d = k_{[\mathbf{t}]_\sim} \cdot |A|$, it follows by the pigeon-hole principle that at least $k_{[\mathbf{t}]_\sim}$ of these variables are assigned by h to the same element of A. Let $S_{[\mathbf{t}]_\sim}$ be a subset of $V_{[\mathbf{t}]_\sim}$ containing $k_{[\mathbf{t}]_\sim}$ of these variables (we know that this subset belongs to $\mathbb{S}_{[\mathbf{t}]_\sim}$ by condition (a) of Lemma 6.16). Now consider the constraint $R(\mathbf{s})$ in $\mathbf{I}_1$ associated to $S := \{S_{[\mathbf{t}]_\sim} \mid [\mathbf{t}]_\sim \in A^r_\sim\}$, which belongs to $\mathcal{C}_{\mathbf{I}_1}$. If h is a homomorphism, then the restriction of h to $\mathbf{s}$ corresponds to an r-ary polymorphism of $\mathbf{A}$. But h assigns the same value to any two related tuples $\mathbf{t} \sim \mathbf{t}'$, which implies that h is symmetric,

a contradiction.

We now turn our focus to $\mathbf{I}_2$. Let f be any r-ary polymorphism of $\mathbf{A}$ (for example the i^{th} projection, $1 \leq i \leq r$). We shall construct a homomorphism h from $\mathbf{I}_2$ to $\mathbf{U}$ in the following way. Recall that in the definition of $\mathbf{I}_2$ we have partitioned the elements of each set $V_{[\mathbf{t}]_\sim}$ in $k_{[\mathbf{t}]_\sim}$ consecutive blocks. In the first stage, all the elements in each block are placed in the same coordinate of U. So, if $\mathbf{t}_1, \ldots, \mathbf{t}_{|A|^r}$ are the tuples associated to coordinates $1, \ldots, |A|^r$ and hence blocks $1, \ldots, |A|^r$ respectively, then we only need that all variables in the i^{th} block are assigned to $f(\mathbf{t}_i)$ to satisfy all constraints added in the first stage. This assignment also satisfies the constraints added in the second stage, because if f is an r-ary polymorphism of $\mathbf{A}$, then for every permutation τ on $\{1, \ldots, r\}$, the operation g defined as $g(x_1, \ldots, x_r) = f(x_{\tau(1)}, \ldots, x_{\tau(r)})$ is also a polymorphism of $\mathbf{A}$. ■

To sum up, under the assumption that $\mathbf{A}$ does not have symmetric polymorphisms of all arities, we were able to construct two connected $\equiv_1$-equivalent instances $\mathbf{I}_1$ and $\mathbf{I}_2$ of DCSP($\mathbf{A}$), of which $\mathbf{I}_2$ is satisfiable while $\mathbf{I}_1$ is not. It follows from Corollary 6.5 that any distributed algorithm will give the same output on both instances, meaning that no algorithm can solve DCSP($\mathbf{U}$). As anticipated at the beginning of the proof then it follows from Remark 6.8 that there are also no algorithms that solve DCSP($\mathbf{A}$). □

6.4.2 Tractable Templates

In this section we turn our attention to the tractable case. In particular we shall show the following:

Theorem 6.21. *Let* $\mathbf{A}$ *be a finite* σ*-structure such that* $\mathsf{Pol}(\mathbf{A})$ *contains symmetric operations of all arities. Then there is an algorithm* Alg *that solves* DCSP($\mathbf{A}$)*. The total running time, number of rounds, and maximum message size of* Alg *are, respectively,* $\mathcal{O}(n^3 m \log n)$*,* $\mathcal{O}(n^2)$*, and* $\mathcal{O}(m \log n)$ *where* n *and* m *are the number of variables and constraints, respectively, of the input instance.*

Note that this implies the "if" part of Theorem 6.13. Alg is composed of two phases. In the first phase, a distributed version of the colour re-

finement algorithm allows every process to calculate its iterated degree. Then, thanks to Theorem 6.12 we can use the degree of a variable as its ID for the second phase, implying that a distributed adapted version of the jpq-consistency algorithm [Koz21] where messages are tagged with a process's iterated degree solves the decision problem for $\mathbf{A}$.

Distributed Colour Refinement. Let $(\mathbf{X}, \Psi, \psi)$ be an instance of DCSP($\mathbf{A}$) and let $n = |X|$ and $m = |\mathcal{C}_{\mathbf{X}}|$. There is a very natural way to calculate (a finite representation of) an agent's iterated degree in a distributed way, both for variables and for constraints. This is a simple adaptation of the colour refinement algorithm. The algorithm proceeds in rounds. At round $k = 0$, each agent $\psi(v)$ for $v \in X \cup \mathcal{C}_{\mathbf{X}}$ computes $\delta_0^{\mathbf{X}}(v)$ and broadcasts it to all its neighbours. At round $k > 0$, each agent $\psi(v)$ knows the $(k-1)^{\text{th}}$ degrees of its neighbours which it had received in the previous round, uses them to compute $\delta_k^{\mathbf{X}}(v)$, and broadcasts it to its neighbours. If $k = 2n$ (see Proposition 5.2) then for every $v, v' \in X \cup \mathcal{C}_{\mathbf{X}}$ satisfying $\delta_k^{\mathbf{X}}(v) = \delta_k^{\mathbf{X}}(v')$ we have that $v \equiv_1 v'$, which implies that we can essentially regard the k^{th} iterated degree as the unique common ID for all agents controlling equivalent variables or constraints. Then in $2n$ rounds each agent $\psi(v)$ can compute $\delta_{2n}^{\mathbf{X}}$. As we described it, the distributed colour refinement algorithm is not particularly efficient in terms of message complexity. Although this is not necessary to achieve polynomial time, we can reduce the space required to encode $\delta_{2n}^{\mathbf{X}}(v)$.

Lemma 6.22. *Let $s_{\max}$ denote the size of the encoding of $\delta_{2n}^{\mathbf{X}}(v)$. A modified version of the distributed colour refinement algorithm that runs over $\mathcal{O}(n^2)$ rounds achieves $s_{\max} = \mathcal{O}(\log n)$. The time at each round and the maximum size of a message are both bounded above by $\mathcal{O}(ms_{max})$.*

Proof. We describe a variation of the distributed colour refinement algorithm that achieves the required bounds. After computing the k^{th} degree and before proceeding to compute the $(k+1)^{\text{th}}$ degree, all agents broadcast their k^{th} degree to their neighbours. At the next round, every agent broadcasts all the k^{th} degrees received (removing repetitions) to its neighbours so that in $2n$ rounds every agent has received a complete list of all the k^{th} degrees of all nodes. Every agent $\psi(v)$ orders all k^{th} degrees (this can easily be done in such a way that all agents produce the same order),

and sets $\delta_k^{\mathbf{X}}(v)$ to be the rank of its own degree in the order. Then it proceeds to send out this new encoding of $\delta_k^{\mathbf{X}}(v)$ and to calculate $\delta_{k+1}^{\mathbf{X}}(v)$ accordingly.

In this way, we have $s_{\max} = \mathcal{O}(\log(n+m)) = \mathcal{O}(\log n)$. Note that the total number of rounds of this algorithm is $\mathcal{O}(n^2)$ and that, provided every set of degrees is stored as an ordered array, the cost of each computation done locally by an agent at a given round is bounded above by the size, $\mathcal{O}((n+m)s_{\max}) = \mathcal{O}(ms_{\max})$, of the largest message sent. Note that in order to obtain these bounds we have used the standard facts that any finite relational structure $\mathbf{X}$ over a fixed signature σ is such that $m = \mathcal{O}(n^r)$ and $n = \mathcal{O}(rm)$, where r is the largest arity of a relation symbol in σ. □

For the rest of this chapter, when talking about the iterated degree of some $v \in \mathrm{G}_{\mathbf{X}}$ we will mean this reduced size encoding of $\delta_{2n}^{\mathbf{X}}(v)$, which we will denote $\delta_{\infty}^{\mathbf{X}}(v)$ to highlight that it is a fixed point. As we will see, the price of an increase in the number of rounds (from n to n^2) is compensated by the effect of $s_{\max}$ on both time complexity and the size of the messages.

The Distributed Consistency Algorithm. It follows from Theorem 5.12 that if a constraint language $\mathbf{A}$ has symmetric operations of all arities then it has bounded width. Recall that, informally, CSPs of bounded width are those that can be solved by a local consistency algorithm. As mentioned in Section 5.3, it has been shown in [Koz21] that if $\mathbf{A}$ has bounded width and an instance $\mathbf{X}$ of CSP($\mathbf{A}$) satisfies a combinatorial condition called *jpq*-consistency, then $\mathbf{X}$ is satisfiable. Instead of stating literally the result in [Koz21] we shall state a weaker version that uses a different notion of consistency, more suitable to the model of distributed computation introduced in this thesis.

A *set system* S is a subset of $X \times A$. We shall use S_x to denote the set $\{a \in A \mid (x,a) \in S\}$. A walk of length ℓ in $\mathbf{X}$ is any sequence $x_0 c_0 \dots c_{\ell-1} x_\ell$ where $x_0, \dots, x_\ell$ are variables, $c_0, \dots, c_{\ell-1}$ are constraints, and $x_i, x_{i+1} \in c_i$ for every $0 \leq i < \ell$. Note that walks are precisely the walks in the factor graph $\mathrm{G}_{\mathbf{X}}$ (in the standard graph-theoretic sense) starting and finishing in X.

Let S be a set system, p be a walk, and $B \subseteq S_x$ where x is the starting

node of p. The *propagation* of B via p under S, denoted $B +_S p$, is the subset of A defined inductively on the length ℓ of p as follows. If $\ell = 0$ then $B +_S p = B$. Otherwise, $p = p' c_{\ell-1} x_\ell$ where p' is a walk of length $\ell - 1$ ending at $x_{\ell-1}$. Let $c_{\ell-1} = R(\mathbf{x})$. Then we define $B +_S p$ to contain all $b \in A$ such that there exists $a \in B +_S p'$ and $\mathbf{a} \in R^{\mathbf{A}}$ such that for every $1 \leq i \leq \mathrm{ar}(R)$, $\mathbf{a}[i]$ satisfies the following conditions:

1. $\mathbf{a}[i] \in S_{\mathbf{x}[i]}$,
2. if $\mathbf{x}[i] = x_{\ell-1}$ then $\mathbf{a}[i] = a$, and
3. if $\mathbf{x}[i] = x_\ell$ then $\mathbf{a}[i] = b$.

We are now ready to state the result from [Koz21] that we shall use.

Theorem 6.23 (follows from [Koz21])**.** *Let* $\mathbf{X}$ *be an instance of* $\mathrm{CSP}(\mathbf{A})$ *where* $\mathbf{A}$ *has bounded width and let* S *be a set system such that* $S_x \neq \emptyset$ *for every* $x \in X$ *and such that for every walk* p *starting and finishing at the same node* x *and for every* $a \in S_x$*,* a *belongs to* $\{a\} +_S p$*. Then* $\mathbf{X}$ *is satisfiable.*

Our goal is to design a distributed algorithm that either correctly determines that an instance $\mathbf{X}$ is not satisfiable, or produces a set system S verifying the conditions of Theorem 6.23. This is not possible in general due to the fact that agents are anonymous and hence a hypothetical algorithm that would generate a walk in a distributed way would be unable to determine if the initial and end nodes are the same. However, thanks to the structure established by Theorem 6.12, this difficulty can be overcome when $\mathbf{A}$ has symmetric polymorphisms of all arities because, essentially, the iterated degree of a node can act as its unique identifier. To make this intuition precise we will need to introduce a few more definitions.

We say that a pair $(x, a) \in S$ is *S-supported* if for every walk p starting at x and finishing at a node y with $x \equiv_1 y$, we have that $\{a\} +_S p$ contains a.

Remark 6.24

We note that if $(x, a) \in S$ is not S-supported and $p = x_0 c_0 \dots x_\ell$ is a walk of minimal length among all walks witnessing that (x, a) is not

S-supported then $\ell \leq n2^{|A|}$. Indeed if we let $B_i = \{a\} + x_0c_0 \dots x_i$, $i = 0, \dots, \ell$ then we have that $(x_i, B_i) \neq (x_j, B_j)$ for every $0 \leq i < j \leq \ell$, since otherwise the shorter walk $x_0c_0 \dots x_ic_j \dots x_\ell$ would contradict the minimality of p. Since there are n choices for each x_i and $2^{|A|}$ choices for B_i, the bound follows.

We say that a set system S is *safe* if for every homomorphism h from $\mathbf{X}$ to $\mathbf{A}$ we have

$$h(x) = h(y) \text{ for all } x, y \in X \text{ with } x \equiv_1 y \Rightarrow h(x) \in S_x \text{ for all } x \in X.$$

Then, we have

Lemma 6.25. *Let S be a safe set system and let $(x, a) \in S$ be a pair that is not S-supported. Then $S \setminus \{(x, a)\}$ is safe.*

Proof. Let h be any homomorphism from $\mathbf{X}$ to $\mathbf{A}$ satisfying $h(y) = h(z)$ for every $y, z \in X$ with $y \equiv_1 z$ and let $p = x_0c_0 \dots x_\ell$ be any walk in S witnessing that (x, a) is not S-supported, (i.e, p is such that $x_0 = x$, $x_0 \equiv_1 x_\ell$, and $a \notin \{a\} +_S p$). Since S is safe we have that $h(y) \in S_y$ for every $y \in X$. It remains to see that $h(x) \neq a$, so that the safety condition remains unaltered when (x, a) is removed. First, it follows easily by induction that for every $1 \leq i \leq \ell$, $h(x_i) \in \{h(x)\} +_S p_i$ where $p_i = x_0c_0 \dots x_i$. Then, since $h(x_\ell) \in \{h(x)\} +_S p$, $h(x) = h(x_\ell)$, and $a \notin \{a\} +_S p$, it follows that $h(x) \neq a$. □

The distributed consistency algorithm (that is, the second phase of Alg) works as follows. Every variable agent $\psi(x)$ maintains a set $S_x \subseteq A$ in such a way that the set system S is guaranteed to be safe at all times. As a result of an iterative process S is modified. We shall use S^i to denote the content of S at the i^{th} iteration, where an iteration is, in turn, a loop of $T = 2n(2^{|A|} + 1) = \mathcal{O}(n)$ consecutive rounds. The rationale behind this exact value will be made clear later. Initially, S^0_x is set to A for every $x \in X$. At iteration i for $i \geq 1$, S^i is obtained by removing all the elements in S^{i-1} that are not S^{i-1}-supported. By Lemma 6.25, S^i is safe. Then, in at most $n|A| = \mathcal{O}(n)$ iterations we shall obtain a fixed point S^∞.

The key observation is that when $\mathbf{A}$ has symmetric polymorphisms of all arities, the satisfiability of $\mathbf{X}$ can be determined from S^∞. Indeed, if

$S_x^\infty = \emptyset$ for some $x \in X$ then we can conclude from the fact that S^∞ is safe and Theorem 6.12 that $\mathbf{X}$ has no solution. Otherwise, S^∞ satisfies the conditions of Theorem 6.23 and, hence, $\mathbf{X}$ is satisfiable.

It remains to see how to compute S^{i+1} from S^i. In an initial preparation step for every iteration, every variable agent $\psi(x)$ sends S_x^i to all its neighbours. To compute S^{i+1} the algorithm proceeds in rounds. All the messages sent are sets containing triplets of the form (δ, a, B) where $a \in A$, $B \subseteq A$, and δ is the iterated degree of some variable $x \in X$. It follows from the fact that there are at most n possibilities for the iterated degree of a variable that the size of each message is $\mathcal{O}(ns_{\max})$.

The agents controlling variables and constraints alternate. That is, variables perform internal and send events at even rounds and receive messages at odd rounds, while constraints perform internal and send events at odd rounds and receive messages at even rounds. More specifically, in round $j = 0$ of iteration i, every variable agent $\psi(x)$ sends to its neighbours the message M containing all triplets of the form $(\delta_\infty^{\mathbf{X}}(x), a, \{a\})$ with $a \in S_x^i$. At round $2j$ for $j > 0$, $\psi(x)$ computes $M = M_1 \cup \cdots \cup M_d$ where $M_1, \ldots, M_d$ are the messages it received at the end of round $2j - 1$. Subsequently, for every triplet $(\delta, a, B) \in M$ with $\delta = \delta_\infty^{\mathbf{X}}(x)$ and $a \notin B$, $\psi(x)$ marks a as 'not S^i-supported'. Finally, it sends message M to all its neighbours. This computation can be done in time $\mathcal{O}(dns_{\max}) = \mathcal{O}(mns_{\max})$ provided that each message is stored as an ordered array.

In round $2j + 1$, every constraint agent $\psi(c)$ computes from the messages M_x (received from each neighbour $\psi(x)$ in the previous round) the set M'_x, which contains for every variable $y \in c$ and every (δ, a, B) in M_y, the triplet $(\delta, a, B +_{S^i} p)$ where $p = ycx$. Finally, it sends to each neighbour $\psi(x)$ the corresponding message M'_x. Note that while $\psi(c)$ doesn't know the address of $\psi(x)$ specifically, knowing the label of the channel that connects them is sufficient to calculate M'_x correctly and send the message accordingly. Moreover, for given y and x, $\psi(c)$ can compute $B +_{S^i} p$ in $\mathcal{O}(1)$ time as $\psi(c)$ knows both S_y^i and S_x^i. Hence, since the arity of c is constant (as σ is fixed) the total running time at iteration $2j + 1$ of a constraint agent $\psi(c)$ is $\mathcal{O}(ns_{\max})$.

Now it is immediate to show by induction that for every $j \geq 0$, every $x \in X$ and every $c \in \mathcal{C}_{\mathbf{X}}$ with $x \in c$ the message sent by $\psi(x)$ to $\psi(c)$ at

the end of round $2j$ is precisely

$$\{(\delta_\infty^{\mathbf{X}}(y), a, \{d\} + p) \mid y \in X, a \in S_y^i, p = y \dots x \text{ has length } j\}$$

and the message sent by $\psi(c)$ to $\psi(x)$ at the end of round $2j+1$ is precisely

$$\{(\delta_\infty^{\mathbf{X}}(y), a, \{d\} + p) \mid y \in X, a \in S_y^i, p = y \dots cx \text{ has length } j+1\}.$$

By Remark 6.24 only $2n2^{|A|} = T - 2n$ iterations are needed to identify all elements in S^i that are not S^i-supported. Hence, after exactly $T - 2n$ rounds every variable agent $\psi(x)$ computes S_x^{i+1} by removing all the elements in S^i that are marked as "not S^i-supported". If $S_x^{i+1} = \emptyset$, then $\psi(x)$ initiates a wave, which is propagated by all its neighbours, broadcasting that an inconsistency was detected. In this case, in at most $2n$ additional rounds all agents can correctly declare that $\mathbf{X}$ is unsatisfiable. Otherwise, a new iteration begins.

To sum up, the distributed consistency algorithm consists of $\mathcal{O}(n)$ iterations consisting, each, of $\mathcal{O}(n)$ rounds where the total running time for internal events at a given round is $\mathcal{O}(mns_{\max})$ and the maximum size of each message transmitted is $\mathcal{O}(ns_{\max})$. Together with the bounds given by Lemma 6.22 for the distributed colour refinement phase, this completes the proof of Theorem 6.21.

6.4.3 The Search Algorithm

We conclude by presenting the proof of Theorem 6.14. The hardness part follows immediately from Theorem 6.13 as the search problem is as difficult as the decision problem. For the positive result we shall present an adaptation of the algorithm solving the decision version. Let $(\mathbf{X}, \Psi, \psi)$ be an instance of DCSP-Search($\mathbf{A}$) where $\mathbf{A}$ has symmetric polymorphisms of all arities. In what follows we shall use intensively the fact that $\mathsf{Pol}(\mathbf{A})$ is closed under composition. Let $J \subseteq A$ be minimal with the property that $f(A) = J$ for some unary polymorphism f in $\mathsf{Pol}(\mathbf{A})$. It is fairly standard to show that for every $r \geq 0$ there is a r-ary symmetric operation g in $\mathsf{Pol}(\mathbf{A})$ such that $g(x, \dots, x) = x$ for every $x \in J$. Indeed, let f satisfy $f(A) = J$ and let g' be any r-ary symmetric polymorphism in $\mathsf{Pol}(\mathbf{A})$. Then the unary operation h defined by $h(x) = f \circ g'(x, \dots, x)$ is a unary

polymorphism of $\mathbf{A}$. By the choice of f we have $h(A) \subseteq J$. We note that $h(J) = J$ since otherwise h^2 would contradict the minimality of f. Consequently, h restricted to J is a (partial) isomorphism, and so h^{-1} preserves all relations of $\mathbf{A}$ restricted to the subdomain J. Hence, the r-ary operation g defined as $g = h^{-1} \circ f \circ g'$ satisfies the claim. This implies that if we enlarge the template by adding all singletons $\{a\}$, $a \in J$, the resulting structure, which we shall denote by $\mathbf{A}'$, still has symmetric polymorphisms of all arities. For convenience we also include A as a unary relation in $\mathbf{A}'$.

The algorithm has two phases. In the first phase it runs the decision algorithm to determine whether the instance is satisfiable. As a byproduct, every variable agent $\psi(x)$ has computed its iterated degree $\delta^{\mathbf{X}}_{\infty}(x)$ and knows as well its rank in a prescribed ordering of all the variables' iterated degrees $\delta^1, \ldots, \delta^r$, $r \leq n$. This (partial) order will be used to coordinate between the agents. An i-agent, $1 \leq i \leq r$ is any agent $\psi(x)$ with $\delta^{\mathbf{X}}_{\infty}(x) = \delta^i$. We also assume a fixed ordering on the elements in A. If the instance is unsatisfiable nothing else remains to be done so from now on we shall assume that the instance is satisfiable.

In the second phase the algorithm searches for a solution. Every variable agent $\psi(x)$ maintains a set $F_x \subseteq A$ with the property that there is a homomorphism h from $\mathbf{X}$ to $\mathbf{A}$ that *falls within* the set system F, i.e, such that $h(x) \in F_x$ for every $x \in X$. Initially every agent $\psi(x)$ sets $F_x = A$ so it is only necessary to make sure that this condition is preserved during the execution of the algorithm. The second phase contains two nested loops. The outer loop has r iterations and the inner loop consists of at most $|A|$ iterations so that we shall use iteration (i, a) to indicate the run of the algorithm at the $i = 1, \ldots, r$ iteration of the outer loop and at the iteration a of the inner loop.

At the beginning of iteration (i, a) every variable agent $\psi(x)$ defines $S_x \subseteq A$ to be $S_x = \{a\}$ whenever $\psi(x)$ is an i-agent and $S_x = F_x$ elsewhere. Then it runs the distributed consistency algorithm starting at S obtaining a fixed point S^{∞}. We note that since all initial sets S_x are relations in $\mathbf{A}'$ and $\mathbf{A}'$ has symmetric polymorphisms of all arities then the obtained fixed point S^{∞} correctly determines whether there exists a homomorphism h that falls within S. Then every i-agent $\psi(x)$ checks whether $S^{\infty}_x = \emptyset$. In case of positive answer nothing else is done and round (i, a) finishes.

Otherwise, there must be a homomorphism h that falls within S, so $\psi(x)$ sets F_x to $\{a\}$ and starts a wave to indicate to all processes that the i^{th} iteration of the outer loop is finished and that the next iteration of the outer loop can start. When the r iterations of the outer loop have been completed the set system F contains only singletons. The map that assigns every variable $x \in X$ to the only element in F_x is necessarily a homomorphism. This concludes the proof of Theorem 6.14.

7 Weisfeiler-Leman Invariant CSPs

In this chapter we establish a close link between LP relaxations for isomorphism and homomorphism by decomposing solutions to the SA^1 *program into sequences of homomorphisms and fractional isomorphisms. We use this decomposition to characterize Weisfeiler-Leman invariant CSPs and to obtain back the results of Chapter 6 in a streamlined manner.*

7.1 Introduction

Linear programming relaxations, among other relaxations such as SDP-based, have been largely used in the study of both the isomorphism and the homomorphism problem. The motivation for this chapter stems from the similarity between the linear program defining fractional isomorphism (see Section 5.1) and the relaxations of homomorphism defined in Section 5.2. For the sake of simplicity we will introduce this similarity for the case of graphs first.

Recall that the isomorphism problem for two graphs $\mathbf{X}$, $\mathbf{A}$ can be reformulated as an integer program which asks whether there exists a permutation matrix P such that $PN_{\mathbf{X}} = N_{\mathbf{A}}P$, where $N_{\mathbf{X}}$ and $N_{\mathbf{A}}$ are the adjacency matrices of $\mathbf{X}$, $\mathbf{A}$ respectively. If we relax this condition to only require that P is doubly stochastic, we obtain the fractional isomorphism linear program. Now, this condition can be equivalently expressed as the existence of a pair of doubly stochastic matrices P and Q such that $PM_{\mathbf{X}} = M_{\mathbf{A}}Q$ and $M_{\mathbf{X}}Q^T = P^T M_{\mathbf{A}}$, where $M_{\mathbf{X}}$ denotes the *incidence* matrix of $\mathbf{X}$.

If we relax this condition further to only require that P and Q are left stochastic and, additionally, we drop the second equation, then we obtain a relaxation of graph homomorphism equivalent to the SA^1 system introduced in Section 5.2.[1] With some minor variations depending on whether the objective function is present (as in MaxCSP) or not and how repeated elements in a tuple are treated, this LP formulation has been extensively used [KMTV11, KOT+12, DK13, DKM18, GT18, BGWŽ20].

The main result of this chapter is a two-fold characterization of the first level of the Sherali-Adams hierarchy applied to the homomorphism problem for relational structures: algebraic, in terms of the linear program described above, and combinatorial, relating SA^1 to the equivalence relation $\equiv_1$. In particular, we show that for any two similar structures $\mathbf{X}$, $\mathbf{A}$ the linear program $\mathrm{SA}^1(\mathbf{X}, \mathbf{A})$ is feasible if and only if there exists a sequence of structures $\mathbf{Y}_0, \mathbf{Y}_1, \ldots, \mathbf{Y}_n$ with $\mathbf{X} = \mathbf{Y}_0$, $\mathbf{A} = \mathbf{Y}_n$, and for every $i < n$, $\mathbf{Y}_i$ is either homomorphic or fractionally isomorphic to $\mathbf{Y}_{i+1}$. Moreover, if such a chain of structures exists then there exists a chain of length 3.

Each of these morphisms - homomorphism and fractional isomorphism - from $\mathbf{Y}_i$ to $\mathbf{Y}_{i+1}$ can be naturally associated with a rational matrix of dimensions $|Y_i| \times |Y_{i+1}|$. It can be calculated that the product of these matrices is a matrix associated to a solution to the $\mathrm{SA}^1(\mathbf{X}, \mathbf{A})$ linear program. This is why we regard this result as a decomposition theorem. In Chapter 8 we will obtain a similar decomposition for valued structures, and in Chapter 9 we will extend this decomposition to the higher levels of the SA hierarchy (for crisp structures).

The established connection between SA^1 and $\equiv_1$ is essentially due to the fact that the LP relaxation of homomorphism inherits the symmetries of $\mathbf{X}$ and $\mathbf{A}$. However, the main interest of this result lies in the opposite direction: that is, the fact that SA^1 is able to certify that $\mathbf{X}$ is not homomorphic to $\mathbf{A}$ unless $\mathbf{X}$ belongs to the backwards closure of $\mathbf{A}$ under

[1] We remark that in [BD21a], this algebraic condition - and its equivalent characterizations - was alternatively phrased as the existence of a fractional homomorphism, to stress that it is a "fractional" relaxation of homomorphism in the same way that fractional isomorphism is a relaxation of isomorphism. Nonetheless, in this thesis we avoid this terminology as it clashes with the notion of fractional homomorphism defined in Section 8.3 as a unary fractional polymorphism.

homomorphism and $\equiv_1$-equivalence (or, even more strongly, unless $\mathbf{X}$ is homomorphic to a structure $\mathbf{Y}_1$ which is $\equiv_1$-equivalent to a structure $\mathbf{Y}_2$ which in turn is homomorphic to $\mathbf{A}$).

In fact, we will conclude this chapter by applying our results to extend Theorem 5.6 with two additional characterizations: the first, in terms of Weisfeiler-Leman invariance, is a direct consequence of the decomposition theorem. The second, in terms of solvability by distributed algorithms, shows that the complexity classification obtained for DCSP in Chapter 6 can be derived again using substantially different techniques.

7.2 The Decomposition Theorem

We now state the decomposition theorem for SA^1. Note that for graphs, condition (5) below is naturally seen as the homomorphism counterpart of the notion of fractional isomorphism (see condition (1) in Lemma 5.4).

Theorem 7.1. *Let $\mathbf{X}$, $\mathbf{A}$ be σ-structures. Then, the following are equivalent:*

1. $\mathrm{SA}^1(\mathbf{X}, \mathbf{A})$ *is feasible;*
2. *There exist left stochastic matrices P, Q such that for every $\ell = (S, R) \in \mathcal{L}_\sigma$ it holds that $PM^\ell_{\mathbf{X}} \leq \sum_{\ell'} M^{\ell'}_{\mathbf{A}} Q$, where ℓ' ranges over all $(S', R) \in \mathcal{L}_\sigma$ with $S \subseteq S'$;*
3. *There exists a sequence of structures $\mathbf{Y}_0, \ldots, \mathbf{Y}_n$ such that $\mathbf{Y}_0 = \mathbf{X}$, $\mathbf{Y}_n = \mathbf{A}$, and for all $i = 0, \ldots, n-1$ we have that $\mathbf{Y}_i \to \mathbf{Y}_{i+1}$ or $\mathbf{Y}_i \equiv_1 \mathbf{Y}_{i+1}$;*
4. *There exists a pair of structures $\mathbf{Y}_1, \mathbf{Y}_2$ such that $\mathbf{X} \to \mathbf{Y}_1$, $\mathbf{Y}_1 \equiv_1 \mathbf{Y}_2$, and $\mathbf{Y}_2 \to \mathbf{A}$.*

If in addition none of the relations in $\mathbf{X}$ and $\mathbf{A}$ have repetitions, then the following condition is also equivalent:

5. *There exist left stochastic matrices P, Q such that for every $\ell \in \mathcal{L}_\sigma$ it holds that $PM^\ell_{\mathbf{X}} = M^\ell_{\mathbf{A}} Q$.*

Proof. The equivalence (1) ⇔ (2) is merely syntactic. In particular we shall show that there is a one-to-one satisfiability-preserving correspondence between pairs of matrices and variable assignments of $\mathrm{SA}^1(\mathbf{X}, \mathbf{A})$. However, we first need to massage a bit the two formulations. First, we can assume that for every $R(\mathbf{x}) \in \mathcal{C}_{\mathbf{X}}$ and $R'(\mathbf{a}) \in \mathcal{C}_{\mathbf{A}}$, the corresponding entry in Q is null unless $R = R'$ and $f(\mathbf{x}) = \mathbf{a}$ for some $f : \{\mathbf{x}\} \to \{\mathbf{a}\}$, since otherwise it is impossible that Q forms part of a feasible solution. Secondly, we note that the feasibility of $\mathrm{SA}^1(\mathbf{X}, \mathbf{A})$ does not change if in (SA3) we replace $=$ by $\leq$ obtaining a new set of inequalities (which to avoid confusion we shall denote by (SA3′)) and, in addition, we add for every $R(\mathbf{x}) \in \mathcal{C}_{\mathbf{X}}$ the equality

$$\sum_{f:\{\mathbf{x}\}\to A} p_{R(\mathbf{x})}(f) = 1. \tag{SA5}$$

Finally, note that in $\mathrm{SA}^1(\mathbf{X}, \mathbf{A})$ we can ignore (SA2).

Then we can establish the following correspondence between pairs of matrices P, Q and assignments $\mathrm{SA}^1(\mathbf{X}, \mathbf{A})$: for every $x \in X$ and $a \in A$, we set $p_x(a) = P[a, x]$ and for every $R(\mathbf{x}) \in \mathcal{C}_{\mathbf{X}}$ and $f : \{\mathbf{x}\} \to A$ we define $p_{R(\mathbf{x})}(f) = Q[R(f(\mathbf{x})), R(\mathbf{x})]$. Then, it is easy to see that (SA3′) corresponds to $PM^{\ell}_{\mathbf{X}} \leq \sum_{\ell' \in \mathcal{L}_\ell} M^{\ell'}_{\mathbf{A}} Q$ for every $\ell \in \mathcal{L}_\sigma$ (where $\mathcal{L}_{(S,R)} := \{(S', R) \in \mathcal{L}_\sigma \mid S \subseteq S'\}$), P being left stochastic corresponds to (SA1), and Q being left stochastic corresponds to (SA5).

The equivalence (1) ⇔ (5) is obtained as in (1) ⇔ (2). We just need to notice that when $\mathbf{X}$ and $\mathbf{A}$ have no loops, then none of the entries in $M_{\mathbf{X}}$ and $M_{\mathbf{A}}$ contain any label $\ell = (S, R) \in \mathcal{L}_\sigma$ where $|S| > 1$ and hence it is only necessary to consider labels $\ell = (S, R) \in \mathcal{L}_\sigma$ where S is a singleton. Observe that, in this case, the equation in (2) becomes $PM^{\ell}_{\mathbf{X}} \leq M^{\ell}_{\mathbf{A}} Q$ since for every label $\ell = (S, R)$ where S is a singleton, the only label (S', R) with $S \subseteq S'$ and $M^{\ell}_{\mathbf{A}}$ not a zero matrix is ℓ itself. Finally, in order to replace $\leq$ by $=$ in the previous equation we just need to use (SA3) instead of (SA3′).

Notice that (4) ⇒ (3) is trivial.

The proof of (3) ⇒ (2) is by induction on n. If $n = 0$ the claim is immediate, so assume that $n \geq 1$. Let $\mathbf{Y}_0, \mathbf{Y}_1, \ldots, \mathbf{Y}_n$ be a sequence of structures satisfying (3). By the induction hypothesis, there exist left stochastic matrices P, Q such that $PM^{\ell}_{\mathbf{Y}_1} \leq \sum_{\ell' \in \mathcal{L}_\ell} M^{\ell'}_{\mathbf{Y}_n} Q$ for all $\ell \in \mathcal{L}_\sigma$.

If $\mathbf{Y}_0 \equiv_1 \mathbf{Y}_1$ then it follows from Theorem 5.4 that there exist doubly stochastic matrices P' and Q' such that $P'M^{\ell}_{\mathbf{Y}_0} = M^{\ell}_{\mathbf{Y}_1}Q'$ for all $\ell \in \mathcal{L}_\sigma$, and so it is easy to verify that PP', QQ' are such that (2) holds. Assume that $\mathbf{Y}_0 \to \mathbf{Y}_1$. We shall show that there exist left stochastic matrices P' and Q' such that for all $a \in A$, for all $R(\mathbf{x}) \in \mathcal{C}_{\mathbf{X}}$, and for all $\ell \in \mathcal{L}_\sigma$ there exists $\hat{\ell} = \hat{\ell}(a, R(\mathbf{x}), \ell) \in \mathcal{L}_\ell$ such that $PM^{\ell}_{\mathbf{X}}[a, R(\mathbf{x})] \leq M^{\hat{\ell}}_{\mathbf{A}}Q[a, R(\mathbf{x})]$. Assuming that this holds, again it follows by the induction hypothesis that PP', QQ' are left stochastic matrices such that $PP'M^{\ell}_{\mathbf{Y}_0} \leq \sum_{\ell' \in \mathcal{L}_\ell} M^{\ell'}_{\mathbf{Y}_n}QQ'$ for all $\ell \in \mathcal{L}_\sigma$.

Let h be a homomorphism from $\mathbf{X}$ to $\mathbf{A}$. We define $P'[a, x] = 1$ if $a = h(x)$ and $P'[a, x] = 0$ otherwise. Similarly, we set $Q'[R'(\mathbf{a}), R(\mathbf{x})] = 1$ if $\mathbf{a} = h(\mathbf{x})$ and $R' = R$ and $Q'[R'(\mathbf{a}), R(\mathbf{x})] = 0$ otherwise. It is easy to see that P' and Q' are left stochastic. Now let $\ell = (S, R) \in \mathcal{L}_\sigma$, $a \in A$ and $R(\mathbf{x}) \in \mathcal{C}_{\mathbf{X}}$. If $M^{\ell}_{\mathbf{X}}[x, R(\mathbf{x})] = 0$ for all $x \in X$ then $PM^{\ell}_{\mathbf{X}}[a, R(\mathbf{x})] = 0$ and there is nothing to prove. So we can assume that there is $x \in X$ such that for all $i \in [\mathrm{ar}(R)]$, $\mathbf{x}[i] = x$ if and only if $i \in S$. Then we have that $PM^{\ell}_{\mathbf{X}}[a, R(\mathbf{x})] = 1$ if $a = h(x)$, and $PM^{\ell}_{\mathbf{X}}[a, R(\mathbf{x})] = 0$ otherwise. Again in the latter case there is nothing to prove so let us assume that $a = h(x)$. It follows that $h(\mathbf{x})[i] = a$ for all $i \in S$ and hence there exists $\hat{\ell} = (R, S')$ with $S \subseteq S'$ such that $M^{\hat{\ell}}_{\mathbf{A}}[a, R(h(\mathbf{x}))] = 1$, which completes the proof.

We first presented a construction which witnesses (1) ⇒ (4) in [BD21a]. Subsequently, in [BB22] we were able to obtain a simplified version which also applies to the more general framework of valued structure. Here we present the non-valued version of the construction from [BB22], as it is substantially simpler. The valued version of this construction will be presented in Chapter 8.

Assume that $\mathrm{SA}^1(\mathbf{X}, \mathbf{A})$ is feasible. Let $p_x(a)$, $p_{R(\mathbf{x})}(\mathbf{a})$ form a feasible solution of $\mathrm{SA}^1(\mathbf{X}, \mathbf{A})$ and let $m > 0$ be an integer such that all the values $mp_x(a)$ and $mp_{R(\mathbf{x})}(\mathbf{a})$ are (non-negative) integers.

We define the universe of both structures $\mathbf{Y}_1$ and $\mathbf{Y}_2$ as $Y_1 = Y_2 = [m] \times X$. The structure $\mathbf{Y}_1$ is simply a disjoint union of m copies of $\mathbf{X}$: for every $R \in \sigma$, $\mathbf{x} \in X^{\mathrm{ar}(R)}$ and $k \in [m]$ we set

$$((k, \mathbf{x}[1]), (k, \mathbf{x}[2]), \ldots, (k, \mathbf{x}[\mathrm{ar}(R)])) \in R^{\mathbf{Y}_1} \Leftrightarrow \mathbf{x} \in R^{\mathbf{X}}.$$

Observe that clearly $\mathbf{X} \to \mathbf{Y}_1$ since for all $(k, x) \in Y_1$, any map $h_k : x \mapsto$

(k, x) is a homomorphism. Also notice that for all $k \in [m]$, the iterated degree $\delta^{\mathbf{Y}_1}(k, x)$ is the same as $\delta^{\mathbf{X}}(x)$.

The structure $\mathbf{Y}_2$ is a "twisted" version of $\mathbf{Y}_1$ (the construction is a version of the twisted product from [Kun13]). For every $x \in X$, fix a tuple $\mathbf{p}_x \in A^m$ in which $a \in A$ appears exactly $mp_x(a)$ times – note that this is possible since the $mp_x(a)$ sum up to m by (SA1). Moreover, for every $R(\mathbf{x}) \in \mathcal{C}_{\mathbf{X}}$, denote $r = \mathrm{ar}(R)$, and consider an $m \times r$ matrix $T = T(R(\mathbf{x}))$ that has, for each $\mathbf{a} \in A^r$, exactly $mp_{R(\mathbf{x})}(\mathbf{a})$ rows equal to $\mathbf{a}$. Note that then all the rows of T are elements of $R^{\mathbf{A}}$ by (SA4), and that the i^{th} column of T contains $a \in A$ exactly $mp_{\mathbf{x}[i]}(a)$ times by (SA3), in other words, the multiset of elements of this columns is equal to $\{\{\mathbf{p}_{\mathbf{x}[i]}\}\}$. In particular, T indeed has m rows. Moreover, if $\mathbf{x}[i] = \mathbf{x}[j]$, then the i^{th} and j^{th} columns of T are identical by (↻). It follows that there are permutations $\rho_1, \ldots, \rho_r : [m] \to [m]$ such that

1. for every $k \in [m]$, $(\mathbf{p}_{\mathbf{x}[1]}[\rho_1(k)], \mathbf{p}_{\mathbf{x}[2]}[\rho_2(k)], \ldots, \mathbf{p}_{\mathbf{x}[r]}[\rho_r(k)])$ is equal to the k^{th} row of T;

2. for every $i, j \in [r]$, if $\mathbf{x}[i] = \mathbf{x}[j]$ then $\rho_i = \rho_j$.

Then, for every $R \in \sigma$, $\mathbf{x} \in X^{\mathrm{ar}(R)}$ and $k \in [m]$ we set

$$((\rho_1(k), \mathbf{x}[1]), (\rho_2(k), \mathbf{x}[2]), \ldots, (\rho_r(k), \mathbf{x}[r])) \in R^{\mathbf{Y}_2} \Leftrightarrow \mathbf{x} \in R^{\mathbf{X}}.$$

We define $h : Y_2 \to A$ by $h(k, x) = \mathbf{p}_x[k]$ for all $k \in [m]$ and $x \in X$. It is easy to see that the image of any tuple in $R^{\mathbf{Y}_2}$ under h is a row of $T(R(\mathbf{x}))$ for some $R(\mathbf{x}) \in \mathcal{C}_{\mathbf{X}}$, and hence belongs to $R^{\mathbf{A}}$. In particular, for every $\mathbf{y} \in R^{\mathbf{Y}_2}$, there exists some $\mathbf{x} \in R^{\mathbf{X}}$ and $k \in [m]$ such that $\mathbf{y}[i] = (\rho_i(k), \mathbf{x}[i])$ for all $i \in [\mathrm{ar}(R)]$. Then, $h(\mathbf{y}[i]) = \mathbf{p}_{\mathbf{x}[i]}(\rho_i(k))$ and hence $h(\mathbf{y})$ is the k^{th} row of $T(R(\mathbf{x}))$ by (1).

Moreover, for all $(k, x) \in Y_2$ the iterated degree $\delta^{\mathbf{Y}_2}(k, x)$ is the same as $\delta^{\mathbf{X}}(x)$ and hence the same as in $\mathbf{Y}_1$ (note here that item (2) above guarantees that repeated entries are handled correctly). It follows that $\mathbf{Y}_1 \equiv_1 \mathbf{Y}_2$, and the proof is concluded. □

7.3 Weisfeiler-Leman invariant CSPs

The principal novelty of our result is that it leads to an alternative combinatorial characterization of solvability by the first level of the Sherali-Adams relaxation: that is, it allows us to extend Theorem 5.6 and ultimately improve our understanding of solvability of CSPs by linear programs. A concrete application is the answer to the following question: for which structures $\mathbf{A}$ is $\mathrm{CSP}(\mathbf{A})$ closed under $\equiv_1$-equivalence? This question arises in the context of distributed CSPs (see Chapter 6). In fact, the connection between the Weisfeiler-Leman algorithm and distributed computation goes back to the influential paper of Angluin on networks of processors [Ang80]. We saw in Chapter 6 that any distributed message passing algorithm necessarily behaves in an identical manner on every two input instances that are $\equiv_1$-equivalent (see Corollary 6.5). Hence, it follows that $\mathrm{CSP}(\mathbf{A})$ can only be solved by a distributed algorithm if $\mathrm{CSP}(\mathbf{A})$ is closed under $\equiv_1$-equivalence. We then obtain the following characterization of Weisfeiler-Leman invariant CSPs:

Theorem 7.2. *Let $\mathbf{A}$ be a fixed finite σ-structure. The following are equivalent:*

(i) *There is a distributed algorithm that solves* $\mathrm{CSP}(\mathbf{A})$. *Moreover, in this case, there is a polynomial-time distributed algorithm that solves* $\mathrm{CSP}(\mathbf{A})$.

