The Constraint Satisfaction Problem: Complexity and Approximability

Dagstuhl Seminar 15301

Edited by

Andrei Krokhin

Stanislav Živný

DAGSTUHL
FOLLOW-UPS

Editors

Andrei Krokhin
School of Engineering and Computing Sciences
University of Durham, UK
`andrei.krokhin@durham.ac.uk`

Stanislav Živný
Department of Computer Science
University of Oxford, UK
`standa.zivny@cs.ox.ac.uk`

ACM Classification 1998
F.2.2 Nonnumerical Algorithms and Problems

ISBN 978-3-95977-003-3

Published online and open access by
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at http://www.dagstuhl.de/dagpub/978-3-95977-003-3.

Publication date
February, 2017

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at http://dnb.d-nb.de.

Digital Object Identifier: 10.4230/DFU.Vol7.15301.i

DFU – Dagstuhl Follow-Ups

The series *Dagstuhl Follow-Ups* is a publication format which offers a frame for the publication of peer-reviewed papers based on Dagstuhl Seminars. DFU volumes are published according to the principle of Open Access, i.e., they are available online and free of charge.

ISSN 1868-8977

http://www.dagstuhl.de/dfu

Contents

Preface
Andrei Krokhin and Stanislav Živný vii

Chapter 01
Polymorphisms, and How to Use Them
Libor Barto and Andrei Krokhin and Ross Willard 1

Chapter 02
Absorption in Universal Algebra and CSP
Libor Barto and Marcin Kozik 45

Chapter 03
Constraint Satisfaction Problems over Numeric Domains
Manuel Bodirsky and Marcello Mamino 79

Chapter 04
Hybrid Tractable Classes of Constraint Problems
Martin C. Cooper and Stanislav Živný 113

Chapter 05
Backdoor Sets for CSP
Serge Gaspers, Sebastian Ordyniak, and Stefan Szeider 137

Chapter 06
On the Complexity of Holant Problems
Heng Guo and Pinyan Lu 159

Chapter 07
Parameterized Constraint Satisfaction Problems: a Survey
Gregory Gutin and Anders Yeo 179

Chapter 08
Counting Constraint Satisfaction Problems
Mark Jerrum 205

Chapter 09
The Complexity of Valued CSPs
Andrei Krokhin and Stanislav Živný 233

Chapter 10
Algebra and the Complexity of Digraph CSPs: a Survey
Benoît Larose 267

Chapter 11
Approximation Algorithms for CSPs
Konstantin Makarychev and Yury Makarychev 287

Chapter 12
Quantified Constraints in Twenty Seventeen
Barnaby Martin 327

The Constraint Satisfaction Problem: Complexity and Approximability. *Dagstuhl Follow-Ups*, Volume 7, ISBN 978-3-95977-003-3.
Editors: Andrei Krokhin and Stanislav Živný
DAGSTUHL FOLLOW-UPS Dagstuhl Publishing
Schloss Dagstuhl – Leibniz Center for Informatics, Germany

Preface

This volume is based on the Dagstuhl Seminar 15301 "The Constraint Satisfaction Problem: Complexity and Approximability" held in July 2015 and organised by Andrei A. Bulatov (Simon Fraser University), Venkatesan Guruswami (Carnegie Mellon University), Andrei Krokhin (Durham University), and Dániel Marx (Hungarian Academy of Sciences).

Overview of the Seminar

The constraint satisfaction problem, or CSP in short, provides a unifying framework in which it is possible to express, in a natural way, a wide variety of computational problems dealing with mappings and assignments, including satisfiability, graph colourability, and systems of equations. The CSP framework originated 30-35 years ago independently in artificial intelligence, database theory, and graph theory, under three different guises, and it was realised only in the late 1990s that these are in fact different faces of the same fundamental problem. Nowadays, the CSP is extensively used in theoretical computer science, being a mathematical object with very rich structure that provides an excellent laboratory both for classification methods and for algorithmic techniques, while in AI and more applied areas of computer science this framework is widely regarded as a versatile and efficient way of modelling and solving a variety of real-world problems, such as planning and scheduling, software verification and natural language comprehension, to name just a few. An instance of CSP consists of a set of variables, a set of values for the variables, and a set of constraints that restrict the combinations of values that certain subsets of variables may take. Given such an instance, the possible questions include (a) deciding whether there is an assignment of values to the variables so that every constraint is satisfied, or optimising such assignments in various ways, (b) counting satisfying assignments, exactly or approximately, or (c) finding an assignment satisfying as many constraints as possible. There are many important modifications and extensions of this basic framework, e.g. those that deal with valued or global constraints.

Constraint satisfaction has always played a central role in computational complexity theory; appropriate versions of CSPs are classical complete problems for most standard complexity classes. CSPs constitute a very rich and yet sufficiently manageable class of problems to give a good perspective on general computational phenomena. For instance, they help to understand which mathematical properties make a computational problem tractable (in a wide sense, e.g. polynomial-time solvable or non-trivially approximable, fixed-parameter tractable or definable in a weak logic). It is only natural that CSPs play a role in many high-profile conjectures in complexity theory, exemplified by the Dichotomy Conjecture of Feder and Vardi and the Unique Games Conjecture of Khot.

The recent flurry of activity on the topic of the seminar is witnessed by three previous Dagstuhl seminars, titled "Complexity of constraints" (06401) and "The CSP: complexity and approximability" (09441, 12541), that were held in 2006, 2009, and 2012 respectively. This seminar was a follow-up to the 2009 and 2012 seminars. Indeed, the exchange of ideas at the 2009 and 2012 seminars has led to new ambitious research projects and to establishing regular communications channels, and there is a clear potential of a further systematic interaction that will keep on cross-fertilizing the areas and opening new research directions. The 2015 seminar brought together forty three researchers from different highly advanced areas of constraint satisfaction and involved many specialists who use universal-algebraic,

The Constraint Satisfaction Problem: Complexity and Approximability. *Dagstuhl Follow-Ups*, Volume 7, ISBN 978-3-95977-003-3.
Editors: Andrei Krokhin and Stanislav Živný
DAGSTUHL FOLLOW-UPS Dagstuhl Publishing
Schloss Dagstuhl – Leibniz Center for Informatics, Germany

combinatorial, geometric and probabilistic techniques to study CSP-related algorithmic problems. The participants presented, in 28 talks, their recent results on a number of important questions concerning the topic of the seminar. One particular feature of this seminar was a significant increase in the number of talks involving multiple subareas and approaches within its research direction – a definite sign of the growing synergy, which is one of the main goals of this series of seminars.

The seminar was well received as witnessed by the high rate of accepted invitations and the great degree of involvement by the participants. Because of the multitude of impressive results reported during the seminar and the active discussions between researchers with different expertise areas, the organisers regard this seminar as a great success. With steadily increasing interactions between such researchers, we foresee a new seminar focussing on the interplay between different approaches to studying the complexity and approximability of the CSP.

Follow-Up

For some of the topics presented at the Dagstuhl Seminar 15301 there are excellent surveys. Some other topics are still too nascent to justify survey articles at this point. For several topics for which no surveys presently exist or the current ones are already outdated due to the recent progress, we felt that the time is ripe to produce such surveys as a follow-up to the Dagstuhl Seminar 15301.

Overview of the Volume

The first chapter in this volume is introductory and provides detailed explanations of the so-called algebraic approach to decision CSPs over finite domains. The algebraic approach has been behind several breakthroughs in the last decade. The remaining chapters are more advanced and specialised.

The second chapter gives an overview of absorption, a powerful algebraic technique used in the proof of the "bounded width theorem" and more generally in the study of decision CSPs. The third chapter is about CSPs with infinite numerical domains and constraints defined by using arithmetical operations. The fourth chapter explores so-called hybrid CSPs, which are classes of CSPs that are neither language- nor structure-based. The fifth chapter provides an overview of research on CSPs and backdoors, which is a concept that allows for a formalisation of being "close to a tractable class". The sixth chapter deals with Holant problems, which are special types of CPSs in which every variable appears in exactly two constraints. The seventh chapter studies CSPs from the parametrised perspective. The eight chapter gives an overview of the counting variants of CSPs. The ninth chapter is concerned with valued CSPs, which are generalisations of CSPs to optimisation problems. The tenth chapter investigates CSPs specialised to digraphs. The eleventh chapter gives a good account of the available approximation algorithms for CSPs. Finally, the twelfth chapter provides an overview of quantified CSPs.

Acknowledgements

On behalf of the Dagstuhl Seminar 15301 organisers, we wish to express their gratitude to the Scientific Directors of the Dagstuhl Centre for their support of the seminar. We are grateful to Dr. Marc Herbstritt, Member of the Scientific Staff of the Schloss Dagstuhl –

Leibniz Center for Informatics, for his help with preparing this volume. Finally, we wish to thank the participants of the seminar, the authors of the chapters in this volume, and the anonymous reviewers of the chapters. This volume would not have been possible without the contributions of all these colleagues.

January 2017 *Andrei Krokhin and Stanislav Živný*

List of Authors

Libor Barto
Charles University in Prague, Czech republic
libor.barto@gmail.com

Manuel Bodirsky
TU Dresden, Germany
manuel.bodirsky@tu-dresden.de

Martin C. Cooper
University of Toulouse III, France
cooper@irit.fr

Serge Gaspers
UNSW Australia and Data61, CSIRO
sergeg@cse.unsw.edu.au

Heng Guo
Queen Mary, University of London, UK
h.guo@qmul.ac.uk

Gregory Gutin
Royal Holloway, University of London, UK
gutin@cs.rhul.ac.uk

Mark Jerrum
Queen Mary, University of London, UK
m.jerrum@qmul.ac.uk

Marcin Kozik
Jagiellonian University, Poland
marcin.kozik@uj.edu.pl

Andrei Krokhin
University of Durham, UK
andrei.krokhin@durham.ac.uk

Benoît Larose
Université du Québec à Montréal, Canada
blarose@lacim.ca

Pinyan Lu
Shanghai University of Finance and Economics, China
lu.pinyan@mail.shufe.edu.cn

Konstantin Makarychev
Microsoft Research, USA
komakary@microsoft.com

Yury Makarychev
Toyota Technological Institute at Chicago, USA
yury@ttic.edu

Marcello Mamino
TU Dresden, Germany
marcello.mamino@tu-dresden.de

Barnaby Martin
University of Durham, UK
barnaby.d.martin@durham.ac.uk

Sebastian Ordyniak
TU Wien, Austria
ordyniak@ac.tuwien.ac.at

Stefan Szeider
TU Wien, Austria
szeider@ac.tuwien.ac.at

Ross Willard
University of Waterloo, Canada
ross.willard@uwaterloo.ca

Anders Yeo
Singapore University of Technology and Design, Singapore and
University of Johannesburg, South Africa
anders.yeo.work@gmail.com

Stanislav Živný
University of Oxford, UK
standa.zivny@cs.ox.ac.uk

The Constraint Satisfaction Problem: Complexity and Approximability. *Dagstuhl Follow-Ups*, Volume 7, ISBN 978-3-95977-003-3.
Editors: Andrei Krokhin and Stanislav Živný
DAGSTUHL FOLLOW-UPS Dagstuhl Publishing
Schloss Dagstuhl – Leibniz Center for Informatics, Germany

Polymorphisms, and How to Use Them

Libor Barto[*1], **Andrei Krokhin**[2], **and Ross Willard**[†3]

1 **Department of Algebra, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic**
`libor.barto@gmail.com`
2 **School of Engineering and Computing Sciences, University of Durham, Durham, UK**
`andrei.krokhin@durham.ac.uk`
3 **Department of Pure Mathematics, University of Waterloo, Waterloo, Canada**
`ross.willard@uwaterloo.ca`

Abstract

This article describes the algebraic approach to Constraint Satisfaction Problem that led to many developments in both CSP and universal algebra. No prior knowledge of universal algebra is assumed.

1998 ACM Subject Classification F.2.0 [Analysis of Algorithms and Problem Complexity] General, F.2.0 [Discrete Mathematics] General

Keywords and phrases Constraint satisfaction, Complexity, Universal algebra, Polymorphism

Digital Object Identifier 10.4230/DFU.Vol7.15301.1

1 Introduction

The Constraint Satisfaction Problem (CSP) provides a common framework for expressing a wide range of both theoretical and real-life combinatorial problems [111]. Roughly, these are problems where one is given a collection of constraints on overlapping sets of variables and the goal is to assign values to the variables so as to satisfy the constraints. This computational paradigm is very general, and includes many specific well-known problems from several areas of Computer Science and Mathematics. In the last 20 years, complexity-theoretic aspects of CSPs have attracted a great deal of attention at the top level in Theoretical Computer Science. The main reason for this is that the CSP paradigm strikes a perfect balance between generality and structure: it is general enough to reflect very many important computational phenomena, yet it has enough structure that can be exploited to gain very deep insights into these phenomena. The CSP paradigm is often used to tackle the following fundamental general question: *What kind of mathematical structure in computational problems allows for efficient algorithms?*

The topic of this paper is a very active theoretical subfield which studies the computational complexity and other algorithmic properties of the decision version of CSP over a fixed constraint language on a finite domain. This restricted framework is still broad enough to include many decision problems from the class NP, yet it is narrow enough to potentially allow for complete classifications of all such CSP problems.

[*] L. Barto gratefully acknowledges the support of the Grant Agency of the Czech Republic, grant GAČR 13-01832S.
[†] R. Willard gratefully acknowledges the support of the Natural Sciences and Engineering Research Council of Canada.

One particularly important achievement is the understanding of what makes the problems over a fixed constraint language computationally easy or hard. It is not surprising that hardness comes from a lack of symmetry. However, the usual objects capturing symmetry, automorphisms (or endomorphisms) and their groups (or semigroups), are not sufficient in this context. It turns out that the complexity of the CSP over a fixed constraint language is determined by more general symmetries of it: polymorphisms and their clones.

Our aim is to introduce the basics of this exciting area in a way that is understandable to readers with a basic knowledge of computational complexity (see [106, 3]). Our particular focus is on explaining, with worked-out examples, how polymorphisms are applied to obtain positive and negative algorithmic results and why this approach is natural for many classification problems about CSPs and constraint languages. We do not assume any knowledge of algebra and minimize the algebraic terminology that we (define and) use, so the deep universal algebra, which is at the technical core of many advanced results in this direction, stays in the background. Many papers in the reference list contain deep algebra, though.

The structure of the survey is as follows. In Section 2 we give the basic definitions and examples of CSPs over a fixed constraint language, and discuss the main goals of the research programme that we survey. In Section 3 we describe various standard reductions between CSPs with different constraint languages, which on the one hand allows one to group together constraint languages with the same computational properties of the corresponding decision CSPs, and on the other hand paves the path to the algebraic approach to our classification problems. In Section 4 we explain how polymorphisms are used as classifiers for constraint languages and how this leads to hardness results and complexity classification conjectures. In Section 5 we explain how polymorphisms are used to guarantee correctness of algorithms that do not use polymorphisms in their execution. In Section 6 we discuss an algorithm that does use polymorphisms in an essential way in its execution.

2 CSP over a Fixed Constraint Language

A constraint – such as $R(x_3, x_1, x_4)$ – restricts the allowed values for a tuple of variables – in this case (x_3, x_1, x_4) – to be an element of a particular relation on the domain – in this case $R \subseteq D^3$.[1] By an n-ary *relation* R on a domain D we mean a subset of the n-th cartesian power D^n. It is sometimes convenient to work with the corresponding *predicate* which is a mapping from D^n to $\{\text{true}, \text{false}\}$ specifying which tuples are in R. We will use both formalisms, so e.g. $(a, b, c) \in R$ and $R(a, b, c)$ both mean that the triple $(a, b, c) \in D^3$ is from the relation R.

An instance of the CSP is a list of constraints, e.g.,

$$R(x),\ S(y, y, z),\ T(y, w),$$

where R, S, T are relations of appropriate arity on a common fixed domain D and x, y, z, w are variables. A mapping f assigning values from the domain to the variables is a *solution* if it satisfies all the constraints, that is, in our example,

$$R(f(x)) \text{ and } S(f(y), f(y), f(z)) \text{ and } T(f(y), f(w)) \ .$$

A standard formal definition of an instance of the CSP over a finite domain goes as follows.

[1] There are also other types of constraints considered in the literature, e.g. valued and global constraints [111].

▶ **Definition 1.** An *instance of the CSP* is a triple $P = (V, D, \mathcal{C})$ with

- V a finite set of *variables*,
- D a finite *domain*,
- $\mathcal{C}$ a finite list of constraints, where each constraint is a pair $C = (\mathbf{x}, R)$ with
  - $\mathbf{x}$ a tuple of variables of length n, called the *scope* of C, and
  - R an n-ary relation on D, called the *constraint relation* of C.

An *assignment*, that is, a mapping $f : V \to D$, *satisfies* a constraint $C = (\mathbf{x}, R)$ if $f(\mathbf{x}) \in R$, where f is applied component-wise. An assignment f is a *solution* if it satisfies all constraints.

Three basic computational problems associated with an instance are the following:

- **Decision.** Does the given instance have a solution? (A related problem, the *search problem*, is to find some solution if at least one solution exists.)
- **Optimization.** Even if the instance has no solution, find an optimal assignment, i.e., one that satisfies the maximum possible number of constraints. (*Approximation algorithms* are also extensively studied, where the aim is, for example, to find an assignment that satisfies at least 80% of the number of constraints satisfied by an optimal assignment.)
- **Counting.** How many solutions does the given instance have? (This problem also has an approximation version: *approximate counting.*)

To study the computational complexity of these problems we need to specify a representation of instances. In particular, we will assume that the constraint relation in every constraint is given by a list of all its members. Note, however, that for most of the problems considered in this article any reasonable representation can be taken.

2.1 Constraint Languages

Even the easiest of the problems, decision, is computationally hard: It contains many NP-complete problems including, e.g., 3-SAT (see Example 3). However, certain natural restrictions ensure tractability. The main types of restrictions that have been studied are *structural restrictions*, which limit how constraints interact, and *language restrictions*, which limit the choice of constraint relations.

In this paper, we focus just on decision problems with language restrictions. See [92] for optimization problems and a generalization to valued CSPs, [70, 102] for approximation, [81] for counting, [24, 25, 107] for a generalization to infinite domains, and [105] for work on structural restrictions.

▶ **Definition 2.** A *constraint language* $\mathcal{D}$ is a finite set of relations on a common finite domain, D. We use CSP($\mathcal{D}$) to denote the restriction of the general CSP decision problem to instances in which the domain is D and all constraint relations are from $\mathcal{D}$.

We remark that constraint languages (on a finite domain) are often defined to also include infinite sets of relations. For such languages, one can define the complexity in terms of finite subsets, or else one has to specify the choice of representation of instances. For simplicity, we focus on finite constraint languages.

2.2 Examples

▶ **Example 3.** An instance of the standard NP-complete problem [106, 3], 3-SAT, is a Boolean formula in conjunctive normal form with exactly three literals per clause. For example, the formula,

$$\varphi \quad = \quad (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_4 \vee x_5 \vee \neg x_1) \wedge (\neg x_1 \vee \neg x_4 \vee \neg x_3)$$

is a satisfiable instance of 3-SAT. (Any assignment making x_1 and x_2 false, satisfies φ.) 3-SAT is equivalent to CSP($\mathcal{D}_{3\text{SAT}}$), where $D_{3\text{SAT}} = \{0,1\}$ and

$$\mathcal{D}_{3\text{SAT}} = \{S_{ijk} : i,j,k \in \{0,1\}\}, \text{ where } S_{ijk} = \{0,1\}^3 \setminus \{(i,j,k)\} \ .$$

For example, the above formula φ corresponds to the following instance of CSP($\mathcal{D}_{3\text{SAT}}$)

$$S_{010}(x_1,x_2,x_3),\ S_{101}(x_4,x_5,x_1),\ S_{111}(x_1,x_4,x_3) \ .$$

More generally, for a natural number k, k-SAT denotes a similar problem where each clause is a disjunction of k literals.

Since 3-SAT is NP-complete, it follows that k-SAT is NP-complete for each $k \geq 3$. On the other hand, 2-SAT is solvable in polynomial time, and is in fact complete for the complexity class NL (non-deterministic logarithmic space) under log-space reductions [106, 3] (see also Example 9).

▶ **Example 4.** 1-in-3-SAT is CSP($\mathcal{D}_{1\text{in}3\text{SAT}}$) where $\mathcal{D}_{1\text{in}3\text{SAT}}$ contains the single relation $\{(0,0,1),(0,1,0),(1,0,0)\}$. This problem is well known to be NP-complete [112].

▶ **Example 5.** HORN-3-SAT is a restricted version of 3-SAT, where each clause may have at most one positive literal. This problem is equivalent to CSP($\mathcal{D}_{\text{HornSAT}}$) for $\mathcal{D}_{\text{HornSAT}} = \{S_{110}, S_{111}, C_0, C_1\}$ where $C_0 = \{0\}$ and $C_1 = \{1\}$. HORN-3-SAT is solvable in polynomial time, in fact, it is a P-complete problem under log-space reductions [3, 106].

▶ **Example 6.** For a fixed natural number k, the k-COLORING problem is to decide whether it is possible to assign colors $\{0,1,\dots,k-1\}$ to the vertices of an input graph in such a way that adjacent vertices receive different colors. This problem is equivalent to CSP($\mathcal{D}_{\text{kCOLOR}}$), where $D_k = \{0,1,2,\dots,k-1\}$ and $\mathcal{D}_{\text{kCOLOR}} = \{\neq_k\}$ consists of a single relation – the binary inequality relation $\neq_k = \{(a,b) \in D_k^2 : a \neq b\}$.

Indeed, given an instance of CSP($\mathcal{D}_{\text{kCOLOR}}$), we can form a graph whose vertices are the variables and whose edges correspond to the binary constraints (that is, x has an edge to y iff the instance contains the constraint $x \neq_k y$). It is easily seen that the original instance has a solution if and only if the obtained graph is k-colorable. The translation in the other direction is similar.

The k-COLORING problem is NP-complete for $k \geq 3$ [106, 3]. 2-COLORING is equivalent to deciding whether an input graph is bipartite. It is solvable in polynomial time, in fact, it is in the complexity class L (where L stands for logarithmic space) by a celebrated result of Reingold [109], and it is an L-complete problem under first-order reductions.

▶ **Example 7.** Given two digraphs $G = (V(G), E(G))$ and $H = (V(H), E(H))$, a mapping $f : V(G) \to V(H)$ is a *homomorphism* from G to H if f preserves edges, that is, $(u,v) \in E(G)$ implies $(f(u), f(v)) \in E(H)$. The problem whether an input digraph G admits a homomorphism to a fixed digraph H is also known as the H-COLORING problem and has been actively studied in graph theory [72], see also [96]. The k-COLORING problem is a special case of H-COLORING where H is the complete graph on k vertices.

For any digraph H, let $D = V(H)$ and let $\mathcal{D}_H$ be the language that contains just the binary relation $E(H)$. For any digraph H, the problem CSP($\mathcal{D}_H$), corresponds to the H-COLORING problem, where the input digraph G is given by the scopes of the constraints. If we add all nonempty subsets of $V(H)$ as unary relations to $\mathcal{D}_H$, then the resulting CSP is known as LIST H-COLORING [72]. If we add just the singleton subsets of $V(H)$ as unary relations to $\mathcal{D}_H$, then the resulting CSP is known as One-or-All LIST H-COLORING [63, 64].

▶ **Example 8.** Let p be a prime number. An input of 3-LIN(p) is a system of linear equations over the p-element field GF(p), where each equation contains 3 variables, and the question is whether the system has a solution. This problem is equivalent to CSP($\mathcal{D}_{\mathrm{3LINp}}$), where $D_{\mathrm{3LINp}} = \mathrm{GF}(p)$ and $\mathcal{D}_{\mathrm{3LINp}}$ consists of all affine subspaces R_{abcd} of $\mathrm{GF}(p)^3$ of dimension 2, where

$$R_{abcd} = \{(x, y, z) \in \mathrm{GF}(p)^3 : ax + by + cz = d\} \ .$$

This problem is solvable in polynomial time, e.g. by Gaussian elimination.[2] It is complete for a somewhat less familiar complexity class $\mathrm{Mod}_p\mathrm{L}$ [46].

▶ **Example 9.** An instance of the s,t-connectivity problem, STCON, consists of a directed graph and two of its vertices, s and t. The question is whether there exists a directed path from s to t.

A closely related (but not identical) problem is CSP($\mathcal{D}_{\mathrm{STCON}}$), where the domain is $D_{\mathrm{STCON}} = \{0, 1\}$ and $\mathcal{D}_{\mathrm{STCON}} = \{C_0, C_1, I\}$, $C_0 = \{0\}$, $C_1 = \{1\}$, $I = \{(0, 0), (0, 1), (1, 1)\}$. Indeed, given an instance of CSP($\mathcal{D}_{\mathrm{STCON}}$) we form a directed graph much as we did in Example 6 and label some vertices 0 or 1 according to the unary constraints. Then the original instance has a solution if and only if there is no directed path from a vertex labeled 1 to a vertex labeled 0. Thus CSP($\mathcal{D}_{\mathrm{STCON}}$) can be solved by invoking the complement of STCON, the s,t-non-connectivity problem, several times.

Both STCON and CSP($\mathcal{D}_{\mathrm{STCON}}$) can clearly be solved in polynomial time. By the Immerman-Szelepcsényi theorem [75, 115] both problems are NL-complete (under log-space reductions).

In the same way, the s,t-connectivity problem for undirected graphs is closely related to CSP($\mathcal{D}_{\mathrm{USTCON}}$), where $D_{\mathrm{USTCON}} = \{0, 1\}$ and $\mathcal{D}_{\mathrm{USTCON}} = \{C_0, C_1, =\}$. These problems are L-complete by [109].

2.3 The Dichotomy Conjecture

The most fundamental problem in the area was formulated in the landmark paper by Feder and Vardi [65].

▶ **Conjecture 10** (The Dichotomy Conjecture). *For each finite constraint language $\mathcal{D}$, the problem* CSP($\mathcal{D}$) *is in P or is NP-complete.*[3]

Recall that if P $\neq$ NP, then there are problems of intermediate complexity in NP [95]. Feder and Vardi argued that the class of CSPs over fixed constraint languages is a good candidate for the largest natural class of problems which exhibit a P versus NP-complete dichotomy.

At that time the conjecture was supported by two major cases: the dichotomy theorem for all languages over a two-element domain by Schaefer [112] and the dichotomy theorem for languages consisting of a single binary symmetric relation by Hell and Nešetřil [71].

[2] The problem of solving general systems of linear equations over GF(p) without the restriction on number of variables cannot be faithfully phrased as CSP($\mathcal{D}$), even if we allow $\mathcal{D}$ to consist of all affine subspaces, since the input representation of the latter problem can be substantially larger. However, a system of linear equations can be easily rewritten to an instance of 3-LIN(p) by introducing new variables.

[3] It is conjectured in [34] that the dichotomy remains true without the finiteness assumption on $\mathcal{D}$ (the domain D still needs to be finite), if constraint relations in inputs are given by full lists of their members. Namely, the local-global conjecture states that CSP($\mathcal{D}$) is in P (NP-complete) whenever CSP($\mathcal{D}'$) is in P (NP-complete) for every (some) finite $\mathcal{D}' \subseteq \mathcal{D}$.

Feder and Vardi identified two sources of polynomial-time solvability and made several important contributions towards understanding them. In particular, they observed that the known polynomial cases were tied to algebraic closure properties and asked whether polynomial solvability for CSP can always be explained in such a way. This was confirmed by Jeavons, Cohen and Gyssens [80, 78], and these and subsequent papers based on this connection to algebra brought the area to another level, which probably could not be accessed with only combinatorial tools (such as those in [112] or [71]).

2.4 Alternative Views

Note that if we order (or just name) relations in a constraint language $\mathcal{D}$ with domain D, then $\mathcal{D}$ can be viewed as a *relational structure* $(D; R_1, R_2, \dots)$, or equivalently as a *relational database*, with universe D.

Recall that a *Boolean conjunctive query* over the database $\mathcal{D}$ is an existential sentence whose quantifier-free part is a conjunction of atoms. CSP($\mathcal{D}$) is exactly the problem of deciding whether $\mathcal{D}$ satisfies a given Boolean conjunctive query. For example, the instance

$$R_1(x),\ R_1(w),\ R_3(y,y,z),\ R_7(y,w),\ R_7(x,y) \tag{1}$$

has a solution if and only if the sentence

$$(\exists x, y, z, w \in D)\ R_1(x) \wedge R_1(w) \wedge R_3(y,y,z) \wedge R_7(y,w) \wedge R_7(x,y)$$

is true in $\mathcal{D}$.

From this perspective, it is natural to ask what happens if we allow some other combination of logical connectives from $\{\exists, \forall, \wedge, \vee, \neg, =, \neq\}$. It turns out that out of the 2^7 cases only 3 are interesting (the other cases either reduce to these, or are almost always easy or hard by known results): $\{\exists, \wedge\}$ which is CSP, $\{\exists, \forall, \wedge\}$ which is so-called *quantified CSP*, and $\{\exists, \forall, \wedge, \vee\}$. Determining the complexity of quantified CSP is also an active research area [49] with a possible trichotomy – P, NP-complete or PSPACE-complete. Recently, a tetrachotomy was obtained for the last case [101] – for every $\mathcal{D}$, the corresponding problem is either in P, NP-complete, co-NP-complete, or Pspace-complete.

The CSP over a fixed language can also be formulated as the *homomorphism problem* between relational structures with a fixed target structure [65, 78]. Assume that we have two relational structures $\mathcal{E} = (E; S_1, S_2, \dots)$ and $\mathcal{D} = (D; R_1, R_2, \dots)$ which are similar, i.e. they have the same number of relations and the corresponding relations have the same arity. A *homomorphism* from $\mathcal{E}$ to $\mathcal{D}$ is a mapping $h : E \to D$ such that, for all i, if $\mathbf{a} = (a_1, a_2, \dots) \in S_i$ then $h(\mathbf{a}) = (h(a_1), h(a_2), \dots) \in R_i$. Then CSP($\mathcal{D}$) is equivalent to the problem of deciding whether a given relational structure $\mathcal{E}$ similar to $\mathcal{D}$ has a homomorphism to $\mathcal{D}$. The idea of the translation is shown in Examples 6 and 9. In general, to see the translation from the homomorphism form to the constraint form, view the set E as the set of variables and transform every tuple $\mathbf{x}$ in a relation S_i in $\mathcal{E}$ to a constraint $R_i(\mathbf{x})$. To see the translation back, let E (the domain of $\mathcal{E}$) be the set of all variables appearing in a given CSP instance, and let each relation S_i contain all tuples $\mathbf{x}$ such that this CSP instance contains a constraint $R_i(\mathbf{x})$. For example, if we translate the CSP($\mathcal{D}$) instance appearing above in (1) into a relational structure $\mathcal{E}$, then we have $E = \{x, y, z, w\}$ and $S_1 = \{x, w\}, S_3 = \{(y, y, z)\}, S_7 = \{(y, w), (x, y)\}$, with all the other relations S_i empty.

3 Reductions Between Constraint Languages

This section describes relational constructions that allow one to reduce one CSP with a fixed constraint language to another. These constructions have algebraic counterparts, described in the following section, and this translates many (complexity) classification problems about constraint languages into algebraic classifications.

If a computational problem $\mathcal{A}$ can simulate (in some sense) another problem $\mathcal{B}$, then $\mathcal{A}$ is at least as hard as $\mathcal{B}$. This simple idea is widely used in computational complexity; for instance, NP-completeness is often shown by a gadget reduction of a known NP-complete problem to the given one. A crucial fact for the algebraic theory of the CSP is that a so called primitive positive (pp-, for short) interpretation between constraint languages gives such a reduction between corresponding CSPs (more precisely, if $\mathcal{D}$ pp-interprets $\mathcal{E}$, then $\mathrm{CSP}(\mathcal{E})$ is reducible to $\mathrm{CSP}(\mathcal{D})$). Pp-interpretations have been, indirectly, one of the main subjects of universal algebra for the last 80 years!

We will define three increasingly more general techniques for simulation between CSPs:

$$\text{pp-definition} \ \subseteq\ \text{pp-interpretation} \ \subseteq \text{pp-construction}\,.$$

In Section 4 we give algebraic characterizations for these techniques, which guide the algebraic approach.

The algebraic theory of CSPs was developed in a number of papers including [80, 78, 34, 97, 20]. The viewpoint taken here is close to [20, 24]. All results in this section come from these sources unless stated otherwise.

To simplify formulations, all structures (relational or algebraic) are assumed to have finite domains, all constraint languages are assumed to contain *finitely many* relations, all of them *nonempty*. By a *reduction* we mean a log-space reduction (although first-order reductions are often possible under additional weak assumptions).

3.1 Primitive Positive Interpretations (= Gadgets)

An important special case of pp-interpretability is pp-definability.

▶ **Definition 11.** Let $\mathcal{D}, \mathcal{E}$ be constraint languages on the same domain $D = E$. We say that $\mathcal{D}$ *pp-defines* $\mathcal{E}$ (or $\mathcal{E}$ is pp-definable from $\mathcal{D}$) if each relation in $\mathcal{E}$ can be defined by a first order formula which only uses relations in $\mathcal{D}$, the equality relation, conjunction and existential quantification.

This terminology comes from model theory, where a first order formula is called *primitive* if it has the form $\exists \bar{y} \bigwedge_{i<n} \alpha_i(\bar{x}, \bar{y})$ where each $\alpha_i(\bar{x}, \bar{y})$ is an atomic or negated atomic formula. A *primitive positive* (or pp-) formula is a negation-free primitive formula.

Rephrasing the above definition without using logic, $\mathcal{D}$ pp-defines $\mathcal{E}$ if $\mathcal{D}$ and $\mathcal{E}$ have the same domain and every relation R_i in $\mathcal{E}$ can be represented by a gadget using relations from $\mathcal{D}$, as follows. There is an instance P_i of $\mathrm{CSP}(\mathcal{D} \cup \{=\})$ and subset X_i of variables in P_i such that the set of all solutions to P_i, when projected down to X_i, gives precisely the relation R_i.

▶ **Example 12.** Recall constraint languages $\mathcal{D}_{\mathrm{3SAT}}, \mathcal{D}_{\mathrm{Horn3SAT}}$, and $\mathcal{D}_{\mathrm{STCON}}$ from Examples 3, 5, and 9. We now show that $\mathcal{D}_{\mathrm{3SAT}}$ pp-defines $\mathcal{D}_{\mathrm{Horn3SAT}}$, which in turn pp-defines $\mathcal{D}_{\mathrm{STCON}}$. To show the first definition, notice that $C_0(x) = S_{111}(x,x,x)$ and $C_1(x) = S_{000}(x,x,x)$. For the second, it is enough to check that $I(x,y)$ holds if and only if $\exists z (C_1(z) \wedge S_{110}(z,x,y))$ holds.

▶ **Theorem 13.** *If* $\mathcal{D}$ *pp-defines* $\mathcal{E}$*, then* $\mathrm{CSP}(\mathcal{E})$ *is reducible to* $\mathrm{CSP}(\mathcal{D})$*.*

Proof by Example. Let R be an arbitrary ternary relation on a domain D. Consider the relations on D defined by

$$S(x,y) \text{ iff } (\exists z) R(x,y,z) \wedge R(y,y,x), \quad T(x,y) \text{ iff } R(x,x,x) \wedge (x=y) \ ,$$

where the existential quantification is understood over D. The relations S and T are defined by pp-formulae, therefore the constraint language $\mathcal{D} = \{R\}$ pp-defines the constraint language $\mathcal{E} = \{S, T\}$.

We sketch the reduction of CSP($\mathcal{E}$) to CSP($\mathcal{D}$) using the instance

$$S(x_3,x_2),\ T(x_1,x_4),\ S(x_2,x_4) \ .$$

We first replace S and T with their pp-definitions by introducing a new variable for each quantified variable:

$$R(x_3,x_2,y_1),\ R(x_2,x_2,x_3), \quad R(x_1,x_1,x_1),\ x_1 = x_4, \quad R(x_2,x_4,y_2),\ R(x_4,x_4,x_2)$$

and then we get rid of the equality constraint $x_1 = x_4$ by identifying these variables. This way we obtain an instance of CSP($\mathcal{D}$):

$$R(x_3,x_2,y_1),\ R(x_2,x_2,x_3),\ R(x_1,x_1,x_1),\ R(x_2,x_1,y_2),\ R(x_1,x_1,x_2) \ .$$

Clearly, the new instance of CSP($\mathcal{D}$) has a solution if and only if the original instance does. ◀

This simple theorem provides a quite powerful tool for comparing CSPs over different languages *on the same domain*. A more powerful tool, which can also be used to compare languages with different domains, is pp-interpretability. Informally, a constraint language $\mathcal{D}$ pp-interprets $\mathcal{E}$, if the domain of $\mathcal{E}$ is a pp-definable relation (from $\mathcal{D}$) modulo a pp-definable equivalence, and the relations of $\mathcal{E}$ (viewed, in a natural way, as relations on D) are also pp-definable from $\mathcal{D}$.[4] Formally:

▶ **Definition 14.** Let $\mathcal{D}, \mathcal{E}$ be constraint languages. We say that $\mathcal{D}$ *pp-interprets* $\mathcal{E}$ if there exist a natural number n, a set $F \subseteq D^n$, and an onto mapping $f : F \to E$ such that $\mathcal{D}$ pp-defines

- the relation F,
- the f-preimage of the equality relation on E, and
- the f-preimage of every relation in $\mathcal{E}$,

where by the f-preimage of a k-ary relation S on E we mean the nk-ary relation $f^{-1}(S)$ on D defined by

$$f^{-1}(S)(x_{11},\dots,x_{1k},x_{21},\dots,x_{2k},\dots,x_{n1},\dots,x_{nk})$$

iff

$$S(f(x_{11},\dots,x_{n1}),\dots,f(x_{1k},\dots,x_{nk})) \ .$$

When $F = E = D^n$ and f is the identity mapping, we also say that $\mathcal{E}$ is a *pp-power* of $\mathcal{D}$.

▶ **Theorem 15.** *If $\mathcal{D}$ pp-interprets $\mathcal{E}$, then* CSP($\mathcal{E}$) *is reducible to* CSP($\mathcal{D}$).

Proof Sketch. The properties of the mapping f from Definition 14 allow us to rewrite an instance of CSP($\mathcal{E}$) to an instance of the CSP over a constraint language which is pp-definable from $\mathcal{D}$. Then we apply Theorem 13. ◀

[4] This is the classical notion of interpretation from model theory restricted to pp-formulas.

3.2 Homomorphic Equivalence, Cores and Singleton Expansions

Let $\mathcal{D}$ and $\mathcal{E}$ be constraint languages with domains D and E, respectively. We say that $\mathcal{D}$ and $\mathcal{E}$ are *homomorphically equivalent* if the relations in them can be ordered so that $\mathcal{D}$ and $\mathcal{E}$ become similar structures and there exist homomorphisms $e : \mathcal{D} \to \mathcal{E}$ and $g : \mathcal{E} \to \mathcal{D}$ (recall definitions from Section 2.4).

▶ **Theorem 16.** *Let $\mathcal{D}$ and $\mathcal{E}$ be homomorphically equivalent constraint languages. Then* $\mathrm{CSP}(\mathcal{D})$ *and* $\mathrm{CSP}(\mathcal{E})$ *are reducible to each other.*

Proof Idea. An instance of $\mathrm{CSP}(\mathcal{D})$ has a solution if and only if the corresponding instance of $\mathrm{CSP}(\mathcal{E})$, obtained by replacing each $R_i \in \mathcal{D}$ with $S_i \in \mathcal{E}$, has a solution. Specifically, direct applications of mappings e and g transform solutions of one instance to solutions of another one. ◀

A mapping $f : D \to D$ is called an *endomorphism* of $\mathcal{D}$ if it is a homomorphism from $\mathcal{D}$ to itself, that is, $f(R) := \{(f(a_1), f(a_2), \dots) \mid (a_1, a_2, \dots) \in R\} \subseteq R$ for every $R \in \mathcal{D}$.

A language $\mathcal{D}$ is a *core* if every endomorphism of $\mathcal{D}$ is a bijection. It is not hard to show that if f is an endomorphism of a constraint language $\mathcal{D}$ with minimal range, then $f(\mathcal{D}) = \{f(R) \mid R \in \mathcal{D}\}$ is a core. Moreover, this core is unique up to isomorphism, therefore we speak about *the core* of $\mathcal{D}$.

An important fact is that we can add all singleton unary relations to a core constraint language without increasing the complexity of its CSP. For a constraint language $\mathcal{D}$, its *singleton expansion* is the language $\mathcal{E} = \mathcal{D} \cup \{C_a : a \in D\}$, where C_a denotes the unary relation $C_a = \{a\}$.

▶ **Theorem 17.** *Let $\mathcal{D}$ be a core constraint language and let $\mathcal{E}$ be the singleton expansion of $\mathcal{D}$. Then* $\mathrm{CSP}(\mathcal{E})$ *is reducible to* $\mathrm{CSP}(\mathcal{D})$.

Proof Idea. The crucial step is to observe that the set of endomorphisms of $\mathcal{D}$, viewed as a $|D|$-ary relation, is pp-definable from $\mathcal{D}$. More precisely, the relation

$$S = \{(f(a_1), \dots, f(a_n)) : f \text{ is an endomorphism of } \mathcal{D}\} \ ,$$

where $a_1, \dots, a_n$ is a list of all elements of D, is pp-definable from $\mathcal{D}$ (even without existential quantification). Indeed, f is, by definition, an endomorphism of $\mathcal{D}$ if for every $R \in \mathcal{D}$ of arity $\mathrm{ar}(R)$ and every $(b_1, \dots, b_{\mathrm{ar}(R)}) \in R$ we have $(f(b_1), \dots, f(b_{\mathrm{ar}(R)})) \in R$. This directly leads to a pp-definition of S:

$$S(x_{a_1}, \dots, x_{a_n}) \text{ iff } \bigwedge_{R \in \mathcal{D}} \ \bigwedge_{(b_1, \dots, b_{\mathrm{ar}(R)}) \in R} R(x_{b_1}, \dots, x_{b_{\mathrm{ar}(R)}}) \ .$$

Given an instance of $\mathrm{CSP}(\mathcal{E})$ we introduce new variables $x_{a_1}, \dots, x_{a_n}$, we replace every constraint of the form $C_a(x)$ by $x = x_a$, and we add the constraint $S(x_{a_1}, \dots, x_{a_n})$. In this way we obtain an instance of $\mathrm{CSP}(\mathcal{D} \cup \{=\})$. Clearly, if the original instance has a solution, then the new instance has a solution as well. In the other direction, if g is a solution to the new instance, then its values on $x_{a_1}, \dots, x_{a_n}$ determine an endomorphism f of $\mathcal{D}$. As $\mathcal{D}$ is a core, f is a bijection, thus f^{-1} is an endomorphism as well, and $f^{-1} \circ g$ restricted to the original variables is a solution of the original instance. ◀

We will call constraint languages containing all singleton unary relations *idempotent*. Note that an idempotent constraint language is automatically a core as the only endomorphism is the mapping that sends each element to itself.

An interesting property of an idempotent constraint language $\mathcal{D}$ is that the search problem for CSP($\mathcal{D}$) is solvable in polynomial time whenever CSP($\mathcal{D}$) is. The idea is to use self-reduction: for a satisfiable instance, find good values for variables, in order, by checking satisfiability of the instance enhanced with appropriate unary singleton constraints.

3.3 Example

▶ **Example 18.** We show that 3-SAT is reducible to 3-COLORING via singleton expansion and a pp-interpretation with $n = 1$.

Recall the constraint language $\mathcal{D}_{3\mathrm{COLOR}} = \{\neq_3\}$ of 3-COLORING from Example 6 and the constraint language $\mathcal{D}_{3\mathrm{SAT}} = \{S_{000}, \ldots, S_{111}\}$ of 3-SAT from Example 3.

Since $\mathcal{D}_{3\mathrm{COLOR}}$ is a core, CSP($\mathcal{D}'_{3\mathrm{COLOR}}$), where $\mathcal{D}'_{3\mathrm{COLOR}} = \{\neq_3, C_0, C_1, C_2\}$, is reducible to CSP($\mathcal{D}_{3\mathrm{COLOR}}$) by Theorem 17. By Theorem 15, it is now enough to show that $\mathcal{D}'_{3\mathrm{COLOR}}$ pp-interprets $\mathcal{D}_{3\mathrm{SAT}}$. We give a pp-interpretation with $n = 1$, $F = \{0,1\}$, and f the identity map (see Definition 14). The unary relation $\{0,1\}$ can be pp-defined by

$$F(x) \ \text{ iff } \ (\exists y)\ C_2(y) \wedge x \neq_3 y \quad (\text{iff } x \neq 2) \ .$$

The preimage of the equality relation is the equality relation on $\{0,1\}$ which is clearly pp-definable. The relation S_{000} can be defined by

$$S_{000}(x_1,x_2,x_3) \ \text{ iff } \ (\exists y_1,y_2,y_3,z)\ C_2(z) \wedge y_1 \neq_3 y_2 \wedge y_2 \neq_3 y_3 \wedge y_1 \neq_3 y_3 \\ \wedge \bigwedge_{i=1,2,3} z \neq_3 x_i \wedge T(x_i,y_i) \ ,$$

where T is the binary relation

$$T(x,y) \ \text{ iff } \ (\exists u,v)\ C_1(u) \wedge u \neq v \wedge x \neq v \wedge y \neq v$$

The other relations S_{ijk} are defined similarly.

While it is easy to verify that the presented pp-definitions work, it is not so easy to find them without any tools. The proof of Theorem 32 gives an algorithm to produce pp-definitions whenever they exist (although the obtained definitions will usually be very long).

3.4 Pp-Constructibility

We now discuss how the reductions from the previous two subsections can be combined.

▶ **Definition 19.** A constraint language $\mathcal{D}$ *pp-constructs* a constraint language $\mathcal{E}$ if there is a sequence of constraint languages $\mathcal{D} = \mathcal{C}_1, \ldots, \mathcal{C}_k = \mathcal{E}$ such that, for each $1 \leq i < k$

- $\mathcal{C}_i$ pp-interprets $\mathcal{C}_{i+1}$, or
- $\mathcal{C}_i$ is homomorphically equivalent to $\mathcal{C}_{i+1}$, or
- $\mathcal{C}_i$ is a core and $\mathcal{C}_{i+1}$ its singleton expansion.

The following is a corollary of Theorems 15, 16, and 17.

▶ **Corollary 20.** *If $\mathcal{D}$ pp-constructs $\mathcal{E}$, then* CSP($\mathcal{E}$) *is reducible to* CSP($\mathcal{D}$).

It turns out that any finite sequence of operations in pp-constructibility can be replaced by only two operations. Recall the notion of pp-power from Definition 14.

▶ **Theorem 21.** *A constraint language $\mathcal{D}$ pp-constructs a constraint language $\mathcal{E}$ if and only if $\mathcal{E}$ is homomorphically equivalent to a pp-power of $\mathcal{D}$.*

An example of idempotent constraint languages $\mathcal{D}$ and $\mathcal{E}$ such that D pp-constructs $\mathcal{E}$, but does not pp-interpret $\mathcal{E}$, can be found in [20].

3.5 Tractability Conjecture

Pp-constructibility is a reflexive and transitive relation on the class of constraint languages. By identifying equivalent languages, i.e. languages which mutually pp-construct each other, we get a partially ordered set, the *pp-constructibility poset*, in which $\mathcal{D} \leq \mathcal{E}$ iff $\mathcal{D}$ pp-constructs $\mathcal{E}$. Corollary 20 then says that the "higher" we are in the poset the "easier" the CSP we are dealing with. 3-SAT is terribly hard – we will see later (see Example 33) that its constraint language is the least element of this poset. Strikingly, all known NP-complete CSPs have this property. Bulatov, Jeavons and Krokhin [34] conjectured that this is not a coincidence.

▶ **Conjecture 22** (Tractability Conjecture). *If a constraint language $\mathcal{D}$ does not pp-construct the language of* 3-SAT*, then* CSP($\mathcal{D}$) *is solvable in polynomial time.*

This conjecture (together with the matching hardness result) is also known as the *algebraic dichotomy conjecture* because many equivalent formulations, including the original one, are stated in terms of algebraic operations; see subsection 4.4.

Actually, the original conjecture in [34] was stated (in an equivalent algebraic form) for the case when $\mathcal{D}$ is an idempotent language and instead of pp-construction it used what was essentially pp-interpretation with $n = 1$, but this is equivalent to the conjecture stated above (see [33, 20]).

Remarkably, the seminal paper of Feder and Vardi included a conjecture very similar to the Tractability conjecture; see [65, Conjecture 2]. In essence, their conjecture was that CSP($\mathcal{D}$) should be solvable in polynomial time provided the core of $\mathcal{D}$ does not pp-define a constraint language whose core is isomorphic to the language of 1-in-3-SAT (see Example 4). This conjecture as stated is false. Indeed, the language $\mathcal{D}_{3\text{COLOR}} = \{\neq_3\}$ of 3-COLORING is a core, it is invariant under under all permutations of $\{0, 1, 2\}$, and it is easy to see that all relations pp-definable in $\mathcal{D}_{3\text{COLOR}}$ also have this invariance property. Therefore, $\mathcal{D}_{3\text{COLOR}}$ cannot pp-define any relation whose core has a two-element domain, and yet CSP($\mathcal{D}_{3\text{COLOR}}$) is obviously NP-complete. Note that we showed in Example 18 that the singleton expansion of $\mathcal{D}_{3\text{COLOR}}$ can pp-interpret (with $n = 1$) the language of 3-SAT. In fact, it follows (see Example 33) that $\mathcal{D}_{3\text{COLOR}}$ pp-constructs all constraint languages.

Similar hardness results and conjectures have been formulated for other computational/-descriptive complexity classes. See subsection 4.4.

3.6 Other Reductions

Recall that if two constraint languages pp-construct each other, then their corresponding CSP problems are equivalent up to logspace reductions. Thus to understand the complexity of CSPs (for example, to resolve the Tractability conjecture), it suffices to consider just one constraint language from each equivalence class in the pp-constructibility poset.

By combining Theorems 16 and 17, we obtain the "reduction to the idempotent case."

▶ **Theorem 23.** *For every constraint language $\mathcal{D}$ there is an idempotent constraint language $\mathcal{D}'$ such that $\mathcal{D}$ and $\mathcal{D}'$ pp-construct each other.*

The following theorem has appeared in various closely related forms in the literature. It is useful because it allows one to work only with binary constraints, which often simplifies the design and analysis of algorithms for CSPs.

▶ **Theorem 24.** *For any constraint language $\mathcal{D}$, there is a constraint language $\mathcal{D}'$ such that*

- *all relations in $\mathcal{D}'$ are at most binary, and*
- *$\mathcal{D}$ and $\mathcal{D}'$ pp-construct each other.*

Proof Sketch. Let ℓ be the maximum arity of a relation in $\mathcal{D}$. Define a constraint language $\mathcal{D}'$ as follows. Let $D' = D^\ell$. For each relation R (say of arity k) in $\mathcal{D}$, $\mathcal{D}'$ contains a unary relation R' such that $(a_1, \ldots, a_\ell) \in R'$ if and only if $(a_1, \ldots, a_k) \in R$. In addition, for all $1 \leq k \leq \ell$, $\mathcal{D}'$ contains a binary relation E_k defined as follows: $((a_1, \ldots, a_\ell), (b_1, \ldots, b_\ell)) \in E_k$ if and only if $a_1 = b_k$. It can be seen directly from definitions that $\mathcal{D}'$ is a pp-power of $\mathcal{D}$, and so $\mathcal{D}$ pp-constructs $\mathcal{D}'$.

In the opposite direction, for each unary relation $R' \in \mathcal{D}'$, consider the following relation on D': $R''(x_1, \ldots, x_k) = \exists x_R : R'(x_R) \wedge E_1(x_1, x_R) \wedge \ldots \wedge E_k(x_k, x_R)$. Let $\mathcal{D}'' = \{R'' \mid R \in \mathcal{D}\}$, so $\mathcal{D}'$ pp-defines $\mathcal{D}''$. Now, it is straightforward to check that $\mathcal{D}$ and $\mathcal{D}''$ are homomorphically equivalent, with mappings $e : D \to D^\ell$ and $g : D^\ell \to D$ defined as $e(x) = (x, \ldots, x)$ and $g((x_1, \ldots, x_\ell)) = x_1$. ◀

By using Theorem 23, Theorem 24 can be strengthened to make $\mathcal{D}'$ idempotent.

At the expense of forgoing pp-constructible equivalence, we can replace any constraint language with a language consisting of a single binary relation. The following result is essentially from [65]; see also the improvement in [45].

▶ **Theorem 25.** *For every constraint language $\mathcal{D}$ there is a digraph $H = (V, E)$ such that*

1. *$\{E\}$ pp-constructs $\mathcal{D}$, and*
2. *$\mathrm{CSP}(\{E\})$ is logspace-reducible to (and hence equivalent to) $\mathrm{CSP}(\mathcal{D})$.*

It is known [83] that the previous theorem cannot be improved so that $\mathcal{D}$ and $\{E\}$ each pp-construct the other.

It follows from the previous theorem that every $\mathrm{CSP}(\mathcal{D})$ is logspace-equivalent to an H-coloring problem. In a similar vein, Feder and Vardi proved [65] that every $\mathrm{CSP}(\mathcal{D})$ is logspace-equivalent to some $\mathrm{CSP}(\mathcal{E})$ where $\mathcal{E}$ is the singleton expansion of a partial ordering, and also to $\mathrm{CSP}(\mathcal{E}')$ where $\mathcal{E}'$ is the singleton expansion of the (symmetric irreflexive) edge relation of a bipartite graph. These results, while undoubtedly interesting, might be taken to suggest that the CSP classification problem needs to be tackled through a careful analysis of combinatorial objects such as digraphs, posets, or bipartite graphs. However, another interpretation is that these objects are complex enough to encode all CSPs. The algebraic approach, that we are about to describe, gives a better, more fruitful, alternative for complexity analysis of CSPs.

4 Polymorphisms as Classifiers of Constraint Languages

4.1 Definitions and Examples

The link between relations and operations is provided by a natural notion of compatibility. An n-ary operation f on a finite set D (that is, a mapping $f : D^n \to D$) is *compatible* with a k-ary relation $R \subseteq D^k$ if f applied component-wise to any n-tuple of elements of R gives an element of R. In more detail, whenever (a_{ij}) is an $n \times k$ matrix such that every row is in R, then f applied to the columns gives a k-tuple which is in R as well. If f is compatible with R then one also says that f is a *polymorphism* of R, and that R is *invariant* under f.

▶ **Example 26.** Consider the ternary *majority* operation f on $\{0, 1\}$, which always returns the (unique) repeated value among its arguments. It is a very easy exercise to check that this operation is compatible with any binary relation on $\{0, 1\}$: indeed, applying f to the columns of any 3×2 matrix with $0/1$ entries always gives one of the rows of this matrix.

We say that an operation f on D is a *polymorphism* of a constraint language $\mathcal{D}$ if f is compatible with every relation in $\mathcal{D}$. Note that a unary polymorphism is the same as an endomorphism. Endomorphisms can be thought of as symmetries, so polymorphisms can be viewed as symmetries of higher arities.

▶ **Example 27.** Every polymorphism f of an idempotent constraint language is *algebraically idempotent*, that is, it satisfies $f(a, a, \dots, a) = a$ for each a in the domain.

▶ **Example 28.** It is very easy to check that the binary operation $f(x, y) = \min(x, y)$ on $\{0, 1\}$ is a polymorphism of $\mathcal{D}_{\text{HornSAT}}$ from Example 5. For example, to see that f is compatible with the relation $S_{110} = \{0, 1\}^3 \setminus \{(1, 1, 0)\}$, notice that, for any 2×3 matrix with 0/1 entries, if f applied to the columns of the matrix gives tuple $(1, 1, 0)$ then one of the rows of the matrix must be this same tuple.

▶ **Example 29.** The ternary operation $f(x, y, z) = x - y + z \mod p$ on $\mathrm{GF}(p)$ is a polymorphism of $\mathcal{D}_{\text{3LINp}}$ from Example 8. Indeed, each relation in this structure is an affine subspace of $\mathrm{GF}(p)^3$ and f applied to the columns of a 3×3 matrix gives a triple, which is an affine combination of its rows (with coefficients 1, -1, 1).

In fact, it is easy to check that f is compatible with $R \subseteq \mathrm{GF}(p)$ if and only if it is an affine subspace of $\mathrm{GF}(p)$. It follows that if f is a polymorphism of $\mathcal{D}$, then an instance of $\mathrm{CSP}(\mathcal{D})$ can be written as a system of linear equations over $\mathrm{GF}(p)$ and therefore $\mathrm{CSP}(\mathcal{D})$ is solvable in polynomial time, for example, by Gaussian elimination.

Observe that for $p = 2$, f is the ternary *minority* function $f(x, y, z) = x + y + z \mod 2$.

▶ **Example 30.** Consider the following generalisation of the operation f from Example 26. For any finite set D, the *dual discriminator* operation on D is the ternary operation d such that $d(x, y, z) = x$ if x, y, z are all different and, otherwise, $d(x, y, z)$ is the repeated value among x, y, z. It is a useful easy exercise to check that d is compatible with a binary relation R on D if and only if the relation has one of the following forms:

($\vee$) $x = a \vee y = b$ for $a, b \in D$,

(π) $x = \pi(y)$ where π is a permutation on D,

($\times$) $A \times B$ where A and B are subsets of D,

($\cap$) intersection of a relation of type ($\vee$) or (π) with a relation of type ($\times$).

The key observation is that, for any binary relation R compatible with d, and for any $a, a', b, b', c \in D$ with $b \neq b'$, we have $d((a', c)(a, b), (a, b')) = (a, c)$. We remark that, generally, it is rare to have such an explicit description of relations having a given polymorphism.

Another useful way of viewing polymorphisms is that they provide an algebraic structure on solution sets of instances. In other words, they provide a uniform way to combine solutions to instances to form a new solution, which we illustrate with the following example.

▶ **Example 31.** Recall the majority operation f on $\{0, 1\}$ from Example 26. Consider any 2-SAT instance, for example, this one: $(x \vee \overline{y}) \wedge (y \vee \overline{z}) \wedge (y \vee \overline{u}) \wedge (x \vee u)$. Take any three solutions to this instance: for example, $\mathbf{a}, \mathbf{b}, \mathbf{c}$ described in the diagram below. It is easy to check that they are indeed solutions: simply check that each of them satisfies each constraint in the instance. If we apply f to these solutions coordinate-wise, as described in the diagram, we obtain a new assignment $f(\mathbf{a}, \mathbf{b}, \mathbf{c})$. It is also a solution to this instance, and there is no need to go through the constraints in the instances to check that each constraint is satisfied, since this is directly guaranteed by the fact that f is compatible with all binary relations on $\{0, 1\}$.

$$\begin{array}{rcccccccl} & & x & y & z & u & & \\ \mathbf{a} = & (& 1 & 1 & 1 & 0 &) & \text{sat} \\ \mathbf{b} = & (& 1 & 1 & 0 & 1 &) & \text{sat} \\ \mathbf{c} = & (& 1 & 0 & 0 & 0 &) & \text{sat} \\ & & f\downarrow & f\downarrow & f\downarrow & f\downarrow & & \\ f(\mathbf{a},\mathbf{b},\mathbf{c}) = & (& 1 & 1 & 0 & 0 &) & \text{sat} \end{array}$$

Notice that f cannot be used to combine solutions to HORN-3-SAT or 3-SAT instances. Indeed, the relation $S_{111} = \{0,1\}^3 \setminus \{(1,1,1)\}$ is not compatible with f; it is easy to see that applying f to tuples $(0,1,1), (1,0,1), (1,1,0) \in S_{111}$ gives $(1,1,1)$. We discuss polymorphisms of $\mathcal{D}_{3\text{SAT}}$ again in Example 33.

We remark that many algorithms that we discuss in the subsequent sections use polymorphisms, in design or in analysis, to combine solutions of subinstances of a given CSP instance in order to maintain or improve useful problem-specific properties of such solutions.

4.2 Polymorphisms as an Algebraic Counterpart of pp-Definability

The set of all polymorphisms of $\mathcal{D}$ will be denoted by $\mathbf{D}$. This algebraic object has the following two properties.

- $\mathbf{D}$ *contains all projections*[5], that is, all operations of the form

  $$\pi_i^n(a_1,\ldots,a_n) = a_i.$$

- $\mathbf{D}$ *is closed under composition*, that is, any operation built from operations in $\mathbf{D}$ by composition also belongs to $\mathbf{D}$.

For example, if $\mathbf{D}$ contains a unary operation h, a binary operation g and a ternary operation f then the operation $f'(x,y,z) = f(f(x,x,h(y)), g(y,z), g(z,x))$ is also in $\mathbf{D}$.

Sets of operations with these properties are called *concrete clones* (or *function clones*, or simply *clones*); therefore we refer to $\mathbf{D}$ as the *clone of polymorphisms* of $\mathcal{D}$ [6]. It is known that every concrete clone is the clone of polymorphisms of some (possibly infinite) constraint language [67, 26].

The notions of polymorphism and invariance form the basis of a well-known Galois correspondence between sets of relations and operations on a finite set [67, 26], which implies that the clone of polymorphisms controls pp-definability in the following sense.

▶ **Theorem 32.** *Let $\mathcal{D}$, $\mathcal{E}$ be constraint languages with $D = E$. Then $\mathcal{D}$ pp-defines $\mathcal{E}$ if and only if $\mathbf{D} \subseteq \mathbf{E}$.*

Proof Sketch. The implication "$\Rightarrow$" follows directly from definitions. For the other implication it is enough to prove that if R is a relation compatible with every polymorphism of $\mathcal{D}$, then R is pp-definable from $\mathcal{D}$. A crucial step is a more general version of the observation made in the proof of Theorem 17: For any k, the set of k-ary polymorphisms of $\mathcal{D}$ can be viewed as a $|D|^k$-ary relation S on D, and this relation is pp-definable from $\mathcal{D}$. Now R can be defined from such a relation S (where k is the number of tuples in R) by existential quantification over suitable coordinates. This proof is illustrated in Example 34 below. ◀

[5] In some research communities such operations are called dictators.
[6] Similar fonts will be used to denote other languages and their corresponding clones, e.g. $\mathcal{E}$ and $\mathbf{E}$.

In view of this result, Theorem 13 says that the complexity of CSP($\mathcal{D}$) only depends on the clone $\mathbf{D}$. More precisely, if $\mathbf{D} \subseteq \mathbf{E}$, then CSP($\mathcal{E}$) is reducible to CSP($\mathcal{D}$). Moreover, the proof of Theorem 32 gives a generic pp-definition of $\mathcal{E}$ from $\mathcal{D}$, which gives us a generic reduction of CSP($\mathcal{E}$) to CSP($\mathcal{D}$).

▶ **Example 33.** It is a nice exercise to show that the language $\mathcal{D}_{3\mathrm{SAT}}$ of 3-SAT has no polymorphisms except for the projections, and the same holds for the language of 1-in-3-SAT. This means (by Theorem 32) that $\mathcal{D}_{3\mathrm{SAT}}$ pp-defines every constraint language with domain $\{0,1\}$. It follows (see also Theorem 38) that $\mathcal{D}_{3\mathrm{SAT}}$ pp-constructs (in fact, even pp-interprets) every constraint language, so it is the least element of the pp-constructibility poset, as claimed earlier. Moreover, it follows from Theorem 32 that $\mathcal{D}_{3\mathrm{SAT}}$ and $\mathcal{D}_{1\mathrm{in}3\mathrm{SAT}}$ pp-define each other, hence they are in the same (i.e. the least) element of the pp-constructibility poset.

▶ **Example 34.** Another nice exercise for the reader is to show that the language $\mathcal{D}'_{3\mathrm{COLOR}} = \{\neq_3, C_0, C_1, C_2\}$ on the domain $\{0,1,2\}$ (see Example 18) also does not have any polymorphisms except for projections. It follows that every relation on $\{0,1,2\}$ is pp-definable from $\mathcal{D}'_{3\mathrm{COLOR}}$. We show how the proof of Theorem 32 produces a pp-definition of some relation, say, the binary relation

$$R = \{(0,1),(0,2),(1,1),(2,2)\} \ .$$

Since R contains 4 pairs, we pp-define the 3^4-ary relation

$$S = \{(f(0,0,0,0), f(0,0,0,1), \dots, f(2,2,2,2)) : f \text{ is a 4-ary polymorphism of } \mathcal{D}'_{3\mathrm{COLOR}}\}.$$

which corresponds to the set of all 4-ary polymorphisms of $\mathcal{D}'_{3\mathrm{COLOR}}$:

$$S(x_{0000}, \dots, x_{2222}) \ \text{ iff } \ \bigwedge_i x_{iiii} = i \wedge \bigwedge_{i_1 \neq i_2, j_1 \neq j_2, k_1 \neq k_2, l_1 \neq l_2} x_{i_1 j_1 k_1 l_1} \neq_3 x_{i_2 j_2 k_2 l_2} \ .$$

Let's see that the above formula actually defines S: it is clear that indices show how to interpret each 3^4-tuple in S as a 4-ary operation on $\{0,1,2\}$. The first $\bigwedge$-part in the above formula states that the interpreted operation is compatible with C_0, C_1, C_2, while the second one states that it is compatible with $\neq_3$. Now we existentially quantify over all variables in S but x_{0012} and x_{1212} – the exceptions are those variables whose indices correspond to the first and the second (resp.) coordinates of pairs in R. The obtained binary relation $R'(x_{0012}, x_{1212})$ contains R since S contains the four tuples corresponding to the cases when f is a projection $\pi_i^4 : \{0,1,2\}^4 \to \{0,1,2\}$, and R' is contained in R since R is compatible with every polymorphism of $\mathcal{D}'_{3\mathrm{COLOR}}$.

Note that the definition of S_{000} from Example 18 obtained in this way contains 3^7 variables.

For other examples similar to Example 34, but worked out in more detail, see [77].

The proof of Theorem 32 (see also the above example) gives an algorithm for constructing a pp-definition of a relation R from a given structure $\mathcal{D}$, whenever there is one. This pp-definition can be terribly long. However, the algorithm is optimal in the sense that the problem of deciding of whether such a pp-definition exists is co-NEXPTIME-hard [118].

4.3 Height-1 Identities and pp-Constructibility

We explained above that the complexity of CSP($\mathcal{D}$) depends only on the polymorphisms of $\mathcal{D}$. In this section, we refine this statement by specifying the properties of polymorphisms that determine the complexity: these are *identities* satisfied by polymorphisms, i.e. equations that hold for all choices of values for the variables, and more specifically *height-1* identities, i.e. those in which each side has exactly one occurrence of an operation symbol.

We will explain identities by example.

▶ **Example 35.** A binary operation f on D is called a *semilattice* operation if it satisfies the following three *identities*

$$f(f(x,y),z) = f(x,f(y,z)),\ f(x,y) = f(y,x),\ \text{and } f(x,x) = x,$$

which means that the above equalities hold for all choices of values in D for the variables. The first identity above (known as associativity) is not height-1, as both sides have two occurrences of f. The second identity (commutativity) is height-1. The third identity (idempotence) is not height-1 as its right side has no occurrence of an operation symbol.

In general, identities can involve more than one operation, e.g. $f_3(x,x,y) = f_4(x,x,x,y)$. Note that height-1 identities involving operations in a clone can be expressed "within" the clone. For example, the identity $f_3(x,x,y) = f_4(x,x,x,y)$ is satisfied iff the statement $f_3(\pi^2_1,\pi^2_1,\pi^2_2) = f_4(\pi^2_1,\pi^2_1,\pi^2_1,\pi^2_2)$ is true in the clone.

▶ **Definition 36.** A mapping H from a clone $\mathbf{D}$ to a clone $\mathbf{E}$ is called an *h1 clone homomorphism* if

- it preserves the arities of operations,
- it preserves height-1 identities; that is,

  $$H(g(\pi^k_{i_1},\ldots,\pi^k_{i_n})) = H(g)(\pi^k_{i_1},\ldots,\pi^k_{i_n})$$

  where $g \in \mathbf{D}$ is n-ary.

Note that while an h1 clone homomorphism preserves height-1 identities, it is not required to preserve non-height-1 identities.

▶ **Example 37.** Assume that $\mathbf{D}$ contains a semilattice operation f and H is an h1 clone homomorphism from $\mathbf{D}$ to $\mathbf{E}$. Then the operation $H(f)$ on E is commutative, but not necessarily idempotent or associative. However, if $g = f(f(x,y),z)$ then $H(g)$ satisfies identities such as $H(g)(x,y,z) = H(g)(y,z,x) = H(g)(y,x,z)$ and $H(g)(x,x,y) = H(g)(x,y,y)$ because they are height-1 and g satisfies them.

The following is proved in [20].

▶ **Theorem 38.** *Let $\mathcal{D}$, $\mathcal{E}$ be constraint languages. Then $\mathcal{D}$ pp-constructs $\mathcal{E}$ if and only if there exists an h1 clone homomorphism from $\mathbf{D}$ to $\mathbf{E}$.*

Thus if $\mathcal{D} \leq \mathcal{E}$ in the pp-constructibility ordering (i.e. $\mathcal{D}$ pp-constructs $\mathcal{E}$), then $\mathbf{E}$ satisfies all properties of the form "there exist operations $f_1,\ldots,f_n$ satisfying height-1 identities $\Lambda_1,\ldots,\Lambda_k$" which are satisfied by $\mathbf{D}$.[7] A compactness argument shows that the converse is also true: if $\mathcal{D} \not\leq \mathcal{E}$, then there is a finite system of height-1 identities which is satisfied by some operations from $\mathbf{D}$ but is not satisfied by any operations in $\mathbf{E}$. Since the position of $\mathcal{D}$ in the pp-constructibility ordering determines the complexity of CSP($\mathcal{D}$) up to logspace reductions, we have that the following holds *unconditionally*:

[7] Algebraists call such properties *strong (height-1) Mal'tsev conditions.*

The complexity of CSP($\mathcal{D}$), up to logspace reductions, depends only on the finite systems of height-1 identities satisfied by the operations in the clone of polymorphisms of $\mathcal{D}$.

More is true. Suppose C is a set of constraint languages with the property that if $\mathcal{D} \in C$ and CSP($\mathcal{E}$) is logspace reducible to CSP($\mathcal{D}$), then $\mathcal{E} \in C$. (For example, C could be the class of $\mathcal{D}$ for which CSP($\mathcal{D}$) is in P.) Then C is an upward-closed subset of the pp-constructibility partial order. Assume that C is not the set of all constraint languages. It follows that

1. There exist $\mathcal{E}_1, \mathcal{E}_2, \ldots$ such that $C = \{\mathcal{D} : \mathcal{D} \not\leq \mathcal{E}_i \text{ for all } i\}$, and hence
2. C is the set of constraint languages $\mathcal{D}$ such that for every i there exists a finite system of height-1 identities which is not satisfied by $\mathbf{E}_i$ but is satisfied by $\mathbf{D}$.

In other words, C can be characterized by a set of "forbidden" constraint languages (with respect to pp-constructibility), and also by a (possibly infinite) disjunction of (possibly infinite) systems of height-1 identities on polymorphisms. One advantage of this perspective is that it replaces a negative characterization of a class of interest (forbidden constraint languages) with a positive one (existence of polymorphisms satisfying height-1 identities). For this observation to be useful, however, the height-1 identities must be manageable, and the discovery of manageable height-1 identities characterizing classes of interest is one of the achievements of the algebraic method. We will illustrate this in the following subsection.

4.4 Classifications and Conjectures

In this section we give examples in which classes of constraint languages are characterized both by forbidden constraint languages and by height-1 identities of polymorphisms.

For the sake of readability, our terminology will stray from longstanding traditions in the literature. Specifically, our definitions of *Taylor, weak NU, cyclic*, and *Siggers* operations do *not* require that the operation is idempotent. We will write *idempotent Taylor, idempotent weak NU*, etc. for what is ordinarily called *Taylor, weak NU,* etc.[8] (Note that we cannot bring ourselves to apply this convention to Mal'tsev, majority and NU operations.) It's easy to see that a language $\mathcal{D}$ has one of the polymorphisms mentioned in this section (e.g. Taylor) if and only if the core of $\mathcal{D}$ has an idempotent version of that polymorphism. Hence, even though we state all conjectures in this section without assuming idempotence, it is enough to prove them in the idempotent case.

▶ **Definition 39.** A *Taylor* operation is a k-ary ($k \geq 2$) operation f such that, for each $1 \leq i \leq k$, f satisfies an identity of the form

$$f(z_{i,1}, \ldots, z_{i,i-1}, \underset{i}{x}, z_{i,i+1}, \ldots, z_{i,k}) = f(z'_{i,1}, \ldots, z'_{i,i-1}, \underset{i}{y}, z'_{i,i+1}, \ldots, z'_{i,k}) \qquad (2)$$

where all $z_{i,j}, z'_{i,j}$ are in $\{x, y\}$.

A useful way to view this definition is that the identities (2) prevent f from being the ith projection (for each i) whenever the domain has more than one element. In fact, it is easy to see that Taylor identities are the weakest height-1 identities involving a single operation that prevent this operation from being a projection.

The following theorem can be derived from [116] (see also [34]).

[8] [50] uses *quasi-Taylor, quasi-WNU* etc. for the not-necessarily-idempotent versions.

▶ **Theorem 40.** *For any constraint language $\mathcal{D}$, the following are equivalent:*
1. *$\mathcal{D}$ does not pp-construct the language of* 3-SAT*.*
2. *$\mathcal{D}$ has a Taylor polymorphism of some arity.*

The class of constraint languages which do not pp-construct the language of 3-SAT is precisely the class of $\mathcal{D}$ for which the Tractability conjecture asserts that CSP($\mathcal{D}$) is in P. Hence constraint languages with Taylor polymorphisms are of particular interest. Although the defining condition of a Taylor operation looks rather cumbersome, it has been shown that, rather surprisingly, for any constraint language $\mathcal{D}$, the property of having a Taylor polymorphism is equivalent to a number of simpler conditions, as described below.

- A *weak near-unanimity (WNU)* operation is a k-ary ($k \geq 2$) operation f satisfying the identities
$$f(y, x, x, \ldots, x, x) = f(x, y, x, \ldots, x, x) = \ldots = f(x, x, x, \ldots, x, y); \tag{3}$$
- A *cyclic* operation is a k-ary ($k \geq 2$) operation f satisfying the identity
$$f(x_1, x_2, \ldots, x_k) = f(x_2, \ldots, x_k, x_1); \tag{4}$$
- A *Siggers* operation is a 4-ary operation f satisfying the identity [9]
$$f(y, x, y, z) = f(x, y, z, x). \tag{5}$$

▶ **Theorem 41.** *For every constraint language $\mathcal{D}$, the following are equivalent:*
1. *$\mathcal{D}$ has a Taylor polymorphism;*
2. *$\mathcal{D}$ has a WNU polymorphism* [104]*;*
3. *$\mathcal{D}$ has a cyclic polymorphism* [13]*;*
4. *$\mathcal{D}$ has a Siggers polymorphism* [85, 113]*.*

For other conditions equivalent to the presence of an (idempotent) Taylor polymorphism, see [18, 33, 34, 73, 94].

▶ **Corollary 42.** *If a constraint language $\mathcal{D}$ has no Taylor (equivalently, no WNU, cyclic, or Siggers) polymorphism then* CSP($\mathcal{D}$) *is NP-complete.*

Using the above results, the Tractability conjecture (Conjecture 22), combined with Corollary 42, can be re-stated as follows.

▶ **Conjecture 43** (Algebraic Dichotomy Conjecture)**.** *If a constraint language $\mathcal{D}$ has a Taylor (equivalently, a WNU, cyclic, or Siggers) polymorphism, then* CSP($\mathcal{D}$) *is solvable in polynomial time; otherwise, it is NP-complete.*

More informally, the (open part of the) Algebraic Dichotomy Conjecture says:

If $\mathcal{D}$ has a nontrivial higher-dimensional symmetry (hence providing a nontrivial way to combine solutions), then CSP($\mathcal{D}$) should be tractable.

We remark that Conjecture 43 is often misquoted as "CSP($\mathcal{D}$) is NP-complete if all polymorphisms of $\mathcal{D}$ are projections, and CSP($\mathcal{D}$) is tractable otherwise." This form of the conjecture is false, as the following example shows.

[9] Using different variables, $f(r, a, r, e) = f(a, r, e, a)$ – mnemonic due to Ryan O'Donnell.

▶ **Example 44.** Consider the map $f : \{0,1,2\} \to \{0,1\}$ such that $f(0) = 0$ and $f(1) = f(2) = 1$ and consider the constraint language $\mathcal{D} = \{f^{-1}(S_{ijk}) \mid S_{ijk} \in \mathcal{D}_{3\text{SAT}}\}$ on $\{0,1,2\}$. Note that any relation in $\mathcal{D}$ cannot distinguish elements 1 and 2, in the sense that any 1 anywhere in it can be replaced by 2 and vice versa. Hence, if we take any projection π_i^n on $\{0,1,2\}$ and any operation $f : \{1,2\}^n \to \{1,2\}$ and define f' on $\{0,1,2\}$ so that

$$f'(\mathbf{t}) = \begin{cases} f(\mathbf{t}), & \text{if } \mathbf{t} \in \{1,2\}^n \\ \pi_i^n(\mathbf{t}), & \text{otherwise} \end{cases}$$

then f' is a polymorphism of $\mathcal{D}$. It is clear that CSP($\mathcal{D}$) is NP-complete and has many polymorphisms other than projections. One can even take the singleton expansion of $\mathcal{D}$ – it would be a core, and each operation f' built above would remain a polymorphism provided f is idempotent. However, all height-1 identities satisfied by such operations must also be satisfied by projections.

▶ **Example 45.** We show how to apply cyclic operations to prove the dichotomy theorem for undirected graphs [71].

Let R be a symmetric binary relation viewed as an undirected graph and let $\mathcal{D} = \{R\}$. If R contains a loop then CSP($\mathcal{D}$) is trivially tractable. If R is bipartite, then the core of $\mathcal{D}$ is an edge and CSP($\mathcal{D}$) is essentially 2-COLORING, which is tractable.

Let $\mathcal{D}' = \{R', ...\}$ be the singleton expansion of the core of $\mathcal{D}$. If R is not bipartite and does not contain a loop, then R' does not contain a loop, but does contain a closed walk $a_1, a_2, \dots, a_p, a_1$ for some prime $p > |D'|$. It was shown in [13] that if $\mathbf{D}'$ contains some idempotent cyclic operation then it contains such an operation of every prime arity greater than $|D'|$, so we can assume that $\mathbf{D}'$ contains a cyclic operation t of arity p. Since t is a polymorphism, the pair

$$t((a_1,a_2),\dots,(a_{p-1},a_p),(a_p,a_1)) = (t(a_1,\dots,a_p), t(a_2,\dots,a_p,a_1))$$

is in R', but it is a loop since t is cyclic. This contradiction shows that $\mathbf{D}'$ does not contain an idempotent cyclic operation of arity p, and hence it does not have an idempotent cyclic operation of any arity. Because $\mathcal{D}'$ is idempotent, $\mathbf{D}'$ has no cyclic operation. $\mathcal{D}$ pp-constructs $\mathcal{D}'$; hence $\mathcal{D}$ has no cyclic operation (as the height-1 identity characterizing a cyclic operation of $\mathbf{D}$ would also be satisfied by some operation of $\mathbf{D}'$). Hence $\mathcal{D}$ does not have a cyclic polymorphism, so CSP($\mathcal{D}$) is NP-complete.

▶ **Example 46.** A digraph is called *smooth* if it contains neither sources nor sinks, i.e. all its vertices have positive in- and out-degrees. It was shown in [17] that if a smooth digraph has a WNU polymorphism then its core must be a disjoint union of directed cycles. If a smooth digraph H has such a form, it is an easy exercise to show that the corresponding H-COLORING problem CSP($\mathcal{D}_H$) is solvable in polynomial time. If H does not have such a form, then CSP($\mathcal{D}_H$) is NP-complete by Corollary 42.

Consider the digraph H having only 3 vertices x, y, z and 4 edges yx, xy, yz, zx. It is a smooth core digraph, so it follows from the above that the corresponding problem CSP($\mathcal{D}_H$) is NP-complete. It can be easily checked that it is the smallest digraph with this property. Moreover, this digraph was used in [85] to obtain identity (5) above, which can be viewed as follows: if a digraph with a Siggers polymorphism contains H as a subdigraph, then it also contains a loop.

Next, we consider the class of constraint languages $\mathcal{D}$ for which CSP($\mathcal{D}$) has *bounded width* (meaning that all unsatisfiable instances of CSP($\mathcal{D}$) can be refuted via local propagation).

The "obvious" obstructions to bounded width, besides 3-SAT, are 3-LIN(p) for primes p. Let C be the set of constraint languages which do not pp-construct the language of 3-LIN(p) for any prime p. A notable success of the algebraic method was the proof that C is precisely the set of constraint relations having bounded width [14, 31]. The original proof of Barto and Kozik [12] hinged on the equivalence of the first and second items in the following theorem.

▶ **Theorem 47.** *For any constraint language $\mathcal{D}$, the following are equivalent:*
1. *$\mathcal{D}$ does not pp-construct the language of* 3-LIN(p)*, for any p;*
2. *$\mathcal{D}$ has WNU polymorphisms of all but finitely many arities* [99, 104]*;*
3. *$\mathcal{D}$ has a k-ary WNU polymorphism for each $k \geq 3$ (see* [91]*);*
4. *$\mathcal{D}$ has a ternary WNU polymorphism f_3 and a 4-ary WNU polymorphism f_4 such that $f_3(x, x, y) = f_4(x, x, x, y)$* [91]*;*
5. *for some $k \geq 3$, $\mathcal{D}$ has a k-ary polymorphism f satisfying, for each $1 \leq i \leq k$, an identity of the form*

$$f(z_{i,1}, \ldots, z_{i,i-1}, \underset{i}{x}, z_{i,i+1}, \ldots, z_{i,k}) = f(z_{i,1}, \ldots, z_{i,i-1}, \underset{i}{y}, z'_{i,i+1}, \ldots, z'_{i,k}) \tag{6}$$

where all $z_{i,j}, z'_{i,j}$ are in $\{x, y\}$ [91].

For other conditions equivalent to those from Theorem 47, see [18, 73, 82, 94].

We will discuss CSPs of bounded width in detail in Section 5.

As a third example, consider the class of constraint languages $\mathcal{D}$ for which CSP($\mathcal{D}$) has *bounded linear width*, meaning that all unsatisfiable instances of CSP($\mathcal{D}$) can be refuted via local propagation in a linear fashion (think refuting unsatisfiable 2-SAT instances by following paths). This refutation property can be formalised via Linear Datalog, among other equivalent ways, see [44]. Another way to express this property is that every instance CSP($\mathcal{D}$) can solved by forming, in logspace, a certain directed graph (of local inferences) and solving STCON on this digraph, see [44, 62]. This shows that problems CSP($\mathcal{D}$) of bounded linear width are in complexity class NL.

The obvious obstructions to bounded linear width are 3-LIN(p) for primes p and HORN-3-SAT. Let C' be the class of constraint languages which do not pp-construct the language of 3-LIN(p) for any prime p, nor the language of HORN-3-SAT. Thus C' contains the class of constraint languages whose CSP has bounded linear width (and possibly more). The following theorem gives a manageable algebraic characterization of C'.

▶ **Theorem 48.** *For any constraint language $\mathcal{D}$, the following are equivalent:*
1. *$\mathcal{D}$ pp-constructs neither the language of* 3-LIN(p)*, for any p, nor that of* HORN-3-SAT*;*
2. *for some $n \geq 2$, $\mathcal{D}$ has ternary polymorphisms $d_0, \ldots, d_n$ satisfying the following identities* [73]*:*

$$d_0(x, y, z) = d_0(x, x, x), \tag{7}$$

$$d_n(x, y, z) = d_n(z, z, z), \tag{8}$$

$$d_i(x, y, y) = d_{i+1}(x, y, y) \ \ \textit{and} \ \ d_i(x, y, x) = d_{i+1}(x, y, x) \ \textit{if } i \textit{ is even, } i < n, \tag{9}$$

$$d_i(x, x, y) = d_{i+1}(x, x, y) \ \textit{if } i \textit{ is odd, } i < n. \tag{10}$$

3. *for some $k \geq 3$, $\mathcal{D}$ has a k-ary polymorphism f satisfying, for each $1 \leq i \leq k$, an identity of the form*

$$f(x, \ldots, x, \underset{i}{x}, z_{i,i+1}, \ldots, z_{i,k}) = f(x, \ldots, x, \underset{i}{y}, z'_{i,i+1}, \ldots, z'_{i,k}) \tag{11}$$

where all $z_{i,j}, z'_{i,j}$ are in $\{x, y\}$ [66].

It is known [91] that the above theorem cannot have an equivalent condition involving only a bounded number of functions of bounded arity (such as Siggers polymorphism or condition (4) from Theorem 47), so at least one of the number and the arity of operations involved in any characterization must be unbounded.

This theorem has an interesting complexity-theoretic consequence. HORN-3-SAT is P-complete (and thus unlikely to be in NL). The relationship of problems 3-LIN(p), and hence of classes Mod$_p$L, with the class NL is unknown (though there is evidence that NL is contained in Mod$_p$L for every p [2, 110]). As mentioned above, bounded linear width guarantees membership in NL [52], and moreover, all problems CSP($\mathcal{D}$) known to be in NL have bounded linear width.

▶ **Corollary 49.** *Let $\mathcal{D}$ be a constraint language having polymorphisms satisfying one of the equivalent conditions in Theorem 47. If $\mathcal{D}$ has no ternary polymorphisms satisfying conditions* (7)–(10)*, or equivalently, no polymorphism satisfying condition* (11)*, of Theorem 48, then* CSP($\mathcal{D}$) *is P-complete and cannot have bounded linear width.*

As mentioned earlier, the class of constraint languages having polymorphisms witnessing the equivalent conditions of Theorem 48 contains the class of constraint languages whose CSP has bounded linear width. It was suggested in [97] that the two classes actually coincide.

▶ **Conjecture 50** (Bounded Linear Width Conjecture)**.** *If a constraint language $\mathcal{D}$ has polymorphisms as described in one of the equivalent conditions in Theorem 48 then* CSP($\mathcal{D}$) *has bounded linear width and hence belongs to NL.*

We discuss progress towards resolving Conjecture 50 in Section 5.7.

If for some reason you are interested in the class of constraint languages which do not pp-construct the language of HORN-3-SAT, but with no restriction on pp-constructing the language of any 3-LIN(p), then you are in luck; algebraic descriptions of this class are also known. For example:

▶ **Theorem 51** ([73])**.** *For any constraint language $\mathcal{D}$, the following are equivalent:*

- *$\mathcal{D}$ does not pp-construct the language of* HORN-3-SAT*.*
- *for some $k \geq 3$, $\mathcal{D}$ has a k-ary polymorphism f satisfying, for each $1 \leq i \leq k$, an identity of the form*

$$f(x, \dots, x, \underset{i}{x}, z_{i,i+1}, \dots, z_{i,k}) = f(z'_{i,1}, \dots, z'_{i,i-1}, \underset{i}{y}, z'_{i,i+1}, \dots, z'_{i,k}) \tag{12}$$

where all $z_{i,j}, z'_{i,j}$ are in $\{x, y\}$.

Example 12 shows that the structure $\mathcal{D}_{\text{HornSAT}}$ from Example 5 pp-defines the structure $\mathcal{D}_{\text{STCON}}$ from Example 9. Thus, by forbidding $\mathcal{D}_{\text{STCON}}$ instead of $\mathcal{D}_{\text{HornSAT}}$ in Theorem 51, we further restrict the class of constraint languages, but again obtain a class with a known algebraic characterization.

▶ **Theorem 52** ([68])**.** *For any constraint language $\mathcal{D}$, the following are equivalent:*

- *$\mathcal{D}$ does not pp-construct $\mathcal{D}_{STCON}$;*
- *for some $t \geq 2$, $\mathcal{D}$ has ternary polymorphisms $p_0, \dots, p_t$ satisfying the following identities:*

$$p_0(x, y, z) = p_0(x, x, x), \tag{13}$$

$$p_t(x, y, z) = p_t(z, z, z), \tag{14}$$

$$p_i(x, x, y) = p_{i+1}(x, y, y) \quad \textit{for all } i < t. \tag{15}$$

Here is one final example characterizing forbidden languages by height-1 identities.

▶ **Theorem 53** ([73]). *For any constraint language $\mathcal{D}$, the following are equivalent:*

- *$\mathcal{D}$ pp-constructs neither the language of* 3-LIN(p)*, for any p, nor that of* STCON*;*
- *for some $n \geq 0$, $\mathcal{D}$ has 4-ary polymorphisms $f_0, \ldots, f_n$ satisfying the following identities:*

$$f_0(x,y,y,z) = f_0(x,x,x,x) \tag{16}$$

$$f_n(x,x,y,z) = f_n(z,z,z,z) \tag{17}$$

$$f_i(x,x,y,x) = f_{i+1}(x,x,y,x) \quad \textit{and} \quad f_i(x,x,y,y) = f_{i+1}(x,x,y,y), \quad \textit{for } i < n. \tag{18}$$

Some problems CSP($\mathcal{D}$) can be solved by forming a certain <u>un</u>directed graph (of local inferences) on a given instance and then solving STCON on this graph, see [62]. This property is called bounded symmetric width, and the corresponding CSPs belong to the complexity class L.

It was shown in [97] that any language not satisfying the conditions of Theorem 53 cannot have bounded symmetric width and that the corresponding CSP is hard for at least one of complexity classes NL and Mod$_p$L (for some p).

▶ **Conjecture 54** (Bounded Symmetric Width Conjecture, [97]). *If a constraint language $\mathcal{D}$ has polymorphisms as described in one of the equivalent conditions in Theorem 53 then* CSP($\mathcal{D}$) *has bounded symmetric width and hence belongs to L.*

We discuss progress towards resolving Conjecture 54 in Section 5.7.

It is interesting to note that the complexity classifications obtained or conjectured through these algebraic results are obviously conditional on complexity-theoretic assumptions, while the classifications related to (various notions of) width are unconditional.

4.5 Taxonomy of Systems of Linear Identities

Although the reader might find the identities in Theorems 40, 41, 47, 48, 51, 52 and 53 to be somewhat random, in fact it is something of a minor miracle that the classes of constraint languages considered in those theorems have manageable algebraic descriptions. The proofs of these theorems are highly nontrivial and use sophisticated tools from universal algebra. What made their discovery possible is the serendipitous fact that universal algebraists have been studying the connection between identities and "good algebraic structure" for almost 50 years.

In particular, the systems of identities appearing in Theorems 40, 41, 51, 52 and 53, as well as some equivalent characterizations of the classes described in Theorems 47 and 48, when strengthened by adding idempotency, have been known to be of fundamental importance since the 1980s in the context of "tame congruence theory" [73]. Height-1 identities and idempotency identities $f(x,x,\ldots,x) = x$ are both examples of *linear* identities; these are identities in which each side has *at most* one occurrence of an operation symbol.

In addition to these well-studied systems of linear identities arising in tame congruence theory, there is a robust taxonomy of "classical" systems of linear identities, all stronger than the Taylor identities. This allows one to approach classification problems, and especially their positive parts, for language-based CSP (such as Conjecture 43) as follows. First prove the result for stronger systems of identities, and then move to weaker identities, gradually approaching the identities which determine the (conjectured) boundary.

We now introduce some (more) of the more important linear identities that have played a role in the algebraic theory of CSP. Most of these identities have been studied in universal algebra before the link with the CSP was discovered.

- A *near-unanimity (NU)* operation is an n-ary ($n \geq 3$) operation f which satisfies

$$f(y, x, x, \dots, x, x) = f(x, y, x, \dots, x, x) = \dots = f(x, x, x, \dots, x, y) = x; \tag{19}$$

  The last equality in (19) is the difference between NU and WNU operations. A ternary NU operation is usually called a *majority* operation. The dual discriminator operation from Example 30 is a majority operation which was often used as a starting point in many CSP-related classifications.
  For example, the operation $f(x_1, \dots, x_n) = \bigvee_{i<j}(x_i \wedge x_j)$ on $\{0, 1\}$ is an NU operation.
- A *symmetric* operation is an n-ary operation satisfying all identities of the form

$$f(x_1, x_2, \dots, x_n) = f(x_{\pi(1)}, x_{\pi(2)}, \dots x_{\pi(n)}), \tag{20}$$

  where π is a permutation of the set $\{1, 2, \dots, n\}$.
- A *totally symmetric (TS)* operation is an n-ary operation satisfying all identities of the form

$$f(x_1, x_2, \dots, x_n) = f(y_1, y_2, \dots y_n), \tag{21}$$

  where $\{y_1, y_2, \dots, y_n\} = \{x_1, x_2, \dots, x_n\}$. This includes all identities of the form (20) as well as identities such as $f(x, x, y) = f(x, y, y)$.
  If $f_2(x_1, x_2)$ is a semilattice operation (Example 35) then, for all $n \geq 2$, $f_n(x_1, \dots, x_n) = f_2(x_1, f_2(x_2, (\dots f_2(x_{n-1}, x_n))))$ is a TS operation.
- A *Mal'tsev* [10] operation is a ternary operation f satisfying the identities

$$f(x, x, y) = f(y, x, x) = y. \tag{22}$$

  If in addition it satisfies $f(x, y, x) = y$ then it is called a *minority* operation.
  A typical Mal'tsev operation is $f(x, y, z) = x - y + z$ where $(D, +)$ is an Abelian group (see Example 29).
- Ternary operations $d_0, d_1, \dots, d_n$ are *Jónsson* operations if they satisfy the identities of Theorem 48, as well as the identities

$$d_i(x, y, x) = x \quad \text{for all } i \leq n. \tag{23}$$

- Ternary operations $d_0, d_1, \dots, d_n, p$ are *Gumm* operations if $d_0, \dots, d_n$ satisfy all the identities of Jónsson operations *except* equation (8) from Theorem 48, as well as the identities

$$d_n(x, y, y) = p(x, y, y) \quad \text{and} \quad p(x, x, y) = y. \tag{24}$$

- For $k \geq 2$, an k-*edge* operation is a $(k+1)$-ary operation f satisfying identities

$$\begin{aligned}
f(x, x, y, y, y, \dots, y, y) &= y \\
f(x, y, x, y, y, \dots, y, y) &= y \\
f(y, y, y, x, y, \dots, y, y) &= y \\
f(y, y, y, y, x, \dots, y, y) &= y \\
&\vdots \\
f(y, y, y, y, y, \dots, y, x) &= y
\end{aligned} \tag{25}$$

[10] Sometimes spelled as Maltsev/Mal'cev/Malcev – correctly pronounced with the soft 'l' followed by 'ts'.

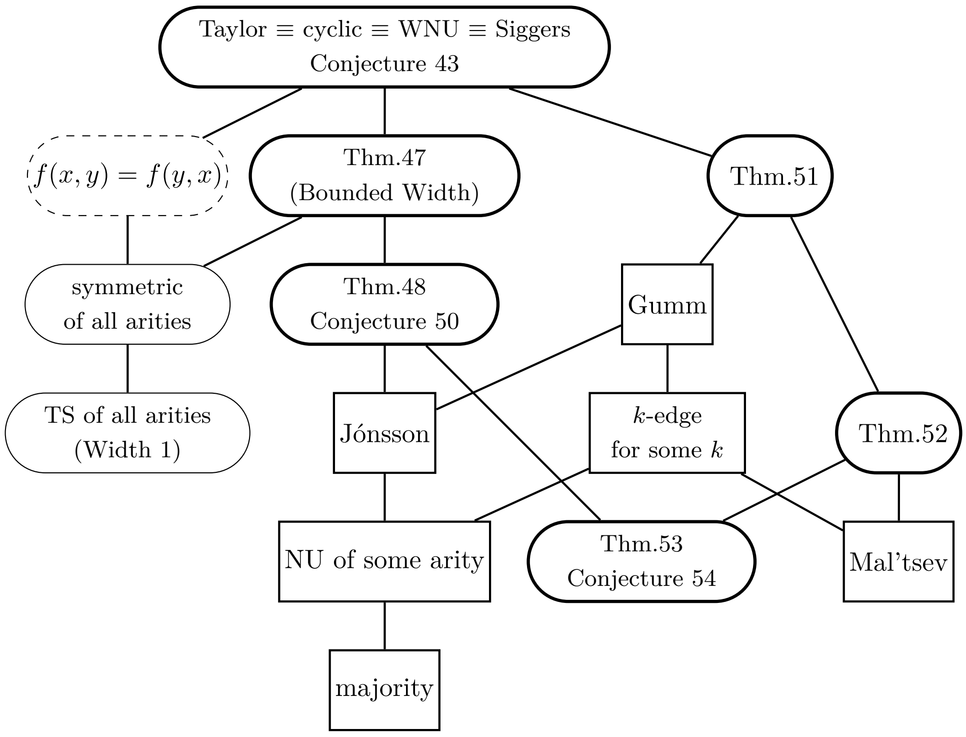


Figure 1 Taxonomy of important systems of linear identities. Upward paths in the diagram correspond to weakening conditions, i.e. increasing corresponding classes of constraint languages.