(ii) $\mathrm{CSP}(\mathbf{A})$ *is closed under* $\equiv_1$*-equivalence;*

(iii) SA^1 *decides* $\mathrm{CSP}(\mathbf{A})$*;*

(iv) BLP *decides* $\mathrm{CSP}(\mathbf{A})$*;*

(v) $\mathbf{A}$ *has symmetric polymorphisms of all arities.*

Proof. The implication (ii) $\Rightarrow$ (iii) is an immediate corollary of the decomposition theorem 7.1.

The equivalence of (iii), (iv) and (v) is precisely Theorem 5.6.

The implication (v) $\Rightarrow$ (i) is precisely Theorem 6.21. In fact, we will see in Chapter 8 that there is also a simpler proof of this implication that

does not require Theorem 5.12 and the deep theory of bounded width structures.

(i) ⇒ (ii). From the nature of the distributed model, it follows that agents with the same iterated degree will be in the same state at any time during the execution of any distributed algorithm. Therefore, if (i) holds and $\mathbf{X} \equiv_1 \mathbf{Y}$ are connected, then a terminating distributed algorithm will report the same decision when run on input $\mathbf{X}$ or $\mathbf{Y}$ (see Proposition 6.3 and Corollary 6.5), so we obtain that (ii) holds for all connected $\mathbf{X}$, $\mathbf{Y}$. We now show how (ii) in its full generality follows from (ii) restricted to connected $\mathbf{X}$ and $\mathbf{Y}$.

We claim that $\mathbf{X} \to \mathbf{A}$ iff $\mathbf{Y} \to \mathbf{A}$ whenever $\mathbf{X}$ and $\mathbf{Y}$ are weakly congruent and connected. The claim clearly holds when $|X| = 1$ or $|Y| = 1$ (since this would imply that $\mathbf{X} = \mathbf{Y}$), so assume that $|X|, |Y| \geq 2$. For any positive integer k, we define a connected σ-structure $\mathbf{X}^{(k)}$ (and similarly $\mathbf{Y}^{(k)}$ for $\mathbf{Y}$) as follows. Let $X = \{x_0, x_1, \ldots, x_{|X|-1}\}$ and let the universe of $\mathbf{X}^{(k)}$ be $\{0, 1, \ldots, k-1\} \times X$. Then, for each constraint $R^{\mathbf{X}}(x_{i_1}, \ldots, x_{i_{\mathrm{ar}(R)}}) \in \mathcal{C}_{\mathbf{X}}$ and each $j \in \{0, 1, \ldots, k-1\}$, we add to $\mathcal{C}_{\mathbf{X}^{(k)}}$ the constraints

$$R^{\mathbf{X}^{(k)}}((j, x_{i_1}), \ldots, (j, x_{i_{\mathrm{ar}(R)}}))$$

and

$$R^{\mathbf{X}^{(k)}}((j + i_1 \pmod k, x_{i_1}), \ldots, (j + i_{\mathrm{ar}(R)} \pmod k, x_{i_{\mathrm{ar}(R)}})).$$

Then $\mathbf{X}$ and $\mathbf{X}^{(k)}$ are homomorphically equivalent, since for instance the map f given by $f(x) = (0, x)$ is a homomorphism from $\mathbf{X}$ to $\mathbf{X}^{(k)}$, and the projection from $\{0, 1, \ldots, k-1\} \times X$ onto X is a homomorphism in the opposite direction. Moreover, notice that by construction $\mathbf{X}^{(k)}$ is connected. Finally, if k is large enough ($k \geq |X|$ suffices), then the iterated degree of (j, x_i) in $\mathbf{X}^{(k)}$ is obtained from the iterated degree of x_i in $\mathbf{X}$ by multiplying all the variable multisets in each of the elements of $\delta^{\mathbf{X}}(x_i)$ by 2 (in each inductive step in the definition of iterated degree). It follows from the weak congruence of $\mathbf{X}$ and $\mathbf{Y}$ that, for all k', $\mathbf{Y}^{(k'|X|)}$ and $\mathbf{X}^{(k'|Y|)}$ are connected and, when k' is large enough, have the same iterated degree sequence. By item (ii) for connected structures, we get that $\mathbf{X}^{(k'|Y|)} \to \mathbf{A}$ iff $\mathbf{Y}^{(k'|X|)} \to \mathbf{A}$ and therefore $\mathbf{X} \to \mathbf{A}$ iff $\mathbf{Y} \to \mathbf{A}$.

Now it follows from Remark 5.3 that if $\mathbf{X} \equiv_1 \mathbf{Y}$, then for every connected component $\mathbf{X}'$ of $\mathbf{X}$ there is a connected component $\mathbf{Y}'$ of $\mathbf{Y}$ such that $\mathbf{X}'$ and $\mathbf{Y}'$ are weakly congruent and vice versa. By observing that $\mathbf{X} \to \mathbf{A}$ if and only if all the connected components of $\mathbf{X}$ are homomorphic to $\mathbf{A}$ (and similarly for $\mathbf{Y}$), we obtain that $\mathbf{X} \to \mathbf{A}$ iff $\mathbf{Y} \to \mathbf{A}$ as required. □

8 Weisfeiler-Leman Invariant Promise Valued CSPs

In this chapter we deal with valued structures. We start by introducing fractional operations and their properties. We then proceed to generalize the decomposition theorem and the characterization of WL-invariance to Promise Valued CSPs. We conclude by discussing where the equivalence of BLP *and* SA^1 *breaks in the Promise Valued setting.*

8.1 Introduction

The Promise Valued Constraint Satisfaction Problem (PVCSP) combines and generalizes both Promise and Valued CSPs. A template is a pair of valued structures of the same signature and the problem is, given a sum such as (2.1) and a rational number τ, to distinguish sums whose minimum computed in $\mathbf{A}$ is at most τ from those whose minimum computed in $\mathbf{B}$ is greater than τ.

We believe that the PVCSP is an extremely promising research direction for two reasons. First, it is very broad: it includes, for example, all constant factor approximation problems for MaxCSP (both the version where the aim is to approximately maximize the number of satisfied constraints and the version where the aim is to approximately minimize the number of unsatisfied constraints). Second, the approach via generalized polymorphisms, so successful in both the promise and the valued cases, is still available [Kaz22] (this work is not yet published). The only published work on PVCSP that we are aware of (other than [BB22]) is [VŽ21] where

the authors, among other results, generalize the equivalence of (iv) and (v) in Theorem 7.2 to the PVCSP setting and even consider the more general infinite-domain case (recall that this equivalence was already known to hold in both the VCSP and the PCSP frameworks by [KTŽ15] and [BBKO21] respectively).

The main result of this chapter, Theorem 8.7, lifts the equivalence of (i), (ii), and (iii) in Theorem 7.2 to the PVCSP framework (and hence for PCSPs and VCSPs as well). The generalization of implication (i) ⇒ (ii) for valued structures is similar to the crisp case, but requires some extra care when taking disconnected structures into consideration. For the implication (ii) ⇒ (iii) we also employ the approach of Chapter 7 and "decompose" the solution to the SA^1 relaxation of a PVCSP into three components. One component is a kind of morphism, called here a dual fractional homomorphism, which appeared before in the context of VCSPs with left-hand side (i.e., structural) restrictions [CRŽ22].[1] The decomposition theorem for valued structures, stated as Theorem 8.6, might be of independent interest.

The distributed algorithm that we design to prove (iii) ⇒ (i) is different from the one used for the CSP case. The algorithm of Chapter 6 relies on a deep theorem from the algebraic CSP theory about the strength of jpq-consistency algorithm (see Theorem 6.23), yet this approach is no longer applicable, even in the (non-valued) PCSP setting [AD22]. However, we show that a substantially more simple idea of directly computing an adjusted form of SA^1 – while more distant in spirit from the message-passing systems approach – works even in the more general PVCSP framework.

Surprisingly, the implication (iii) ⇒ (iv) is no longer true for PVCSPs: in Example 8.8 we present a PVCSP template that is decided by SA^1 but not decided by BLP. The converse implication remains valid since SA^1 is a stronger relaxation than BLP.

[1] [CRŽ22] uses the terminology "inverse fractional homomorphism", however we feel that "dual" might better fit the meaning of this concept.

8.2 Promise Valued CSPs

PVCSP. Let $\mathbb{Q}$ be the set of rational numbers. We denote by $\mathbb{Q}_{\geq 0}$ the set of non-negative rationals and by $\mathbb{Q}_\infty$ the set $\mathbb{Q} \cup \{\infty\}$, where ∞ is an additional symbol interpreted as a positive infinity. We set $0 \cdot \infty = 0$ and $c \cdot \infty = \infty$ for all $c > 0$.

A k-ary *valued relation* on A is a function $R : A^k \to \mathbb{Q}_\infty$. A *valued σ-structure* $\mathbf{A}$ consists of a finite universe A, together with a valued relation $R^{\mathbf{A}}$ of arity $\mathrm{ar}(R)$ on A for each $R \in \sigma$. Valued structures are sometimes referred to as *general-valued* in the literature [KTŽ15, TŽ17] to emphasize that relations in $\mathbf{A}$ may take non-finite values. A σ-structure $\mathbf{A}$ is said to be *non-negative finite-valued* if for every $R \in \sigma$, the range of $R^{\mathbf{A}}$ is contained in $\mathbb{Q}_{\geq 0}$.

Let $\mathbf{X}$, $\mathbf{A}$ be valued σ-structures, where $\mathbf{X}$ is non-negative finite-valued. The *value* of a map $h : X \to A$ for $(\mathbf{X}, \mathbf{A})$, and the *optimum value* for $(\mathbf{X}, \mathbf{A})$ are given by

$$\mathrm{Val}(\mathbf{X}, \mathbf{A}, h) = \sum_{R \in \sigma} \sum_{\mathbf{x} \in X^{\mathrm{ar}(R)}} R^{\mathbf{X}}(\mathbf{x}) R^{\mathbf{A}}(h(\mathbf{x})),$$
$$\mathrm{Opt}(\mathbf{X}, \mathbf{A}) = \min_{h : X \to A} \mathrm{Val}(\mathbf{X}, \mathbf{A}, h).$$

For two valued σ-structures $\mathbf{A}$ and $\mathbf{B}$, the *Promise Valued CSP over* $(\mathbf{A}, \mathbf{B})$ [VŽ21, Kaz22], denoted $\mathrm{PVCSP}(\mathbf{A}, \mathbf{B})$, is defined as follows: given a pair $(\mathbf{X}, \tau)$, where $\mathbf{X}$ is a non-negative finite-valued σ-structure and $\tau \in \mathbb{Q}$ is a *threshold*, output Yes if $\mathrm{Opt}(\mathbf{X}, \mathbf{A}) \leq \tau$, and output No if $\mathrm{Opt}(\mathbf{X}, \mathbf{B}) > \tau$. We call $(\mathbf{A}, \mathbf{B})$ a *PVCSP template* if the sets of Yes and No instances are disjoint. We show in Proposition 8.4 that this least restrictive meaningful requirement on a PVCSP template coincides with the choice taken in [VŽ21].

Notice that the values of R have a different intended meaning in the template valued structures $\mathbf{A}$, $\mathbf{B}$ and in the input valued structure $\mathbf{X}$. For the template, $R^{\mathbf{A}}(\mathbf{a})$ and $R^{\mathbf{B}}(\mathbf{b})$ should be understood as the *cost* of $\mathbf{a}$ and $\mathbf{b}$: we wish an assignment h to map tuples of variables to tuples of domain elements that are as cheap as possible (and, in fact, $R^{\mathbf{A}}$ or $R^{\mathbf{B}}$ is often referred to as a *cost function*). On the other hand, $R^{\mathbf{X}}(\mathbf{x})$ is the *weight* of the tuple of variables $\mathbf{x}$: we need to be more concerned about heavy tuples,

while we may ignore the tuples of zero weight (recall that $0 \cdot \infty = 0$). As an example, observe that the PCSP over a pair of structures $(\mathbf{A}', \mathbf{B}')$ is essentially the same problem as the PVCSP over the pair of $\{0, \infty\}$-valued structures $(\mathbf{A}, \mathbf{B})$, where tuples in the latter template are given zero cost iff they belong to the corresponding relations in the former template; while to an instance $\mathbf{X}'$ of the PCSP corresponds a non-negative finite-valued structure $\mathbf{X}$ where the cost of a tuple is zero iff the tuple does *not* belong to the corresponding relation in $\mathbf{X}'$ (and costs of the remaining tuples are arbitrary positive rationals), together with any threshold $\tau \in \mathbb{Q}_{\geq 0}$.

For a PVCSP input valued σ-structure $\mathbf{X}$ we define the set of *constraints* $\mathcal{C}_{\mathbf{X}}$ as the set of formal expressions of the form $R(\mathbf{x})$ where $R \in \sigma$, $\mathbf{x} \in X^{\mathrm{ar}(R)}$, and $R^{\mathbf{X}}(\mathbf{x}) > 0$; the value $R^{\mathbf{X}}(\mathbf{x})$ is the *weight* of the constraint. This almost translates the presented definition of (P)VCSP to the version introduced in Section 2.2.2: weights of constraints can be emulated by repeating constraints in (2.1). However, this encoding choice can cause an exponential blow up of the instance size. Nevertheless, this difference between the two formalisms is inessential for our purposes.

We say that a valued relation $R^{\mathbf{X}}$ has no repetitions if $R^{\mathbf{X}}(\mathbf{x}) = 0$ whenever $\mathbf{x}$ has a repetition. Similarly, we say that an input valued structure $\mathbf{X}$ has no repetitions if none of its valued relations has a repetition.

Example 8.1 (Constant factor approximation).

As mentioned in the introduction, the PVCSP framework can be used to model a decision version of constant factor approximation problems for MaxCSP. More concretely, suppose that we want to find a c-approximation for $\mathrm{CSP}(\mathbf{A})$ for some (non-valued) σ-structure $\mathbf{A}$ and some $c < 1$. One can model this problem as $\mathrm{PVCSP}(\mathbf{A}', \mathbf{B}')$ where $A' = B' = A$ and for all $R \in \sigma$ and $\mathbf{a} \in A^{\mathrm{ar}(R)}$, $R^{\mathbf{A}'}(\mathbf{a}) = -1$ if $\mathbf{a} \in R^{\mathbf{A}}$ and $R^{\mathbf{A}'}(\mathbf{a}) = 0$ otherwise; and $R^{\mathbf{B}'}(\mathbf{a}) = \frac{1}{c} R^{\mathbf{A}'}(\mathbf{a})$. Given an instance $\mathbf{X}$ of $\mathrm{CSP}(\mathbf{A})$ and a parameter $0 < \beta \leq 1$, we turn it into an instance $(\mathbf{X}', -\beta m)$ of $\mathrm{PVCSP}(\mathbf{A}', \mathbf{B}')$ in a natural way, where $\mathbf{X}'$ is a 0-1 valued structure and m is the number of constraints in $\mathbf{X}'$. Then, $\mathrm{Opt}(\mathbf{X}', \mathbf{A}') \leq -\beta m$ if a β-fraction of all constraints of $\mathbf{X}$ can be satisfied in $\mathbf{A}$, and $\mathrm{Opt}(\mathbf{X}', \mathbf{B}') > -\beta m$ if not even a $c\beta$-fraction of the constraints of $\mathbf{X}$ can be satisfied in $\mathbf{A}$.

Iterated degree and distributed model. As in the crisp setting, we use the factor graph representation of valued structures to define the iterated degree as well as the distributed model. This is defined analogously to the crisp setting, except that each edge $\{a, R(\mathbf{a})\}$ in the factor graph of $\mathbf{A}$ is labelled by a triple (S, R, q) where $q = R^{\mathbf{A}}(\mathbf{a}) > 0$ is the value of the constraint $R(\mathbf{a}) \in \mathcal{C}_{\mathbf{A}}$ and, as above, $S = \{i \in [\mathrm{ar}(R)] \mid a_i = a\}$. The notions of connectedness and iterated degree, the equivalence relation $\equiv_1$, and the distributed computational model are then defined using the factor graph for valued structures in the same way as for crisp structures.

Analogously to the case of distributed CSPs, we say that a distributed algorithm solves an instance $(\mathbf{X}, \tau)$ of $\mathrm{PVCSP}(\mathbf{A}, \mathbf{B})$ if the algorithm terminates and the terminating state of every process is Yes if $(\mathbf{X}, \tau)$ is a Yes instance of $\mathrm{PVCSP}(\mathbf{A}, \mathbf{B})$, and No if $(\mathbf{X}, \tau)$ is a No instance of $\mathrm{PVCSP}(\mathbf{A}, \mathbf{B})$. We say that a distributed algorithm solves $\mathrm{PVCSP}(\mathbf{A}, \mathbf{B})$ if it solves every connected instance of $\mathrm{PVCSP}(\mathbf{A}, \mathbf{B})$.

Linear programming relaxations. Given two valued σ-structures $\mathbf{X}$ and $\mathbf{A}$ where $\mathbf{X}$ is non-negative finite-valued, the systems of inequalities $\mathrm{BLP}(\mathbf{X}, \mathbf{A})$ and $\mathrm{SA}^1(\mathbf{X}, \mathbf{A})$ are given by adapting equations (BLP1), (BLP2), (BLP3) and (in the case of $\mathrm{SA}^1(\mathbf{X}, \mathbf{A})$) (↻) to the valued case, and importantly, adding an objective function. Concretely, $\mathrm{BLP}(\mathbf{X}, \mathbf{A})$ for valued $\mathbf{X}$ and $\mathbf{A}$ is the following linear program.

$$\mathrm{Opt}^{\mathrm{BLP}}(\mathbf{X}, \mathbf{A}) := \min \sum_{R(\mathbf{x}) \in \mathcal{C}_{\mathbf{X}}} \sum_{\mathbf{a} \in A^{\mathrm{ar}(R)}} p_{R(\mathbf{x})}(\mathbf{a}) R^{\mathbf{X}}(\mathbf{x}) R^{\mathbf{A}}(\mathbf{a}) \quad (\star)$$

subject to:

$$\sum_{a \in A} p_x(a) = 1 \qquad x \in X \qquad \text{(vBLP1)}$$

$$p_x(a) = \sum_{\mathbf{a} \in A^{\mathrm{ar}(R)},\ a_i = a} p_{R(\mathbf{x})}(\mathbf{a}) \qquad a \in A, R(\mathbf{x}) \in \mathcal{C}_{\mathbf{X}}, \ i \in [\mathrm{ar}(R)] \text{ such that } x_i = x \qquad \text{(vBLP2)}$$

$$p_{R(\mathbf{x})}(\mathbf{a}) = 0 \qquad R(\mathbf{x}) \in \mathcal{C}_{\mathbf{X}},\ \mathbf{a} \in A^{\mathrm{ar}(R)} \text{ such that } R^{\mathbf{A}}(\mathbf{a}) = \infty \qquad \text{(vBLP3)}$$

As for the linear program $\mathrm{SA}^1(\mathbf{X}, \mathbf{A})$, the objective function, denoted $\mathrm{Opt}^{\mathrm{SA}^1}(\mathbf{X}, \mathbf{A})$, is given by the same objective function as in $\mathrm{BLP}(\mathbf{X}, \mathbf{A})$. The variables are subject to all the constraints in $\mathrm{BLP}(\mathbf{X}, \mathbf{A})$, but in addition, they are also subject to the following constraint which (as in the non-valued case) handles the repetitions in the constraints of $\mathbf{X}$.

$$p_{R(\mathbf{x})}(\mathbf{a}) = 0 \qquad R(\mathbf{x}) \in \mathcal{C}_{\mathbf{X}},\ \mathbf{a} \in A^{\mathrm{ar}(R)} \colon \exists i, j \in [\mathrm{ar}(R)] \text{ such that } x_i = x_j \text{ and } a_i \neq a_j \qquad (\mathrm{v}\circlearrowleft)$$

Notice that in general $\mathrm{Opt}^{\mathrm{BLP}}(\mathbf{X}, \mathbf{A}) \leq \mathrm{Opt}^{\mathrm{SA}^1}(\mathbf{X}, \mathbf{A})$. Additionally, in the particular case where $\mathbf{X}$ has no repetitions, BLP and SA^1 are the same linear program and so $\mathrm{Opt}^{\mathrm{BLP}}(\mathbf{X}, \mathbf{A}) = \mathrm{Opt}^{\mathrm{SA}^1}(\mathbf{X}, \mathbf{A})$.

Moreover, for $\mathrm{L} \in \{\mathrm{BLP}, \mathrm{SA}^1\}$, if there exists a rational solution to $\mathrm{L}(\mathbf{X}, \mathbf{A})$ then $\mathrm{Opt}^{\mathrm{L}}(\mathbf{X}, \mathbf{A}) < \infty$ since $R^{\mathbf{A}}(\mathbf{a}) = \infty$ implies $p_{R(\mathbf{x})} = 0$ and $0 \cdot \infty = 0$ (formally, one should skip these summands in ($\star$)). If L is infeasible, then we set $\mathrm{Opt}^{\mathrm{L}}(\mathbf{X}, \mathbf{A}) = \infty$.

The inner sum in ($\star$) is equal to the expected "cost" of the constraint $R(\mathbf{x})$ with weight $R^{\mathbf{X}}(\mathbf{x})$ when $\mathbf{x}$ is evaluated according to this distribution. From this observation it is apparent that $\mathrm{Opt}^{\mathrm{L}}(\mathbf{X}, \mathbf{A}) \leq \mathrm{Opt}(\mathbf{X}, \mathbf{A})$. We say that L *decides* $\mathrm{PVCSP}(\mathbf{A}, \mathbf{B})$ if, for every input structure $\mathbf{X}$, we have $\mathrm{Opt}(\mathbf{X}, \mathbf{B}) \leq \mathrm{Opt}^{\mathrm{L}}(\mathbf{X}, \mathbf{A})$. Note that in this case the algorithm for $\mathrm{PVCSP}(\mathbf{A}, \mathbf{B})$ that answers Yes iff $\mathrm{Opt}^{\mathrm{L}}(\mathbf{X}, \mathbf{A}) \leq \tau$ (where τ is the input threshold) is correct, so the definition makes sense.

8.3 Fractional operations

In this section we introduce the notions of fractional operation, which played a major role in the development of the algebraic approach to Valued CSPs [CCC+13, KO15], and discuss some of their basic properties.

An n-ary *fractional polymorphism* [VŽ21] of a pair of valued σ-structures $(\mathbf{A}, \mathbf{B})$ is a probability distribution ω on the set $B^{A^n} := \{f : A^n \to B\}$ such that for every $R \in \sigma$ and every list of n tuples $\mathbf{a}_1, \ldots, \mathbf{a}_n \in A^{\mathrm{ar}(R)}$ we have that

$$\sum_{f \in B^{A^n}} \omega(f) R^{\mathbf{B}}(f(\mathbf{a}_1, \ldots, \mathbf{a}_n)) \leq \frac{1}{n} \sum_{i=1}^{n} R^{\mathbf{A}}(\mathbf{a}_i)$$

where f is applied to $\mathbf{a}_1, \dots, \mathbf{a}_n \in A^{\mathrm{ar}(R)}$ component-wise.[2] The *support* of ω is the set of functions $f : A^n \to B$ such that $\omega(f) > 0$. We say that ω is *symmetric* if every operation in its support if symmetric.

The equivalence of solvability by BLP to invariance under symmetric operations lifts to the Promise Valued setting.

Theorem 8.2 ([VŽ21])**.** *Let* $(\mathbf{A}, \mathbf{B})$ *be a promise valued template of signature* σ*. Then the following are equivalent.*

(iv) BLP *decides* $\mathrm{PVCSP}(\mathbf{A}, \mathbf{B})$*;*

(v) $(\mathbf{A}, \mathbf{B})$ *has symmetric fractional polymorphisms of every arity.*

Example 8.3 (BLP-decidable (P)(V)CSPs).

A CSP that can be decided by BLP is e.g. the Horn-3-Sat, where the template has domain $\{\mathrm{true}, \mathrm{false}\}$ and two ternary relations defined by $\neg x \vee \neg y \vee \neg z$ and $\neg x \vee \neg y \vee z$ (as well as both constants). A simple PCSP template decidable by BLP is e.g. $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ where A, B are ordered sets, $|A| \leq |B|$, and $\mathbf{A}$ and $\mathbf{B}$ both have a single binary relation defined by $x < y$. A well-known class of templates with BLP-decidable VCSPs are those that contain only submodular valued relations (see [Kol13, KŽ17a]). Finally, the 2-approximation of the Vertex Cover problem [KŽ17a] is a PVCSP decidable by BLP. In all the mentioned examples, it is not hard to find symmetric (fractional) polymorphisms of every arity.

We will defer the proof of Theorem 8.2 to the end of this section as it makes use of a concept that we introduce next.

A *fractional homomorphism* [TŽ12, VŽ21] from $\mathbf{A}$ to $\mathbf{B}$ is a unary fractional polymorphism of $(\mathbf{A}, \mathbf{B})$, or equivalently, a probability distribution μ over B^A such that for every $R \in \sigma$ and every $\mathbf{a} \in A^{\mathrm{ar}(R)}$ we have that

$$\sum_{f \in B^A} \mu(f) R^{\mathbf{B}}(f(\mathbf{a})) \leq R^{\mathbf{A}}(\mathbf{a}). \tag{8.1}$$

[2] We use a simpler concept than fractional polymorphism as defined in [VŽ21], which will be sufficient for our purposes.

If there exists a fractional homomorphism from $\mathbf{A}$ to $\mathbf{B}$, we say that $\mathbf{A}$ is fractionally homomorphic to $\mathbf{B}$ and we write $\mathbf{A} \to_f \mathbf{B}$.

The following is a characterization of PVCSP templates in terms of fractional homomorphisms. This result will also be useful in the proof of Theorem 8.2.

Proposition 8.4. *For any two valued σ-structures $\mathbf{A}$ and $\mathbf{B}$, the following are equivalent.*

1. *There exists a fractional homomorphism from $\mathbf{A}$ to $\mathbf{B}$.*
2. *For all non-negative finite-valued σ-structures $\mathbf{X}$, $\operatorname{Opt}(\mathbf{X},\mathbf{B}) \le \operatorname{Opt}(\mathbf{X},\mathbf{A})$.*

Proof. (1) $\Rightarrow$ (2). Let μ be a fractional homomorphism from $\mathbf{A}$ to $\mathbf{B}$, let $g : X \to A$ be such that $\operatorname{Opt}(\mathbf{X}, \mathbf{A}) = \operatorname{Val}(\mathbf{X}, \mathbf{A}, g)$, and let $f \in B^A$ be some map that minimizes $\operatorname{Val}(\mathbf{X}, \mathbf{B}, f \circ g)$. Then

$$\begin{aligned}\operatorname{Opt}(\mathbf{X}, \mathbf{B}) &\le \operatorname{Val}(\mathbf{X}, \mathbf{B}, f \circ g) \le \sum_{f' \in B^A} \mu(f') \operatorname{Val}(\mathbf{X}, \mathbf{B}, f' \circ g) \\ &= \sum_{R \in \sigma} \sum_{\mathbf{x} \in X^{\operatorname{ar}(R)}} R^{\mathbf{X}}(\mathbf{x}) \sum_{f' \in B^A} \mu(f') R^{\mathbf{B}}(f' \circ g(\mathbf{x})) \\ &\le \sum_{R \in \sigma} \sum_{\mathbf{x} \in X^{\operatorname{ar}(R)}} R^{\mathbf{X}}(\mathbf{x}) R^{\mathbf{A}}(g(\mathbf{x})) = \operatorname{Val}(\mathbf{X}, \mathbf{A}, g) = \operatorname{Opt}(\mathbf{X}, \mathbf{A}).\end{aligned}$$

(2) $\Rightarrow$ (1). The idea for this proof is to assume that there is no fractional homomorphism from $\mathbf{A}$ to $\mathbf{B}$, formulate this fact as infeasibility of a system of linear inequalities, and then use a version of Farkas' Lemma [Sch86] to find a valued structure $\mathbf{X}$ with $\operatorname{Opt}(\mathbf{X}, \mathbf{B}) > \operatorname{Opt}(\mathbf{X}, \mathbf{A})$.

The existence of a fractional homomorphism from $\mathbf{A}$ to $\mathbf{B}$ can be reformulated as the following system of linear inequalities, where there is a rational-valued variable μ_f for every $f \in B^A$.

$$\begin{aligned}&\text{variables:} && \mu_f \text{ for all } f \in B^A \\ &\text{constraints:} && \sum_{f \in B^A} \mu_f R^{\mathbf{B}}(f(\mathbf{a})) \le R^{\mathbf{A}}(\mathbf{a}) && \text{for all } R \in \sigma \text{ and } \mathbf{a} \in A^{\operatorname{ar}(R)} && (8.2a) \\ &&& \sum_{f \in B^A} \mu_f \ge 1 && && (8.2b) \\ &&& \mu_f \ge 0 \quad \text{for all } f \in B^A. && && (8.2c)\end{aligned}$$

If there is no fractional homomorphism from $\mathbf{A}$ to $\mathbf{B}$, the system (8.2) is infeasible.

We now deal with infinite coefficients. Define $B^A_{<\infty} = \{f \in B^A : \forall R \in \sigma, \forall \mathbf{a} \in A^{\mathrm{ar}(R)},\ R^{\mathbf{A}}(\mathbf{a}) < \infty \text{ implies } R^{\mathbf{B}}(f(\mathbf{a})) < \infty\}$. Now consider the new linear system obtained from (8.2) by first removing all the inequalities in (8.2a) where $R^{\mathbf{A}}(\mathbf{a}) = \infty$ (since these inequalities are always satisfied), and second, by removing from (8.2) the variable μ_f for all $f \in B^A \setminus B^A_{<\infty}$ and changing (8.2a) so that the sums run over $B^A_{<\infty}$ only (since we need to have $\mu_f = 0$ for $f \in B^A \setminus B^A_{<\infty}$ in any feasible solution). Clearly, the system of linear inequalities resulting from this procedure remains infeasible and does not contain infinite coefficients.

This system of linear inequalities can be rewritten in matrix form as $M\mathbf{f} \leq \mathbf{a}$ subject to $\mathbf{f} \geq 0$, where $\mathbf{f} \in \mathbb{Q}_{\geq}^{B^A_{<\infty}}$ is the vector of unknowns, and M is a real-valued matrix. By Farkas' Lemma, if the program (8.2) is not feasible, then the system of inequalities $M^T\mathbf{y} \geq 0$ subject to $\mathbf{a}^T\mathbf{y} < 0$ and $\mathbf{y} \geq 0$ is feasible. Explicitly, the latter system is the following.

$$\begin{aligned}
&\text{variables:} && y, x_{R,\mathbf{a}} \text{ for every } R \in \sigma \text{ and } \mathbf{a} \in A^{\mathrm{ar}(R)} \text{ with } R^{\mathbf{A}}(\mathbf{a}) < \infty \\
&\text{constraints:} && \sum_{R\in\sigma} \sum_{\substack{\mathbf{a}\in A^{\mathrm{ar}(R)} \\ R^{\mathbf{A}}(\mathbf{a})<\infty}} x_{R,\mathbf{a}} R^{\mathbf{B}}(f(\mathbf{a})) \geq y \quad \text{for all } f \in B^A_{<\infty} && \text{(8.3a)} \\
& && \sum_{R\in\sigma} \sum_{\substack{\mathbf{a}\in A^{\mathrm{ar}(R)} \\ R^{\mathbf{A}}(\mathbf{a})<\infty}} x_{R,\mathbf{a}} R^{\mathbf{A}}(\mathbf{a}) < y && \text{(8.3b)} \\
& && x_{R,\mathbf{a}} \geq 0 \quad \text{for all } R \in \sigma, \mathbf{a} \in A^{\mathrm{ar}(R)} && \text{(8.3c)} \\
& && y \geq 0. && \text{(8.3d)}
\end{aligned}$$

Eliminating y, and adding trivially satisfied constraints to (8.3a) for all $f \in B^A \setminus B^A_{<\infty}$, we get that the following system is feasible.

$$\begin{aligned}
&\text{variables:} && x_{R,\mathbf{a}} \text{ for every } R \in \sigma \text{ and } \mathbf{a} \in A^{\mathrm{ar}(R)} \text{ with } R^{\mathbf{A}}(\mathbf{a}) < \infty \\
&\text{constraints:} && \sum_{R\in\sigma} \sum_{\substack{\mathbf{a}\in A^{\mathrm{ar}(R)} \\ R^{\mathbf{A}}(\mathbf{a})<\infty}} x_{R,\mathbf{a}} R^{\mathbf{B}}(f(\mathbf{a})) > \sum_{R\in\sigma} \sum_{\substack{\mathbf{a}\in A^{\mathrm{ar}(R)} \\ R^{\mathbf{A}}(\mathbf{a})<\infty}} x_{R,\mathbf{a}} R^{\mathbf{A}}(\mathbf{a}) \\
& && \qquad\qquad \text{for all } f \in B^A && \text{(8.4a)} \\
& && x_{R,\mathbf{a}} \geq 0 \quad \text{for all } R \in \sigma,\ \mathbf{a} \in A^{\mathrm{ar}(R)}. && \text{(8.4b)}
\end{aligned}$$

Let $x_{R,\mathbf{a}}$ for $R \in \sigma$, $\mathbf{a} \in A^r$ be a feasible solution to (8.4), and consider the structure $\mathbf{X}$ with domain $X = A$ and relations given by $R^{\mathbf{X}}(\mathbf{a}) = x_{R,\mathbf{a}}$ for $\mathbf{a} \in A^{\mathrm{ar}(R)}$ with $R^{\mathbf{A}}(\mathbf{a}) < \infty$ and $R^{\mathbf{X}}(\mathbf{a}) = 0$ whenever $R^{\mathbf{A}}(\mathbf{a}) = \infty$. Notice that $\mathbf{X}$ is non-negative finite-valued, that the right-hand side in (8.4a) is equal to $\mathrm{Val}(\mathbf{X}, \mathbf{A}, \mathrm{id})$ (where id denotes the identity function), and that the left-hand side is equal to $\mathrm{Val}(\mathbf{X}, \mathbf{B}, f)$. Therefore $\mathrm{Opt}(\mathbf{X}, \mathbf{B}) > \mathrm{Opt}(\mathbf{X}, \mathbf{A})$, as required. □

The valued version of the decomposition theorem for SA^1 uses a concept that is "dual" to fractional homomorphism, as suggested by Proposition 8.5 below. While only the easier implication (1) ⇒ (2) is needed for the proof of Theorem 8.6, we sketch the proof of the other implication as well.

We define a *dual fractional homomorphism* from $\mathbf{X}$ to $\mathbf{Y}$ ($\mathbf{X} \to_{df} \mathbf{Y}$) to be a probability distribution η over Y^X such that for every $R \in \sigma$ and every $\mathbf{y} \in Y^{\mathrm{ar}(R)}$ we have that

$$R^{\mathbf{Y}}(\mathbf{y}) \geq \sum_{f \in Y^X} \eta(f) \sum_{\substack{\mathbf{x} \in X^{\mathrm{ar}(R)} \\ \mathbf{y} = f(\mathbf{x})}} R^{\mathbf{X}}(\mathbf{x}). \tag{8.5}$$

Proposition 8.5 ([CRŽ22])**.** *For any two non-negative finite-valued σ-structures $\mathbf{X}$ and $\mathbf{Y}$, the following are equivalent.*

1. *There exists a dual fractional homomorphism from $\mathbf{X}$ to $\mathbf{Y}$.*
2. *For all valued σ-structures $\mathbf{A}$, $\mathrm{Opt}(\mathbf{X}, \mathbf{A}) \leq \mathrm{Opt}(\mathbf{Y}, \mathbf{A})$.*

Proof. (1) ⇒ (2). Let η be a dual fractional homomorphism from $\mathbf{X}$ to $\mathbf{Y}$, and $g : Y \to A$ be such that $\mathrm{Opt}(\mathbf{Y}, \mathbf{A}) = \mathrm{Val}(\mathbf{Y}, \mathbf{A}, g)$. Then

$$\begin{aligned}
\mathrm{Opt}(\mathbf{Y}, \mathbf{A}) &= \sum_{R \in \sigma} \sum_{\mathbf{y} \in Y^{\mathrm{ar}(R)}} R^{\mathbf{Y}}(\mathbf{y}) R^{\mathbf{A}}(g(\mathbf{y})) \\
&\geq \sum_{R \in \sigma} \sum_{\mathbf{y} \in Y^{\mathrm{ar}(R)}} \sum_{f \in Y^X} \eta(f) \sum_{\substack{\mathbf{x} \in X^{\mathrm{ar}(R)} \\ \mathbf{y} = f(\mathbf{x})}} R^{\mathbf{X}}(\mathbf{x}) R^{\mathbf{A}}(g \circ f(\mathbf{x})) \\
&= \sum_{f \in Y^X} \eta(f) \, \mathrm{Val}(\mathbf{X}, \mathbf{A}, g \circ f),
\end{aligned}$$

which implies that there exists some function $f' : X \to Y$ such that $\mathrm{Val}(\mathbf{X}, \mathbf{A}, g \circ f') \leq \mathrm{Opt}(\mathbf{Y}, \mathbf{A})$, hence $\mathrm{Opt}(\mathbf{X}, \mathbf{A}) \leq \mathrm{Opt}(\mathbf{Y}, \mathbf{A})$ as required. Notice that this holds regardless of whether $\mathbf{A}$ is finite-valued or general-valued.

(2) $\Rightarrow$ (1). The contrapositive is proved in a similar way to Proposition 8.4. The proof is in fact somewhat simpler since we do not need to deal with infinities.

The existence of a dual fractional homomorphism from $\mathbf{X}$ to $\mathbf{Y}$ can be reformulated as the following linear program:

$$\begin{aligned}
&\text{variables:} && \eta_f \text{ for all } f \in Y^X \\
&\text{constraints:} && \sum_{f \in Y^X} \eta_f \sum_{\substack{\mathbf{x} \in X^r \\ \mathbf{y} = f(\mathbf{x})}} R^{\mathbf{X}}(\mathbf{x}) \leq R^{\mathbf{Y}}(\mathbf{y}) \quad \text{for all } R \in \sigma \text{ and } \mathbf{y} \in Y^{\mathrm{ar}(R)} && (8.6a) \\
& && \sum_{f \in Y^X} \eta_f \geq 1 && (8.6b) \\
& && \eta_f \geq 0 \quad \text{for all } f \in Y^X. && (8.6c)
\end{aligned}$$

Assume that there is no dual fractional homomorphism from $\mathbf{X}$ to $\mathbf{Y}$. Then, the (8.6) is infeasible.

Notice that $\mathbf{X}$ and $\mathbf{Y}$ are assumed to be finite-valued, so all the coefficients in this system of linear inequalities are finite-valued too. Then, (8.6) can be rewritten in matrix form as $M\mathbf{f} \leq \mathbf{y}$ subject to $\mathbf{f} \geq 0$, where $\mathbf{f} \in \mathbb{Q}_{\geq}^{Y^X}$ is the vector of unknowns, and M is a real-valued matrix. By Farkas' Lemma, if (8.6) is not feasible, then the linear program $M^T\mathbf{z} \geq 0$ subject to $\mathbf{y}^T\mathbf{z} < 0$ and $\mathbf{z} \geq 0$ is feasible. Explicitly, this program is the following.

$$\begin{aligned}
&\text{variables:} && t, z_{R,\mathbf{u}} \text{ for every } R \in \sigma \text{ and } \mathbf{y} \in Y^{\mathrm{ar}(R)} \\
&\text{constraints:} && \sum_{R \in \sigma} \sum_{\mathbf{y} \in Y^{\mathrm{ar}(R)}} z_{R,\mathbf{y}} \sum_{\substack{\mathbf{x} \in X^{\mathrm{ar}(R)} \\ \mathbf{y} = f(\mathbf{x})}} R^{\mathbf{X}}(\mathbf{x}) \geq t \quad \text{for all } f \in Y^X && (8.7a) \\
& && \sum_{R \in \sigma} \sum_{\mathbf{y} \in Y^{\mathrm{ar}(R)}} z_{R,\mathbf{y}} R^{\mathbf{Y}}(\mathbf{y}) < t && (8.7b) \\
& && z_{R,\mathbf{y}} \geq 0 \quad \text{for all } R \in \sigma, \mathbf{y} \in Y^{\mathrm{ar}(R)} && (8.7c) \\
& && t \geq 0. && (8.7d)
\end{aligned}$$

Eliminating t and rearranging we get that the following program is feasible.

$$\begin{aligned}
\text{variables:} \quad & z_{R,\mathbf{y}} \text{ for every } R \in \sigma \text{ and } \mathbf{y} \in Y^{\mathrm{ar}(R)} \\
\text{constraints:} \quad & \sum_{R\in\sigma} \sum_{\mathbf{y}\in Y^{\mathrm{ar}(R)}} z_{R,\mathbf{y}} \sum_{\substack{\mathbf{x}\in X^{\mathrm{ar}(R)} \\ \mathbf{y}=f(\mathbf{x})}} R^{\mathbf{X}}(\mathbf{x}) > \sum_{R\in\sigma} \sum_{\mathbf{y}\in Y^{\mathrm{ar}(R)}} z_{R,\mathbf{y}} R^{\mathbf{Y}}(\mathbf{y}) \\
& \qquad\qquad \text{for all } f \in Y^X \qquad (8.8a) \\
& z_{R,\mathbf{y}} \geq 0 \text{ for } R \in \sigma, \mathbf{y} \in Y^{\mathrm{ar}(R)}. \qquad (8.8b)
\end{aligned}$$

Let $z_{R,\mathbf{y}}$ for $R \in \sigma$, $\mathbf{y} \in Y^{\mathrm{ar}(R)}$ be a feasible solution to (8.8), and consider the valued structure $\mathbf{A}$ with domain $A = Y$ and valued relations given by $R^{\mathbf{A}}(\mathbf{y}) = z_{R,\mathbf{y}}$ for each $\mathbf{y} \in Y^{\mathrm{ar}(R)}$. For all $f \in Y^X$, we have

$$\begin{aligned}
\mathrm{Val}(\mathbf{Y}, \mathbf{A}, \mathrm{id}) &= \sum_{R\in\sigma} \sum_{\mathbf{y}\in Y^{\mathrm{ar}(R)}} R^{\mathbf{A}}(\mathbf{y}) R^{\mathbf{Y}}(\mathbf{y}) \\
&< \sum_{R\in\sigma} \sum_{\mathbf{y}\in Y^{\mathrm{ar}(R)}} R^{\mathbf{A}}(\mathbf{y}) \sum_{\substack{\mathbf{x}\in X^{\mathrm{ar}(R)} \\ \mathbf{y}=f(\mathbf{x})}} R^{\mathbf{X}}(\mathbf{x}) = \mathrm{Val}(\mathbf{X}, \mathbf{A}, f),
\end{aligned}$$

therefore $\mathrm{Opt}(\mathbf{Y}, \mathbf{A}) < \mathrm{Opt}(\mathbf{X}, \mathbf{A})$, which is the negation of (2). □

Sketch of proof of Theorem 8.2. The proof is similar in spirit to the equivalence of (2) and (3) in Theorem 5.6, except that for each $m \geq 1$, the valued relation of the valued structure $\mathrm{LP}^m(\mathbf{A})$ [TŽ12, VŽ21] are defined by:

$$R^{\mathrm{LP}^m(\mathbf{A})}(s_1, \ldots, s_r) := \frac{1}{m} \min_{\substack{\mathbf{t}_1,\ldots,\mathbf{t}_r \in A^m \\ \{\{\mathbf{t}_i\}\}=s_i}} \sum_{i=1}^{m} R^{\mathbf{A}}(\mathbf{t}_1[i], \ldots, \mathbf{t}_r[i]).$$

The following properties are the equivalents of 1 and 2 in the valued setting:

1. $\mathrm{Opt}^{\mathrm{BLP}}(\mathbf{X}, \mathbf{A}) = \min_{m\geq 1} \mathrm{Opt}(\mathbf{X}, \mathrm{LP}^m(\mathbf{A}))$ for all non-negative finite-valued $\mathbf{X}$.
2. For all $m \geq 1$, $\mathrm{LP}^m(\mathbf{A}) \to_f \mathbf{B}$ if and only if $(\mathbf{A}, \mathbf{B})$ has an m-ary symmetric fractional polymorphism.