An easy way to parse these identities is this: these are Mal'tsev identities glued with the NU identities as follows. If f depends on the first three variables only, then these identities are the same as (22), but with the first two coordinates swapped (in particular, 2-edge is essentially permuted Mal'tsev). If f depends on all but the first two variables, then these identities are the same as (19), but with x and y swapped.

Figure 1 shows the relative strength of the mentioned linear identities on finite domains. The higher items correspond to weaker conditions, and hence to larger classes of constraint languages. More precisely, if a constraint language has polymorphisms witnessing one set of identities in the diagram, then these operations can be composed with themselves and projections to obtain operations satisfying the identities "higher" in the diagram. Such a composition is possible in any function clone on any (finite or infinite) domain. Each edge shown in the diagram is strict in the foregoing sense; however, Barto has shown that two of the edges collapse in the following weaker sense: if a constraint language (on a finite domain, with finitely many relations) has Jónsson polymorphisms then it must also have an NU polymorphism [8]. A similar statement holds for Gumm and edge polymorphisms [9].

By the convention we have followed in this paper, the systems of identities enclosed in rectangles are idempotent by definition, while the systems of identities circled in ovals are height-1 and hence are not assumed to be idempotent. The systems circled in thick ovals, when restricted to the idempotent case, correspond to robust tame congruence theoretic conditions.

We note that [76] contains a useful list (and a diagram) of many universal-algebraic conditions relevant for the CSP, though their list is oriented more (than ours) towards prominence of conditions in universal algebra.

Not all natural-looking linear identities make a good choice for attacks on the open conjectures. For example, commutativity (in the dashed oval in Figure 1) is one of the simplest of the Taylor-type identities, but approaching Conjecture 43 by looking at commutative (or commutative idempotent) polymorphisms is not (known to be) a great idea. On the other hand, finding a CSP algorithm for constraint languages satisfying the identities of Theorem 51 would likely be viewed as a more "natural" (and more feasible) step.

The problems of deciding whether a given constraint language has a given type of polymorphism(s) are sometimes called "meta-problems." See [50] for a survey and new results on the complexity of meta-problems.

5 Polymorphisms in Algorithms I: Proving Correctness

The most natural idea to decide an instance of the CSP is to first derive some "obvious" consequences of the constraints. If, for example, an instance of HORN-3-SAT (see Example 5) contains the constraints $C_1(x)$, $C_1(y)$, and $S_{110}(x,y,z)$, then we can derive a new constraint $C_1(z)$. This information can be then used to derive, e.g., $C_1(w)$ from $S_{110}(z,x,w)$, etc. In fact, for HORN-3-SAT, if we cannot derive any new unary constraints in this way and we do not get contradictory unary constraints like $C_0(x)$ and $C_1(x)$, then the instance has a solution: simply obey the unary constraints and set the values of the remaining variables to 0. This is essentially the standard unit propagation (polynomial) algorithm for HORN-3-SAT.

More generally, given an instance of CSP($\mathcal{D}$) we can try to derive the strongest "obvious" unary constraints.[11] For some constraint languages all unsatisfiable instances can be refuted in this way. Such languages are said to have *width 1* and they are discussed in Subsection 5.1.

We illustrate a stronger constraint propagation (or local consistency) procedure by considering the following instance of CSP($\mathcal{D}_{2\text{COLOR}}$) (see Example 6).

$$x_1 \neq x_2,\ x_2 \neq x_3,\ x_3 \neq x_4,\ x_4 \neq x_5,\ x_5 \neq x_1 \ .$$

From $x_1 \neq x_2$ and $x_2 \neq x_3$ we can derive a new binary constraint $x_1 = x_3$. Using this and $x_3 \neq x_4$, we can derive $x_1 \neq x_4$. Finally, from $x_1 \neq x_4$, $x_4 \neq x_5$, we conclude that $x_1 = x_5$, which contradicts the constraint $x_1 \neq x_5$, and we refute the instance. In this particular example, it was enough to derive new binary constraints by considering at most three variables at a time. In fact, every unsatisfiable instance of 2-COLORING can be refuted in this way; we say that $\mathcal{D}_{2\text{COLOR}}$ has *width (2,3)*. In Subsection 5.3, we give a more general result that will also apply to e.g. 2-SAT.

An even stronger consistency algorithm could derive all k-ary constraints that can be derived from the instance by considering at most l variables at a time. As long as k and l are fixed, we get a polynomial time algorithm that, for some constraint languages, correctly decides every instance of the CSP. The limit of such approaches is now fully understood, see Subsection 5.5.

We warn the reader that there are many related, but somewhat different notions capturing the required level of local consistency. In order to add to the confusion, we use the term "width (k,l)" for what is more often called "relational width (k,l)". However, the notion of "solvability by a local consistency algorithm" (meaning "having width (k,l) for some k,l") is the same for all common choices of definitions.

[11] The relations of the derived constraints do not need to belong to $\mathcal{D}$.

5.1 1–Minimality and TS Polymorphisms

The idea of the strongest obvious unary constraints can be formalized as follows.

▶ **Definition 55.** An instance of the CSP is 1*–minimal* (or *arc consistent*) if it contains a unique unary constraint $P_x(x)$ for each variable x and, for any constraint $R(x_1, \dots, x_k)$ and any $i = 1, \dots k$, the projection[12] of R onto the i-th coordinate is equal to P_{x_i}.

Every instance of the CSP can be converted in polynomial time to a 1-minimal instance with the same set of solutions. A straightforward (although not optimal) way to achieve this is as follows.

```
for every variable $x$ do $P_x := D$; add the constraint $D(x)$
repeat
    for every constraint $R(x_1, \dots, x_k)$ do
        let $R' := R \cap \prod_i P_{x_i}$
        for $i = 1$ to $k$ do $P_{x_i} := P_{x_i} \cap \mathrm{proj}_i R'$
        replace $R(x_1, \dots, x_k)$ with $R'(x_1, \dots, x_k)$
    end for
until none of the $P_x$'s changed
```

We allow ourselves to call this algorithm *the* 1*-minimality algorithm*, since different implementations will derive the same unary constraints.

If any of the P_x's are empty after running the algorithm, then the original instance has no solution and we say that the 1-minimality algorithm refutes the instance. For some constraint languages, every non-refuted instance has a solution and thus we obtain a polynomial time algorithm for the corresponding CSP.

▶ **Definition 56.** We say that a constraint language $\mathcal{D}$, or CSP($\mathcal{D}$), has *width* 1 if the 1-minimality algorithm refutes every unsatisfiable instance of CSP($\mathcal{D}$) (and thus CSP($\mathcal{D}$) is solvable in polynomial time).

The following theorem [65, 60] characterizes width 1 languages in terms of polymorphisms. The proof shows a simple application of polymorphisms, namely TS polymorphisms, to prove correctness of an algorithm. Recall that the value of a TS operation t depends only on the set of its arguments, and we will write $t(\{d_1, \dots, d_n\})$ instead of $t(d_1, \dots, d_n)$.

▶ **Theorem 57.** *A constraint language $\mathcal{D}$ admits TS polymorphisms of all arities if and only if $\mathcal{D}$ has width* 1.

Sketch of Proof. Assume that $\mathcal{D}$ admits TS polymorphisms of all arities, take an instance of CSP($\mathcal{D}$), run the 1-minimality algorithm, and assume that the instance is not refuted, that is, P_x is nonempty for every variable x. We need to show that the resulting instance has a solution.

Note that the constraint relations of the resulting 1-minimal instance may not belong to $\mathcal{D}$. However, each new relation is defined in a primitive positive way from the original constraints; therefore all polymorphisms of $\mathcal{D}$ are still compatible with the relations in the new instance. Take a TS polymorphism t of a sufficiently large arity n and define $f(x) := t(d_1, \dots, d_n)$ where $P_x = \{d_1, \dots, d_n\}$.

[12] Not to be confused with projection operations π_i^n.

We claim that f is a solution. So, let us consider a constraint $R(x_1, \dots, x_k)$ and show that $(f(x_1), \dots, f(x_k)) \in R$. Take an $n \times k$ matrix so that its rows are in R and the set of elements in the i-th column is equal to P_{x_i}. This is easily achieved from 1-minimality as soon as $n \geq |D| \cdot k$. Now the identities that define a TS operation guarantee that t applied to the columns gives $(f(x_1), \dots, f(x_k))$ and this tuple is in R, as t is compatible with R.

For the other implication, it is possible to create an instance of CSP($\mathcal{D}$) which is not refuted by 1-minimality and which essentially says that $\mathcal{D}$ has TS polymorphisms of all arities. We refer the reader to [65, 60] for details. ◀

Note that we do not need to (explicitly) know any TS polymorphism to run the 1-minimality algorithm, but these polymorphisms guarantee that the algorithm correctly decides all instances.

5.2 Linear Programming and Symmetric Polymorphisms

Another line of research discovered a connection between consistency notions and certain convex programming relaxations of the CSP, namely the *canonical Linear Programming (LP) relaxation* and the *canonical Semidefinite Programming (SDP) relaxation*. We now outline the LP relaxation here and describe the SDP relaxation in Section 5.6.

For simplicity, we will restrict to instances that contain variables $x_1, \dots, x_n$ and exactly one binary constraint $P_{i,j}(x_i, x_j)$ for every $1 \leq i < j \leq n$ (and no other constraints). In particular, there are $m = n(n-1)/2$ constraints. The task to find an assignment satisfying the maximum number of constraints can be phrased as a 0–1 integer program as follows. The variables are $\lambda_{i,a} \in \{0,1\}$ for each $1 \leq i \leq n$, $a \in D$ (where D is the domain) and $\sigma_{i,j,a,b} \in \{0,1\}$ for each $1 \leq i < j \leq n$, $a, b \in D$. The intended meaning is that $\lambda_{i,a} = 1$ iff x_i is assigned the value a and $\sigma_{i,j,a,b} = 1$ iff x_i is assigned a and x_j is assigned b. The task is to maximize

$$\mathrm{Opt} = \frac{1}{m} \sum_{1 \leq i < j \leq n} \sum_{(a,b) \in P_{i,j}} \sigma_{i,j,a,b}$$

subject to the constraints

$$\sum_{a \in D} \lambda_{i,a} = 1 \quad \text{for each } 1 \leq i \leq n$$

$$\sum_{b \in D} \sigma_{i,j,a,b} = \lambda_{i,a} \quad \text{for each } 1 \leq i < j \leq n,\ a \in D$$

$$\sum_{a \in D} \sigma_{i,j,a,b} = \lambda_{j,b} \quad \text{for each } 1 \leq i < j \leq n,\ b \in D$$

Since we are only concerned with deciding whether a solution exists, we only care whether all the constraints are satisfied, that is, whether $\mathrm{Opt} = 1$.

The *canonical (or basic) LP relaxation* relaxes the 0–1 constraints to $\lambda_{i,a} \in [0,1]$ and $\sigma_{i,j,a,b} \in [0,1]$. The optimization problem thus becomes solvable in polynomial time, but the new optimum $\mathrm{Opt}_{\mathrm{LP}}$ may be greater than Opt. We say that this relaxation decides the CSP if an instance has a solution whenever $\mathrm{Opt}_{\mathrm{LP}} = 1$. The canonical LP relaxation is at least as strong as 1-minimality. Indeed, it is not hard to see that if $\mathrm{Opt}_{\mathrm{LP}} = 1$, then $P_x = \{a \in D : \lambda_{i,a} > 0\}$, together with appropriately restricted binary constraints, is 1-minimal. In fact, the relaxation is somewhat stronger [93]:

▶ **Theorem 58.** CSP($\mathcal{D}$) *is decided by the canonical LP relaxation if and only if* $\mathcal{D}$ *has symmetric polymorphisms of all arities.*

Sketch of Proof. Assume that $\mathcal{D}$ admits symmetric polymorphisms of all arities. Take an instance of CSP($\mathcal{D}$) and solve its canonical LP relaxation. If $\mathrm{Opt}_{\mathrm{LP}} < 1$ then clearly the instance is not satisfiable. Assume that $\mathrm{Opt}_{\mathrm{LP}} = 1$ and show that in this case the instance is satisfiable. In fact, we show that a symmetric polymorphism of appropriate arity can be used to round an optimal LP solution to a satisfying assignment.

Take an optimal LP solution and denote the values taken by variables by $\sigma^*_{i,j,a,b}$ and $\lambda^*_{i,a}$. We can assume that all these values are rational. Let N be an integer such that $N \cdot \sigma^*_{i,j,a,b}$ is integer for all i, j, a, b (and hence, from LP constraints, all numbers $N \cdot \lambda^*_{i,a}$ are also integers). Let s be a symmetric polymorphism of arity N. For each variable x_i in the original CSP instance, let $f(x_i) = s(d_1, \ldots, d_N)$ where each value a appears among the d_i's exactly $N \cdot \lambda^*_{i,a}$ times. This definition is correct because $\sum_{a \in D} \lambda^*_{i,a} = 1$ (from LP constraints) and because s is symmetric (so the order of the d_i's is irrelevant). We will show that f is a solution.

Note that, since $\mathrm{Opt}_{\mathrm{LP}} = 1$, we have that, for each i, j, $\sum_{(a,b) \in P_{i,j}} \sigma_{i,j,a,b} = 1$. In particular, if $\sigma_{i,j,a,b} > 0$ for some a, b then $(a, b) \in P_{ij}$. Fix indices i, j. Take any $N \times 2$ matrix such that each pair $(a, b) \in P_{ij}$ appears as a row in the matrix exactly $N \cdot \sigma^*_{i,j,a,b}$ times. The LP constraints and the fact that s is symmetric directly imply that, when applying s to the columns of this matrix, one gets $(f(x_i), f(x_j))$.

For the other implication, if $\mathcal{D}$ has no symmetric polymorphism of some arity m, then it is possible to construct an unsatisfiable instance of CSP($\mathcal{D}$) with $\mathrm{Opt}_{\mathrm{LP}} = 1$. We refer the reader to [93] for details. ◀

It was claimed in [93] that any structure $\mathcal{D}$ has symmetric polymorphisms of all arities if and only if it has TS polymorphisms of all arities, but this claim turned out to be false – see Example 99 in [94] for a counter-example.

5.3 (2,3)-Minimality and Majority Polymorphisms

Next we introduce a stronger consistency notion than 1-minimality. For clarity, we define this concept only for *binary instances*, that is, instances containing only unary and binary constraints. A general definition will be given in Subsection 5.5.

▶ **Definition 59.** A binary instance of the CSP is $(2, 3)$*–minimal* if it contains a unique unary constraint $P_x(x)$ for each variable x and a unique binary constraint $P_{x,y}(x, y)$ for any pair of distinct variables x, y such that

- for every pairwise distinct variables x, y, z and every $(a, b) \in P_{x,y}$, there exists $c \in P_z$ such that $(a, c) \in P_{x,z}$ and $(b, c) \in P_{y,z}$;
- for any pairwise distinct variables x, y, $P_{x,y} = P^{-1}_{y,x}$ and the projection of $P_{x,y}$ onto the first (second, respectively) coordinate is equal to P_x (P_y, resp.).

It is helpful to visualize a binary $(2, 3)$-minimal instance as a multipartite graph: it has one partite set for each variable x, whose set of vertices is equal to (a disjoint copy of) P_x, and vertices $a \in P_x$, $b \in P_y$ are adjacent if $(a, b) \in P_{x,y}$. The first condition in Definition 59 then means that by restricting to any three partite sets we obtain a graph in which every edge belongs to a triangle. Note that a solution of the instance corresponds to a clique that contains exactly one vertex from each partite set.

Similarly to 1-minimality, every (binary) instance of the CSP can be converted in polynomial time to a $(2, 3)$-minimal instance with the same set of solutions. We say that the original instance is refuted by this algorithm if some P_x is empty.

▶ **Definition 60.** We say that a constraint language $\mathcal{D}$ has *width* $(2, 3)$ if the $(2, 3)$-minimality algorithm refutes every unsatisfiable instance of CSP($\mathcal{D}$).

The following theorem [65, 79] says that each constraint language with a majority polymorphism, such as any set of unary and binary relations on $\{0,1\}$ (see Example 26), has width $(2,3)$. In particular, we get that 2-COLORING and 2-SAT are solvable in polynomial time.

▶ **Theorem 61.** *If a constraint language $\mathcal{D}$ has a majority polymorphism, then $\mathcal{D}$ has width* $(2,3)$.

Sketch of Proof. We will only consider binary instances.

Take a majority polymorphism m of $\mathcal{D}$ and consider the $(2,3)$-minimal instance associated to a given instance of CSP($\mathcal{D}$). As in the proof of Theorem 57, it can be argued that m is compatible with P_x and $P_{x,y}$ for any x, y. We show that the instance has a solution provided every P_x is nonempty.

Recall from the remarks after Definition 59 that "every edge extends to a triangle". We will show that "every triangle extends to a 4–clique". Take any variables x, y, z, w and any $a \in P_x$, $b \in P_y$, $c \in P_z$ that form a triangle, that is, $(a,b) \in P_{x,y}$, $(a,c) \in P_{x,z}$, and $(b,c) \in P_{y,z}$. Using the edge–to–triangle property three times we get $d_1, d_2, d_3 \in P_w$ such that bcd_1, acd_2, and abd_3 are triangles. We claim that $d = m(d_1,d_2,d_3)$ together with a, b, c form a 4–clique. By the second property from Definition 59, there exists $a' \in P_x$ such that $(a', d_1) \in P_{x,w}$. Applying m to the columns of the 3×2 matrix with rows (a', d_1), (a, d_2), (a, d_3) (all in $P_{x,w}$) gives $(m(a', a, a), m(d_1, d_2, d_3)) \in P_{x,w}$, which is equal to (a, d) – here we use one of the identities defining a majority operation. Similarly, $(b, d) \in P_{y,w}$ and $(c, d) \in P_{z,w}$, which proves the claim.

In a similar way, we can show that every 4–clique extends to a 5–clique, and so on. Summarizing, every edge extends to a $|V|$–clique – a solution to the instance. ◀

The proof shows that for a constraint language $\mathcal{D}$ with a majority polymorphism, a greedy algorithm can be used to find a solution to any instance of CSP($\mathcal{D}$): after running the $(2,3)$–minimality algorithm, we pick, one by one, assignments to variables so that they stay consistent with the constraints. Such languages are said to have *strict width* $(2,3)$ and they are in fact characterized by the existence of a majority polymorphism [65].

The arguments can be easily generalized to NU polymorphisms if an appropriate consistency level is enforced. Namely, a constraint language has an NU polymorphism of arity n if and only if it has strict width $(n-1, n)$ [65].

5.4 Interlude: Boolean CSPs

Before we move on to discuss the general concept of bounded width, we show how to use polymorphisms to prove Schaefer's dichotomy theorem [112] for CSPs over a two–element domain. The only additional fact we need is the following lemma.

▶ **Lemma 62.** *Every idempotent (recall Example 27) clone on $D = \{0,1\}$ that contains a non–projection contains one of the following operations: the binary* max*, the binary* min*, the ternary majority, or the ternary minority.*

Sketch of Proof. An old result by Post [108] completely describes all clones on $\{0,1\}$ and we can thus simply use his classification. However, proving this lemma is significantly easier than the full classification and we sketch one possible approach.

The only binary idempotent operations are the two projections, max, and min. Therefore, let us assume that the only binary operations in our idempotent clone are projections. Then, for each ternary operation f, the operations $h(x,y) = f(x,x,y)$, $h'(x,y) = f(x,y,x)$, and

$h''(x,y) = f(y,x,x)$ must all be projections. This reduces the number of idempotent ternary operations to be considered to 8. Now 3 of them are projections, then there are the majority and the minority. The remaining 3 operations differ only in the order of arguments, so we are left with 1. This is the so called Pixley operation and both majority and minority can be composed from it. It remains to show that each idempotent operation f is a projection provided the clone does not contain any non–projection binary or ternary operation. This can be done e.g. by case analysis using only that each $g(x,y,z) = f(x/y/z, \dots, x/y/z)$ is a projection. ◀

We are ready to show a simple proof of the dichotomy theorem for Boolean CSPs.

▶ **Theorem 63.** *Let $\mathcal{D}$ be a constraint language with domain $D = \{0,1\}$. Then*

- *either its polymorphism clone contains a constant unary operation or one of the four operations in Lemma 62 and then* $\mathrm{CSP}(\mathcal{D})$ *is solvable in polynomial time, or*
- $\mathrm{CSP}(\mathcal{D})$ *is NP–complete.*

Proof. If the polymorphism clone $\mathbf{D}$ of $\mathcal{D}$ contains a constant unary operation with value a, then all relations contain a constant tuple $(a,a,\dots,a)$ and then all instances of $\mathrm{CSP}(\mathcal{D})$ have a solution. Otherwise, $\mathcal{D}$ is a core. Then either $\mathbf{D}$ contains only essentially unary operations, or not. In the first case, the singleton expansion $\mathcal{E}$ has only trivial polymorphisms (=projections), $\mathcal{E}$ pp-defines (by Theorem 32) all structures on $\{0,1\}$ including $\mathcal{D}_{3SAT}$, and then $\mathrm{CSP}(\mathcal{E})$ as well as $\mathrm{CSP}(\mathcal{D})$ are NP–complete by Theorem 13 and Theorem 17. In the second case, the polymorphism clone also contains an idempotent non–projection. By the previous lemma, $\mathbf{D}$ contains one of the four operations and then $\mathrm{CSP}(\mathcal{D})$ is solvable in polynomial time by Theorem 57, Theorem 61, or Example 29. ◀

As a non–trivial exercise, the reader may verify a finer description of the polynomial cases: if $\mathcal{D}$ has min as a polymorphism then $\mathcal{D}$ is pp-definable from $\mathcal{D}_{\mathrm{HornSAT}}$ (and dually for max), if $\mathcal{D}$ has the majority polymorphism then $\mathcal{D}$ is pp-definable from $\mathcal{D}_{\mathrm{2SAT}}$, and if $\mathcal{D}$ has the minority polymorphism then $\mathcal{D}$ is pp-definable from $\mathcal{D}_{\mathrm{3LIN2}}$.

We remark that classifications of Boolean CSPs with respect to other complexity classes and with respect to width notions mentioned in previous section can be found in [1, 97].

5.5 Characterization of Bounded Width

An elegant way to formalize "polynomial solvability by constraint propagation" in general is by means of (k,l)-minimality.

▶ **Definition 64.** Let $1 \leq k \leq l$ be integers. An instance of the CSP is (k,l)-minimal if

- no scope of a constraint contains repeated variables,
- every l-element set of variables is within the scope of some constraint, and
- for any at most k-element set of variables W and any two constraints whose scope contains W, the projections of these constraints onto W coincide.

The reader may notice a formal difference between this definition and its special case in Definition 59. Indeed, a $(2,3)$-minimal instance in the latter does not contain any ternary constraints, while the former one requires that any triple of variables is covered by a constraint. However, the difference is only cosmetic. The sole purpose of the second condition in Definition 64 is to ensure an analogue of the first condition in Definition 59.

For fixed k,l, there is a straightforward polynomial algorithm, the *(k,l)–minimality algorithm*, to transform any CSP instance into a (k,l)-minimal instance with the same set of solutions. As before, instances with an empty constraint relation are refuted.

▶ **Definition 65.** We say that a constraint language $\mathcal{D}$ has width (k,l) if the (k,l)–minimality algorithm refutes every unsatisfiable instance of CSP($\mathcal{D}$).

We say that $\mathcal{D}$ has *width* k if it has width (k,l) for some l and it has *bounded width* if it has width k for some k. (Recall again that bounded width CSPs are solvable in polynomial time.)

The notion of bounded width comes in various versions and equivalent forms. Bounded width is equivalent to solvability by a Datalog program [65], to the existence of a winning strategy in a certain pebble game [65], to having bounded treewidth duality [44], and to definability in an infinitary finite-variable logic [4, 87].

As mentioned in Subsection 4.4, a typical problem which cannot be efficiently solved by local propagation algorithms is solving systems of linear equations [65].

▶ **Theorem 66.** *For any prime p, the constraint language $\mathcal{D}_{\mathrm{3LIN_p}}$ does not have bounded width.*

There is an analogue of Corollary 20 for bounded width [100]: If $\mathcal{D}$ pp-constructs $\mathcal{E}$ and $\mathcal{D}$ has bounded width, then so does $\mathcal{E}$. Therefore, pp-constructing $\mathcal{D}_{\mathrm{3LIN_p}}$ is the "obvious" obstruction to having bounded width. Is it the only obstruction? A positive answer was conjectured in several equivalent forms in [65, 100, 36] (see also [98]).

The process of resolving this, so called *bounded width conjecture*, nicely illustrates the role of universal algebra in identifying meaningful intermediate classes.

- Extending the positive result for semilattice polymorphisms, Bulatov [37] confirmed the conjecture for constraint languages with so called 2–semilattice polymorphisms.
- Very natural candidates for extending the positive result for near unanimity polymorphisms are constraint languages with Jónsson polymorphisms (see Subsection 4.5). Indeed, clones with Jónsson operations are among the most studied objects in universal algebra. Partial results were obtained in [86, 48] and a full solution for Jónsson polymorphisms given in [11].

Helped greatly by these partial results, the bounded width conjecture was confirmed in [14] and independently in an unpublished manuscript by Bulatov [31] (see also [42]).

▶ **Theorem 67.** *A constraint language $\mathcal{D}$ has bounded width if and only if $\mathcal{D}$ does not pp-construct $\mathcal{D}_{\mathrm{3LIN_p}}$ for any prime p.*

The proof from [14] is, to some extent, explained in another survey [16] in this volume. Here we only mention two differences from the arguments in Theorems 57 and 61. First, polymorphisms characterizing the necessary condition for bounded width, such as those in Theorem 47, are not used directly. They are first iteratively composed to get polymorphisms of large arities with properties helpful for the proof. Second, the solution is not directly obtained from a sufficiently consistent instance. Instead, polymorphisms serve to "condense" the constraints to smaller and smaller subsets of the domain, while preserving a sufficient degree of consistency.

5.6 Sufficient Levels of Consistency

Bounded width CSPs are those for which enforcing a certain level of local consistency guarantees a solution. What degree of consistency is actually needed?

The proof of Theorem 66 from [14] uses a consistency notion which is, for binary instances, stronger than 1-minimality and weaker than $(2,3)$–minimality. A small refinement from [10] shows that an appropriate notion of consistency, still weaker than $(2,3)$–minimality, is enough in general. In particular, any bounded width CSP has width $(2,3)$.

A substantial strengthening of Theorem 66 by Kozik [90] weakens the consistency requirement even further. In particular, his result implies that the following consistency procedure, so called *Singleton Arc Consistency*, or *SAC*, is sufficient: ensure that, for each variable x and any $a \in P_x$, the 1-minimality algorithm does not refute the instance even with the added unary constraint $C_a(x)$.

As we mentioned before, there is a connection between consistency notions and certain convex programming relaxations of the CSP, namely the *canonical Linear Programming (LP) relaxation* (see Section 5.2) and the *canonical Semidefinite Programming (SDP) relaxation.*

Again for simplicity, we will restrict to instances that contain variables $x_1, \ldots, x_n$ and exactly one binary constraint $P_{i,j}(x_i, x_j)$ for every $1 \leq i < j \leq n$ (and no other constraints). In particular, there are $m = n(n-1)/2$ constraints. To describe the *canonical SDP relaxation*, we will re-use notation from the Section 5.2, but the variables $\lambda_{i,a}$ now become vectors in Euclidean space (of dimension $O(n)$), $\sigma_{i,j,a,b}$ is required to be equal to the dot product $\lambda_{i,a} \cdot \lambda_{j,b}$ (which is required to be non-negative) and the first constraint is replaced by the requirement that $\lambda_{i,a}$, $a \in D$ are pairwise orthogonal and sum up to a fixed unit vector (the remaining two constraints are then redundant). As before, we say that this relaxation decides a CSP if $\mathrm{Opt}_{\mathrm{SDP}} = 1$ only if the instance has a solution. It was proved in [15] that the canonical SDP relaxation not only decides every bounded width CSP, but it can also be used to give a polynomial time *robust algorithm* for such CSPs, that is, provide "almost solutions" even to "almost satisfiable" instances. More formally, a robust algorithm is an approximation algorithm which, on every instance where $(1-\epsilon)$-fraction of constraints can be satisfied, returns an assignment that satisfies at least $(1-g(\epsilon))$-fraction of constraints, where g is such that $g(\epsilon) \to 0$ as $\epsilon \to 0$. By combining this result from [15] with [69, 57], we get that efficient robust solvability is equivalent to bounded width.

▶ **Theorem 68.** *The following are equivalent for any constraint language $\mathcal{D}$.*
- *$\mathcal{D}$ has bounded width.*
- *$\mathcal{D}$ has width $(2,3)$.*
- $\mathrm{CSP}(\mathcal{D})$ *is solvable by SAC.*
- $\mathrm{CSP}(\mathcal{D})$ *is decided by the canonical SDP relaxation.*
- $\mathrm{CSP}(\mathcal{D})$ *has a robust polynomial algorithm (this item is only equivalent to the rest if $P \neq NP$).*

5.7 Results About Linear and Symmetric Width

We briefly discussed the notions of linear width and symmetric width in Section 4.4. These notions, introduced in [52] and [62], respectively, attract attention for several reasons. They have many natural equivalent descriptions in terms of logic and in combinatorial terms (see, e.g. [44]). They have natural necessary conditions in terms of very simple forbidden constraint languages and well-known algebraic conditions (see Theorems 48 and 53). These necessary conditions are conjectured to be sufficient, see Conjectures 50 and 54. Moreover, having bounded linear (resp. symmetric) width is conjectured to be the single reason for $\mathrm{CSP}(\mathcal{D})$ to be in NL and L, respectively.

Progress towards Conjecture 50 has been made by using the taxonomy from Section 4.5. The conjecture has been confirmed for increasingly weaker assumptions: that $\mathcal{D}$ has the dual discriminator polymorphism [52], a majority polymorphism [56], an NU polymorphism [19] (and hence Jónsson polymorphisms [8]). In [47], the conjecture was also confirmed for a class of constraint languages consisting of (possibly all) languages of width 1 that satisfy the conditions of Theorem 48; this class contains languages without NU polymorphisms. It seems that that the current results approach the full conjecture very closely, and the next step will probably be the full resolution of the conjecture.

To have bounded symmetric width for a language $\mathcal{D}$, it is necessary that $\mathcal{D}$ has bounded linear width and satisfies the conditions of Theorem 52 (see Fig. 1). It was shown in [59] that any constraint language of bounded linear width that has a Mal'tsev polymorphism has bounded symmetric width. Other partial results towards Conjecture 54 (specifically related to H-COLORING) can be found in [54]. Recently Kazda proved [84] that every structure $\mathcal{D}$ having bounded linear width and satisfying the conditions of Theorem 52 in fact has bounded symmetric width. Thus, a characterization of CSPs of bounded linear width would also give a characterization of CSPs of bounded symmetric width. In particular, Conjecture 54 reduces to Conjecture 50.

6 Polymorphisms in Algorithms II: Cogs in the Works

Gaussian elimination not only solves 3-LIN(p), it also describes all the solutions in the sense that the algorithm can output a small (polynomial in n, the number of variables) set of points in $\mathrm{GF}(p)^n$ so that the affine hull of these points is equal to the solution set of the original instance. A sequence of papers [65, 35, 32, 53] culminating in [74, 23] pushed this idea, in a way, to its limit.

6.1 Few Subpowers

We need some terminology to state the result. Let $\mathcal{D}$ be a constraint language and $\mathbf{D}$ its clone of polymorphisms. Let us call a relation on D a *subpower* of $\mathbf{D}$ if it is pp-definable from $\mathcal{D}$, or equivalently by Theorem 32, if it is invariant under all the polymorphisms of $\mathcal{D}$. Note that the set of solutions of any instance of CSP$(\mathcal{D})$ can be viewed as a subpower of $\mathbf{D}$. If R is a subpower of $\mathbf{D}$ and $X \subseteq R$, then we say that X is an *algebraic generating set* of R if R is the smallest subpower of $\mathbf{D}$ containing X. In this case R is precisely the set of values of polymorphisms of $\mathcal{D}$ applied coordinate-wise to tuples from X. Now $\mathbf{D}$ has *few subpowers* if it satisfies any of the equivalent conditions in the following theorem.

▶ **Theorem 69** ([23]). *For any clone $\mathbf{D}$, the following are equivalent.*

1. *There is a polynomial p such that $|\{R \subseteq D^n \mid R \text{ is a subpower of } \mathbf{D}\}| \leq 2^{p(n)}$*
2. *There is a polynomial q such that each subpower $R \subseteq D^n$ of $\mathbf{D}$ has an algebraic generating set with at most $q(n)$ elements.*
3. *For some $k \geq 2$, $\mathbf{D}$ contains a k-edge operation.*

The name "few subpowers" comes from condition (1) of the above theorem, but we will use conditions (2) and (3). See Section 4.5 for the definition of a k-edge operation.

To describe the examples to follow, we need a few more definitions.

▶ **Definition 70.** Let $X \subseteq D^n$.

1. If $i_1, i_2, \ldots, i_k$ is a sequence of indices from $\{1, \ldots, n\}$, then the *projection of X onto coordinates $i_1, \ldots, i_k$*, denoted $\mathrm{proj}_{i_1,\ldots,i_k} X$, is the set $\{(a_{i_1}, \ldots, a_{i_k}) : (a_1, \ldots, a_n) \in R\}$.
2. A *fork* (of arity n) is a triple (i, a, b) with $1 \leq i \leq n$, $a, b \in D$, and $a \neq b$.
3. A *realization* of a fork (i, a, b) is a pair $\mathbf{a}, \mathbf{b} \in D^n$ satisfying $a_j = b_j$ for all $j < i$ and $(a_i, b_i) = (a, b)$. If $\mathbf{a}, \mathbf{b} \in X$ then we say that (i, a, b) is *realized in X*.

▶ **Example 71.** Suppose $\mathcal{D}$ has a Mal'tsev polymorphism (see Subsection 4.5) and $\mathbf{D}$ is its clone of polymorphisms. Then $\mathbf{D}$ has few subpowers [32]. To prove this, let $R \subseteq D^n$ be a subpower of $\mathbf{D}$. We will show that R has a generating set of size at most $q(n) := cn$ where $c = |D|^2$.

We say that a subset $X \subseteq R$ *witnesses single projections and forks* if (i) $\mathrm{proj}_i R = \mathrm{proj}_i X$ for all $i = 1, \ldots, n$, and (ii), R and X realize the same forks. It is easy to see that if X is a minimal subset of R which witnesses single projections and forks, then $|X| \leq q(n)$.

It turns out that if $X \subseteq R$ and X witnesses single projections and forks, then X generates R. To see this, let R' be the set of tuples obtained by applying polymorphisms of $\mathcal{D}$ coordinate-wise to tuples from X. Clearly, R' is a subpower of $\mathbf{D}$ and $X \subseteq R' \subseteq R$. To prove that $R' = R$, let i be maximum index so that $\mathrm{proj}_{1,2,\ldots,i} R = \mathrm{proj}_{1,2,\ldots,i} R'$, and suppose for the sake of contradiction that $i < n$. Choose $\mathbf{a} \in R$ satisfying $\mathrm{proj}_{1,2,\ldots,i+1}(\mathbf{a}) \notin \mathrm{proj}_{1,2,\ldots,i+1} R'$ (such $\mathbf{a}$ must exist by our choice of i and assumption that $i \neq n$). Also by our choice of i, there is a tuple $\mathbf{b} \in R'$ satisfying $\mathrm{proj}_{1,2,\ldots,i}(\mathbf{b}) = \mathrm{proj}_{1,2,\ldots,i}(\mathbf{a})$. Since $\mathbf{a}, \mathbf{b} \in R$, they witness that R realizes the fork $(i+1, a_{i+1}, b_{i+1})$. X must also realize this fork, say by $\mathbf{c}, \mathbf{d}$. Let $\mathbf{e} = f(\mathbf{b}, \mathbf{d}, \mathbf{c})$ where f is the Mal'tsev polymorphism of $\mathcal{D}$ and the application of f to $\mathbf{b}, \mathbf{d}, \mathbf{c}$ is done coordinate-wise. The first $i+1$ coordinates of $\mathbf{b}, \mathbf{d}, \mathbf{c}$ have the form

$$\begin{aligned} \mathbf{b} &= (a_1, \ldots, a_i, b_{i+1}, \underline{\quad}) \\ \mathbf{d} &= (c_1, \ldots, c_i,\, b_{i+1}, \underline{\quad}) \\ \mathbf{c} &= (c_1, \ldots, c_i,\, a_{i+1}, \underline{\quad}). \end{aligned}$$

Because f is a Mal'tsev polymorphism, it satisfies $f(y, x, x) = f(x, x, y) = y$ for all $x, y \in D$. In particular, $e_j = f(a_j, c_j, c_j) = a_j$ for $j = 1, \ldots, i$, and $e_{i+1} = f(b_{i+1}, b_{i+1}, a_{i+1}) = a_{i+1}$. That is, $\mathbf{a}$ and $\mathbf{e}$ agree on the first $i+1$ coordinates. Also note that $\mathbf{e} \in R'$ since $\mathbf{b}, \mathbf{c}, \mathbf{d} \in R'$ and R' is invariant under f. But this is contrary to the choice of $\mathbf{a}$.

The above proof that $R' = R$ used only the following properties of R': (i) $X \subseteq R' \subseteq R$, and (ii) R' is invariant under the Mal'tsev polymorphism f. This proves that R is the *closure* of X under f, meaning that R is the output Y of the following "closure" algorithm.

```
input $X$, $f$
let $Y := X$
repeat
    for every $\mathbf{a}, \mathbf{b}, \mathbf{c} \in Y$ do
        let $Y := Y \cup \{f(\mathbf{a}, \mathbf{b}, \mathbf{c})\}$
    end for
until $Y$ doesn't change
```

This observation is used crucially in the few subpowers algorithm.

▶ **Example 72.** Suppose $\mathcal{D}$ has a majority polymorphism (see Subsection 4.5). Then its clone of polymorphisms $\mathbf{D}$ has few subpowers [6]. Indeed, if $R \subseteq D^n$ is a subpower of $\mathbf{D}$, $X \subseteq R$, and X and R have the same "double projections," that is, $\mathrm{proj}_{i,j} X = \mathrm{proj}_{i,j} R$ for all $1 \leq i, j \leq n$, then X generates R. Indeed, let R' be the subpower generated by X, so $X \subseteq R' \subseteq R$. The idea of the proof that $R' = R$ is very similar to the proof of Theorem 61; show that R' and R have the same projection onto any triple of coordinates, then any 4-tuple of coordinates, etc. Since a suitable set $X \subseteq R$ having the same double projections as R can be found satisfying $|X| \leq cn^2$, R has a small generating set. As in the previous example, the proof actually shows that R is the closure of X under the majority polymorphism.

More generally, if $\mathcal{D}$ has a k-ary NU polymorphism, then a subpower $R \subseteq D^n$ is generated by any set $X \subseteq R$ which has the same projection as R onto each set of coordinates of size

$k-1$. Such an X can always be found satisfying $|X| \leq dn^{k-1}$ (for a suitable constant d). Again R is the closure of X under the NU polymorphism.

Note that if $\mathcal{D}, R$, and X are as in either Example 71 or 72, then one cannot expect to efficiently run the closure algorithm to construct R from X, since the size of R could be exponential in the size of X. However, one can efficiently calculate the projection of R onto any small set s of coordinates (say of size at most 5), and even find a (small) subset of R containing X whose projection onto s agrees with R. This is because R is the closure of X under a single fixed operation f, so we can simply run the closure algorithm for X and f, but adding new elements to Y only when their projection onto s is new. With this restriction, there is a constant upper bound ($|D|^{|s|}$) to the number of times the main loop of the closure algorithm will be repeated; hence this restricted algorithm runs in polynomial time. This observation is one of the fundamental tools of the few subpowers algorithm.

6.2 The Few Subpowers Algorithm

▶ **Theorem 73** ([74]). *Let $\mathcal{D}$ be an idempotent constraint language. If the clone of polymorphisms $\mathbf{D}$ has few subpowers, then* $\mathrm{CSP}(\mathcal{D})$ *can be solved in polynomial time (moreover, the algorithm can output a generating set for the set of all solutions).*

Suppose $\mathbf{D}$ has few subpowers. The main idea of the few subpowers algorithm, which can be traced back to [65], is the following. Given a $\mathrm{CSP}(\mathcal{D})$ instance $P = (V, D, \mathcal{C})$ with, say, all constraint relations at most binary, we can enumerate $V = \{x_1, \dots, x_n\}$ and $\mathcal{C} = (C_1, \dots, C_m)$ and consider the decreasing sequence

$$D^n = R_0 \supseteq R_1 \supseteq \cdots \supseteq R_i \supseteq \cdots \supseteq R_m \tag{26}$$

of subpowers of $\mathbf{D}$, where R_i is the set of solutions to the first i constraints. The aim of the algorithm is to construct a small generating set X_i for each R_i. Then X_m will be a small generating set for the set of solutions to P, so P has a solution if and only if X_m is not empty. It should be easy to construct the generating set X_0 for D^n. Thus the chief task of the algorithm is that of finding the "next" X_{i+1} given X_i and C_{i+1}. We sketch how this is done in the two easiest cases: $\mathcal{D}$ has a majority or Mal'tsev polymorphism.

Case 1. $\mathcal{D}$ has a majority polymorphism f. We assume that X is a small subset of the subpower R which has the same double projections as R, and C is a (say) binary constraint $((x_k, x_\ell), S)$. Let R' be the "next" subpower determined by X and C; that is, $R' = \{\mathbf{a} \in R : S(a_k, a_\ell)\}$. The goal is to find a small subset X' of R' which has the same double projections as R'. This is easy. As shown in Example 72, R is the closure under f of X. Thus for each pair (i, j) of coordinates, we can (using the restricted closure algorithm for f) find a small subset $X_{i,j}$ of R satisfying $\mathrm{proj}_{i,j,k,\ell} X_{i,j} = \mathrm{proj}_{i,j,k,\ell} R$. If we let $X_1 = \bigcup_{i<j} X_{i,j}$, then the set $X' = \{\mathbf{a} \in X_1 : S(a_k, a_\ell)\}$ is a small subset of R' and has the same double projections as R', as required. Note that the few subpowers algorithm in this case consists of repeated applications of the restricted closure algorithm using the majority polymorphism.

The case when $\mathcal{D}$ has an NU polymorphism is handled similarly.

Case 2. $\mathcal{D}$ has a Mal'tsev polymorphism f. We assume that X is a small subset of the subpower R which witnesses single projections and forks, and C is an at-most binary constraint. Again let R' be the "next" subpower determined by X and C. The goal this time is to find a small subset X' of R' witnessing single projections and forks (of R'). Witnessing single projections is done analogously to the argument in the majority case. Witnessing forks requires some ingenuity.[13] We first show how to do this in a very special case: C is a singleton unary constraint $(x_k, \{c\})$ (recall that $\mathcal{D}$ is idempotent) *and* the projections of R onto each coordinate $j < k$ have size 1. In this situation, necessary conditions for a fork (i, a, b) to be realized in R' are

1. $i > k$
2. (i, a, b) is realized in X, say by $\mathbf{a}, \mathbf{b}$.
3. R has a tuple $\mathbf{c}$ whose projection on coordinates k, i satisfies $(c_k, c_i) = (c, a)$.

(Note that we can efficiently find $\mathbf{a}, \mathbf{b}, \mathbf{c}$ when they exist, or determine that they do not exist.) In fact, these conditions are also sufficient, because the tuples $\mathbf{c}$ and $\mathbf{d} := f(\mathbf{c}, \mathbf{a}, \mathbf{b})$ belong to R' and can be shown to realize the fork (i, a, b) by reasoning similar to that in Example 71, using the Mal'tsev operation identities.

Now we consider the general case where C is a (say) binary constraint $((x_k, x_\ell), S)$. Let (i, a, b) be a fork. A necessary condition for (i, a, b) to be realized in R' is that R contain a tuple $\mathbf{c}$ whose projection onto coordinates i, k, ℓ is in $\{a\} \times S$. Because R is the closure under f of X, we can find such $\mathbf{c}$ when it exists, by the restricted closure algorithm. Suppose such $\mathbf{c}$ is found. Note that $\mathbf{c} \in R'$ and $c_i = a$. Now a key property of R' is that if the fork (i, a, b) is realized in R', then $\mathbf{c}$ is one half of such a realization. For suppose $\mathbf{u}, \mathbf{v} \in R'$ realize (i, a, b). Then $\mathbf{d} := f(\mathbf{c}, \mathbf{u}, \mathbf{v})$ is in R' and, arguing as above, $\mathbf{c}, \mathbf{d}$ realize (i, a, b).

So we just need to search for a tuple $\mathbf{d}$ in R which agrees with $\mathbf{c}$ on its first $i-1$ coordinates, equals b at coordinate i, and which satisfies $(d_k, d_\ell) \in S$. This search is accomplished by cutting down R to the subrelation $R_1 = \{\mathbf{x} \in R : x_1 = c_1 \ \& \ x_2 = c_2 \ \& \ \cdots \ \& \ x_{i-1} = c_{i-1}\}$. Using the previous "special case" argument $i - 1$ times, we can find a small subset X_1 of R_1 which witnesses projections and forks for R_1, and then use X_1 and the restricted closure algorithm to search for an element $\mathbf{d} \in R_1$ (if it exists) whose projection onto coordinates i, k, ℓ is in $\{b\} \times S$. These searches can be done in polynomial time, using the restricted closure algorithm.

In summary, the implementation of the few subpowers algorithm in the Mal'tsev case is a carefully orchestrated sequence of applications of the restricted closure algorithm for the Mal'tsev polymorphism, with some additional computations using this polymorphism.

In general, the few subpowers algorithm [74] works more or less as the union of the NU and Mal'tsev cases, following Dalmau [53] who was the first to combine these two cases. By Theorem 69, if the polymorphism clone of $\mathcal{D}$ has few subpowers, then $\mathcal{D}$ has a k-edge polymorphism for some $k \geq 2$. A notion of "nice" small generating sets (analogous to witnessing all $(k-1)$-ary projections and forks) for subpowers is worked out in [23], and the above argument in the Mal'tsev case is more or less repeated verbatim. The point we wish to make here is that the edge polymorphism is used essentially and repeatedly by the algorithm. Without an edge polymorphism, the above implementation of the few subpowers algorithm cannot be executed.

6.3 Limits of the Few Subpowers Algorithm

Given an operation f on a finite set D, let $\mathcal{D}_f$ be the (infinite) set of all relations invariant under f. CSP($\mathcal{D}_f$) is called a "global" problem encompassing all of the "local" problems CSP($\mathcal{D}$) where $\mathcal{D}$ ranges over finite subsets of $\mathcal{D}_f$.

[13] This argument is due to Dalmau [32, 53].

If the polymorphism clone $\mathbf{D}$ of $\mathcal{D}_f$ contains a k-edge operation, then the few subpowers algorithm as formulated in [74] actually solves the global problem CSP($\mathcal{D}_f$) in polynomial time.[14] Conversely, suppose there exists a polynomial-time algorithm which solves the global problem CSP($\mathcal{D}_f$) *and* broadly follows the “small generating sets” algorithm-idea outlined at the beginning of Subsection 6.2. An essential requirement of this algorithm-idea is that

> There is a polynomial $p(n)$ such that if $R \subseteq D^n$ is a subpower of $\mathbf{D}$ *and* R occurs as R_i in equation (26), for some instance of CSP($\mathcal{D}_f$), then R has a generating set of size at most $p(n)$.