The proof can be now finished using Proposition 8.4. For (iv) ⇒ (v) suppose that $(\mathbf{A}, \mathbf{B})$ does not have a symmetric polymorphism of some arity m. Then, there is no fractional homomorphism from $\mathrm{LP}^m(\mathbf{A})$ to $\mathbf{B}$. It follows from Proposition 8.4 that there exists some non-negative finite-valued structure $\mathbf{X}$ such that $\mathrm{Opt}(\mathbf{X}, \mathbf{B}) > \mathrm{Opt}(\mathbf{X}, \mathrm{LP}^m(\mathbf{A})) \geq \mathrm{Opt}^{\mathrm{BLP}}(\mathbf{X}, \mathbf{A})$. Hence, BLP does not decide $\mathrm{PVCSP}(\mathbf{A}, \mathbf{B})$. On the other hand, for (v) ⇒ (iv), assume that $(\mathbf{A}, \mathbf{B})$ has symmetric fractional polymorphisms of every arity. Let $\mathbf{X}$ be non-negative finite-valued and $m \geq 1$ be such that $\mathrm{Opt}(\mathbf{X}, \mathrm{LP}^m(\mathbf{A}))$ is minimal. We know that $\mathrm{LP}^m(\mathbf{A})$ is fractionally homomorphic to $\mathbf{B}$ and therefore by Proposition 8.4 $\mathrm{Opt}(\mathbf{X}, \mathbf{B}) \leq \mathrm{Opt}(\mathbf{X}, \mathrm{LP}^m(\mathbf{A})) = \mathrm{Opt}^{\mathrm{BLP}}(\mathbf{X}, \mathbf{A})$. Hence, BLP decides $\mathrm{PVCSP}(\mathbf{A}, \mathbf{B})$. □

8.4 The Decomposition Theorem for valued structures

Finally, we state the decomposition theorem for valued structures. This provides a connection between the combinatorial and the LP-based characterizations of the class of PVCSP templates that are the subject of the main result of this chapter.

Theorem 8.6. *Let $\mathbf{X}$, $\mathbf{A}$ be a pair of similar valued structures, where $\mathbf{X}$ is non-negative and finite-valued. Then there exist non-negative finite-valued structures $\mathbf{Y}_1, \mathbf{Y}_2$ such that*

1. $\mathbf{X} \to_{df} \mathbf{Y}_1$,
2. $\mathbf{Y}_1 \equiv_1 \mathbf{Y}_2$, *and*
3. $\mathrm{Opt}(\mathbf{Y}_2, \mathbf{A}) \leq \mathrm{Opt}^{\mathrm{SA}^1}(\mathbf{X}, \mathbf{A})$.

Proof. If $\mathrm{SA}^1(\mathbf{X}, \mathbf{A})$ is not feasible, then we can take $\mathbf{Y}_1 = \mathbf{Y}_2 = \mathbf{X}$, and the statement follows trivially, so from now on we shall assume that $\mathrm{SA}^1(\mathbf{X}, \mathbf{A})$ is feasible. In this case, the construction of the valued structures $\mathbf{Y}_1$ and $\mathbf{Y}_2$ is analogous to the construction of Theorem 7.1, except that one has to additionally take into consideration the weights of the constraints in $\mathbf{X}$. In particular, let $m > 0$ be the least common

denominator of a rational solution to $\mathrm{SA}^1(\mathbf{X}, \mathbf{A})$. For every constraint $R(\mathbf{x}) \in \mathcal{C}_{\mathbf{X}}$ of arity r, let $\rho_1, \dots, \rho_r$ be the permutations of $[m]$ obtained as in the proof of Theorem 7.1 so that they satisfy conditions (1) and (2) in said proof. Then, for each $k \in [m]$ we set $R^{\mathbf{Y}_1}((k, \mathbf{x}[1]), \dots, (k, \mathbf{x}[r])) = R^{\mathbf{Y}_2}((\rho_1(k), \mathbf{x}[1]), \dots, (\rho_r(k), \mathbf{x}[r])) = 1/m \cdot R^{\mathbf{X}}(\mathbf{x})$, and all the other tuples in $\mathbf{Y}_1$ and $\mathbf{Y}_2$ are given zero weight.

Then $\mathbf{X} \to_{df} \mathbf{Y}_1$ by the dual fractional homomorphism given by the uniform distribution over f_k, $k \in [m]$, where $f_k : X \to Y_1$ is defined by $f_k(x) = (k, x)$ for all $x \in X$.

On the other hand, let $h : Y_2 \to A$ be the same map that gives a homomorphism from $\mathbf{Y}_2$ to $\mathbf{A}$ in the crisp setting. For each $R(\mathbf{x}) \in \mathcal{C}_{\mathbf{X}}$ the weights of the tuples that correspond to $R(\mathbf{x})$ are selected so that their contribution to $\mathrm{Val}(\mathbf{Y}_2, \mathbf{A}, h)$ is equal to the inner sum in the objective function $(\star)$ of $\mathrm{SA}^1(\mathbf{X}, \mathbf{A})$; therefore, the total value of h for $(\mathbf{Y}_2, \mathbf{A})$ is equal to $\mathrm{Opt}^{\mathrm{SA}^1}(\mathbf{X}, \mathbf{A})$. It follows that $\mathrm{Opt}(\mathbf{Y}_2, \mathbf{A}) \leq \mathrm{Opt}^{\mathrm{SA}^1}(\mathbf{X}, \mathbf{A})$.

Moreover, the iterated degree of (k, x) in both $\mathbf{Y}_1$ and $\mathbf{Y}_2$ is obtained from the iterated degree of x by scaling down each label (S, R, q) to $(S, R, q/m)$. It follows that $\mathbf{Y}_1 \equiv_1 \mathbf{Y}_2$, and the proof is concluded. □

As in the crisp setting, the dual fractional homomorphism $\mathbf{X} \to_{df} \mathbf{Y}_1$, the equivalence $\mathbf{Y}_1 \equiv_1 \mathbf{Y}_2$, and the assignment $Y_2 \to A$ that witness $\mathrm{Opt}(\mathbf{Y}_2, \mathbf{A})$ from the proof of Theorem 8.6 can all be associated with rational matrices of the appropriate dimensions, the product of which is associated to a solution to $\mathrm{SA}^1(\mathbf{X}, \mathbf{A})$. Hence, again, Theorem 8.6 can be regarded as a decomposition theorem.

8.5 Weisfeiler-Leman invariant PVCSPs

We are ready to prove the main result. The appropriate generalization of $\equiv_1$-invariance (see Section 5.1) to the PVCSP setting is the following: for any promise valued template $(\mathbf{A}, \mathbf{B})$, $\mathrm{PVCSP}(\mathbf{A}, \mathbf{B})$ is closed under $\equiv_1$ iff for any two non-negative finite-valued structures $\mathbf{X}_1$ and $\mathbf{X}_2$ similar to $\mathbf{A}$ and $\mathbf{B}$ and any rational threshold τ, $\mathbf{X}_1 \equiv_1 \mathbf{X}_2$ implies that $\mathrm{Opt}(\mathbf{X}_2, \mathbf{B}) \leq \mathrm{Opt}(\mathbf{X}_1, \mathbf{A})$.

Theorem 8.7. *Let* $(\mathbf{A}, \mathbf{B})$ *be a promise valued template of signature* σ*. Then the following are equivalent.*

(i) *There is a distributed algorithm that solves* $\mathrm{PVCSP}(\mathbf{A}, \mathbf{B})$*. Moreover, in this case, there is a polynomial-time distributed algorithm that solves* $\mathrm{PVCSP}(\mathbf{A}, \mathbf{B})$*.*

(ii) $\mathrm{PVCSP}(\mathbf{A}, \mathbf{B})$ *is closed under* $\equiv_1$*-equivalence;*

(iii) SA^1 *decides* $\mathrm{PVCSP}(\mathbf{A}, \mathbf{B})$*.*

Proof. The proof of (i) ⇒ (ii) is similar to the corresponding implication in the proof of Theorem 7.2, but one has to use extra care when dealing with valued structures. As before, it follows from Corollary 6.5 that if (i) holds and $\mathbf{X} \equiv_1 \mathbf{Y}$ are connected, then a terminating distributed algorithm will report the same decision when run on input $(\mathbf{X}, \tau)$ or $(\mathbf{Y}, \tau)$, so by setting $\tau = \mathrm{Opt}(\mathbf{X}, \mathbf{A})$ we obtain that (ii) holds for all connected $\mathbf{X}$, $\mathbf{Y}$. We need to show that (ii) in its full generality follows from (ii) restricted to connected structures.

We now claim that for any two finite-valued σ-structures $\mathbf{X}$ and $\mathbf{Y}$, $\mathrm{Opt}(\mathbf{Y}, \mathbf{B})/|Y| \leq \mathrm{Opt}(\mathbf{X}, \mathbf{A})/|X|$ whenever $\mathbf{X}$ and $\mathbf{Y}$ are weakly congruent and connected. As before, the claim holds trivially when either X or Y is a singleton. For $|X|, |Y| \geq 2$, we present a valued construction similar in spirit to that of Theorem 7.2, but which accounts for valued relations.

For any positive integer k, we define a pair of connected finite-valued σ-structures $\mathbf{X}'^{(k)}$ and $\mathbf{X}^{(k)}$ (and similarly $\mathbf{Y}'^{(k)}$, $\mathbf{Y}^{(k)}$) as follows. Let $X = \{x_0, x_1, \ldots, x_{|X|-1}\}$ and let the universe of $\mathbf{X}'^{(k)}$ be $X^{(k)} = \{0, 1, \ldots, k-1\} \times X$. Let η be the probability distribution over the mappings $X \to X^{(k)}$ assigning probability $1/2k$ to each of the $2k$ mappings f_j, f'_j, $j \in \{0, 1, \ldots, k-1\}$, where $f_j(x_i) = (j, x_i)$ and $f'_j(x_i) = (i + j \pmod k, x_i)$ for each $x_i \in X$. We define the weights in $\mathbf{X}'^{(k)}$ in the unique way so that (8.5) holds for η with equality instead of inequality. Then η is a dual fractional homomorphism from $\mathbf{X}$ to $\mathbf{X}'^{(k)}$, and the probability distribution which assigns probability 1 to the projection onto X is a dual fractional homomorphism in the opposite direction. By Proposition 8.5, $\mathrm{Opt}(\mathbf{X}, \mathbf{C}) = \mathrm{Opt}(\mathbf{X}'^{(k)}, \mathbf{C})$ for any valued σ-structure $\mathbf{C}$. Finally, let $\mathbf{X}^{(k)}$ be the valued σ-structure obtained from $\mathbf{X}'^{(k)}$ by multiplying weights by

$2k$; clearly, $\mathrm{Opt}(\mathbf{X}^{(k)}, \mathbf{C}) = 2k\,\mathrm{Opt}(\mathbf{X}'^{(k)}, \mathbf{C})$ for any $\mathbf{C}$. As in Chapter 7, $\mathbf{X}^{(k)}$ is connected for all k and for k' large enough, the valued structures $\mathbf{Y}^{(k'|X|)}$ and $\mathbf{X}^{(k'|Y|)}$ have the same iterated degree. By item (ii) for connected valued structures, we get $\mathrm{Opt}(\mathbf{Y}^{(k'|X|)}, \mathbf{B}) \leq \mathrm{Opt}(\mathbf{X}^{(k'|Y|)}, \mathbf{A})$ and the claim follows using the equalities above and rearranging.

This observation allows us to finish the proof as follows. Let $\mathbf{X}$ and $\mathbf{Y}$ be valued structures such that $\mathbf{X} \equiv_1 \mathbf{Y}$ and let $n = |X| = |Y|$. From Remark 5.3 we get that there exists a sequence $(\mathbf{X}_1, \ldots, \mathbf{X}_n)$ (resp., $(\mathbf{Y}_1, \ldots, \mathbf{Y}_n)$) that contains each connected component $\mathbf{X}'$ of $\mathbf{X}$ (resp., $\mathbf{Y}'$ of $\mathbf{Y}$) exactly $|X'|$ times (resp., $|Y'|$ times), and $\mathbf{X}_i$ and $\mathbf{Y}_i$ are weakly congruent for every $i \in [n]$. From the claim above, we get that $\mathrm{Opt}(\mathbf{Y}_i, \mathbf{B})/|Y_i| \leq \mathrm{Opt}(\mathbf{X}_i, \mathbf{A})/|X_i|$ for every $i \in [n]$. Summing up these inequalities and observing that $\mathrm{Opt}(\mathbf{X}, \mathbf{A})$ is equal to the sum of $\mathrm{Opt}(\mathbf{X}', \mathbf{A})$ over all connected components $\mathbf{X}'$ of $\mathbf{X}$ (and similarly for $\mathbf{Y}$), item (ii) follows.

(ii) $\Rightarrow$ (iii). We need to show that for every non-negative finite-valued σ-structure $\mathbf{X}$, $\mathrm{Opt}(\mathbf{X}, \mathbf{B}) \leq \mathrm{Opt}^{\mathrm{SA}^1}(\mathbf{X}, \mathbf{A})$. Let $\mathbf{Y}_1, \mathbf{Y}_2$ be the structures obtained from Theorem 8.6, i.e., $\mathbf{X} \to_{df} \mathbf{Y}_1$, $\mathbf{Y}_1 \equiv_1 \mathbf{Y}_2$, and $\mathrm{Opt}(\mathbf{Y}_2, \mathbf{A}) \leq \mathrm{Opt}^{\mathrm{SA}^1}(\mathbf{X}, \mathbf{A})$. Then, by (ii) we have that $\mathrm{Opt}(\mathbf{Y}_1, \mathbf{B}) \leq \mathrm{Opt}^{\mathrm{SA}^1}(\mathbf{X}, \mathbf{A})$, and by Proposition 8.5, $\mathrm{Opt}(\mathbf{X}, \mathbf{B}) \leq \mathrm{Opt}^{\mathrm{SA}^1}(\mathbf{X}, \mathbf{A})$ too as required (see Figure 8.1 for a diagram of this proof).

(iii) $\Rightarrow$ (i). From Theorem 6.9 (adapted to the valued setting), if $\mathrm{Opt}^{\mathrm{SA}^1}(\mathbf{X}, \mathbf{A}) < \infty$, then there is a solution to the linear program that assigns the same value to every class of variables and constraints of $\mathbf{X}$ that have the same iterated degree.[3] This allows us to reduce the linear program as follows. Let $X/\equiv_1$ and $\mathcal{C}_\mathbf{X}/\equiv_1$ denote the sets of equivalence classes of variables and constraints, respectively, under the equivalence $\equiv_1$. The new linear program, denoted $\mathrm{SA}^1_\equiv(\mathbf{X}, \mathbf{A})$, contains one variable $p_{[x]}(a)$ for every class $[x] \in X/\equiv_1$ and one variable $p_{[R(\mathbf{x})]}(\mathbf{a})$ for every class $[R(\mathbf{x})] \in \mathcal{C}_\mathbf{X}/\equiv_1$. The variables of the new program $\mathrm{SA}^1_\equiv(\mathbf{X}, \mathbf{A})$ are subject to the same constraints as in $\mathrm{SA}^1(\mathbf{X}, \mathbf{A})$, except the sums run over the new reduced set of variables. The new objective function is

$$\mathrm{Opt}^{\mathrm{SA}^1_\equiv}(\mathbf{X}, \mathbf{A}) := \min \sum_{[R(\mathbf{x})] \in \mathcal{C}_\mathbf{X}/\equiv_1} k_{[R(\mathbf{x})]} \sum_{\mathbf{a} \in A^{\mathrm{ar}(R)}} p_{[R(\mathbf{x})]}(\mathbf{a}) R^\mathbf{X}(\mathbf{x}) R^\mathbf{A}(\mathbf{a}), \quad (8.9)$$

[3] While Theorem 6.9 is stated in terms of BLP, it is easy to see that the same proof also gives an equivalent result for SA^1.

where $k_{[R(\mathbf{x})]} = |[R(\mathbf{x})]|$ is the number of constraints equivalent to $R(\mathbf{x})$. By the above discussion, we have $\mathrm{Opt}^{\mathrm{SA}^1_{\equiv}}(\mathbf{X}, \mathbf{A}) = \mathrm{Opt}^{\mathrm{SA}^1}(\mathbf{X}, \mathbf{A})$. Therefore, since SA^1 decides $\mathrm{PVCSP}(\mathbf{A}, \mathbf{B})$, so does $\mathrm{SA}^1_{\equiv}$. (We remark here that two input structures with the same iterated degree have the same reduced $\mathrm{SA}^1_{\equiv}$ up to renaming of variables; this can be used e.g. to show that (iii) implies (ii).)

In order to show that (iii) implies (i), assume that $\mathbf{X}$ is a connected input structure. We show that every agent in the distributed network can obtain the reduced linear program $\mathrm{SA}^1_{\equiv}$ via a polynomial-time distributed algorithm. As $\mathrm{SA}^1_{\equiv}$ decides $\mathrm{PVCSP}(\mathbf{A}, \mathbf{B})$, this will conclude the proof.

The agents can calculate a finite and effectively computable representation of their iterated degree in polynomial time using the distributed version of the color refinement algorithm described in Section 6.4.2 (see Lemma 6.22.) Each agent $\psi(v), v \in X \cup \mathcal{C}_{\mathbf{X}}$ can then use the representation of the iterated degree as an identifier. Every agent can obtain sufficient information from its neighbours to compute the equations in (vBLP1), (vBLP2), (vBLP3) and (v↻) that constrain its relevant LP variables of the reduced system (and use the identifiers to name the LP variables), and can subsequently broadcast these along the network. We are left to show that every agent can also compute the objective function of $\mathrm{SA}^1_{\equiv}(\mathbf{X}, \mathbf{A})$. In fact, it is sufficient that every agent $\psi(R(\mathbf{x}))$ computes the summand of $\mathrm{Opt}^{\mathrm{SA}^1_{\equiv}}(\mathbf{X}, \mathbf{A})$ that corresponds to $[R(\mathbf{x})]$ and then broadcasts it in order to obtain the complete objective function. The only nontrivial piece of information to compute is the value of the coefficients $k_{[R(\mathbf{x})]}$.

By Remark 5.3, for each $R(\mathbf{x}) \in \mathcal{C}_{\mathbf{X}}$, each participating variable $x \in \{\mathbf{x}\}$, and $\ell = \ell_{\{x, R(\mathbf{x})\}}$, the coefficient $k_{[R(\mathbf{x})]}$ is equal to the number of ℓ-labeled edges from x into members of $[R(\mathbf{x})]$ (denoted $x[\ell, R(\mathbf{x})]$ in Remark 5.3) multiplied by the size of $[x]$. The former value can be computed by the agent $\psi(x)$, so $\psi(x)$ can compute the ratio $k_{[R(\mathbf{x}_1)]} : k_{[R(\mathbf{x}_2)]}$ for any two constraints $R(\mathbf{x}_1), R(\mathbf{x}_2)$ in which x participates. After broadcasting all these ratios, each agent can compute the ratios between any two $k_{[R(\mathbf{x})]}$ and, since the sum of these coefficients is $|\mathcal{C}_{\mathbf{X}}|$ (which is known to the agents), they can compute the coefficients. □

Clearly, the implication (iv) ⇒ (iii) in Theorem 7.2 remains true for PVCSP (so the equivalent statements in Theorem 8.7 are satisfied in, e.g.,

the PVCSPs in Example 8.3). The following example shows that, unlike for PCSPs, the converse implication does not hold in general: we provide an example of a PVCSP template that is decided by SA^1 but not by BLP.

Example 8.8

Let $\mathbf{A}$, $\mathbf{B}$ be σ-structures where σ contains a single binary relation symbol R. Let $A = B = \{0,1\}$, $R^{\mathbf{A}}(a,a) = R^{\mathbf{B}}(a,a) = 3$ for $a \in \{0,1\}$, and $R^{\mathbf{A}}(a,b) = 2$, $R^{\mathbf{B}}(a,b) = 0$ for $a \neq b \in \{0,1\}$. The probability distribution which assigns probability 1 to the identity function is a fractional homomorphism, so $(\mathbf{A},\mathbf{B})$ is a PVCSP template.

We claim that BLP does not decide $\mathrm{PVCSP}(\mathbf{A},\mathbf{B})$. Indeed, let $\mathbf{X}$ be the PVCSP input structure given by $X = \{x\}$ and $R^{\mathbf{X}}(x,x) = 1$. Then, there is a feasible solution to $\mathrm{BLP}(\mathbf{X},\mathbf{A})$ given by $p_x(a) = 1/2$ for $a \in \{0,1\}$ and $p_{R(x,x)}(a,a) = 0$, $p_{R(x,x)}(a,b) = 1/2$ for $a \neq b \in \{0,1\}$. This solution witnesses that $\mathrm{Opt}^{\mathrm{BLP}}(\mathbf{X},\mathbf{A}) \leq 2$, however, it is easy to see that $\mathrm{Opt}(\mathbf{X},\mathbf{B}) = 3$ and so BLP does not decide $\mathrm{PVCSP}(\mathbf{A},\mathbf{B})$.

On the other hand, we show that $\mathrm{Opt}(\mathbf{X},\mathbf{B}) \leq \mathrm{Opt}^{\mathrm{SA}^1}(\mathbf{X},\mathbf{A})$ for any input valued structure $\mathbf{X}$. Let $V_l(\mathbf{X}) = \sum_{x\in X} R^{\mathbf{X}}(x,x)$ and $V_e(\mathbf{X}) = \sum_{x\neq y} R^{\mathbf{X}}(x,y)$ be the total weight of the constraints in $\mathbf{X}$ with and without repetitions, respectively. We choose an assignment $h : X \to B$ at random: each $h(x)$ is chosen independently and uniformly (both 0 and 1 with probability 1/2). The expected value of $\mathrm{Val}(\mathbf{X},\mathbf{B},h)$ is $3V_l(\mathbf{X}) + 3/2V_e(\mathbf{X})$, which implies that $\mathrm{Opt}(\mathbf{X},\mathbf{B}) \leq 3V_l(\mathbf{X}) + 3/2V_e(\mathbf{X})$. As for SA^1, we know that any feasible solution must have $p_{(x,x)}(a,b) = 0$ whenever $a \neq b$. Therefore, we get

$$\begin{aligned}\mathrm{Opt}^{\mathrm{SA}^1}(\mathbf{X},\mathbf{A}) = \quad & \min\Big[\sum_{x\in X}\sum_{a\in A} p_{R(x,x)}(a,a)R^{\mathbf{X}}(x,x)R^{\mathbf{A}}(a,a) + \\ & \sum_{x\neq y\in X}\sum_{a,b\in A} p_{R(x,y)}(a,b)R^{\mathbf{X}}(x,y)R^{\mathbf{A}}(a,b)\Big] \\ \geq \quad & 3V_l(\mathbf{X}) + 2V_e(\mathbf{X}) > \mathrm{Opt}(\mathbf{X},\mathbf{B}).\end{aligned}$$

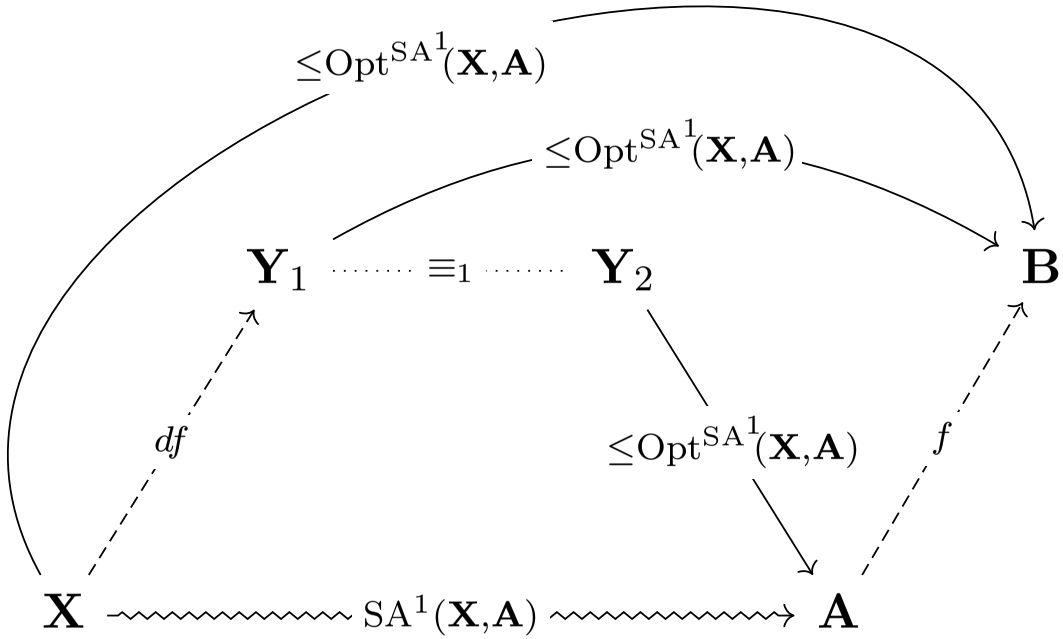

Figure 8.1: Diagram of the proof of (ii) ⇒ (iii) of Theorem 8.7. Fractional $(\text{-- } f \to)$ and dual fractional $(\text{-- } df \to)$ homomorphisms are pictured as dashed arrows, the equivalence relation $\equiv_1$ as a dotted line $(\cdots\cdots)$, the feasibility of the linear program SA^1 as a squiggly arrow $(\rightsquigarrow)$, and an upper bound of k on the optimum value for a pair of structures as a standard arrow $(- \leq k \to)$.

9 A Glimpse on the Higher Levels

In this chapter we discuss the higher-dimensional version of the Weisfeiler-Leman method and its equivalent combinatorial, logical, and algebraic characterizations. We then extend the notions of $\equiv_1$-equivalence to higher dimensions, and we present a combinatorial decomposition of the higher levels of the Sherali-Adams hierarchy.

9.1 k-WL, Counting Logics, and Treewidth

As mentioned in Section 5.1, colour refinement is a very powerful heuristic for testing isomorphism of graphs, in fact, it decides the isomorphism problem on almost all graphs (that is, on all but $o(2^{\binom{n}{2}})$ graphs on n vertices for every positive integer n) [BK79, BES80], and on all trees [IL90]. However, it also has some important limitation, for instance, colour refinement fails on specific classes of graphs, such as regular graphs (see Figure 9.1). For this reason, an increasingly powerful hierarchy of relaxations based on the Weisfeiler-Leman method, known as the *k-dimensional Weisfeiler-Leman algorithm* (k-WL), has been developed independently by multiple researchers.

As before, k-WL proceeds in iterations. At every iteration j on a graph $G = (V; E)$, k-WL produces a colouring $\chi_j^{G,k}$ of the set of k-tuples of V. In particular, for $\mathbf{v} \in V^k$, $\chi_0^{G,k}(\mathbf{v})$ is given by the isomorphism type of $\mathbf{v}$: that is, two k-tuples $\mathbf{u} = (u_1, \ldots, u_k)$ and $\mathbf{v} = (v_1, \ldots, v_k)$ have the same isomorphism type if and only if the mapping $u_i \mapsto v_i$ is an isomorphism between the subgraphs of G induced by $\{u_1, \ldots, u_k\}$ and

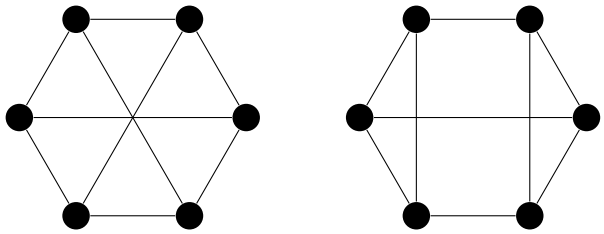

Figure 9.1: Both graphs depicted above are 3-regular and hence they are not distinguished by the colour refinement algorithm. However, they are clearly not isomorphic, since the left graph is bipartite while the right graph is not.

$\{v_1, \ldots, v_k\}$ respectively. For $j \geq 1$, k-WL sets

$$\chi_{j+1}^{G,k}(\mathbf{v}) = (\chi_j^{G,k}(\mathbf{v}), \mathcal{M}_j(\mathbf{v}))$$

where

$$\mathcal{M}_j(\mathbf{v}) = \{\{(\chi_j^{G,k}(\zeta_1(\mathbf{v}, w)), \ldots, \chi_j^{G,k}(\zeta_k(\mathbf{v}, w))) \mid w \in V\}\}$$

and

$$\zeta_i(\mathbf{v}, w) = (v_1, \ldots, v_{i-1}, w, v_{i+1}, \ldots, v_k)$$

for $k \geq 2$, and $\mathcal{M}_j(\mathbf{v}) = \{\{(\chi_j^{G,k}(w) \mid w \in N(\mathbf{v})\}\}$ for $k = 1$. Notice that in the latter case, 1-WL is precisely the colour refinement algorithm.

As in the 1-dimensional case, a stable colouring is eventually reached, and two graphs are said to be distinguished by k-WL if their stable colourings differ. Notice that for all practical purposes (for example in terms of isomorphism testing), it is enough to compute the partition induced by the stable colouring instead of the actual colours viewed as multisets, which can quickly become very large. In this setting, the stable k-WL colouring of an n-vertex graph can be computed in time $\mathcal{O}(n^{k+1} \log n)$ [IL90].

We say that k-WL *identifies* a graph G if it distinguishes G from all non-isomorphic graphs. As mentioned above, 1-WL identifies almost all graphs (the complete characterization of the graphs that are identified by 1-WL can be found in [KSS22]). It turns out that 2-WL is sufficient to identify almost all regular graphs [Bol82], however, it fails on strongly

regular graphs (the standard example of this being the 4×4 rook's graph and the Shrikhande graph; for an insightful digression on this, see the blog post [Ger17]). For a class of graphs $\mathcal{G}$, the *WL dimension* of $\mathcal{G}$ is the least integer k such that k identifies all graphs in $\mathcal{G}$. Remarkably, Grohe [Gro12] showed that every class of graphs with excluded minors has bounded WL dimension, where we say that H is a minor of G if it can be obtained by G via the operation of vertex and edge deletion and edge contraction. More recently, Kiefer et al. showed that the class of planar graphs has WL dimension 3 [KPS19] and that the class of graphs of treewidth at most k has WL dimension at most k [KN22]. Fore a more comprehensive survey on the power of the Weisfeiler-Leman method see [Kie20], and for its applications to isomorphism testing and an overview of Babai's algorithmm, see [GS20, GN21] as well as the recent 2-WL-based quasipolynomial parametrized algorithm for graphs with excluded minors [GWN20].

It had initially been conjectured that the k-dimensional Weisfeiler-Leman algorithm would provide a polynomial time isomorphism test for graphs of bounded degree. In their seminal paper, Cai, Fürer, and Immerman disprove this conjecture by constructing for every integer k a pair of non-isomorphic graphs on $\mathcal{O}(k)$ vertices which cannot be distinguished by k-WL (yet they are distinguished by $(k+1)$-WL).

Theorem 9.1 ([CFI92])**.** *For all $k \in \mathbb{N}$, there exist two graphs on $\mathcal{O}(k)$ vertices such that they are not distinguished by k-WL.*

The proof of indistinguishability of the graphs from Theorem 9.1, which came to be known as the *CFI construction*, relies on the equivalence between k-WL and the counting logic $\mathscr{C}^{k+1}$, which is the other main contribution in [CFI92]. While Theorem 9.1 shows the limitations of k-WL, the algorithm turned out to have a number of useful applications, such as the important role it plays in the quasipolynomial algorithm of Babai.

On the other hand, it is known from an old result of Lovász [Lov67] that counting homomorphisms characterizes isomorphism classes: that is, two graphs G and H are isomorphic if and only if for every graph F, F has the same number of homomorphisms into G and into H. We already saw in Chapter 5 how restricting this condition to counting homomorphisms from trees yields a combinatorial characterization of indistinguishability by 1-

WL. It turns out that considering not just trees but tree-like structures in a broader sense corresponds to indistinguishability under k-WL [Dvo10, DGR18]. To make this more precise we will need to introduce some more definitions.

Let $\mathscr{P}(A)$ denote the power set of A. A *tree-decomposition* [RS84] of a structure $\mathbf{A}$ is a pair (G, β) where $G = (V, E)$ is a tree and $\beta : V \to \mathscr{P}(A)$ is a mapping such that the following conditions are satisfied:

1. For every constraint $R(\mathbf{a})$ in $\mathcal{C}_{\mathbf{A}}$ there exists a node $v \in V$ such that $\{\mathbf{a}\} \subseteq \beta(v)$;
2. If $a \in \beta(u) \cap \beta(v)$ then $a \in \beta(w)$ for every node w in the unique path in G joining u to v.

The *width* of a tree-decomposition (G, β) is $\max\{|\beta(v)| \mid v \in V\} - 1$ and the *treewidth* of $\mathbf{A}$ is defined as the smallest width among all its tree-decompositions. It is not difficult to see that the only (undirected, loopless) graphs that have treewidth 1 are trees and forests (i.e. disjoint unions of trees). Moreover, a σ-ftree has treewidth at most $r - 1$, where r is the maximum arity of a relation in σ.

Then we have that the logical and combinatorial characterizations of colour refinement from Theorem 5.1 can be lifted to higher dimensions, as summarized in the following result:

Theorem 9.2 ([CFI92, Dvo10, DGR18])**.** *Let G, H be graphs. The following are equivalent:*

1. *k-WL does not distinguish G from H.*
2. *G and H satisfy the same formulae in the logic $\mathscr{C}^{k+1}$;*
3. *$\hom(T; G) = \hom(T; H)$ for all graphs T of treewidth at most k.*

Recall that 1-WL also has an algebraic characterization in terms of feasibility of the fractional isomorphism linear program (see item 1 in Theorem 5.1). Surprisingly, Atserias and Maneva [AM13] and Malkin [Mal14]) were able to also lift this correspondence to higher dimensions: in particular, they showed that a hierarchy of linear programs obtained by applying the Sherali-Adams method to the system of linear equations

defining fractional isomorphism interleaves with indistinguishability under k-WL. Subsequently, in [GO15] Grohe and Otto gave a variant of the SA-based relaxation of isomorphism whose levels correspond tightly with k-WL, and showed that the interleaving of the two hierarchies from [AM13, Mal14] is strict.

9.2 Sherali-Adams meets Weisfeiler-Leman

It is from this correspondence between the Sherali-Adams hierarchy applied to fractional isomorphism and the Weisfeiler-Leman algorithm that the inspiration for our work arises. More precisely, we ask the question: can a similar correspondence be established when applying the Sherali-Adams method to relaxations of homomorphism (instead of isomorphism) of arbitrary relational structures (instead of graphs)? The results in this section seem to indicate that the answer to this question is positive. A crucial step in establishing this correspondence is defining a transformation that allows us to reduce arbitrary relational structures to the world of binary structures (essentially, vertex- and edge-coloured digraphs).

For every $k > 0$ we define an operator $*_k$ that maps a σ-structure to a new structure whose signature we will denote σ_k^*. Let $\mathbf{A}$ be a σ-structure. Then we define the universe of $\mathbf{A}_k^*$ to be $A_k^* := \cup_{j \leq k} A^j \cup \mathcal{C}_{\mathbf{A}}$. Additionally, $\mathbf{A}_k^*$ contains the following unary ($T_{j,S}$, R_S) and binary ($T_{j,\mathbf{i}}$, $R_{\mathbf{i}}$) relations:

$$
\begin{aligned}
T_{j,S}^{\mathbf{A}_k^*} &= \{\mathbf{a} \in A^j \mid a_i = a_{i'}\ \forall i, i' \in S\} && j \leq k,\ S \subseteq [j] \\
T_{j,\mathbf{i}}^{\mathbf{A}_k^*} &= \{(\mathbf{a}, \pi_{\mathbf{i}}\mathbf{a}) \mid \mathbf{a} \in A^j\} && j', j \leq k,\ \mathbf{i} \in [j]^{j'} \\
R_S^{\mathbf{A}_k^*} &= \{R(\mathbf{a}) \in R^{\mathbf{A}} \mid a_i = a_{i'}\ \forall i, i' \in S\} && R \in \sigma,\ S \subseteq [\mathrm{ar}(R)] \\
R_{\mathbf{i}}^{\mathbf{A}_k^*} &= \{(R(\mathbf{a}), \pi_{\mathbf{i}}\mathbf{a}) \mid \mathbf{a} \in R^{\mathbf{A}}\} && R \in \sigma,\ j \leq k,\ \mathbf{i} \in [\mathrm{ar}(R)]^j.
\end{aligned}
$$

In the following remark we motivate the definition of $*_k$ from the point of view of pp-powers, a notion that proved to be very powerful in the context of the algebraic approach to CSP [BOP18].

Remark 9.3 ($\mathbf{A}_k^*$ as a multi-sorted pp-power).

An *n-sorted signature* σ is a list of relation symbols, each of which has an associated arity and an associated sort. As in the standard (1-sorted) case, the arity of a relation symbol R is a positive integer. The sort of R, denoted $\mathrm{sort}(R)$, is a list $\langle j_1, \ldots, j_{\mathrm{ar}(R)} \rangle$, where $j_i \in [n]$ for each $i \in [\mathrm{ar}(R)]$. An *n-sorted σ-structure* $\mathbf{A}$ consists of a list of domains $A_1, \ldots, A_n$, and a *sorted relation* $R^{\mathbf{A}} \subseteq A_{j_1} \times \ldots \times A_{j_{\mathrm{ar}(R)}}$ for each relation symbol $R \in \sigma$ with $\mathrm{sort}(R) = \langle j_1, \ldots, j_{\mathrm{ar}(R)} \rangle$.

A structure $\mathbf{B}$ is said to be a *pp-power* [BOP18] of a structure $\mathbf{A}$ (note that the signatures can differ) if $\mathbf{B}$ is isomorphic to a structure $\mathbf{D}$ whose domain D is a subset of A^k (for some $k \geq 1$) which is pp-definable from $\mathbf{A}$, and each r-ary relation in $\mathbf{D}$ is pp-definable from $\mathbf{A}$ if we regard it as a rk-ary relation on A. Notice that the power structure is a canonical example of a pp-power.

Then, $\mathbf{A}_k^*$ can be seen as a multi-sorted pp-power of $\mathbf{A}$ with domains $A^1, \ldots, A^k, R_1^{\mathbf{A}}, R_2^{\mathbf{A}}, \ldots$ (notice $R_i^{\mathbf{A}} \subseteq A^{\mathrm{ar}(R_i)}$). For instance, for $R \in \sigma$, $j \leq k$, and $\mathbf{i} \in [\mathrm{ar}(R)]^j$ the sorted relation $R_{\mathbf{i}}^{\mathbf{A}_k^*} \subseteq R^{\mathbf{A}} \times A^j$ would be defined from $\mathbf{A}$ by

$$R_{\mathbf{i}}(a_1, \ldots, a_{\mathrm{ar}(R)}, a'_1, \ldots, a'_j) = R(a_1, \ldots, a_{\mathrm{ar}(R)}) \wedge \bigwedge_{i=1}^{j} (a'_i = a_{\mathbf{i}[i]}).$$

To begin with, thanks to the $*_k$ transformation, we are able to reduce feasibility of SA^k to feasibility of SA^1.

Lemma 9.4. *Let* $\mathbf{X}$, $\mathbf{A}$ *be* σ*-structures. Then* $\mathrm{SA}^k(\mathbf{X}, \mathbf{A})$ *is feasible if and only if* $\mathrm{SA}^1(\mathbf{X}_k^*, \mathbf{A}_k^*)$ *is feasible.*

As an immediate consequence of Lemma 9.4 together with the decomposition theorem 7.1, we obtain the characterization of the Sherali-Adams relaxation for the homomorphism problem in terms of fractional isomorphism that we promised in Chapter 7.

Theorem 9.5. *Let* $\mathbf{X}$, $\mathbf{A}$ *be relational structures. Then, the following are equivalent:*

1. $\mathrm{SA}^k(\mathbf{X}, \mathbf{A})$ *is feasible;*
2. *There exists a sequence of structures* $\mathbf{Y}_0, \dots, \mathbf{Y}_n$ *such that* $\mathbf{Y}_0 = \mathbf{X}_k^*$, $\mathbf{Y}_n = \mathbf{A}_k^*$, *and for all* $i = 0, \dots, n-1$ *we have that* $\mathbf{Y}_i \to \mathbf{Y}_{i+1}$ *or* $\mathbf{Y}_i \equiv_1 \mathbf{Y}_{i+1}$*;*
3. *There exists a pair of structures* $\mathbf{Y}_1, \mathbf{Y}_2$ *such that* $\mathbf{X}_k^* \to \mathbf{Y}_1$, $\mathbf{Y}_1 \equiv_1 \mathbf{Y}_2$, *and* $\mathbf{Y}_2 \to \mathbf{A}_k^*$.

In particular, this implies that feasibility of SA^k (for a fixed right-hand structure $\mathbf{A}$) is closed under $\equiv_k$.

Corollary 9.6. *Let* $\mathbf{X}$, $\mathbf{X}'$ *and* $\mathbf{A}$ *be* σ*-structures and* $k \geq 1$. *Suppose that* $\mathbf{X} \equiv_k \mathbf{X}'$. *Then,* $\mathrm{SA}^k(\mathbf{X}, \mathbf{A})$ *is feasible if and only if* $\mathrm{SA}^k(\mathbf{X}', \mathbf{A})$ *is feasible.*

Furthermore, the operator $*_k$ allows us to define, similarly to the case of graphs, a hierarchy of increasingly tighter relaxations of isomorphism for relational structures. In particular, for every $k \geq 1$, we shall denote $\mathbf{A} \equiv_k \mathbf{B}$ whenever $\mathbf{A}_k^*$ and $\mathbf{B}_k^*$ satisfy the conditions of Theorem 5.4. It is easy to see that the case $k = 1$ would be unchanged if one replaces $\mathbf{A}_1^*$ and $\mathbf{B}_1^*$ by $\mathbf{A}$ and $\mathbf{B}$ respectively and, hence, $\equiv_k$ correctly extends $\equiv_1$. For other small values of k other than $k = 1$, the characterization of $\equiv_k$ is not so straightforward. However, as long as k is at least as large as the arity of any relation in the signature, then we have the following characterization, extending the analogous result for graphs (i.e., Theorem 9.2).

Theorem 9.7. *Let* r *be the maximum arity among all relations in* σ *and assume that* $r \leq k$. *Then for every pair of structures* $\mathbf{A}$, $\mathbf{B}$ *the following are equivalent:*

1. $\mathbf{A} \equiv_k \mathbf{B}$*;*
2. $\hom(\mathbf{X}; \mathbf{A}) = \hom(\mathbf{X}; \mathbf{B})$ *for every* σ*-structure* $\mathbf{X}$ *of treewidth* $< k$*;*
3. $\mathbf{A}$ *and* $\mathbf{B}$ *satisfy the same formulae in the logic* $\mathscr{C}^k$.