Because of how $\mathcal{D}_f$ is defined, *every* subpower $R \subseteq D^n$ of $\mathbf{D}$ can arise as R_i in equation (26) for some instance of CSP($\mathcal{D}_f$) (for the simple reason that every subpower R belongs to $\mathcal{D}_f$). Hence the displayed requirement and Theorem 69 imply $\mathbf{D}$ must contain a k-edge operation. It is in this sense that polymorphism clones having a k-edge operation form the natural “limit” of the “small generating sets” algorithm-idea outlined at the beginning of Subsection 6.2.

However if one is primarily interested (as we are) in CSP over finite constraint languages, rather than in global problems, then the limit of the “small generating sets” algorithm-idea is not settled. If $\mathcal{D}$ is finite, then the subpowers of the polymorphism clone which arise as R_i in equation (26) are precisely the relations which are pp-definable from $\mathcal{D}$ without using $\exists$. It is not known if there exists a finite constraint language $\mathcal{D}$ which does not have a k-edge polymorphism for any k, and yet which has the property that every $\exists$-free pp-defined relation R has a small generating set. If such a constraint language $\mathcal{D}$ exists, it might be a candidate for a “small generating sets” algorithm more general than the few subpowers algorithm.

6.4 Combining Algorithms

Suppose $\mathcal{D}$ is idempotent, has a Taylor polymorphism, but does not have polymorphisms as described in Theorem 47 implying bounded width (see Theorem 67), nor does $\mathcal{D}$ have an edge polymorphism implying few subpowers. What tools are available to solve CSP($\mathcal{D}$)?

Starting from an instance $P = (V, D, \mathcal{C})$ of CSP($\mathcal{D}$), one strategy is to “shrink” P to a 1-minimal CSP instance Q satisfying the following properties:

1. You can prove that Q has a solution if and only if P does.
2. The constraint relations of Q are pp-definable from $\mathcal{D}$.
3. For each $x \in V$, if $Q_x(x)$ is the unique unary constraint of Q on x, then Q_x is a proper subset of D.

Suppose now that there exists a polymorphism f of $\mathcal{D}$ such that for each $x \in V$, the restriction of f to Q_x is an edge operation. Then the few subpowers algorithm, suitably adapted, can determine whether Q (and hence P) has a solution. Similarly, if $\mathcal{D}$ has polymorphisms which, when restricted to each Q_x, satisfy identities implying bounded width, then the (2,3)-minimality algorithm will correctly tell whether Q (and hence P) has a solution.

Thus a possible strategy is to shrink P to the point where some polymorphism(s) of $\mathcal{D}$ become “nice” when restricted to the new domain of each variable. This is essentially the strategy followed (along with much deeper tricks) to prove the results described in this subsection. The first result of this kind is the following result of Bulatov [38].

[14] For this result it is important to maintain the convention that each constraint relation is given by a list of its members – not, for example, by a small generating set.

▶ **Theorem 74.** *Suppose $\mathcal{D}$ has a Taylor polymorphism and $|D| = 3$. Then* CSP($\mathcal{D}$) *is solvable in polynomial time. Hence the Dichotomy Conjecture holds for constraint languages with domains of size 3.*

Reading [38] is not for the faint at heart – the core argument is over 40 pages of dense mathematical argument involving consideration of many cases.

The second result that we want to mention is the solution [39, 7, 41], due originally to Bulatov, of the Dichotomy Conjecture in the "conservative" case.

▶ **Theorem 75.** *Suppose $\mathcal{D}$ includes every nonempty subset of D as a unary relation. If $\mathcal{D}$ has a Taylor polymorphism, then* CSP($\mathcal{D}$) *is solvable in polynomial time.*

An operation f is called *conservative* if $f(a_1, \ldots, a_n) \in \{a_1, \ldots, a_n\}$ for all $a_i \in D$. The above theorem can be equivalently re-stated as "if a constraint language $\mathcal{D}$ has a conservative Taylor polymorphism then CSP($\mathcal{D}$) is tractable." The first proof from [39] was quite long and involved, but the more recent papers [7, 41] give two different much shorter proofs.

Technically, Bulatov's approach in [39, 41] is via a careful analysis of a colored graph associated with a constraint language (see [36, 42]). The domain of this graph is the same as the domain of the languages, and the colored edges reflect the local structure of polymorphisms. The approach in [7] is based on absorption theory, see survey [16].

See also [43] for a brief overview of other attempts, as well as a new approach, to combine known algorithms for solving CSP($\mathcal{D}$).

7 Conclusion

We have seen that the complexity of the decision problem for CSP over a fixed constraint language on a finite domain depends on "higher arity symmetries" – polymorphisms of the language, and more specifically, on the identities satisfied by these polymorphisms. Significant progress has been achieved using this insight, but the main problem, the dichotomy conjecture, is still open. The view shared by many experts at the Dagstuhl seminar (to which this volume is a follow-up) is that the main obstacle towards further progress is the insufficient understanding of the convoluted ways in which system of linear equations can appear in CSP instances.

The authors are often being asked to give a specific example of a (preferably small) constraint language $\mathcal{D}$ that has a Taylor polymorphism, but such that CSP($\mathcal{D}$) is not yet known to be tractable. A computer-assisted analysis of small digraphs and their polymorphisms (with respect to the taxonomy from Fig.1) can be found in [22]. There is an example of a 6-element digraph $\mathcal{D}$ there, whose singleton expansion has a Taylor polymorphism, but such that neither the bounded width algorithm nor the few subpowers algorithm alone can solve CSP($\mathcal{D}$). However, it was shown [103] that an ad hoc combination of the two algorithms solves it. It is the authors' belief that such explicit examples are not easy to find, but they tend not to provide useful insights into how to overcome the difficulties in resolving the dichotomy conjecture.

Polymorphisms have been successfully applied to other variants of CSP over a fixed constraint language. For the (exact) counting problem, the dichotomy has been proved in [40] and then substantially simplified in [61], with polymorphisms (and universal algebra in general) playing a considerable role. This is discussed in survey [81] in this volume, along with many developments in complexity classification of approximate counting problems and for computing partition functions, where there are still many open problems.

A generalization of the algebraic theory for the exact optimization problem (where the goal is find an optimal solution), and for the more general valued CSPs, was given in [51]. The so-called fractional polymorphisms, which are special probability distributions on polymorphisms, play a key role there. Very strong classification results are known in this direction [88, 89, 117], see survey [92] in this volume. In essence, complexity classifications here are complete, but there are many open questions regarding approximate optimization.

The two different views on optimization – maximize satisfaction or minimize dissatisfaction – are obviously equivalent when the goal is to find an optimal solution. However, they exhibit very different behavior with respect to approximation. For the emerging applicability of polymorphisms for the analysis of approximability of the maximization version, see [29, 30]. For the minimization version, there is a series of results about the so-called robust approximation algorithms for CSPs where polymorphisms (and their taxonomy) play a significant role. The full characterization of constraint languages admitting a robust algorithm is known [15] (see Theorem 47), but the refined classification (taking the quality of approximation into account) is far from complete, see [57, 58, 55].

The so-called weak polymorphisms have recently been applied to the so-called promise CSPs [5, 27, 28], with motivation coming from the study of inapproximability. An interesting feature of weak polymorphisms is that they cannot be composed, and yet they can be useful for complexity analysis.

We have only discussed languages with finite domains. The algebraic theory extends to interesting subclasses of the infinite domain CSP; see surveys [24, 25, 107] and recent results [20, 21]. This area contains very many open problems.

Is the polymorphism approach only applicable to CSPs over fixed languages? Or are we merely seeing a piece of a bigger theory?

Polymorphisms have recently been applied, for example, in a non-CSP context of social choice theory [114]. Are there other applications of polymorphisms beyond algebra and CSP?

Acknowledgment. The authors thank Eric Allender for explaining state-of-the-art knowledge about relationships between complexity classes NL and Mod$_p$L. The authors also thank anonymous referees for providing many useful comments.

References

1 Eric Allender, Michael Bauland, Neil Immerman, Henning Schnoor, and Heribert Vollmer. The complexity of satisfiability problems: Refining Schaefer's theorem. *Journal of Computer and System Sciences*, 75(4):245–254, 2009.

2 Eric Allender, Klaus Reinhardt, and Shiyu Zhou. Isolation, matching, and counting uniform and nonuniform upper bounds. *J. Comput. Syst. Sci.*, 59(2):164–181, 1999.

3 Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach.* Cambridge University Press, New York, NY, USA, 1st edition, 2009.

4 Albert Atserias, Andrei A. Bulatov, and Anuj Dawar. Affine systems of equations and counting infinitary logic. *Theor. Comput. Sci.*, 410(18):1666–1683, 2009.

5 Per Austrin, Johan Håstad, and Venkatesan Guruswami. $(2+\epsilon)$-Sat is NP-hard. In *Proceedings, Foundations of Computer Science (FOCS)*, pages 1–10, 2014.

6 Kirby A. Baker and Alden F. Pixley. Polynomial interpolation and the Chinese remainder theorem for algebraic systems. *Mathematische Zeitschrift*, 143:165–174, 1975.

7 Libor Barto. The dichotomy for conservative constraint satisfaction problems revisited. In *Proceedings, Logic in Computer Science (LICS)*, pages 301–310, 2011.

8 Libor Barto. Finitely related algebras in congruence distributive varieties have near unanimity terms. *Canad. J. Math.*, 65(1):3–21, 2013.

9 Libor Barto. Finitely related algebras in congruence modular varieties have few subpowers. to appear in *J. European Math. Soc.*, 2015.

10 Libor Barto. The collapse of the bounded width hierarchy. *J. Log. Comput.*, 26(3):923–943, 2016.

11 Libor Barto and Marcin Kozik. Congruence distributivity implies bounded width. *SIAM Journal on Computing*, 39(4):1531–1542, 2009.

12 Libor Barto and Marcin Kozik. Constraint satisfaction problems of bounded width. In *Proceedings, Foundations of Computer Science (FOCS)*, pages 595–603, 2009.

13 Libor Barto and Marcin Kozik. Absorbing subalgebras, cyclic terms, and the constraint satisfaction problem. *Logical Methods in Computer Science*, 8(1), 2012.

14 Libor Barto and Marcin Kozik. Constraint satisfaction problems solvable by local consistency methods. *J. ACM*, 61(1):3:1–3:19, January 2014.

15 Libor Barto and Marcin Kozik. Robustly solvable constraint satisfaction problems. *SIAM Journal on Computing*, 45(4):1646–1669, 2016.

16 Libor Barto and Marcin Kozik. Absorption in universal algebra and CSP. In *The Constraint Satisfaction Problem: Complexity and Approximability*, pages 45–78. 2017.

17 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, 2008/09.

18 Libor Barto, Marcin Kozik, and David Stanovský. Mal'tsev conditions, lack of absorption, and solvability. *Algebra universalis*, 74(1):185–206, 2015.

19 Libor Barto, Marcin Kozik, and Ross Willard. Near unanimity constraints have bounded pathwidth duality. In *Proceedings, Logic in Computer Science (LICS)*, pages 125–134, 2012.

20 Libor Barto, Jakub Opršal, and Michael Pinsker. The wonderland of reflections. to appear in *Israel J. Math.*, 2016.

21 Libor Barto and Michael Pinsker. The algebraic dichotomy conjecture for infinite domain constraint satisfaction problems. In *Proceedings, Logic in Computer Science (LICS)*, pages 615–622, 2016.

22 Libor Barto and David Stanovský. Polymorphisms of small digraphs. *Novi Sad J. Math.*, 40(2):95–109, 2010.

23 Joel Berman, Pawel Idziak, Petar Marković, Ralph McKenzie, Matthew Valeriote, and Ross Willard. Varieties with few subalgebras of powers. *Transactions of The American Mathematical Society*, 362:1445–1473, 2010.

24 Manuel Bodirsky. Constraint satisfaction problems with infinite templates. In Nadia Creignou, Phokion G. Kolaitis, and Heribert Vollmer, editors, *Complexity of Constraints*, volume 5250 of *Lecture Notes in Computer Science*, pages 196–228. Springer, 2008.

25 Manuel Bodirsky and Marcello Mamino. Constraint satisfaction problems over numeric domains. In *The Constraint Satisfaction Problem: Complexity and Approximability*, pages 79–111. 2017.

26 V. G. Bodnarčuk, L. A. Kalužnin, V. N. Kotov, and B. A. Romov. Galois theory for Post algebras. I, II. *Kibernetika (Kiev)*, (3):1–10; ibid. 1969, no. 5, 1–9, 1969.

27 Joshua Brakensiek and Venkatesan Guruswami. New hardness results for graph and hypergraph colorings. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:29, 2016.

28 Joshua Brakensiek and Venkatesan Guruswami. Promise constraint satisfaction: Algebraic structure and a symmetric boolean dichotomy. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:183, 2016.

29 Jonah Brown-Cohen and Prasad Raghavendra. Combinatorial optimization algorithms via polymorphisms. *CoRR*, abs/1501.01598, 2015.

30 Jonah Brown-Cohen and Prasad Raghavendra. Correlation decay and tractability of CSPs. In *Proceedings, Automata, Languages, and Programming (ICALP)*, pages 79:1–79:13, 2016.

31 Andrei Bulatov. Bounded relational width. manuscript, 2009.

32 Andrei Bulatov and Víctor Dalmau. A simple algorithm for Mal'tsev constraints. *SIAM J. Comput.*, 36(1):16–27 (electronic), 2006.

33 Andrei Bulatov and Peter Jeavons. Algebraic structures in combinatorial problems. Technical Report MATH-AL-4-2001, Technische Universität Dresden, 2001.

34 Andrei Bulatov, Peter Jeavons, and Andrei Krokhin. Classifying the complexity of constraints using finite algebras. *SIAM J. Comput.*, 34:720–742, March 2005.

35 Andrei A. Bulatov. Mal'tsev constraints are tractable. *Electronic Colloquium on Computational Complexity (ECCC)*, (034), 2002.

36 Andrei A. Bulatov. A graph of a relational structure and constraint satisfaction problems. In *Proceedings, Logic in Computer Science (LICS)*, pages 448–457, 2004.

37 Andrei A. Bulatov. Combinatorial problems raised from 2-semilattices. *J. Algebra*, 298(2):321–339, 2006.

38 Andrei A. Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-element set. *J. ACM*, 53(1):66–120 (electronic), 2006.

39 Andrei A. Bulatov. Complexity of conservative constraint satisfaction problems. *ACM Trans. Comput. Logic*, 12(4):24:1–24:66, July 2011.

40 Andrei A. Bulatov. The complexity of the counting constraint satisfaction problem. *J. ACM*, 60(5):34:1–34:41, October 2013.

41 Andrei A. Bulatov. Conservative constraint satisfaction re-revisited. *J. Comput. System Sci.*, 82(2):347–356, 2016.

42 Andrei A. Bulatov. Graphs of relational structures: restricted types. In *Proceedings, Logic in Computer Science (LICS)*, pages 642–651, 2016.

43 Andrei A. Bulatov. Constraint Satisfaction Problems over semilattice block Mal'tsev algebras. *CoRR*, arXiv:1701.02623, 2017.

44 Andrei A. Bulatov, Andrei Krokhin, and Benoit Larose. Complexity of constraints. chapter Dualities for Constraint Satisfaction Problems, pages 93–124. Springer-Verlag, Berlin, Heidelberg, 2008.

45 Jakub Bulin, Dejan Delic, Marcel Jackson, and Todd Niven. A finer reduction of constraint problems to digraphs. *Logical Methods in Computer Science*, 11(4), 2015.

46 Gerhard Buntrock, Carsten Damm, Ulrich Hertrampf, and Christoph Meinel. Structure and importance of logspace-mod class. *Mathematical systems theory*, 25(3):223–237, 1992.

47 Catarina Carvalho, Víctor Dalmau, and Andrei Krokhin. CSP duality and trees of bounded pathwidth. *Theoretical Computer Science*, 411(34-36):3188–3208, 2010.

48 Catarina Carvalho, Víctor Dalmau, Petar Marković, and Miklós Maróti. CD(4) has bounded width. *Algebra Universalis*, 60(3):293–307, 2009.

49 Hubie Chen. Meditations on quantified constraint satisfaction. In *Logic and Program Semantics*, pages 35–49, 2012.

50 Hubie Chen and Benoit Larose. Asking the metaquestions in constraint tractability. arXiv:1604.00932, April 2016.

51 David A. Cohen, Martin C. Cooper, Páidí Creed, Peter Jeavons, and Stanislav Živný. An algebraic theory of complexity for discrete optimisation. *SIAM Journal on Computing*, 42(5):1915–1939, 2013.

52 Víctor Dalmau. Linear Datalog and bounded path duality for relational structures. *Logical Methods in Computer Science*, 1(1), 2005.

53 Víctor Dalmau. Generalized majority-minority operations are tractable. *Log. Methods Comput. Sci.*, 2(4):4:1, 14, 2006.

54 Víctor Dalmau, László Egri, Pavol Hell, Benoit Larose, and Arash Rafiey. Descriptive complexity of list h-coloring problems in logspace: A refined dichotomy. In *Proceedings, Logic in Computer Science (LICS)*, pages 487–498, 2015.

55 Víctor Dalmau, Marcin Kozik, Andrei Krokhin, Konstantin Makarychev, Yury Makarychev, and Jakub Opršal. Robust algorithms with polynomial loss for near-unanimity CSPs. In *Proceedings, Discrete Algorithms (SODA)*, pages 340–357, 2017.

56 Víctor Dalmau and Andrei Krokhin. Majority constraints have bounded pathwidth duality. *European Journal of Combinatorics*, 29(4):821–837, 2008.

57 Víctor Dalmau and Andrei Krokhin. Robust satisfiability for CSPs: Hardness and algorithmic results. *ACM Trans. Comput. Theory*, 5(4):15:1–15:25, November 2013.

58 Víctor Dalmau, Andrei Krokhin, and Rajsekar Manokaran. Towards a characterization of constant-factor approximable Min CSPs. In *Proceedings, Discrete Algorithms (SODA)*, pages 847–857, 2015.

59 Víctor Dalmau and Benoit Larose. Maltsev + datalog –> symmetric datalog. In *Proceedings, Logic in Computer Science (LICS)*, pages 297–306, 2008.

60 Víctor Dalmau and Justin Pearson. Closure functions and width 1 problems. In *Proceedings, Principles and Practice of Constraint Programming (CP)*, pages 159–173, 1999.

61 Martin E. Dyer and David Richerby. An effective dichotomy for the counting constraint satisfaction problem. *SIAM J. Comput.*, 42(3):1245–1274, 2013.

62 László Egri, Benoit Larose, and Pascal Tesson. Symmetric datalog and constraint satisfaction problems in logspace. In *Proceedings, Logic in Computer Science (LICS)*, pages 193–202, 2007.

63 Tomas Feder and Pavol Hell. List homomorphisms to reflexive graphs. *J. Combin. Theory Ser. B*, 72(2):236–250, 1998.

64 Tomas Feder, Pavol Hell, and Jing Huang. List homomorphisms and circular arc graphs. *Combinatorica*, 19(4):487–505, 1999.

65 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.

66 Ralph Freese and Ralph McKenzie. Maltsev families of varieties closed under join or Maltsev product. *Algebra Universalis*, in press.

67 David Geiger. Closed systems of functions and predicates. *Pacific J. Math.*, 27:95–100, 1968.

68 J. Hagemann and A. Mitschke. On n-permutable congruences. *Algebra Universalis*, 3(1):8–12, 1973.

69 Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48:798–859, July 2001.

70 Johan Håstad. On the efficient approximability of constraint satisfaction problems. In *Surveys in combinatorics 2007*, volume 346 of *London Math. Soc. Lecture Note Ser.*, pages 201–221. Cambridge Univ. Press, Cambridge, 2007.

71 Pavol Hell and Jaroslav Nešetřil. On the complexity of H-coloring. *J. Combin. Theory Ser. B*, 48(1):92–110, 1990.

72 Pavol Hell and Jaroslav Nešetřil. *Graphs and homomorphisms*, volume 28 of *Oxford Lecture Series in Mathematics and its Applications*. Oxford University Press, Oxford, 2004.

73 David Hobby and Ralph McKenzie. *The structure of finite algebras*, volume 76 of *Contemporary Mathematics*. American Mathematical Society, Providence, RI, 1988.

74 Pawel M. Idziak, Petar Markovic, Ralph McKenzie, Matthew Valeriote, and Ross Willard. Tractability and learnability arising from algebras with few subpowers. *SIAM J. Comput.*, 39(7):3023–3037, 2010.

75 Neil Immerman. Nondeterministic space is closed under complementation. *SIAM J. Comput.*, 17(5):935–938, October 1988.

76 Marcel Jackson, Tomasz Kowalski, and Todd Niven. Complexity and polymorphisms for digraph constraint problems under some basic constructions. *International Journal of Algebra and Computation*, 26(07):1395–1433, 2016.

77 Peter Jeavons. Constructing constraints. In *Proceedings, Principles and Practice of Constraint Programming (CP)*, pages 2–16, 1998.

78 Peter Jeavons. On the algebraic structure of combinatorial problems. *Theoretical Computer Science*, 200(1–2):185–204, 1998.

79 Peter Jeavons, David Cohen, and Martin Cooper. Constraints, consistency and closure. *Artificial Intelligence*, 101(1-2):251–265, 1998.

80 Peter Jeavons, David Cohen, and Marc Gyssens. Closure properties of constraints. *J. ACM*, 44(4):527–548, 1997.

81 Mark Jerrum. Counting constraint satisfaction problems. In *The Constraint Satisfaction Problem: Complexity and Approximability*, pages 201–228. 2017.

82 Jelena Jovanović, Petar Marković, Ralph McKenzie, and Matthew Moore. Optimal strong Mal'cev conditions for congruence meet-semidistributivity in locally finite varieties. *Algebra universalis*, 76(3):305–325, 2016.

83 Alexandr Kazda. Maltsev digraphs have a majority polymorphism. *European J. Combin.*, 32(3):390–397, 2011.

84 Alexandr Kazda. \$n\$-permutability and linear datalog implies symmetric datalog. *CoRR*, abs/1508.05766, 2015.

85 Keith Kearnes, Petar Marković, and Ralph McKenzie. Optimal strong Mal'cev conditions for omitting type 1 in locally finite varieties. *Algebra Universalis*, 72(1):91–100, 2014.

86 Emil Kiss and Matthew Valeriote. On tractability and congruence distributivity. *Log. Methods Comput. Sci.*, 3(2):2:6, 20 pp. (electronic), 2007.

87 Phokion G. Kolaitis and Moshe Y. Vardi. Conjunctive-query containment and constraint satisfaction. *J. Comput. Syst. Sci.*, 61(2):302–332, 2000.

88 Vladimir Kolmogorov, Andrei Krokhin, and Michal Rolinek. The complexity of general-valued CSPs. In *Proceedings, Foundations of Computer Science (FOCS)*, pages 1246–1258, 2015.

89 Vladimir Kolmogorov, Johan Thapper, and Stanislav Živný. The power of linear programming for general-valued CSPs. *SIAM J. Comput.*, 44(1):1–36, 2015.

90 Marcin Kozik. Weak consistency notions for all the CSPs of bounded width. In *Proceedings, Logic in Computer Science (LICS)*, pages 633–641, 2016.

91 Marcin Kozik, Andrei Krokhin, Matt Valeriote, and Ross Willard. Characterizations of several Maltsev conditions. *Algebra universalis*, 73(3):205–224, 2015.

92 Andrei Krokhin and Stanislav Živný. The complexity of valued CSPs. In *The Constraint Satisfaction Problem: Complexity and Approximability*, pages 229–261, 2017.

93 Gabor Kun, Ryan O'Donnell, Suguru Tamaki, Yuichi Yoshida, and Yuan Zhou. Linear programming, width-1 CSPs, and robust satisfaction. In *Proceedings, Innovations in Theoretical Computer Science (ITCS)*, pages 484–495, 2012.

94 Gábor Kun and Mario Szegedy. A new line of attack on the dichotomy conjecture. *European Journal of Combinatorics*, 52, Part B:338–367, 2016. Special Issue: Recent Advances in Graphs and Analysis.

95 Richard E. Ladner. On the structure of polynomial time reducibility. *J. ACM*, 22:155–171, 1975.

96 Benoît Larose. Algebra and the complexity of digraph CSPs: a survey. In *The Constraint Satisfaction Problem: Complexity and Approximability*, pages 263–282. 2016.

97 Benoît Larose and Pascal Tesson. Universal algebra and hardness results for constraint satisfaction problems. *Theor. Comput. Sci.*, 410:1629–1647, April 2009.

98 Benoit Larose, Matt Valeriote, and László Zádori. Omitting types, bounded width and the ability to count. *Internat. J. Algebra Comput.*, 19(5):647–668, 2009.

99 Benoit Larose and László Zádori. Taylor terms, constraint satisfaction and the complexity of polynomial equations over finite algebras. *Internat. J. Algebra Comput.*, 16(3):563–581, 2006.

100 Benoit Larose and László Zádori. Bounded width problems and algebras. *Algebra Universalis*, 56(3-4):439–466, 2007.

101 Florent Madelaine and Barnaby Martin. A tetrachotomy for positive first-order logic without equality. In *Proceedings, Logic in Computer Science (LICS)*, pages 311–320, 2011.

102 Konstantin Makarychev and Yuri Makarychev. Approximation algorithms for CSPs. In *The Constraint Satisfaction Problem: Complexity and Approximability*, pages 283–320. 2017.

103 Petar Marković. Private communication, 2011.

104 Miklós Maróti and Ralph McKenzie. Existence theorems for weakly symmetric operations. *Algebra Universalis*, 59(3-4):463–489, 2008.

105 Dániel Marx. Tractable hypergraph properties for constraint satisfaction and conjunctive queries. *J. ACM*, 60(6):42:1–42:51, November 2013.

106 Christos H. Papadimitriou. *Computational complexity.* Addison-Wesley Publishing Company, Reading, MA, 1994.

107 Michael Pinsker. Algebraic and model theoretic methods in constraint satisfaction. *CoRR*, abs/1507.00931, 2015.

108 Emil L. Post. *The Two-Valued Iterative Systems of Mathematical Logic. (AM-5).* Annals of Mathematics Studies. Princeton University Press, 1941.

109 Omer Reingold. Undirected connectivity in log-space. *J. ACM*, 55(4):17:1–17:24, September 2008.

110 Klaus Reinhardt and Eric Allender. Making nondeterminism unambiguous. *SIAM J. Comput.*, 29(4):1118–1131, 2000.

111 Francesca Rossi, Peter van Beek, and Toby Walsh. *Handbook of Constraint Programming (Foundations of Artificial Intelligence).* Elsevier Science Inc., New York, NY, USA, 2006.

112 Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings, Theory of Computation (STOC)*, pages 216–226, 1978.

113 Mark H. Siggers. A strong Mal'cev condition for locally finite varieties omitting the unary type. *Algebra universalis*, 64(1-2):15–20, 2010.

114 Mario Szegedy and Yixin Xu. Impossibility theorems and the universal algebraic toolkit. *CoRR*, abs/1506.01315, 2015.

115 Róbert Szelepcsényi. The method of forced enumeration for nondeterministic automata. *Acta Inf.*, 26(3):279–284, November 1988.

116 Walter Taylor. Varieties obeying homotopy laws. *Canad. J. Math.*, 29(3):498–527, 1977.

117 Johan Thapper and Stanislav Živný. The complexity of finite-valued CSPs. *J. ACM*, 63(4):37:1–37:33, 2016.

118 Ross Willard. Testing expressibility is hard. In *Proceedings, Principles and Practice of Constraint Programming (CP)*, pages 9–23, 2010.

Absorption in Universal Algebra and CSP

Libor Barto*[1] and Marcin Kozik†[2]

1 Department of Algebra, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic
libor.barto@gmail.com

2 Theoretical Computer Science, Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland
marcin.kozik@uj.edu.pl

Abstract

The algebraic approach to Constraint Satisfaction Problem led to many developments in both CSP and universal algebra. The notion of absorption was successfully applied on both sides of the connection. This article introduces the concept of absorption, illustrates its use in a number of basic proofs and provides an overview of the most important results obtained by using it.

1998 ACM Subject Classification G.2.1. Combinatorics

Keywords and phrases Constraint satisfaction problem, Algebraic approach, Absorption

Digital Object Identifier 10.4230/DFU.Vol7.15301.45

1 Introduction

Absorption is a simple concept, which has found several interesting applications in universal algebra and constraint satisfaction. The aim of this survey is to show *what* results have been achieved using absorption and, more importantly, to explain *how* absorption is applied to prove these results.

1.1 Results

In constraint satisfaction, absorption is mostly applied in the study of the computational and descriptive complexity of the Constraint Satisfaction Problem (CSP) over a fixed finite relational structure (also known as a *template* or a *constraint language*). In this paper, a *relational structure* $\mathbb{A} = (A; R_1, \dots, R_k)$ consists of a **finite** set A, called a *domain* or a *universe*, and a finite sequence of (finitary) relations $R_1, \dots, R_n$. A *primitive positive formula*, or *pp-formula*, over $\mathbb{A}$ is a first order formula over $\mathbb{A}$ that uses only existential quantification, conjunction, and equality. The *constraint satisfaction problem over* $\mathbb{A}$, written CSP($\mathbb{A}$), is the problem of deciding whether an input pp-sentence is true. Thus, for a relational structure $\mathbb{A}$ with a ternary relation $R \subseteq A^3$ and a binary relation $S \subseteq A^2$, an instance of CSP($\mathbb{A}$) is e.g.

$$(\exists x_1)(\exists x_2)(\exists x_3)(\exists x_4)\, R(x_1, x_3, x_2) \wedge S(x_1, x_1) \wedge S(x_1, x_4) \wedge (x_2 = x_1).$$

The clauses in the instance are often called *constraints* as they are constraining the possible values of the tuples of variables.

* L. Barto was partially supported by the Grant Agency of the Czech Republic grant 13-01832S.
† M. Kozik was partially supported by NSC grant UMO-2014/13/B/ST6/01812.

The Constraint Satisfaction Problem: Complexity and Approximability. *Dagstuhl Follow-Ups*, Volume 7, ISBN 978-3-95977-003-3.
Editors: Andrei Krokhin and Stanislav Živný; pp. 45–77
DAGSTUHL FOLLOW-UPS Dagstuhl Publishing
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Germany

Known results suggest that, for any relational structure $\mathbb{A}$, the problem CSP($\mathbb{A}$) is tractable (i.e. solvable in polynomial time), or NP-complete. The conjecture postulating this separation is known as the *CSP dichotomy conjecture* [30]. The concept of absorption allowed the confirmation of this conjecture for the CSPs over digraphs with no sources or sinks [11] and the simplification of the proof of the dichotomy theorem [3] for conservative CSPs [24] (i.e. CSPs over structures that contain all unary relations).

A closely related line of research studies the power of consistency methods in CSP. The applicability of consistency algorithms to CSPs with fixed template was determined [10] using absorption (independently in [20] using different tools). Moreover, it was shown that basic algorithms, such as $(2, 3)$-minimality [6] or Singleton Arc Consistency [42], solve all the CSPs solvable by local consistency. At the same time, the templates solvable by local consistency were proved to be exactly those with CSPs having robust approximation algorithms [9] – all these proofs are based on absorption.

A significant step towards understanding the power of "linear consistency" and characterizing the CSPs in NL has been made in [13], and a related result studying robust approximation with a polynomial loss appeared in [27] – both of these proofs rely on absorption as well.

The contributions of absorption to universal algebra mostly concern equational conditions for finite algebras. In this paper, an *algebra* $\mathbf{A} = (A; f_1, f_2, \dots)$ consists of a **finite** universe A and a set of (finitary) operations on $\mathbf{A}$, called the *basic operations* (this set sometimes needs to be indexed so that, e.g, one can define direct products). A *term operation* of $\mathbf{A}$ is any operation on A obtained by composing the basic operations. An *equational condition* stipulates the existence of term operations satisfying certain *identities*, that is, universally quantified equations (the term "equational condition" is nonstandard and used instead of a closely related, but different concept of a Mal'tsev condition). Equational conditions often characterize properties of invariant relations, for instance, the existence of a term operations m satisfying the identities $m(x, x, y) = y = m(y, x, x)$ characterizes permutability of compatible equivalences in a sense which is made precise in Theorem 13. Nontrivial information about the shape of invariant relations under some equational condition is also the core of some CSP results, such as the aforementioned dichotomy theorem for digraphs with no sources and sinks, see the discussion after Theorem 18.

Equational conditions are intimately related to the fixed template CSPs in that the complexity of CSP($\mathbb{A}$) is determined by the equational conditions satisfied by the associated algebra of polymorphisms, see Subsection 1.3 for a brief explanation and references. A chief product of absorption in this context is a characterization of the conjectured borderline [22] between tractable and NP-complete CSPs by means of cyclic operations [8]. Other contributions of absorption are concerned with new equational and relational conditions for properties that are important in CSP and/or universal algebra, including congruence distributivity (see Section 5.3), modularity [38], and meet semi-distributivity [12]. For polymorphism algebras of relational structures, a surprising collapse of equational conditions has emerged [4, 2], which also impacted some other computational problems parametrized by relational structures [19, 26]. Not so closely related to computational complexity is the connection of solvability and absorption discovered in [12] (see Theorem 35), which allowed to greatly simplify the proofs of some classical universal algebraic results.

The results on the CSP and universal algebra coming from absorption have been used in several other works, including the reduction of valued CSP to CSP in [40] (which uses cyclic operations), or further characterizations of the conjectured borderline between tractable/NP-complete CSPs in [49, 7, 8].

1.2 Why is Absorption Useful

The success of absorption is a product of three factors.

Absorption transfers connectivity. The connectivity in the slogan is meant in a wide sense, it might be strong connectivity or connectivity in a directed graph as well as any other property resembling connectivity. The most basic example of "transferring connectivity" appears already in Proposition 3. Further in Section 4.2 we display the notion of a Prague instance which can be viewed as a connectivity condition. Finally, in Section 7.2 we exhibit other properties which are transferred by absorption; these properties do not resemble connectivity – it is usually hidden in the proofs.

Connectivity is common. The reason why absorbing connectivity, or structural conditions in general, is useful is that equational conditions are often reflected in structural/connectivity properties of compatible relations (see Section 5). In the CSP, connectivity can be provided by local consistency checking algorithms running in polynomial time, and transferring it to smaller instances sometimes allows to construct a solution. Section 4 gives some examples of this phenomenon.

Absorption is common. The two factors would not be so useful if absorption was rare. Fortunately, quite mild assumptions enforce either a significant restriction on the shape of compatible relations, or an interesting absorption. This is shown in Section 6 together with some applications.

1.3 CSP and Universal Algebra

The link between the fixed template CSP and universal algebra hinges on two Galois connections: the Pol–Inv Galois connection between relational structures and algebras [32, 18] and the Mod-Id Galois connection between classes of algebras and sets of identities [16]. The first connection implies that the complexity CSP($\mathbb{A}$) depends only on a certain algebra associated to $\mathbb{A}$ [18, 36, 35], and the second one that only the equational conditions satisfied by the algebra matter [22].

We proceed to introduce definitions and results that are behind [35] and that are essential for understanding the next section. Other concepts and results are introduced when the need for them arises; the index at the end of the paper is constructed to help with such scattered definitions. For further details we refer the reader to the recent survey on CSP basics [5] or its revision [14].

A *homomorphism* between two relational structures $\mathbb{A} = (A; R_1, \dots, R_n)$ and $\mathbb{A}' = (A'; R'_1, \dots, R'_n)$ is a map from A to A' which, when computed coordinatewise, maps R_i to R'_i. The *n-th power* of relational structure $\mathbb{A} = (A; R_1, \dots, R_n)$ is $\mathbb{A}^n$ with the universe A^n and relations R'_i defined coordinatewise (i.e. a tuple of elements of $\mathbb{A}^n$ is in R'_i if they are in R_i on every coordinate). Polymorphisms of a structure are generalizations of endomorphisms. An operation $f : A^n \to A$ is a *polymorphism* of $\mathbb{A}$ if it is a homomorphism from $\mathbb{A}^n$ to $\mathbb{A}$. When f is a polymorphism of $\mathbb{A} = (A; R)$, we also say that f is *compatible* with R, or that R is invariant under f.

To every relational structure $\mathbb{A}$, we associate an algebra $\mathbf{A}$, denoted $\mathbf{A} = \mathrm{Pol}(\mathbb{A})$, on the same domain whose operations are all the polymorphisms of $\mathbb{A}$. By [18, 36, 35], a relation is pp-definable (that is, definable by a pp-formula) from $\mathbb{A}$ if and only if it is compatible with every polymorphism of $\mathbb{A}$. Since CSP($\mathbb{B}$) can be easily reduced to CSP($\mathbb{A}$) whenever $\mathbb{A}$ pp-defines $\mathbb{B}$ (ie. every relation of $\mathbb{B}$ is pp-definable from $\mathbb{A}$), it follows that $\mathbf{A}$ determines the complexity of CSP($\mathbb{A}$).

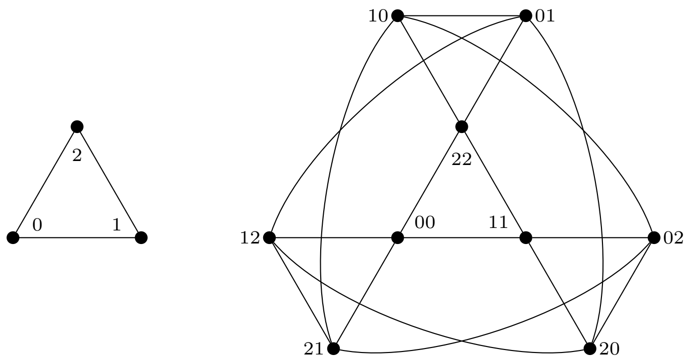


Figure 1 The underlying graph of $\mathbb{K}_3^c$ and its second power.

Given an algebra $\mathbf{A}$, a subset A' of A closed with respect to basic operations of $\mathbf{A}$ is called a *subuniverse* of $\mathbf{A}$. Such subset, if it is non empty, defines a subalgebra of $\mathbf{A}$ denoted by $\mathbf{A}' \leq \mathbf{A}$. A subuniverse is the same as a unary relation invariant under all the basic operations in $\mathbf{A}$.

2 Example

In this section, we work out an example, which illustrates several basic concepts and motivates the concept of absorption (this concept actually emerged in a quite similar context). We will show that the undirected complete graph with constants, that is, the relational structure

$$\mathbb{K}_3^c = (\{0,1,2\}; R, C_0, C_1, C_2), \quad R = \{(x,y) \in \{0,1,2\}^2 : x \neq y\}, \, C_i = \{i\}$$

has no other polymorphisms than the projections. By remarks in Subsection 1.3, this is equivalent to proving that each relation on $\{0,1,2\}$ is pp-definable from $\mathbb{K}_3^c$. In particular, any computational problem parametrized by a relational structure, whose complexity depends on the pp-definability strength of the structure, is bound to be hard over $\mathbb{K}_3^c$.

In our example, a polymorphism of arity n is a homomorphism from the n-th power of $\mathbb{K}_3^c$ to $\mathbb{K}_3^c$. Figure 1 shows the second power of the relation R; the relations corresponding to C_i in this power are $\{(ii)\}$.

2.1 One and Two Element Sets Are Preserved by Polymorphisms

The first observation is that every vertex in the power graph is mapped to an element which appears on some coordinate; in other words, each subset is a subuniverse of the polymorphism algebra $\mathrm{Pol}(\mathbb{K}_3^c)$.

In the power graph, the element $(2,\dots,2)$ is in the relation corresponding to C_2 and therefore $f : (2,\dots,2) \mapsto 2$. This shows that $\{2\}$ is a subuniverse of the polymorphism algebra; similarly, the other two singletons are subuniverses as well. The neighbors of $(2,\dots,2)$ must be mapped to neighbors of 2, but each vertex indexed by 0's and 1's is a neighbor of $(2,\dots,2)$ and thus it has to be mapped into $\{0,1\}$. That $\{0,2\}$ and $\{1,2\}$ are subuniverses is shown similarly.

A more compact way to show that $U = \{0,1\}$ is invariant under every polymorphism is by observing that this unary relation is pp-definable from $\mathbb{K}_3^c$ by the formula

$$U(x) \text{ iff } (\exists y)\, C_2(y) \wedge R(y,x) \ .$$

In words, U is the set of R-neighbors of the invariant relation C_2.

2.2 Unary and Binary Polymorphisms Are Trivial

The relational structure $\mathbb{K}_3^c$ has no endomorphisms other than the identity. Indeed, we have already observed that each unary f maps i to i.

For binary polymorphisms, consider the picture in Figure 1 and let f be any homomorphism from the second power of $\mathbb{K}_3^c$ into $\mathbb{K}_3^c$. The function f maps (i,i) to i for every i, and thus the values on the inner triangle are fixed. Choose an arbitrary vertex, say $(0,1)$. By the previous section, it can be mapped to 0 or to 1. Without loss of generality, assume it maps to 0. Then $(1,0)$, as a neighbor of $(0,1)$ and $(2,2)$ which are mapped to 0 and 2 respectively, has to map to 1. Further $(2,0)$, as a neighbor of $(0,1)$ and $(1,1)$, needs to be mapped to 2. Continuing in this way we establish that f is the first projection, and if $(0,1)$ were mapped to 1 we would obtain the second projection.

2.3 Polymorphisms of Higher Arities

Consider now an arbitrary polymorphism f of arity $n \geq 3$. We define binary operations f_i, $i \in [n]$, by

$$f_i(x,y) = f(x,\dots,x,y,x,\dots,x) \text{ with } y \text{ at the } i\text{-th place.}$$

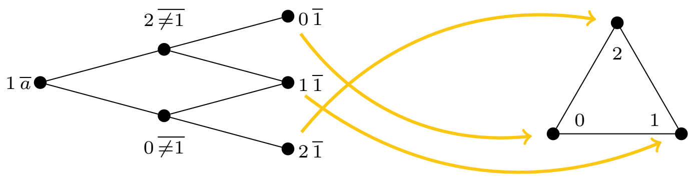
The set of polymorphisms of any relational structure is a *clone*, that is, it contains all the projections and is closed under composition. In particular, the binary operations f_i are also polymorphisms of $\mathbb{K}_3^c$. Since the only binary polymorphisms of our structure are projections, for every $i \in [n]$, either $f_i(x,y) = x$ for all $x,y \in \{0,1,2\}$, or $f_i(x,y) = y$ for all $x,y \in \{0,1,2\}$. We distinguish two cases.

(a) There exist i such that $f_i(x,y) = y$.

(b) For all i, $f_i(x,y) = x$.

2.4 Polymorphisms with $f_i(x,y) = y$ for some i

For simplicity assume $i = 1$. The reasoning is illustrated by the following figure.



Take an arbitrary tuple and, without loss of generality, assume that it has 1 on the first coordinate (tuple $1\overline{a}$ in the figure). Find a neighbour of this tuple with 2 on the first coordinate and elements different from 1 on the remaining coordinates. This element is denoted by $2\overline{\neq 1}$ and an analogous element with 0 on the first coordinate is denoted by $0\overline{\neq 1}$. Both of theses elements are adjacent to $1\,\overline{1}$ (the vertex with 1's only); the first is also adjacent to $0\,\overline{1}$ and the second to $2\,\overline{1}$.

The three elements $1\,\overline{1}, 0\,\overline{1}$, and $2\,\overline{1}$ are mapped to $1, 0$, and 2, respectively, which forces $2\overline{\neq 1} \mapsto 2$ and $0\overline{\neq 1} \mapsto 0$. This in turn forces $1\,\overline{a} \mapsto 1$, i.e. the polymorphism is the first projection.

From what we have just shown, it follows that $f_i(x,y) = y$ cannot simultaneously hold for two different i's. There is a deeper reason to it. If, say, $f_1(x,y) = f_2(x,y) = y$, then the ternary polymorphism

$$m(x, y, z) = f(x, z, y, y, \ldots, y)$$

is a Mal'tsev operation, that is, it satisfies $m(x, x, y) = y = m(y, x, x)$ for all $x, y \in \{0, 1, 2\}$. A Mal'tsev polymorphism drastically restricts the shape of relations, for instance, a binary relation invariant under a Mal'tsev operation is rectangular (see Section 5.2), which is not the case for R.

2.5 Polymorphisms with $f_i(x, y) = x$ for all i

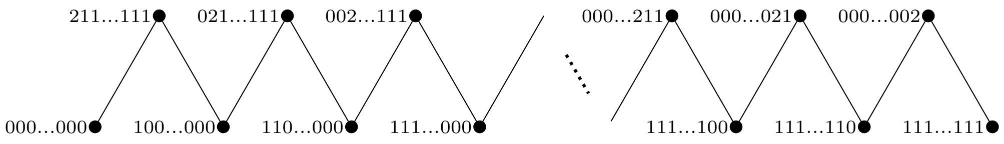
This is the most interesting part of the analysis. In this case, the polymorphism f satisfies

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

Operations satisfying these identities are called *near unanimity* (or *NU*) operations.

It is possible to derive a contradiction by an ad hoc argument (as in the previous section) by considering where various vertices of the power graph need to be mapped. We will show a nicer argument, which can be used in more general situations.

Consider the following part of the powergraph:



Every vertex in the bottom row is adjacent to $(2, \ldots, 2)$ which is mapped to 2, so bottom elements are mapped to $\{0, 1\}$ (as already observed). But, also, every vertex in the top row is adjacent to a vertex of the form $(2, \ldots, 2, 0, 2, \ldots, 2)$, which is mapped to 2 by the assumption on f. Thus all the vertices in the path are mapped into $\{0, 1\}$ and we get a path from 0 to 1 of even length – this is clearly impossible.

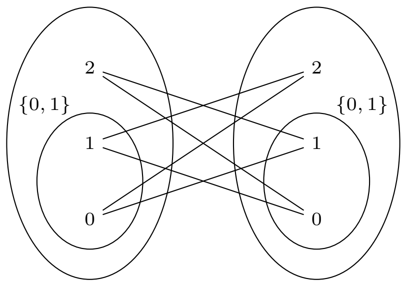
It is often useful to look at a binary relation $R \subseteq A^2$ as a bipartite graph. The partite sets are disjoint copies of A (one copy is on the left, the other one on the right) and edges correspond to pairs in R. The relation R in our example is shown in Figure 2 on the left. Note that the elements 0 and 1 (on the left) are disconnected in the subgraph induced by both copies of $\{0, 1\}$. However, the above path provides a connection from 0 to 1 (see the right part of Figure 2), a contradiction again.

Crucial for the given argument was a pleasant property of the set $\{0, 1\}$ and operation f: not only are elements consisting of 0's and 1's mapped to $\{0, 1\}$, f tolerates one exception.

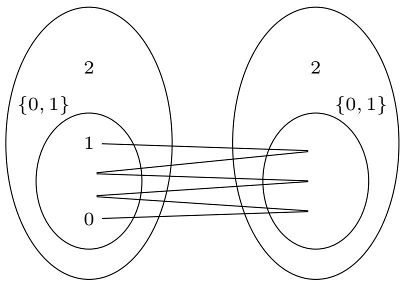
3 Absorption and More Absorption

The notion of absorption (defined below) generalizes the property of singletons and the set $\{0, 1\}$ that made the reasoning in Subsection 2.5 possible. However, before diving into the definitions we make a couple of remarks.

First, we restrict to idempotent algebras. An algebra $\mathbf{A}$ is *idempotent* if, for every operation f and all $a \in A$, $f(a, \ldots, a) = a$. Equivalently, one can require that every one-element subset of A is a subuniverse of $\mathbf{A}$. This is not a severe restriction: In the CSP, one can often restrict to the relational structures that contain the singleton unary relations; their



(a) R as a bipartite graph



(b) A connection from 0 to 1

Figure 2 Bipartite viewpoint.

polymorphism algebras are idempotent. In universal algebra, many properties of algebras depend only on certain idempotent algebras associated to them, their full idempotent reducts. We wish to stress that in all definitions and theorems we will implicitly assume that **algebras are finite and idempotent**.