The proof of this theorem requires item (4) of Theorem 5.4, and so we will provide the proofs of both Theorem 5.4 and Theorem 9.7 below. In fact, it is not surprising that a result similar to Theorem 9.2 can be shown for $\equiv_k$ since, after all, the k-WL algorithm can be seen as the 1-WL algorithm applied to k-ary tuples. However, note that the bound on the treewidth and the number of variables allowed in the logic differ in one unit between k-WL and $\equiv_k$ – that is, $\mathbf{A} \equiv_k \mathbf{B}$ corresponds to homomorphism indistinguishability from structures of treewidth *less than* k (and not *at most* k) and to indistinguishability in $\mathscr{C}^k$ (and not $\mathscr{C}^{k+1}$).

We conclude by presenting the missing proofs of the results mentioned in this chapter.

9.3 Proof of Lemma 9.4

Lemma 9.4. *Let* $\mathbf{X}$, $\mathbf{A}$ *be* σ*-structures. Then* $\mathrm{SA}^k(\mathbf{X}, \mathbf{A})$ *is feasible if and only if* $\mathrm{SA}^1(\mathbf{X}^*_k, \mathbf{A}^*_k)$ *is feasible.*

The proof is purely syntactical, although it is convenient to first slightly modify the LP formulations $\mathrm{SA}^k(\mathbf{X}, \mathbf{A})$ and $\mathrm{SA}^1(\mathbf{X}^*_k, \mathbf{A}^*_k)$. We shall refer to the solutions of $\mathrm{SA}^k(\mathbf{X}, \mathbf{A})$ and $\mathrm{SA}^1(\mathbf{X}^*_k, \mathbf{A}^*_k)$ by appropriately indexed sets of variables p, q respectively.

- In $\mathrm{SA}^k(\mathbf{X}, \mathbf{A})$, it follows from (SA4) that we can safely replace all variables $p_{R(\mathbf{x})}(f)$ with $f(\mathbf{x}) \notin R^{\mathbf{A}}$ by 0.

- In $\mathrm{SA}^1(\mathbf{X}^*_k, \mathbf{A}^*_k)$ some more substantial observations are needed. First, for each $j \leq k$ and each $\mathbf{x} \in X^j$, it follows from conditions (SA3) and (SA4) for $T_{j,S}$ ($S \subseteq [j]$) that for every v in A^*_k, $q_{\mathbf{x}}(v)$ must take value 0 unless $v = f(\mathbf{x})$ for some function $f : \{\mathbf{x}\} \to A$. Hence, in a first stage we set $q_{\mathbf{x}}(v)$ to zero for each $j \leq k$, each $\mathbf{x} \in X^j$ and each v that is not a tuple of the form $f(\mathbf{x})$ for some function $f : \{\mathbf{x}\} \to A$.

  Furthermore, it follows from condition (SA3) for $T_{j,\mathbf{i}}$ that $q_{\mathbf{x}}(f(\mathbf{x})) = q_{\mathbf{x}'}(f(\mathbf{x}'))$ for every $\mathbf{x}$, $\mathbf{x}'$ satisfying $\{\mathbf{x}\} = \{\mathbf{x}'\}$ and every $f : \{\mathbf{x}\} \to A$. Hence, in a second stage, for each $V \subseteq X$ with $|V| \leq k$ and every $f : V \to A$ we identify all variables $q_{\mathbf{x}}(f(\mathbf{x}))$ which satisfy $\{\mathbf{x}\} = V$.

Then, consider now the variables of the form $q_{R(\mathbf{x})}(v)$, $v \in A_k^*$. It follows from conditions (SA3) and (SA4) for R_S ($S \subseteq [\mathrm{ar}(R)]$) that $q_{R(\mathbf{x})}(v)$ must be set to 0 unless $v = R(f(\mathbf{x}))$ for some function $f : \{\mathbf{x}\} \to A$.

The other variables in $\mathrm{SA}^1(\mathbf{X}_k^*, \mathbf{A}_k^*)$ are of the form $q_C(f)$ where $C \in \mathcal{C}_{\mathbf{X}_k^*}$. As we shall see they can always safely be identified with some of the other variables. Let us start first with the case in which C is a unary constraint. If $C = T_{j,S}(\mathbf{x})$ or $C = R_S(\mathbf{x})$, then it follows from (SA3) that $q_C(f) = q_{\mathbf{x}}(f(\mathbf{x}))$. Assume now that C is a binary constraint, that is $C = T_{j,\mathbf{i}}(\mathbf{x}, \pi_{\mathbf{i}}\mathbf{x})$ or $C = R_{\mathbf{i}}(\mathbf{x}, \pi_{\mathbf{i}}\mathbf{x})$. It follows again from (SA3) that $q_C(f) = q_{\mathbf{x}}(f(\mathbf{x}))$.

Now we are ready to prove the lemma. In particular, consider the following one-to-one correspondence between the assignments in $\mathrm{SA}^k(\mathbf{X}, \mathbf{A})$ and $\mathrm{SA}^1(\mathbf{X}_k^*, \mathbf{A}_k^*)$:

- Every variable $p_V(f)$ of $\mathrm{SA}^k(\mathbf{X}, \mathbf{A})$ is assigned to the variable $q_{\mathbf{x}}(f(\mathbf{x}))$ of $\mathrm{SA}^1(\mathbf{X}_k^*, \mathbf{A}_k^*)$, where $\mathbf{x}$ is any tuple satisfying $\{\mathbf{x}\} = V$.
- Every variable $p_{R(\mathbf{x})}(f)$ of $\mathrm{SA}^k(\mathbf{X}, \mathbf{A})$ is assigned to the variable $q_{R(\mathbf{x})}(R(f(\mathbf{x})))$ of $\mathrm{SA}^1(\mathbf{X}_k^*, \mathbf{A}_k^*)$.

It is not difficult to see that this correspondence preserves feasibility.

9.4 Proof of Theorem 5.4

Theorem 5.4. *Let* $\mathbf{A}$, $\mathbf{B}$ *be* σ*-structures. The following are equivalent:*

1. *There exist doubly stochastic matrices* P, Q *such that for every* $\ell \in \mathcal{L}_\sigma$ *it holds that* $PM_{\mathbf{A}}^\ell = M_{\mathbf{B}}^\ell Q$ *and* $M_{\mathbf{A}}^\ell Q^T = P^T M_{\mathbf{B}}^\ell$*;*
2. $\mathbf{A}$ *and* $\mathbf{B}$ *have the same iterated degree sequence;*
3. $\mathbf{A}$ *and* $\mathbf{B}$ *have a common equitable partition;*
4. $\hom(\mathbf{T}; \mathbf{A}) = \hom(\mathbf{T}; \mathbf{B})$ *for all* σ*-ftrees* $\mathbf{T}$.

If, additionally, $\mathbf{A}$ *and* $\mathbf{B}$ *are graphs, then the following is also equivalent:*

6. *There exists a doubly stochastic matrix P such that $PN_{\mathbf{A}} = N_{\mathbf{B}}P$.*

Before starting we note that two structures $\mathbf{A}$ and $\mathbf{B}$ have a common equitable partition if there is an equitable partition $(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{Q}}) = (\{\mathcal{P}_i \mid i \in I\}, \{\mathcal{Q}_j \mid j \in J\})$ of the disjoint union $\mathbf{A} \cup \mathbf{B}$ such that

1. $|\mathcal{P}_i \cap A| = |\mathcal{P}_i \cap B|$ for every $i \in I$, and
2. $|\mathcal{Q}_j \cap \mathcal{C}_{\mathbf{A}}| = |\mathcal{Q}_j \cap \mathcal{C}_{\mathbf{B}}|$ for every $j \in J$.

Note that since $(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{Q}})$ is equitable, then any of conditions (1) or (2) implies the other. We might freely switch between the two alternative characterizations of common equitable partition.

We also need a few additional definitions before embarking on the proof. We denote by J_V the $V \times V$ matrix whose entries are all ones and we use $\oplus$ to denote direct sums of matrices. Let M be a $V \times W$ matrix and consider two subsets $S_V \subseteq V$, $S_W \subseteq W$. The restriction of M to S_V and S_W, denoted by $M[S_V, S_W]$, is the matrix obtained by removing from M all the rows that do not belong in S_V and all the columns that do not belong in S_W.

A matrix $M \in \mathbb{R}^{V \times W}$ is *decomposable* if there exists a partition (V_1, V_2) of V and a partition (W_1, W_2) of W such that for every two distinct $i, j \in [2]$ and every $v \in V_i$ and $w \in W_j$, $M_{v,w} = 0$. In this case we can write $M = M_1 \oplus M_2$ where $M_i = M[V_i, W_i]$. Otherwise, M is said to be *indecomposable*.

Let $P \in [0,1]^{V \times V}$ be a doubly stochastic matrix. Note that P has a unique decomposition $P = \oplus_{i \in I} P_i$ where each P_i is an indecomposable doubly stochastic matrix. The *row partition* of P is defined to be the partition of V into classes $\mathcal{P}_i, i \in I$ where $\mathcal{P}_i$ contains the rows of P_i, and the *column partition* is defined in an analogous manner.

We will also need the following lemma (see for example [SU11]).

Lemma 9.8. *Let $P \in \mathbb{R}^{V \times V}$, $Q \in \mathbb{R}^{W \times W}$ be doubly stochastic indecomposable matrices.*

1. *Let $\mathbf{a}, \mathbf{b} \in \mathbb{R}^V$ such that $P \cdot \mathbf{a} = \mathbf{b}$ and $P^T \cdot \mathbf{b} = \mathbf{a}$. Then, there exists $c \in \mathbb{R}$ such that*
$$\mathbf{a} = \mathbf{b} = c \cdot \mathbf{1}.$$

2. *Let $M_1, M_2 \in \mathbb{R}^{V\times W}$ such that $PM_1 = M_2Q$ and $M_1Q^T = P^TM_2$. Then, there exist $c, d \in \mathbb{R}$ such that the following identities hold:*

$$M_1\mathbf{1} = M_2\mathbf{1} = c\cdot\mathbf{1} \qquad\qquad \mathbf{1}^TM_1 = \mathbf{1}^TM_2 = d\cdot\mathbf{1}^T$$

Proof. Item (1) is Theorem 6.2.4 (ii) from [SU11]. We enclose a proof of item (2). Let $\mathbf{a} = M_1\cdot\mathbf{1}$ and $\mathbf{b} = M_2\cdot\mathbf{1}$. We have $PM_1\cdot\mathbf{1} = M_2Q\cdot\mathbf{1} = M_2\cdot\mathbf{1}$ since Q is doubly stochastic, which implies $P\cdot\mathbf{a} = \mathbf{b}$. Similarly, we have $M_1\cdot\mathbf{1} = M_1Q^T\cdot\mathbf{1} = P^TM_2\cdot\mathbf{1}$ which implies $\mathbf{a} = P^T\cdot\mathbf{b}$. Then, we apply item (1) to deduce that there exists $c\in\mathbb{R}$ such that $M_1\mathbf{1} = M_2\mathbf{1} = c\cdot\mathbf{1}$. The other identity is proven in an analogous way. □

(1) ⇒ (3). Let P and Q be doubly stochastic matrices satisfying (1). Let $P = \oplus_{i\in I}P_i$, $Q = \oplus_{j\in J}Q_j$ be decompositions of P and Q. Denote by $(\boldsymbol{\mathcal{P}}^{\mathbf{A}}, \boldsymbol{\mathcal{P}}^{\mathbf{B}})$ the column and row partitions of P respectively and by $(\boldsymbol{\mathcal{Q}}^{\mathbf{A}}, \boldsymbol{\mathcal{Q}}^{\mathbf{B}})$ the column and row partitions of Q. Then, the restrictions of $M^\ell_{\mathbf{A}}$ and $M^\ell_{\mathbf{B}}$ to $(\mathcal{P}^{\mathbf{A}}_i, \mathcal{Q}^{\mathbf{A}}_j)$, $(\mathcal{P}^{\mathbf{B}}_i, \mathcal{Q}^{\mathbf{B}}_j)$ respectively satisfy

$$P_iM^\ell_{\mathbf{A}}[\mathcal{P}^{\mathbf{A}}_i, \mathcal{Q}^{\mathbf{A}}_j] = M^\ell_{\mathbf{B}}[\mathcal{P}^{\mathbf{B}}_i, \mathcal{Q}^{\mathbf{B}}_j]Q_j,$$

$$M^\ell_{\mathbf{A}}[\mathcal{P}^{\mathbf{A}}_i, \mathcal{Q}^{\mathbf{A}}_j]Q^T_j = P^T_iM^\ell_{\mathbf{B}}[\mathcal{P}^{\mathbf{B}}_i, \mathcal{Q}^{\mathbf{B}}_j].$$

Hence it follows from Lemma 9.8 that there exist $c^\ell_{i,j}$, $d^\ell_{i,j}$ such that

$$M^\ell_{\mathbf{A}}[\mathcal{P}^{\mathbf{A}}_i, \mathcal{Q}^{\mathbf{A}}_j]\cdot\mathbf{1} = M^\ell_{\mathbf{B}}[\mathcal{P}^{\mathbf{B}}_i, \mathcal{Q}^{\mathbf{B}}_j]\cdot\mathbf{1} = c^\ell_{i,j}\cdot\mathbf{1}$$
$$\mathbf{1}^T\cdot M^\ell_{\mathbf{A}}[\mathcal{P}^{\mathbf{A}}_i, \mathcal{Q}^{\mathbf{A}}_j] = \mathbf{1}^T\cdot M^\ell_{\mathbf{B}}[\mathcal{P}^{\mathbf{B}}_i, \mathcal{Q}^{\mathbf{B}}_j] = \mathbf{1}^T\cdot d^\ell_{i,j}$$

which is equivalent to conditions (P1) and (P2) showing that partitions $(\boldsymbol{\mathcal{P}}^{\mathbf{A}}, \boldsymbol{\mathcal{Q}}^{\mathbf{A}})$ and $(\boldsymbol{\mathcal{P}}^{\mathbf{B}}, \boldsymbol{\mathcal{Q}}^{\mathbf{B}})$ have the same parameters.

(3) ⇒ (2). Assume that $(\{\mathcal{P}^{\mathbf{A}}_i \mid i\in I\}, \{\mathcal{Q}^{\mathbf{A}}_j \mid j\in J\})$ and $(\{\mathcal{P}^{\mathbf{B}}_i \mid i\in I\}, \{\mathcal{Q}^{\mathbf{B}}_j \mid j\in J\})$ define a common equitable partition of $\mathbf{A}$ and $\mathbf{B}$. We shall prove that any two elements that are in the same set of the partition of $\mathbf{A}$ and $\mathbf{B}$ must have the same iterated degree. That is, we show by induction on k that for all $k\geq 0$, $\delta^{\mathbf{A}}_k(a) = \delta^{\mathbf{B}}_k(b)$ whenever there exists $i\in I$ such that $a\in\mathcal{P}^{\mathbf{A}}_i$ and $b\in\mathcal{P}^{\mathbf{B}}_i$ (the case $R(\mathbf{a})\in\mathcal{Q}^{\mathbf{A}}_j, R(\mathbf{b})\in\mathcal{Q}^{\mathbf{B}}_j$ is analogous). The base case ($k=0$) is immediate. For the inductive case, assume that the statement holds for $k-1$. Let (ℓ,δ) be any arbitrary

element in $\delta_k^{\mathbf{A}}(a)$. We shall show that it has the same multiplicity in $\delta_k^{\mathbf{A}}(a)$ and in $\delta_k^{\mathbf{B}}(b)$. By the inductive hypothesis it follows that there exists $J_\delta \subseteq J$ such that

$$\{R(\mathbf{a}) \in \mathcal{C}_{\mathbf{A}} \mid \delta_{k-1}^{\mathbf{A}}(R(\mathbf{a})) = \delta\} = \bigcup_{j \in J_\delta} \mathcal{Q}_j^{\mathbf{A}}$$

$$\{R(\mathbf{b}) \in \mathcal{C}_{\mathbf{B}} \mid \delta_{k-1}^{\mathbf{B}}(R(\mathbf{a})) = \delta\} = \bigcup_{j \in J_\delta} \mathcal{Q}_j^{\mathbf{B}}$$

Then, the multiplicity of (ℓ, δ) in $\delta_k^{\mathbf{A}}(a)$ is

$$|\{R(\mathbf{a}) \in \bigcup_{j \in J_\delta} \mathcal{Q}_j^{\mathbf{A}} \mid M_{\mathbf{A}}[a, R(\mathbf{a})] = \ell\}|. \tag{9.1}$$

Since $(\boldsymbol{\mathcal{P}}^{\mathbf{A}}, \boldsymbol{\mathcal{Q}}^{\mathbf{A}})$ is an equitable partition it follows that (9.1) is equal to $\sum_{j \in J_\delta} c_{i,j}^{\mathbf{A},\ell}$ where $c_{i,j}^{\mathbf{A},\ell}$ are the parameters of the partition.

It can analogously be shown that the the multiplicity of (ℓ, δ) in $\delta_k^{\mathbf{B}}(b)$ is $\sum_{j \in J_\delta} c_{i,j}^{\mathbf{B},\ell}$. Since $(\boldsymbol{\mathcal{P}}^{\mathbf{A}}, \boldsymbol{\mathcal{Q}}^{\mathbf{A}})$ and $(\boldsymbol{\mathcal{P}}^{\mathbf{B}}, \boldsymbol{\mathcal{Q}}^{\mathbf{B}})$ define a common equitable partition it follows that $c_{i,j}^{\mathbf{A},\ell} = c_{i,j}^{\mathbf{B},\ell}$ for every $i \in I, j \in J$ and we are done.

(2) $\Rightarrow$ (3). Assume that $\mathbf{A}$ and $\mathbf{B}$ have the same iterated degree sequence. Let $\boldsymbol{\mathcal{P}} = \{\mathcal{P}_i \mid i \in I\}$ and $\boldsymbol{\mathcal{Q}} = \{\mathcal{Q}_j \mid j \in J\}$ be the partition of $A \cup B$ and $\mathcal{C}_{\mathbf{A}} \cup \mathcal{C}_{\mathbf{B}}$ induced by the fixed point $\delta^{\mathbf{A} \cup \mathbf{B}}$. It is easy to verify that $(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{Q}})$ defines a common equitable partition.

(3) $\Rightarrow$ (1). Assume that $(\{\mathcal{P}_i^{\mathbf{A}} \mid i \in I\}, \{\mathcal{Q}_j^{\mathbf{A}} \mid j \in J\})$ and $(\{\mathcal{P}_i^{\mathbf{B}} \mid i \in I\}, \{\mathcal{Q}_j^{\mathbf{B}} \mid j \in J\})$ define a common equitable partition. For ease of notation it is convenient that matrices $M_{\mathbf{A}}^\ell$ and $M_{\mathbf{B}}^\ell$ are indexed by the same sets of rows V and columns W, which can be done by fixing a one-to-one correspondence between A, B, and V and similarly between $\mathcal{C}_{\mathbf{A}}$, $\mathcal{C}_{\mathbf{B}}$, and W. Furthermore, the correspondence can be established in such a way that under this correspondence $\boldsymbol{\mathcal{P}}^{\mathbf{A}}$ and $\boldsymbol{\mathcal{P}}^{\mathbf{B}}$ become the same partition $\boldsymbol{\mathcal{P}}$ over V, and $\boldsymbol{\mathcal{Q}}^{\mathbf{A}}$ and $\boldsymbol{\mathcal{Q}}^{\mathbf{B}}$ become the same partition $\boldsymbol{\mathcal{Q}}$ over W.

Define $P = \oplus_{i \in I} \frac{1}{|\mathcal{P}_i|} J_{\mathcal{P}_i}$ and $Q = \oplus_{j \in J} \frac{1}{|\mathcal{Q}_j|} J_{\mathcal{Q}_j}$. Clearly P and Q are doubly stochastic. It remains to show that $PM_{\mathbf{A}}^\ell = M_{\mathbf{B}}^\ell Q$ and $M_{\mathbf{A}}^\ell Q^T = P^T M_{\mathbf{B}}^\ell$ for all labels $\ell \in \mathcal{L}_\sigma$. Now since $(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{Q}})$ is an equitable partition,

it follows that for every $i \in I$, $j \in J$, and $\ell \in \mathcal{L}_\sigma$ there exist parameters c_{ij}^ℓ and d_{ji}^ℓ which satisfy conditions (P1) and (P2).

Then for all $i \in I$, $j \in J$ and $\ell \in \mathcal{L}_\sigma$ it holds that

$$|\mathcal{P}_i| c_{ij}^\ell = |\mathcal{Q}_j| d_{ji}^\ell,$$

and that the sum of the elements of the respective $\mathcal{P}_i \times \mathcal{Q}_j$ portions of $M_{\mathbf{A}}^\ell$ and $M_{\mathbf{B}}^\ell$ are equal. Now let $b \in \mathcal{P}_i \cap B$ and $R(\mathbf{a}) \in \mathcal{Q}_j \cap \mathcal{C}_{\mathbf{A}}$. Then

$$(PM_{\mathbf{A}}^\ell)[b, R(\mathbf{a})] = \frac{1}{|\mathcal{P}_i|} d_{ji}^\ell = \frac{1}{|\mathcal{Q}_j|} c_{ij}^\ell = (M_{\mathbf{B}}^\ell Q)[b, R(\mathbf{a})],$$

showing that $PM_{\mathbf{A}}^\ell = M_{\mathbf{B}}^\ell Q$ for all labels $\ell \in \mathcal{L}_\sigma$ as required, and similarly, noting that $P^T = P$ and $Q^T = Q$, we obtain that $M_{\mathbf{A}}^\ell Q^T = P^T M_{\mathbf{B}}^\ell$ for all $\ell \in \mathcal{L}_\sigma$ too.

(1) $\Rightarrow$ (6). Assume that P and Q satisfy (1). We shall prove that P satisfies (6).

Let R be the unique binary edge relation symbol in σ. We define two matrices $U_{\mathbf{A}}$, $Z_{\mathbf{A}}$ where $U_{\mathbf{A}} = (M_{\mathbf{A}}^{(1,R)} + M_{\mathbf{A}}^{(2,R)})/\sqrt{2}$ and $Z_{\mathbf{A}} = 2M_{\mathbf{A}}^{(1,R)}/\sqrt{2}$. Then we have that $N_{\mathbf{A}} = U_{\mathbf{A}} U_{\mathbf{A}}^T - Z_{\mathbf{A}} Z_{\mathbf{A}}^T$. We similarly obtain $N_{\mathbf{B}} = U_{\mathbf{B}} U_{\mathbf{B}}^T - Z_{\mathbf{B}} Z_{\mathbf{B}}^T$ if we define $U_{\mathbf{B}}$ and $Z_{\mathbf{B}}$ accordingly. Note that the identities of (1) are still satisfied if we replace $M_{\mathbf{A}}^\ell$ and $M_{\mathbf{B}}^\ell$ by $U_{\mathbf{A}}$ and $U_{\mathbf{B}}$ or $Z_{\mathbf{A}}$ and $Z_{\mathbf{B}}$ respectively.

It then follows that

$$PU_{\mathbf{A}} U_{\mathbf{A}}^T = U_{\mathbf{B}} Q U_{\mathbf{A}}^T = U_{\mathbf{B}} U_{\mathbf{B}}^T P.$$

and

$$PZ_{\mathbf{A}} Z_{\mathbf{A}}^T = Z_{\mathbf{B}} Q Z_{\mathbf{A}}^T = Z_{\mathbf{B}} Z_{\mathbf{B}}^T P.$$

Therefore, $PN_{\mathbf{A}} = N_{\mathbf{B}} P$ as required.

(6) $\Rightarrow$ (3). Let R be the unique binary edge relation symbol in σ. Let P be a doubly stochastic matrix satisfying $PN_{\mathbf{A}} = N_{\mathbf{B}} P$. Since $N_{\mathbf{A}}$ and $N_{\mathbf{B}}$ are symmetric it also holds that $N_{\mathbf{A}} P^T = P^T N_{\mathbf{B}}$. Let $P = \oplus_{i \in I} P_i$ be a decomposition of P and denote by $\{\mathcal{P}_i^{\mathbf{A}} \mid i \in I\}$ and $\{\mathcal{P}_i^{\mathbf{B}} \mid i \in I\}$ the column and row partitions of P. Applying the same reasoning as in (1) $\Rightarrow$ (3) it follows that for every $i, i' \in I$, there exists $c_{i,i'}$ such that for every $\mathbf{D} \in \{\mathbf{A}, \mathbf{B}\}$ and every $d \in \mathcal{P}_i^{\mathbf{D}}$

$$\{d' \in \mathcal{P}_{i'}^{\mathbf{D}} \mid R(d, d') \in \mathcal{C}_{\mathbf{A}}\} = c_{i,i'}$$

It then easily follows that if we let $\boldsymbol{\mathcal{Q}}^{\mathbf{A}}$, $\boldsymbol{\mathcal{Q}}^{\mathbf{B}}$ be the partitions of $\mathcal{C}_{\mathbf{A}}$, $\mathcal{C}_{\mathbf{B}}$ respectively given by assigning two edges to the same partition class if the partition classes of their vertices coincide then $(\boldsymbol{\mathcal{P}}^{\mathbf{A}}, \boldsymbol{\mathcal{Q}}^{\mathbf{A}})$ and $(\boldsymbol{\mathcal{P}}^{\mathbf{B}}, \boldsymbol{\mathcal{Q}}^{\mathbf{B}})$ define a common equitable partition of $\mathbf{A}$ and $\mathbf{B}$.

The proof of (3) $\Leftrightarrow$ (4) requires some more work. We shall use the following technical lemma.

Claim 9.9. Let $n \geq 1$, let X be a family of n-ary vectors of non-negative real numbers such that for every $\mathbf{a}, \mathbf{b} \in X$, if the new vector $\mathbf{a} \cdot \mathbf{b}$ obtained by multiplying $\mathbf{a}$ and $\mathbf{b}$ component-wise is different from $\mathbf{0}$ then it also belongs to X. Let $S_X = \{i \in [n] \mid \forall \mathbf{a} \in X(\mathbf{a}[i] \neq 0)\}$ and $D_X = \{(i,j) \mid \exists \mathbf{a} \in X(\mathbf{a}[i] \neq \mathbf{a}[j])\}$. Then

1. There exists $\mathbf{a} \in X$ such that $\mathbf{a}[i] = 0$ for every $i \notin S_X$ and $\mathbf{a}[i] \neq \mathbf{a}[j]$ for every $(i,j) \in D_X \cap (S_X \times S_X)$.
2. Furthermore if D_X contains all non reflexive pairs in $[n] \times [n]$ then X contains n linearly independent vectors.

Proof of Claim 9.9. (1). We shall prove that for every subset Y of X there exists an assignment satisfying (1) replacing S_X by S_Y and $D_X \cap (S_X \times S_X)$ by $D_Y \cap (S_Y \times S_Y)$. The proof is by induction on $|Y|$. The base case when Y only contains a single vector is trivial. For the inductive case, let $Y = Z \cup \{\mathbf{b}\}$ and let $\mathbf{a}$ be the tuple satisfying the claim for Z. It follows immediately that, for $d > 0$ large enough, $(\mathbf{a})^d \cdot (\mathbf{b})$ satisfies the claim for Y.

(2). The proof is by induction on n. The base case is when $S_X = [n]$. In this case, let $\mathbf{a} \in X$ be the tuple given by item (1). It follows immediately that the matrix A with rows $(\mathbf{a})^1, (\mathbf{a})^2, \ldots, (\mathbf{a})^n$ is non-singular as a Vandermonde matrix (see Figure 9.2) can be easily obtained if we multiply column i by $1/\mathbf{a}[i]$, for every $i \in [n]$.

For the inductive case, we can assume that $S_X \neq [n]$. Pick a vector $\mathbf{y} \in X$ where the set $I = I_{\mathbf{y}}$ defined as $I_{\mathbf{y}} := \{i \in [n] \mid \mathbf{y}[i] \neq 0\}$ is non-empty and as small as possible. Note that $I \neq [n]$ since $S_X \neq [n]$. Let

$$Y := \{\mathbf{a} \in X \mid \forall i \in I(\mathbf{a}[i] \neq 0)\}.$$

Note that $S_Y \subseteq I$ since $\mathbf{y} \in Y$ and $I \subseteq S_Y$ by the minimality of Y. Also, D_Y contains all irreflexive pairs in $I \times I$. Indeed, for every such pair (i,j),

$$\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ a_1 & a_2 & a_3 & \cdots & a_m \\ a_1^2 & a_2^2 & a_3^2 & \cdots & a_m^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_1^{n-1} & a_2^{n-1} & a_3^{n-1} & \cdots & a_m^{n-1} \end{pmatrix}$$

Figure 9.2: The shape of a Vandermonde matrix. Notice that in our case $n = m$.

let $\mathbf{b}$ be the tuple in X witnessing (i, j). Clearly $\mathbf{b}[i] \neq 0$ or $\mathbf{b}[j] \neq 0$. It follows that $\mathbf{b} \in Y$, since otherwise $I_{\mathbf{y}\cdot\mathbf{b}}$ would be strictly smaller than I yet non-empty.

Now, let $\mathbf{z}$ be the tuple obtained by applying item (1) to Y.

Consider the set Z defined as

$$Z := \{\mathbf{a} \in Y \mid \forall i \notin I(\mathbf{a}[i] = 0)\}$$

Note that $\mathbf{z} \in Z$ and $\mathbf{z}[i] \neq \mathbf{z}[j]$ for every $i \neq j \in I$. Then by the inductive hypothesis,

$$\pi_I Z := \{\pi_I \mathbf{a} \mid \mathbf{a} \in Z\}$$

has $|I|$ independent vectors $\pi_I \mathbf{a}_1 \ldots, \pi_I \mathbf{a}_{|I|}$. Also, note that $D_{\pi_{\overline{I}} X}$ has all irreflexive pairs in $\overline{I} \times \overline{I}$ and, hence, by the inductive hypothesis $\pi_{\overline{I}} X$ has $n - |I|$ independent vectors $\pi_{\overline{I}} \mathbf{a}_{|I|+1}, \ldots, \pi_{\overline{I}} \mathbf{a}_n$. It follows easily that $\mathbf{a}_1 \ldots, \mathbf{a}_n$ are linearly independent. ■

It will be convenient to endow a structure $\mathbf{D}$ with designated elements and constraints. That is, given $d_1, \ldots, d_n$ where $d_i \in D \cup \mathcal{C}_{\mathbf{D}}$ we shall use $(\mathbf{D}, d_1, \ldots, d_n)$ to denote the structure obtained from $\mathbf{D}$ where $d_1, \ldots, d_n$ have been *designated*. Homomorphisms are extended in a natural way to this more general setting. That is, a homomorphism from $(\mathbf{D}, d_1, \ldots, d_n)$ to $(\mathbf{E}, e_1, \ldots, e_n)$ is any homomorphism h from $\mathbf{D}$ to $\mathbf{E}$ such that, additionally, $h(d_i) = e_i$ for all $i \in [n]$ where if d_i is a constraint, say, $d_i = R(\mathbf{d})$ then $h(d_i)$ is defined to be the constraint $R(h(\mathbf{d}))$. Similarly, we shall denote the number of homomorphisms from $(\mathbf{D}, d_1, \ldots, d_n)$ to $(\mathbf{E}, e_1, \ldots, e_n)$ by $\hom(\mathbf{D}, d_1, \ldots, d_n; \mathbf{E}, e_1, \ldots, e_n)$. A *rooted ftree* is any structure $(\mathbf{T}, t)$ where $\mathbf{T}$ is an ftree and $t \in T$.

The following definitions, although somewhat technical, will be used a few times. Assume that we want to compute $\hom(\mathbf{T}, t, R(\mathbf{t}); \mathbf{D}, d, R'(\mathbf{d}))$ where $t \in T$ and $d \in D$. We say that $(t, R(\mathbf{t}))$ *can be mapped to* $(d, R'(\mathbf{d}))$ if the following conditions hold:

1. $R = R'$, and
2. for every $s, s' \in [\mathrm{ar}(R)]$, $\mathbf{t}[s] = \mathbf{t}[s']$ implies $\mathbf{d}[s] = \mathbf{d}[s']$, and
3. for every $s \in [\mathrm{ar}(R)]$, $\mathbf{t}[s] = t$ implies $\mathbf{d}[s] = d$.

Alternatively, $(t, R(\mathbf{t}))$ *can be mapped to* $(d, R'(\mathbf{d}))$ if $R = R'$ and there is a mapping $h : \{t\} \cup \{\mathbf{t}\} \to \{d\} \cup \{\mathbf{d}\}$ such that $h(t) = d$ and $h(\mathbf{t}) = \mathbf{d}$.

Assume that, additionally, $\mathbf{T}$ is an ftree. Let $\mathbf{T}_m$, $m \in M$ be the collection of all maximal subftrees of $\mathbf{T}$ satisfying the following conditions:

1. $\mathbf{T}_m$ does not contain $R(\mathbf{t})$
2. Every $\mathbf{T}_m$ contains an element t_m in $\{\mathbf{t}\}$ (which by condition (1) must be unique) and, additionally, t_m participates in only one constraint in $\mathbf{T}_m$.

Alternatively, we can regard collection $\mathbf{T}_m, m \in M$ as constructed in two stages from $\mathbf{T}$ as follows. In a first stage, we remove constraint $R(\mathbf{t})$ from $\mathbf{T}$ obtaining a collection of ftrees satisfying condition (1). Note that each one of the ftrees $\mathbf{X}$ obtained in this way contains a unique element x in $\{\mathbf{t}\}$ but x might appear in several constraints $C_1, \ldots, C_r$ of $\mathbf{X}$. Then, $\mathbf{X}$ is further divided in r ftrees $\mathbf{X}_1, \ldots, \mathbf{X}_r$ in the following way: $\mathbf{X}_i$, $i \in [r]$ is the substructure of $\mathbf{X}$ induced by all the elements that are still connected to x if we remove constraints $C_1, \ldots, C_{i-1}, C_{i+1}, \ldots, C_r$.

Then, we define $\mathbf{T} \setminus R(\mathbf{t})$ to be the collection $(\mathbf{T}_m, t_m)$, $m \in M$ of rooted ftrees obtained in this way. The following claim is immediate.

Claim 9.10. Assume that $t \in R(\mathbf{t})$. Then, $\hom(\mathbf{T}, t, R(\mathbf{t}); \mathbf{D}, d, R'(\mathbf{d})) = 0$ if $(t, R(\mathbf{t}))$ cannot be mapped to $(d, R'(\mathbf{d}))$. Otherwise,

$$\hom(\mathbf{T}, t, R(\mathbf{t}); \mathbf{D}, d, R'(\mathbf{d})) = \prod_{m \in M} \hom(\mathbf{T}_m, t_m; \mathbf{D}, \mathbf{d}[s_m])$$

where $s_m \in [\mathrm{ar}(R)]$ is such that $t_m = \mathbf{t}[s_m]$.

Then we have the following lemmas:

Lemma 9.11. *Let* $\mathbf{D}$ *be a* σ*-structure and let* $(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{Q}}) = (\{\mathcal{P}_i \mid i \in I\}, \{\mathcal{Q}_j \mid j \in J\})$ *be an equitable partition of* $\mathbf{D}$*. Then, for every* d_1, d_2 *in the same* $\boldsymbol{\mathcal{P}}$*-class and every rooted ftree* $(\mathbf{T}, t)$*,* $\hom(\mathbf{T}, t; \mathbf{D}, d_1) = \hom(\mathbf{T}, t; \mathbf{D}, d_2)$*.*

Proof. The proof follows easily by induction on the number of constraints of $\mathbf{T}$. Assume that $\mathbf{T}$ contains no constraints. In this case $\mathbf{T}$ consists only of node t and the claim is immediate. Otherwise, let $R(\mathbf{t})$ be any constraint in $\mathbf{T}$ containing t and let $(\mathbf{T}_m, \mathbf{t}[s_m])$, $m \in M$ be $\mathbf{T} \setminus R(\mathbf{t})$.

For every constraint $C \in \mathcal{C}_{\mathbf{D}}$ and every $d \in D$ we shall use $n_{d,C}$ to denote $\hom(\mathbf{T}, t, R(\mathbf{t}); \mathbf{D}, d, C)$. We claim that for every two constraints C_1, C_2 in the same $\boldsymbol{\mathcal{Q}}$-class satisfying $M_{\mathbf{D}}[d_1, C_1] = M_{\mathbf{D}}[d_2, C_2]$ we have $n_{d_1,C_1} = n_{d_2,C_2}$.

Let $C_1 = R_1(\mathbf{d}_1)$ and $C_2 = R_2(\mathbf{d}_2)$. Note that since $M_{\mathbf{D}}[d_1, C_1] = M_{\mathbf{D}}[d_2, C_2]$ we can assume that C_1 and C_2 share the same relation symbol. Let us do a case analysis using Claim 9.10:

- If $(t, R(\mathbf{t}))$ does not map to $(d_1, R_1(\mathbf{d}_1))$ then $n_{d_1,C_1} = 0$. Since C_2 belongs to the same $\boldsymbol{\mathcal{Q}}$-class and $M_{\mathbf{D}}[d_1, C_1] = M_{\mathbf{D}}[d_2, C_2]$ it follows that $(t, R(\mathbf{t}))$ does not map to $(d_2, R_2(\mathbf{d}_2))$ either, and hence $n_{d_2,C_2} = 0$.
- Otherwise $n_{d_q,C_q} = \prod_{m \in M} \hom(\mathbf{T}_m, t_m; \mathbf{D}, \mathbf{d}_q[s_m])$ $(q = 1, 2)$. Since C_1 and C_2 belong to the same $\boldsymbol{\mathcal{Q}}$-class, it follows that $\mathbf{d}_1[s]$ and $\mathbf{d}_2[s]$ belong to the same $\boldsymbol{\mathcal{P}}$-class for every $s \in [\mathrm{ar}(R)]$. It then follows by induction that $n_{d_1,C_1} = n_{d_2,C_2}$.

Then, for every $i \in I$, $j \in J$, and $\ell \in \mathcal{L}_\sigma$ there exists some unique $n^\ell_{i,j}$ such that $n_{d,C} = n^\ell_{i,j}$ for every $d \in \mathcal{P}_i$ and every $C \in \mathcal{Q}_j$ with $M_{\mathbf{D}}[d, C] = \ell$. Let $\mathcal{P}_i$ be the $\boldsymbol{\mathcal{P}}$-class of d_1 and d_2. To finalize the proof it is enough to see that for every $q \in \{1, 2\}$,

$$\hom(\mathbf{T}, t; \mathbf{D}, d_q) = \sum_{C \in \mathcal{C}_{\mathbf{D}}} \hom(\mathbf{T}, t, R(\mathbf{t}); \mathbf{D}, d_q, C) = \sum_{C \in \mathcal{C}_{\mathbf{D}}} n_{d_q,C} = \sum_{j \in J, \ell \in \mathcal{L}_\sigma} n^\ell_{i,j} c^\ell_{i,j}$$

where $c^\ell_{i,j}$ is as in condition (P1) of equitable partition. Hence, in the previous expression we use that $c^\ell_{i,j} = |\{C \in \mathcal{Q}_j \mid M_{\mathbf{D}}[d_q, C] = \ell\}|$ for $q \in \{1, 2\}$.

□

Lemma 9.12. *Let* $\mathbf{D}$ *be a* σ*-structure, let* $(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{Q}}) = (\{\mathcal{P}_i \mid i \in I\}, \{\mathcal{Q}_j \mid j \in J\})$ *defined as follows:*

1. $\boldsymbol{\mathcal{P}}$ *is the partition of* D *that places two elements* d_1, d_2 *in the same class if* $\hom(\mathbf{T}, t; \mathbf{D}, d_1) = \hom(\mathbf{T}, t; \mathbf{D}, d_2)$ *for every rooted ftree* $(\mathbf{T}, t)$*;*
2. $\boldsymbol{\mathcal{Q}}$ *is the partition of* $\mathcal{C}_{\mathbf{D}}$ *that places two constraints* $R_1(\boldsymbol{d}_1)$*,* $R_2(\boldsymbol{d}_2)$ *in the same class if the following conditions hold:*
   - $R_1 = R_2$*, and*
   - *for every* $s, s' \in [\mathrm{ar}(R_1)]$*,* $\boldsymbol{d}_1[s] = \boldsymbol{d}_1[s']$ *iff* $\boldsymbol{d}_2[s] = \boldsymbol{d}_2[s']$*, and*
   - *for every* $s \in [\mathrm{ar}(R_1)]$*,* $\boldsymbol{d}_1[s]$ *and* $\boldsymbol{d}_2[s]$ *belong to the same* $\boldsymbol{\mathcal{P}}$*-class.*

Then $(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{Q}})$ *is an equitable partition.*

Proof. We only need to prove that for every $i \in I$, every $j \in J$, and every $\ell \in \mathcal{L}_\sigma$ there exists $c^\ell_{i,j}$ such that $c^\ell_{d,j} = c^\ell_{i,j}$ for every $d \in \mathcal{P}_i$, where $c^\ell_{d,j} = |\{C \in \mathcal{Q}_j \mid M_{\mathbf{D}}[d, C] = \ell\}|$ (since the other condition of equitable partition follows immediately from the definition of $(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{Q}})$).

Let $\mathcal{Z}$ be the set of all $(\mathbf{T}, t, R(\mathbf{t}))$ where $\mathbf{T}$ is an ftree, $R(\mathbf{t}) \in \mathcal{C}_{\mathbf{T}}$, and $t \in \{\mathbf{t}\}$.

We shall start by proving that for every $i \in I$, $j \in J$, $\ell \in \mathcal{L}_\sigma$, and $z \in \mathcal{Z}$, there exists $n^{z,\ell}_{i,j}$ such that $n^z_{d,C} = n^{z,\ell}_{i,j}$ for every $d \in \mathcal{P}_i$ and every $C \in \mathcal{Q}_j$, where $n^z_{d,C}$ is defined to be $\hom(\mathbf{T}, t, R(\mathbf{t}); \mathbf{D}, d, C)$. That is, we want to prove the following claim.

Claim 9.13. For every d_1, d_2 inside the same class $\mathcal{P}_i$ and every C_1, C_2 inside the same class $\mathcal{Q}_j$ such that $M_{\mathbf{D}}[d_1, C_1] = M_{\mathbf{D}}[d_2, C_2]$ we have $n^z_{d_1,C_1} = n^z_{d_2,C_2}$.

Proof of Claim 9.13. Again, let us do a case analysis using Claim 9.10:

- If $(t, R(\mathbf{t}))$ cannot be mapped to (d_1, C_1) then since C_2 belongs to the same $\boldsymbol{\mathcal{Q}}$-class as C_1 and $M_{\mathbf{D}}[d_1, C_1] = M_{\mathbf{D}}[d_2, C_2]$ it follows that $(t, R(\mathbf{t}))$ cannot be mapped to (d_2, C_2) either. Then, we have $n^z_{d_1,C_1} = n^z_{d_2,C_2} = 0$.

- Assume that we are not in the previous case. Let $C_1 = R(\mathbf{d}_1)$ and $C_2 = R(\mathbf{d}_2)$ and let $(\mathbf{T}_m, \mathbf{t}[s_m])$, $m \in M$ be $\mathbf{T} \setminus R(\mathbf{t})$. Then, we have that for every $q \in \{1, 2\}$ $n^z_{d_q, C_q} = \prod_{m \in M} \hom(\mathbf{T}_m, \mathbf{t}[s_m]; \mathbf{D}, \mathbf{d}_q[s_m])$. Since C_1 and C_2 belong to the same $\boldsymbol{\mathcal{Q}}$-class, it follows that $\mathbf{d}_1[s]$ and $\mathbf{d}_2[s]$ belong to the same $\boldsymbol{\mathcal{P}}$-class for every $s \in [\mathrm{ar}(R)]$. It then follows from the definition of $(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{Q}})$ that $n^z_{d_1, C_1} = n^z_{d_2, C_2}$.