Second, the operation defining absorption in an algebra is not always one of the basic operations of the algebra – it can be any term operation. Polymorphisms of any relational structure are closed under composition, so there is no difference in such a situation.

3.1 Absorption

There are in fact several useful notions of absorption: (directed) Jónsson absorption (see Section 5.3) or (directed) Gumm absorption [2, 38]. The one that appears the most useful resembles near unanimity operations.

▶ **Definition 1** (Absorption). A subalgebra $\mathbf{B}$ of $\mathbf{A}$ is *absorbing* with respect to an n-ary term operation f of $\mathbf{A}$ if $f(a_1, \ldots, a_n) \in B$ whenever the set of indices $\{i : a_i \notin B\}$ has at most one element. The fact is denoted $\mathbf{B} \trianglelefteq_f \mathbf{A}$, or $\mathbf{B} \trianglelefteq \mathbf{A}$ if f is not important. We also say that $\mathbf{B}$ absorbs $\mathbf{A}$, that f absorbs $\mathbf{A}$ into $\mathbf{B}$, and so on.

In Subsection 2.5, we have observed that $\{0, 1\}$ absorbs $\mathbf{A} = \mathrm{Pol}(\mathbb{K}_3^c)$ with respect to f. More formally, we should say that the subalgebra of $\mathrm{Pol}(\mathbb{K}_3^c)$ with universe $\{0, 1\}$ absorbs $\mathbf{A}$, but subalgebras are determined by their universes, so we can safely disregard this formal distinction when $\mathbf{A}$ is clear from the context.

Algebras with absorbing subuniverses are common. For example, most two-element algebras have proper absorbing subalgebras: It is known that if a two-element algebra contains an operation which is not affine over the two-element field, then it contains the binary minimum operation, or the binary maximum operation, or the majority (the only ternary NU operation on a two–element universe). In the first case, $\{0\}$ is absorbing; in the second case, $\{1\}$ is absorbing; and in the third case, both singletons are absorbing. These absorptions are behind the polynomial algorithm for Horn-SAT, and can be used to construct polynomial algorithms for 2-SAT as well.

In any algebra with a near–unanimity operation, every one-element subalgebra is absorbing with respect to this operation. The converse is also true, if every one element subuniverse absorbs $\mathbf{A}$, then $\mathbf{A}$ has a near unanimity term. It is not immediate, since the absorptions can be witnessed by different operations, but this problem can be fixed by composing terms in a way introduced in the next paragraph.

If $\mathbf{B} \trianglelefteq_f \mathbf{A}$ and $\mathbf{C} \trianglelefteq_g \mathbf{A}$, where f is n-ary and g is m-ary, then both absorptions are also witnessed by the *star composition* of f and g (denoted $f \star g$) which is an nm-ary operation defined by

$$f \star g(x_1, \dots, x_{nm}) = f(g(x_1, \dots, x_m), g(x_{m+1}, \dots, x_{2m}), \dots, g(x_{nm-m+1}, \dots, x_{nm})).$$

Several other simple properties of absorption can be shown using the star composition, e.g. one can prove that the relation "is an absorbing subuniverse of" is transitive, or that an intersection of absorbing subuniverses is again absorbing.

3.2 Absorption from Absorption: Propagation

This section explains how to use compatible relations, or subpowers (defined in the next paragraph), in order to propagate the property of "being an absorbing subuniverse" from one subuniverse to another. An example of such a situation appeared already in Subsection 2.5: the fact that $\{2\}$ was absorbing implied that the set $\{0, 1\}$, the set of all R-neighbors of $\{2\}$, was absorbing as well.

Before stating the general version of the property, we recall several basic definitions. The *n-th power* of $\mathbf{A}$ with universe A is the algebra with universe A^n and the operations computed coordinatewise. A *subpower* of $\mathbf{A}$ is any subuniverse (or a subalgebra) of a power of $\mathbf{A}$. In other words, a subpower of $\mathbf{A}$ is an n-ary relation invariant under coordinate-wise action of any operation of $\mathbf{A}$. When $\mathbf{A} = \mathrm{Pol}(\mathbb{A})$, then subuniverses of $\mathbf{A}$ are exactly the relations pp-definable from $\mathbb{A}$.

It is easy to prove that the set of all the subpowers of an algebra is closed under pp-definitions. The following proposition gives an analogue for absorption. The proof of the proposition is left as an exercise.

▶ **Proposition 2** (Propagation of Absorption). *Let $\mathbf{A}$ be an algebra and let $\mathbf{R} \leq \mathbf{A}^n$ be a subpower defined from subpowers $\mathbf{S}_1, \dots, \mathbf{S}_k$ by a pp-formula ϕ. Moreover, let $\mathbf{S}'_1 \trianglelefteq \mathbf{S}_1$, …, $\mathbf{S}'_k \trianglelefteq \mathbf{S}_k$.*

Then the subpower defined by the pp-formula obtained from ϕ by replacing each $S_i(\dots)$ by $S'_i(\dots)$ absorbs $\mathbf{R}$.

Note that $\mathbf{A} \trianglelefteq \mathbf{A}$ for any algebra $\mathbf{A}$, so, in the proposition, $\mathbf{S}'_i$ can be equal to $\mathbf{S}_i$.

The proposition above is often used to "walk" with absorption, a first example of the "walking" was already in Subsection 2.5. To put the construction into slightly more general terms, consider subalgebras $\mathbf{R} \leq \mathbf{A}^2$ and $\mathbf{B} \trianglelefteq \mathbf{A}$. The set C of out-neighbors of B in the directed graph with edge-set R is pp-defined by the formula

$$C(y) \text{ iff } (\exists x)\, B(x) \wedge R(x, y).$$

It absorbs the subalgebra $\mathbf{D}$ of $\mathbf{A}$ defined by

$$D(y) \text{ iff } (\exists x)\, A(x) \wedge R(x, y) \quad \text{equivalently} \quad D(y) \text{ iff } (\exists x)\, R(x, y).$$

In particular, if every $a \in A$ has an in-neighbor, then $D = A$ and we get that $\mathbf{C}$ absorbs $\mathbf{A}$. This construction will be generalized in the next section.

4 Connectivity

In this section, we will show that if a smaller subpower absorbs a bigger one, then some structural properties (like connectivity) of the bigger subpower transfer to the smaller one.

A similar situation appeared in Section 2.5: the relation R defined a connected bipartite graph, but its restriction to $\{0, 1\}$ (on both sides) was absorbing and disconnected – this contradiction concluded the proof in Section 2.

We will illustrate the slogan "absorption transfers connectivity" using two examples: First we study a single binary relation and obtain a result which will be used later to prove, e.g., the Loop Lemma (which is Theorem 18). Later we focus on a more complex example: a microstructure graph arising from an instance of a CSP.

In order to simplify the applications of Proposition 2, we will be working with subdirect subpowers. A subset R of $A_1 \times \cdots \times A_n$ is called *subdirect* if, for each i, the projection of R onto the i-th coordinate is equal to A_i.

4.1 Absorbing Linkedness

The first notion that is preserved by absorption is "linkedness". A subset $R \subseteq A^2$ is called *linked* if R is connected when regarded as a bipartite graph (exactly like in Section 2.5). Similarly, we can talk about $a, b \in A$ being linked, but, as every element of A has two copies in the bipartite graph, we need to specify whether we mean left or right a and left or right b.

The same relation $R \subseteq A^2$ can also be regarded as a directed graph and we talk about in/out-neighbors, sinks, sources, directed walks, etc. The digraph R is *smooth* if R is subdirect in A^2, in other words, R has no sources and no sinks. The *smooth part* of R is the maximal subset B of A such that $R \cap B^2$ is smooth.

Note that the linkedness (i.e. the connectivity of the bipartite graph) is equivalent to neither strong nor weak connectivity of the directed graph (but it implies the weak one). The following proposition states that the linkedness transfers to absorbing subuniverses.

▶ **Proposition 3.** *Let $\mathbf{R} \leq \mathbf{A}^2$ be subdirect, linked, and let $\mathbf{A}$ have a proper absorbing subalgebra. Then there exists $\mathbf{B}$, a proper subalgebra of $\mathbf{A}$, such that $\mathbf{R} \cap \mathbf{B}^2$ is a linked and subdirect in $\mathbf{B}^2$.*

Proof. First we look for a proper absorbing subalgebra $\mathbf{D} \trianglelefteq \mathbf{A}$ such that $R \cap D^2$ has a nonempty smooth part. This is achieved by "walking": think of $C \subseteq A$ as being on the left side of the bipartite graph, define C' as the set of all neighbors (i.e. on the right) of vertices from C, and put $(C, \text{left}) \sqsubseteq (C', \text{right})$ – this is a step from left to right. Similarly, for a step from C' on the right to C'' on the left (C'' contains all the neighbors of C'), put $(C', \text{right}) \sqsubseteq (C'', \text{left})$. Finally, close $\sqsubseteq$ under composition with itself.

Let $\mathbf{B}'$ be proper absorbing subalgebra of $\mathbf{A}$. Whenever $(B', \text{left}) \sqsubseteq (C, \text{left})$ or $(B', \text{left}) \sqsubseteq (C, \text{right})$, then C, by Proposition 2, is an absorbing subuniverse of $\mathbf{A}$. Since R is linked, $(B', \text{left}) \sqsubseteq (A, \text{left})$ and therefore there is D, say on the left, such that $(B', \text{left}) \sqsubseteq (D, \text{left})$ and by stepping to the right from D we obtain A, i.e. every vertex in the right A has a neighbor in the left D.

Looking at R as a directed graph, this property of D means that every vertex in A has an incoming edge from a vertex in D. It follows that there exists an arbitrarily long directed walk entirely in D, which immediately provides a directed cycle in the directed graph induced by R on D. Therefore, the smooth part of $R \cap D^2$, denoted by B, is nonempty.

The subuniverse D absorbs $\mathbf{A}$ by Proposition 2. Moreover, by the same proposition, the smooth part of $D \cap R^2$ (i.e. B) absorbs the smooth part of R (which is the whole A). This last fact holds since the smooth part of a directed graph can be pp-defined as the set of vertices with a directed walk of length $|A|$ from them and to them.

Finally, a generalization of the argument from Section 2.5 shows that $B \cap A^2$ is linked: Take any $a, b \in B$ (on the left) and a link $a = a_0, a_1, a_2, \ldots, a_{2k} = b$ from a to b (even members are on the left, the odd ones on the right). Consider a term operation f witnessing $\mathbf{B} \trianglelefteq \mathbf{A}$. Then, for any i, the sequence

$$f(\underbrace{a,\dots,a}_{(i-1)\times}, a_0, b, \dots, b),\ f(a,\dots,a,a_1,b,\dots,b),\ \dots,\ f(a,\dots,a,a_{2k},b,\dots,b)$$

provides a link from

$$f(\underbrace{a,\dots,a}_{i\times}, b, \dots, b) \text{ to } f(\underbrace{a,\dots,a}_{(i-1)\times}, b, \dots, b)$$

which lies fully in B. By concatenating these links we get a link in B from $a = f(a,\dots,a)$ to $b = f(b,\dots,b)$. ◀

The proof above exhibits a structure common to almost all the proofs using absorption. It splits into two stages:

- **Walking stage** finds a substructure which is "subdirect" and "absorbing" (here finds $\mathbf{B}$ absorbing $\mathbf{A}$ such that the restriction of $\mathbf{R}$ to $\mathbf{B}^2$ is subdirect in $\mathbf{B}^2$).
- **Reducing stage** uses absorption to transfer an additional property (here linkedness of $\mathbf{R}$ is transferred to the restriction of $\mathbf{R}$ by $\mathbf{B}^2$).

In the remaining part of the paper, we will see more proofs following this pattern. Here we present a corollary which is an easy consequence of the proposition above.

▶ **Corollary 4.** *Let* $\mathbf{R}$ *be a subdirect linked subalgebra of* $\mathbf{A}^2$ *and assume that* every non–singleton subalgebra of $\mathbf{A}$ *(including* $\mathbf{A}$ *itself) has a proper absorbing subalgebra. Then* $\mathbf{R}$ *contains a constant pair.*

Proof. The corollary is proved by a repeated application of Proposition 3. Indeed, after a first application of Proposition 3 to $\mathbf{R}$ and $\mathbf{A}$, we obtain $\mathbf{B}$ and if $|B| = 1$, we have a constant tuple in $\mathbf{R}$. Otherwise, $\mathbf{R} \cap \mathbf{B}^2$ is linked and subdirect in $\mathbf{B}$ and, as $\mathbf{B}$ has a proper absorbing subuniverse, we can apply Proposition 3 again. In a finite number of steps, we arrive at a one–element B, which finishes the proof. ◀

The structure of this proof is also typical, most of the proofs using absorption perform a sequence of reductions decreasing the sizes of underlying algebras until each one has only one element.

Note that algebras with a near-unanimity term, or algebras with a semilattice term, or products of such algebras all satisfy the assumptions of the corollary.

4.2 Connectivity in CSP

The results presented in this section are parts of a proof of the bounded width conjecture of Feder and Vardi [30]. This conjecture, its motivation and resolution, are discussed in more detail in Section 7. In here we focus on a single obstacle, which had to be overcome in order for the proof to work. The obstacle can be phrased as follows: is there a single consistency algorithm solving the CSP over all the binary templates with near-unanimity polymorphisms? We ought to note that already Feder and Vardi [30] showed that near-unanimity templates can be solved in polynomial time. However, their algorithm depends on the arity of the near-unanimity term and therefore does not reach our goal.

We move on to a list of consistency notions in search of a consistency notion that will be transferred by absorption the way the linkedness was transferred in Proposition 3. We start, however, with a consistency notion which plays the role played by subdirectness in Proposition 3 – it is the most basic and important consistency notion, the *arc consistency.*

4.2.1 Arc Consistency of a CSP Instance

In practical applications, arc consistency is often used to quickly disqualify some of the instances with no solutions. Unfortunately, a rather strong structure of the template is needed [30, 28] for arc consistency to solve the associated CSP.

We say that an instance with a variable set V is arc consistent with $\{P_x\}_{x \in V}$ if for every constraint $R(x_1, \dots, x_n)$, the relation R is subdirect in $P_{x_1} \times \dots \times P_{x_n}$. The following algorithm turns an arbitrary instance into an arc consistent instance with the same set of solutions.

```
for every variable $x$ do $P_x := A$
repeat
    for every constraint $R(x_1, \dots, x_n)$ do
        let $R' := R \cap \prod_i P_{x_i}$
        for $i = 1$ to $n$ do $P_{x_i} := P_{x_i} \cap \mathrm{proj}_i R'$
        substitute constraint $R(x_1, \dots, x_n)$ with $R'(x_1, \dots, x_n)$
    end for
until none of the $P_x$'s changed
```

It is clear that the output instance is arc consistent and has the same set of solutions as the input instance. The AC algorithm *derives a contradiction* if at least one of the sets P_x is empty; this means that the algorithm correctly detected an unsolvable instance.

Note that all the sets P_x as well as the new relations in the output instance are pp-definable from the relations in the original instance, therefore all the polymorphisms of the template are compatible with the new instance. In other words, if $\mathbb{A}$ is a template and $\mathbf{A} = \mathrm{Pol}(\mathbb{A})$ the associated algebra, then every P_x is a subuniverse of $\mathbf{A}$ and determines a subalgebra $\mathbf{P}_x$ of $\mathbf{A}$.

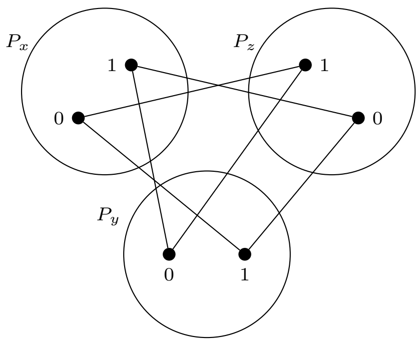
Arc consistency solves a CSP over $\mathbb{A}$ if it derives a contradiction on every unsolvable instance over $\mathbb{A}$. Such CSPs are said to have *width 1* and include the CSP over the template $\mathbb{A} = (\{0,1\}; C_0, C_1, \leq)$, which is essentially the problem of finding a directed path in a directed graph, or over $\mathbb{A} = (\{0,1\}; \{0,1\}^3 \setminus \{(1,1,0)\}, \{0,1\}^3 \setminus \{(1,1,1\})$, which is Horn-3-SAT.

A simple example of an instance where the arc consistency algorithm fails to detect a problem uses the template $\mathbb{A} = (\{0,1\}; \neq)$, whose CSP is essentially 2-colorability, and the instance

$$(\exists x)(\exists y)(\exists z)\, x \neq y \wedge y \neq z \wedge z \neq x,$$

which corresponds to the triangle. The arc consistency algorithm on this instance does not update any constraints and outputs the original instance with the sets $P_x = P_y = P_z = \{0,1\}$.

The following picture shows the problematic instance as a multipartite graph called the *microstructure graph* of the instance: the graph has a copy of P_x for every variable x and the edges between P_x and P_y are given by a constraint $R(x,y)$.



Such a graph is well defined if all the constraints are binary, and every pair of variables appears in at most one constraint. This is not a very restrictive condition and we discuss such instances in the next section.

Note that the template $\mathbb{A} = (\{0,1\}; \neq)$ has a majority polymorphism, and therefore arc consistency fails to work for binary templates with near unanimity polymorphisms. In order to answer the questions posted in the section above we need to work with a different consistency notion.

4.2.2 (2,3)-Consistent, Simplified Instances

In order to simplify presentation, we impose the following restrictions on CSP instances:

- all the constraints are binary and
- for every two distinct variables x, y, there is a unique constraint $P_{xy}(x,y)$ and $P_{xy} = P_{yx}^{-1}$.

We call an instance *simplified* if these conditions are satisfied. We note that, by an appropriate preprocessing, every instance can be turned into a simplified instance on a, possibly different, template with the same complexity of the associated CSP.

The arc consistency algorithm, over a simplified instance, finds algebras $\mathbf{P}_x$ and restricts the constraints so that $\mathbf{P}_{xy}$ is subdirect in $\mathbf{P}_x \times \mathbf{P}_y$. If arc consistency fails to solve a particular CSP($\mathbb{A}$) (as was the case in the example in the previous section) we may try to solve the problem by enforcing a stronger form of consistency. The next, after arc consistency, standard consistency notion is the $(2,3)$-consistency, also known as path consistency.

▶ **Definition 5.** A simplified instance is $(2,3)$-*consistent* if it is arc consistent and for every pairwise different variables x, y, z and any $(a,b) \in P_{xy}$ there exists $c \in P_z$ such that $(a,c) \in P_{xz}$ and $(b,c) \in P_{yz}$.

Both arc consistency and $(2,3)$-consistency have simple interpretations in the microstructure graphs of simplified instances: arc-consistency means that every vertex is adjacent to some vertex in every other partite set, $(2,3)$-consistency further ensures that, for any pairwise different x, y, z, every edge in $P_x \cup P_y$ (or, more precisely, in the union of the disjoint copies of P_x and P_y in the microstructure graph) extends to a triangle in $P_x \cup P_y \cup P_z$.

Already Feder and Vardi [30] noted that, over a template with majority polymorphism, every simplified $(2,3)$-consistent instance has a solution. The reasoning, however, did not extend to near unanimity operations of higher arities.

At present, we know that the $(2,3)$-consistency is transferred by absorption in the same way the linkedness is transferred in Proposition 3:

▶ **Proposition 6.** *Take a $(2,3)$-consistent simplified instance such that at least one $\mathbf{P}_x$ has a proper absorbing subuniverse. Then there exist $\mathbf{P}'_x \trianglelefteq \mathbf{P}_x$ (at least one proper) and $\mathbf{P}'_{xy} \trianglelefteq \mathbf{P}_{xy}$ which form a $(2,3)$-consistent instance.*

This would finish our search for a consistency notion transferred by absorption, except for the fact that more involved tools are required in order to prove this proposition. These tools are presented and discussed in Section 7, while here we continue the search for a consistency notion for which an analogue of Proposition 3 can be proved directly.

4.2.3 Prague Instances

In this section, we study notions weaker than $(2,3)$-consistency. A concept underlying all the definitions in this section is the notion of a pattern. In a simplified instance, *a pattern p* is a sequence of variables $(x_1, x_2, \ldots, x_{k-1}, x_k)$. If the first and the last variable of a pattern coincide, we call it a *circle*. An element a is connected to b by $p = (x_1, x_2, \dots, x_{k-1}, x_k)$ if there exists $(a = a_1, a_2, \dots, a_{k-1}, a_k = b)$ such that $(a_i, a_{i+1}) \in P_{x_i x_{i+1}}$ if $x_i \neq x_{i+1}$, and $a_i = a_{i+1}$ otherwise. We write $p + q$ for the concatenation of patterns, and kp for the concatenation of k copies of p.

The following notion of consistency is a first approximation to the notion of a Prague instance:

▶ **Definition 7.** A simplified instance is a *circle instance* if it is arc-consistent and for every pattern $p = (x = x_1, x_2, \dots, x_k = x)$, every $a \in P_x$ is connected to itself by p.

Unfortunately, the notion suffers from the same problem as $(2, 3)$-consistency.

One can state a result transferring the consistency:

▶ **Proposition 8.** *Take a simplified circle instance such that at least one* $\mathbf{P}_x$ *has a proper absorbing subuniverse. Then there exist* $\mathbf{P}'_x \trianglelefteq \mathbf{P}_x$ *(at least one proper) which together with* $P'_{xy} = P_{xy} \cap (P'_x \times P'_y)$ *form a simplified circle instance.*

The proof, yet again, requires tools from Section 7.

In order to introduce the final consistency notion of this section, we need one more definition. In a simplified instance, $a, b \in P_x$ are connected *in the set of variables* I (x needs to belong to I) if they are connected by a pattern with all the variables in I. Equivalently, we can restrict the microstructure graph to P_y's with $y \in I$ and ask for the usual undirected connectivity of vertices. We say that a, b are connected *in the variables of* p if they are connected in I, where the set I consists of the variables that appear in p.

Finally, we can define the notion of a Prague instance, a consistency notion for which an analogue of Propositions 3, 6 or 8 can be shown directly.

▶ **Definition 9.** A simplified instance is a *Prague instance* if it is arc consistent and for every circle pattern at x and every $a, b \in P_x$, the vertices a, b are connected by kp, for some natural number k, whenever they are connected in the variables of p.

It is left as an exercise for the reader to prove that every $(2, 3)$-consistent instance is a circle and a Prague instance. Moreover, the number k, in the definition of Prague instance, can be chosen to depend only on p and not on a, b.

Note that the example in Section 4.2.1, although arc consistent, is neither a circle nor a Prague instance. Indeed, $0, 1 \in P_x$ are connected in $\{x, y, z\}$ but not connected by any power of (x, y, x, z, x) (nor any power of (x, y, z, x, y, z, x)) and therefore does not contradict the following proposition (which is an analogue of Proposition 3).

▶ **Proposition 10.** *Take a simplified Prague instance such that at least one* $\mathbf{P}_x$ *has a proper absorbing subuniverse. Then there exist* $\mathbf{P}'_x \trianglelefteq \mathbf{P}_x$ *(at least one proper) which together with* $P'_{xy} = P_{xy} \cap P'_x \times P'_y$ *form a simplified Prague instance.*

Proof. As usual, the proof splits into two stages. In the walking stage, we find $\mathbf{P}'_x$'s such that

- $\mathbf{P}'_x \trianglelefteq \mathbf{P}_x$ for every x,
- for some x, the algebra $\mathbf{P}'_x$ is a proper subalgebra of $\mathbf{P}_x$, and
- putting $P'_{xy} = P_{xy} \cap P_x \times P_y$ produces an arc consistent instance.

In the reduction stage, we will show that this instance is a Prague instance.

The walking stage is similar to the one in the proof of Proposition 3: we put $(B, x) \sqsubseteq (C, y)$ whenever the set C consists of elements of P_y which have a neighbor in B in the bipartite graph P_{xy} (B is on the left, while C on the right). Yet again, we close $\sqsubseteq$ under composition with itself and disregard the pairs with the full sets P_x, i.e. the pairs of the form (x, P_x).

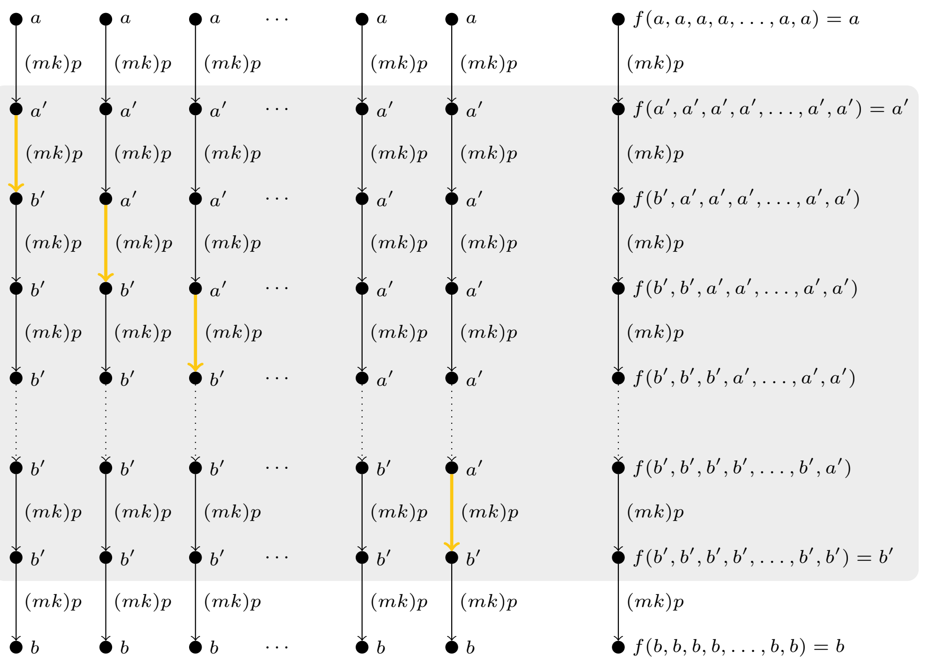
Let $\mathbf{B}_0$ be a proper absorbing subuniverse of $\mathbf{P}_{x_0}$. We walk, as far as we can, from (B_0, x_0) to end up in a maximal strong component of $\sqsubseteq$ (which may contain a single pair). We denote the set of all the pairs in this component by $\mathcal{P}$. Note that if $(B, x) \in \mathcal{P}$ and C consists of neighbors of B in P_{xy} (from left to right), then C is either P_y, or the pair (C, y) belongs to $\mathcal{P}$.

If (B, x) and (B', x) are in $\mathcal{P}$, then $(B, x) \sqsubseteq (B', x) \sqsubseteq (B, x)$. Let p denote the pattern describing the walk witnessing $(B, x) \sqsubseteq (B', x) \sqsubseteq (B, x)$. Pick $a \in B' \setminus B$ and $b \in B$ such that a is reachable from b by the appropriate initial part of p. But then b is connected to a in the vertices of p but, as $a \notin B$, not by any kp, a contradiction.

Thus, for a given x, there is at most one pair $(B, x) \in \mathcal{P}$ and we let $P'_x = B$ in such a case. If there is no such a pair, we put $P'_x = P_x$. Each set P'_x absorbs $\mathbf{P}_x$ (exactly like in the proof of Proposition 3), at least one of them is proper (actually all that arise from $\mathcal{P}$ are proper), and they define an arc consistent instance. That last property follows from the choice of $\mathcal{P}$ as the maximal strong component. We are done with the walking stage and proceed to the reducing stage.

Let a be in $\mathbf{P}'_x$ and p be a pattern such that $b \in \mathbf{P}'_x$ is connected to a in the variables of p. Find m and a', b' such that a' is reachable from itself and from a by mp in the new instance; and, similarly, from b' one can reach b' and b by mp also in the new instance. Since a' and b' are connected in the variables of p in the original instance, we get k such that b' is reachable from a' in by $(mk)p$ in the original instance.

Now we take a term operation f witnessing the absorptions $P'_x \trianglelefteq \mathbf{P}_x$ and apply it as shown in the following picture. The black arrows are realizations of $(mk)p$ in the new instance, and the yellow arrows are realizations of the same pattern in the original instance. On the right-hand side of the picture is the result of the pointwise application of f to the realizations of $(mk)p$ and the grey part indicates where absorption is used to ensure that the resulting elements are in the new instance.


a a a $\cdots$ a a
$f(a, a, a, a, \ldots, a, a) = a$
$(mk)p$
a' a' a' $\cdots$ a' a'
$f(a', a', a', a', \ldots, a', a') = a'$
b' a' a' $\cdots$ a' a'
$f(b', a', a', a', \ldots, a', a')$
b' b' a' $\cdots$ a' a'
$f(b', b', a', a', \ldots, a', a')$
b' b' b' $\cdots$ a' a'
$f(b', b', b', a', \ldots, a', a')$
b' b' b' $\cdots$ b' a'
$f(b', b', b', b', \ldots, b', a')$
b' b' b' $\cdots$ b' b'
$f(b', b', b', b', \ldots, b', b') = b'$
b b b $\cdots$ b b
$f(b, b, b, b, \ldots, b, b) = b$

Thus b can be reached from a by a sufficiently large multiple of p in the new instance. This finishes the proof of the reduction. ◀

The following corollary, which is an analogue of Corollary 4, immediately follows from the previous proposition.

▶ **Corollary 11.** *Take a simplified Prague instance. If every non–singleton subalgebra of every $\mathbf{P}_x$ (including $\mathbf{P}_x$ itself) has a proper absorbing subuniverse, then the instance has a solution.*

The corollary gives a polynomial time algorithm for CSPs over templates with a near-unanimity polymorphism, semilattice polymorphism, etc. using a single consistency notion: the notion of a Prague instance. This settles the question posed at the beginning of Section 4.2.

5 Equational Descriptions

In this section, we present several results showing how equational conditions impact properties of invariant relations. In fact, equational conditions influence invariant relations of all algebras in a *variety* rather than an individual algebra. We start by defining this concept.

An equivalence on the universe of an algebra $\mathbf{A}$ is a *congruence*

if it is invariant, as a binary relation, under the operations in $\mathbf{A}$; in other words, it is a subpower of $\mathbf{A}$. An algebra $\mathbf{A}$ can be factored modulo its congruence α to obtain a *quotient* $\mathbf{A}/\alpha$: the compatibility of $\mathbf{A}$ with α ensures that operations can be defined using arbitrarily chosen representatives. A *variety generated by an algebra* $\mathbf{A}$, denoted $\mathcal{V}(\mathbf{A})$, is the smallest class of algebras containing $\mathbf{A}$ and closed under taking powers, subalgebras, and quotients (and isomorphic copies).

Varieties provide the second step in the algebraic approach to the CSP: from the algebra $\mathbf{A} = \mathrm{Pol}(\mathbb{A})$ to the variety generated by $\mathbf{A}$. This step is meaningful as every relational structure $\mathbb{B}$ compatible with an (as always finite) algebra $\mathbf{B} \in \mathcal{V}(\mathbf{A})$ defines a CSP not harder than CSP($\mathbf{A}$) [22]. The "not harder" statement can be understood as an existence of a LOGSPACE reduction from CSP($\mathbb{B}$) to CSP($\mathbb{A}$), but the connection between $\mathbb{A}$ and $\mathbb{B}$ is much closer: many structural properties are transferred from $\mathbb{A}$ to $\mathbb{B}$.

We can talk about $\mathbb{B}$'s, compatible with algebras in the variety generated by $\mathbf{A}$, totally bypassing the algebraic nomenclature and using pp-interpretations instead [17]. A relational structure $\mathbb{B} = (B, S_1, \ldots, S_m)$ is *pp-interpretable* in $\mathbb{A}$ if there are

- a relation $R' \subseteq A^n$ and an equivalence α on R', both pp-definable[1] in $\mathbb{A}$;
- relations $S'_1, \ldots, S'_m$ on R'/α also pp-definable[1] in $\mathbb{A}$

such that $\mathbb{B}$ and $(R'/\alpha, S'_1, \ldots, S'_m)$ are isomorphic. It is easy to see that a relational structure is pp-interpretable in $\mathbb{A}$ if and only if it compatible with an algebra in $\mathcal{V}(\mathrm{Pol}(\mathbb{A}))$.

A part of the Mod–Id Galois connection gives a link between identities and varieties [16]: An algebra $\mathbf{B}$ (of the same signature as $\mathbf{A}$) is in $\mathcal{V}(\mathbf{A})$ if and only if $\mathbf{B}$ satisfies all the identities satisfied by $\mathbf{A}$, equivalently, by all members of $\mathcal{V}(\mathbf{A})$.

The identities true in a variety are closely connected to particular members of the variety: the free algebras. The *free algebra* in $\mathcal{V}(\mathbf{A})$ over n generators can be described as the subalgebra of $\mathbf{A}^{A^n}$ whose universe is the set of n-ary term operations of $\mathbf{A}$. For a k-ary

[1] The relation α is viewed here as a binary relation on R' and a $(2n)$-ary relation over A. The relations S'_i are k_i-ary over R' or $(k_i n)$-ary over A and independent of the choice of representatives for an α-class on any coordinate.

basic or term operation f of $\mathbf{A}$, the corresponding operation f in the free algebra acts as composition: for any $g_1, \dots, g_k$ in the free algebra, we have

$$f(g_1, \dots, g_k) : (x_1, \dots, x_n) \mapsto f(g_1(x_1, \dots, x_k), g_2(x_1, \dots, x_k), \dots, g_k(x_1, \dots, x_k))$$

The free algebra is the smallest (with respect to inclusion) subalgebra of $\mathbf{A}^{A^n}$ containing the projections $\pi_1, \pi_2, \dots, \pi_n$ (where $\pi_i : (x_1, \dots, x_n) \mapsto x_i$). We also say that the free algebra is *generated by* the projections. Very often, important properties of algebras are determined by the structure of subpowers of $\mathbf{F}$, in particular, smooth digraphs on the free algebra on two elements are used in Sections 5.2 and 6.3. Free algebras are also behind both the Pol–Inv and Mod–Id Galois connections.

Looking from the relational side, note that if $\mathbf{A}$ is the algebra of polymorphisms of a relational structure $\mathbb{A}$, i.e. $\mathbf{A} = \mathrm{Pol}(\mathbb{A})$, then the universe of the n-generated free algebra in $\mathcal{V}(\mathbf{A})$ is the set of n-ary polymorphisms of $\mathbb{A}$.

5.1 Decomposable Relations and near Unanimity

The near unanimity operations were among the most prominent operations in the previous sections. One of the relational descriptions is by means of decomposable relations, which appear naturally in the study of CSP (comp. [30]).

A k-decomposition of an n-ary relation R over A is another n-ary relation R_k over A defined by:

$$(a_1, \dots, a_n) \in R_k \text{ if } \big(\text{for all } 1 \leq j_1 < \dots < j_k \leq n \text{ tuple } (a_{j_1}, \dots, a_{j_k}) \in \mathrm{proj}_{\{j_1, \dots, j_k\}}(R)\big)$$

where $\mathrm{proj}_{\{j_1, \dots, j_k\}}(R)$ is the k-ary relation obtained by taking elements of R and selecting only the coordinates $j_1, \dots, j_k$. Clearly, $R \subseteq R_k$ and the relation R is called *k-decomposable* if $R_k = R$. Note that, for example, 2-decomposability of all the relations in the template of a CSP allows a trivial transformation of every instance to an equivalent simplified instance (it suffices to take the binary projections of constraints and intersect them if necessary).

The Baker–Pixley theorem [1] below provides identities equivalent to decomposability of all relations in a variety.

▶ **Theorem 12.** *For an algebra $\mathbf{A}$ the following are equivalent:*
1. *the algebra $\mathbf{A}$ has a near-unanimity term operation of arity $k+1$;*
2. *for every $\mathbf{B} \in \mathcal{V}(\mathbf{A})$, every subpower of $\mathbf{B}$ is k-decomposable.*

Proof. To prove item 1 from 2 we consider the free algebra $\mathbf{F}$ for $\mathbf{A}$ over 2 generators and let $\mathbf{R} \leq \mathbf{F}^{k+1}$ be generated by the tuples which are π_1 on all coordinates except for one where they are π_2.

As $\mathbf{R}$ is the smallest subalgebra of $\mathbf{F}^{k+1}$ containing the generators, its elements are obtained by applying the term operations of $\mathbf{F}$ to the generators. More formally,

$$\begin{aligned}
R &= \{f\big((\pi_1, \dots, \pi_1, \pi_2), (\pi_1, \dots, \pi_1, \pi_2, \pi_1), \dots, (\pi_2, \pi_1, \dots, \pi_1)\big) : \\
&\qquad\qquad\qquad\qquad f \text{ is a } k+1\text{-ary term op.}\} \\
&= \{\big(f(\pi_1, \dots, \pi_1, \pi_2), \dots, f(\pi_1, \pi_2, \pi_1, \dots, \pi_1), f(\pi_2, \pi_1, \dots, \pi_1)\big) : \\
&\qquad\qquad\qquad\qquad f \text{ as above}\} \\
&= \{\big((x,y) \mapsto f(x, \dots, x, y), \dots, (x,y) \mapsto f(x, y, x, \dots, x), (x,y) \mapsto f(y, x, \dots, x)\big) : \\
&\qquad\qquad\qquad\qquad f \text{ as above}\}
\end{aligned}$$

To simplify the notation, we will write $f(x,\dots,x,y)$ instead of $(x,y) \mapsto f(x,\dots,x,y)$, so that the generators are $(x,\dots,x,y),(x,\dots,x,y,x),\dots,(y,x,\dots,x)$ and

$$R = \{\big(f(x,\dots,x,y),\dots,f(x,y,x\dots,x),f(y,x,\dots,x)\big) : f \text{ is } (k+1)\text{-ary term op. of } \mathbf{A}\}.$$

Since R is k-decomposable, the tuple $(x,\dots,x)$ is in R. The description of the elements of $\mathbf{R}$ implies that there is a $(k+1)$-ary term f such that

$$\big(f(x,\dots,x,y),\dots,f(x,y,x\dots,x),f(y,x,\dots,x)\big) = (x,\dots,x).$$

This f is clearly a near unanimity operation.

It remains to show that if an algebra has a $(k+1)$-ary near unanimity term operation, denote it by f, then all the subpowers of any $\mathbf{B} \in \mathcal{V}(\mathbf{A})$ are k-decomposable.

Let $\mathbf{R} \leq \mathbf{B}^{(k+1)}$ and let R_k be the k-decomposition of R. To show that $R_k \subseteq R$, we take an arbitrary tuple $(a_1,\dots,a_{k+1})$ from R_k and will show that it belongs to R. The structure of R_k implies that for every i there is a tuple in R which differs from $(a_1,\dots,a_{k+1})$ only on the i-th coordinate. Applying the near-unanimity operation f to such tuples we get

$$f\big((?,a_2,\dots,a_{k+1}),(a_1,?,a_2,\dots,a_{k+1}),\dots,(a_1,\dots,a_k,?)\big) = (a_1,\dots,a_{k+1}).$$

This shows that $R_k = R$, i.e. that R is k-decomposable. For relations of higher arity, the reasoning is similar and we conclude that all the subpowers of $\mathbf{B}$ are k-decomposable. ◀

Note that the algebras in the statement of the theorem can be equivalently characterized as algebras with every one-element subuniverse absorbing (comp. Section 3).

In case that $\mathbf{A} = \mathrm{Pol}(\mathbb{A})$, item 1 says that $\mathbb{A}$ has a near unanimity polymorphism and item 2 is equivalent to the following statement: for every $\mathbb{B} = (B;R)$ pp-interpretable in $\mathbb{A}$, the relation R is k-decomposable.

We proceed to studying other classes of algebras. Every class will be defined by identities and posses a structural counterpart (playing a role similar to the one played by decomposability for near unanimity algebras).

5.2 Rectangular Relations and Mal'tsev Term

Rectangularity is another natural property of relations. A subset R of $B \times C$ is called *rectangular* if it is a disjoint union of products of the form $B' \times C'$, where $B' \subseteq B$ and $C' \subseteq C$. Equivalently R is rectangular if, when regarded as a bipartite graph, it is a disjoint union of bicliques, which means that every two linked elements $b \in B$ and $c \in C$ are adjacent. The following theorem [45] is a counterpart of Theorem 12 for rectangular relations.

▶ **Theorem 13.** *For an algebra* $\mathbf{A}$ *the following are equivalent:*

1. *the algebra* $\mathbf{A}$ *has a Maltsev term operation i.e.* f *such that*

$$f(x,x,y) = f(y,x,x) = y\,;$$

2. *for any* $\mathbf{B} \in \mathcal{V}(\mathbf{A})$, *any* $R \leq \mathbf{B}^2$ *is rectangular.*

Proof. In order to prove the implication from 2 to 1, we proceed exactly like in the proof of Theorem 12: We choose $\mathbf{R}$ to be the subalgebra of $\mathbf{F}^2$ ($\mathbf{F}$ is still the free algebra on two generators) generated by $(\pi_1,\pi_2),(\pi_1,\pi_1)$ and (π_2,π_1) also denoted as $(x,y),(x,x),(y,x)$. The relation $\mathbf{R}$ is rectangular and thus includes the pair (y,y). This implies that we have a ternary term operation which generates this pair in $\mathbf{R}$, i.e.

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

This operation is clearly the required Mal'tsev operation.

For the other implication, take any $\mathbf{R} \leq \mathbf{B} \times \mathbf{C}$ and let $(a,b),(a',b),(a',b') \in R$. Then $f((a,b),(a',b),(a',b')) = (a,b') \in R$ which proves rectangularity. ◀

A more familiar form of the second item is the following.

3. for any $\mathbf{B} \in \mathcal{V}(\mathbf{A})$ and α, β congruences on $\mathbf{B}$ the α and β permute, i.e. $\alpha \circ \beta = \beta \circ \alpha$.

We leave it as an exercise to the reader to show that this additional property is equivalent to item 2. A hint: to prove that 3 implies 2, use the kernels of the projections of R onto the two coordinates.

For relational structures associated to algebras with a near unanimity operation, the tractability of CSP was provided by [30] or by Corollary 11. The relational structures with associated Mal'tsev algebras define tractable CSPs as well [21], but the algorithm is beyond the scope of this article.

5.3 Congruence Distributivity

In this section, we describe a class of algebras that significantly benefited from absorption. It also motivated a study of weaker forms of absorption (comp. Section 5.3.2) which, in many cases, allow for stronger version of theorems. For instance, in Proposition 10, the standard absorption can be substituted with any of the weaker forms from Section 5.3.2.

In order to define the class, we need to introduce a new notion: For equivalences α, β on a set A,

- the smallest equivalence containing α and β is denoted by $\alpha \vee \beta$ and
- by $\alpha \wedge \beta$ we denote the intersection of α and β.

We say that an algebra $\mathbf{A}$ generates a *congruence distributive variety* (CD variety), if for every $\mathbf{B} \in \mathcal{V}(\mathbf{A})$, and every α, β, γ congruences on $\mathbf{B}$, we have

$$\alpha \wedge (\beta \vee \gamma) = (\alpha \wedge \beta) \vee (\alpha \wedge \gamma).$$

Note that equivalence on the right side is always contained in the one on the left, so only one of the inclusions bears consequences.

To illustrate the connection between distributivity of congruences and the structure of relations, we consider $R \subseteq B \times C \times D$ and assume that the projection kernels η_B, η_C, and η_D, satisfy the distributive law, i.e.

$$\eta_B \wedge (\eta_C \vee \eta_D) \subseteq (\eta_B \wedge \eta_C) \vee (\eta_B \wedge \eta_D).$$

Two triples (b,c,d) and (b',c',d') are equivalent modulo the left equivalence, if they are equivalent modulo η_B and modulo $\eta_C \vee \eta_D$. The former simply means that $b = b'$, while the latter takes place exactly when there is a sequence $(b,c,d) = (b_1,c_1,d_1)$, (b_2,c_1,d_2), (b_3,c_2,d_2), (b_4,c_2,d_3), $\ldots$, $(b_{2k-1},c_k,d_k) = (b',c',d')$; put otherwise, c and c' are linked in the projection of R to the second and third coordinates.

The two triples are equivalent modulo the right side if and only if such a sequence exists with $b_1 = b_2 = \cdots = b_{2k-1} = b$. To give this description a more lucid form, we regard R as a B-labeled bipartite graph with partitions C and D – a triple (b,c,d) corresponds to an edge (c,d) labeled by b (so the edges can have multiple labels). Now the inclusion can be interpreted in the following way: If $c, c' \in C$ are incident to a b-labeled edge and they are linked, then they are linked by b-labeled edges.

The identities characterizing congruence distributivity are derived from the connectivity property of a subpower of a free algebra, by a proof similar to the proofs in Sections 5.1

and 5.2: Let $\mathbf{F}$ denote the 2–generated free algebra for $\mathbf{A}$ and let $\mathbf{R}$ be the subalgebra of $\mathbf{F}^3$ generated by

$$(x,x,x),(y,x,y),(x,y,y)$$

which can be described as

$$R = \{((p(x,y,x),p(x,x,y),p(x,y,y)) : p \text{ is a ternary term operation of } \mathbf{A}\}$$

Since $\mathbf{A}$ generates a CD variety, the relation R satisfies the connectivity property discussed above.

The vertices x and y are incident to x-labeled edges (namely (x,x) and (y,y) coming from the generators of the algebra) and they are linked. Therefore, they must be linked by x-labeled edges. This produces a sequence $(x, x = b_1, c_1), (x, b_2, c_1), (x, b_2, c_2), (x, b_3, c_2), \ldots, (x, b_n = y, c_{n-1})$, and the ternary term operations $p_1, \ldots, p_{2n-1}$ generating this sequence satisfy

$$\begin{aligned} x &= p_1(x,x,y) && \\ p_i(x,y,y) &= p_{i+1}(x,y,y) && \text{for odd } 1 \le i \le 2n-3 \\ p_i(x,x,y) &= p_{i+1}(x,x,y) && \text{for even } 1 \le i \le 2n-2 \\ p_i(x,y,x) &= x && \text{for all } 1 \le i \le 2n-1 \qquad (\ddagger) \\ p_{2n-1}(x,x,y) &= y. && \end{aligned}$$

Term operations satisfying such identities are called *Jónsson terms*, and the following theorem [37] states an equivalence in the spirit of Theorems 12 and 13.

▶ **Theorem 14.** *The following are equivalent for an algebra* $\mathbf{A}$.
1. $\mathbf{A}$ *has Jónsson terms.*
2. *For each subalgebra* $\mathbf{R}$ *of* $\mathbf{B} \times \mathbf{C} \times \mathbf{D}$ *regarded as a* B*-labeled bipartite graph as above and for each* $b \in B$, $c, c' \in C$,
   - *if* $c, c' \in C$ *are incident to a* b*-labeled edge and*
   - *they are linked,*

   then they are linked by b*-labeled edges.*
3. *Any three congruences* α, β, *and* γ *of any algebra in* $\mathcal{V}(\mathbf{A})$ *satisfy*

$$\alpha \wedge (\beta \vee \gamma) = (\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$$

5.3.1 Near Unanimity and Directed Jónsson Terms

Near unanimity terms are stronger than Jónsson terms. This can be seen in the structure of subpowers i.e. one can prove item 2 of Theorem 14 using near unanimity operations; or by a direct syntactic argument as follows.

Let $\mathbf{A}$ be an algebra with an n-ary near unanimity term operation f. Define term operations $q_1(x,y,z), \ldots, q_n(x,y,z)$ by putting

$$q_i(x,y,z) = f(x,\ldots,x,\underset{(n-i+1)}{\underset{\uparrow}{y}},z,\ldots,z).$$

These terms satisfy

$$\begin{aligned} x &= q_1(x,x,y) && \\ q_i(x,y,y) &= q_{i+1}(x,x,y) && \text{for all } 1 \le i < n \\ q_i(x,y,x) &= x && \text{for all } 1 \le i \le n \qquad (\ddagger) \\ q_n(x,y,y) &= y. && \end{aligned}$$

Term operations satisfying such identities are called *directed Jónsson terms.* It is easy to see that putting $p_{2i-1}(x,y,z) = q_i(x,y,z), p_{2i}(x,y,z) = q_i(x,z,z)$ we obtain Jónsson terms and therefore directed Jónsson terms imply Jónsson terms.

It can be shown that the reverse implication also holds, even for infinite algebras [38]. Moreover, directed Jónsson terms have their own relational condition, i.e. we can extend Theorem 14 by two additional, equivalent conditions:

4. $\mathbf{A}$ has directed Jónsson terms.
5. For each subalgebra $\mathbf{R}$ of $\mathbf{B} \times \mathbf{C} \times \mathbf{C}$ regarded as a B-labeled digraph $b \in B$, $c, c' \in C$,
   - if both c and c' have b-labeled loops
   - and there is a directed walk from c to c'

   then there is a directed b-labeled walk from c to c'.

5.3.2 Jónsson Absorption

The definition of absorption is similar to the conditions imposed on near unanimity terms. In a similar way, we can talk about Jónsson absorption or directed Jónsson absorption: we say that $\mathbf{B}$ *Jónsson absorbs (directed Jónsson absorbs)* $\mathbf{A}$ when there is sequence of terms like in the definition of Jónsson terms (directed Jónsson terms) but with the condition (‡) replaced by $p_i(B,A,B) \subseteq B$.

Quite a few results (e.g. Proposition 10) can be strengthen by relaxing the assumptions and allowing Jónsson absorption or directed Jónsson absorption instead of the absorption from Definition 1. However, as an analysis of such relaxations is beyond the scope of this article, we move to yet another application of absorption.

6 Absorption in Taylor Algebras and Its Consequences

Proper absorption is common, even if algebras do not satisfy restrictive conditions. In fact, relatively mild assumption on the algebra forces either a strong restriction on the shape of compatible relations, or proper absorption.

From the algebraic perspective, the aforementioned mild assumption is, roughly, that the algebra is not "equationally trivial". By this, we mean that it has a set of term operations which satisfy some identities that cannot be satisfied by projections. By the Mod–Id Galois correspondence, this is equivalent to requiring that no algebra in $\mathcal{V}(A)$ is a G-set, where a *G-set* (in our idempotent world) is an algebra whose every operation is a projection and which has at least 2 elements. The theorem of Taylor [50] provides an equational characterization of this class:

▶ **Theorem 15.** *The following are equivalent for an algebra* $\mathbf{A}$.

1. $\mathcal{V}(\mathbf{A})$ *does not contain a* G*-set.*
2. $\mathbf{A}$ *has a* Taylor *term operation, that is, a term operation* t *that for each* i *satisfies an identity of the form*

$$t(\dots, \underset{\overset{\uparrow}{i}}{x}, \dots) = t(\dots, \underset{\overset{\uparrow}{i}}{y}, \dots),$$

where $\dots$ *stand for some sequences of* x*'s and* y*'s.*

An algebra satisfying the equivalent conditions in Theorem 15 is called a *Taylor* algebra. Taylor algebras are central to the algebraic approach to CSP: whenever $\mathrm{Pol}(\mathbb{A})$ does not contain a Taylor operation then $\mathrm{CSP}(\mathbb{A})$ is NP-complete and the tractability conjecture (also known as the algebraic dichotomy conjecture) states that, otherwise it is solvable in polynomial time. The reason behind the hardness part is that if Pol($\mathbb{A}$) is not Taylor, then $\mathbb{A}$ pp-interprets every relational structure compatible with the G-set in the variety, but every relational structure is compatible with a G-set.

6.1 Absorption Theorem

The following theorem is used to produce a proper absorption [8].

▶ **Theorem 16** (Absorption Theorem)**.** *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be Taylor algebras (of the same signature) and* $\mathbf{R}$ *a subdirect linked subalgebra of* $\mathbf{A} \times \mathbf{B}$*. Then either* $\mathbf{A}$ *or* $\mathbf{B}$ *has a proper absorbing subalgebra, or* $R = A \times B$*.*

Proof Strategy. Assume that neither of the algebras have a proper absorbing subalgebra. The strategy of the proof is the following.

- We produce a transitive term operation in $\mathbf{A}$ and $\mathbf{B}$. An operation t on A is *transitive* if, for each $a, b \in A$ and each coordinate i, there exists a tuple $(a_1, \ldots, a_n)$ with $a_i = a$ such that $t(a_1, \ldots, a_n) = b$. Such terms are produced by star composing a Taylor term (see Proposition 2.7 in [8]) using the fact that $\mathbf{A}$ has no proper absorbing subalgebra with respect to the binary operations which appear in the Taylor equations.
- We show that a maximal set $X \subseteq A$ (or $Y \subseteq B$) such that each $a \in A$ ($b \in B$) is adjacent to a common neighbor of all elements of X (Y), absorbs $\mathbf{A}$ ($\mathbf{B}$). Having X or Y nonempty can be obtained by replacing R by a suitable relational composition of the form $R \circ R^{-1} \circ R \circ \ldots$.
- The last item implies that necessarily $X = A$ or $Y = B$. In the first case, R has a nonempty *right center* – the set of elements $b \in B$ adjacent to every element of A. we show that the right center absorbs $\mathbf{B}$. Therefore it is equal to B and then $R = A \times B$.

◀

The theorem implies that the non-rectangularity of a binary relation $\mathbf{R} \leq \mathbf{A} \times \mathbf{B}$ enforces proper absorption in a subalgebra of $\mathbf{A}$ or $\mathbf{B}$. Indeed, let R be a non-rectangular relation. Viewing R as a bipartite graph we get a connected component which is not a biclique. The elements of this component which are on the left form a set A' which is a subuniverse of $\mathbf{A}$, and similarly elements on the right form, denoted by B', form a subuniverse of $\mathbf{B}$. The relation $R \cap (A' \times B')$ is subdirect and linked in $\mathbf{A}' \times \mathbf{B}'$ and is not the full product. The absorption theorem guarantees a proper absorption in $\mathbf{A}'$ or $\mathbf{B}'$.

▶ **Example 17.** Going back to the example in Section 2, we will use the Absorption theorem to show that the polymorphism algebra $\mathbf{A}$ of $\mathbb{K}_3^c$ is not Taylor. This is equivalent to proving that $\mathbb{K}_3^x$ pp-interprets every finite structure. In Section 2, we proved a stronger claim, but, by the discussion above, the weaker claim still implies that CSP($\mathbb{K}_3^c$) is NP-complete.

Assume for a contradiction that $\mathbf{A}$ is Taylor. The inequality relation R is a subdirect subalgebra of $\mathbf{A}^2$ and it is linked. By the absorption theorem, $\mathbf{A}$ has a proper absorbing subalgebra. This however directly contradicts Proposition 3, as there is no $\mathbf{B}$ with $R \cap B^2$ subdirect and linked.

6.2 Loop Lemma

The absorption theorem makes it possible to relax the assumptions of Corollary 4 in two ways:

- instead of assuming that every non-singleton subalgebra of $\mathbf{A}$ has a proper absorbing subuniverse, we assume that it is Taylor,
- instead of assuming that the relation is linked, we assume that R has *algebraic length* 1, i.e. there is a closed oriented walk in R with one more forward than backward edges.

The generalized theorem [11, 8] states:

▶ **Theorem 18** (Loop Lemma). *Let* $\mathbf{A}$ *be a Taylor algebra. If* $\mathbf{R} \leq \mathbf{A}^2$ *is subdirect and has algebraic length* 1*, then it has a loop.*

Proof. We present only a sketch of the proof. The proof supposes that $\mathbf{A}$ has more than one element (as otherwise the claim holds) and restricts $\mathbf{R}$ to a proper subalgebra of $\mathbf{A}$ while preserving subdirectness and algebraic length 1. The reasoning splits in two parts depending on the existence of a proper absorbing subuniverse in $\mathbf{A}$.

If $\mathbf{A}$ has an absorbing subuniverse, then the standard two-stage reasoning, as in the proof of Proposition 3, can be used. In the walking stage, we find $\mathbf{B}$ a proper absorbing subuniverse of $\mathbf{A}$ such that $\mathbf{R}$ restricted to $\mathbf{B}$ is subdirect. Then, in the reduction stage, we show that R restricted to $\mathbf{B}$ has algebraic length 1.

If $\mathbf{A}$ has no absorbing subuniverse, we take the smallest n such that R composed n-times with itself is linked. Such an n exists since the algebraic length of R is one. Since $\mathbf{A}$ has no absorbing subuniverses, the n-fold composition of R with itself is full (note that if $n = 1$ we found the needed loop). We set $\mathbf{B}'$ to consist of all the vertices on the right-hand side of a linked component of R composed with itself $(n-1)$-times. It is easy to see that R restricted to $\mathbf{B}'$ contains at least one cycle and we define $\mathbf{B}$ as the set of all the elements which are in some cycle in $\mathbf{B}'$. Direct graph-theoretical considerations, using the fact that the n-fold composition of R is full, show that $\mathbf{R}$ restricted to $\mathbf{B}$ has algebraic length one. ◀

A relatively simple consequence of the loop lemma is the CSP dichotomy over relational structures consisting of a single subdirect binary relation [11, 8] (these are exactly the smooth digraphs in the terminology of that paper). Theorem 18 is used as the main tool in showing that a core of such a graph is a disjoint union of directed cycles (which puts the CSP of such a graph in P), or it has no Taylor polymorphism and the CSP is NP-complete.

6.3 Siggers Term

The Loop lemma, and earlier a similar result for undirected graphs [33, 23], can be used to prove [49, 39] that every Taylor algebra has Taylor operations of a very particular form.

▶ **Corollary 19** (Taylor Implies Siggers). *Every Taylor algebra has a* 4*-ary term operation* s *and a* 6*-ary term* s' *such that*

$$s(x,y,z,x) = s(y,x,y,z) \text{ and } s'(x,y,x,z,y,z) = s'(y,x,z,x,z,y).$$

Proof. The argument is very similar to the reasoning in the proof of Theorem 13. Consider $\mathbf{F}$ – the free algebra on three generators and let $\mathbf{R}$ be the subalgebra of $\mathbf{F}^2$ generated by (x,y), (y,x), (z,y), and (x,z) for s (or $(x,y),(y,x),(x,z),(z,x),(y,z),(z,y)$ for s'). In both cases, $\mathbf{R}$ is subdirect in $\mathbf{F}^2$ and has algebraic length one. (For s', the relation R is additionally symmetric.) The loop lemma provides a loop in $\mathbf{R}$ which implies the appropriate term. ◀

The operations from the corollary are called Sigger's terms (a 4-ary one and a 6-ary one). The identities of 4-ary Siggers can be rewritten as $s(a,r,e,a) = s(r,a,r,e)$ which serves as an easy mnemonic. As both of the Sigger's operations are automatically Taylor, Corollary 19 provides an alternative characterization of Taylor algebras. It is quite surprising that (for finite idempotent algebras!) nontrivial identities always imply one specific nontrivial identity, a relatively nice and simple one at that. Even more surprising is a very recent result of Olšák [48] providing specific nontrivial identities satisfied by any idempotent Taylor algebra, not necessarily finite. His proof also uses absorption in a substantial way.

6.4 Cyclic Terms

Another equational characterization of Taylor algebras uses cyclic operations [8]. An operation t of arity $n \geq 2$ on a set A is *cyclic* if $t(x_1, \ldots, x_n) = t(x_2, \ldots, x_n, x_1)$.

▶ **Theorem 20** (Taylor Implies Cyclic). *If* $\mathbf{A}$ *is Taylor and* p *is a prime number greater than* $|A|$*, then* $\mathbf{A}$ *has a cyclic term operation of arity* p*.*

Proof. The proof splits into two, uneven, parts. The first important step is to reduce the problem to a question about compatible relations. Namely, it is enough to prove that each nonempty p-ary subpower $\mathbf{R}$ of $\mathbf{A}$ which is invariant under cyclic shifts (called *cyclic relation*) contains a constant tuple. This fact provides term operations which are cyclic with respect to one tuple. These "local cyclic operations" can be composed to a global one which finishes the proof.

In order to prove the constant tuples in cyclic relations we employ the Loop lemma. Consider a cyclic subpower $\mathbf{R} \leq \mathbf{A}^p$ and let $\mathbf{S}$ be the projection of $\mathbf{R}$ onto all but the last coordinate. By cyclic invariance, $\mathbf{S}$ is equal to the projection to all but the first coordinate, therefore the binary relation

$$T = \{((a_1, \ldots, a_{p-1}), (a_2, \ldots, a_p)) : (a_1, \ldots, a_p) \in R\}$$

is a subdirect subalgebra of $\mathbf{S}^2$. If we knew that T has algebraic length one, the Loop lemma would give us a loop in T, which clearly implies a constant tuple in T. Showing that T has algebraic length 1 is the technical core of the proof. ◀

The cyclic terms strengthen the characterization of Taylor algebras by means of *weak NU operations* [47] – these are such that their value on the tuples $(x, \ldots, x, y, x, \ldots, x)$ does not depend on the position of y (but is not necessarily equal to x like for the NU operations). Corollary 19 also follows from Theorem 20 as both Siggers operations can be obtained by identification of variables in a cyclic operation of suitable chosen prime arity, as observed in [43].

The cyclic terms, or more precisely their characterization by cyclic relations, allow to formulate the algebraic CSP dichotomy conjecture by means of properties of pp-definable relations as follows:

▶ **The Algebraic CSP Dichotomy Conjecture.** *Let* $\mathbb{A}$ *be a relational structure. If, for some (equivalently all) prime number* $p > |A|$*, every* p*-ary cyclic relation pp-definable from* $\mathbb{A}$ *has a constant tuple, then* $\mathrm{CSP}(\mathbb{A})$ *is tractable. Otherwise, it is NP-complete.*

6.5 Conservative CSPs

One of the biggest classes of CSPs known to exhibit the dichotomy was obtained by Bulatov [24]. His result confirms the algebraic dichotomy conjecture for all *conservative* CSPs, that is, CSPs over relational structures that contain all the unary relations.

▶ **Theorem 21** (Bulatov). *Assume that* $\mathbb{A}$ *contains all the unary relations and let* $\mathbf{A} = \mathrm{Pol}(\mathbb{A})$*. If* $\mathbf{A}$ *is a Taylor algebra, then* $\mathrm{CSP}(\mathbb{A})$ *is tractable, else it is NP-complete.*

Note that the conservativity of $\mathbb{A}$ is equivalent to the fact that each subset of A is a subuniverse of $\mathbf{A}$.

Bulatov's proof of Theorem 21 uses his technique of local analysis of finite algebras and is rather long and technical. Absorption allowed to provide a significantly shorter proof [3].

One ingredient of this alternative proof is the following fact stated in Theorem 36: If $\mathbf{P}$ is a Taylor algebra such that no subalgebra of $\mathbf{P}$ has a proper absorbing subuniverse (such algebras are called *hereditarily absorption free*), then $\mathbf{P}$ has a Maltsev term operation (comp. Theorem 36). For conservative algebras, this fact can be easily proved from Theorem 20: Consider a cyclic term t and observe that $t(a, a, \dots, a, b)$ is necessarily equal to b for any $a, b \in P$, since the result must be either a or b (from conservativity) and it cannot be a as otherwise $\{a\}$ would absorb $\{a, b\}$. But then $m(x, y, z) = t(x, y, y, \dots, y, z)$ is a Maltsev term.

The next ingredient is a certain "Rectangularity theorem" for conservative algebras (Theorem III.7 in [3]). Its simplified version is as follows.

▶ **Proposition 22.** *Let $\mathbf{P}, \mathbf{P}'$ be conservative Taylor algebras and R a subdirect subalgebra of $\mathbf{P} \times \mathbf{P}'$. Let, moreover, $\mathbf{Q}$ and $\mathbf{Q}'$ be minimal absorbing subalgebras of $\mathbf{P}$ and $\mathbf{P}'$ such that $R \cap (Q \times Q') \neq \emptyset$ and $R \cap (Q \times (P' \setminus Q')) \neq \emptyset$. Then $Q \times Q' \subseteq R$.*

Proof. The conservativity and the last assumption on R can be used to show that $R \cap (Q \times Q')$ is a subdirect, linked subuniverse of $\mathbf{Q} \times \mathbf{Q}'$ (we omit the proof here). Then the claim follows from the minimality of Q, Q' and the Absorption theorem. ◀

The Rectangularity theorem together with the reduction techniques shown in the proof of Proposition 10 are used to transform a CSP instance into an arc consistent instance with each $\mathbf{P}_x$ hereditarily absorption free and which has a solution whenever the original instance had. The idea is, imprecisely, that Proposition 10 allows to find a subinstance of a Prague instance where the $\mathbf{P}_x$'s in the subinstance are minimal absorbing subuniverses of the $\mathbf{P}_x$'s in the original instance. The Rectangularity theorem now guarantees the propagation of a solution to a suitably chosen subinstance. We continue in this way until every $\mathbf{P}_x$ is hereditarily absorption free. After the transformation is performed, all $\mathbf{P}_x$'s have Maltsev operations and we can apply the Bulatov–Dalmau algorithm for Maltsev constraints mentioned in Section 5.2.

Using Maróti's technique from [46], Bulatov has revisited and significantly simplified his original proof [25]. Notably, some parts of the revised proof (including e.g. the Rectangularity theorem) turned out to be very similar to the sketched proof by means of absorption – the interaction between absorption and Bulatov's local analysis deserves further attention.

7 Applications of Absorption to Local Consistency Checking

One of the main achievements of absorption is the characterization of the CSPs solvable by "local consistency methods". In general, a local consistency checking algorithm (LCC algorithm), operates on a family of local solutions to a CSP by removing the local solutions which are "inconsistent". The arc-consistency checking algorithms was an example of such an algorithm.

Systems of linear equations over the p-element field can be solved in polynomial time, but not by any LCC algorithm [30]. The restriction of the problem to equations involving at most 3 variables have the same properties and is equivalent to CSP($\mathbb{Z}_p$), where the domain of $\mathbb{Z}_p$ is the p-element field $GF(p)$ and the relations are affine subspaces of $GF(p)^3$.

The solvability by local consistency checking is preserved by pp-interpretations and homomorphic equivalence [44], therefore a necessary condition for $\mathrm{CSP}(\mathbb{A})$ to be solvable by LCC is that $\mathbb{A}$ does not pp-interpret a structure homomorphically equivalent (that is with homomorphisms to and from) with $\mathbb{Z}_p$. We call structures satisfying this necessary condition $\mathbb{Z}_p$–avoiding. The bounded width theorem states that, as conjectured in [44], this necessary condition is also sufficient. The theorem was proved using absorption [10] and independently by Bulatov [20] using his local analysis technique.

▶ **Theorem 23.** *If $\mathbb{A}$ is $\mathbb{Z}_p$-avoiding, then $\mathrm{CSP}(\mathbb{A})$ is solvable in polynomial time by local consistency checking.*

The plan for this section is first to prove the theorem for simplified instances using Prague instances from Section 4.2.3. Then we will move on to discuss other consistency notions for simplified instances, including consistency notions easier to compute. The section is concluded by an overview of results concerning consistency notions and algorithms that do not assume the simplicity of instances.

7.1 Prague Instances and Local Consistency Checking

Recall the definition of the simplified Prague instance from Section 4.2.3, and note that Proposition 10 reduces the Prague instance if at least one of the $\mathbf{P}_x$'s has a proper absorbing subuniverse. Unfortunately, this is not always the case for $\mathbb{Z}_p$-avoiding structures. Luckily, we can use the following lemma (Lemma 7.6 in [10], see also [12]) instead.

▶ **Lemma 24.** *Let $\mathbb{A}$ be $\mathbb{Z}_p$-avoiding. If $\mathbf{A} = \mathrm{Pol}(\mathbb{A})$ is simple (has only trivial congruences) and has no proper absorbing subalgebra, then for every $b \in A$ there is an n-ary operation f of $\mathbf{A}$ and elements $a_1, \dots, a_n$ such that*

$$f(a_1, \dots, a_{i-1}, a, a_{i+1}, \dots, a_n) = b \text{ for all } i \text{ and all } a \in A.$$

In the situation of the lemma, we say that f *points to* b. This operation is the missing tool necessary to tackle the following theorem:

▶ **Theorem 25.** *Let $\mathbb{A}$ be $\mathbb{Z}_p$-avoiding. Then every non-trivial Prague instance over $\mathbb{A}$ has a solution.*

Proof. The basic idea, and the structure of the proof is the same as in the proof of Theorem 18: we will show that every nontrivial Prague instance over $\mathbf{A} = \mathrm{Pol}(\mathbb{A})$ has a solution by shrinking the sets P_x until they are singletons. In case some $\mathbf{P}_x$ has a proper absorbing subalgebra, we use Proposition 10. It remains to deal with the case of no absorption.

In this case we choose x so that $|P_x| > 1$ and take a maximal proper congruence α_x of $\mathbf{P}_x$. For any $y \neq x$ consider the quotient of P_{xy} modulo α_x:

$$R_y = \{(a/\alpha_x, b) : (a, b) \in P_{xy}\} \leq \mathbf{P}_x/\alpha_x \times \mathbf{P}_y.$$

The partition of P_x/α_x into the components of linkedness of R_y defines a congruence of $\mathbf{P}_x/\alpha_x$, which, as α_x is maximal, is either the equality relation, or the full relation $P_x/\alpha_x \times P_x/\alpha_x$. We will say that the variable y is of type (1) when the relation is equality and of type (2) if it is full.

In both cases, we get a non-trivial information about P_{xy}. In the first case, R_y^{-1} is a graph of a surjection $P_y \to P_x/\alpha_x$. Its kernel, denoted α_y, is a congruence of $\mathbf{P}_y$ and P_{xy} modulo $\alpha_x \times \alpha_y$ is an isomorphism between P_x/α_x and P_y/α_y. In the second case, R_y is linked. Neither $\mathbf{P}_y$ nor $\mathbf{P}_x/\alpha_x$ have any proper absorbing subuniverses (because absorption can be lifted from quotients) and the Absorption theorem implies that R_y is the full product. Translating this to the original relation P_{xy} we get that each vertex in P_y is adjacent to an element in each α_x-block.

To shrink the instance, we choose one equivalence block of α_x and, for all y's of type (1), we choose a block of α_y mapped to the chosen block of α_x by R_y. For all the other y we take the whole P_y. Such a new, strictly smaller simplified instance is arc consistent because of the information we obtained in the previous paragraph.

So far the argument only required that $\mathbf{A}$ is Taylor, but now we need to use Lemma 24 to choose an operation pointing in $\mathbf{P}_x/\alpha_x$ to the chosen block of α_x. Using this operation, together with the structure of the R_y's, we are able to imitate the reasoning from the proof of Proposition 10 and prove that the new instance is Prague. This finishes the reduction. ◀

The theorem, modulo the reduction to simplified instances, proves Theorem 23. This reduction is rather direct but results in very inefficient consistency notions for the original CSP. In Section 7.3 we discuss consistency notions which can be applied without reducing to simplified instances.

7.2 (2,3)-Consistency, Circle Instances and Semidefinite Programming

To complete section 4.2.2, we will provide a sketch of a proof of Proposition 6. The proposition states that if, in a simplified $(2,3)$-consistent instance, at least one of the $\mathbf{P}_x$'s has an absorbing subuniverse then the instance has a proper $(2,3)$-consistent subinstance.

In order to prove this proposition, we require the following theorem which, despite surprising assumptions, is extremely useful [42].

▶ **Theorem 26.** *Let an instance of the CSP be arc consistent and such that for every variable x and $a \in P_x$, the map $x \mapsto a$ extends to a solution. If, for every x, we have $\mathbf{P}'_x \trianglelefteq \mathbf{P}_x$, and the restriction of the instance to the P'_x's is arc consistent, then the restriction to the P'_x's has a solution.*

We will not provide a sketch of the proof, only mention that it hinges on the following proposition from [13] which vaguely resembles the loop lemma:

▶ **Proposition 27.** *Let $\mathbf{R}' \trianglelefteq \mathbf{R}$ be subdirect subpowers of $\mathbf{A}^n$ and for every $a \in A$ the tuple $(a, \ldots, a)$ belongs to $\mathbf{R}$. Then $\mathbf{R}'$ contains a constant tuple.*

Proof. We present a sketch of the proof which splits into the usual stages. The walking stage finds a proper $\mathbf{B}$ such that $\mathbf{R}' \cap \mathbf{B}^n$ is subdirect in $\mathbf{B}^n$. The reduction stage is easy, indeed, having such $\mathbf{B}$ we can restrict both $\mathbf{R}'$ and $\mathbf{R}$ to $\mathbf{B}^n$. The assumptions of the theorem are satisfied for such restrictions and we can repeat the argument until $\mathbf{A}$ is a one-element algebra and the theorem holds.

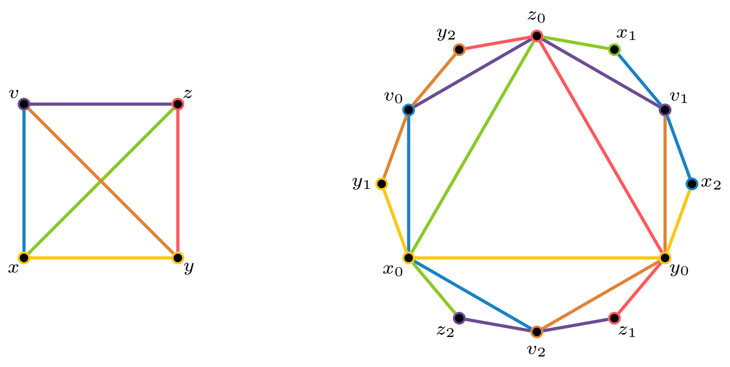
In order to complete the walking stage, we use pp-formulas which are trees and define a pre-order on subalgebras of $\mathbf{A}$: $\mathbf{B} \sqsubseteq \mathbf{B}'$ if $\mathbf{B}'$ can be pp-defined from $\mathbf{R}'$ and $\mathbf{B}$ by a tree pp-formula. We can show, *using absorption*, that this preorder contains elements which are not below the empty set, and, by the (omitted) definition of tree pp-formulas, we can choose an appropriate $\mathbf{B}$ there. ◀

Before launching into the proof of Proposition 6, we need to establish one more fact. Given a simplified CSP instance, we can construct another simplified instance (usually infinite) which has a solution if and only if the original instance contains a $(2,3)$-consistent subinstance. The idea is to, following Definition 5, construct the instance in steps:

1. start with a copy $P_{x_0y_0}$ of any constraint P_{xy} from the original instance
2. for each new constraint $P_{x_iy_j}$, consider every variable w (different from x and y) and introduce into the constructed instance new constraints P_{x_i,w_k} and P_{y_j,w_k} (where k is such that w_k is a new variable) and repeat.

The following example illustrates first few steps of this procedure:

▶ **Example 28.** The following picture presents an instance on four variables $\{x, y, z, v\}$ with the constraints $\mathbf{P_{xy}}, \mathbf{P_{yz}}, \mathbf{P_{zv}}, \mathbf{P_{vx}}, \mathbf{P_{zx}}, \mathbf{P_{yv}}$ together with a part of the instance which is responsible for its $(2, 3)$-consistency.



The constructed instance is infinite, but by a compactness argument (using the fact that $\mathbf{A}$ is finite), a large enough, finite part of the infinite instance plays the same role. This final remark allows us to finish the proof of Proposition 6.

Proof of Proposition 6. The proof starts with a $(2, 3)$-consistent instance over $\mathbf{P}_x$'s. The walking stage which, for every x, finds $\mathbf{P}'_x \trianglelefteq \mathbf{P}_x$ defining a proper arc consistent, absorbing subinstance of the original instance goes in exactly the same way as it was done in the proof of Proposition 10. The proof is actually simpler as we work with strictly stronger assumptions here.

The reduction stage follows from Theorem 26. Indeed, take a finite part of the simplified instance responsible for $(2, 3)$-consistency of the original instance. After setting P_{x_i} to P_x in this new instance, we get that for every $a \in P_{x_i}$ the map $x_i \mapsto a$ extends to a solution. The restriction of this instance to $P'_{x_i} = P'_x$ is an arc consistent, absorbing subinstance which, by Theorem 26, has a solution in P'_{x_i}'s.

Thus every finite part of the instance responsible for consistency of the original instance can be solved in the P'_x's and therefore, by compactness reasoning, we can find a $(2, 3)$-consistent subinstance of the original instance inside the P'_x's, which proves the proposition. ◀

The analogue for a circle instance, Proposition 8, can be proved in an almost identical way. Using Proposition 8 and a refinement of the non-absorbing part of the proof of Theorem 25 we can establish the following theorem.

▶ **Theorem 29.** *Let $\mathbb{A}$ be $\mathbb{Z}_p$-avoiding. Then every non-trivial circle instance over $\mathbb{A}$ has a solution.*

Now we are very close to defining a consistency notion which corresponds directly to semidefinite programming (SDP) relaxations of CSP instances. Each simplified CSP instance can be relaxed to a problem solvable by the semidefinite programming. This relaxation has very useful properties: among other things, it allows us to "almost solve almost solvable instances" of CSP of bounded width [9]. More precisely, if the instance is "almost solvable", i.e. solvable after forgetting a small number of constraints, we can use the solution to the SDP relaxation to produce an instance which does not forget many constraints and is pq-consistent [41].

▶ **Definition 30.** An arc consistent simplified instance is pq-consistent if for every $a \in P_x$ and every p, q circle patterns at x, a is reachable from itself via $j(p+q)+p$ for some j.

The pq-consistency implies solvability for instances which are $\mathbb{Z}_p$-avoiding by a proof almost identical to the proof for circle instances:

▶ **Theorem 31.** *Let $\mathbb{A}$ be $\mathbb{Z}_p$-avoiding. Then every pq-consistent instance over $\mathbb{A}$ has a solution.*

7.3 Consistency Notions for All Instances

A characterization of the set of templates whose CSP is solvable by local consistency checking was conjectured by Feder and Vardi in [30]. Even after the conjecture was confirmed [10], it was not clear whether a single consistency notion suffices to deal with all these problems. Note that up till now, all the consistency notions worked for simplified instances, and it is not hard to generalize them to all instances over binary constraints. But incorporating constraints of higher arities destroys the uniformity of the reasoning.

The first result identifying a consistency notion that works for all the $\mathbb{Z}_p$-avoiding templates is [6, 20]. The result uses the concept of $(2,3)$-minimality which will not be defined in this paper. The proof of this result in [6] is a small refinement of the proof of Theorem 23.

▶ **Theorem 32.** *Let $\mathbb{A}$ be $\mathbb{Z}_p$-avoiding. Then every $(2,3)$-minimal instance over $\mathbb{A}$ has a solution.*

Further results established other consistency notions which work for all the $\mathbb{Z}_p$-avoiding templates. Here we define Singleton Arc Consistency (SAC) [29], a well established notion of consistency. We present an algorithm for SAC using a pseudocode similar to that used for arc consistency:

```
for every variable $x$ do  add constraint $P_x := A$ to the instance
repeat
    for every variable $x$ and every $a \in P_x$ do
        run arc consistency with additional, temporary constraint $x = a$
        if  the last AC derived a contradiction  do  substitute $P_x$ with $P_x \setminus \{a\}$
    end for
until none of the $P_x$'s changed
```

Similarly as in the case of arc consistency, we say that an instance is a SAC instance if it can be returned by the algorithm above. The following theorem states that SAC works, uniformly, for all the CSPs solvable by local consistency checking.

▶ **Theorem 33.** *Let $\mathbb{A}$ be $\mathbb{Z}_p$-avoiding. Then every SAC instance over $\mathbb{A}$ has a solution.*

This theorem follows from generalizations of Theorems 29 and 31 to arbitrary instances.

8 Abelianness Versus Absorption

One of the chief achievements of universal algebra is finding suitable generalizations of several concepts in group theory, like abelianness, solvability, and the commutator [31, 34]. Here we only introduce the most basic concept of an abelian algebra.

▶ **Definition 34.** An algebra $\mathbf{A}$ is *abelian* if one of the equivalent conditions is satisfied:

- For every term function t elements a, b and tuples $\overline{c}, \overline{d}$:

  $$t(a,\overline{c}) = t(a,\overline{d}) \text{ implies that } t(b,\overline{c}) = t(b,\overline{d});$$

- the set $\{(a,a) : a \in \mathbf{A}\}$ is a block of some congruence on $\mathbf{A}^2$.

Examples of abelian algebras include the polymorphism algebras of $\mathbb{Z}_p$ from Section 7. This indicates that abelian algebras are natural obstacles for proving the dichotomy conjecture by means of refining the local consistency algorithms. In fact, $\mathbb{A}$ is not $\mathbb{Z}_p$-avoiding if and only if some subalgebra of $\mathrm{Pol}(\mathbb{A})$ has a nontrivial abelian quotient [51].

8.1 Abelianness Prevents Absorption

The notion of absorption is, in a sense, complementary to abelianness: the following theorem says that an abelian algebra has no non-trivial absorbing subuniverses. In fact, even a weaker property, solvability, prevents absorption [12].

▶ **Theorem 35.** *If $\mathbf{A}$ is abelian, then no subalgebra of $\mathbf{A}$ has a proper absorbing subalgebra.*

Proof. We only show a special case, that an abelian algebra cannot have a 1-element absorbing subuniverse.

Assume that $\{a\}$ absorbs an algebra $\mathbf{A}$ and α is a congruence of $\mathbf{A}^2$ from the definition of abelianness (the second item). Then $\{(a,a)\}$ absorbs $\mathbf{A}^2$ and thus $\{(a,a)/\alpha\}$ absorbs $\mathbf{A}^2/\alpha$. This in turns implies that $(a,a)/\alpha$ absorbs $\mathbf{A}^2$, but, from the abelianness of $\mathbf{A}$, $(a,a)/\alpha$ is not linked while $\mathbf{A}^2$ is. Since the linkedness is absorbed by a version of the argument from Proposition 3, we reach a contradiction. ◀

The algebras satisfying the conclusion of the previous theorem are called *hereditarily absorption free*, or *HAF* for short (note that they have already appeared in Section 6.5). This theorem is interesting in combination with the following simple consequence of the Absorption theorem [12].

▶ **Theorem 36.** *If $\mathbf{A}$ is HAF and Taylor, then $\mathbf{A}$ has a Maltsev term operation.*

Proof. In context of Sections 5.2 and 6.1 the proof is natural: to prove Maltsev it suffices to show that the free algebra $\mathbf{F}$ has rectangular subpowers. On the other hand, if an algebra $\mathbf{B}$ is Taylor and HAF, then all its subalgebras have rectangular subpowers because of the Absorption theorem. It is, therefore, enough to show that $\mathbf{F}$ is HAF. But $\mathbf{F}$ is a subpower of $\mathbf{A}$ and every subpower of a HAF algebra is HAF. We leave it for the reader as an exercise to prove the latter fact. ◀

By combining the last two theorems, we get that each abelian (or just solvable) Taylor algebra has a Maltsev term operation. This fact was known before absorption [34], but its proof was quite long and used heavy machinery.

8.2 Absorption Theorem for Higher Arity Relations

Abelianness is also an obstacle for generalizing the Absorption theorem to higher arities. The following example shows that a naive generalization does not work in general – even if all the binary projections of a ternary relation are full and the relation is not full, no absorbing subalgebra needs to exist.

▶ **Example 37.** Consider an algebra $\mathbf{A}$ over $\{0,1\}$ with a single ternary plus, i.e. $x, y, z \mapsto x + y + z \bmod 2$. The relation $\{(a, b, c) : a + b + c = 0 \mod 2\}$ is a non-full subuniverse of $\mathbf{A}^3$ and has full binary projections. However it is easy to see, that the algebra has no non-trivial absorbing subuniverses.

Actually, every abelian algebra can participate in a problematic ternary relation. Indeed, take an abelian algebra $\mathbf{A}$ and let α be a congruence on $\mathbf{A}^2$ from the definition of abelianness. It is easy to see that $\{(a, b, (a, b)/\alpha) : a, b \in \mathbf{A}\}$ is a non-trivial, subdirect subuniverse of $\mathbf{A}^2 \times (\mathbf{A}^2/\alpha)$ and has all the binary projections linked.

We finish with a theorem witnessing that abelianness is the only obstacle. Its proof is left as a harder exercise for the reader.

▶ **Theorem 38.** *Let $\mathbf{A}_1, \dots \mathbf{A}_n$ be Taylor algebras (in the same signature) and let $\mathbf{R}$, subdirect in $\prod_{i=1}^{n} \mathbf{A}_i$, be such that all the binary projections of $\mathbf{R}$ are linked. Then*
1. *some $\mathbf{A}_i$ has a proper absorbing subuniverse, or*
2. *some $\mathbf{A}_i$ has a proper congruence α such that $\mathbf{A}_i/\alpha$ is abelian, or*
3. *$R = \prod_{i=1}^{n} A_i$.*

9 Conclusions

The simple concept of absorption proved surprisingly useful in universal algebra and CSP. The main contribution of absorption to CSP is a proof of the characterization of CSPs of bounded width, and the main contribution to algebra is the existence of cyclic terms. However, the concept is not yet well understood even for finite Taylor algebras.

The main obstacle to applying absorption outside of CSPs of bounded width is the incompatibility with abelian algebras. In particular, to prove the CSP dichotomy conjecture one needs to be able to operate on instances which in some parts have absorption, but in other parts are e.g. abelian. Currently, apart from very few basic results, we lack the knowledge to work with such instances.

Another challenge in the field is to bridge the gap between the absorption theory and the local approach used by Bulatov. The structure of an algebra imposed by Bulatov's colored graphs is similar to the one imposed by absorption (or lack of absorption), but the concepts are seemingly different.

An active direction of research is to extend the results obtained by absorption to infinite algebras. In this direction, we already know that the characterization by directed Jónsson terms extends [38], an analogue of a Sigger's term exists [48], etc. However, many questions remain. In particular, we do not know the correct extent and statement of the loop lemma for infinite algebras, although some facts are known [15].

References

1 Kirby A. Baker and Alden F. Pixley. Polynomial interpolation and the Chinese remainder theorem for algebraic systems. *Mathematische Zeitschrift*, 143:165–174, 1975. `doi:10.1007/BF01187059`.

2 Libor Barto. Finitely related algebras in congruence modular varieties have few subpowers. to appear in JEMS.

3 Libor Barto. The dichotomy for conservative constraint satisfaction problems revisited. In *26th Annual IEEE Symposium on Logic in Computer Science—LICS 2011*, pages 301–310. IEEE Computer Soc., Los Alamitos, CA, 2011.

4 Libor Barto. Finitely related algebras in congruence distributive varieties have near unanimity terms. *Canad. J. Math.*, 65(1):3–21, 2013. doi:10.4153/CJM-2011-087-3.

5 Libor Barto. The constraint satisfaction problem and universal algebra. *The Bulletin of Symbolic Logic*, 21:319–337, 9 2015. doi:10.1017/bsl.2015.25.

6 Libor Barto. The collapse of the bounded width hierarchy. *Journal of Logic and Computation*, published online 2014. doi:10.1093/logcom/exu070.

7 Libor Barto and Marcin Kozik. New conditions for Taylor varieties and CSP. In *Proc. of the 2010 25th Annual IEEE Symposium on Logic in Computer Science*, LICS'10, pages 100–109, Washington, DC, USA, 2010. IEEE Computer Society. doi:10.1109/LICS.2010.34.

8 Libor Barto and Marcin Kozik. Absorbing subalgebras, cyclic terms, and the constraint satisfaction problem. *Logical Methods in Computer Science*, 8(1), 2012. doi:10.2168/LMCS-8(1:7)2012.

9 Libor Barto and Marcin Kozik. Robust satisfiability of constraint satisfaction problems. In *Proceedings of the 44th symposium on Theory of Computing*, STOC'12, pages 931–940, New York, NY, USA, 2012. ACM. doi:10.1145/2213977.2214061.

10 Libor Barto and Marcin Kozik. Constraint satisfaction problems solvable by local consistency methods. *J. ACM*, 61(1):3:1–3:19, January 2014. doi:10.1145/2556646.

11 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, 2008/09.

12 Libor Barto, Marcin Kozik, and David Stanovský. Mal'tsev conditions, lack of absorption, and solvability. *Algebra universalis*, 74(1):185–206, 2015. doi:10.1007/s00012-015-0338-z.

13 Libor Barto, Marcin Kozik, and Ross Willard. Near unanimity constraints have bounded pathwidth duality. In *Proceedings of the 2012 27th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 125–134. IEEE Computer Soc., Los Alamitos, CA, 2012. doi:10.1109/LICS.2012.24.

14 Libor Barto, Jakub Opršal, and Michael Pinsker. The wonderland of reflections. Manuscript, 2015.

15 Libor Barto and Michael Pinsker. The algebraic dichotomy conjecture for infinite domain constraint satisfaction problems. Manuscript, 2016.

16 Garrett Birkhoff. On the structure of abstract algebras. *Proc. Camb. Philos. Soc.*, 31:433–454, 1935.

17 Manuel Bodirsky. Constraint satisfaction problems with infinite templates. In Nadia Creignou, Phokion G. Kolaitis, and Heribert Vollmer, editors, *Complexity of Constraints*, volume 5250 of *Lecture Notes in Computer Science*, pages 196–228. Springer, 2008. doi:10.1007/978-3-540-92800-3_8.

18 V. G. Bodnarčuk, L. A. Kalužnin, V. N. Kotov, and B. A. Romov. Galois theory for Post algebras. I, II. *Kibernetika (Kiev)*, 5(3):1–10; ibid. 1969, no. 5, 1–9, 1969.

19 Simone Bova, Hubie Chen, and Matthew Valeriote. Generic expression hardness results for primitive positive formula comparison. *Inf. Comput.*, 222:108–120, January 2013. doi:10.1016/j.ic.2012.10.008.

20 Andrei Bulatov. Bounded relational width. Manuscript, 2009.

21 Andrei Bulatov and Víctor Dalmau. A simple algorithm for Mal′tsev constraints. *SIAM J. Comput.*, 36(1):16–27 (electronic), 2006.

22 Andrei Bulatov, Peter Jeavons, and Andrei Krokhin. Classifying the complexity of constraints using finite algebras. *SIAM J. Comput.*, 34:720–742, March 2005. doi:10.1137/S0097539700376676.

23 Andrei A. Bulatov. H-coloring dichotomy revisited. *Theoret. Comput. Sci.*, 349(1):31–39, 2005.

24 Andrei A. Bulatov. Complexity of conservative constraint satisfaction problems. *ACM Trans. Comput. Logic*, 12(4):24:1–24:66, July 2011. doi:10.1145/1970398.1970400.

25 Andrei A. Bulatov. Conservative constraint satisfaction re-revisited. *J. Comput. Syst. Sci.*, 82(2):347–356, March 2016. doi:10.1016/j.jcss.2015.07.004.

26 Hubie Chen and Matthew Valeriote. Learnability of solutions to conjunctive queries: The full dichotomy. In *COLT*, volume 40 of *JMLR Workshop and Conference Proceedings*, pages 326–337. JMLR.org, 2015.

27 Victor Dalmau, Marcin Kozik, Andrei Krokhin, Konstantin Makarychev, Yury Makarychev, and Jakub Opršal. Robust algorithms with polynomial loss for near-unanimity CSPs. submitted.

28 Víctor Dalmau and Justin Pearson. Closure functions and width 1 problems. In *Principles and Practice of Constraint Programming – CP'99, 5th International Conference, Alexandria, Virginia, USA, October 11-14, 1999, Proceedings*, pages 159–173, 1999. doi:10.1007/978-3-540-48085-3_12.

29 Romuald Debruyne and Christian Bessiere. Some practicable filtering techniques for the constraint satisfaction problem. In *In Proceedings of IJCAI'97*, pages 412–417, 1997.

30 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. doi:10.1137/S0097539794266766.

31 Ralph Freese and Ralph McKenzie. *Commutator theory for congruence modular varieties*, volume 125 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, Cambridge, 1987.

32 David Geiger. Closed systems of functions and predicates. *Pacific J. Math.*, 27:95–100, 1968.

33 Pavol Hell and Jaroslav Nešetřil. On the complexity of H-coloring. *J. Combin. Theory Ser. B*, 48(1):92–110, 1990.

34 David Hobby and Ralph McKenzie. *The structure of finite algebras*, volume 76 of *Contemporary Mathematics*. American Mathematical Society, Providence, RI, 1988.

35 Peter Jeavons. On the algebraic structure of combinatorial problems. *Theoretical Computer Science*, 200(1–2):185–204, 1998. doi:10.1016/S0304-3975(97)00230-2.

36 Peter Jeavons, David Cohen, and Marc Gyssens. Closure properties of constraints. *J. ACM*, 44(4):527–548, 1997.

37 Bjarni Jónsson. Algebras whose congruence lattices are distributive. *Math. Scand.*, 21:110–121 (1968), 1967.

38 Alexandr Kazda, Marcin Kozi, Ralph McKenzie, and Matthew Moore. Absorption and directed Jónsson terms. submitted.

39 Keith Kearnes, Petar Marković, and Ralph McKenzie. Optimal strong mal'cev conditions for omitting type 1 in locally finite varieties. *Algebra universalis*, 72(1):91–100, 2014. doi:10.1007/s00012-014-0289-9.

40 Vladimir Kolmogorov, Andrei A. Krokhin, and Michal Rolinek. The complexity of general-valued csps. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 1246–1258. IEEE Computer Society, 2015. URL: http://dx.doi.org/10.1109/FOCS.2015.80, doi:10.1109/FOCS.2015.80.

41 Marcin Kozik. Weak consistency notions for all the CSPs of bounded width. URL: `https://arxiv.org/abs/1605.00565`.

42 Marcin Kozik. Weak consistency notions for all the CSPs of bounded width. In *Proceedings of the Thirty-First Annual ACM-IEEE Symposium on Logic in Computer Science (LICS 2016)*, pages 633–641. IEEE Computer Society Press, July 2016.

43 Marcin Kozik, Andrei Krokhin, Matt Valeriote, and Ross Willard. Characterizations of several maltsev conditions. *Algebra universalis*, 73(3):205–224, 2015. `doi:10.1007/s00012-015-0327-2`.

44 Benoit Larose and László Zádori. Bounded width problems and algebras. *Algebra Universalis*, 56(3-4):439–466, 2007.

45 A. I. Mal'tsev. On the general theory of algebraic systems. *Mat. Sb. N.S.*, 35(77):3–20, 1954.

46 Miklóś Maróti. Tree on top of maltsev. Manuscript, 2010.

47 Miklós Maróti and Ralph McKenzie. Existence theorems for weakly symmetric operations. *Algebra Universalis*, 59(3-4):463–489, 2008.

48 Miroslav Olšák. The weakest non-trivial term condition for idempotent algebras. Manuscript, 2016.

49 Mark H. Siggers. A strong mal'cev condition for locally finite varieties omitting the unary type. *Algebra universalis*, 64(1-2):15–20, 2010. `doi:10.1007/s00012-010-0082-3`.

50 Walter Taylor. Varieties obeying homotopy laws. *Canad. J. Math.*, 29(3):498–527, 1977.

51 Matthew A. Valeriote. A subalgebra intersection property for congruence distributive varieties. *Canad. J. Math.*, 61(2):451–464, 2009. `doi:10.4153/CJM-2009-023-2`.

Constraint Satisfaction Problems over Numeric Domains*

Manuel Bodirsky[1] and Marcello Mamino[2]

1 Institut für Algebra, TU Dresden, Dresden, Germany
manuel.bodirsky@tu-dresden.de
2 Institut für Algebra, TU Dresden, Dresden, Germany
marcello.mamino@tu-dresden.de

Abstract

We present a survey of complexity results for constraint satisfaction problems (CSPs) over the integers, the rationals, the reals, and the complex numbers. Examples of such problems are feasibility of linear programs, integer linear programming, the max-atoms problem, Hilbert's tenth problem, and many more. Our particular focus is to identify those CSPs that can be solved in polynomial time, and to distinguish them from CSPs that are NP-hard. A very helpful tool for obtaining complexity classifications in this context is the concept of a polymorphism from universal algebra.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Constraint satisfaction problems, Numerical domains

Digital Object Identifier 10.4230/DFU.Vol7.15301.79

1 Introduction

Many computational problems from many different research areas in theoretical computer science can be formulated as Constraint Satisfaction Problems (CSPs) where the variables might take values from an infinite domain. There is a considerable literature about the computational complexity of particular infinite domain CSPs, but there are only few systematic complexity results. Most of these results belong to two research directions. One is the development of the *universal-algebraic approach*, which has been so successful for studying the complexity of finite-domain constraint satisfaction problems. The other direction is to study constraint satisfaction problems over some of the most basic and well-known infinite domains, such as the numeric domains $\mathbb{Z}$ (the integers), $\mathbb{Q}$ (the rational numbers), $\mathbb{R}$ (the reals), or $\mathbb{C}$ (the complex numbers), and to focus on constraint relations that are first-order definable from usual addition and multiplication on those domains. In this way, many computational problems that are of fundamental importance in computer science and mathematics can be studied in the same framework. Several recent results are concerned with obtaining a systematic understanding of the computational complexity of such CSPs; and this survey article is devoted to presenting these recent results in a common context, and to highlight some of the common threads that are likely to be fruitful also in the future.