■

Then for every $i \in I$ and every $d \in \mathcal{P}_i$ we have

$$\begin{aligned} \hom(\mathbf{T}, t; \mathbf{D}, d) &= \sum_{C \in \mathcal{C}_{\mathbf{D}}} \hom(\mathbf{T}, t, R(\mathbf{t}); \mathbf{D}, d, C) \\ &= \sum_{j \in J, \ell \in \mathcal{L}_\sigma} n^{z,\ell}_{i,j} c^\ell_{d,j} = \sum_{(j,\ell) \in K_i} n^{z,\ell}_{i,j} c^\ell_{d,j} \end{aligned} \tag{9.2}$$

where $K_i = \{(j, \ell) \in J \times \mathcal{L}_\sigma \mid n^{z,\ell}_{i,j} \neq 0 \text{ for some } z \in \mathcal{Z}\}$. The following is immediate.

Claim 9.14. Let $i \in I$ and $d \in \mathcal{P}_i$. For every $(j, \ell) \in J \times \mathcal{L}_\sigma$, $(j, \ell) \in K_i$ iff there exists a constraint $C = R'(\mathbf{d})$ in $\mathcal{Q}_j$ with $M_{\mathbf{D}}[d, C] = \ell$ and $d \in \{\mathbf{d}\}$.

In what remains of the proof i is any arbitrary element of I. Then, it follows from Claim 9.14 that if $(j, \ell) \notin K_i$ we can safely set $c^\ell_{i,j} = 0$. It now remains to show the existence of $c^\ell_{i,j}$ for every $(j, \ell) \in K_i$. To this end we only need to observe that (9.2) holds for every $z \in \mathcal{Z}$ and invoke Claim 9.9(2) to conclude that $c^{d_1,\ell}_{i,j} = c^{d_2,\ell}_{i,j}$ for every $j \in J$,$\ell \in \mathcal{L}_\sigma$ and every $d_1, d_2 \in \mathcal{P}_i$.

However, we need first to guarantee that the hypothesis of the claim are met. First, we prove the following:

Claim 9.15. Let $(j_1, \ell_1), (j_2, \ell_2)$ be different elements in K_i. Then, there exists $z \in \mathcal{Z}$ such that $n^{z,\ell_1}_{i,j_1} \neq n^{z,\ell_2}_{i,j_2}$.

Proof of Claim 9.15. Let $d \in \mathcal{P}_i$, and let $C_q = R_q(\mathbf{d}_q), q \in \{1, 2\}$ be a constraint in $\mathcal{Q}_{j_q}$ with $M_{\mathbf{D}}[d, C_q] = \ell_q$ such that $d \in \{\mathbf{d}_q\}$. If (d, C_1) cannot be mapped to (d, C_2) then we just need to set z to be $(\mathbf{T}, d, C_1)$ where $\mathbf{T}$ is the ftree with universe $\{\mathbf{d}_1\}$ containing only constraint C_1. A similar construction takes care of the case when (d, C_2) cannot be mapped to (d, C_1). Hence, we are left with the case in which each of $(d, C_1), (d, C_2)$

can be mapped to each other, that is, where $(d, C_1), (d, C_2)$ are identical modulo renaming the elements. This implies that $\ell_1 = \ell_2$ and, hence, $j_1 \neq j_2$, which implies that that there exists $s \in [\mathrm{ar}(R_1)]$ such that $\mathbf{d}_1[s]$ and $\mathbf{d}_2[s]$ belong to a different $\boldsymbol{\mathcal{P}}$-class. Consequently, by the definition of $(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{Q}})$ we have that $\hom(\mathbf{T}, t; \mathbf{D}, \mathbf{d}_1[s]) \neq \hom(\mathbf{T}, t; \mathbf{D}, \mathbf{d}_2[s])$ for some rooted ftree $(\mathbf{T}, t)$. Now, let $z = (\mathbf{X}, d, R_1(\mathbf{d}_1))$ where $\mathbf{X}$ is the ftree obtained by adding constraint $R_1(\mathbf{d}_1)$ to $\mathbf{T}$ (we can assume, renaming variables if necessary that T does not contain any element in $\{\mathbf{d}_1\}$) and identifying t with $\mathbf{d}_1[s]$. It follows immediately that $n^{z,\ell_q}_{i,j_q} = \hom(\mathbf{X}, d, R_1(\mathbf{d}_1); \mathbf{D}, d, R_q(\mathbf{d}_q)) = \hom(\mathbf{T}, t; \mathbf{D}, \mathbf{d}_q[s])$ and we are done. ■

Denote by $\mathbf{n}^z$ the K_i-vector with entries $n^{z,\ell}_{i,j}$, $(j, \ell) \in K_i$.

Claim 9.16. Let $z_q = (\mathbf{T}_q, t_q, R_q(\mathbf{t}_q)) \in \mathcal{Z}$, $q = 1, 2$ such that $\mathbf{n}^{z_1} \cdot \mathbf{n}^{z_2} \neq \mathbf{0}$. Then there exists $z \in \mathcal{Z}$ such that $\mathbf{n}^z = \mathbf{n}^{z_1} \cdot \mathbf{n}^{z_2}$

Proof of Claim 9.16. Clearly if $R_1 \neq R_2$ then $n^{z_1,\ell}_{i,j} \cdot n^{z_2,\ell}_{i,j} = 0$ for every $(j, \ell) \in K_i$ and nothing needs to be done. Otherwise, let $z = (\mathbf{T}, t, R(\mathbf{t}))$ where $\mathbf{T}$ is obtained by first computing the disjoint union of $\mathbf{T}_1$ and $\mathbf{T}_2$ and then, for every $s \in [\mathrm{ar}(R)]$, merging $\mathbf{t}_1[s]$ and $\mathbf{t}_2[s]$ into a single node, as well as identifying t_1 and t_2 into a single element t. Note that after the identification $R_1(\mathbf{t}_1)$ and $R_2(\mathbf{t}_2)$ become the same constraint, which we denote by $R(\mathbf{t})$.

It only remains to see that $n^{z_1,\ell}_{i,j} \cdot n^{z_2,\ell}_{i,j} = n^{z,\ell}_{i,j}$ for every $(j, \ell) \in K_i$. Let $d \in \mathcal{P}_i$, and let $C \in \mathcal{Q}_j$ be such that $\ell = M_{\mathbf{D}}[d, C]$. Clearly, if $(t_1, R(\mathbf{t}_1))$ cannot be mapped to (d, C) then $n^{z_1,\ell}_{i,j} = 0$. Note that in this case $(t, R(\mathbf{t}))$ cannot be mapped to (d, C) either, and hence $n^{z,\ell}_{i,j} = 0$ as desired. A similar reasoning applies if $(t_2, R(\mathbf{t}_2))$ cannot be mapped to (d, C) and, hence, we only need to deal with the case in which both $(t_1, R(\mathbf{t}_1))$ and $(t_2, R(\mathbf{t}_2))$ can be mapped to (d, C). In this case it follows that $(t, R(\mathbf{t}))$ can be mapped to (d, C). Let $(\mathbf{T}_m, \mathbf{t}[s_m])$, $m \in M$ be $\mathbf{T} \setminus R(\mathbf{t})$ and let $C = R(\mathbf{d})$. Then

$$n^z_{d,C} = \hom(\mathbf{T}, t, R(\mathbf{t}); \mathbf{D}, d, R(\mathbf{d})) = \prod_{m \in M} \hom(\mathbf{T}_m, \mathbf{t}[s_m]; \mathbf{D}, \mathbf{d}[s_m])$$

Note that M can be partitioned in two sets M_1, M_2 such that for every $q \in \{1, 2\}$ $\mathbf{T}_q \setminus R(\mathbf{t}_q)$ is precisely $(\mathbf{T}_m, \mathbf{t}[s_m])$, $m \in M_q$.

It follows that

$$\begin{aligned} n^{z,\ell}_{i,j} = n^{z}_{d,C} &= \prod_{m\in M} \hom(\mathbf{T}_m, \mathbf{t}[s_m]; \mathbf{D}, \mathbf{d}[s_m]) \\ &= \prod_{m\in M_1} \hom(\mathbf{T}_m, \mathbf{t}[s_m]; \mathbf{D}, \mathbf{d}[s_m]) \cdot \prod_{m\in M_2} \hom(\mathbf{T}_m, \mathbf{t}[s_m]; \mathbf{D}, \mathbf{d}[s_m]) \\ &= \hom(\mathbf{T}_1, t_1, R(\mathbf{t}_1); \mathbf{D}, d, R(\mathbf{d})) \cdot \hom(\mathbf{T}_2, t_2, R(\mathbf{t}_2); \mathbf{D}, d, R(\mathbf{d})) \\ &= n^{z_1}_{d,C} \cdot n^{z_2}_{d,C} = n^{z_1,\ell}_{i,j} \cdot n^{z_2,\ell}_{i,j} \end{aligned}$$

as desired. ■

□

We are finally ready to give a proof of the equivalence of (3) and (4).

(3) ⇒ (4). Let $\mathbf{D}$ denote the disjoint union of $\mathbf{A}$ and $\mathbf{B}$, let $(\{\mathcal{P}_i \mid i \in I\}, \{\mathcal{Q}_j \mid j \in J\})$ be an equitable partition of $\mathbf{D}$ witnessing that $\mathbf{A}$ and $\mathbf{B}$ have a common equitable partition, and let $\mathbf{T}$ be an ftree and $t \in T$. It follows from Lemma 9.11 that for every $i \in I$ there is a value n_i such that for every $d \in \mathcal{P}_i$, $\hom(\mathbf{T}, t; \mathbf{D}, d) = n_i$.

Then, for every $\mathbf{E} \in \{\mathbf{A}, \mathbf{B}\}$,

$$\hom(\mathbf{T}; \mathbf{E}) = \sum_{e\in E} \hom(\mathbf{T}, t; \mathbf{E}, e) = \sum_{i\in I} n_i \cdot p^{\mathbf{E}}_i$$

where $p^{\mathbf{E}}_i = |\mathcal{P}_i \cap E|$. Since $(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{Q}})$ is an equitable partition of $\mathbf{A} \cup \mathbf{B}$, it follows that $p^{\mathbf{A}}_i = p^{\mathbf{B}}_i$ for every $i \in I$ and we are done.

(4) ⇒ (3). Let $\mathbf{D}$ be the disjoint union of $\mathbf{A}$ and $\mathbf{B}$ and let $(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{Q}}) = (\{\mathcal{P}_i \mid i \in I\}, \{\mathcal{Q}_j \mid j \in J\})$ be the partition of $\mathbf{D}$ defined as in Lemma 9.12. We first show that $p^{\mathbf{A}}_i = p^{\mathbf{B}}_i$ for every $i \in I$, where $p^{\mathbf{E}}_i = |\mathcal{P}_i \cap E|$ for $\mathbf{E} \in \{\mathbf{A}, \mathbf{B}\}$.

Let $z = (\mathbf{T}, t)$ be any rooted ftree. Note that from the definition of $(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{Q}})$ it follows that for every $i \in I$ there exists n^z_i such that $n^z_i = \hom(\mathbf{T}, t; \mathbf{D}, d)$ for every $d \in \mathcal{P}_i$. Consequently, for $\mathbf{E} \in \{\mathbf{A}, \mathbf{B}\}$ we have:

$$\hom(\mathbf{T}; \mathbf{E}) = \sum_{e\in E} \hom(\mathbf{T}, t; \mathbf{E}, e) = \sum_{i\in I} n^z_i \cdot p^{\mathbf{E}}_i$$

Since the previous identity holds for every rooted ftree z, it is only necessary to invoke Claim 9.9 to conclude that $p^{\mathbf{A}}_i = p^{\mathbf{B}}_i$ for every $i \in I$. However, we must first verify that the conditions of Claim 9.9 are satisfied.

First, we need to show that for every $i \neq i' \in I$, there exists a rooted ftree z such that $n_i^z \neq n_{i'}^z$. This follows immediately from the definition of $\boldsymbol{\mathcal{P}}$. Secondly, we need to show that for every pair of rooted ftrees $z_1 = (\mathbf{T}_1, t_1)$, $z_2 = (\mathbf{T}_2, t_2)$, there exists a rooted ftree $z = (\mathbf{T}, t)$ such that $n_i^z = n_i^{z_1} \cdot n_i^{z_2}$. Indeed, it is easy to verify that the condition is satisfied if we let $\mathbf{T}$ be the rooted ftree obtained by merging t_1 and t_2 into a single node t in the disjoint union of $\mathbf{T}_1$ and $\mathbf{T}_2$.

9.5 Proof of Theorem 9.7

Theorem 9.7. *Let r be the maximum arity among all relations in σ and assume that $r \leq k$. Then for every pair of structures $\mathbf{A}$, $\mathbf{B}$ the following are equivalent:*

1. $\mathbf{A} \equiv_k \mathbf{B}$*;*
2. $\hom(\mathbf{X}; \mathbf{A}) = \hom(\mathbf{X}; \mathbf{B})$ *for every σ-structure $\mathbf{X}$ of treewidth $< k$;*
3. $\mathbf{A}$ *and* $\mathbf{B}$ *satisfy the same formulae in the logic $\mathscr{C}^k$.*

For the equivalence of (1) and (2), we use Theorem 5.4. In particular, we have that for a pair of σ-structures $\mathbf{A}$ and $\mathbf{B}$, $\mathbf{A}_k^* \equiv_1 \mathbf{B}_k^*$ if and only if $\hom(\mathbf{T}; \mathbf{A}_k^*) = \hom(\mathbf{T}; \mathbf{B}_k^*)$ for every σ_k^*-ftree $\mathbf{T}$. So to show that (1) $\Leftrightarrow$ (2) it only remains to prove the following.

Claim 9.17. Assume that $r \leq k$. Then the following are equivalent:

2. $\hom(\mathbf{X}; \mathbf{A}) = \hom(\mathbf{X}; \mathbf{B})$ for every σ-structure $\mathbf{X}$ of treewidth $< k$;
4. $\hom(\mathbf{T}; \mathbf{A}_k^*) = \hom(\mathbf{T}; \mathbf{B}_k^*)$ for every σ_k^*-ftree $\mathbf{T}$.

Proof of Claim 9.17. 2 $\Rightarrow$ 4. Let $\mathbf{T}$ be a σ_k^*-ftree. It follows immediately that if 2 holds then both $\mathbf{A}$ and $\mathbf{B}$ must have the same number of elements and constraints. It then follows that 4 holds for $\mathbf{T}$ if $\mathbf{T}$ consists of a single element and no constraints at all. Consequently we can safely assume that all elements in $\mathbf{T}$ participate in at least one constraint.

In what follows $\mathbf{D}_k^* \in \{\mathbf{A}_k^*, \mathbf{B}_k^*\}$. Let t be any node in T. Since t participates in a constraint it follows that the possible image of t under a

homomorphism from $\mathbf{T}$ is heavily restricted. In particular, if the image of t according to some homomorphism from $\mathbf{T}$ to $\mathbf{D}_k^*$ is in D^j for some $j \leq k$ then necessarily the image of t under any homomorphism from $\mathbf{T}$ to any structure $\mathbf{C}_k^* \in \{\mathbf{A}_k^*, \mathbf{B}_k^*\}$ must be in C^j. This means that we can safely add constraint $T_{j,\emptyset}(t)$ to $\mathbf{T}$ without altering $\hom(\mathbf{T}; \mathbf{A}_k^*)$ or $\hom(\mathbf{T}; \mathbf{B}_k^*)$.

Likewise, if some homomorphism from $\mathbf{T}$ to a structure $\mathbf{D}_k^*$ maps t to a constraint $R(\mathbf{d})$, then we can assume that constraint $R_\emptyset(t)$ belongs to $\mathbf{T}$.

To complete the proof we shall show that it is always possible to construct from $\mathbf{T}$ a σ-structure $\mathbf{X}$ of treewidth $< k$ such that $\hom(\mathbf{T}; \mathbf{D}_k^*) = \hom(\mathbf{X}; \mathbf{D})$. It is convenient to construct $\mathbf{X}$ in two stages. First, let us construct a σ-structure $\mathbf{Y}$ (not necessarily of treewidth $< k$) satisfying that $\hom(\mathbf{T}; \mathbf{D}_k^*) = \hom(\mathbf{Y}; \mathbf{D})$. We shall allow to use equalities in $\mathbf{Y}$, i.e., constraints of the form $y_1 = y_2$, indicating that y_1 and y_2 must be assigned to the same element in D.

We shall define $\mathbf{Y}$ along with a function α mapping every element t of $\mathbf{T}$ to a j-ary tuple of elements in $\mathbf{Y}$ ($j \leq k$) inductively on the number of elements of $\mathbf{T}$ as follows.

Assume (base case) that $\mathbf{T}$ contains a unique element t. As discussed above we can assume that $\mathbf{T}$ contains constraint $T_{j,\emptyset}(t)$ for some $j \leq k$ or $R_\emptyset(t)$ for some $R \in \sigma$. In the first case, we set the universe of $\mathbf{Y}$ to contain j new elements $y_1, \ldots, y_j$. Furthermore, for every unary constraint $T_{j,S}(t)$ in $\mathbf{T}$ and every $i, i' \in S$, we include in $\mathbf{Y}$ the equality $y_i = y_{i'}$, and we define $\alpha(t) = (y_1, \ldots, y_j)$. In the second case, we set the universe of $\mathbf{Y}$ to contain $\mathrm{ar}(R)$ new elements $y_1 \ldots, y_{\mathrm{ar}(R)}$ and we include in $\mathbf{Y}$ the constraint $R(y_1, ..., y_{\mathrm{ar}(R)})$. Similarly to the previous case, for every unary constraint $R_S(t)$ in $\mathbf{T}$ and every $i, i' \in S$, we include in $\mathbf{Y}$ the equality $y_i = y_{i'}$. Finally, we set $\alpha(t) = (y_1, \ldots, y_{\mathrm{ar}(R)})$.

Let us consider now the inductive case. Let t_1 and t_2 be nodes that participate in a binary constraint $U(t_1, t_2)$ (recall that U is either $T_{j,\mathbf{i}}$ or $R_\mathbf{i}$) in $\mathbf{T}$. By removing this constraint $\mathbf{T}$ gets divided in two ftrees $\mathbf{T}_1$ and $\mathbf{T}_2$ such that $\mathbf{T}_1$ contains t_1 and $\mathbf{T}_2$ contains t_2. Now, assume that $\mathbf{Y}_i$ and α_i are already constructed for $\mathbf{T}_i$, $i = 1, 2$. We are ready to define $\mathbf{Y}$. First, we compute the disjoint union of $\mathbf{Y}_1$ and $\mathbf{Y}_2$. Then, we add some further equalities depending on constraint $U(t_1, t_2)$. Consider first the case that $U = T_{j_1,\mathbf{i}}$, $\mathbf{i} = (i_1, \ldots, i_{j_2}) \in [j_1]^{j_2}$ for some $j_1, j_2 \leq k$ and

let $\alpha_i(t_i) = (y^i_1, \ldots, y^i_{j_i})$, $i = 1, 2$. Then, for every $\ell \leq j_2$ we add the equality $y^2_\ell = y^1_{i_\ell}$. Finally, for every $t \in T$ we define $\alpha(t)$ to be $\alpha_i(t)$ where $\mathbf{T}_i$ contains t. The procedure is identical for $U = R_{\mathbf{i}}$, where we just substitute j_1 by $\mathrm{ar}(R)$. It follows immediately from the definition that $\hom(\mathbf{Y}; \mathbf{D}) = \hom(\mathbf{T}; \mathbf{D}^*_k)$.

Finally, let us define $\mathbf{X}$ to be the structure obtained by identifying (i.e, merging into a single element) all elements in $\mathbf{Y}$ joined by a chain of equalities. It is immediate that $\hom(\mathbf{Y}; \mathbf{D}) = \hom(\mathbf{X}; \mathbf{D})$.

We shall conclude by giving a tree-decomposition (G, β) of $\mathbf{X}$ of width $< k$. In particular, let G be the tree where the vertex set is precisely the universe of $\mathbf{T}$ and two different nodes are adjacent if both participate in some common constraint in $\mathbf{T}$ and let $\beta(t) = \{\alpha(t)\}$.

4 $\Rightarrow$ 2. Let $\mathbf{X}$ be a σ-structure of treewidth $< k$ and let $\mathbf{D} \in \{\mathbf{A}, \mathbf{B}\}$. We note here that we can assume that $\mathbf{X}$ is connected since if $\mathbf{X}$ is the disjoint union of structures $\mathbf{X}_1$ and $\mathbf{X}_2$ then $\hom(\mathbf{X}; \mathbf{D}) = \hom(\mathbf{X}_1; \mathbf{D}) \cdot \hom(\mathbf{X}_2; \mathbf{D})$. We shall show that there exists a σ^*_k-ftree $\mathbf{T}$ such that $\hom(\mathbf{T}; \mathbf{D}^*_k) = \hom(\mathbf{X}; \mathbf{D})$. Let (G, β) be a tree-decomposition of width at most k of $\mathbf{X}$. It is well-known and easy to prove that, since $\mathbf{X}$ is connected, we can always construct (G, β) in such a way that for every pair u, v of adjacent nodes in G, $\beta(u) \subseteq \beta(v)$ or $\beta(v) \subseteq \beta(u)$. Furthermore, it is easy to see that we can further enforce that for every constraint $R(\mathbf{x})$ in $\mathbf{X}$ there exists a node $v \in G$ such that $\beta(v) = \{\mathbf{x}\}$.

The universe T of $\mathbf{T}$ is $V \cup \mathcal{C}_{\mathbf{X}}$ where V is the node-set of G. Furthermore $\mathbf{T}$ contains the following constraints.

Let us start with the unary constraints. Let t be an element in $\mathbf{T}$. If $t = v \in G$ then we include in $\mathbf{T}$ a constraint $T_{j_v,\emptyset}(t)$ where $j_v = |\beta(v)|$. Otherwise, if $t = R(\mathbf{x}) \in \mathcal{C}_{\mathbf{X}}$ then we include in $\mathbf{T}$ all constraints $R_{\{i,i'\}}(t)$ where $\mathbf{x}[i] = \mathbf{x}[i']$.

Now, let us turn our attention to the binary constraints. Fix some arbitrary ordering on X and for every $v \in V$ let $\mathbf{x}^v = (x^v_1, \ldots, x^v_{j_v})$ be an array containing the nodes in $\beta(v)$ following this fixed order.

Then, for every edge (u, v) in G include constraint $T_{j_u,\mathbf{i}}(u, v)$ where $\mathbf{i} = (i_1, \ldots, i_{j_v})$ is defined as follows. First, we assume without loss of generality that $\beta(v) \subseteq \beta(u)$. Then, for every $\ell \leq j_v$, i_ℓ is defined to be such that $\mathbf{x}^u[i_\ell] = \mathbf{x}^v[\ell]$.

Finally, for every constraint $t = R(\mathbf{x})$ in $\mathbf{X}$ we pick some element $v \in V$ satisfying $\{\mathbf{x}\} = \beta(v)$ and we add the constraint $R_{\mathbf{i}}(t, v)$ with $\mathbf{i} = (i_1, \ldots, i_{j_v})$ where i_ℓ satisfies $\mathbf{x}[i_\ell] = \mathbf{x}^v[\ell]$. It is immediate to see that $\mathbf{T}$ is an ftree and that $\hom(\mathbf{X}; \mathbf{D}) = \hom(\mathbf{T}; \mathbf{D}^*_k)$ ■

For the equivalence of (1) and (3) , we will need to introduce a simple combinatorial game that will help us characterize equivalence in counting logic.

The *bijective k-pebble game* is played by two players, Spoiler and Duplicator, by placing k pairs of pebbles on a pair of structures $\mathbf{A}$, $\mathbf{B}$ of the same size. We shall denote each pair of pebbles by (x_i, y_i), where x_i belongs to Spoiler and y_i belongs to Duplicator. At every round, Spoiler picks up a pebble x_i, and Duplicator picks up the corresponding pebble y_i. At this point, Duplicator chooses a bijection f between A and B. Then, Spoiler places a pebble x_i on an element $a \in A$, and Duplicator must place the corresponding pebble y_i on $f(a) \in B$.

Duplicator wins a round of the bijective k-pebble game if the partial map defined by $x_i \mapsto y_i$ (i.e., where the element of A under pebble x_i is mapped to the element of B under pebble y_i) is a partial isomorphism between $\mathbf{A}$ and $\mathbf{B}$. Otherwise, Spoiler wins the round. We say that Duplicator has a *winning strategy* for the bijective k-pebble game if she has a strategy to win every round of the game (note that the game has infinitely many rounds).

More formally, for a j-tuple $\mathbf{d} = (d_1, \ldots, d_j)$, an element d, and $i \in [j+1]$, define $\mathbf{d}^i_d = (d_1, \ldots, d_{i-1}, d, d_i, \ldots, d_j)$. Additionally, for $j \in [k]$ and $i \in [j]$, define $\mathbf{i}(j, i) = (1, \ldots, i-1, i+1, \ldots, j)$. Then, a winning strategy for the bijective k-pebble game is a non-empty set $W \subseteq \cup_{0 \leq j \leq k}(A^j \times B^j)$ such that, for every $\mathbf{a} = (a_1, \ldots, a_j)$ and $\mathbf{b} = (b_1, \ldots, b_j)$ with $(\mathbf{a}, \mathbf{b}) \in W$, the following conditions hold:

1. The partial map given by $a_i \mapsto b_i$, $i \in [j]$ is a partial isomorphism between $\mathbf{A}$ and $\mathbf{B}$;
2. If $j \geq 1$, then $(\pi_{\mathbf{i}(j,i)}\mathbf{a}, \pi_{\mathbf{i}(j,i)}\mathbf{b}) \in W$ for every $i \in [j]$;
3. If $j < k$, there exists a bijection $f : A \to B$ such that $(\mathbf{a}^i_a, \mathbf{b}^i_{f(a)}) \in W$ for every $a \in A$ and every $i \in [j+1]$.

Before continuing, we remark that when we talk about a partial isomorphism, we mean an isomorphism between the *vertex-induced* substructures of $\mathbf{A}$ and $\mathbf{B}$ on the universes defined by the pebble sets. In particular then, any relation of arity larger than k would not be accounted for by a partial isomorphism between substructures of size at most k, hence why we restrict our scope to structures with relations of smaller arity.

We have the following theorem of Hella (see also [GO15, ADW17]).

Theorem 9.18 ([Hel96])**.** $\mathbf{A}$ *and* $\mathbf{B}$ *satisfy the same formulae in the logic* $\mathscr{C}^k$ *if and only if Duplicator has a winning strategy for the bijective* k*-pebble game on* $\mathbf{A}$, $\mathbf{B}$.

Therefore, in order to prove (1) $\Leftrightarrow$ (3) in Theorem 9.7, it is sufficient to show that the following claim holds.

Claim 9.19. Assume that $r \leq k$. Then the following are equivalent:

1. $\mathbf{A} \equiv_k \mathbf{B}$;

5. Duplicator has a winning strategy for the bijective k-pebble game on $\mathbf{A}$, $\mathbf{B}$.

Proof of Claim 9.19. In what follows, we will assume without loss of generality that A and B are disjoint, so when we talk about the union of $\mathbf{A}$ and $\mathbf{B}$, we will always mean the *disjoint* union.

(1) $\Rightarrow$ (5). It will be convenient to assume that A_k^* and B_k^* additionally include a tuple of null arity each (this will correspond to the configuration where no pebbles are placed yet). Let $(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{Q}})$ be an equitable partition of $\mathbf{A}_k^* \cup \mathbf{B}_k^*$. A winning strategy W for Duplicator can be obtained as the set of all pairs $(\mathbf{a}, \mathbf{b}) \in A_k^* \times B_k^*$ such that $\mathbf{a}$ and $\mathbf{b}$ belong to the same class of $\boldsymbol{\mathcal{P}}$.

To see that W is a winning strategy, first observe that, by the definition of the transformation $*_k$, any two tuples of $\mathbf{A}_k^* \cup \mathbf{B}_k^*$ in the same partition class of $\boldsymbol{\mathcal{P}}$ must have the same isomorphism type in the original structure $\mathbf{A} \cup \mathbf{B}$. It follows that for all elements $(\mathbf{a}, \mathbf{b}) \in W$, the map $a_i \mapsto b_i$ is a partial isomorphism from $\mathbf{A}$ to $\mathbf{B}$, hence W satisfies condition (1).

Second, it is easy to see that W satisfies condition (2), since for each $j \in [k]$ and each $\mathbf{d} \in A^j \cup B^j$, $\mathbf{d}$ has a unique out-neighbour of type $T_{j,\mathbf{i}(j,i)}$,

which is precisely $\pi_{\mathbf{i}(j,i)}\mathbf{d}$. Therefore, for every $(\mathbf{a},\mathbf{b}) \in W$, $\pi_{\mathbf{i}(j,i)}\mathbf{a}$ and $\pi_{\mathbf{i}(j,i)}\mathbf{b}$ must belong to the same class of $\boldsymbol{\mathcal{P}}$ and hence $(\pi_{\mathbf{i}(j,i)}\mathbf{a}, \pi_{\mathbf{i}(j,i)}\mathbf{b}) \in W$.

Finally, to see that W satisfies condition (3), for each $(\mathbf{a},\mathbf{b}) \in W$ of arity $j < k$ we need to find a bijection $f : A \to B$ in such a way that for every $a \in A$ and every $i \in [j+1]$, the tuples $\mathbf{a}^i_a$ and $\mathbf{b}^i_{f(a)}$ belong to the same partition class of $\boldsymbol{\mathcal{P}}$. Since $\mathbf{a}$ and $\mathbf{b}$ belong to the same class of $\boldsymbol{\mathcal{P}}$ and $(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{Q}})$ is equitable, there is a one-to-one correspondence that respects the partition between the constraints that $\mathbf{a}$ and $\mathbf{b}$ participate in; in particular, this holds for the constraints of type $T_{j+1,\mathbf{i}(j+1,i)}$. Since for each $a \in A$ and $b \in B$, $T_{j+1,\mathbf{i}(j+1,i)}(\mathbf{a}^i_a, \mathbf{a}) \in \mathcal{C}_{\mathbf{A}^*_k}$ and $T_{j+1,\mathbf{i}(j+1,i)}(\mathbf{b}^i_b, \mathbf{b}) \in \mathcal{C}_{\mathbf{B}^*_k}$, there is a one-to-one partition-preserving correspondence between tuples $\mathbf{a}^i_a \in A^{j+1}$, $\mathbf{b}^i_b \in B^{j+1}$ that induces an appropriate bijection between A and B. In particular, the unary constraints of $\mathbf{A}^*_k$, $\mathbf{B}^*_k$ of the form $T_{j+1,S}$, $S \subseteq [j+1]$ imply that $f(a_i) = b_i$ for each $i \in [j]$.

(5) $\Rightarrow$ (1). We begin with the simple observation that, since $\mathrm{ar}(R) \leq k$ for all $R \in \sigma$, the transformation $*_k$ can be substantially simplified without its basic properties being affected (particularly, Lemma 9.4 and the definition of $\equiv_k$ remain unchanged). Specifically, this simplification consists of the following changes:

1. For each $R \in \sigma$, identify $R(\mathbf{a})$ with $\mathbf{a}$, so the universe of $\mathbf{A}^*_k$ is just $\cup_{j\leq k} A^j$;

2. Remove from σ^*_k all binary relation symbols of the form $R_{\mathbf{i}}$ (since for all $\mathbf{i} \in [\mathrm{ar}(R)]^j$ and all $\mathbf{a} \in R^{\mathbf{A}}$, $R_{\mathbf{i}}(\mathbf{a}, \pi_{\mathbf{i}}\mathbf{a})$ is equivalent to $T_{\mathrm{ar}(R),\mathbf{i}}(\mathbf{a}, \pi_{\mathbf{i}}\mathbf{a}) \wedge R_{\emptyset}(\mathbf{a})$);

3. Remove from σ^*_k all unary relation symbols of the form R_S for all $S \neq \emptyset$ (since for all $S \subseteq [\mathrm{ar}(R)]$ and all $\mathbf{a} \in R^{\mathbf{A}}$, $R_S(\mathbf{a}) = T_{\mathrm{ar}(R),S}(\mathbf{a}) \wedge R_{\emptyset}(\mathbf{a})$);

4. Remove from σ^*_k all binary relation symbols of the form $T_{j,\mathbf{i}}$ whenever $|[j] \setminus \{\mathbf{i}\}| > 1$. This is because, if $|[j] \setminus \{\mathbf{i}\}| > 1$, then there exists a sequence $\mathbf{i}_1 \in [j_0]^{j_1}$, $\mathbf{i}_2 \in [j_1]^{j_2}$, $\ldots$, $\mathbf{i}_m \in [j_{m-1}]^{j_m}$ with $j_0 = j$ and $m \geq 2$ such that $|[j_{i-1}] \setminus \{\mathbf{i}_i\}| \leq 1$ for each $i \in [m]$, and

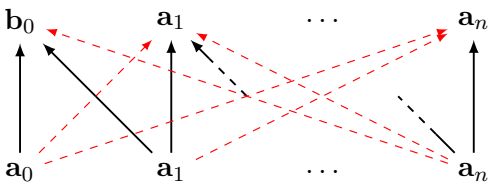

Figure 9.3: A representation of the closure of $\overline{W}$ under a chain of length $2n+1$ in W. Pairs in W are represented as solid black arrows and the added pairs in $\overline{W}$ are represented as dashed red arrows.

$\pi_{\mathbf{i}_m} \dots \pi_{\mathbf{i}_2} \pi_{\mathbf{i}_1} = \pi_{\mathbf{i}}$. Then, we have that

$$T_{j,\mathbf{i}}(\mathbf{a}, \pi_{\mathbf{i}}\mathbf{a}) = T_{j,\mathbf{i}_1}(\mathbf{a}, \pi_{\mathbf{i}_1}\mathbf{a}) \wedge T_{j_1,\mathbf{i}_2}(\pi_{\mathbf{i}_1}\mathbf{a}, \pi_{\mathbf{i}_2}\pi_{\mathbf{i}_1}\mathbf{a}) \wedge \dots$$
$$\dots \wedge T_{j_{m-1},\mathbf{i}_m}(\pi_{\mathbf{i}_{m-1}} \dots \pi_{\mathbf{i}_1}\mathbf{a}, \pi_{\mathbf{i}_m}\pi_{\mathbf{i}_{m-1}} \dots \pi_{\mathbf{i}_1}\mathbf{a}).$$

We are now in the position to prove the backwards direction of Claim 9.19.

Assume that Duplicator has a winning strategy for the bijective k-pebble game. We need to find an equitable partition of $\mathbf{A}_k^* \cup \mathbf{B}_k^*$ such that each class of the partition has the same number of elements from A_k^* and from B_k^*. Since $\mathbf{A}_k^* \cup \mathbf{B}_k^*$ is a (vertex- and edge-coloured) digraph, it is enough to define a partition $\boldsymbol{\mathcal{P}} = \{\mathcal{P}_i \mid i \in I\}$ of $A_k^* \cup B_k^*$ where any two elements in the same partition class of $\boldsymbol{\mathcal{P}}$ are subject to the same unary constraints, and have the same number of in- and out-neighbours in any other class of $\boldsymbol{\mathcal{P}}$ connected by an edge of any given colour.

Let W be a winning strategy for Duplicator. We define the set $\overline{W}$ to contain precisely all the pairs $(\mathbf{a}, \mathbf{b}) \in \cup_{j \leq k} A^j \times B^j$ such that there exist $\mathbf{a}_0, \mathbf{a}_1, \dots, \mathbf{a}_n \in A^j$ and $\mathbf{b}_0, \mathbf{b}_1, \dots, \mathbf{b}_n \in B^j$ such that $\mathbf{a}_0 = \mathbf{a}$, $\mathbf{b}_n = \mathbf{b}$, $(\mathbf{a}_0, \mathbf{b}_0) \in W$, and for each $i \in [n]$, the pairs $(\mathbf{a}_i, \mathbf{b}_{i-1})$ and $(\mathbf{a}_i, \mathbf{b}_i)$ both belong to W. That is, we think of $\overline{W}$ as the closure of W under odd chains (see Figure 9.3).

We shall show that $\overline{W}$ is also a winning strategy for the bijective k-pebble game over $\mathbf{A}$, $\mathbf{B}$. Conditions (1) and (2) are easily verified by using the corresponding conditions for W, plus the fact that isomorphisms compose. For (3), let $(\mathbf{a}, \mathbf{b}) \in \overline{W}$ have arity j and let $\mathbf{a}_0, \mathbf{b}_0, \dots, \mathbf{a}_n, \mathbf{b}_n$ be the odd chain witnessing this. Let $f_{i,i}$, $f_{i+1,i}$ be the bijections given by condition (3) of W for $(\mathbf{a}_i, \mathbf{b}_i)$, $(\mathbf{a}_{i+1}, \mathbf{b}_i)$ respectively. We claim that

$f := f_{n,n} \circ f_{n,n-1}^{-1} \circ \ldots \circ f_{1,1} \circ f_{1,0}^{-1} \circ f_{0,0}$ witnesses that (3) is satisfied for $(\mathbf{a}_0, \mathbf{b}_n) = (\mathbf{a}, \mathbf{b})$ in $\overline{W}$. Recall that for a j-tuple $\mathbf{d}$, an element d, and $l \in [j+1]$, we denote by $\mathbf{d}_d^l$ the tuple obtained from $\mathbf{d}$ by adding d to $\mathbf{d}$ before the l^{th} coordinate. By condition (3) for W, we have that for every $l \in [j]$, every $i = 0, \ldots, n$, and every $\alpha_i \in A$,

$$((\mathbf{a}_i)^l_{\alpha_i}, (\mathbf{b}_i)^l_{f_{i,i}(\alpha_i)}) \in W,$$

and similarly for every $l \in [j]$, every $i = 0, \ldots, n-1$, and every $\beta_i \in A$,

$$((\mathbf{a}_{i+1})^l_{f^{-1}_{i+1,i}(\beta_i)}, (\mathbf{b}_i)^l_{\beta_i}) \in W.$$

Therefore, if for $i = 0, \ldots, n-1$ we choose

$$\beta_i = f_{i,i}(\alpha_i), \quad \alpha_{i+1} = f^{-1}_{i+1,i}(\beta_i)$$

we get that for every choice of $\alpha_0 \in A$, $((\mathbf{a})^l_{\alpha_0}, (\mathbf{b})^l_{f(\alpha_0)}) \in \overline{W}$ as required. Hence, $\overline{W}$ is a winning strategy.

The partition $\boldsymbol{\mathcal{P}}$ of $A_k^* \cup B_k^*$ is defined by the transitive closure of $\overline{W}$. That is, $\mathbf{a} \in A^j$ and $\mathbf{b} \in B^j$ are in the same set of $\boldsymbol{\mathcal{P}}$ if $(\mathbf{a}, \mathbf{b}) \in \overline{W}$; any two $\mathbf{a}, \mathbf{a}' \in A^j$ are in the same set of $\boldsymbol{\mathcal{P}}$ if there exists some $\mathbf{b}$ such that $(\mathbf{a}, \mathbf{b}), (\mathbf{a}', \mathbf{b}) \in \overline{W}$; and similarly for any two $\mathbf{b}, \mathbf{b}' \in B^j$. In particular, there is a partial isomorphism between any two tuples $\mathbf{d}, \mathbf{d}'$ in the same class of $\boldsymbol{\mathcal{P}}$. Recall that, from item (1) in the observation above, we do not need to worry about defining a partition of $\mathcal{C}_{\mathbf{A}} \cup \mathcal{C}_{\mathbf{B}}$.

Since Duplicator has a winning strategy for the game on $\mathbf{A}$, $\mathbf{B}$, and since the game is completely symmetric (i.e., Duplicator would still win if we inverted $\mathbf{A}$ and $\mathbf{B}$ so that Spoiler places pebbles on B and Duplicator on A), it is easy to see that the condition $|\mathcal{P}_l \cap A_k^*| = |\mathcal{P}_l \cap B_k^*|$ is satisfied for each class $\mathcal{P}_l$ of the partition. It remains to show that the $\boldsymbol{\mathcal{P}}$ is equitable.

It is easily seen that any two tuples in the same class of $\boldsymbol{\mathcal{P}}$ are subject to the same unary constraints. In particular, if there is a partial isomorphism between $\mathbf{d}$ and $\mathbf{d}'$, then they are subject to the same relations in $\mathbf{A} \cup \mathbf{B}$ (accounting for unary constraints of the type $R_\emptyset$, $R \in \sigma$) and they have the same repetition structure (accounting for unary constraints of the type $T_{j,S}$ for $j \leq k$, $S \subseteq [j]$).

Now we are left to deal with the binary constraints of type $T_{j,\mathbf{i}}$ for $j, j' \leq k$ and $\mathbf{i} \in [j]^{j'}$. It is easy to see that any two tuples of arity j in

the same class of $\boldsymbol{\mathcal{P}}$ have the same number of outgoing edges of type $T_{j,\mathbf{i}}$ that are incident to each other class of $\boldsymbol{\mathcal{P}}$, since for any $j, j' \in k$, any two isomorphic tuples $\mathbf{d}, \mathbf{d}' \in A^j \cup B^j$, and any $\mathbf{i} \in [j]^{j'}$, $\pi_{\mathbf{i}}\mathbf{d}$ and $\pi_{\mathbf{i}}\mathbf{d}'$ will still be isomorphic (equivalently, any winning strategy for the bijective k-pebble game is trivially closed under removing pairs of pebbles, under adding pairs of pebbles on already pebbled elements, and under permuting pairs of pebbles).

Finally, let us deal with the incoming edges of type $T_{j,\mathbf{i}}$. For simplicity, let us start by considering two elements of the same class of $\boldsymbol{\mathcal{P}}$ belonging to different structures $\mathbf{A}$, $\mathbf{B}$. That is, let $\mathbf{a} \in A^{j'}$, $\mathbf{b} \in B^{j'}$ for some $j' \leq k$, $(\mathbf{a}, \mathbf{b}) \in \overline{W}$, and let $j \leq k$ and $\mathbf{i} \in [j]^{j'}$. For each partition class $\mathcal{P}_l$ of $\boldsymbol{\mathcal{P}}$, we need to show that $\mathbf{a}$ and $\mathbf{b}$ have the same number of incoming edges labelled $T_{j,\mathbf{i}}$ from $\mathcal{P}_l$.

Notice that, if $\{\mathbf{i}\} = [j]$, then there is nothing to prove (since as we pointed out above, the winning strategy is trivially closed under adding pebbles on already pebbled elements and permuting pairs of pebbles), so we may assume that $\{\mathbf{i}\} \subsetneq [j]$, and for the same reasons, we may assume that $\mathbf{i}$ has no repetitions and that its entries are in increasing order. Moreover, from item (4) above, we have that $|[j] \setminus \{\mathbf{i}\}| = 1$. All together then we may assume that $j' = j - 1$ and $\mathbf{i} = \mathbf{i}(j, i)$ for some $i \in [j]$. Therefore, for any j'-tuple $\mathbf{d}$, $\pi_{\mathbf{i}}\mathbf{d}^i_d = \mathbf{d}$.

Since $(\mathbf{a}, \mathbf{b}) \in \overline{W}$, there exists a bijection $f : A \to B$ such that, for each $a \in A$, $(\mathbf{a}^i_a, \mathbf{b}^i_{f(a)}) \in \overline{W}$. Hence, for each $a \in A$, $\mathbf{a}^i_a$ and $\mathbf{b}^i_{f(a)}$ belong to the same class of $\boldsymbol{\mathcal{P}}$. By the definition of $T_{j,\mathbf{i}}$, this means that $\mathbf{a}$ and $\mathbf{b}$ have the same number of incoming edges labelled $T_{j,\mathbf{i}}$ from each class of $\boldsymbol{\mathcal{P}}$.