For a fixed structure Γ with finite relational signature τ, the constraint satisfaction problem CSP(Γ) is the problem of deciding whether a given finite conjunction of atomic

* The authors have received funding from the German Science Foundation (DFG, project number 622397) and from the European Research Council (ERC, grant agreement number 681988, CSP-Infinity).

The Constraint Satisfaction Problem: Complexity and Approximability. *Dagstuhl Follow-Ups*, Volume 7, ISBN 978-3-95977-003-3.
Editors: Andrei Krokhin and Stanislav Živný; pp. 79–111
DAGSTUHL FOLLOW-UPS Dagstuhl Publishing
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Germany

τ-formulas is satisfiable in Γ. That is, for a given instance

$$R_1(\bar{x}_1) \wedge \cdots \wedge R_m(\bar{x}_m)$$

of this problem where $\bar{x}_1, \ldots, \bar{x}_m$ are tuples of variables, we want to decide whether there exists an assignment of values to those variables such that all the conjuncts (the 'constraints') are satisfied. We often refer to Γ as the *template* of the CSP. The computational complexity of CSP(Γ) has been studied intensively when the template Γ has a finite domain; it is always in NP, and Feder and Vardi [35] conjectured that it is either in P or NP-complete; see Barto [6] for a short survey on the state of the art concerning progress towards proving the conjecture.

When the template Γ might have an infinite domain, there is no hope for a complete classification of the complexity of CSP(Γ) in general [16]: every computational problem is equivalent (under polynomial-time Turing reductions) to a problem of the form CSP(Γ). Indeed, even individual templates over numeric domains, usually as a consequence of their practical significance, are the focus of large branches of current mathematical research. This research fixes many points on the complexity map of CSPs. More recently, we have seen systematic results that provide complexity results for entire areas of interesting, albeit less expressive, templates in this map. In this introduction, and in our survey, we first discuss known concrete templates and then move on to more systematic classifications.

A famous example of a computational problem that can be formulated as a CSP over a numeric domain is Hilbert's 10th problem. It is CSP($\mathbb{Z}; R_+, R_*, R_{=1}$) where

- R_+ stands for the ternary addition relation $\{(x, y, z) \in \mathbb{Q}^3 \mid x + y = z\}$,
- R_* stands for the ternary multiplication relation $\{(x, y, z) \in \mathbb{Q}^3 \mid x * y = z\}$, and
- $R_{=1}$ stands for the unary relation $\{1\}$.

Matiyasevich [79], building upon results of Davis, Putnam, and Robinson, showed that this computational problem is undecidable (for a reference see [80]).

Another example, sitting at the opposite end of the complexity spectrum, is the feasibility problem for linear programs. It can be formulated as CSP(Γ_{lin}) for $\Gamma_{\text{lin}} := (\mathbb{Q}; \leq, R_+, R_{=1})$ where R_+ and $R_{=1}$ are defined as before, but over the rational numbers instead of the integers. Then CSP(Γ_{lin}) is easily seen to be polynomial-time equivalent to the feasibility problem for linear programs (see Section 2.3), which is a computational problem of outstanding theoretical and practical interest [94]. The complexity of this problem has been an open problem until Khachiyan's discovery that the ellipsoid method gives a polynomial-time algorithm [64] (see Section 3). It is natural to ask which relations can be added to Γ_{lin} so that the resulting expanded structure still has a CSP that can be solved in polynomial time; this is discussed in Section 4.1.

The choice of the domain of Γ might or might not have an impact on the computational complexity of CSP(Γ). For example, the structure $(\mathbb{Q}; \leq, R_+, R_{=1})$ has the same CSP as the structure $(\mathbb{R}; \leq, R_+, R_{=1})$ (where R_+ and $R_{=1}$ are defined as above but over $\mathbb{R}$ instead of $\mathbb{Q}$). On the other hand, if we consider the problem CSP($\mathbb{Z}; R_+, R_*, R_{=1}$) and replace the integers $\mathbb{Z}$ by the reals or the complex numbers, and adapt the interpretation of the relations R_+, R_*, and $R_{=1}$ correspondingly, the complexity of the CSP changes dramatically: CSP($\mathbb{R}; R_+, R_*, R_{=1}$) is equivalent to the existential theory of the reals, which is decidable (see Section 5). Even better complexity results are known for CSP($\mathbb{C}; R_+, R_*, R_{=1}$) (again, see Section 5). If we consider the structure $(\mathbb{Z}; \leq, R_+, R_{=1})$ instead of $(\mathbb{Q}; \leq, R_+, R_{=1})$ we obtain the famous NP-complete integer program feasibility problem.

Also over the integers there are many natural CSPs that can be solved in polynomial time. Well-known examples are

- linear Diophantine equation systems: here the constraints are of the form

  $$a_1x_1 + \cdots + a_kx_k = a_0$$

  for constants $a_0, a_1, \ldots, a_k \in \mathbb{Z}$. Such systems can be solved efficiently using linear algebra, via an appropriate implementation of an algorithm computing the Smith Normal form and a careful analysis of the size of the coefficients that appear during the computation [61, 29].
- Difference logic: here the constraints are of the form $x - y \leq c$ for $c \in \mathbb{Z}$. Such systems can be solved efficiently using shortest path computations (see, e.g., [32]).

So far, we have seen a powerful framework that captures many natural and important computational problems, but a systematic picture is missing. For some restricted settings, however, there is a complete classification of the polynomial-time tractable and the NP-hard cases. We present two such settings; more follows in the main body of the text. The first such setting is the class of structures with domain $\mathbb{Q}$ that are definable over $(\mathbb{Q}; <)$ with the order alone. This includes for example the structure $(\mathbb{Q}; \text{Betw})$ where

$$\text{Betw} := \{(x, y, z) \in \mathbb{Q}^3 \mid x < y < z \vee z < y < x\}$$

is the so-called *betweenness relation*. The problem $\text{CSP}(\mathbb{Q}; \text{Betw})$ is known as the *betweenness problem* in theoretical computer science and known to be NP-complete. Also the cyclic ordering problem, and CSPs from temporal reasoning such as the Point Algebra [101] and the Ord-Horn class [85] can be cast in this form.

In general, when Γ is a structure that has the same domain as some structure Δ, and all relations in Γ are first-order definable in Δ (we do allow equality in first-order formulas), we refer to Γ as a *first-order reduct* of Δ (we first consider the expansion of Δ by all first-order definable relations, and then take a reduct in the usual sense, that is, we drop some of the relations). Classifying the CSPs of first-order reducts of some structure Δ therefore corresponds to a *bottom-up* approach to classifying CSPs, since first-order reducts of a structure Δ are typically 'simpler' than Δ.

The main result of Bodirsky and Kára [20] states that $\text{CSP}(\Gamma)$ is, for all first-order reducts Γ of $(\mathbb{Q}; <)$, either in P or NP-complete (Section 7). One of the central ideas of the classification in [20] is that the powerful *universal-algebraic approach* to constraint satisfaction, which has been developed for finite-domain CSPs, can be applied here, too. The reason for this is that first-order reducts of $(\mathbb{Q}; <)$ satisfy a strong model-theoretic condition, *ω-categoricity*, which can be seen as a finiteness condition via the characterization of Ryll-Nardzewski: a countably infinite structure Γ is *ω-categorical* if and only if the automorphism group of Γ has finitely many orbits of k-tuples, for all $k \geq 1$. For ω-categorical Γ, the complexity of $\text{CSP}(\Gamma)$ is completely captured by the so-called *polymorphisms* of Γ (see Section 2.2); they are a generalization of the concept of an endomorphism to higher arities. The extension of the theory of finite-domain CSPs to (subclasses of) ω-categorical structures has advanced significantly [8, 9, 23] and is outside the scope of this survey.

Structures on numerical domains are typically *not* ω-categorical. Consider for example the structure $(\mathbb{Z}; \text{Succ})$ for $\text{Succ} = \{(x, y) \mid x = y + 1\}$, the integers with the successor relation. Its automorphism group has only one orbit, but infinitely many orbits of pairs, and is therefore not ω-categorical by the theorem of Ryll-Nardzewski mentioned above. The structure $(\mathbb{Z}; \text{Succ})$ can be seen as one of the 'simplest' structures over a numerical domain and with finite signature that is not ω-categorical. Following again the bottom-up approach, we study the class of CSPs for first-order reducts of $(\mathbb{Z}; \text{Succ})$. This class contains non-trivial CSPs that can be solved in polynomial time; let us mention for example

$$\text{CSP}(\mathbb{Z}; \{(x, x, x + 1), (x, x + 1, x), (x, x + 1, x + 1) \mid x \in \mathbb{Z}\}) \ . \tag{1}$$

For first-order reducts Γ of $(\mathbb{Z}; \mathrm{Succ})$, we almost have a dichotomy: $\mathrm{CSP}(\Gamma)$ is in P, or NP-complete, or there exists a finite structure Γ' such that Γ and Γ' have the same CSP [24] (Section 8). Hence, the truth of the Feder-Vardi conjecture would imply that also the class of structures definable over the integers using the successor relation has a complexity dichotomy.

The border between polynomial-time tractable and NP-hard CSPs for first-order reducts of $(\mathbb{Z}; \mathrm{Succ})$ can again be described using polymorphisms, and polymorphisms also play an important role in the proof of the classification [24]. However, in order to work with polymorphisms even when the structure Γ is not ω-categorical, we might have to pass to a structure which has the same CSP as Γ, but a different domain. The point is that polymorphisms with certain properties might only exist when the structure is sufficiently saturated, in the model-theoretic sense. For example, instead of $(\mathbb{Z}; \mathrm{Succ})$, we would consider the structure $(\mathbb{Q}; \{(x, y) \mid x = y + 1\})$, which has the same CSP, but a richer set of (automorphisms and) polymorphisms. These phenomena are one of the reasons why the numeric domains $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, and $\mathbb{C}$ should not be discussed in isolation; indeed, there are many fruitful cross-connections between classifications over these domains.

Certain polymorphisms are of particular importance over numeric domains. An important example for first-order reducts of $(\mathbb{Q}; +, *)$ and of $(\mathbb{R}; +, *)$ is the operation $(x, y) \mapsto (x+y)/2$; such a reduct Γ has this polymorphism if and only if all relations of Γ are convex. Convexity has recently become a very active topic also in real algebraic geometry [50, 66, 49, 48]. Many of the central questions that are relevant to CSPs are open.

Another remarkable polymorphism for numeric domains is $(x, y) \mapsto \max(x, y)$ (or, dually, $(x, y) \mapsto \min(x, y)$). If a finite structure Γ has max as a polymorphism, then $\mathrm{CSP}(\Gamma)$ is known to be in P [53, 54]. The same is true for first-order reducts of $(\mathbb{Q}; <)$, and for first-order reducts of $(\mathbb{Z}; \mathrm{Succ})$ (an example of such a reduct is the structure from (1)). But already for first-order reducts Γ over $(\mathbb{Q}; <, +, 1)$ it is an open problem whether having max as a polymorphisms implies that $\mathrm{CSP}(\Gamma)$ can be solved by a polynomial-time algorithm; such an algorithm would also imply polynomial-time tractability for many problems of open computational complexity in seemingly different areas of theoretical computer science: the model-checking problem of the propositional μ-calculus, solving mean payoff games and simple stochastic games; these connections are discussed in Section 6.

Researchers in model theory have obtained remarkable results in the classification of first-order reducts of structures like $(\mathbb{C}; +, *)$ or $(\mathbb{R}; +, *)$ up to first-order interdefinability. For instance, Marker and Pillay [76] prove that any first-order reduct Γ of $(\mathbb{C}; +, *)$ that contains $+$ is either a first-order reduct of an expansion of $(\mathbb{C}; +)$ by constants, or the complex multiplication $*$ has a first-order definition in Γ. Similar results are available for $(\mathbb{R}; +, *)$ instead of $(\mathbb{C}; +, *)$; see [86, 75, 77, 39]. In order to be applicable for complexity analysis for CSPs, we would need refinements of these results for *primitive positive definability* (see Section 2.1) instead of first-order definability.

Finally, we also discuss classes of constraint satisfaction problems over numeric domains that provably do *not* have a complexity dichotomy: this turns out to be the case for first-order reducts of $(\mathbb{Z}; +, *)$; a proof can be found in Section 9. We were unable to prove such a non-dichotomy result for first-order reducts of $(\mathbb{R}; +, *)$. On the other hand, we have seen that already classifying first-order reducts of $(\mathbb{Q}; 1, +, \leq)$ presents considerable challenges. We discuss in Section 10 what the next steps for the bottom-up approach to classifying CSPs on numerical domains might be.

2 Constraint Satisfaction Problems

We use standard notation and terminology from model theory; see e.g. [52]. For better readability we abuse notation and identify formulas with the relations they define; e.g., we write $(\mathbb{Q}; x > y^2)$ for the relational structure $(\mathbb{Q}; \{(x, y) \in \mathbb{Q}^2 \mid x > y^2\})$.

2.1 Primitive Positive Formulas

A first-order formula is *primitive positive* if it is of the form

$$\exists x_1, \ldots, x_n \, (\psi_1 \wedge \cdots \wedge \psi_m)$$

where $\psi_1, \ldots, \psi_n$ are atomic formulas; that is, no negation, disjunction, and universal quantification is allowed (but equality is allowed). The constraint satisfaction problem for a relational τ-structure Γ can then be rephrased as follows: given a primitive positive τ-sentence ϕ (i.e., a primitive positive formula without free variables, formed with relations from Γ), does ϕ hold in Γ?

The relevance of primitive positive formulas for the CSP comes from the following lemma, due to which works over finite and infinite structures alike [53]).

▶ **Lemma 1** (Jeavons–Cohen–Gyssens). *Let Γ be a relational structure, and R a relation that can be defined using a primitive positive formula over Γ . Then* $\mathrm{CSP}(\Gamma, R)$*, that is, the CSP for the expansion of Γ by the relation R, is log-space equivalent to* $\mathrm{CSP}(\Gamma)$*.*

The proof idea is to replace occurrences of R in an instance by their primitive positive definition, introducing new variables for the existentially quantified variables in these definitions (in optimization, even if not presented at this level of generality, this idea is commonly used, and the newly introduced variables are called *slack variables*).

2.2 Polymorphisms

Which relations are primitive positive definable over a given structure? This can be a difficult question, but we have seen in the previous section that it is an important question when we want to study the computational complexity of a CSP. A very important tool for answering this question are *polymorphisms*.

▶ **Definition 2.** A function $f \colon B^k \to B$ *preserves* a relation $R \subseteq B^m$ if for all

$$(a_1^1, \ldots, a_1^m), \ldots, (a_k^1, \ldots, a_k^m) \in R$$

we have that $(f(a_1^1, \ldots, a_k^1), \ldots, f(a_1^m, \ldots, a_k^m)) \in R$. When Γ is a relational structure then f is called a *polymorphism* of Γ if f preserves all relations of Γ.

In other words, f is a polymorphism of Γ if and only if f is a homomorphism from Γ^k to Γ. Unary polymorphisms are also called *endomorphisms*, and *automorphisms* are precisely the bijective endomorphisms whose inverse is also an endomorphism. It is well-known that the set of all automorphisms forms a *permutation group*, the set of all endomorphisms a *transformation monoid*, and the set of all polymorphisms a *function clone*, which are a central topic in universal algebra. The key fact that links polymorphisms with primitive positive definability is that when R is primitive positive definable in Γ , then R is preserved by the polymorphisms of Γ (for any structure Γ over any domain). This is useful in the contrapositive to show that something is not primitive positive definable over Γ : it suffices to exhibit a polymorphism of Γ that does not preserve R. This test gives a necessary and sufficient criterion for structures Γ over a finite domain [25], and also for many infinite structures Γ , e.g., when Γ is countably categorical (see Section 7). And even when Γ is not countably categorical, polymorphisms can be an important tool (see Section 8).

2.3 Infinite Signatures

We now come to an important issue that we hid so far, but which is an important aspect of CSPs, in particular of CSPs over numeric domains: the problem of encoding instances if the structure Γ has an *infinite signature*. An infinite signature is natural when we want to view for example the feasibility problem for linear programs as a CSP. There, the constraints in an instance of the CSP are of the form $a_1x_1 + \cdots + a_nx_n \geq a_0$, for some $a_0, a_1, \dots, a_n \in \mathbb{Q}$. So we would consider the signature that contains a relation symbol for each of those relations; there is a countably infinite number of them.

Also over finite domains, many well-known computational problems call for CSP formulations with templates Γ having an infinite signature, for example:

- Horn-SAT: the signature contains for every $n \geq 0$ a symbol for the relations defined over $\{0, 1\}$ by the Boolean expression $\neg x_1 \vee \cdots \vee \neg x_n \vee x_0$ or by $\neg x_1 \vee \cdots \vee \neg x_n$.
- Linear equations over a finite field F: the signature contains for every $n \geq 0$ and $a_0, a_1, \dots, a_n \in F$ a symbol for the n-ary relation defined by $a_1x_1 + \cdots + a_nx_n = a_0$.

However, when the signature of Γ is infinite, the computational complexity of CSP(Γ) depends on how the symbols of the signature of Γ are represented in the input instances to CSP(Γ). For finite structures Γ , the standard way to represent a relation symbol R from the signature of Γ is by an explicit list of tuples that are in the relation R. When the domain of Γ is infinite, this is typically no longer an option (already the basic relations $<$ over $\mathbb{Q}$ or Succ over $\mathbb{Z}$ contain infinitely many tuples, so they cannot be stored explicitly). Alternatively, we can use symbolic representations of relations; how precisely these representations might look like depends on the domain, and might also depend on the specific algorithmic result that we want to establish. We list a couple of options.

1. In general, when Γ is a first-order reduct of some structure Δ with a finite signature, we can represent a relation R of Γ by its first-order definition in Δ.
2. More specifically, for first-order reducts Γ of $(\mathbb{Q}; <)$ we can use the fact that $(\mathbb{Q}; <)$ has quantifier elimination (see, e.g., Hodges [52]), so we can represent a relation R from Γ by its quantifier-free definition over $(\mathbb{Q}; <)$ in disjunctive normal form (DNF).
3. Likewise, the structure $(\mathbb{Z}; s)$ has quantifier elimination, where s is the unary successor function, and again we might use quantifier-free formulas in DNF to represent the relations of first-order reducts of these structures. In this situation, we have to point out another subtlety, namely how to represent terms of the form $s^n(x) := s(s(\cdots s(x) \cdots))$: whether n is coded in unary or in binary can make the difference between an easy and a hard problem (see e.g. the problems from Definition 38 and Definition 40 in Section 6).
4. Finally, also for the structure $(\mathbb{Q}; <, +)$ we can use the fact that $(\mathbb{Q}; <, +)$ has quantifier elimination in the language that additionally contains a constant symbol for each rational number [36], so we can represent a relation R from Γ by its quantifier-free definition in disjunctive normal form (DNF). Here it is natural to assume that the constants for the rational numbers are represented in binary.

Option (1) listed above is not well-suited for obtaining polynomial-time results for CSPs, since already deciding whether a single relation, even when represented by a quantifier-free formula, is empty or not is NP-hard. For the representations given in (2)-(4), on the other hand, many interesting algorithms exist that solve the CSP for infinite-signature reducts in polynomial time. We would like to stress that when the signature of Γ is finite, these input representation issues do not arise: all the representations given above would give the same complexity results for CSP(Γ).

In some cases there is a different approach to deal with infinite signatures. Let Γ be a relational structure and let Γ' be the structure obtained from Γ by dropping some of the relations (i.e., Γ' is a reduct of Γ in the classical sense). We say that the relations of Γ' form a *basis* for Γ if all relations in Γ have a primitive positive definition over Γ'. In this case, and if the signature of Γ' is finite, then we can use primitive positive formulas over Γ' to represent the relations from Γ . The following lemma illustrates this approach (and justifies our presentation of linear program feasibility in the introduction); the proof is easy and can be found in [18].

▶ **Lemma 3.** *Every relation*

$$R := \{(x_1, \dots, x_k) \in \mathbb{Q}^k \mid a_1x_1 + \dots + a_kx_k \geq a_0\}$$

for $a_0, a_1, \dots, a_k \in \mathbb{Q}$ has a primitive positive definition over $(\mathbb{Q}; \leq, R_+, R_{=1})$. Moreover, we can find a primitive positive definition whose size is polynomial in the representation size of $a_0, a_1, \dots, a_k$ when represented in binary.

We would like to mention that for CSPs of templates with infinite signature it makes sense to consider the restricted version where only k variables are allowed in the input, for a fixed natural number k. The fact that k-variable integer linear program feasibility can be decided in polynomial time, for example, is a celebrated result of Lenstra [72].

For finite domains it is an open problem whether there are infinite constraint languages Γ such that CSP(Γ) is computationally hard if the relations in the constraints are represented explicitly, but where the computational hardness is not already witnessed by a finite subset of the relations of Γ; see the discussion in [26].

3 Linear Programming

Our journey through constraint satisfaction problems over numeric domains begins with the linear program feasibility problem. This problem will serve as a model, and also as a tool for further investigations.

▶ **Definition 4.** The problem LINEAR PROGRAM FEASIBILITY is the CSP of the structure with domain $\mathbb{R}$ and all the relations of the form

$$R^{\mathrm{LP}}_{a_1,\dots,a_n,c} := \left\{x \in \mathbb{R}^n \;\middle|\; a_1x_1 + \dots + a_nx_n \geq c\right\}$$

with $a_1, \dots, a_n, c \in \mathbb{Q}$.

We assume, in our definition, that the variables range over the real numbers, but the coefficients must be rational. The restriction on the coefficients is justified by the need to manipulate them computationally. Nevertheless, one might want to abstract from the details of number representation. This might have practical reasons, because some applications rely on fixed precision floating point arithmetic implemented in hardware. Also, theoretically at least, one might pick the coefficients in a subset of the reals (or even of a non-Archimedean real-closed field) that is larger than the rational subfield, and yet admits an explicit representation. This approach leads to a model of computation in which (real) numbers are treated as black-box entities, and algorithms have access to an oracle that performs a certain set of basic operations on them. The complexity of algorithms is thus measured by the number of arithmetic operations and (crucially) order comparisons performed as a function of the amount of numerical inputs: this is the so-called Blum-Shub-Smale model of computation. An algorithm is said to be *strongly polynomial* if it is polynomial in both the Turing machine (or *bit*) model, and in the Blum-Shub-Smale model.

Linear programming is usually formulated as an optimization problem where the goal is to maximize a given linear function over the feasibility region. It is well known that linear programming is polynomial-time equivalent to the linear program feasibility problem (both in the Turing and in the Blum-Shub-Smale model). The groundbreaking application, due to Khachiyan, of the ellipsoid method provided the first polynomial-time algorithm for the linear program feasibility problem [64]. Other polynomial time algorithms addressing directly the optimization problem followed, notably Karmarkar's interior point *projective* method [62], and later barrier-function interior point methods (see for instance [103]). What all this techniques have in common is that they rely on infinite approximation procedures. To get from such methods a polynomial time decision procedure, one needs some a priori information of such nature, as to guarantee that the decision is determined by a degree of approximation obtainable in polynomial time. For example, Khachiyan's ellipsoid method needs a bound from below to the volume of the feasibility region (assuming that it has non-empty interior), and interior point methods require, essentially, to bound the representation size of the optimum (which is necessarily a rational number or $\pm\infty$). Thus neither of the known polynomial time algorithms could solve LINEAR PROGRAM FEASIBILITY on a non-Archimedean field, and none, in particular, is strongly polynomial.

The existence of a strongly polynomial algorithm for linear programming is, today, possibly the most important related problem. Dantzig's simplex method requires the complement of a *pivoting rule* to make a complete algorithm. Despite its practical effectiveness, no known pivoting rule has provably polynomial time, and most can be explicitly defeated [65, 55, 5, 43, 38, 37]. Sub-exponential randomized simplex algorithms have been constructed by Kalai [60] and Matoušek, Sharir, and Welzl [81]; see [47] for a recent improvement. If one restricts LINEAR PROGRAM FEASIBILITY to instances with specific properties, in some cases, strongly polynomial algorithms are known. By work of Tardos [97], this is the case if the coefficients a_i in Definition 4 are integral and bounded by a fixed constant. Megiddo describes a strongly polynomial algorithm for the case of two variables per inequality [82] and the same author gives a linear time combinatorial algorithm in fixed dimension [83] (i.e., for a fixed number of variables).

By formulating linear programming in the integer domain we obtain a well-known variation.

▶ **Definition 5.** The problem INTEGER LINEAR PROGRAM FEASIBILITY is the CSP of the structure with domain $\mathbb{Z}$ and all the relations of the form

$$R^{\text{ILP}}_{a_1,\dots,a_n,c} := \{x \in \mathbb{Z}^n \mid a_1x_1 + \dots + a_nx_n \geq c\}$$

with $a_1, \dots, a_n, c \in \mathbb{Z}$.

As opposed to the real numbers formulation, INTEGER LINEAR PROGRAM FEASIBILITY is NP-complete. Indeed, even the special case in which the variables are restricted to the set $\{0, 1\}$ is among Karp's 21 problems [63]. We list a few notable polynomial-time restrictions. In the totally unimodular case (i.e., the instances are of the form $A\bar{x} = \bar{c}$ where the matrix A is totally unimodular), satisfiability in $\mathbb{R}$ implies satisfiability in $\mathbb{Z}$, hence INTEGER LINEAR PROGRAM FEASIBILITY can be solved in strongly polynomial time by Tardos's algorithm.

The fixed dimension case is solved in polynomial time by an algorithm of Lenstra [72]. By contrast to the real domain situation, the restriction to two variables per inequality is still NP-complete, by a result of Lagarias [71].

We will devote the next two sections to CSPs that expand linear programming. In particular, we will consider semilinear and algebraic constraints.

4 Semilinear Constraints

We say that a subset of $\mathbb{R}^n$ is *semilinear* if it can be defined by a finite Boolean combination of linear inequalities with integer coefficients. A relational structure with domain $\mathbb{R}$ is called *semilinear* if all its relations are. In particular, the template for LINEAR PROGRAM FEASIBILITY is semilinear. As for linear programming, in the semilinear context, the domains $\mathbb{R}$ and $\mathbb{Q}$ are interchangeable. Formally, we choose to state the results hereafter for $\mathbb{R}$.

4.1 Semilinear Expansions of Linear Programming

For semilinear expansions of linear programming, a P–NP-complete dichotomy has been proven by Bodirsky, Jonsson, and von Oertzen [18]. This dichotomy has then been extended by Jonsson and Thapper to all semilinear expansions of $(\mathbb{R}; +)$ in [59]. The dichotomy is based on the notion of essential convexity.

▶ **Definition 6.** A subset S of $\mathbb{R}^n$ is called *essentially convex* if for all $a, b \in S$ the straight line segment intersects the complement $\bar{S}$ of S in finitely many points.

▶ **Theorem 7** (Bodirsky–Jonsson–von Oertzen). *Let $R_1, \ldots, R_n$ be semilinear relations. Then* $\mathrm{CSP}(\mathbb{R}; R_{=1}, R_+, \leq, R_1, \ldots, R_n)$ *is in P if $R_1, \ldots, R_n$ are essentially convex, and it is NP-complete otherwise.*

The hardness result in Theorem 7 follows from an even more general condition formulated in Lemma 30 in the next section. The algorithmic part is provided by the equivalence, for semilinear relations, of essential convexity and the class *Horn-DLR* proposed by Jonsson and Bäckström [57].

▶ **Definition 8.** A semilinear relation is called *Horn-DLR* (disjunctive linear relations) if it can be defined by a conjunction of clauses either of the form

$$p_1 \neq 0 \vee \cdots \vee p_n \neq 0$$

or of the form

$$p_1 \neq 0 \vee \cdots \vee p_n \neq 0 \vee p_0 \leq 0$$

where $p_0, \ldots, p_n$ are linear terms with coefficients in $\mathbb{Z}$.

▶ **Lemma 9** (Bodirsky–Jonsson–von Oertzen). *A semilinear relation is essentially convex if and only if it is Horn-DLR.*

In turn, Horn-DLR constraints (represented by conjunctions of clauses as in Definition 8, with the coefficients expressed in binary) can be solved in polynomial time thanks to a resolution-like algorithm discovered by Jonsson and Bäckström [57] and independently by Koubarakis [69].

Convex semilinear relations can be characterised by a polymorphism. In fact, if $S \subset \mathbb{R}^n$ is semilinear then S is convex if and only if it is preserved by the midpoint function

$$(x, y) \mapsto \frac{x+y}{2}$$

(to see this, observe that if $p, q \in S$, then S contains a dense subset of the segment between p and q, so by being semilinear, S contains all but finitely many of the points of that segment). We mention that the same argument also works in an even more general setting, namely for *semialgebraic* relations, that will be introduced in Definition 18 in the next section. It would be desirable to also have a polymorphism characterisation of essentially convex semilinear relations. This is, unfortunately, not possible.

▶ **Observation 10.** *There is no function $f : \mathbb{R}^n \to \mathbb{R}$ such that a semilinear relation S is essentially convex if and only if f preserves S.*

Proof. For a contradiction, assume that such a function f exists. Without loss of generality, f depends on all its arguments. We will prove that f is injective, but first we see how to obtain a contradiction from this fact. Consider restriction of f to the diagonal

$$\begin{aligned} g &: \mathbb{R} \to \mathbb{R} \\ &x \mapsto f(x, x, \dots, x) \end{aligned}$$

For each rational x, the singleton $\{x\}$ is clearly semilinear and essentially convex, therefore $g|_{\mathbb{Q}}$ is the identity. The order relation $\{(x, y) \mid x < y\}$ is a semilinear essentially convex set, therefore g is order preserving. It follows from the density of $\mathbb{Q}$ in $\mathbb{R}$ that g is the identity from $\mathbb{R}$ to $\mathbb{R}$. In particular, the image of g is precisely $\mathbb{R}$, so f can not be injective unless $n = 1$. In this case, $f = g$ is the identity and it preserves every set.

It remains to prove that f is injective. The relation $x = y \Rightarrow u = v$ is essentially convex, therefore it must be preserved by f. We claim that this implies the injectivity. Suppose that there are distinct $a, b \in \mathbb{R}^n$ such that $f(a) = f(b)$. We want to prove that f violates $x = y \Rightarrow u = v$. Pick i such that $a_i \neq b_i$. Since f depends on every argument, there are $c, d \in \mathbb{R}^n$ with $c_j = d_j$ for all $j \neq i$, and $c_i \neq d_i$ such that $f(c) \neq f(d)$. We claim that (a, b, c, d) witnesses that f violates $x = y \Rightarrow u = v$. In fact, for all $j \neq i$, we have $c_j = d_j$, and, a fortiori, $a_j = b_j \Rightarrow c_j = d_j$. Also $a_i = b_i \Rightarrow c_i = d_i$, because, by construction, the premise of the implication is false. We conclude that for all $j \in \{1, \dots, n\}$ we have that $a_i = b_i$ implies $c_i = d_i$. However, $f(a) = f(b)$ and $f(c) \neq f(d)$. ◀

Nevertheless, one can see that essentially convex sets can be characterised by a polymorphism in a non-Archimedean extension on $\mathbb{Q}$. All totally ordered vector spaces over $\mathbb{Q}$ have the same semilinear CSP. In fact, the first-order theory of non-trivial totally ordered vector spaces over $\mathbb{Q}$ is complete (see for instance [99][Chapter 2, Remark 7.9]). Therefore we can replace $\mathbb{R}$ with such a non-Archimedean extension in the study of semilinear CSPs.

▶ Remark. Let $\mathbb{Q}[\epsilon]$ denote the $\mathbb{Q}$-vector space of all polynomials in one indeterminate ϵ with rational coefficients. Consider the order on $\mathbb{Q}[\epsilon]$ which is induced by viewing ϵ as a positive number that is smaller than all positive rational numbers; formally, we order the polynomials lexicographically with respect to their coefficients, starting with the constant term, then the coefficient of degree one, and so on in increasing order of degree. Semilinear and essentially convex subsets of $(\mathbb{Q}[\epsilon])^d$ are defined as for the reals. Now, let S be a semilinear relation, let ϕ be the Boolean combination of inequalities with integer coefficients that defines S, and let S' be the semilinear relation defined by ϕ on $\mathbb{Q}[\epsilon]$. Then the following are equivalent:

- S is essentially convex;
- S' is essentially convex;
- S' is preserved by the following function

$$\begin{aligned} f\colon (\mathbb{Q}[\epsilon])^2 &\to \mathbb{Q}[\epsilon] \\ (\alpha(\epsilon), \beta(\epsilon)) &\mapsto \gamma(\epsilon) := \left(\frac{1}{2} + \epsilon\right) \alpha(\epsilon^2) + \left(\frac{1}{2} - \epsilon\right) \beta(\epsilon^2) \,. \end{aligned}$$

Proof Sketch. If S is not essentially convex, then this is witnessed by rational points, and it is easy to see that S' cannot be preserved by f. Conversely, we prove that f preserves all essentially convex sets over $\mathbb{Q}[\epsilon]$. Observe that f is injective, because

$$\alpha(\epsilon^2) = \left(\frac{1}{2} + \frac{1}{4\epsilon}\right) \gamma(\epsilon) + \left(\frac{1}{2} - \frac{1}{4\epsilon}\right) \gamma(-\epsilon)$$

$$\text{and} \quad \beta(\epsilon^2) = \left(\frac{1}{2} - \frac{1}{4\epsilon}\right) \gamma(\epsilon) + \left(\frac{1}{2} + \frac{1}{4\epsilon}\right) \gamma(-\epsilon) \,.$$

Therefore, using the syntactic characterization of essential convexity in [57] (which holds in all ordered $\mathbb{Q}$-vector spaces by the completeness mentioned above), we only need to prove that f preserves all the rational constants, $+$, and the positivity relation $x > 0$, and this is immediate. ◀

Jonsson and Thapper prove a dichotomy result for the larger class of all semilinear expansions of $(\mathbb{R}; +)$ [59]. These include all the CSPs covered by Theorem 7, but also contain templates that may not be able to express full linear programming. For instance, expanding linear programming by the non essentially convex relation $|x - y| > 1$ produces an NP-complete CSP, nevertheless $(\mathbb{R}; +, |x - y| > 1)$ is tractable. In its short form, Jonsson and Thapper's result reads as follows.

▶ **Theorem 11** (Jonsson–Thapper). *Let $R_1, \dots, R_n$ be semilinear relations. Then the problem* $\mathrm{CSP}(\mathbb{R}; R_+, R_1, \dots, R_n)$ *is either in P or NP-complete.*

Theorem 11 comes with explicit tractability conditions. Unfortunately, these conditions are too complex to fit comfortably into our survey. However, we can get a feeling of the insights by looking at the special case of semilinear expansions of $(\mathbb{R}; R_+, R_{=1})$.

▶ **Corollary 12.** *Let $R_1, \dots, R_n$ be semilinear relations. Then* $\mathrm{CSP}(\mathbb{R}; R_+, R_{=1}, R_1, \dots, R_n)$ *is in P if one of the following conditions is satisfied:*

1. *all unary relations primitively positively definable in* $(\mathbb{R}; R_+, R_{=1}, R_1, \dots, R_n)$ *are essentially convex;*
2. *all unary relations primitively positively definable in* $(\mathbb{R}; R_+, R_{=1}, R_1, \dots, R_n)$ *are either singletons or unbounded (i.e. not contained in an interval);*

otherwise, it is NP-complete.

Here the first case is dealt with by reduction to the essentially convex case, the second by a procedure, called *affine consistency*, that approximates from above the affine span of the feasibility region of the input constraints.

For every semilinear structure Γ with finite signature $\mathrm{CSP}(\Gamma)$ is in NP; this can be observed from a more general result which also holds for semilinear structures with an infinite signature. When the semilinear relations in Γ are represented by quantifier-free formulas where the constants in the polynomials are represented in binary, then we can check satisfiability of an instance non-deterministically by first guessing a disjunct for each disjunction in the formula, and then checking in polynomial time the satisfiability of the resulting feasibility problem obtained as the conjunction of the selected (strict or non-strict) linear inequalities, e.g. using the algorithm of Jonsson and Bäckström [57].

4.2 Semilinear Constraints over the Integers

Semilinear subsets of $\mathbb{Z}^n$ and semilinear structures with domain $\mathbb{Z}$ are defined analogously as above for $\mathbb{R}$. A special case of tractable semilinear constraints over the integers has been found by Jonsson and Lööw [58].

▶ **Definition 13.** A semilinear structure Γ over $\mathbb{R}$ is called *scalable* if for every relation R of Γ and for every $(x_1, \dots, x_k) \in R$ there exists a positive $a \in \mathbb{R}$ such that $(bx_1, \dots, bx_k) \in R$ for all $b > a$.

▶ **Definition 14.** A semilinear structure Γ with domain $\mathbb{R}$ has the *integer property* if all satisfiable instances of $\mathrm{CSP}(\Gamma)$ are satisfiable over $\mathbb{Z}$.

▶ **Theorem 15** (Jonsson–Lööw). *All scalable semilinear structures have the integer property.*

A partial converse to Theorem 15 is given by the following result of Jonsson and Thapper [59].

▶ **Theorem 16** (Jonsson–Thapper). *Let Γ be a semilinear structure such that $\mathrm{CSP}(\Gamma, R_+)$ has the integer property. Then Γ is scalable.*

The following observation shows that CSPs of semilinear structures over the integers in fixed dimension are in P, using Lenstra's algorithm.

▶ **Observation 17.** *For every fixed $d \in \mathbb{N}$ there is a polynomial time algorithm deciding the satisfiability over $\mathbb{Z}$ of Boolean combinations of formulas of the form $p(x_1, \dots, x_n) \geq 0$ where p is a linear polynomial with variables $x_1, \dots, x_n$ and integer coefficients that are represented in binary.*

Proof. For each input formula ϕ, our algorithm first extracts the list of the linear polynomials $p_1, \dots, p_k$ appearing in ϕ. We say that two points $\bar{x}, \bar{y} \in \mathbb{R}^d$ have the same type if $\mathrm{sign}(p_i(\bar{x})) = \mathrm{sign}(p_i(\bar{y}))$ for all $i = 1, \dots, k$, where

$$\mathrm{sign}(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases} .$$

Clearly the truth value of $\phi(\bar{x})$ depends only on the type of $\bar{x}$.

It is easy to prove that there are points in $\mathbb{R}^d$ of at most $\tau_d(k)$ distinct types, where

$$\tau_d(k) = \sum_{i=0}^{d} 2^i \binom{k}{i} .$$

In fact, we claim that there are at most $\tau_d(k)$ types, and that this bound is tight when the hyperplanes $p_k = 0$ are in general position. By induction on k, we add the polynomials one by one. At the k-th step, we create at most $\tau_{d-1}(k-1)$ new types with $p_k = 0$, and exactly as many in general position. Now we count the $p_k \gtrless 0$ types. For each of the new $p_k = 0$ types, we see that its restriction to $p_1, \dots, p_{k-1}$, that we had at the $(k-1)$-th step, splits into two new types corresponding to $p_k < 0$ and $p_k > 0$. The remaining part of the $(k-1)$-th step types is incompatible with either $p_k < 0$ or $p_k > 0$, so it does not produce any new non-empty type. Thus we get the recurrence $\tau_d(k) = 2\tau_{d-1}(k-1) + \tau_d(k-1)$, hence the formula.

In particular, $\tau_d(k)$ is polynomial in k, therefore we can compute a list of all these types in polynomial time by solving at most $\sum_{i=0}^{k-1} \tau_d(i)$ linear programs. Now, using Lenstra's algorithm, we can exclude those types that do not contain points in $\mathbb{Z}^d$, and finally we check the truth value of ϕ for each of the remaining ones. ◀

5 Algebraic Constraints

Adding zero sets of polynomials to the set of basic constraints will rapidly bring about intractability. In the integer domain, the general problem of the satisfiability of a single (multi-variate) polynomial equation, also known as Hilbert's 10th problem, is undecidable by the celebrated result of Davis, Matiyasevich, Putnam, and Robinson. In fact, they show that satisfiability of a 13 variable polynomial is undecidable over $\mathbb{N}$, later improved by Matiyasevich to 9 (see [78] and [56]). Using the 4 squares theorem, therefore, satisfiability of a single polynomial equation in $\mathbb{Z}$ is also undecidable for a fixed dimension (just replace each variable x_i with $y_{i,1}^2 + y_{i,2}^2 + y_{i,3}^2 + y_{i,4}^2$). We are not aware of any decidability result in dimension 2. It has been proven by Manders and Adleman [74] that deciding the satisfiability of a single equation of the form

$$ax^2 + bx = c$$

over $\mathbb{N}$ is NP-complete. The decidability of satisfiable polynomial equations over the rationals is a major open problem. For more details on Hilbert's tenth problem and its extension to $\mathbb{Q}$, we refer the reader to [80] and [95].

One might consider a reduct of integer arithmetic in which *only* multiplication, and no addition, is available. It is well known that this fragment, called Skolem arithmetic, has a decidable first-order theory. Exploratory work on CSPs of reducts of Skolem arithmetic has been published by Glaßer, Jonsson, Martin [42].

The natural relations to consider as basic constraints over the real numbers are the semialgebraic relations.

▶ **Definition 18.** A subset of $\mathbb{R}^n$ is called *semialgebraic* if it can be represented as a finite Boolean combination of *basic semialgebraic sets* of the form $\{\bar{x} \in \mathbb{R}^n \mid p(\bar{x}) \geq 0\}$ where p is a polynomial with integer coefficients.

Semialgebraic sets over $\mathbb{R}$ are a rich yet manageable class that forms the basis of real algebraic geometry. The computational treatment of semialgebraic geometry goes back to Tarski's decision procedure for the first-order theory of the reals [98].

▶ **Theorem 19** (Tarski). *There is an effective quantifier elimination procedure for the first-order theory of the structure* $(\mathbb{R}; 0, 1, <, +, \times)$.

The complexity bound on Tarski's original procedure is very large: a tower of exponentials as high as the size of the input. Collins' cylindrical algebraic decomposition provides a method running in polynomial time for fixed dimension (fixed number of variables in the formula), and doubly exponential in the dimension [30]. This bound has been further improved to exponential in the dimension by Renegar [88, 89, 90]. Of special interest to us is the subproblem known as the existential theory of the reals.

▶ **Definition 20.** The problem EXISTENTIAL THEORY OF THE REALS is the CSP for the structure with domain $\mathbb{R}$ and all the relations of the form

$$R^{\text{ETR}}_{n,p} = \{\bar{x} \in \mathbb{R}^n \mid p(\bar{x}) = 0\}$$

where $p \in \mathbb{Z}[x_1, \ldots, x_n]$ is represented as a list of coefficients, in binary.

The best bound currently known on the complexity of EXISTENTIAL THEORY OF THE REALS is the following, obtained by Canny [28].

▶ **Theorem 21** (Canny). *The problem* EXISTENTIAL THEORY OF THE REALS *is in* PSPACE.

We chose basic relations of the form $p(x) = 0$. Adding to this set also all the relations of the form $p(x) \geq 0$ and $p(x) > 0$ would not increase the complexity of EXISTENTIAL THEORY OF THE REALS. In fact, the first is clearly equivalent to $\exists y\; p(x) - y^2 = 0$ and $x \neq 0$ is defined by $\exists y\; xy + 1 = 0$. More generally, it is easy to see that the semialgebraic relations are precisely those that have a primitive positive definition over the structure $(\mathbb{R}; (R^{\text{ETR}}_{n,p})_{n,p})$. As for linear programming, we chose to present the existential theory of the reals as an infinite signature CSP, it is however an easy exercise to find a finite basis for it. A non-trivial equivalence between the existential theory of the reals and the following CSP has been established by Schaefer and Štefankovič [92][Theorem 4.1]

▶ **Definition 22.** The problem STRICT INEQUALITIES is the CSP of the structure with domain $\mathbb{R}$ and the relations of the form

$$R^{\text{STRICT}}_{n,p} = \{\bar{x} \in \mathbb{R}^n \mid p(\bar{x}) > 0\}$$

where $p \in \mathbb{Z}[x_1, \ldots, x_n]$ is represented as a list of coefficients, in binary.

▶ **Theorem 23** (Schaefer–Štefankovič). *The problems* STRICT INEQUALITIES *and* EXISTENTIAL THEORY OF THE REALS *are polynomial time equivalent.*

Observe that the sets primitively positively definable in $(\mathbb{R}; (R^{\text{STRICT}}_{n,p})_{n,p})$ are necessarily open, therefore a strict subset of all semialgebraic sets. A number of other problems, many of a geometric flavour, have been proven to be equivalent to EXISTENTIAL THEORY OF THE REALS. We will mention two such results due to Kratochvíl and Matoušek [70], and Schaefer [91] respectively.

▶ **Definition 24.** The problem INTERSECTION GRAPH OF SEGMENTS is the CSP of the structure with domain $\mathbb{R}^4$ and the binary relations Int and its negation $\neg$Int, defined as follows. Let $S^{x_2,y_2}_{x_1,y_1}$ denote the straight line segment in $\mathbb{R}^2$ with endpoints (x_1, y_1) and (x_2, y_2).

$$\big((x_1, y_1, x_2, y_2)(x_1', y_1', x_2', y_2')\big) \in \text{Int} \;\Leftrightarrow\; S^{x_2,y_2}_{x_1,y_1} \cap S^{x_2',y_2'}_{x_1',y_1'} \neq \emptyset$$

▶ **Theorem 25** (Kratochvíl–Matoušek). *The problems* INTERSECTION GRAPH OF SEGMENTS *and* EXISTENTIAL THEORY OF THE REALS *are polynomial time equivalent.*

▶ **Definition 26.** The problem UNIT LENGTH LINKAGES is the CSP for the structure with domain $\mathbb{R}^2$ and the relation

$$R^{\text{ULL}} = \{\big((x, y), (x', y')\big) \mid (x - x')^2 + (y - y')^2 = 1\}$$

denoting that the Euclidean distance of (x, y) and (x', y') is one.

▶ **Theorem 27** (Schaefer). *The problems* UNIT LENGTH LINKAGES *and* EXISTENTIAL THEORY OF THE REALS *are polynomial time equivalent.*

The existential theory of the reals has a natural analogue in the complex field. The problem of deciding the satisfiability of a system of polynomial equations in $\mathbb{C}$ has been called the *Hilbert Nullstellensatz problem.*

▶ **Definition 28.** HILBERT NULLSTELLENSATZ PROBLEM is the CSP of the structure with domain $\mathbb{C}$ and the relations of the form

$$R^{\mathrm{HN}}_{n,p} = \{\bar{x} \in \mathbb{C}^n \mid p(\bar{x}) = 0\}$$

where $p \in \mathbb{Z}[x_1, \dots, x_n]$ is represented as a list of coefficients, in binary.

It is not hard to reduce HILBERT NULLSTELLENSATZ PROBLEM to EXISTENTIAL THEORY OF THE REALS, but a far lower complexity bound for HILBERT NULLSTELLENSATZ PROBLEM is known conditionally to the generalized Riemann Hypothesis, thanks to the following result of Koiran [67, 68].

▶ **Theorem 29** (Koiran). *Assuming the generalized Riemann Hypothesis,* HILBERT NULLSTELLENSATZ PROBLEM *is in the class Arthur-Merlin, hence, a fortiori, at the second level of the polynomial hierarchy (Π_2).*

For the Blum-Shub-Smale model of computation, HILBERT NULLSTELLENSATZ PROBLEM and EXISTENTIAL THEORY OF THE REALS are both NP-complete, for the complex and the real Turing machine model respectively, see [12].

In order for a semialgebraic expansion of linear programming to attain a complexity class below NP, it is necessary to restrict the template to essentially convex relations, because of the following result of Bodirsky, Jonsson, and von Oertzen [18][Lemma 3.5].