Now we still need to check that this condition holds for any two elements of the same class of $\boldsymbol{\mathcal{P}}$ that belong to the same structure, say $\mathbf{A}$. But this is immediate since if $\mathbf{a}, \mathbf{a}' \in A^{j'}$ belong to the same class of $\boldsymbol{\mathcal{P}}$, then there exists $\mathbf{b} \in B^{j'}$ such that $(\mathbf{a}, \mathbf{b})$ and $(\mathbf{a}', \mathbf{b})$ belong to $\overline{W}$, and since both $\mathbf{a}$ and $\mathbf{b}$ and $\mathbf{a}'$ and $\mathbf{b}$ have the same number of incoming edges labelled $T_{j,\mathbf{i}}$ from each class of $\boldsymbol{\mathcal{P}}$, the same must hold for $\mathbf{a}$ and $\mathbf{a}'$. It follows that $\boldsymbol{\mathcal{P}}$ is equitable. ■

10 Conclusion

In this section we lay out some of the open questions that arise from this work.

In Chapter 6, we analysed the complexity of the fixed-template distributed constraint satisfaction problem on a synchronous, anonymous network. We gave a dichotomy theorem for the complexity of DCSP($\mathbf{A}$) in terms of the polymorphisms of $\mathbf{A}$. A number of natural open questions arise in this context. For instance, it is not clear whether asynchronous networks are strictly more powerful than their synchronous counterpart. Moreover, it would be interesting to explore the role of allowing agents to make random choices - provided this is not used to create and share unique IDs.

In the spirit of [Gro07], one could consider characterizing the structural restrictions on tractable distributed CSPs, or in other words, determining which *classes of networks* are tractable in the DCSP framework, regardless of the template. The starting point for this analysis could be the work on fibrations by Boldi et al. (see for example [BSV+96, BV01]). In particular, we propose the question of establishing a connection between the universal fibration of a graph and its iterated degree sequence.

In Chapters 7 and 8, we showed that solvability by the SA^1 relaxation of a CSP and PVCSP respectively is equivalent to invariance under the Weisfeiler-Leman-like equivalence $\equiv_1$, and also to solvability in the distributed model. The distributed algorithm for the narrower CSP setting from Chapter 6 works also for the search version of the problem, but this is unfortunately not the case for the algorithm presented for PVCSPs. Is

there an algorithm solving the search version of PVCSP($\mathbf{A}, \mathbf{B}$) whenever the PVCSP is solvable by SA^1?

Another open problem emerges from Example 8.8, which shows that BLP and SA^1 are not equivalent for PVCSPs. It follows from the sparse incomparability lemma that BLP and SA^1 are equivalent for PCSPs and from [TŽ16] that they are also equivalent for finite-valued VCSPs. Are these relaxations equivalent for general-valued VCSPs? Moreover, can we find an algebraic condition (i.e., in terms of polymorphisms) for tractability of a PVCSP by SA^1?

In Chapter 9, we defined the notion of $\equiv_k$-equivalence and characterized it in terms of homomorphism count from structures of bounded treewidth and in terms of indistinguishability in counting logic. There are a number of open-ended questions regarding the applicability of this notion. Can we define a tighter correspondence between the linear program SA^k and the equivalence relation $\equiv_k$? Even more speculatively: can we use our results to better understand the relationship between CSPs *definable in* and CSPs *closed under* k-variable logics with counting?

Part II:
Promise Model Checking

23. Everyone has the right to understand.
24. Everyone has the right to understand nothing.

Užupis republic constitution

11 Introduction

11.1 Introduction

The motivation for this second part of the thesis is in line with the goals that we set out in Chapter 1: that is, we would like to get a better understanding of the sources of tractability and the reasons for hardness in computation.[1] As mentioned in the opening chapter, known results from CSP theory endorse the principle that *symmetries*, defined loosely, are the determinant of tractability in computation. We have already seen how these symmetries are captured, for instance, by the notion of polymorphisms in the case of CSPs, and fractional polymorphisms in the case of Valued CSPs.

In this part of the thesis, motivated by recent developments in the area, we study a class of computational problems that extends the CSP – seen as the model checking problem over the existential conjunctive fragment of first-order logic – in two simultaneous directions. One direction, discussed in Subsection 11.1.1, is to consider different fragments of first-order logic. Another direction, discussed in Subsection 11.1.2, is to consider two versions of each relation, strong and weak, and only ask to distinguish strongly satisfiable inputs from those that are not even weakly satisfiable (a so-called promise problem).

[1] So far, we have used the words 'tractable' and 'hard' as synonyms for 'solvable in polynomial time' and 'NP-complete', respectively. In this part of the thesis, we will see that tractability and hardness assume a much broader meaning of upper and lower complexity bounds.

While this family of problems might not display the wide range of practical applications that one can boast for the CSP and its best-known extensions (e.g. the PCSP and VCSP), we believe they are worth studying for a variety of reasons. First, they are very natural extensions of well-studied problems, and thus also very general. Second, the structure of these problems lends itself to be studied using similar techniques to those used in the classical CSP field, thus allowing for the development of very neat theories. In particular, this is witnessed by the fact that for each of these problems we are able to establish an appropriate notion of symmetry which shows that, once again, the complexity of computational problems is governed, via the corresponding Galois connection, by the presence or absence of symmetries in the template.

11.1.1 Model checking problem parametrized by the model

Throughout this part of the thesis we will adopt the logical formulation for the constraint satisfaction problem, which we recall here. Given a fixed template $\mathbf{A}$, also called a *model*, the CSP over $\mathbf{A}$ is the problem of deciding whether a given $\{\exists, \wedge\}$-sentence in the signature of $\mathbf{A}$ is true in $\mathbf{A}$. To see that this formalization indeed expresses *constraint* satisfaction problems, consider for instance the sentence $\exists x \exists y \exists z\ R(x, y) \wedge S(y, z)$: this sentence is true in a structure $\mathbf{A}$ if the variables x, y, z can be evaluated so that both atomic formulas (constraints) are satisfied in $\mathbf{A}$.

More generally, the model checking problem (see for instance [MM18]) takes as input a finite relational structure $\mathbf{A}$ (the template or model) and a sentence ϕ in a specified logic and asks whether $\mathbf{A}$ satisfies ϕ. As in the case of the CSP, in order to obtain tractable classes we can restrict our attention to the the situation where $\mathbf{A}$ is a fixed, so the input is simply ϕ. Moreover, the logic is a fragment of the first-order logic obtained by restricting the allowed quantifiers and connectives to a fixed subset $\mathscr{L}$ of $\{\exists, \forall, \wedge, \vee, =, \neq, \neg\}$. Thus, for each $\mathbf{A}$ and each of the 2^7 choices for $\mathscr{L}$, we obtain a computational problem, which we call the *$\mathscr{L}$-Model Checking Problem over* $\mathbf{A}$ and denote $\mathscr{L}$-MC($\mathbf{A}$).

As we already mentioned, $\{\exists, \wedge\}$-MC($\mathbf{A}$) corresponds to the CSP over $\mathbf{A}$, for which a complexity classification is given by the celebrated dichotomy theorem of Bulatov [Bul17] and Zhuk [Zhu20]: each CSP over $\mathbf{A}$

is in P or is NP-complete. For the case $\mathscr{L} = \{\exists, \forall, \wedge\}$, $\mathscr{L}$-MC($\mathbf{A}$) is the Quantified CSP, another well-studied class of problems, see Section 2.2.3. It was widely believed that this class exhibits a P/NP-complete/PSPACE-complete trichotomy [Che12]. A recent breakthrough [ZM20] shows that at least three more complexity classes appear within quantified CSPs, and ongoing work suggests that even 6 is not the final number. In any case, the full complexity classification of $\{\exists, \forall, \wedge\}$-MC($\mathbf{A}$) is a challenging open problem.

The remaining $2^7 - 2$ choices for $\mathscr{L}$ do not need to be considered separately. For instance, $\{\exists, \wedge, =\}$-MC($\mathbf{A}$) is no harder than $\{\exists, \wedge\}$-MC($\mathbf{A}$) because equalities can be propagated out in this case, hence we shall freely switch between $\{\exists, \wedge, =\}$-MC($\mathbf{A}$) and $\{\exists, \wedge\}$-MC($\mathbf{A}$). Moreover, $\{\forall, \vee\}$-MC($\mathbf{A}$) is dual to $\{\exists, \wedge\}$-MC($\mathbf{A}$) so we immediately obtain a P/coNP-complete dichotomy for this problem. Finally, some choices of $\mathscr{L}$, such as $\mathscr{L} = \{\exists, \vee\}$, lead to very simple problems. It turns out [Mar08] (see Subsection 11.3) that, in addition to $\mathscr{L} = \{\exists, \wedge\}$ and $\mathscr{L} = \{\exists, \forall, \wedge\}$, only two more fragments need to be considered in order to fully understand the complexity of $\mathscr{L}$-MC($\mathbf{A}$), namely $\mathscr{L} = \{\exists, \wedge, \vee\}$ and $\mathscr{L} = \{\exists, \forall, \wedge, \vee\}$.

The former fragment was addressed in [Mar08]: except for a simple case solvable in polynomial time (in fact, L, the logarithmic space), all the remaining problems are NP-complete. The latter fragment turned out to be more challenging but, after a series of partial results [Mar08, MM12, MM10] (see also [Mar10, CM21]), the full complexity classification was given in [MM11, MM18]: each problem in this class is in L, or is NP-complete, coNP-complete, or PSPACE-complete. These results are summarized in Figure 11.1.

The starting point of the algebraic approach to $\mathscr{L}$-MC has been to find a characterization of definability in terms of certain "compatible functions" or symmetries (polymorphisms for $\mathscr{L} = \{\exists, \wedge, =\}$ [BKW17], surjective polymorphisms for $\mathscr{L} = \{\exists, \forall, \wedge\}$ [Mar17], multi-endomorphisms for $\mathscr{L} = \{\exists, \wedge, \vee\}$, surjective multi-endomorphisms for $\mathscr{L} = \{\exists, \forall, \wedge, \vee\}$ [MM18]; see also [Bör08]). We will see how corresponding characterizations of definability for promise templates will allow us to deploy the algebraic approach in the context of promise model checking.

$\mathscr{L}$-MC($\mathbf{A}$)	Complexity
$\{\exists, \wedge\}$-MC($\mathbf{A}$) (CSP)	dichotomy: P or NP-complete
$\{\exists, \forall, \wedge\}$-MC($\mathbf{A}$) (QCSP)	≥ 6 classes
$\{\exists, \wedge, \vee\}$-MC($\mathbf{A}$)	dichotomy: L or NP-complete
$\{\forall, \exists, \wedge, \vee\}$-MC($\mathbf{A}$)	tetrachotomy: L, NP-complete, coNP-complete, PSPACE-complete

Figure 11.1: Known complexity results for $\mathscr{L}$-MC($\mathbf{A}$).

11.1.2 Promise model checking problem

The Promise CSP is a recently introduced extension of the CSP framework motivated by open problems in (in)approximability of satisfiability and colouring problems, see Section 2.2.1. As for the case of the CSP, we can also interpret the PCSP as a (promise) model checking problem: the template consists of two structures $\mathbf{A}$ and $\mathbf{B}$ of the same signature, and the task is to distinguish $\{\exists, \wedge\}$-sentences that are true in $\mathbf{A}$ from those that are not true in $\mathbf{B}$.

The generalization of Promise CSP over $(\mathbf{A}, \mathbf{B})$ to an arbitrary choice of $\mathscr{L} \subseteq \{\exists, \forall, \wedge, \vee, =, \neq, \neg\}$ is referred to as the *$\mathscr{L}$-Promise Model Checking Problem over* $(\mathbf{A}, \mathbf{B})$ and is denoted $\mathscr{L}$-PMC($\mathbf{A}, \mathbf{B}$). Similarly as in the special case $\mathbf{A} = \mathbf{B}$, which is precisely $\mathscr{L}$-MC($\mathbf{A}$), it is sufficient to consider only four fragments.

A full complexity classification for $\{\exists, \wedge\}$-PMC (i.e., Promise CSP) is much desired but widely open, and $\{\exists, \forall, \wedge\}$-PMC is likely even harder. In this thesis we focus on the remaining two classes of problems, $\{\exists, \wedge, \vee\}$-PMC and $\{\exists, \forall, \wedge, \vee\}$-PMC.

Our motivation is that these cases might be substantially simpler, as indicated by the non-promise special case, and at the same time, the investigation could uncover interesting intermediate problems towards the grand endeavor of understanding the sources of tractability and hardness in computation.

Example 11.1

Consider structures $\mathbf{A}$ and $\mathbf{B}$ with a single relation symbol $=$ interpreted as the equality on the three-element domain in $\mathbf{A}$ and as the equality on the two-element domain in $\mathbf{B}$. For $\mathscr{L} = \{\exists, \forall, \wedge, \vee\}$, both $\mathscr{L}$-MC($\mathbf{A}$) and $\mathscr{L}$-MC($\mathbf{B}$) are PSPACE-complete problems, see [Mar08].

It is not hard to see that every $\mathscr{L}$-sentence true in $\mathbf{A}$ is also true in $\mathbf{B}$. In this sense, the relation in $\mathbf{A}$ is stronger than the relation in $\mathbf{B}$. On the other hand, there are $\mathscr{L}$-sentences that are true in $\mathbf{B}$ but not in $\mathbf{A}$, for instance, $\phi = \forall x \exists y \forall z\ (z = x) \vee (z = y)$. Therefore, $\mathscr{L}$-PMC($\mathbf{A}, \mathbf{B}$) could potentially be easier than the above non-promise problems – instances such as ϕ need not be considered (there is no requirement on the algorithm for such inputs). Nevertheless, the problem remains PSPACE-complete, as shown in Proposition 12.15.

The contributions of this work are organised as follows. In Theorem 12.3 and Theorem 12.10 we provide the basics for an algebraic approach to $\{\exists, \wedge, \vee\}$-PMC and $\{\exists, \forall, \wedge, \vee\}$-PMC by characterizing definability in terms of compatible functions (symmetries): multi-homomorphisms for the $\{\exists, \wedge, \vee\}$ fragment and surjective multi-homomorphisms for $\{\exists, \forall, \wedge, \vee\}$. The proofs can be obtained as relatively straightforward generalizations of the proofs for MC in [MM18]; however, we believe that our approach is somewhat more transparent. In particular, it allows us to easily characterize meaningful templates for these problems (Propositions 12.2 and 12.8).

For $\{\exists, \wedge, \vee\}$-PMC, we obtain an L/NP-complete dichotomy in Theorem 12.5. It turns out that, apart from some simple cases, the problem is NP-complete. Interestingly, there is a "single reason" for hardness: the NP-hardness of colouring a rainbow colourable hypergraph from [GL18].

For the $\{\exists, \forall, \wedge, \vee\}$ fragment, we show that there are $\{\exists, \forall, \wedge, \vee\}$-PMCs that are complete for at least four complexity classes: L, NP, coNP, and PSPACE. While our results are only partial, leaving two gaps for further investigation, they are sufficient for a full complexity classification of $\mathscr{L}$-PMC($\mathbf{A}, \mathbf{B}$) in the case that $\mathscr{L} = \{\exists, \forall, \wedge, \vee\}$ and at least one of the structures $\mathbf{A}$, $\mathbf{B}$ has a two-element domain, and also in the case that

$\mathscr{L} \supsetneq \{\exists, \forall, \wedge, \vee\}$. In Section 13, we conclude by giving some examples where our efforts have failed so far. One such example is a particularly interesting $\{\exists, \forall, \wedge, \vee\}$-PMC over 3-element domains: given a $\{\exists, \forall, \wedge, \vee\}$-sentence ϕ whose atomic formulas are all of the form $R^i(x)$, $i \in \{1, 2, 3\}$, distinguish between the case where ϕ is true when $R^i(x)$ is interpreted as "$x = i$", and the case where ϕ is false when $R^i(x)$ is interpreted as "$x \neq i$".

Our complexity results are summarized in Figure 11.2, where the conditions for $\mathscr{L} = \{\exists, \forall, \wedge, \vee\}$ are stated in terms of two special types of surjective multi-homomorphisms which we introduce in Subsection 12.2.3.

$\mathscr{L}$-PMC$(\mathbf{A}, \mathbf{B})$	**Condition**	**Complexity**
$\{\exists, \forall, \wedge\}$-PMC$(\mathbf{A}, \mathbf{B})$		Dichotomy: L or NP-c
$\{\exists, \forall, \wedge, \vee\}$-PMC$(\mathbf{A}, \mathbf{B})$	AE-smuhom, or A-smuhom and E-smuhom and $\mathbf{A}, \mathbf{B}$ digraphs	L
	A-smuhom and E-smuhom	NP $\cap$ coNP
	A-smuhom, no E-smuhom	NP-c
	E-smuhom, no A-smuhom	coNP-c
	no A-smuhom and no E-smuhom	NP-hard and coNP-hard
$\{\exists, \forall, \wedge, \vee, =\}$-PMC$(\mathbf{A}, \mathbf{B})$, $\{\exists, \forall, \wedge, \vee, \neq\}$-PMC$(\mathbf{A}, \mathbf{B})$, $\{\exists, \forall, \wedge, \vee, \neg\}$-PMC$(\mathbf{A}, \mathbf{B})$		Dichotomy: L or PSPACE-c

Figure 11.2: Complexity results for $\mathscr{L}$-PMC$(\mathbf{A}, \mathbf{B})$. We use '-c' as an abbreviation for '-complete'.

11.2 Preliminaries

Structures. For the rest of this thesis we will make a few additional assumptions. In particular, for every relational structure $\mathbf{A}$, we will assume that the (finite) universe A has size at least 2, and that each relation is non-empty and *proper*, i.e., $\emptyset \subsetneq R^{\mathbf{A}} \subsetneq A^{\mathrm{ar}(R)}$ for each relation symbol R in

σ. These nonstandard requirements are placed for technical convenience and do not significantly decrease the generality of our results. In fact, in the case where at least one of the domains is a singleton, all the problems are trivially solvable in constant time.

The *complement* of a relation S is denoted $\overline{S} := A^n \setminus S$. The complement of a relational structure $\mathbf{A}$ is obtained by taking complements of all relations in the structure and is denoted $\overline{\mathbf{A}}$. Note that the fact that all relations are nonempty and proper guarantees that complements are always well-defined.

Multi-homomorphisms. A *multi-valued function* f from A to B is a mapping from A to $\mathscr{P}_{\neq\emptyset}(B)$, the set of all nonempty subsets of B. Note that all multi-valued functions are therefore assumed to be *total*, that is, for all $a \in A$ there is some $b \in B$ such that $b \in f(a)$. A multi-valued function f is called *surjective* if for every $b \in B$ there exists $a \in A$ such that $b \in f(a)$. The *inverse* of a surjective multi-valued function f from A to B is the multi-valued function from B to A defined by $f^{-1}(b) = \{a : b \in f(a)\}$. Observe that the inverse of a surjective total multi-valued function is surjective and total. For a tuple $\mathbf{a} \in A^n$ we write $f(\mathbf{a})$ for $f(a_1) \times \cdots \times f(a_n)$. The value $\max\{|f(a)| : a \in A\}$ is referred to as the *multiplicity* of f; in particular, multi-valued functions of multiplicity one are essentially functions. For two multi-valued functions f and f' from A to B, we say that f' is *contained in* f if $f'(a) \subseteq f(a)$ for each $a \in A$. For a multi-valued function f from A to B and a set $S \subseteq A$ we define $f(S) = \cup_{a\in S} f(a)$. Composition of multi-valued functions is then defined in the natural way.

Given two similar structures $\mathbf{A}$ and $\mathbf{B}$, a multi-valued function f from A to B is called a *multi-homomorphism*[2] from $\mathbf{A}$ to $\mathbf{B}$ if for any R in the signature and any $\mathbf{a} \in R^{\mathbf{A}}$ we have $f(\mathbf{a}) \subseteq R^{\mathbf{B}}$, i.e., $\mathbf{b} \in R^{\mathbf{B}}$ whenever $b_i \in f(a_i)$ for each $i \in [\mathrm{ar}(R)]$. A (multi)-endomorphism is a (multi)-homomorphism from some structure $\mathbf{A}$ to itself. Notice that if f is a multi-homomorphism from $\mathbf{A}$ to $\mathbf{B}$, then so is any multi-valued function contained in f. In particular, if f is a multi-homomorphism from $\mathbf{A}$ to $\mathbf{B}$, then any function $g : A \to B$ such that g is contain ed in f is a

[2]We deviate here from the terminology of [MM11, MM12] because it would not work well in the promise setting.

homomorphism from $\mathbf{A}$ to $\mathbf{B}$. The converse does not hold in general, as witnessed by the following example.

Example 11.2

Let $\mathbf{A} = \mathbf{B}$ be the structure with a single binary equality relation on the Boolean domain, i.e., $\mathbf{A} = \mathbf{B} = (\{0,1\}; =_2)$. Then, both g_1 and g_2 are endomorphisms of $\mathbf{A}$, however f is not a multi-endomorphism of $\mathbf{A}$, where $g_1(a) = a$; $g_2(a) = 1 - a$ ($a \in \{0,1\}$), and $f(0) = f(1) = \{0,1\}$.

The set of multi-homomorphisms from $\mathbf{A}$ to $\mathbf{B}$ is denoted by $\mathrm{MuHom}(\mathbf{A}, \mathbf{B})$ and the set of surjective multi-homomorphisms by $\mathrm{SMuHom}(\mathbf{A}, \mathbf{B})$.

Fragments of first-order logic, definability, and model checking. Let $\mathscr{L} \subseteq \{\exists, \forall, \wedge, \vee, =, \neq, \neg\}$ and fix some signature. By an *$\mathscr{L}$-sentence* (resp., *$\mathscr{L}$-formula*) we mean a sentence (resp., formula) of first-order logic that only uses variables (denoted x_i, y_i, z_i), relation symbols in the signature, and connectives and quantifiers in $\mathscr{L}$. We refer to this fragment of first-order logic as the *$\mathscr{L}$-logic*.

The *prenex normal form* of an $\mathscr{L}$-formula is an equivalent formula that begins with quantified variables followed by a quantifier-free formula. The prenex normal form can be computed in logarithmic space and it is an $\mathscr{L}$-formula whenever $\mathscr{L}$ does not contain the negation symbol $\neg$, so we shall assume that all the inputs to the MC problem are in prenex normal form.

For a structure $\mathbf{A}$ in the signature and an $\mathscr{L}$-sentence ϕ, we write $\mathbf{A} \vDash \phi$ if ϕ is satisfied in $\mathbf{A}$ (i.e., ϕ is true when $\forall v, \exists v, R(\mathbf{v})$ are replaced by $(\forall\, v \in A)$, $(\exists\, v \in A)$, $R^{\mathbf{A}}(\mathbf{v})$, respectively). More generally, given an $\mathscr{L}$-formula ψ, a tuple of distinct variables $(v_1, \ldots, v_n)$ which contains every free variable of ψ and a tuple $(a_1, \ldots, a_n) \in A^n$, we write $\mathbf{A} \vDash \psi(a_1, \ldots, a_n)$ if ψ is satisfied when $v_1, \ldots, v_n$ are evaluated as $\varepsilon_A(v_1) = a_1, \ldots, \varepsilon_A(v_n) = a_n$, respectively. Notice that variables $v_1, \ldots, v_n$ indeed need to be pairwise distinct, otherwise this notation would not make sense. The tuple $(v_1, \ldots, v_n)$ is often specified by writing $\psi = \psi(v_1, \ldots, v_n)$.

Recall that in Chapter 3 we introduced the notion of primitive positive definability. We can generalize this concept as follows. Let $\mathscr{L} \subseteq \{\exists, \forall, \wedge, \vee, =, \neq, \neg\}$. We say that a relation $S \subseteq A^n$ is *$\mathscr{L}$-definable* from $\mathbf{A}$ if there exists an $\mathscr{L}$-formula $\psi(v_1, \ldots, v_n)$ such that, for all $(a_1, \ldots, a_n) \in A^n$, we have $(a_1, \ldots, a_n) \in S$ if and only if $\mathbf{A} \models \psi(a_1, \ldots, a_n)$. In this case, we also say that $\psi(v_1, \ldots, v_n)$ defines S in $\mathbf{A}$. Notice that then the notion of pp-definability introduced in Chapter 3 corresponds to $\{\exists, \wedge, =\}$-definability, and the notion of efpp-definability introduced in Chapter 6 corresponds to $\{\exists, \wedge\}$-definability.

Let $\mathscr{L} \subseteq \{\exists, \forall, \wedge, \vee, =, \neq, \neg\}$ and $\mathbf{A}$ be a σ-structure. The *$\mathscr{L}$-Model Checking Problem over* $\mathbf{A}$, denoted $\mathscr{L}$-MC($\mathbf{A}$), is the problem of deciding whether a given $\mathscr{L}$-sentence ϕ (in the signature σ of $\mathbf{A}$) is true in $\mathbf{A}$.

Then, we have the following reduction between definable problems, which generalizes Theorem 2.1.

Theorem 11.3. *Let* $\mathbf{A}$ *and* $\mathbf{C}$ *be relational structures on the same domain such that* $\mathbf{C}$ *is* $\mathscr{L}$*-definable from* $\mathbf{A}$*. Then,* $\mathscr{L}$-MC($\mathbf{C}$) *is log-space reducible to* $\mathscr{L}$-MC($\mathbf{A}$).

The reduction, much like in the primitive positive (i.e., $\mathscr{L} = \{\exists, \wedge, =\}$) case, amounts to replacing atomic formulas of the form $R(\mathbf{v})$ by their $\mathscr{L}$-definitions.

Promise Model Checking and Promise definability. Let $\mathscr{L} \subseteq \{\exists, \forall, \wedge, \vee, =, \neq, \neg\}$. A pair of similar structures $(\mathbf{A}, \mathbf{B})$ is called an $\mathscr{L}$-PMC *template* if for every $\mathscr{L}$-sentence ϕ in the signature of $\mathbf{A}$ and $\mathbf{B}$, $\mathbf{A} \models \phi$ implies $\mathbf{B} \models \phi$. Equivalently, every $\mathscr{L}$-sentence that is true in $\mathbf{A}$ is also true in $\mathbf{B}$.

Given an $\mathscr{L}$-PMC template $(\mathbf{A}, \mathbf{B})$, the *$\mathscr{L}$-Promise Model Checking Problem over* $(\mathbf{A}, \mathbf{B})$, denoted $\mathscr{L}$-PMC($\mathbf{A}, \mathbf{B}$), is the following problem: given an $\mathscr{L}$-sentence ϕ in the signature of $\mathbf{A}$ and $\mathbf{B}$, output Yes if $\mathbf{A} \models \phi$, and output No if $\mathbf{B} \not\models \phi$.

Notice that $\{\exists, \wedge\}$-PMC is precisely the PCSP, and that the definition of $\mathscr{L}$-PMC template then generalizes that of PCSP template (see Chapter 3), i.e., it guarantees that the sets of Yes and No instances are disjoint. However, as for the PCSP, their union need not be the whole set of $\mathscr{L}$-sentences; an algorithm for $\mathscr{L}$-PMC is only required to produce correct

outputs for Yes instances and No instances. Alternatively, we are *promised* that the input sentence is a Yes instance or a No instance.

The complexity-theoretic notions (such as membership in NP, NP-completeness, reductions) can be adjusted naturally for the promise setting. In particular, we write $\mathscr{L}$-PMC$(\mathbf{C},\mathbf{D}) \le \mathscr{L}$-PMC$(\mathbf{A},\mathbf{B})$ if the former problem can be reduced to the latter problem by a logarithmic space reduction, that is, a logarithmic space transformation that maps each Yes instance ϕ of $\mathscr{L}$-PMC$(\mathbf{C},\mathbf{D})$ to a Yes instance ψ of $\mathscr{L}$-PMC$(\mathbf{A},\mathbf{B})$ (equivalently, $\mathbf{C} \models \phi$ must imply $\mathbf{A} \models \psi$) and No instances of $\mathscr{L}$-PMC$(\mathbf{C},\mathbf{D})$ to No instances $\mathscr{L}$-PMC$(\mathbf{A},\mathbf{B})$ (equivalently, $\mathbf{B} \models \psi$ must imply $\mathbf{D} \models \phi$).

An appropriate adjustment of definability for the promise setting is as follows. Assume $\neg \notin \mathscr{L}$ and let $(\mathbf{A},\mathbf{B})$ be a pair of similar structures. We say that a pair of relations (S,T), where $S \subseteq A^n$ and $T \subseteq B^n$, is *promise-$\mathscr{L}$-definable* (or *p-$\mathscr{L}$-definable*) from $(\mathbf{A},\mathbf{B})$ if there exist relations S' and T' and an $\mathscr{L}$-formula $\psi(v_1,\dots,v_n)$ such that $S \subseteq S'$, $T' \subseteq T$, $\psi(v_1,\dots,v_n)$ defines S' in $\mathbf{A}$, and $\psi(v_1,\dots,v_n)$ defines T' in $\mathbf{B}$.

Note that the notion of promise definability does not allow the negation in $\mathscr{L}$, as including $\neg$ would not work well with the inclusions in the definition.

We say that an $\mathscr{L}$-PMC template $(\mathbf{C},\mathbf{D})$ is p-$\mathscr{L}$-definable from $(\mathbf{A},\mathbf{B})$ (the signatures can differ) if $(Q^{\mathbf{C}}, Q^{\mathbf{D}})$ is p-$\mathscr{L}$-definable from $(\mathbf{A},\mathbf{B})$ for each relation symbol Q in the signature of $\mathbf{C}$ and $\mathbf{D}$. We have the following complexity reduction between promise problems.

Theorem 11.4. *Assume $\neg \notin \mathscr{L}$. If $(\mathbf{A},\mathbf{B})$ and $(\mathbf{C},\mathbf{D})$ are $\mathscr{L}$-PMC templates such that $(\mathbf{C},\mathbf{D})$ is p-$\mathscr{L}$-definable from $(\mathbf{A},\mathbf{B})$, then $\mathscr{L}$-PMC$(\mathbf{C},\mathbf{D})$ $\le \mathscr{L}$-PMC$(\mathbf{A},\mathbf{B})$.*

Proof. The reduction is to replace each atomic $Q(\mathbf{v})$ in every instance ϕ of $\mathscr{L}$-PMC$(\mathbf{C},\mathbf{D})$ by the corresponding formula ψ which p-$\mathscr{L}$-defines $(Q^{\mathbf{C}}, Q^{\mathbf{D}})$ in $(\mathbf{A},\mathbf{B})$. For correctness of this reduction, observe that an $\mathscr{L}$-sentence which is true in a structure $\mathbf{E}$ remains true when we add tuples to the relations of $\mathbf{E}$ (since $\mathscr{L}$ does not contain the negation). □

11.3 Interesting fragments

As mentioned in Section 11.1.1, only four fragments of first-order logic need to be considered in order to fully understand $\mathscr{L}$-PMC($\mathbf{A}, \mathbf{B}$). We now explain why in a little more details.

First, observe that if $\mathscr{L}$ does not contain any connective ($\wedge, \vee$), or $\mathscr{L}$ does not contain any quantifier ($\exists, \forall$), or $\mathscr{L} \subseteq \{\exists, \vee\}$, then each $\mathscr{L}$-PMC is in L, i.e., it is solvable in logarithmic space (in fact, in some of these cases we do not even have any valid inputs.)

Secondly, notice that ($\mathscr{L} \cup \{=\}$)-PMC($\mathbf{A}, \mathbf{B}$) is essentially the same as $\mathscr{L}$-PMC($\mathbf{A}', \mathbf{B}'$), where $\mathbf{A}'$ and $\mathbf{B}'$ are obtained from the original structures by adding a fresh binary symbol $=$ to the signature which is interpreted as $=_A$ in $\mathbf{A}'$ and as $=_B$ in $\mathbf{B}'$. The disequality is dealt with analogously, thus we can and shall restrict to fragments with $\mathscr{L} \subseteq \{\exists, \forall, \wedge, \vee, \neg\}$.

Next, we deal with the negation. If $\neg$ is in $\mathscr{L}$, and $\mathscr{L}$ contains at least a quantifier and a connective, then it is sufficient to consider the case $\mathscr{L} = \{\exists, \forall, \wedge, \vee, \neg\}$, since the possibly remaining quantifiers and connectives can be expressed using negation. Moreover, the complements of relations can also be expressed, so we may assume that each template ($\mathbf{A}, \mathbf{B}$) is *closed under complementation*, meaning that for every symbol R in the signature, we have a symbol $\overline{R}$ interpreted as $\overline{R}^{\mathbf{A}} = \overline{R^{\mathbf{A}}}$, $\overline{R}^{\mathbf{B}} = \overline{R^{\mathbf{B}}}$. But then, $\neg$ is no longer necessary since we can propagate the negations inwards in an input sentence. We are down to $\mathscr{L} \subseteq \{\exists, \forall, \wedge, \vee\}$.

Finally, note that $\mathbf{E} \models \neg\phi$, where ϕ is an $\mathscr{L}$-sentence, is equivalent to $\overline{\mathbf{E}} \models \phi'$ where ϕ' is an $\mathscr{L}'$-sentence and $\mathscr{L}'$ is obtained from $\mathscr{L}$ by swapping $\forall \leftrightarrow \exists$ and $\vee \leftrightarrow \wedge$ (ϕ' can be, again, computed from $\neg\phi$ by inward propagation). It follows that $\phi \mapsto \phi'$ transforms every Yes instance (resp., No instance) of $\mathscr{L}$-PMC($\mathbf{A}, \mathbf{B}$) to a No instance (resp., Yes instance) of $\mathscr{L}'$-PMC($\overline{\mathbf{B}}, \overline{\mathbf{A}}$), and a similar "dual" reduction works in the opposite direction. Therefore, the latter PMC has the "dual" complexity to the former PMC, e.g., if the former is NP-complete, then the latter is coNP-complete; and if the former is PSPACE-complete, then the latter is PSPACE-complete as well. We will refer to this reasoning as the *duality argument*.

Eliminating one of the logic fragments from each of the "dual" pairs,

we are left with only four fragments: $\mathscr{L} = \{\exists, \wedge\}$ (whose $\mathscr{L}$-PMC is the PCSP), $\mathscr{L} = \{\exists, \forall, \wedge\}$ (the Promise Quantified CSP), $\mathscr{L} = \{\exists, \wedge, \vee\}$, and $\mathscr{L} = \{\exists, \forall, \wedge, \vee\}$. We investigate the last two separately in the next Chapter.

12 The Complexity of Promise Model Checking

In this chapter we set the basics for an algebraic approach to the existential positive and the positive promise model checking problems. This allows us to give a full complexity classification for the $\{\exists, \wedge, \vee\}$-PMC *and to provide a number of upper and lower bounds for the* $\{\exists, \forall, \wedge, \vee\}$-PMC.

12.1 Existential positive fragment

This section concerns the existential positive equality-free logic, that is, the $\mathscr{L}$-logic with $\mathscr{L} = \{\exists, \wedge, \vee\}$. We fix this $\mathscr{L}$ for the entire section.

12.1.1 Characterization of templates and p-$\mathscr{L}$-definability

We start by characterizing $\mathscr{L}$-PMC templates. We will need the following observation.

Lemma 12.1. *Let* f *be a multi-homomorphism from* $\mathbf{A}$ *to* $\mathbf{B}$*, let* $\phi(x_1, \ldots, x_n)$ *be a quantifier-free* $\mathscr{L}$*-formula in the same signature, and let* $\boldsymbol{a} \in A^n$*,* $\boldsymbol{b} \in B^n$*. If* $\mathbf{A} \models \phi(\boldsymbol{a})$ *and* $\boldsymbol{b} \in f(\boldsymbol{a})$*, then* $\mathbf{B} \models \phi(\boldsymbol{b})$*.*

Proof. The claim holds for atomic formulas by the definition of multi-homomorphisms. The proof is then finished by induction on the complexity of ϕ; both $\vee$ and $\wedge$ are dealt with in a straightforward way. □

Proposition 12.2. *A pair* $(\mathbf{A}, \mathbf{B})$ *of similar structures is an* $\mathscr{L}$-PMC *template if and only if there exists a homomorphism from* $\mathbf{A}$ *to* $\mathbf{B}$*.*

Proof. Suppose that there exists a homomorphism from $\mathbf{A}$ to $\mathbf{B}$ and $\mathbf{A} \models \phi$, where $\phi = \exists x_1 \exists x_2 \dots \exists x_n \phi'(x_1, \dots, x_n)$ is in prenex normal form. Then we have $\mathbf{A} \models \phi'(\mathbf{a})$ for some $\mathbf{a} \in A^n$, therefore $\mathbf{B} \models \phi'(f(\mathbf{a}))$ by Lemma 12.1, and it follows that $\mathbf{B} \models \phi$.

For the forward implication, assume $A = [k] := \{1, \dots, k\}$ and consider the following formula.

$$\phi(x_1, \dots, x_k) := \bigwedge_{R \in \sigma} \bigwedge_{\mathbf{r} \in R^{\mathbf{A}}} R(x_{r_1}, \dots, x_{r_{\mathrm{ar}(R)}}) \tag{12.1}$$

It follows immediately from the definitions that, for any σ-structure $\mathbf{E}$, $\mathbf{E} \models \phi(e_1, \dots, e_k)$ if and only if the mapping defined by $i \mapsto e_i$ for each $i \in [k]$ is a homomorphism from $\mathbf{A}$ to $\mathbf{E}$.

By existentially quantifying all the variables in (12.1) we obtain a sentence that is true in $\mathbf{A}$ (as there exists a homomorphism from $\mathbf{A}$ to $\mathbf{A}$ – the identity), so it must be true in $\mathbf{B}$. Therefore, let $(b_1, \dots, b_k)$ be such that $\mathbf{B} \models \phi(b_1, \dots, b_k)$. It follows that the map $i \mapsto b_i$ is a homomorphism from $\mathbf{A}$ to $\mathbf{B}$. □

We point out here that an equivalent characterization of $\mathscr{L}$-templates is available in terms of multi-homomorphisms: that is, a pair $(\mathbf{A}, \mathbf{B})$ of similar structures is an $\mathscr{L}$-PMC template if and only if there exists a multi-homomorphism from $\mathbf{A}$ to $\mathbf{B}$. It is immediate to see that this condition is equivalent to Proposition 12.2: one direction of this equivalence is trivial; for the other direction, simply observe that every function h contained in a multi-homomorphism from $\mathbf{A}$ to $\mathbf{B}$ is a homomorphism from $\mathbf{A}$ to $\mathbf{B}$.

Furthermore, note that this characterization would remain the same if we add $=$ to $\mathscr{L}$ (and/or remove $\vee$). Conversely, for the following characterization of promise definability, the absence of the equality relation does make a difference, which is why we need to use multi-homomorphisms instead of homomorphisms for the equality-free fragment.

Theorem 12.3. *Let* $(\mathbf{A}, \mathbf{B})$ *and* $(\mathbf{C}, \mathbf{D})$ *be* $\mathscr{L}$-PMC *templates such that* $A = C$ *and* $B = D$. *Then* $(\mathbf{C}, \mathbf{D})$ *is p-*$\mathscr{L}$*-definable from* $(\mathbf{A}, \mathbf{B})$ *if and only if* $\mathrm{MuHom}(\mathbf{A}, \mathbf{B}) \subseteq \mathrm{MuHom}(\mathbf{C}, \mathbf{D})$. *Moreover, in this case,* $\mathscr{L}\text{-PMC}(\mathbf{C}, \mathbf{D}) \leq \mathscr{L}\text{-PMC}(\mathbf{A}, \mathbf{B})$.

Proof. It is enough to verify the equivalence, since then the second claim follows from Theorem 11.4. To prove the forward implication, assume that $(\mathbf{C}, \mathbf{D})$ is p-$\mathscr{L}$-definable from $(\mathbf{A}, \mathbf{B})$, let $f \in \mathrm{MuHom}(\mathbf{A}, \mathbf{B})$, and let Q be a symbol in the signature of $\mathbf{C}$ and $\mathbf{D}$. To show that $f(\mathbf{a}) \subseteq Q^{\mathbf{D}}$ for any $\mathbf{a} \in Q^{\mathbf{C}}$ we apply Lemma 12.1 as follows. We have $\mathbf{A} \models \psi(\mathbf{a})$, where $\psi(\mathbf{x}) = \exists y_1 \exists y_2 \dots \exists y_m \psi'(\mathbf{x}, \mathbf{y})$ is a formula which p-$\mathscr{L}$-defines $(\mathbf{C}, \mathbf{D})$ from $(\mathbf{A}, \mathbf{B})$, turned into prenex normal form. Then $\mathbf{A} \models \psi'(\mathbf{a}, \mathbf{a}')$ for some $\mathbf{a}' \in A^m$, thus $\mathbf{B} \models \psi'(\mathbf{b}, \mathbf{b}')$ for any $\mathbf{b} \in f(\mathbf{a})$ and $\mathbf{b}' \in f(\mathbf{a}')$ by Lemma 12.1. Therefore, $\mathbf{B} \models \psi(\mathbf{b})$ and, finally, $\mathbf{b} \in Q^{\mathbf{D}}$, as required.

For the backward implication, assume that $\mathrm{MuHom}(\mathbf{A}, \mathbf{B}) \subseteq \mathrm{MuHom}(\mathbf{C}, \mathbf{D})$, denote by σ the signature of $\mathbf{A}$ and $\mathbf{B}$, and consider an n-ary relational symbol Q in the signature of $\mathbf{C}$ and $\mathbf{D}$. To prove the claim, we need to find a formula $\psi(x_1, \dots, x_n)$ that defines, in $\mathbf{A}$, a relation containing $Q^{\mathbf{C}}$ and, in $\mathbf{B}$, a relation contained in $Q^{\mathbf{D}}$.

For simplicity, assume $A = [k]$ and consider the formula

$$\phi(x_{1,1}, \dots, x_{1,n}, x_{2,1}, \dots, x_{2,n}, \dots, x_{k,1}, \dots, x_{k,n}) := \bigwedge_{R \in \sigma} \bigwedge_{\mathbf{r} \in R^{\mathbf{A}}} \bigwedge_{\mathbf{j} \in [n]^{\mathrm{ar}(R)}} R(x_{r_1, j_1}, \dots, x_{r_{\mathrm{ar}(R)}, j_{\mathrm{ar}(R)}}) \tag{12.2}$$

It follows immediately from definitions that, for any structure $\mathbf{E}$ in the signature σ, we have $\mathbf{E} \models \phi(e_{1,1}, \dots, e_{k,n})$ if and only if the mapping $i \mapsto \{e_{i,1}, \dots, e_{i,n}\}$, $1 \leq i \leq k$ is a multi-homomorphism from $\mathbf{A}$ to $\mathbf{E}$. Therefore, for any $\mathbf{a} \in A^n$, the formula $\tau_{\mathbf{a}}(x_1, \dots, x_n)$, obtained from ϕ by renaming $x_{a_i, i}$ to x_i and existentially quantifying the remaining variables, defines in $\mathbf{E}$ the union of $f(\mathbf{a})$ over $f \in \mathrm{MuHom}(\mathbf{A}, \mathbf{E})$ of multiplicity at most n. This relation is clearly equal to the union of $f(\mathbf{a})$ over all $f \in \mathrm{MuHom}(\mathbf{A}, \mathbf{E})$. The required formula ψ is then the disjunction of $\tau_{\mathbf{a}}$ over all $\mathbf{a} \in Q^{\mathbf{C}}$: it defines in $\mathbf{A}$ a relation containing $Q^{\mathbf{C}}$ (because of the identity "multi"-homomorphism $\mathbf{A} \to \mathbf{A}$) and, in $\mathbf{B}$, a relation contained in $Q^{\mathbf{D}}$ (because every multi-homomorphism from $\mathbf{A}$ to $\mathbf{B}$ is a multi-homomorphism from $\mathbf{C}$ to $\mathbf{D}$, and hence $f(\mathbf{a}) \subseteq Q^{\mathbf{D}}$ for any $\mathbf{a} \in Q^{\mathbf{C}}$ and any $f \in \mathrm{MuHom}(\mathbf{A}, \mathbf{B})$). □

12.1.2 Complexity classification

Since $\mathscr{L}$-PMC$(\mathbf{A}, \mathbf{B})$ reduces to $\mathscr{L}$-MC$(\mathbf{A})$ (or $\mathscr{L}$-MC$(\mathbf{B})$) by the trivial reduction which does not change the input, and the latter problem is clearly in NP, then the former problem is in NP as well. In Theorem 12.5 below, we show that $\mathscr{L}$-PMC$(\mathbf{A}, \mathbf{B})$ is NP-hard in all the "nontrivial" cases, as in the non-promise setting. However, our proof of hardness requires (in addition to Theorem 12.3) a much more involved hardness result than in the non-promise case: NP-hardness of c-colouring rainbow k-colourable $2k$-uniform hypergraphs from [GL18] (where $c, k \geq 2$).

To translate this result into our formalism, recall the definitions of the n-ary "rainbow colouring" and "not-all-equal" relations on a finite domain D introduced in Chapter 3:

$$\mathrm{Rb}_D^n = \{\mathbf{d} \in D^n : \{d_1, d_2, \ldots, d_n\} = D\},$$
$$\mathrm{NAE}_D^n = \{\mathbf{d} \in D^n : \neg(d_1 = d_2 = \cdots = d_n)\}.$$

Then, the result of Guruswami and Lee can be stated as follows.

Theorem 12.4 (Corollary 1.2 in [GL18])**.** *For any A and B of size at least 2, the problem* $\{\exists, \wedge\}$-PMC$((A; \mathrm{Rb}_A^{2|A|}), (B; \mathrm{NAE}_B^{2|A|}))$ *is* NP*-complete.*[1]

Given this hardness result, the complexity classification is a simple consequence of Theorem 12.3.

Theorem 12.5 ($\mathscr{L} = \{\exists, \wedge, \vee\}$)**.** *Let* $(\mathbf{A}, \mathbf{B})$ *be an* $\mathscr{L}$-PMC *template. If there is a constant homomorphism from* $\mathbf{A}$ *to* $\mathbf{B}$*, then* $\mathscr{L}$-PMC$(\mathbf{A}, \mathbf{B})$ *is in* L *(in fact, decidable in constant time), otherwise* $\mathscr{L}$-PMC$(\mathbf{A}, \mathbf{B})$ *is* NP*-complete.*

Proof. If there exists a constant homomorphism $f : \mathbf{A} \to \mathbf{B}$, say with image $\{b\}$, then all the relations $R^{\mathbf{B}}$ in $\mathbf{B}$ must contain the constant tuple $(b, b, \ldots, b)$. It follows that every input sentence is satisfied in $\mathbf{B}$ by evaluating the existentially quantified variables to b; therefore, Yes is always a correct output.

[1] The result in [GL18] is slightly stronger since the NP-hardness remains true even when the input sentences are required to have distinct variables in each atomic formula (such instances correspond to $2|A|$-uniform hypergraphs).

If there is no constant homomorphism $\mathbf{A} \to \mathbf{B}$, we observe that no multi-homomorphism from $\mathbf{A}$ to $\mathbf{B}$ contains a constant homomorphism (as the set of multi-homomorphisms of a PMC template is closed under containment). It follows that the image of any "rainbow" tuple of A under any multi-homomorphism from $\mathbf{A}$ to $\mathbf{B}$ does not contain any constant tuple, and so any multi-homomorphism from $\mathbf{A}$ to $\mathbf{B}$ is a multi-homomorphism from $(A; \mathrm{Rb}_A^{2|A|})$ to $(B; \mathrm{NAE}_B^{2|A|})$. The reduction from Theorem 12.3 and the hardness from Theorem 12.4 conclude the proof. □

12.2 Positive fragment

We now turn our attention to the more complex case – the positive equality-free logic, that is, the $\mathscr{L}$-logic with $\mathscr{L} = \{\exists, \forall, \wedge, \vee\}$. We again fix this $\mathscr{L}$ for the entire section.

12.2.1 Witnesses for quantified formulas

It will be convenient to work with $\mathscr{L}$-formulas of the special form

$$\phi(x_1, \ldots, x_n) = \forall y_1 \exists z_1 \forall y_2 \exists z_2 \ldots \forall y_m \exists z_m \ \phi'(\mathbf{x}, \mathbf{y}, \mathbf{z}), \tag{12.3}$$

where ϕ' is quantifier-free. Note that each formula is equivalent to a formula in this form (by transforming to prenex normal form and adding dummy variables and quantification as needed) and the conversion can be done in logarithmic space.

Observe that for a structure $\mathbf{A}$ and a tuple $\mathbf{a} \in A^n$, we have $\mathbf{A} \vDash \phi(\mathbf{a})$ if and only if there exist functions $\alpha_1 : A \to A$, $\alpha_2 : A^2 \to A$, …, $\alpha_m : A^m \to A$ which give us evaluations of the existentially quantified variables given the value of the previous universally quantified variables, i.e., these functions satisfy $\mathbf{A} \vDash \phi'(\mathbf{a}, \mathbf{c}, \alpha_1(c_1), \alpha_2(c_1, c_2), \ldots, \alpha_m(c_1, \ldots, c_m))$ for every $\mathbf{c} \in A^m$. We call such functions *witnesses* for $\mathbf{A} \vDash \phi(\mathbf{a})$.

The following is a simple consequence of this viewpoint, a version of Lemma 12.1.

Lemma 12.6. *Let f be a surjective multi-homomorphism from $\mathbf{A}$ to $\mathbf{B}$, let $\phi(x_1, \ldots, x_n)$ be an $\mathscr{L}$-formula in the same signature as $\mathbf{A}$ and $\mathbf{B}$, and let $\boldsymbol{a} \in A^n$, $\boldsymbol{b} \in B^n$. If $\mathbf{A} \vDash \phi(\boldsymbol{a})$ and $\boldsymbol{b} \in f(\boldsymbol{a})$, then $\mathbf{B} \vDash \phi(\boldsymbol{b})$.*

In particular, if there exists a surjective multi-homomorphism from $\mathbf{A}$ *to* $\mathbf{B}$*, and* ϕ *is an* $\mathscr{L}$*-sentence such that* $\mathbf{A} \models \phi$*, then* $\mathbf{B} \models \phi$*.*

Proof. The claim holds for quantifier-free $\mathscr{L}$-formulas by Lemma 12.1.

Next, we assume that ϕ is of the form (12.3) and select witnesses $\alpha_1, \ldots, \alpha_m$ for $\mathbf{A} \models \phi(\mathbf{a})$. Let $g : B \to A$ be any function such that $b \in f(g(b))$ for every $b \in B$, which exists as f is surjective. We claim that any functions $\beta_1, \ldots, \beta_m$ such that $\beta_i(b_1, \ldots, b_i) \in f(\alpha_i(g(b_1), \ldots, g(b_i)))$ for every $i \in [m]$, are witnesses for $\mathbf{B} \models \phi(\mathbf{b})$. Indeed, for all $\mathbf{d} \in B^m$, we have $\mathbf{A} \models \phi'(\mathbf{a}, g(\mathbf{d}), \alpha_1(g(d_1)), \ldots, \alpha_m(g(d_1), \ldots, g(d_m)))$, and also $\mathbf{b} \in f(\mathbf{a})$, $\mathbf{d} \in f(g(\mathbf{d}))$, and $\beta_i(d_1, \ldots, d_i) \in f(\alpha_i(g(d_1), \ldots, g(d_i)))$ (by the assumption, choice of g, and choice of β_i, respectively); therefore, $\mathbf{B} \models \phi'(\mathbf{b}, \mathbf{d}, \beta_1(d_1), \ldots, \beta_m(d_1, \ldots, d_m))$ since ϕ' is quantifier-free. Hence, $\mathbf{B} \models \phi(\mathbf{b})$. □

12.2.2 Characterization of templates and p-$\mathscr{L}$-definability

Unlike in the existential case, both characterizations require surjective and multi-valued functions. The core of these characterizations is an adjustment of (12.2) for surjective homomorphisms.

Lemma 12.7. *Let* $\mathbf{A}$ *be a structure with* $A = [k]$ *and* m, n *be arbitrary positive integers. Then there exists a formula* $\phi(x_{1,1}, \ldots, x_{1,n}, x_{2,1}, \ldots, x_{k,n})$ *such that, for any structure* $\mathbf{E}$ *similar to* $\mathbf{A}$ *with* $|E| \leq m$*, we have* $\mathbf{E} \models \phi(e_{1,1}, \ldots, e_{k,n})$ *if and only if the multi-valued mapping* $i \mapsto \{e_{i,1}, \ldots, e_{i,n}\}$*,* $i \in [k]$ *is contained in a surjective multi-homomorphism from* $\mathbf{A}$ *to* $\mathbf{E}$*.*

Proof. For every function h from $[m]$ to $[k]$ we take a formula $\phi_h(x_{1,1}, \ldots, x_{k,n}, z_1, \ldots, z_m)$ such that, for any structure $\mathbf{E}$ in the signature of $\mathbf{A}$, we have $\mathbf{E} \models \phi_h(e_{1,1}, \ldots, e_{k,n}, e'_1, \ldots, e'_m)$ if and only if the mapping $i \mapsto \{e_{i,1}, \ldots, e_{i,n}\} \cup \bigcup_{h(l)=i} e'_l$, $1 \leq i \leq k$, is a multi-homomorphism from $\mathbf{A}$ to $\mathbf{E}$. Such a formula can be obtained by directly translating the definition of a multi-homomorphism into the language of logic, similarly to (12.2).

We claim that the formula ϕ obtained by taking the disjunction of ϕ_h over all $h : [m] \to [k]$ and universally quantifying the variables $z_1, \ldots, z_m$ satisfies the requirement of the lemma, provided $|E| \leq m$. Indeed, on the one hand, if $\mathbf{E} \models \phi(e_{1,1}, \ldots, e_{k,n})$, then for every evaluation of the z

variables, some ϕ_h must be satisfied. We choose any evaluation that covers the whole set E (which is possible since $|E| \leq m$) and the satisfied disjunct ϕ_h then gives us the required surjective multi-homomorphism from $\mathbf{A}$ to $\mathbf{E}$ (by the choice of ϕ_h). On the other hand, if $i \mapsto \{e_{i,1}, \ldots, e_{i,n}\}$ is contained in a surjective multi-homomorphism f, then for any evaluation $\varepsilon_E(z_1)$, ..., $\varepsilon_E(z_m)$ of the universally quantified variables, a disjunct ϕ_h is satisfied whenever $\varepsilon_E(z_l) \in f(h(l))$ for every $l \in [m]$. Such an h exists since f is surjective. □

Proposition 12.8. *A pair* $(\mathbf{A}, \mathbf{B})$ *of similar structures is an* $\mathscr{L}$-PMC *template if and only if there exists a surjective multi-homomorphism from* $\mathbf{A}$ *to* $\mathbf{B}$.

Proof. For the forward implication, consider the sentence obtained by existentially quantifying all the variables in the formula ϕ provided by Lemma 12.7 (with $m \geq |A|, |B|$). This sentence is true in $\mathbf{A}$ (as there exists a surjective multi-homomorphism from $\mathbf{A}$ to $\mathbf{A}$ – the identity), so it must be true in $\mathbf{B}$, giving us a surjective multi-homomorphism from $\mathbf{A}$ to $\mathbf{B}$. The backward implication follows from Lemma 12.6. □

Example 12.9 (Surjectivity is relevant).

An example which shows that one cannot replace in Proposition 12.8 "surjective multi-homomorphism" by "(multi-)homomorphism" is the input formula $\varphi = \forall x \exists y R(x, y)$ ("there are no sinks") for a template where $\mathbf{A}$ is a digraph with no sinks and $\mathbf{B}$ is, say, $\mathbf{A}$ plus an isolated vertex. Clearly there is a (multi-)homomorphism from $\mathbf{A}$ to $\mathbf{B}$ (the inclusion function), however, φ witnesses that $(\mathbf{A}, \mathbf{B})$ is not an $\mathscr{L}$-PMC template.

The following characterization of promise definability is also a straightforward consequence of Lemmata 12.6 and 12.7.

Theorem 12.10. *Let* $(\mathbf{A}, \mathbf{B})$ *and* $(\mathbf{C}, \mathbf{D})$ *be* $\mathscr{L}$-PMC *templates such that* $A = C$ *and* $B = D$. *Then* $(\mathbf{C}, \mathbf{D})$ *is p-*$\mathscr{L}$*-definable from* $(\mathbf{A}, \mathbf{B})$ *if and only if* $\mathrm{SMuHom}(\mathbf{A}, \mathbf{B}) \subseteq \mathrm{SMuHom}(\mathbf{C}, \mathbf{D})$. *Moreover, in this case,* $\mathscr{L}\text{-PMC}(\mathbf{C}, \mathbf{D}) \leq \mathscr{L}\text{-PMC}(\mathbf{A}, \mathbf{B})$.

Proof. The theorem is proved in the same way as Theorem 12.3; using Lemma 12.6 instead of Lemma 12.1 for the forward implication, and the formula provided by Lemma 12.7 instead of (12.2) for the backward implication. □

12.2.3 Membership

Clearly, every $\mathscr{L}$-MC, as well as $\mathscr{L}$-PMC, is in PSPACE. We now give a generalization of the remaining membership results from [MM11] using an appropriate generalization of the notions of "A-shops" and "E-shops" from that same paper. We say that a surjective multi-homomorphism from $\mathbf{A}$ to $\mathbf{B}$ is an A-smuhom if there exists $a^* \in A$ such that $f(a^*) = B$. We also say that $(\mathbf{A}, \mathbf{B})$ admits an A-smuhom in this case. We call f an E-smuhom if $f^{-1}(b^*) = A$ for some $b^* \in B$. Finally, we call f an AE-smuhom if it is simultaneously an A-smuhom and an E-smuhom.

An additional simple reduction, in the spirit of the reductions given by *homomorphic relaxation* much used in the study of PCSPs, will be useful in the proof of the membership result (Theorem 12.12), and in the proofs of some of the hardness results later on. We say that an $\mathscr{L}$-PMC template $(\mathbf{C}, \mathbf{D})$ is a *relaxation* of an $\mathscr{L}$-PMC template $(\mathbf{A}, \mathbf{B})$ if $(\mathbf{C}, \mathbf{A})$ and $(\mathbf{B}, \mathbf{D})$ are $\mathscr{L}$-PMC templates. Recall that, by Proposition 12.8, this property is equivalent to the existence of surjective multi-homomorphisms from $\mathbf{C}$ to $\mathbf{A}$ and from $\mathbf{B}$ to $\mathbf{D}$.

Proposition 12.11. *Let* $(\mathbf{A}, \mathbf{B})$ *and* $(\mathbf{C}, \mathbf{D})$ *be* $\mathscr{L}$-PMC *templates. If* $(\mathbf{C}, \mathbf{D})$ *is a relaxation of* $(\mathbf{A}, \mathbf{B})$*, then* $\mathscr{L}$-PMC$(\mathbf{C}, \mathbf{D}) \leq \mathscr{L}$-PMC$(\mathbf{A}, \mathbf{B})$*.*

Proof. The trivial reduction, which does not change the input, works. Indeed, Yes instances of $\mathscr{L}$-PMC$(\mathbf{C}, \mathbf{D})$ are Yes instances of $\mathscr{L}$-PMC$(\mathbf{A}, \mathbf{B})$ since $(\mathbf{C}, \mathbf{A})$ is an $\mathscr{L}$-PMC template, and No instances of $\mathscr{L}$-PMC$(\mathbf{C}, \mathbf{D})$ are No instances of $\mathscr{L}$-PMC$(\mathbf{A}, \mathbf{B})$ since $(\mathbf{B}, \mathbf{D})$ is an $\mathscr{L}$-PMC template. □

Theorem 12.12. *Let* $(\mathbf{A}, \mathbf{B})$ *be an* $\mathscr{L}$-PMC *template. Then the following holds.*

1. *If* $(\mathbf{A}, \mathbf{B})$ *admits an* A-smuhom*, then* $\mathscr{L}$-PMC$(\mathbf{A}, \mathbf{B})$ *is in* NP*.*

2. *If* $(\mathbf{A}, \mathbf{B})$ *admits an* E-smuhom*, then* $\mathscr{L}$-PMC$(\mathbf{A}, \mathbf{B})$ *is in* coNP*.*

3. *If* $(\mathbf{A}, \mathbf{B})$ *admits an* AE-smuhom*, then* $\mathscr{L}$-PMC$(\mathbf{A}, \mathbf{B})$ *is in* L*.*

Proof. For the first item, let f be an A-smuhom from $\mathbf{A}$ to $\mathbf{B}$ with $f(a^*) = B$, and consider an input ϕ in the special form (12.3), i.e., $\phi = \forall y_1 \exists z_1 \forall y_2 \exists z_2 \dots \forall y_m \exists z_m \, \phi'(\mathbf{y}, \mathbf{z})$, where ϕ' is quantifier-free. We answer Yes if there exists $\mathbf{a} \in A^m$ such that $\mathbf{A} \models \phi'(a^*, a^*, \dots, a^*, \mathbf{a})$; this can be clearly decided in NP. It is clear that the answer is Yes whenever ϕ is a Yes instance of $\mathscr{L}$-PMC$(\mathbf{A}, \mathbf{B})$. On the other hand, if $\mathbf{A} \models \phi'(a^*, \dots, a^*, \mathbf{a})$, then any functions $\beta_1 : B \to B$, $\dots$, $\beta_m : B^m \to B$ such that $\beta_i(b_1, \dots, b_i) \in f(a_i)$ (for all $i \in [m]$ and $b_1, \dots, b_m \in B$) provide witnesses for $\mathbf{B} \models \phi$ by Lemma 12.1. Therefore, if ϕ is a No instance, then the answer is No, as needed.

The second item follows by the duality argument.

In the case $\mathbf{A} = \mathbf{B}$, the third item can be proved in an analogous way (by eliminating both quantifiers instead of just one), see Corollary 9 in [MM11]. For the general case, we will construct $\mathbf{C}$ such that there is an AE-smuhom from $\mathbf{C}$ to $\mathbf{C}$ and there are surjective multi-homomorphisms from $\mathbf{A}$ to $\mathbf{C}$ and from $\mathbf{C}$ to $\mathbf{B}$. Then $(\mathbf{A}, \mathbf{B})$ will be a relaxation of $(\mathbf{C}, \mathbf{C})$ by Proposition 12.8, and then membership of $\mathscr{L}$-PMC$(\mathbf{A}, \mathbf{B})$ in L will follow from Proposition 12.11 and the mentioned Corollary 9 in [MM11]. Let f be an AE-smuhom from $\mathbf{A}$ to $\mathbf{B}$ with $f(a^*) = B$ and $f^{-1}(b^*) = A$, and define a surjective multi-valued function f' from A to B by $f'(a^*) = B$ and $f'(a) = \{b^*\}$ if $a \neq a^*$. Note that f' is contained in f, so f' is a surjective multi-homomorphism from $\mathbf{A}$ to $\mathbf{B}$. We define $\mathbf{C}$ as the "image" of $\mathbf{A}$ under f', that is, $C = B$ and $R^{\mathbf{C}} = \cup_{\mathbf{a} \in R^{\mathbf{A}}} f'(\mathbf{a})$ for each relation symbol R. Clearly, f' is a surjective multi-homomorphism from $\mathbf{A}$ to $\mathbf{C}$ and the identity is a surjective homomorphism from $\mathbf{C}$ to $\mathbf{B}$. It remains to find an AE-smuhom from $\mathbf{C}$ to $\mathbf{C}$. We claim that g defined by $g(b^*) = \{b^*\}$ and $g(c) = C$ for $c \neq b^*$ is such an AE-smuhom. Indeed, if $\mathbf{c} \in R^{\mathbf{C}}$, then $\mathbf{c} \in f'(\mathbf{a})$ for some $\mathbf{a} \in R^{\mathbf{A}}$. By the definition of f', we necessarily have $a_i = a^*$ whenever $c_i \neq b^*$; therefore, $f'(\mathbf{a}) \supseteq g(\mathbf{c})$. But $f'(\mathbf{a}) \subseteq R^{\mathbf{C}}$ as $f' \in \mathrm{SMuHom}(\mathbf{A}, \mathbf{C})$, and we are done. □

These membership results together with the (more involved) hardness results were sufficient for the tetrachotomy in [MM11]. One problem with

generalizing this tetrachotomy is that, unlike in the non-promise setting, an $\mathscr{L}$-PMC template can admit an A-smuhom and an E-smuhom, but no AE-smuhom (see Chapter 13 for examples) and therefore we have no clear upper bounds in this case other than membership in NP$\cap$coNP. However, such a situation cannot happen for digraphs.

Proposition 12.13. *Let* $(\mathbf{A}, \mathbf{B})$ *be an* $\mathscr{L}$-PMC *template such that* $\mathbf{A}$ *and* $\mathbf{B}$ *are digraphs. If* $(\mathbf{A}, \mathbf{B})$ *admits an* A-smuhom *and an* E-smuhom*, then it admits an* AE-smuhom*.*

Proof. Denote by R the unique binary symbol in the signature. Let f be an A-smuhom from $\mathbf{A}$ to $\mathbf{B}$ with $f(a^*) = B$ and let g be an E-smuhom from $\mathbf{A}$ to $\mathbf{B}$ with $g^{-1}(b^*) = A$.

If a^* is isolated in $\mathbf{A}$ (i.e., $(a, a^*), (a^*, a) \notin R^{\mathbf{A}}$ for every $a \in A$), then we define a surjective multi-valued function h by $h(a^*) = B$ and $h(a) = \{b^*\}$ for every $a \neq a^*$. It is a multi-homomorphism from $\mathbf{A}$ to $\mathbf{B}$ since for any $(a, a') \in R^{\mathbf{A}}$, we have $h(a, a') = \{(b^*, b^*)\}$, which is contained in $R^{\mathbf{B}}$ because $R^{\mathbf{A}}$ is nonempty, so $g(R^{\mathbf{A}}) \ni (b^*, b^*)$.

Suppose next that there is an edge $(a_1, a^*) \in R^{\mathbf{A}}$ but a^* has no outgoing edges in $\mathbf{A}$. Let b_1 be an arbitrary element from $f(a_1)$ and define h by $h(a^*) = B$ and $h(a) = \{b_1\}$ for every $a \neq a^*$. To verify that $h \in \mathrm{SMuHom}(\mathbf{A}, \mathbf{B})$, consider an edge $(a, a') \in R^{\mathbf{A}}$. As a^* has no outgoing edges in $\mathbf{A}$, we get $a \neq a^*$, so $h(a) = \{b_1\}$. Now $h(a, a') \subseteq \{b_1\} \times B$, which is contained in $R^{\mathbf{B}}$ because $R^{\mathbf{B}} \supseteq f(a_1, a^*) \supseteq \{b_1\} \times B$.

If a^* has an outgoing edge $(a^*, a_1) \in R^{\mathbf{A}}$ but no incoming edges, we proceed similarly, defining $h(a^*) = B$ and $h(a) = \{b_1\}$ for all $a \neq a^*$, where b_1 is an arbitrary element from $f(a_1)$.

Finally, suppose that $(a_1, a^*) \in R^{\mathbf{A}}$ and $(a^*, a_2) \in R^{\mathbf{A}}$ for some $a_1, a_2 \in A$. If there is an element $a_3 \in A$ with no outgoing (resp., incoming) edges, define h by $h(a_3) = B$ and $h(a) = \{b'\}$ for all $a \neq a_3$, where b' is an arbitrary element from $f(a_1)$ (resp., $f(a_2)$). If there is no such element a_3, then we define $h(a^*) = B$ and $h(a) = \{b^*\}$ for all $a \neq a^*$. Since g is surjective, and every $a \in A$ has both an incoming and an outgoing edge, then $(b, b^*) \in R^{\mathbf{B}}$ and $(b^*, b) \in R^{\mathbf{B}}$ for all $b \in B$, therefore, $h \in \mathrm{SMuHom}(\mathbf{A}, \mathbf{B})$. □

12.2.4 Hardness

As a consequence of Theorems 12.4 and 12.10, we obtain the following hardness result.

Theorem 12.14. *Let* $(\mathbf{A}, \mathbf{B})$ *be an* $\mathscr{L}$-PMC *template.*

1. *If* $(\mathbf{A}, \mathbf{B})$ *admits no* E-smuhoms*, then* $\mathscr{L}$-PMC$(\mathbf{A}, \mathbf{B})$ *is* NP*-hard.*
2. *If* $(\mathbf{A}, \mathbf{B})$ *admits no* A-smuhoms*, then* $\mathscr{L}$-PMC$(\mathbf{A}, \mathbf{B})$ *is* coNP*-hard.*

Proof. If there are no E-smuhoms from $\mathbf{A}$ to $\mathbf{B}$, then $\mathrm{SMuHom}(\mathbf{A}, \mathbf{B})$ is contained in $\mathrm{SMuHom}((A; \mathrm{Rb}_A^{2|A|}), (B; \mathrm{NAE}_B^{2|A|}))$. Theorem 12.4 and Theorem 12.10 then imply the first item. The second item follows by the duality argument. □

In the non-promise setting, the absence of A-smuhoms and E-smuhoms is sufficient for PSPACE-hardness [MM11, MM18]. This most involved part of the tetrachotomy result seems much more challenging in the promise setting and we do not have strong reasons to believe that templates without A-smuhoms and E-smuhoms are necessarily PSPACE-hard. Nevertheless, we are able to prove some additional hardness results which will cover all the extensions of $\mathscr{L}$.

Proposition 12.15. $\mathscr{L}$-PMC$((A; =_A), (B; =_B))$ *is* PSPACE*-hard for any* A*,* B *such that* $|A| \geq |B| \geq 2$*.*

Note here that the surjective multi-homomorphisms from $(A; =_A)$ to $(B; =_B)$ are exactly the surjective multi-valued functions from A to B of multiplicity one. In particular, if $|A| < |B|$, then $((A; =_A), (B; =_B))$ is not an $\mathscr{L}$-PMC template.

Proof. We start by noticing that the template $((A; =_A), ([2]; =_{[2]}))$ is a relaxation of $(\mathbf{A}, \mathbf{B}) := ((A; =_A), (B; =_B))$. Therefore, by Proposition 12.11, it is enough to prove the claim in the case $B = [2]$. For simplicity, we assume that $A = [k]$ $(k \geq 2)$. We prove the PSPACE-hardness by a reduction from $\mathscr{L}$-MC$(\mathbf{B})$, a PSPACE-hard problem by, e.g., [Mar08]. Consider an input ϕ to $\mathscr{L}$-MC$(\mathbf{B})$ in the special form (12.3), i.e., $\phi = \forall y_1 \exists z_1 \forall y_2 \exists z_2 \dots \forall y_m \exists z_m \; \phi'(\mathbf{y}, \mathbf{z})$, where ϕ' is quantifier-free. We need to

find a log-space computable formula ψ such that $\mathbf{B} \models \phi$ implies $\mathbf{A} \models \psi$ (so that Yes instances of $\mathscr{L}$-MC($\mathbf{B}$) are transformed to Yes instances of $\mathscr{L}$-PMC($\mathbf{A}, \mathbf{B}$)) and $\mathbf{B} \models \psi$ implies $\mathbf{B} \models \phi$ (so that No instances are transformed to No instances).

The rough idea to construct ψ is to reinterpret the values in $A = [k]$ as values in $B = [2]$ via a mapping $A \to B$. We set

$$\psi = \forall x_1 \forall x_2 \; \exists x_3 \exists x_4 \ldots \exists x_k \;\; (x_1 = x_2) \;\vee \bigwedge_{f:A\to B} \rho_f, \qquad \text{where} \tag{12.4}$$

$$\rho_f = (\forall y'_1 \exists z_1 \ldots \forall y'_m \exists z_m) \;\; (\exists y_1 \ldots \exists y_m) \;\; \left(\bigwedge_{i=1}^{m} \sigma[f, y'_i, y_i] \right) \wedge \phi'(\mathbf{y}, \mathbf{z}), \tag{12.5}$$

$$\sigma[f, y'_i, y_i] = \bigvee_{a \in A} \Big((y'_i = x_a) \wedge (y_i = x_{f(a)}) \Big). \tag{12.6}$$

Observe first that ψ can be constructed from ϕ in logarithmic space.

Next, we verify that $\mathbf{B} \models \psi$ implies $\mathbf{B} \models \phi$. So, we suppose $\mathbf{B} \models \psi$ and aim to find witnesses $\beta_1, \ldots, \beta_m$ for $\mathbf{B} \models \phi$; to this end, let $\mathbf{c}$ be some tuple in B^m that corresponds to evaluations of universally quantified variables in ϕ. We evaluate the variables x_1 and x_2 in ψ as $\varepsilon_B(x_1) = 1$ and $\varepsilon_B(x_2) = 2$, and pick an evaluation $\varepsilon_B(x_3), \ldots, \varepsilon_B(x_k)$ making ψ true in $\mathbf{B}$. Set $f(a) = \varepsilon_B(x_a)$, $a \in A$. The first disjunct of (12.4) is not satisfied, so ρ_f is satisfied with this choice of ε_B. When it is the turn to evaluate y'_i, we set $\varepsilon_B(y'_i) = c_i$ and define $\beta_i(c_1, \ldots, c_i) = \varepsilon_B(z_i)$, where $\varepsilon_B(z_i)$ is a satisfactory evaluation of z_i. Inspecting the definition (12.6), we see that $y_1, \ldots, y_m$ are necessarily evaluated as $\varepsilon_B(y_1) = c_1, \ldots, \varepsilon_B(y_m) = c_m$: indeed, if a disjunct $(y'_i = x_a) \wedge (y_i = x_{f(a)})$ is satisfied, then $c_i = \varepsilon_B(y'_i) = \varepsilon_B(x_a)$ and $\varepsilon_B(y_i) = \varepsilon_B(x_{f(a)}) = \varepsilon_B(x_{\varepsilon_B(x_a)}) = \varepsilon_B(x_a)$; in particular, $\varepsilon_B(y_i) = c_i$. Therefore, the conjunct $\phi'(\mathbf{y}, \mathbf{z})$ in (12.5) ensures $\mathbf{B} \models \phi'(\mathbf{c}, \beta_1(c_1), \ldots, \beta_m(c_1, \ldots, c_m))$. As $\mathbf{c}$ was chosen arbitrarily, we get that $\beta_1, \ldots, \beta_m$ are witnesses for $\mathbf{B} \models \phi$, as required.

We now suppose that $\beta_1, \ldots, \beta_m$ are witnesses for $\mathbf{B} \models \phi$, and aim to show that $\mathbf{A} \models \psi$. Because of the first disjunct of (12.4), it is enough to consider only evaluations of x_1 and x_2 with $\varepsilon_A(x_1) \neq \varepsilon_A(x_2)$. Since any bijection, regarded as a surjective multi-homomorphism from $\mathbf{A}$ to $\mathbf{A}$ of multiplicity one, preserves $\mathscr{L}$-formulas (in the sense of Lemma 12.6), then

we can as well assume that $\varepsilon_A(x_1) = 1$ and $\varepsilon_A(x_2) = 2$. We evaluate the remaining x variables as $\varepsilon_A(x_a) = a$, $a = 3, 4, \ldots, k$. We take a function $f : A \to B$ and argue that ρ_f is satisfied in $\mathbf{A}$. Given a selection of $\varepsilon_A(y'_i)$, we evaluate z_i as $\varepsilon_A(z_i) = \beta_i(f(\varepsilon_A(y'_1)), \ldots, f(\varepsilon_A(y'_i)))$, and we define the evaluation of the remaining variables by $\varepsilon_A(y_i) = f(\varepsilon_A(y'_i))$. With these choices, each $\sigma[f, y'_i, y_i]$ is satisfied because of the disjunct $a = \varepsilon_A(y'_i)$ in (12.6). The second conjunct in (12.5), $\phi'(\mathbf{y}, \mathbf{z})$, is also satisfied: we know $\mathbf{B} \models \phi'(\mathbf{c}, \beta_1(c_1), \ldots, \beta_m(c_1, \ldots, c_m))$ in particular for $c_1 = f(\varepsilon_A(y'_1))$, $\ldots$, $c_m = f(\varepsilon_A(y'_m))$ and, with this $\mathbf{c}$, it is apparent from the choice of evaluations that $\mathbf{B} \models \phi'(\mathbf{c}, \beta_1(c_1), \ldots, \beta_m(c_1, \ldots, c_m))$ is equivalent to $\mathbf{A} \models \phi'(\varepsilon_A(y_1), \ldots, \varepsilon_A(y_m), \varepsilon_A(z_1), \ldots, \varepsilon_A(z_m))$. The proof of $\mathbf{A} \models \psi$ is concluded. □

It follows that $\{\exists, \forall, \wedge, \vee, =\}$-PMC over any template is PSPACE-hard and so is, by the duality argument, $\{\exists, \forall, \wedge, \vee, \neq\}$-PMC. The next proposition implies PSPACE-hardness for $\{\exists, \forall, \wedge, \vee, \neg\}$-PMC.

Proposition 12.16. *Let* $(\mathbf{A}, \mathbf{B})$ *be an* $\mathscr{L}$-PMC *template which is closed under complementation. Then* $\mathscr{L}$-PMC$(\mathbf{A}, \mathbf{B})$ *is* PSPACE*-hard.*

Proof. Suppose that $(\mathbf{A}, \mathbf{B})$ is closed under complementation. We define an equivalence relation $\sim_A$ on A by considering two elements equivalent if they play the same role in every relation of $\mathbf{A}$. Formally, $a \sim a'$ if for every symbol R from the signature, every coordinate $i \in [\mathrm{ar}(R)]$, and every $\mathbf{c}, \mathbf{c}' \in A^{\mathrm{ar}(R)}$, if $c_i = a$, $c'_i = a'$, $c_j = c'_j$ for all $j \in [\mathrm{ar}(R)] \setminus \{i\}$, and $\mathbf{c} \in R^{\mathbf{A}}$, then $\mathbf{c}' \in R^{\mathbf{A}}$. We define an equivalence relation $\sim_B$ on B analogously. Notice that $\sim_A$ and $\sim_B$ are indeed equivalence relations, and let m and n denote the number of equivalence classes of $\sim_A$ and $\sim_B$ respectively. Observe that $m, n \geq 2$, as otherwise, any nonempty relation in the corresponding template would contain all the tuples, and we do not allow such structures in this context.

Let $\mathbf{C} = (A; \sim_A)$ and $\mathbf{D} = (B; \sim_B)$. We claim that every surjective multi-homomorphism f from $\mathbf{A}$ to $\mathbf{B}$ preserves $\sim$, i.e., is a surjective multi-homomorphism from $\mathbf{C}$ to $\mathbf{D}$. Consider $a, a' \in A$, and $b, b' \in B$ such that $a \sim_A a'$, $b \in f(a)$, and $b' \in f(a')$. In order to prove $b \sim_B b'$, take arbitrary R, i, $\mathbf{d}$, $\mathbf{d}'$ such that $d_i = b$, $d'_i = b'$, $d_j = d'_j$ for all $j \neq i$, and $\mathbf{d} \in R^{\mathbf{B}}$. Let $\mathbf{c}, \mathbf{c}' \in A^{\mathrm{ar}(R)}$ be tuples such that $c_i = a$, $c'_i = a'$, and $c_j = c'_j \in f^{-1}(d_j)$

for all $j \neq i$ (which exist as f is surjective). If $\mathbf{c} \notin R^{\mathbf{A}}$, then $\mathbf{c} \in \overline{R}^{\mathbf{A}}$ and, consequently, $\mathbf{d} \in f(\mathbf{c}) \subseteq \overline{R}^{\mathbf{B}}$ (as f is a surjective multi-homomorphism from $\mathbf{A}$ to $\mathbf{B}$), a contradiction with $\mathbf{d} \in R^{\mathbf{B}}$. Therefore, $\mathbf{c} \in R^{\mathbf{A}}$ and also $\mathbf{c}' \in R^{\mathbf{A}}$ as $a \sim_A a'$. Now $\mathbf{d}' \in f(\mathbf{c}') \subseteq R^{\mathbf{B}}$, and $b \sim_B b'$ follows.

By Theorem 12.10, $\mathscr{L}\text{-PMC}(\mathbf{C}, \mathbf{D}) \leq \mathscr{L}\text{-PMC}(\mathbf{A}, \mathbf{B})$. Since there exists a surjective multi-valued function from A to B that preserves $\sim$ (namely, any $f \in \text{SMuHom}(\mathbf{A}, \mathbf{B})$), we also know that $m \geq n$. The template $(\mathbf{E}, \mathbf{F}) := (([m]; =_{[m]}), ([n]; =_{[n]})$ is a relaxation of $(\mathbf{C}, \mathbf{D})$, because there exists a surjective multi-homomorphism from $([m]; =_{[m]})$ to $\mathbf{C}$ (a multi-valued function that maps i to the i-th equivalence class of $\sim_A$ under an arbitrary linear ordering of classes) and a surjective multi-homomorphism from $\mathbf{D}$ to $([n]; =_{[n]})$ (a "multi"-valued function that maps every element in the i-th equivalence class of $\sim_B$ to $\{i\}$). By Proposition 12.11, $\mathscr{L}\text{-PMC}(\mathbf{E}, \mathbf{F}) \leq \mathscr{L}\text{-PMC}(\mathbf{C}, \mathbf{D})$; therefore, $\mathscr{L}\text{-PMC}(\mathbf{E}, \mathbf{F}) \leq \mathscr{L}\text{-PMC}(\mathbf{A}, \mathbf{B})$. But $\mathscr{L}\text{-PMC}(\mathbf{E}, \mathbf{F})$ is PSPACE-hard by Proposition 12.15, so $\mathscr{L}\text{-PMC}(\mathbf{A}, \mathbf{B})$ is PSPACE-hard, too. □

Finally, the following reduction puts together the two main techniques that we have used to prove both hardness and tractability conditions in this work, namely, gadget reductions and relaxations.

Lemma 12.17. *Let* $(\mathbf{A}, \mathbf{B})$, $(\mathbf{C}, \mathbf{D})$ *be* $\mathscr{L}$-PMC *templates and* $c : C \to A$, $d : B \to D$ *be surjective multi-valued functions such that for all* $f \in \text{SMuHom}(\mathbf{A}, \mathbf{B})$, *the composition* $dfc \in \text{SMuHom}(\mathbf{C}, \mathbf{D})$. *Then,* $\mathscr{L}\text{-PMC}(\mathbf{C}, \mathbf{D}) \leq \mathscr{L}\text{-PMC}(\mathbf{A}, \mathbf{B})$.

Proof. It is sufficient to construct two structures $\mathbf{A}'$, $\mathbf{B}'$ on the same signature as $(\mathbf{C}, \mathbf{D})$ and with domains $A' = A$, $B' = B$, such that $c \in \text{SMuHom}(\mathbf{C}, \mathbf{A}')$, $d \in \text{SMuHom}(\mathbf{B}', \mathbf{D})$ (hence $(\mathbf{C}, \mathbf{D})$ is a relaxation of $(\mathbf{A}', \mathbf{B}')$) and $\text{SMuHom}(\mathbf{A}, \mathbf{B}) \subseteq \text{SMuHom}(\mathbf{A}', \mathbf{B}')$. Then, by Proposition 12.11 and Theorem 12.10, we have that

$$\mathscr{L}\text{-PMC}(\mathbf{C}, \mathbf{D}) \leq \mathscr{L}\text{-PMC}(\mathbf{A}', \mathbf{B}') \leq \mathscr{L}\text{-PMC}(\mathbf{A}, \mathbf{B}).$$

In particular, for each relation symbol R in the signature of $(\mathbf{C}, \mathbf{D})$, we define its interpretation in $\mathbf{A}'$, $\mathbf{B}'$ respectively as

$$R^{\mathbf{A}'} = c(R^{\mathbf{C}}); \qquad R^{\mathbf{B}'} = d^{-1}(R^{\mathbf{D}}).$$

The result clearly follows from this definition. □

Example 12.18 (Partitions for PSPACE-hardness).

Here we show an example application of Lemma 12.17. In particular, let $(\mathbf{A}, \mathbf{B})$ be an $\mathscr{L}$-PMC template that satisfies the following condition: there exist partitions $\mathcal{A} = (A_1, \dots, A_n)$ of A and $\mathcal{B} = (B_1, \dots, B_m)$ of B with $2 \leq m \leq n$ such that, for all $f \in \mathrm{SMuHom}(\mathbf{A}, \mathbf{B})$ and for all $i \in [n]$, there exists some $j \in [m]$ such that $f(A_i) \subseteq B_j$. Then, we can apply Lemma 12.17 to show that $(\mathbf{A}, \mathbf{B})$ is PSPACE-hard. In particular, it is sufficient to pick $(\mathbf{C}, \mathbf{D}) = (([n]; =_{[n]}), ([m]; =_{[m]}))$, which is PSPACE-hard by Proposition 12.15. Then, the multi-valued functions

$$c : [n] \to A,\ c(i) = A_i \quad \text{and} \quad d : B \to [m],\ d(b) = j \text{ for all } b \in B_j$$

are such that the multiplicity of *dfc* is 1 for all $f \in \mathrm{SMuHom}(\mathbf{A}, \mathbf{B})$ and hence they satisfy the conditions of Lemma 12.17.

Similarly, by choosing $(\mathbf{C}, \mathbf{D}) = (([n]; \neq_{[n]}), ([m]; \neq_{[m]}))$, we can show that $(\mathbf{A}, \mathbf{B})$ is PSPACE-hard whenever there exist partitions $\mathcal{A} = (A_1, \dots, A_n)$ of A and $\mathcal{B} = (B_1, \dots, B_m)$ of B with $2 \leq n \leq m$ such that, for all $f \in \mathrm{SMuHom}(\mathbf{A}, \mathbf{B})$ and for all $j \in [m]$, there exists a unique $i \in [n]$ such that $B_j \subseteq f(A_i)$.

12.2.5 Summary

The claims stated in Figure 11.2 are now immediate consequences of the obtained results. Note that the claims remain true without the imposed restrictions on structures (i.e., we can allow singleton universes, improper relations, etc.); the only nontrivial ingredient is the L-membership of the Boolean Sentence Value Problem [Lyn77]. We summarize our results in Corollary 12.19.

Corollary 12.19 ($\mathscr{L} = \{\exists, \forall, \wedge, \vee\}$)**.** *Let* $(\mathbf{A}, \mathbf{B})$ *be an* $\mathscr{L}$-PMC *template.*

1. *If* $(\mathbf{A}, \mathbf{B})$ *admits an* $\mathsf{A}\mathsf{E}$-smuhom*, or the signature of* $\mathbf{A}$ *and* $\mathbf{B}$ *contains a single binary symbol and* $(\mathbf{A}, \mathbf{B})$ *admits both an* A-smuhom *and an* E-smuhom*, then* $\mathscr{L}$-PMC$(\mathbf{A}, \mathbf{B})$ *is in* L.

2. *If* $(\mathbf{A}, \mathbf{B})$ *admits both an* A-smuhom *and an* E-smuhom*, then* $\mathscr{L}$-PMC $(\mathbf{A}, \mathbf{B})$ *is in* NP $\cap$ coNP*.*

3. *If* $(\mathbf{A}, \mathbf{B})$ *admits an* A-smuhom *but no* E-smuhom*, then* $\mathscr{L}$-PMC$(\mathbf{A}, \mathbf{B})$ *is* NP*-complete.*

4. *If* $(\mathbf{A}, \mathbf{B})$ *admits an* E-smuhom *but no* A-smuhom*, then* $\mathscr{L}$-PMC$(\mathbf{A}, \mathbf{B})$ *is* coNP*-complete.*

5. *If* $(\mathbf{A}, \mathbf{B})$ *admits no* A-smuhom *and no* E-smuhom*, then* $\mathscr{L}$-PMC$(\mathbf{A}, \mathbf{B})$ *is* NP*-hard and* coNP*-hard.*

6. $\{\exists, \forall, \wedge, \vee, =\}$-PMC$(\mathbf{A}, \mathbf{B})$*,* $\{\exists, \forall, \wedge, \vee, \neq\}$-PMC$(\mathbf{A}, \mathbf{B})$*, and* $\{\exists, \forall, \wedge, \vee, \neg\}$-PMC$(\mathbf{A}, \mathbf{B})$ *are* PSPACE*-complete whenever* $|A|, |B| \geq 2$*, and in* L *otherwise (for* $\{\exists, \forall, \wedge, \vee, \neg\}$ *we require a nonempty signature).*

We observe that the results imply a complete complexity classification in the case that one of the two template structures is *Boolean*, i.e., has a two-element universe.

Corollary 12.20 ($\mathscr{L} = \{\exists, \forall, \wedge, \vee\}$)**.** *Let* $(\mathbf{A}, \mathbf{B})$ *be an* $\mathscr{L}$-PMC *template.*

1. *If* $\mathbf{B}$ *is Boolean, then* $\mathscr{L}$-PMC$(\mathbf{A}, \mathbf{B})$ *is in* L*, or is* NP*-complete, or* PSPACE*-complete.*

2. *If* $\mathbf{A}$ *is Boolean, then* $\mathscr{L}$-PMC$(\mathbf{A}, \mathbf{B})$ *is in* L*, or is* coNP*-complete, or* PSPACE*-complete.*

3. *If* $\mathbf{A}$ *and* $\mathbf{B}$ *are Boolean, then* $\mathscr{L}$-PMC$(\mathbf{A}, \mathbf{B})$ *is in* L*, or is* PSPACE*-complete.*

Proof. If $\mathbf{B}$ is Boolean, then every E-smuhom (from $\mathbf{A}$ to $\mathbf{B}$) is also an AE-smuhom. Moreover, if there is no A-smuhom, then every surjective multi-homomorphism is of multiplicity one, so it is also a multi-homomorphism from $(A; =_A)$ to $(B; =_B)$. The first item now follows from Proposition 12.15 and Theorem 12.10. The other items are shown in a similar manner. □

13 Conclusion

In this part of the thesis, we initiated the study of the fixed-template promise model checking problem for the existential positive ($\{\exists, \wedge, \vee\}$) and the positive ($\{\exists, \forall, \wedge, \vee\}$) fragments of first-order logic. We gave a full complexity classification of $\{\exists, \wedge, \vee\}$-PMC, initiated an algebraic approach to $\{\exists, \forall, \wedge, \vee\}$-PMC, and applied it to provide several complexity results about this class of problems.

There are two wide gaps left for further investigation. First, it is unclear what the complexity is for the $\{\exists, \forall, \wedge, \vee\}$-PMC over templates that admit both an A-smuhom and an E-smuhom, but no AE-smuhom. While there are no such templates for digraphs (as shown in Proposition 12.13), there are examples with one ternary or two binary relations, such as the following. We use ij as a shortcut for the pair (i, j).

$\mathbf{A} = ([3]; \{(1,2,3)\}), \quad \mathbf{B} = ([3]; \{1,2,3\} \times \{2\} \times \{3\} \cup \{1,2\} \times \{2\} \times \{2,3\})$
$\mathbf{A} = ([3]; \{12\}, \{13\}), \quad \mathbf{B} = ([3]; \{12, 22, 32\}, \{12, 13, 22, 23, 33\})$

The second gap is between simultaneous NP- and coNP-hardness, and PSPACE-hardness, when the template admits neither an A-smuhom nor an E-smuhom. Examples with unknown complexity include the following.

$\mathbf{A} = ([3]; \{(1,2,3)\}), \quad \mathbf{B} = ([3]; \{2,3\} \times \{1,3\} \times \{1,2\})$
$\mathbf{A} = ([3]; \{(1,2,3)\}), \quad \mathbf{B} = ([3]; \{1,2\} \times \{1,2\} \times \{3\} \cup \{1,3\} \times \{2\} \times \{2\})$
$\mathbf{A} = ([4]; \{12, 34\}), \quad \mathbf{B} = ([4]; \{12, 13, 14, 23, 24, 34, 32\})$

Take for instance the binary example above (also pictured in Figure 13.1). One can check that there is no known PSPACE-hard template that

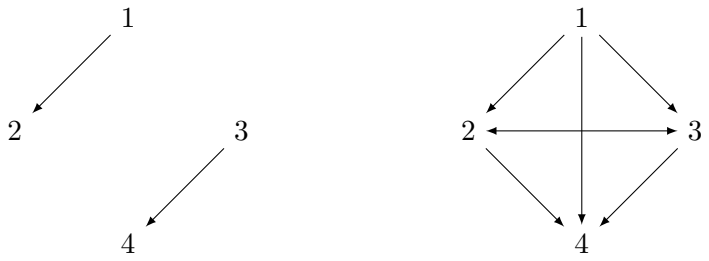

Figure 13.1: An example of a digraph template of unknown complexity.

we can reduce from applying Lemma 12.17; in particular, there are no partitions of A and B that satisfy the criteria described in Example 12.18.

The following equivalent *unary* version of the first example is an especially interesting template, whose $\{\exists, \forall, \wedge, \vee\}$-PMC is the problem described in the introduction.

$$\mathbf{A} = ([3];\ \{1\}, \{2\}, \{3\}), \quad \mathbf{B} = ([3];\ \{2,3\}, \{1,3\}, \{1,2\})$$

As for the theory-building, the next natural step is to capture more complex reductions by means of surjective multi-homomorphisms; namely, the analogue of pp-constructions, which proved to be so useful in the theory of (Promise) CSPs [BKW17, BBKO21]. Lemma 12.15 represents a first step in this direction. It may be also helpful to characterize and study the sets of surjective multi-homomorphisms in the spirit of [Mar10, CM21].

Bibliography

[ABB22] Kristina Asimi, Libor Barto, and Silvia Butti. Fixed-Template Promise Model Checking Problems. In Christine Solnon, editor, *28th International Conference on Principles and Practice of Constraint Programming, CP 2022, Haifa, Israel, August 2-5, 2022*, LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. (5)

[AD22] Albert Atserias and Víctor Dalmau. Promise Constraint Satisfaction and Width. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 1129–1153. SIAM, 2022. (14, 42, 51, 96)

[ADW17] Samson Abramsky, Anuj Dawar, and Pengming Wang. The pebbling comonad in finite model theory. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–12. IEEE Computer Society, 2017. (140)

[AGH17] Per Austrin, Venkatesan Guruswami, and Johan Håstad. $(2+\epsilon)$-Sat Is NP-hard. *SIAM J. Comput.*, 46(5):1554–1573, 2017. (13)

[AHK12] Sanjeev Arora, Elad Hazan, and Satyen Kale. The Multiplicative Weights Update Method: a Meta-Algorithm and Applications. *Theory of Computing*, 8(6):121–164, 2012. (55, 66, 67)

[AM13] Albert Atserias and Elitza Maneva. Sherali–Adams Relaxations and Indistinguishability in Counting Logics. *SIAM Journal on Computing*, 42(1):112–137, 2013. (41, 118, 119)

[Ang80] Dana Angluin. Local and Global Properties in Networks of Processors (Extended Abstract). In *Proceedings of the Twelfth Annual ACM Symposium on Theory of Computing*, STOC '80, page 82–93, New York, NY, USA, 1980. Association for Computing Machinery. (57, 91)

[Bab16] László Babai. Graph Isomorphism in Quasipolynomial Time [Extended Abstract]. In *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '16, page 684–697, New York, NY, USA, 2016. Association for Computing Machinery. (33)

[Bar16] Libor Barto. The collapse of the bounded width hierarchy. *Journal of Logic and Computation*, 26(3):923–943, 2016. (51)

[BB22] Libor Barto and Silvia Butti. Weisfeiler-Leman Invariant Promise Valued CSPs. In Christine Solnon, editor, *28th International Conference on Principles and Practice of Constraint Programming, CP 2022, Haifa, Israel, August 2-5, 2022*, LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. (4, 89, 95)

[BBB+21] Libor Barto, Zarathustra Brady, Andrei Bulatov, Marcin Kozik, and Dmitriy Zhuk. Minimal Taylor Algebras as a Common Framework for the Three Algebraic Approaches to the CSP. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*, pages 1–13. IEEE, 2021. (10)

[BBC+09] Ferdinand Börner, Andrei A. Bulatov, Hubie Chen, Peter Jeavons, and Andrei A. Krokhin. The complexity of constraint satisfaction games and QCSP. *Inf. Comput.*, 207(9):923–944, 2009. (16)

[BBJK03] Ferdinand Börner, Andrei A. Bulatov, Peter Jeavons, and Andrei A. Krokhin. Quantified Constraints: Algorithms and

Complexity. In Matthias Baaz and Johann A. Makowsky, editors, *Computer Science Logic, 17th International Workshop, CSL 2003, 12th Annual Conference of the EACSL, and 8th Kurt Gödel Colloquium, KGC 2003, Vienna, Austria, August 25-30, 2003, Proceedings*, volume 2803 of *Lecture Notes in Computer Science*, pages 58–70. Springer, 2003. (16)

[BBKO21] Libor Barto, Jakub Bulín, Andrei A. Krokhin, and Jakub Opršal. Algebraic Approach to Promise Constraint Satisfaction. *J. ACM*, 68(4):28:1–28:66, 2021. (14, 96, 178)

[BD21a] Silvia Butti and Víctor Dalmau. Fractional Homomorphism, Weisfeiler-Leman Invariance, and the Sherali-Adams Hierarchy for the Constraint Satisfaction Problem. In Filippo Bonchi and Simon J. Puglisi, editors, *46th International Symposium on Mathematical Foundations of Computer Science, MFCS 2021, August 23-27, 2021, Tallinn, Estonia*, volume 202 of *LIPIcs*, pages 27:1–27:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. (4, 5, 44, 86, 89)

[BD21b] Silvia Butti and Victor Dalmau. The Complexity of the Distributed Constraint Satisfaction Problem. In Markus Bläser and Benjamin Monmege, editors, *38th International Symposium on Theoretical Aspects of Computer Science, STACS 2021, March 16-19, 2021, Saarbrücken, Germany (Virtual Conference)*, volume 187 of *LIPIcs*, pages 20:1–20:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. (4)

[BD22] Silvia Butti and Víctor Dalmau. The Complexity of the Distributed Constraint Satisfaction Problem. *Theory of Computing Systems*, 2022. (3, 55)

[BES80] László Babai, Paul Erdős, and Stanley M. Selkow. Random Graph Isomorphism. *SIAM J. Comput.*, 9(3):628–635, 1980. (35, 115)

[BG19] Joshua Brakensiek and Venkatesan Guruswami. An Algorithmic Blend of LPs and Ring Equations for Promise CSPs. In

Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 436–455. SIAM, 2019. (13)

[BG20] Joshua Brakensiek and Venkatesan Guruswami. Symmetric Polymorphisms and Efficient Decidability of Promise CSPs. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 297–304. SIAM, 2020. (13)

[BG21] Joshua Brakensiek and Venkatesan Guruswami. Promise Constraint Satisfaction: Algebraic Structure and a Symmetric Boolean Dichotomy. *SIAM J. Comput.*, 50(6):1663–1700, 2021. (13)

[BGWŽ20] Joshua Brakensiek, Venkatesan Guruswami, Marcin Wrochna, and Stanislav Živný. The Power of the Combined Basic Linear Programming and Affine Relaxation for Promise Constraint Satisfaction Problems. *SIAM J. Comput.*, 49(6):1232–1248, 2020. (13, 14, 86)

[BHB+18] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinícius Flores Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Çaglar Gülçehre, H. Francis Song, Andrew J. Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey R. Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matthew Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *CoRR*, abs/1806.01261, 2018. (54)

[BJK05] Andrei A. Bulatov, Peter Jeavons, and Andrei A. Krokhin. Classifying the Complexity of Constraints Using Finite Algebras. *SIAM J. Comput.*, 34(3):720–742, 2005. (8, 9, 22)

[BK79] László Babai and Ludek Kucera. Canonical Labelling of Graphs in Linear Average Time. In *20th Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 29-31 October 1979*, pages 39–46. IEEE Computer Society, 1979. (35, 115)

[BK09] Libor Barto and Marcin Kozik. Constraint Satisfaction Problems of Bounded Width. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pages 595–603. IEEE Computer Society, 2009. (51)

[BK14] Libor Barto and Marcin Kozik. Constraint Satisfaction Problems Solvable by Local Consistency Methods. *J. ACM*, 61(1):3:1–3:19, 2014. (51)

[BK17] Libor Barto and Marcin Kozik. Absorption in Universal Algebra and CSP. In Andrei A. Krokhin and Stanislav Živný, editors, *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*, pages 45–77. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. (10)

[BKGS01] Ramon Bejar, Bhaskar Krishnamachari, Carla Gomes, and Bart Selman. Distributed constraint satisfaction in a wireless sensor tracking system. In *Workshop on Distributed Constraint Reasoning, International Joint Conference on Artificial Intelligence*, volume 4, 2001. (53)

[BKKR69a] V. G. Bodnarchuk, L. A. Kaluzhnin, V. N. Kotov, and B. A. Romov. Galois theory for Post algebras. I. *Cybernetics*, 5(3):243–252, 1969. (11)

[BKKR69b] V. G. Bodnarchuk, L. A. Kaluzhnin, V. N. Kotov, and B. A. Romov. Galois theory for Post algebras. II. *Cybernetics*, 5(5):531–539, 1969. (11)

[BKL08] Andrei A. Bulatov, Andrei Krokhin, and Benoit Larose. *Dualities for Constraint Satisfaction Problems*, pages 93–124. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. (49)

[BKM12] Manuel Bodirsky, Jan Kára, and Barnaby Martin. The complexity of surjective homomorphism problems - a survey. *Discret. Appl. Math.*, 160(12):1680–1690, 2012. (12)

[BKM+20] Pablo Barceló, Egor V. Kostylev, Mikaël Monet, Jorge Pérez, Juan L. Reutter, and Juan Pablo Silva. The Logical Expressiveness of Graph Neural Networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. (54)

[BKN09] Libor Barto, Marcin Kozik, and Todd Niven. The CSP Dichotomy Holds for Digraphs with No Sources and No Sinks (A Positive Answer to a Conjecture of Bang-Jensen and Hell). *SIAM J. Comput.*, 38(5):1782–1802, 2009. (9)

[BKO19] Jakub Bulín, Andrei A. Krokhin, and Jakub Oprsal. Algebraic approach to promise constraint satisfaction. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 602–613. ACM, 2019. (14)

[BKW17] Libor Barto, Andrei Krokhin, and Ross Willard. Polymorphisms, and How to Use Them. In Andrei Krokhin and Stanislav Živný, editors, *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*, pages 1–44. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2017. (10, 12, 151, 178)

[Bod21] Manuel Bodirsky. *Complexity of Infinite-Domain Constraint Satisfaction*. Lecture Notes in Logic. Cambridge University Press, 2021. (12)

[Bol82] Béla Bollobás. Distinguishing Vertices of Random Graphs. In Béla Bollobás, editor, *Graph Theory*, volume 62 of *North-Holland Mathematics Studies*, pages 33–49. North-Holland, 1982. (116)

[BOP18] Libor Barto, Jakub Opršal, and Michael Pinsker. The wonderland of reflections. *Israel Journal of Mathematics*, 223(1):363–398, 2018. (12, 119, 120)

[Bör08] Ferdinand Börner. Basics of Galois Connections. In Nadia Creignou, Phokion G. Kolaitis, and Heribert Vollmer, editors, *Complexity of Constraints - An Overview of Current Research Themes [Result of a Dagstuhl Seminar]*, volume 5250 of *Lecture Notes in Computer Science*, pages 38–67. Springer, 2008. (151)

[BSV+96] P. Boldi, S. Shammah, S. Vigna, B. Codenotti, P. Gemmell, and J. Simon. Symmetry Breaking in Anonymous Networks: Characterizations. In *ISTCS*, 1996. (57, 145)

[Bul06] Andrei A. Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-element set. *J. ACM*, 53(1):66–120, 2006. (9)

[Bul09] Andrei Bulatov. Bounded relational width. 2009. (51)

[Bul17] A. A. Bulatov. A Dichotomy Theorem for Nonuniform CSPs. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 319–330, Oct 2017. (10, 12, 150)

[BV01] Paolo Boldi and Sebastiano Vigna. An Effective Characterization of Computability in Anonymous Networks. In Jennifer L. Welch, editor, *Distributed Computing, 15th International Conference, DISC 2001, Lisbon, Portugal, October 3-5, 2001, Proceedings*, volume 2180 of *Lecture Notes in Computer Science*, pages 33–47. Springer, 2001. (57, 145)

[CCC+13] David A. Cohen, Martin C. Cooper, Páidí Creed, Peter G. Jeavons, and Stanislav Živný. An Algebraic Theory of Complexity for Discrete Optimization. *SIAM J. Comput.*, 42(5):1915–1939, 2013. (15, 100)

[CCJ06] David A. Cohen, Martin C. Cooper, and Peter Jeavons. An Algebraic Characterisation of Complexity for Valued Constraint. In Frédéric Benhamou, editor, *Principles and Practice of Constraint Programming - CP 2006, 12th International Conference, CP 2006, Nantes, France, September 25-29, 2006, Proceedings*, volume 4204 of *Lecture Notes in Computer Science*, pages 107–121. Springer, 2006. (15)

[CCJK06] David A. Cohen, Martin C. Cooper, Peter Jeavons, and Andrei A. Krokhin. The complexity of soft constraint satisfaction. *Artif. Intell.*, 170(11):983–1016, 2006. (15)

[CCJŽ11] David A. Cohen, Páidí Creed, Peter G. Jeavons, and Stanislav Živný. An Algebraic Theory of Complexity for Valued Constraints: Establishing a Galois Connection. In Filip Murlak and Piotr Sankowski, editors, *Mathematical Foundations of Computer Science 2011 - 36th International Symposium, MFCS 2011, Warsaw, Poland, August 22-26, 2011. Proceedings*, volume 6907 of *Lecture Notes in Computer Science*, pages 231–242. Springer, 2011. (15)

[CDG13] Hubie Chen, Víctor Dalmau, and Berit Grußien. Arc consistency and friends. *J. Log. Comput.*, 23(1):87–108, 2013. (49)

[CFI92] Jin-yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identifications. *Comb.*, 12(4):389–410, 1992. (36, 117, 118)

[Che09] Hubie Chen. A rendezvous of logic, complexity, and algebra. *ACM Computing Surveys (CSUR)*, 42(1):2, 2009. (10)

[Che12] Hubie Chen. Meditations on Quantified Constraint Satisfaction. In Robert L. Constable and Alexandra Silva, editors, *Logic and Program Semantics - Essays Dedicated to Dexter Kozen on the Occasion of His 60th Birthday*, volume 7230 of *Lecture Notes in Computer Science*, pages 35–49. Springer, 2012. (17, 151)

[CKS01] Nadia Creignou, Sanjeev Khanna, and Madhu Sudan. *Complexity classifications of Boolean constraint satisfaction problems*, volume 7 of *SIAM monographs on discrete mathematics and applications*. SIAM, 2001. (16)

[CL17] Hubie Chen and Benoît Larose. Asking the Metaquestions in Constraint Tractability. *ACM Trans. Comput. Theory*, 9(3):11:1–11:27, 2017. (12)

[CLRS13] Siu On Chan, James R. Lee, Prasad Raghavendra, and David Steurer. Approximate Constraint Satisfaction Requires Large LP Relaxations. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 350–359. IEEE Computer Society, 2013. (42)

[CM21] Catarina Carvalho and Barnaby Martin. The lattice and semigroup structure of multipermutations. *International Journal of Algebra and Computation*, 0(0):1–25, 2021. (151, 178)

[CMZ17] Catarina Carvalho, Barnaby Martin, and Dmitriy Zhuk. The Complexity of Quantified Constraints Using the Algebraic Formulation. In Kim G. Larsen, Hans L. Bodlaender, and Jean-François Raskin, editors, *42nd International Symposium on Mathematical Foundations of Computer Science, MFCS 2017, August 21-25, 2017 - Aalborg, Denmark*, volume 83 of *LIPIcs*, pages 27:1–27:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. (17)

[Coo00] Stephen Cook. The P versus NP problem. 2000. (1)

[CRŽ22] Clément Carbonnel, Miguel Romero, and Stanislav Živný. The Complexity of General-Valued Constraint Satisfaction Problems Seen from the Other Side. *SIAM Journal on Computing*, 51(1):19–69, 2022. (15, 96, 104)

[CŽ11] Páidí Creed and Stanislav Živný. On Minimal Weighted Clones. In Jimmy Ho-Man Lee, editor, *Principles and Prac-*

tice of Constraint Programming - CP 2011 - 17th International Conference, CP 2011, Perugia, Italy, September 12-16, 2011. Proceedings, volume 6876 of *Lecture Notes in Computer Science*, pages 210–224. Springer, 2011. (15)

[CZ22a] Lorenzo Ciardo and Stanislav Zivný. CLAP: A New Algorithm for Promise CSPs. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 1057–1068. SIAM, 2022. (14)

[CŽ22b] Lorenzo Ciardo and Stanislav Živný. The Sherali-Adams Hierarchy for Promise CSPs through Tensors. *CoRR*, abs/2203.02478, 2022. (14, 42)

[Dal97] Victor Dalmau. Some dichotomy theorems on constant-free quantified Boolean formulas. 1997. (16)

[DBL13] Ken R Duffy, Charles Bordenave, and Douglas J Leith. Decentralized constraint satisfaction. *IEEE/ACM Transactions on Networking (TON)*, 21(4):1298–1308, 2013. (53)

[DGR18] Holger Dell, Martin Grohe, and Gaurav Rattan. Lovász Meets Weisfeiler and Leman. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 40:1–40:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. (36, 118)

[DJKK08] Vladimir G. Deineko, Peter Jonsson, Mikael Klasson, and Andrei A. Krokhin. The approximability of MAX CSP with fixed-value constraints. *J. ACM*, 55(4):16:1–16:37, 2008. (8, 15)

[DK13] Víctor Dalmau and Andrei A. Krokhin. Robust Satisfiability for CSPs: Hardness and Algorithmic Results. *ACM Trans. Comput. Theory*, 5(4):15:1–15:25, 2013. (86)

[DKM18] Víctor Dalmau, Andrei A. Krokhin, and Rajsekar Manokaran. Towards a characterization of constant-factor approximable finite-valued CSPs. *J. Comput. Syst. Sci.*, 97:14–27, 2018. (86)

[DP99] Víctor Dalmau and Justin Pearson. Closure Functions and Width 1 Problems. In Joxan Jaffar, editor, *Principles and Practice of Constraint Programming - CP'99, 5th International Conference, Alexandria, Virginia, USA, October 11-14, 1999, Proceedings*, volume 1713 of *Lecture Notes in Computer Science*, pages 159–173. Springer, 1999. (49)

[Dvo10] Zdenek Dvorák. On recognizing graphs by numbers of homomorphisms. *J. Graph Theory*, 64(4):330–342, 2010. (36, 118)

[FKOS19] Miron Ficak, Marcin Kozik, Miroslav Olsák, and Szymon Stankiewicz. Dichotomy for Symmetric Boolean PCSPs. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 57:1–57:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. (13)

[Fok13] Wan Fokkink. *Distributed algorithms: an intuitive approach.* MIT Press, 2013. (56)

[FPY18] Ferdinando Fioretto, Enrico Pontelli, and William Yeoh. Distributed Constraint Optimization Problems and Applications: A Survey. *J. Artif. Int. Res.*, 61(1):623–698, January 2018. (21, 54, 55)

[FV98] Tomás Feder and Moshe Y Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory. *SIAM Journal on Computing*, 28(1):57–104, 1998. (8, 9, 49, 51)

[Gei68] David Geiger. Closed systems of functions and predicates. *Pacific Journal of Mathematics*, 27(1):95 – 100, 1968. (11)

[Ger17] Jonathan Gerhard. Visualizing the difference between the 4×4 Rook's graph and the Shrikhande graph. `https://mathematicaladd.wordpress.com/2017/02/06/`, 2017. (117)

[GJ76] M. R. Garey and D. S. Johnson. The Complexity of Near-Optimal Graph Coloring. *J. ACM*, 23(1):43–49, jan 1976. (13)

[GKMS17] Martin Grohe, Kristian Kersting, Martin Mladenov, and Pascal Schweitzer. Color refinement and its applications. *Van den Broeck, G.; Kersting, K.; Natarajan, S*, 2017. (34)

[GL18] Venkatesan Guruswami and Euiwoong Lee. Strong Inapproximability Results on Balanced Rainbow-Colorable Hypergraphs. *Comb.*, 38(3):547–599, 2018. (153, 164)

[GMT09] Konstantinos Georgiou, Avner Magen, and Madhur Tulsiani. Optimal Sherali-Adams Gaps from Pairwise Independence. In Irit Dinur, Klaus Jansen, Joseph Naor, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 12th International Workshop, APPROX 2009, and 13th International Workshop, RANDOM 2009, Berkeley, CA, USA, August 21-23, 2009. Proceedings*, volume 5687 of *Lecture Notes in Computer Science*, pages 125–139. Springer, 2009. (42)

[GN21] Martin Grohe and Daniel Neuen. Recent advances on the graph isomorphism problem. In Konrad K. Dabrowski, Maximilien Gadouleau, Nicholas Georgiou, Matthew Johnson, George B. Mertzios, and Daniël Paulusma, editors, *Surveys in Combinatorics, 2021: Invited lectures from the 28th British Combinatorial Conference, Durham, UK, July 5-9, 2021*, pages 187–234. Cambridge University Press, 2021. (117)

[GO15] Martin Grohe and Martin Otto. Pebble Games and linear equations. *J. Symb. Log.*, 80(3):797–844, 2015. (119, 140)

[Gro07] Martin Grohe. The Complexity of Homomorphism and Constraint Satisfaction Problems Seen from the Other Side. *J. ACM*, 54(1), March 2007. (9, 145)

[Gro12] Martin Grohe. Fixed-point definability and polynomial time on graphs with excluded minors. *J. ACM*, 59(5):27:1–27:64, 2012. (117)

[Gro20] Martin Grohe. Word2vec, Node2vec, Graph2vec, X2vec: Towards a Theory of Vector Embeddings of Structured Data. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS'20, page 1–16, New York, NY, USA, 2020. Association for Computing Machinery. (53)

[GS20] Martin Grohe and Pascal Schweitzer. The graph isomorphism problem. *Commun. ACM*, 63(11):128–134, 2020. (117)

[GT18] Mrinalkanti Ghosh and Madhur Tulsiani. From Weak to Strong Linear Programming Gaps for All Constraint Satisfaction Problems. *Theory Comput.*, 14(1):1–33, 2018. (86)

[GWN20] Martin Grohe, Daniel Wiebking, and Daniel Neuen. Isomorphism Testing for Graphs Excluding Small Minors. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 625–636. IEEE, 2020. (117)

[Hel96] Lauri Hella. Logical hierarchies in PTIME. *Inf. Comput.*, 129(1):1–19, 1996. (140)

[HN90] Pavol Hell and Jaroslav Nešetřil. On the complexity of H-coloring. *J. Comb. Theory, Ser. B*, 48(1):92–110, 1990. (9)

[IL90] Neil Immerman and Eric Lander. *Describing Graphs: A First-Order Approach to Graph Canonization*, pages 59–81. Springer New York, New York, NY, 1990. (115, 116)

[Imp95] Russell Impagliazzo. A Personal View of Average-Case Complexity. In *Proceedings of the Tenth Annual Structure in Complexity Theory Conference, Minneapolis, Minnesota, USA, June 19-22, 1995*, pages 134–147. IEEE Computer Society, 1995. (1)

[JCG97] Peter Jeavons, David A. Cohen, and Marc Gyssens. Closure properties of constraints. *J. ACM*, 44(4):527–548, 1997. (9)

[Jea98] Peter Jeavons. On the Algebraic Structure of Combinatorial Problems. *Theor. Comput. Sci.*, 200(1-2):185–204, 1998. (8, 9, 11)

[Jer17] Mark Jerrum. Counting Constraint Satisfaction Problems. In Andrei A. Krokhin and Stanislav Živný, editors, *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*, pages 205–231. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. (8, 12)

[JLNZ17] Peter Jonsson, Victor Lagerkvist, Gustav Nordh, and Bruno Zanuttini. Strong partial clones and the time complexity of SAT problems. *J. Comput. Syst. Sci.*, 84:52–78, 2017. (64)

[Kaz22] Alexander Kazda. Minion homomorphisms give reductions between promise valued CSPs. In preparation, 2022. (95, 97)

[Kie20] Sandra Kiefer. The Weisfeiler-Leman algorithm: an exploration of its power. *ACM SIGLOG News*, 7(3):5–27, 2020. (117)

[KKR17] Vladimir Kolmogorov, Andrei A. Krokhin, and Michal Rolínek. The Complexity of General-Valued CSPs. *SIAM J. Comput.*, 46(3):1087–1110, 2017. (15)

[KLS00] Sanjeev Khanna, Nathan Linial, and Shmuel Safra. On the Hardness of Approximating the Chromatic Number. *Comb.*, 20(3):393–415, 2000. (13)

[KMTV09] Amit Kumar, Rajsekar Manokaran, Madhur Tulsiani, and Nisheeth K. Vishnoi. On the Optimality of a Class of LP-based Algorithms. *Electron. Colloquium Comput. Complex.*, page 124, 2009. (40)

[KMTV11] Amit Kumar, Rajsekar Manokaran, Madhur Tulsiani, and Nisheeth K. Vishnoi. On LP-Based Approximability for Strict CSPs. In Dana Randall, editor, *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 1560–1573. SIAM, 2011. (86)

[KN22] Sandra Kiefer and Daniel Neuen. The Power of the Weisfeiler-Leman Algorithm to Decompose Graphs. *SIAM J. Discret. Math.*, 36(1):252–298, 2022. (117)

[KO15] Marcin Kozik and Joanna Ochremiak. Algebraic Properties of Valued Constraint Satisfaction Problem. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, volume 9134 of *Lecture Notes in Computer Science*, pages 846–858. Springer, 2015. (15, 100)

[Kol13] Vladimir Kolmogorov. The Power of Linear Programming for Finite-Valued CSPs: A Constructive Characterization. In Fedor V. Fomin, Rusins Freivalds, Marta Z. Kwiatkowska, and David Peleg, editors, *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part I*, volume 7965 of *Lecture Notes in Computer Science*, pages 625–636. Springer, 2013. (101)

[KOT+12] Gabor Kun, Ryan O'Donnell, Suguru Tamaki, Yuichi Yoshida, and Yuan Zhou. Linear Programming, Width-1 CSPs, and Robust Satisfaction. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS

'12, page 484–495, New York, NY, USA, 2012. Association for Computing Machinery. (40, 44, 45, 50, 86)

[Koz16] Marcin Kozik. Weak consistency notions for all the CSPs of bounded width. In *2016 31st Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–9, 2016. (51)

[Koz21] Marcin Kozik. Solving CSPs Using Weak Local Consistency. *SIAM Journal on Computing*, 50(4):1263–1286, 2021. (51, 55, 77, 78, 79)

[KPS19] Sandra Kiefer, Ilia Ponomarenko, and Pascal Schweitzer. The Weisfeiler-Leman Dimension of Planar Graphs Is at Most 3. *J. ACM*, 66(6):44:1–44:31, 2019. (117)

[KR08] Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within $2-\epsilon$. *J. Comput. Syst. Sci.*, 74(3):335–349, 2008. (40)

[KS16] Gábor Kun and Mario Szegedy. A new line of attack on the dichotomy conjecture. *European Journal of Combinatorics*, 52:338 – 367, 2016. Special Issue: Recent Advances in Graphs and Analysis. (50)

[KSS22] Sandra Kiefer, Pascal Schweitzer, and Erkal Selman. Graphs Identified by Logics with Counting. *ACM Trans. Comput. Log.*, 23(1):1:1–1:31, 2022. (116)

[KTŽ15] Vladimir Kolmogorov, Johan Thapper, and Stanislav Živný. The Power of Linear Programming for General-Valued CSPs. *SIAM J. Comput.*, 44(1):1–36, 2015. (15, 96, 97)

[Kun13] Gábor Kun. Constraints, MMSNP and expander relational structures. *Comb.*, 33(3):335–347, 2013. (90)

[KV00] Phokion G. Kolaitis and Moshe Y. Vardi. Conjunctive-Query Containment and Constraint Satisfaction. *J. Comput. Syst. Sci.*, 61(2):302–332, 2000. (8)

[KŽ17a] Andrei Krokhin and Stanislav Živný. The Complexity of Valued CSPs. In Andrei Krokhin and Stanislav Živný, editors, *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*, pages 233–266. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2017. (14, 15, 101)

[KŽ17b] Andrei Krokhin and Stanislav Živný. *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7. Schloss Dagstuhl, 2017. (7)

[Lad75] Richard E. Ladner. On the Structure of Polynomial Time Reducibility. *J. ACM*, 22(1):155–171, 1975. (10)

[Lei82] Frank Thomson Leighton. Finite common coverings of graphs. *Journal of Combinatorial Theory, Series B*, 33(3):231 – 238, 1982. (36)

[LLT07] Benoît Larose, Cynthia Loten, and Claude Tardif. A Characterisation of First-Order Constraint Satisfaction Problems. *Log. Methods Comput. Sci.*, 3(4), 2007. (21)

[Lov67] László Lovász. Operations with structures. *Acta Mathematica Academiae Scientiarum Hungarica*, 18(3-4):321–328, 1967. (117)

[LS91] László Lovász and Alexander Schrijver. Cones of Matrices and Set-Functions and 0-1 Optimization. *SIAM J. Optim.*, 1(2):166–190, 1991. (41)

[LW68] AA Leman and B Weisfeiler. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsiya*, 2(9):12–16, 1968. (34)

[Lyn77] Nancy Lynch. Log Space Recognition and Translation of Parenthesis Languages. *J. ACM*, 24(4):583–590, oct 1977. (175)

[Mal14] Peter Malkin. Sherali–Adams relaxations of graph isomorphism polytopes. *Discrete Optimization*, 12:73–97, 2014. (118, 119)

[Mar08] Barnaby Martin. First-Order Model Checking Problems Parameterized by the Model. In Arnold Beckmann, Costas Dimitracopoulos, and Benedikt Löwe, editors, *Logic and Theory of Algorithms, 4th Conference on Computability in Europe, CiE 2008, Athens, Greece, June 15-20, 2008, Proceedings*, volume 5028 of *Lecture Notes in Computer Science*, pages 417–427. Springer, 2008. (151, 153, 171)

[Mar10] Barnaby Martin. The Lattice Structure of Sets of Surjective Hyper-Operations. In David Cohen, editor, *Principles and Practice of Constraint Programming - CP 2010 - 16th International Conference, CP 2010, St. Andrews, Scotland, UK, September 6-10, 2010. Proceedings*, volume 6308 of *Lecture Notes in Computer Science*, pages 368–382. Springer, 2010. (151, 178)

[Mar17] Barnaby Martin. Quantified Constraints in Twenty Seventeen. In Andrei Krokhin and Stanislav Živný, editors, *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*, pages 327–346. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2017. (16, 151)

[Mei08] Amnon Meisels. *Distributed Search by Constrained Agents: algorithms, performance, communication.* Springer Science & Business Media, 2008. (54)

[MLM+21] Christopher Morris, Yaron Lipman, Haggai Maron, Bastian Rieck, Nils M. Kriege, Martin Grohe, Matthias Fey, and Karsten M. Borgwardt. Weisfeiler and Leman go Machine Learning: The Story so far. *CoRR*, abs/2112.09992, 2021. (54)

[MM10] Barnaby Martin and Jos Martin. The Complexity of Positive First-Order Logic without Equality II: The Four-Element

Case. In Anuj Dawar and Helmut Veith, editors, *Computer Science Logic, 24th International Workshop, CSL 2010, 19th Annual Conference of the EACSL, Brno, Czech Republic, August 23-27, 2010. Proceedings*, volume 6247 of *Lecture Notes in Computer Science*, pages 426–438. Springer, 2010. (151)

[MM11] Florent R. Madelaine and Barnaby Martin. A Tetrachotomy for Positive First-Order Logic without Equality. In *Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science, LICS 2011, June 21-24, 2011, Toronto, Ontario, Canada*, pages 311–320. IEEE Computer Society, 2011. (151, 155, 168, 169, 171)

[MM12] Florent Madelaine and Barnaby Martin. The Complexity of Positive First-Order Logic without Equality. *ACM Trans. Comput. Logic*, 13(1), January 2012. (151, 155)

[MM17] Konstantin Makarychev and Yury Makarychev. Approximation Algorithms for CSPs. In Andrei A. Krokhin and Stanislav Živný, editors, *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*, pages 287–325. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. (15)

[MM18] Florent R. Madelaine and Barnaby Martin. On the Complexity of the Model Checking Problem. *SIAM J. Comput.*, 47(3):769–797, 2018. (8, 16, 150, 151, 153, 171)

[MRF+19] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and Leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4602–4609, 2019. (53, 54)

[NR89] Jaroslav Nešetřil and Vojtěch Rödl. Chromatically optimal rigid graphs. *Journal of Combinatorial Theory, Series B*, 46(2):133–141, 1989. (44)

[RBW06] Francesca Rossi, Peter van Beek, and Toby Walsh. *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier Science Inc., USA, 2006. (7)

[RS84] Neil Robertson and Paul D. Seymour. Graph minors. III. Planar tree-width. *J. Comb. Theory, Ser. B*, 36(1):49–64, 1984. (118)

[RSU94] Motakuri V. Ramana, Edward R. Scheinerman, and Daniel Ullman. Fractional isomorphism of graphs. *Discrete Mathematics*, 132(1):247 – 265, 1994. (34, 36)

[SA90] Hanif D. Sherali and Warren P. Adams. A Hierarchy of Relaxations Between the Continuous and Convex Hull Representations for Zero-One Programming Problems. *SIAM J. Discret. Math.*, 3(3):411–430, 1990. (41)

[Sch78] Thomas J. Schaefer. The Complexity of Satisfiability Problems. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, STOC '78, pages 216–226, New York, NY, USA, 1978. ACM. (9)

[Sch86] Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc., USA, 1986. (102)

[SU11] Edward R Scheinerman and Daniel H Ullman. *Fractional graph theory: a rational approach to the theory of graphs*. Courier Corporation, 2011. (34, 124, 125)

[Tin86] Gottfried Tinhofer. Graph isomorphism and theorems of Birkhoff type. *Computing*, 36(4):285–300, 1986. (36)

[Tin91] Gottfried Tinhofer. A note on compact graphs. *Discret. Appl. Math.*, 30(2-3):253–264, 1991. (36)

[TRWG20] Jan Tönshoff, Martin Ritzert, Hinrikus Wolf, and Martin Grohe. Graph Neural Networks for Maximum Constraint Satisfaction. *Frontiers Artif. Intell.*, 3:580607, 2020. (54)

[TŽ12] Johan Thapper and Stanislav Živný. The Power of Linear Programming for Valued CSPs. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 669–678. IEEE Computer Society, 2012. (15, 45, 101, 106)

[TŽ16] Johan Thapper and Stanislav Živný. The Complexity of Finite-Valued CSPs. *J. ACM*, 63(4):37:1–37:33, 2016. (15, 146)

[TŽ17] Johan Thapper and Stanislav Živný. The Power of Sherali-Adams Relaxations for General-Valued CSPs. *SIAM J. Comput.*, 46(4):1241–1279, 2017. (15, 42, 51, 97)

[VŽ21] Caterina Viola and Stanislav Živný. The Combined Basic LP and Affine IP Relaxation for Promise VCSPs on Infinite Domains. *ACM Trans. Algorithms*, 17(3):21:1–21:23, 2021. (45, 95, 97, 100, 101, 106)

[WŽ20] Marcin Wrochna and Stanislav Živný. Improved hardness for H-colourings of G-colourable graphs. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1426–1435. SIAM, 2020. (14)

[XHLJ19] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. (54)

[YDIK98] Makoto Yokoo, Edmund H Durfee, Toru Ishida, and Kazuhiro Kuwabara. The distributed constraint satisfaction problem: Formalization and algorithms. *IEEE Transactions on knowledge and data engineering*, 10(5):673–685, 1998. (55)

[YH00] Makoto Yokoo and Katsutoshi Hirayama. Algorithms for distributed constraint satisfaction: A review. *Autonomous Agents and Multi-Agent Systems*, 3(2):185–207, 2000. (54, 55)

[YIDK92] Makoto Yokoo, Toru Ishida, Edmund H Durfee, and Kazuhiro Kuwabara. Distributed constraint satisfaction for formalizing distributed problem solving. In *[1992] Proceedings of the 12th International Conference on Distributed Computing Systems*, pages 614–621. IEEE, 1992. (53, 55)

[YK88] Masafumi Yamashita and Tiko Kameda. Computing on an Anonymous Network. In Danny Dolev, editor, *Proceedings of the Seventh Annual ACM Symposium on Principles of Distributed Computing, Toronto, Ontario, Canada, August 15-17, 1988*, pages 117–130. ACM, 1988. (57)

[YZ14] Yuichi Yoshida and Yuan Zhou. Approximation schemes via Sherali-Adams hierarchy for dense constraint satisfaction problems and assignment problems. In Moni Naor, editor, *Innovations in Theoretical Computer Science, ITCS'14, Princeton, NJ, USA, January 12-14, 2014*, pages 423–438. ACM, 2014. (42)

[Zhu17] Dmitriy Zhuk. A Proof of CSP Dichotomy Conjecture. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 331–342, Oct 2017. (10)

[Zhu20] Dmitriy Zhuk. A Proof of the CSP Dichotomy Conjecture. *J. ACM*, 67(5):30:1–30:78, August 2020. (10, 12, 150)

[Živ09] Stanislav Živný. *The complexity and expressive power of valued constraints*. PhD thesis, University of Oxford, UK, 2009. (15)

[ZM20] Dmitriy Zhuk and Barnaby Martin. QCSP monsters and the demise of the Chen conjecture. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 91–104. ACM, 2020. (17, 151)

Abbreviations

BLP	Basic Linear Programming
CSP	Constraint Satisfaction Problem
DCSP	Distributed Constraint Satisfaction Problem
LP	Linear Programming
MC	Model Checking
MuHom	Multi-Homomorphism
MWU	Multiplicative Weight Update
PCSP	Promise Constraint Satisfaction Problem
PMC	Promise Model Checking
PVCSP	Promise Valued Constraint Satisfaction Problem
QCSP	Quantified Constraint Satisfaction Problem
SA	Sherali-Adams
SMuHom	Surjective Multi-Homomorphism
VCSP	Valued Constraint Satisfaction Problem
WL	Weisfeiler-Leman

Funding

The project that gave rise to these results received the support of a fellowship from “la Caixa” Foundation (ID 100010434). The fellowship code is LCF/BQ/DI18/11660056.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 713673

The final year of the PhD was supported by a MICINN grant PID2019-109137GB-C22.