▶ **Lemma 30** (Bodirsky–Jonsson–von Oertzen). *Let R be a semialgebraic set which is not essentially convex, and suppose that the failure of essential convexity is witnessed by a segment with rational endpoints, then* $\mathrm{CSP}(\mathbb{R}; R_+, R_{=1}, \leq, R)$ *is NP-hard.*

The technical assumption about rational endpoints is indeed necessary, see [18][Remark 3.6]. Even convexity is far from being a sufficient condition for tractability. One obstacle comes from the well-known sum of square roots problem.

▶ **Definition 31.** The problem SUM OF SQUARE ROOTS is the following computational task:
Input: $a_1, \dots, a_n, b \in \mathbb{N}$
Output:

- YES if $b \leq \sqrt{a_1} + \cdots + \sqrt{a_n}$,
- NO otherwise.

It is a long standing open problem to characterise the complexity of this problem. The membership in NP of SUM OF SQUARE ROOTS is also open and would imply that the Euclidean travelling salesman problem is NP-complete [40]. In fact, SUM OF SQUARE ROOTS is not even known to be in the polynomial hierarchy, and the best upper bound today is the fourth level of the counting hierarchy [2]. Unfortunately, already very simple CSPs, for instance the expansion of the template for linear program feasibility by the relation $x^2 \leq y$, can simulate SUM OF SQUARE ROOTS.

▶ **Observation 32.** SUM OF SQUARE ROOTS *is polynomial time many-one reducible to*

$$\mathrm{CSP}(\mathbb{R}, R_+; R_{=1}, +, x^2 \leq y)\,.$$

Proof.

$$b \leq \sqrt{a_1} + \cdots + \sqrt{a_n} \;\Leftrightarrow\; \exists x_1, \ldots, x_n \in \mathbb{R} \begin{cases} b = x_1 + \cdots + x_n \\ x_1^2 \leq a_1 \\ \quad\vdots \\ x_n^2 \leq a_n \end{cases}$$

◀

▶ **Observation 33.** SUM OF SQUARE ROOTS *is polynomial time many-one reducible to*

$$\mathrm{CSP}(\mathbb{R}; R_+, R_{=1}, x^2 + y^2 \leq 1)\,.$$

Proof. It suffices to find a primitive positive definition for the relation $0 \leq x \leq \sqrt{k}$ of size $O(\log k)$ for all constant $k \in \mathbb{N}$. We use the following equivalence

$$x^2 \leq k \;\Leftrightarrow\; \exists a, b \in \mathbb{R} \begin{cases} a^2 + b^2 \leq 1 \\ (k+1)a = 2x \\ (k+1)b = k - 1 \end{cases}$$

observing that the linear relations $(k+1)a = 2x$ and $(k+1)b = k-1$ admit a succinct primitive positive definition by iterated doubling. ◀

There is a connection between convex semialgebraic CSPs and semidefinite programming. A *spectrahedron* is a subset of $\mathbb{R}^d$ of the form

$$\left\{\bar{x} \in \mathbb{R}^d \;\middle|\; x_1 A_1 + \cdots + x_d A_d + B \text{ is positive semidefinite}\right\}$$

where $A_1, \ldots, A_d, B$ are real symmetric matrices of the same size. Semidefinite programming is the task of minimizing linear functions over spectrahedral domains. The relations that are primitively positively definable over spectrahedra are called *semidefinite representable*. Clearly all semidefinite representable relations are convex and semialgebraic. The converse has been conjectured by Helton and Nie [48]: they conjecture that every convex semialgebraic set is semidefinite representable. Helton and Nie's conjecture has been proven in dimensions two [93], and it remains one of the most important open problems in the field. The Helton-Nie conjecture implies that every CSP for a template with finitely many convex semialgebraic relations would be a special case of the semidefinite program feasibility problem. Not much is known about the complexity of this problem. By Ramana's duality [87] it is either in NP ∩ coNP or outside of NP ∪ coNP. By results of Tarasov and Vyalyi [96] the evaluation of arithmetic circuits (PosSLP) is reducible to it, thus also SUM OF SQUARE ROOTS (for the reduction from SUM OF SQUARE ROOTS to PosSLP see [2]).

In fixed dimension, all semialgebraic CSPs are in P by the cylindrical decomposition algorithm [30]. However, if we except Boolean combinations of linear inequalities (with algebraic coefficients), we do not know any expansion of linear programming by a convex non-linear semialgebraic relation which has a CSP in P. The theory of primitive positive definability for reducts of the real field structure is not developed. For instance, we do not know whether the relation $x^6 \leq y$ is primitively positively definable over linear relations and $x^2 \leq y$.

6 Maximum as a Polymorphism

The maximum polymorphism describes interesting classes of semi-linear CSPs both in the rational and in the integer domain. To begin with, it has been proven by Jeavons and Cooper [54] that, for finite domains, if all the relations of a template are preserved by the maximum polymorphism (according to some total ordering of the domain) then the CSP is in P. The same authors observe that the CSP of all max-closed relations on a finite domain is maximally tractable, in other words adding any further constraints to it would make it NP-complete. Formally we have:

▶ **Theorem 34** (Jeavons–Cooper). *Any max-closed CSP instance containing c constraints each of them satisfied by at most t tuples can be solved in time $O(c^2t^2)$.*

▶ **Theorem 35** (Jeavons–Cooper). *Given a structure $(D;<)$ where $3 \leq |D| < \infty$ and $<$ is a total order on D, denote by $\mathcal{M}_{2,D}$ the (finite) set of all binary max-closed relations over D. Let R be any relation over D which is not max-closed. Then $\mathrm{CSP}(D;\mathcal{M}_{2,D},R)$ is NP-complete.*

Observe that the statement of Theorem 34 is uniform in the constraints, in the sense that the algorithm takes the constraints (represented in table form) as part of its input. Therefore, Theorem 34 is adapted to situations in which the relevant domain is not fixed. This algorithm is, indeed, the well-known *arc consistency* procedure. Generally speaking, algorithms of this variety keep, for each variable, a list of allowed values, and iterate over the constraints to check whether any of the allowed values can be excluded. Obviously the procedure converges to a fixed point after at most $n \cdot d$ iterations, where n denotes the size of the domain and d the number of variables of the input instance. If at any time some variable has no more allowed values, the instance is unsatisfiable. Otherwise, there might or there might not be a solution.

Jeavons and Cooper obtain Theorem 34 by showing that, despite providing in general only a one-sided test, arc consistency correctly decides all max-closed CSPs. Jeavons, Cohen, and Gyssens extended this result to finite domain CSPs having a semilattice polymorphism, defined as follows.

▶ **Definition 36.** The operation $f\colon D^2 \to D$ is called a *semilattice operation* if it is idempotent, commutative, and associative:

$$f(x,x) = x, \qquad f(x,y) = f(y,x), \qquad f(x,f(y,z)) = f(f(x,y),z).$$

▶ **Theorem 37** (Jeavons–Cohen–Gyssens). *A finite domain CSP whose relations are preserved by a semilattice operation is in P.*

To complete the picture for finite domain problems, let us point out that arc consistency naturally extends to a class of algorithms based on establishing *local consistency*. The finite domain CSPs that can be solved by this method have been recently characterized by Barto and Kozik [7].

Theorem 34 gives us a method to solve infinite domain CSPs through a technique termed *sampling* in [21, Definition 2.1]. We say that an infinite template Γ has a *sampling procedure* if for every instance I of $\mathrm{CSP}(\Gamma)$ we can construct in polynomial time a finite substructure Γ_I of Γ such that I is satisfiable if and only if I is satisfiable as an instance of $\mathrm{CSP}(\Gamma_I)$. Clearly, Theorem 34 coupled with a sampling procedure for Γ provides a polynomial-time decision procedure for $\mathrm{CSP}(\Gamma)$ whenever Γ is closed under maximum. This general method has provided polynomial and weakly polynomial time algorithms for several concrete CSPs in the past. Most notably the following results, published respectively by Hochbaum and Naor [51], and by Bezem, Nieuwenhuis, and Rodríguez-Carbonell [10].

▶ **Definition 38.** The problem MONOTONE TVPI INTEGER PROGRAMMING (for *two variables per inequality*) is the CSP of the structure with domain $\mathbb{Z}$ and all the relations of the form

$$R^{\text{MTVPI}}_{a,b,c} = \big\{(x,y) \in \mathbb{Z}^2 \;\big|\; ax - by \geq c\big\}$$

where $a, b \in \mathbb{N}$ and $c \in \mathbb{Z}$.

▶ **Theorem 39** (Hochbaum–Naor). *A solution to an instance of* MONOTONE TVPI INTEGER PROGRAMMING *taking values from* $\{-N, \ldots, N\}$ *can be computed in time polynomial in* N *and the size of the instance.*

▶ **Definition 40.** The problem MAX-ATOMS is the CSP of the structure with domain $\mathbb{Z}$ and all the relations of the form

$$R^{\text{MA}}_{c} = \big\{(x,y,z) \in \mathbb{Z}^3 \;\big|\; \max(x,y) + c \geq z\big\} \qquad \text{for } c \in \mathbb{Z}.$$

▶ **Theorem 41** (Bezem–Nieuwenhuis–Rodríguez-Carbonell). *The problem* MAX-ATOMS *can be solved in weakly polynomial time.*

The proofs of these theorems rely on the observation that the respective signatures are max-closed (and the relations $R^{\text{MTVPI}}_{a,b,c}$ are indeed also min-closed). Broadly speaking, both results provide algorithms of the weakly polynomial kind: Theorem 41 explicitly, Theorem 39 because one can compute bounds $x_i^{\max}$ and $x_i^{\min}$ polynomial in the size of the instance [94][§17.1]. One might wonder whether these results can be improved to actually yield a polynomial algorithm. At least for Theorem 41 this is unfortunately not the case.

▶ **Theorem 42** (Lagarias). *The problem* MONOTONE TVPI INTEGER PROGRAMMING *is NP-complete.*

Lagarias reduces MONOTONE TVPI INTEGER PROGRAMMING to the problem WEAK PARTITION [71], which was proven NP-complete ultimately by reduction from KNAPSACK [100]. Theorems 39 and 42 together say that MONOTONE TVPI INTEGER PROGRAMMING is weakly NP-complete (assuming P $\neq$ NP). It can be observed, however, that being max-closed does not imply tractability in the integer domain even for finite signature CSPs.

▶ **Example 43.** CSP($\mathbb{Z}$; $x = 1$, $x = -1$, $x = 2y$, $x + y \geq z$) is NP-complete.

Proof. Our CSP is clearly in NP, because it is a sub-problem of INTEGER LINEAR PROGRAMMING. NP-hardness is proven through a reduction from MONOTONE TVPI INTEGER PROGRAMMING. The constraint $ax - by \geq c$ with $a, b \in \mathbb{N}$ is equivalent to $ax + (2^\beta - b)y - c \geq 2^\beta y$, where β is the smallest exponent such that $2^\beta \geq b$. This is, in turn, equivalent to the primitive positive formula

$$\exists\, z_1, z_2, z_3, z_4, z_5, z_6 \;\begin{cases} ax \geq z_1 \\ (2^\beta - b)y \geq z_2 \\ z_1 + z_2 \geq z_3 \\ z_4 = \operatorname{sign}(-c) \\ |c|\, z_4 \geq z_5 \\ z_3 + z_5 \geq z_6 \\ z_6 = 2^\beta y \end{cases} .$$

All atomic relations in this formula are of the types $x+y \geq x$, $x = 2y$, 1 , -1 except $z_6 = 2^\beta y$ and those of the form $kv_1 \geq v_2$ for $k \geq 0$ and v_1, v_2 denoting variables. Relations of the first kind are easily expressed by chaining β constraints of the type $x = 2y$. To express $kv_1 \geq v_2$ we proceed recursively:

$$\begin{cases} 0 \geq v_2 \;\; \Leftrightarrow \;\; \exists t \;\; t = 2 * t \;\wedge\; t+t \geq v_2 & \text{for } k = 0 \\ kv_1 \geq v_2 \;\; \Leftrightarrow \;\; \exists t \;\; k'v_1 \geq t \;\wedge\; t+t \geq v_2 & \text{for } k = 2k' \text{ even} \\ kv_1 \geq v_2 \;\; \Leftrightarrow \;\; \exists t \;\; 2k''v_1 \geq t \;\wedge\; t+v_1 \geq v_2 & \text{for } k = 2k''+1 \text{ odd.} \end{cases}$$

◀

The concept of *primitive positive interpretation* formalises a powerful form of simulation of a CSP by another CSP, generalising the notion of primitive positive definability to templates that have different domains; for the technical definition see [14]. Since the existence of a semilattice polymorphism is preserved by primitive positive interpretations, the CSP of example 43 cannot interpret primitively positively any NP-complete finite domain CSP, yet it is NP-complete.

The situation changes for the MAX-ATOMS problem. First of all, one can formulate MAX-ATOMS also for the rational or the real domains, however, as it was observed already in [10], the real, rational, and integer formulations are polynomial-time equivalent. MAX-ATOMS is also unlikely to be NP-complete because it was proven in the same paper to be in NP ∩ coNP. The original proof of NP ∩ coNP membership constructs *small unsatisfiability certificates* using an appropriate proof system. The result, however, can be better understood through a connection with mean payoff games implicit in the work of Möhring, Skutella, and Stork on scheduling under and/or precedence constraints [84], and later discovered independently by Atserias and Maneva [4].

Mean payoff games are a class of so-called graph-games. The setup for a game $\mathcal{G}$ is a finite graph G whose edges (E_G) are labelled integer weights $\{w_e\}_{e \in E_G}$. The play takes place between two players, which we call Max and Min, taking turns at moving a token along the edges of G. The graph G has no sinks, and the vertices are divided into two subsets $V_G = V_G^{\max} \sqcup V_G^{\min}$, the first *belongs* to Max, the second to Min. The token is initially on some vertex v_0, and each player selects the next move when the token is on a vertex belonging to him. The value of a (infinite) play $v_0, v_1, \ldots$ is

$$\operatorname{val}(v_0, v_1, \ldots) = \liminf_{n \to \infty} \frac{1}{n+1} \sum_{i=0}^{n} w_{(v_i, v_i+1)} \,.$$

Player Max wants to maximize this value, Min wants to minimize it. The following theorem was proven by Eherenfeucht and Mycielski [34].

▶ **Theorem 44** (Ehrenfeucht–Mycielski). *Given a mean payoff game $\mathcal{G}$ and a starting vertex v_0 in the graph of $\mathcal{G}$, there is a value $\operatorname{val}(\mathcal{G}, v_0)$ and a pair of memoryless strategies σ and τ (not depending on v_0) for Max and Min respectively, such that Max can secure a value not smaller than $\operatorname{val}(\mathcal{G}, v_0)$ by playing according to σ, and Min can secure a value not larger than $\operatorname{val}(\mathcal{G}, v_0)$ by playing according to τ.*

Here *memoryless* means that at all times the move to play depends only on the current position of the token (and not, for instance, on the previous play or on a random choice).

▶ **Definition 45.** The problem MEAN PAYOFF GAMES is the following computational task:
Input: a mean payoff game $\mathcal{G}$.
Output:

- YES if $\mathrm{val}(\mathcal{G}, v_0) \geq 0$ for all starting vertices v_0 in the graph of $\mathcal{G}$,
- NO otherwise.

It turns out that MAX-ATOMS and MEAN PAYOFF GAMES are equivalent in the following precise sense [84].

▶ **Theorem 46** (Möhring–Skutella–Stork). *The problems* MAX-ATOMS *and* MEAN PAYOFF GAMES *are polynomial-time many-one reducible to each other.*

The NP ∩ coNP membership of MEAN PAYOFF GAMES was first observed by Zwick and Paterson [104][Theorem 4.2], and it is essentially a consequence of the symmetric nature of the game. In fact, although Definition 45 is not symmetric in its form, standard arguments (see for instance [102][Propositions 2.7, 2.8]) prove that that the decision problem MEAN PAYOFF GAMES is polynomial-time Turing equivalent to solving mean payoff games (i.e., computing a pair of optimal memoryless strategies with the property from Theorem 44).

Theorem 46 is actually a re-discovery of the concept of *potential transformation*, which has been known for games since the work of Gurvich, Karzanov, and Khachiyan [45]. The set of conditions that a potential transformation must satisfy in order to witness $\mathrm{val}(\mathcal{G}, v_0) \geq 0$ for all v_0 is, indeed, precisely the corresponding MAX-ATOMS instance in Atserias and Maneva's reduction.

The problem MAX-ATOMS has been, in turn, proven equivalent to a number of other well-known computational tasks, most notably scheduling under and-or precedence constraints, and solving two-sided systems of max-plus linear equations. The problem TWO SIDED MAX-PLUS LINEAR SYSTEMS is the problem of deciding whether a given system of equalities of the form

$$\max(a_1 + x_1, \dots, a_m + x_m) = \max(b_1 + y_1, \dots, b_n + y_n)$$

with $a_1, \dots, a_m, b_1, \dots, b_n \in \mathbb{Z}$ has a solution. As for MAX-ATOMS, it is immaterial whether we are looking for a solution over $\mathbb{Z}$, $\mathbb{Q}$, or $\mathbb{R}$.

▶ **Theorem 47** (Bezem–Nieuwenhuis). *The problems* MAX-ATOMS *and* TWO SIDED MAX-PLUS LINEAR SYSTEMS *are polynomial-time many-one reducible to each other.*

Max-plus algebra studies the algebraic properties of the *tropical semiring*, which can be defined as the structure $\mathbb{R} \cup \{-\infty\}$ with the standard ring operations $+$ and $\times$ replaced by max and $+$ respectively. Its study is motivated by applications to scheduling and routing problems, as well as more abstract connections to algebraic geometry. TWO-SIDED MAX-PLUS LINEAR SYSTEMS are precisely the analogue in the tropical semiring of standard systems of linear equations.

The best known algorithms for this cluster of equivalent problems come apparently from the game perspective. They are weakly polynomial, for instance Zwick and Paterson's [104], or sub-exponential (randomized), for instance Halman's [46], or both, as Björklund and Vorobyov's [11]. Other algorithms are known to converge quickly in practice (similarly to the simplex algorithm for linear programming), for example Gurvich, Karzanov, and Khachiyan's procedure [45] and its variants.

The problem of the existence of a polynomial-time solution for MEAN PAYOFF GAMES, or equivalently MAX-ATOMS, has been described as presenting now *exactly the same challenge* *as linear programming did before 1979* [102]. This description is grounded essentially on observations of an algorithmic nature. For instance, the two sub-exponential algorithms cited above are based precisely on adapting the sub-exponential method of Matoušek Sharir and Welzl for combinatorial linear programming [81]. A formal connection is presented by Allamigeon, Benchimol, Gaubert, and Joswig [1]: if there is a *combinatorial* pivoting rule for the simplex method yielding polynomial time convergence, then mean payoff games are in P.

We obtain an interesting development considering the following formulation of max-atoms over the real numbers, which we know to be equivalent to the problem over $\mathbb{Z}$.

▶ **Definition 48.** The problem MAX-ATOMS OVER $\mathbb{R}$ is defined as the CSP of $\mathbb{R}$ with all the relations of the form

$$R_c^{\mathrm{MA}\mathbb{R}} = \big\{(x,y,z) \in \mathbb{R}^3 \bigm| \max(x,y) + c \geq z\big\} \quad \text{for } c \in \mathbb{Q}.$$

Obviously the constraints $R_c^{\mathrm{MA}\mathbb{R}}$ are semilinear, closed under maximum, and it is also apparent that these constraints are preserved by all translations $t_k(x) := x + k$ for $k \in \mathbb{Q}$. The class of max-closed translation-invariant semilinear sets has been studied in the context of tropical geometry, where these sets have been called *tropically convex* [33] (in this field, it is customary to consider the dual situation, using min instead of max). Formally, tropically convex sets are the tropical analogue of the convex cones in classical geometry. In this sense, the CSP for tropically convex relations is the notional analogue of LINEAR PROGRAM FEASIBILITY for tropical geometry. One can observe that not all tropically convex relations arise as feasibility regions of instances of MAX-ATOMS over $\mathbb{R}$. An example of such a relation is the relation defined by $x \leq (y+z)/2$, and more generally the relations of the form $\{(x,y,z) \mid (a+b)x \leq ay + bz\}$ for $a, b \neq 0$.

Bodirsky and Mamino [22] give a finite basis for tropically convex semilinear relations.

▶ **Theorem 49** (Bodirsky–Mamino). *A subset of $\mathbb{R}^n$ is semilinear, max-closed, and translation-invariant if and only if it is primitively positively definable over*

$$(\mathbb{R}; x = 1, x = -1, <, 2x \leq y + z\,,\ x \leq y \vee x \leq z)\,.$$

The CSP for tropically convex semilinear relations is equivalent to solving an extension of mean payoff games called *stochastic mean payoff games* [22]. These are defined similarly to the deterministic case, except that the set of vertices is partitioned in three components $V_G = V_G^{\max} \sqcup V_G^{\min} \sqcup V_G^{\mathrm{rand}}$; when the token is on a vertex in the new component V_G^{rand}, the next move is selected uniformly at random. The goal for Max (resp. Min) is to maximize (resp. minimize) the expected value of the play.

▶ **Definition 50.** The problem TROPICALLY CONVEX CONSTRAINTS is

$$\mathrm{CSP}(\mathbb{R}; x = 1, x = -1, <,\ 2x \leq y + z,\ x \leq y \vee x \leq z)\,.$$

▶ **Theorem 51** (Bodirsky–Mamino). *The problem* TROPICALLY CONVEX CONSTRAINTS *is polynomial-time Turing equivalent to solving stochastic mean payoff games, thus in NP $\cap$ coNP.*

Stochastic mean payoff games are a generalization of mean payoff games, and they have the same computational complexity as Condon's *simple stochastic games* [3]. There are many connections between mean payoff games, stochastic mean payoff games, and simple stochastic games: they all admit optimal memoryless strategies [41, 73, 31], they all are in NP $\cap$ coNP, and similar algorithmic approaches are used to solve them [46]. On the other hand, TROPICALLY CONVEX CONSTRAINTS is in some respects a problem more robust than MAX-ATOMS. For instance, TROPICALLY CONVEX CONSTRAINTS is a finite language CSP. On the contrary, MAX-ATOMS does not admit a finite basis, and any finite signature restriction of MAX-ATOMS is immediately in P by sampling.

Max-atoms does not have a polymorphism definition, i.e., the feasibility regions of MAX-ATOMS OVER $\mathbb{R}$ cannot be singled out among all the semilinear sets by means of a polymorphism condition. The class of semilinear relations preserved by all polymorphisms of the template for MAX-ATOMS over $\mathbb{R}$ turns out to be precisely the class of tropically convex semilinear relations [22].

The NP$\cap$coNP membership of TROPICALLY CONVEX CONSTRAINTS can be illustrated, for the special case of closed constraints (i.e., positive Boolean combinations of weak inequalities), by a duality statement.
Let $\mathcal{O}_n$ be the class of functions mapping $\left(\mathbb{Q}\cup\{+\infty\}\right)^n$ to $\mathbb{Q}\cup\{+\infty\}$ of either of the following forms

$$(x_1 \dots x_n) \mapsto \max(x_{j_1} + k_1 \dots x_{j_m} + k_m)$$
$$(x_1 \dots x_n) \mapsto \min(x_{j_1} + k_1 \dots x_{j_m} + k_m)$$
$$(x_1 \dots x_n) \mapsto \frac{\alpha_1 x_{j_1} + \dots + \alpha_m x_{j_m}}{\alpha_1 + \dots + \alpha_m} + k$$

where $k, k_i \in \mathbb{Q}$ and $\alpha_i \in \mathbb{Q}^{>0}$.

For any given vector of operators $\bar{o} \in \mathcal{O}_n^n$ consider the following satisfiability problems: the *primal* $P(\bar{o})$ and the *dual* $D(\bar{o})$

$$P(\bar{o})\colon \begin{cases} \bar{x} \in \mathbb{Q}^n \\ \bar{x} < \bar{o}(\bar{x}) \end{cases} \qquad D(\bar{o})\colon \begin{cases} \bar{y} \in \left(\mathbb{Q}\cup\{+\infty\}\right)^n \setminus \{+\infty\}^n \\ \bar{y} \geq \bar{o}(\bar{y}) \end{cases}$$

where $<$ and $\geq$ are component-wise.

▶ **Theorem 52** (Bodirsky–Mamino). *For any $\bar{o} \in \mathcal{O}_n^n$ one and only one of the problems $P(\bar{o})$ and $D(\bar{o})$ is satisfiable.*

The special case of Theorem 52 for max-atoms has been observed constraints already in [44].

7 Reducts of the Order of the Rationals

If we want to classify the CSP for first-order reducts Γ of an infinite structure Δ, the simplest structure to start with is the structure Δ with the empty signature. An example of such a reduct Γ is $(\mathbb{Q}; \neq, x = y \Rightarrow u = v)$; its CSP can be solved in polynomial time. On the other hand, the CSP for $(\mathbb{Q}; x = y \neq z \vee x \neq y = z)$ is NP-complete. The complexity of the CSP for such reducts has been classified: a reduct Γ of a countably infinite structure without any structure either has a constant polymorphism or a binary injective polymorphism, and CSP(Γ) is in P, or all polymorphisms of Γ can be written as an injection composed with projections, in which case CSP(Γ) is NP-complete [19]. Instead of giving further details of this fundamental result, we immediately step to a larger class that is of particular relevance for numeric domains, namely to reducts Γ of $(\mathbb{Q}; <)$. This class contains many interesting CSPs (some of which have been mentioned in the introduction), and classifying the complexity of CSP(Γ) is considerably more difficult.

We first state an older result, which provides a pre-classification of reducts of $(\mathbb{Q}; <)$ that plays an important role in the proof of the complexity classification. Let Γ be a first-order reduct of $(\mathbb{Q}; <)$ with finite relational signature. Cameron [27] proved that $\mathrm{Aut}(\Gamma)$ equals one out of the following five groups:

- $\mathrm{Aut}(\mathbb{Q}; <)$;
- $\mathrm{Aut}(\mathbb{Q}; \mathrm{Betw})$ where $\mathrm{Betw} = \{(x, y, z) \in \mathbb{Q}^3 \mid x < y < z \vee z < y < x\}$ is the betweenness relation that we have already seen in the introduction;
- $\mathrm{Aut}(\mathbb{Q}; \mathrm{Cycl})$ where $\mathrm{Cycl} = \{(x, y, z) \in \mathbb{Q}^3 \mid x < y < z \vee y < z < x \vee z < y < x\}$ is the so-called *cyclic order relation*;
- $\mathrm{Aut}(\mathbb{Q}; \mathrm{Sep})$ where Sep is the relation that contains all $(x, y, u, v) \in \mathbb{Q}^4$ such that the sets $[x, y] \setminus [u, v]$, $[x, y] \cap [u, v]$, and $[u, v] \setminus [x, y]$ are non-empty, where $[p, q]$ denotes the smallest interval that contains p and q;
- $\mathrm{Aut}(\mathbb{Q}; =)$, the symmetric group on $\mathbb{Q}$ consisting of *all* permutations of $\mathbb{Q}$.

Classifications of the automorphism groups of reducts Γ are too coarse for obtaining complexity results for $\mathrm{CSP}(\Gamma)$; we need to look at polymorphisms. Here are some binary operations over $\mathbb{Q}$ that are of particular importance when classifying the complexity of $\mathrm{CSP}(\Gamma)$ for reducts Γ of $(\mathbb{Q}; <)$. The first one is the maximum operation, $(x, y) \mapsto \max(x, y)$, which we have already seen in the introduction. The following operations are more difficult to describe; for illustrations, see Figure 1.

- Let $\mathrm{mx}\colon \mathbb{Q}^2 \to \mathbb{Q}$ be any operation that satisfies

$$\begin{aligned}\mathrm{mx}(x, y) < \mathrm{mx}(x', y') \Leftrightarrow {} & \min(x, y) < \min(x', y') \\ & \vee (\min(x, y) = \min(x', y') = x' = y' < \max(x, y)).\end{aligned}$$

  The relations $\leq$ and $\neq$ are not preserved by mx; however, mx preserves for example the relation $X := \{(x, y, z) \in \mathbb{Q}^3 \mid x = y < z \vee x = z < y \vee y = z < x\}$.
- Let $\mathrm{mi}\colon \mathbb{Q}^2 \to \mathbb{Q}$ be any operation that satisfies

$$\begin{aligned}\mathrm{mi}(x, y) < \mathrm{mi}(x', y') \Leftrightarrow {} & \min(x, y) < \min(x', y') \\ & \vee (\min(x, y) = \min(x', y') = x < x').\end{aligned}$$

  Relations preserved by mi are for instance $\leq$, $\neq$, and the relation of arity four given by $x \neq u \vee y < u \vee z \leq u$; a syntactic description of all the relations with a first-order definition over $(\mathbb{Q}; <)$ that are preserved by mi, due to Michał Wrona, can be found in [14].
- Let $\mathrm{ll}\colon \mathbb{Q}^2 \to \mathbb{Q}$ be any operation that satisfies

$$\begin{aligned}\mathrm{ll}(x, y) < \mathrm{ll}(x', y') \Leftrightarrow {} & \min(x, y) < \min(x', y') \\ & \vee (\min(x, y) = \min(x', y') \wedge x < x') \\ & \vee (\min(x, y) = \min(x', y') = x < y').\end{aligned}$$

  (Here we slightly deviate from the terminology Bodirsky and Kára [20]; the operation ll that we present here has a simpler behaviour that relates more clearly to the operations mx and mi, but the smallest polymorphism clone of a reduct of $(\mathbb{Q}; <)$ that contains it is the same as for the operation described in [20].) The relation ll preserves $\neq$, $\leq$, but also the relations defined by $x = y \Rightarrow u = v$ and $x > \min(y, z)$ that we have encountered before.

In all cases, it can be shown that such functions exist.

Each of these operations f has a *dual* f^*, defined by $f^*(x, y) := -f(-x, -y)$; for example, the minimum operation is the dual of the maximum operation. Finally, there is the constant

2	0	1	2
1	0	1	1
0	0	0	0
min	0	1	2

2	0	2	4
0	0	3	2
1	1	0	0
mx	0	1	2

2	2	5	6
1	2	3	4
0	0	1	1
mi	0	1	2

2	4	7	8
1	3	5	6
0	0	1	2
ll	0	1	2

Figure 1 Illustration for the operations min, mx, mi, and ll (in this order).

function whose image has cardinality one; clearly, having a constant polymorphism implies that Γ has the same CSP as a one-element structure, and hence CSP(Γ) is in P. But also if Γ has one of the other operations described above as a polymorphism, or one of their duals, then CSP(Γ) can be solved in polynomial time. We mention that the polynomial-time algorithms can also treat reducts Γ with an infinite signature when the relation symbols are represented by quantifier-free formulas in disjunctive normal form (recall that $(\mathbb{Q}; <)$ has quantifier elimination). We can now state the classification result (from [20]; also see [14]).

▶ **Theorem 53** (Bodirsky–Kára). *Let Γ be a reduct of $(\mathbb{Q}; <)$ with finite relational signature. Then Γ has* ll, min, mx, mi, *one of their duals, or a constant operation as polymorphism, and* CSP(Γ) *is in P, or* CSP(Γ) *is NP-hard.*

It is natural to ask which semilinear relations can be added to the tractable cases of this theorem so that the CSP of the resulting expanded structure is still tractable. The polymorphism max and its algorithmic relevance in this context has already been discussed earlier in Section 6. On the other hand, the operations ll, mx, and mi can be adapted to preserve additional relations that are definable in $(\mathbb{Q}; <, +, 1)$, and we quickly reach CSPs of open computational complexity; some will be listed in Section 10.

8 Reducts of the Successor Relation over the Integers

The structure $(\mathbb{Z}; \mathrm{Succ})$ of the integers with the successor relation is among the simplest structures that is not ω-categorical, and it is a reduct of most of the interesting structures over the integers, such as $(\mathbb{Z}; <)$ or $(\mathbb{Z}; +, 1)$. Hence, the class of reducts of these two more expressive structures includes the reducts of $(\mathbb{Z}; \mathrm{Succ})$, and following the bottom-up approach mentioned in the introduction, we study the CSPs of reducts of $(\mathbb{Z}; \mathrm{Succ})$ first.

Moreover, the structure $(\mathbb{Z}; \mathrm{Succ})$ has the same CSP as the structure $(\mathbb{Q}; x = y + 1)$, and reducts of $(\mathbb{Z}; \mathrm{Succ})$ have the same CSP as the corresponding reducts of $(\mathbb{Q}; x = y + 1)$. Hence, even when we want to classify the complexity of the CSP for reducts of $(\mathbb{Q}; +, 1)$ we have to classify the complexity of CSPs for reducts of $(\mathbb{Z}; \mathrm{Succ})$.

Let Γ be a first-order reduct of $(\mathbb{Z}; \mathrm{Succ})$ with finite relational signature. We want to describe the border between those Γ whose CSP can be solved in polynomial time and those whose CSP is NP-hard, because we believe that the shape of this description could be paradigmatic for complexity classifications of larger classes of CSPs over numeric domains. In order to state the classification, we introduce certain binary operations.

▶ **Definition 54.** For $d \in \mathbb{Z}$, $d \geq 1$, the *d-modular max* is the binary operation $\max_d \colon \mathbb{Z}^2 \to \mathbb{Z}$ defined by

$$\max_d(x, y) := \begin{cases} \max(x, y) & \text{if } x \equiv y \text{ modulo } d \\ x & \text{otherwise.} \end{cases}$$

If Γ has a d-modular max as a polymorphism, for some $d \geq 1$, then CSP(Γ) can be solved in polynomial time [15]. In a nutshell, the idea is that the situation for $d > 1$ can be reduced to the situation for $d = 1$. For $d = 1$, we obtain the regular max operation, and we can solve $\mathrm{CSP}(\Gamma)$ using the sampling technique that has been described in Section 6.

We now describe another family of reducts of $(\mathbb{Z}; \mathrm{Succ})$ whose CSP can be solved in polynomial time. Note that the structure $(\mathbb{Q}; \{(x, y) \mid x = y+1\})$ is isomorphic to $(\mathbb{Q}; \{(x, y) \mid x = y + 1\})^2$; let si be any isomorphism. It can be shown that a relation with a first-order definition in $(\mathbb{Q}; \{(x, y) \mid x = y + 1\})$ is preserved by i if and only if it has a Horn definition over $(\mathbb{Q}; s)$ where s is the successor *function*. With this description, it is not difficult to come up with an algorithm for reducts of $(\mathbb{Z}; \mathrm{Succ})$ with i as a polymorphism.

We would like to state that all other first-order reducts Γ of $(\mathbb{Z}; \mathrm{Succ})$ with finite relational signature have an NP-hard CSP; an in fact, this is true when Succ has a primitive positive definition in Γ. But otherwise, unfortunately, life is not as simple as that. It might be that Γ has the same CSP as some other reduct Γ' of $(\mathbb{Z}; \mathrm{Succ})$, and that Γ' has max as a polymorphism. Allowing such changes in the choice of the template greatly helps in classification projects. When Γ is ω-categorical, then there is a good theory for finding the (up to isomorphism unique) 'nicest' template to work with; we have to refer to [13, 17] for a discussion. The main point is that in these nicer templates, many of the basic relations are primitive positive definable. For reducts Γ of $(\mathbb{Z}; \mathrm{Succ})$, we can replace Γ by some structure Γ' with the same CSP such that Succ is primitive positive definable in Γ', unless Γ is of a 'degenerate' form; see Theorem 55 for a formal statement. Unlike the ω-categorical situation, we do not have an a-priori justification for this phenomenon. With this perspective, we can now phrase the complete classification statement from [24].

▶ **Theorem 55.** *Let Γ be a reduct of $(\mathbb{Z}; \mathrm{Succ})$ with finite signature. Then there exists a structure Δ such that $\mathrm{CSP}(\Delta)$ equals $\mathrm{CSP}(\Gamma)$ and one of the following cases applies.*

1. *Δ has a finite domain, and Feder and Vardi conjectured that $\mathrm{CSP}(\Delta)$ is in P or NP-complete.*
2. *Δ is a reduct of $(\mathbb{Q}; =)$, and $\mathrm{CSP}(\Delta)$ is either in P or NP-complete by Theorem 53.*
3. *Δ is a reduct of $(\mathbb{Z}; \mathrm{Succ})$ and Succ is primitive positive definable in Δ. In this case, if Δ has a d-modular* max *or a d-modular* min *polymorphism, then $\mathrm{CSP}(\Delta)$ is in P; otherwise, $\mathrm{CSP}(\Delta)$ is NP-complete.*

9 The Unclassifiable

In this section we make essential use of the following theorem, which is due to Davis, Matiyasevich, Putnam, and Robinson.

▶ **Theorem 56** (See e.g. [80]). *A subset of $\mathbb{Z}$ is recursively enumerable if and only if it has a primitive positive definition in $(\mathbb{Z}; *, +, 1)$, the integers with addition and multiplication.*

▶ **Theorem 57.** *For every recursively enumerable problem $\mathcal{P}$ there exists a reduct Γ of $(\mathbb{Z}; *, +, 1)$ with finite relational signature such that $\mathrm{CSP}(\Gamma)$ is polynomial-time Turing equivalent to $\mathcal{P}$.*

Proof. Code $\mathcal{P}$ as a set L of natural numbers, viewing the binary encodings of natural numbers as bit strings. More precisely, $s \in \mathcal{P}$ if and only if the number represented in binary by the string $1s$ is in L. That is, we append the symbol 1 at the front so that for instance $00 \in \mathcal{P}$ and $01 \in \mathcal{P}$ correspond to different numbers in L. Now consider the structure $\Gamma := (\mathbb{Z}; S, D, L', N)$ where

- S is the binary relation defined by

$$S(x, y) \Leftrightarrow \big((y = x + 1 \wedge x \geq 0) \vee (x = y = -1)\big);$$

- D is the binary relation defined by

$$D(x,y) \Leftrightarrow \big((y = 2x \wedge x \geq 0) \vee (x = y = -1)\big);$$

- $L' := L \cup \{-1\}$;
- $N := \{0\}$.

Clearly, if $\mathcal{P}$ is recursively enumerable, then L and L' are recursively enumerable, too.

We have to verify that CSP(Γ) is polynomial time equivalent to $\mathcal{P}$. We first show that there is a polynomial-time reduction from $\mathcal{P}$ to CSP(Γ). View an instance of $\mathcal{P}$ as a number $n \geq 0$ as above, and let $\eta(x)$ be a primitive positive definition for $x = n$ in Γ. It is possible to find such a definition in polynomial time by repeatedly doubling ($y = x + x$) and incrementing ($y = x + 1$) the value 0 (this also follows from the more general Lemma 3). It is clear that n codes a yes-instance of $\mathcal{P}$ if and only if $\exists x(\eta(x) \wedge L'(x))$ is true in Γ.

To reduce CSP(Γ) to $\mathcal{P}$, we present a polynomial-time algorithm for CSP(Γ) that uses an oracle for $\mathcal{P}$ (so our reduction will be a polynomial-time Turing reduction). Let ϕ be an instance of CSP(Γ), and let H be the undirected graph whose vertices are the variables W of ϕ, and which has an edge between x and y if ϕ contains the constraint $S(x,y)$ or the constraint $D(x,y)$. Compute the connected components of H. If a connected component does not contain x with a constraint $N(x)$ in ϕ, then we can set all variables of that component to -1 and satisfy all constraints involving those variables.

Otherwise, suppose that we have a component C that does contain x_0 with a constraint $N(x_0)$. Observe that by connectivity, if there exists a solution, then all variables in C must take non-negative value. Consider the following linear system: for each constraint of the form $S(x,y)$ for $x, y \in C$ we add $y = x + 1$ and $x \geq 0$ to the system, and for each constraint of the form $D(x,y)$ for $x, y \in D$ we add $z = 2x$ and $x \geq 0$. Subject to $x_0 = 0$ this system has either one or no solution. As we have discussed earlier, one can check in polynomial time whether a linear system with 2 variables per constraint has an integer solution, and if there is no solution, the algorithm rejects. Otherwise, the algorithm assigns to each variable $x \in C$ its unique integer value, and if ϕ contains a constraint $L'(x)$, we call the oracle for $\mathcal{P}$ with the binary encoding of this value. If any of those oracle calls has a negative result, reject. Otherwise, we have found an assignment that satisfies all constraints, and accept. ◀

The universal-algebraic approach fails badly when it comes to analyzing the computational complexity of CSP(Γ) for the structure Γ from the proof of Theorem 57: the semi-lattice operation $(x,y) \mapsto \max(x,y)$ preserves Γ for all structures Γ considered in the previous proof, and from that we cannot draw any consequences for the computational complexity of CSP(Γ).

10 Next Steps for . . .

Constraint Satisfaction Problems over a numeric domain, such as $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$, for a fixed set of constraints that are first-order definable using addition and multiplication, provide an extremely rich class of natural and fundamental computational problems. A great variety of problems from the literature can be expressed in this way. Many of these problems have an open computational complexity status, and belong to the central topics of research in their respective fields.

The universal-algebraic approach, which has originally been developed for finite-domain constraint satisfaction, will most likely only be of limited help for studying the complexity of computational complexity of the famously open problems. However, this approach is useful for for relating these problems, for obtaining classification results, and for identifying new polynomial-time restrictions.

We state a list of concrete open problems, which we structure according to the numeric domain.

10.1 . . . the Integers

There is the obvious goal: classify the complexity of CSP(Γ) for first-order reducts Γ of $(\mathbb{Z}; <, +, 1)$. This being a very ambitious goal, we propose substeps and concrete relevant questions.

1. Classify the complexity of first-order reducts of $(\mathbb{Z}; \mathrm{Succ}, 0)$.
2. Classify the complexity of first-order reducts of $(\mathbb{Z}; +)$.
3. What is the complexity of $\mathrm{CSP}(\mathbb{Z}; \leq, \mathrm{Succ}, x = 2y)$?

10.2 . . . the Rationals

Again, there is the obvious goal: classify the complexity of CSP(Γ) for first-order reducts Γ of $(\mathbb{Q}; <, +, 1)$. A complete answer would involve the solution of long-standing open problems from the literature, e.g., the complexity of MEAN PAYOFF GAMES.

4. What is the complexity of CSP(Γ) for first-order reducts Γ of $(\mathbb{Q}; <, +, 1)$ with max as a polymorphism? Already containment in NP $\cap$ coNP is unclear.
5. What is the complexity of $\mathrm{CSP}(\mathbb{Q}; X, R_{\mathrm{Succ}})$ where $X := \{(x, y, z) \in \mathbb{Q}^3 \mid x = y < z \vee x = z < y \vee y = z < x\}$ is the relation from the introduction?
6. Classify the complexity of CSP(Γ) for first-order reducts Γ of $(\mathbb{Q}; +)$.

10.3 . . . the Reals

Is it possible to classify the complexity of CSP(Γ) for first-order reducts Γ of $(\mathbb{R}; +, *)$? This goal is at least as difficult as the initial goal from Section 10.2, since 1 and $<$ are first-order definable in $(\mathbb{R}; +, *)$, and since every reduct of $(\mathbb{R}; +, <, 1)$ has the same CSP as the corresponding reduct of $(\mathbb{Q}; +, <, 1)$.

7. What is the computational complexity of the problem $\mathrm{CSP}(\mathbb{R}, R_+; R_{=1}, +, x^2 \leq y)$ from Observation 32?
8. What is the computational complexity of the problem $\mathrm{CSP}(\mathbb{R}; R_+, R_{=1}, x^2 + y^2 \leq 1)$ from Observation 33?

The classification for first-order reducts of $(\mathbb{R}; +, *)$ is probably strictly more difficult, because a complete complexity classification might provide the solution to further famous computational problems of open complexity, e.g., the sums-of-square-roots problem or the feasibility problem for semidefinite programs.

10.4 . . . and the Complex Numbers

Also over the complex numbers, many fundamental questions are open. The obvious general question is whether we can classify CSP(Γ) for all first-order reducts of $(\mathbb{C}; +, *)$. Clearly, if Γ is a first-order reduct of $(\mathbb{R}^2; \leq, +, 1, i)$ (with the usual identification of $\mathbb{C}$ and $\mathbb{R}^2$), then CSP(Γ) can be reduced to a linear program feasibility problem. On the other hand, for characterizing the reducts Γ with an NP-hard CSP, it would be interesting to have a primitive positive version of the theorem of Marker and Pillay mentioned in the introduction. More precisely, we are interested in the following mathematical question.

9. Let $R \subseteq \mathbb{C}^n$ be the relation defined over $(\mathbb{C}; +, *)$ by $p(x_1, \dots, x_n) = 0$ for some non-linear polynomial p. Is there a primitive positive definition of complex multiplication in $(\mathbb{C}; +, 1, R)$?

Acknowledgement. The authors would like to thank the referee for the many comments that helped to improve the readability.

References

1 Xavier Allamigeon, Pascal Benchimol, Stéphane Gaubert, and Michael Joswig. Combinatorial simplex algorithms can solve mean payoff games. *SIAM J. Optim.*, 24(4):2096–2117, 2014.

2 Eric Allender, Peter Bürgisser, Johan Kjeldgaard-Pedersen, and Peter Bro Miltersen. On the complexity of numerical analysis. *SIAM J. Comput.*, 38(5):1987–2006, 2008/09. `doi:10.1137/070697926`.

3 Daniel Andersson and Peter Bro Miltersen. The complexity of solving stochastic games on graphs. In *Algorithms and Computation, 20th International Symposium, ISAAC 2009, Honolulu, Hawaii, USA, December 16-18, 2009. Proceedings*, pages 112–121, 2009.

4 A. Atserias and E. Maneva. Mean-payoff games and the max-atom problem, 2009. URL: `https://www.cs.upc.edu/~atserias/papers/mean-payoff-games-max-atom/mpma.pdf`.

5 D. Avis and V. Chvátal. Notes on bland's pivoting rule. In *Polyhedral Combinatorics*, pages 24–34. Springer Berlin Heidelberg, 1978.

6 L. Barto. The constraint satisfaction problem and universal algebra. *The Bulletin of Symbolic Logic*, 21(3):319–337, 2015.

7 Libor Barto and Marcin Kozik. Constraint satisfaction problems solvable by local consistency methods. *Journal of the ACM*, 61(1):3:1–3:19, 2014.

8 Libor Barto, Jakub Opršal, and Michael Pinsker. The wonderland of reflections. Preprint arXiv:1510.04521, 2015.

9 Libor Barto and Michael Pinsker. The algebraic dichotomy conjecture for infinite domain constraint satisfaction problems. In *Proceedings of LICS'16*, pages 615–622, 2016. Preprint arXiv:1602.04353.

10 Marc Bezem, Robert Nieuwenhuis, and Enric Rodríguez-Carbonell. The max-atom problem and its relevance. In *Proceedings of the International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*, pages 47–61, 2008.

11 Henrik Björklund and Sergei Vorobyov. Combinatorial structure and randomized subexponential algorithms for infinite games. *Theoretical Computer Science*, 349(3):347–360, 2005.

12 Lenore Blum, Felipe Cucker, Michael Shub, and Steve Smale. *Complexity and real computation.* Springer-Verlag, New York, 1998. With a foreword by Richard M. Karp.

13 Manuel Bodirsky. Cores of countably categorical structures. *Logical Methods in Computer Science*, 3(1):1–16, 2007.

14 Manuel Bodirsky. Complexity classification in infinite-domain constraint satisfaction. Mémoire d'habilitation à diriger des recherches, Université Diderot – Paris 7. Available at arXiv:1201.0856, 2012.

15 Manuel Bodirsky, Víctor Dalmau, Barnaby Martin, Antoine Mottet, and Michael Pinsker. Distance constraint satisfaction problems. *Information and Computation*, 247:87–105, 2016.

16 Manuel Bodirsky and Martin Grohe. Non-dichotomies in constraint satisfaction complexity. In Luca Aceto, Ivan Damgard, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *Proceedings of the International Colloquium*

on Automata, Languages and Programming (ICALP), Lecture Notes in Computer Science, pages 184–196. Springer Verlag, July 2008.

17 Manuel Bodirsky, Martin Hils, and Barnaby Martin. On the scope of the universal-algebraic approach to constraint satisfaction. *Logical Methods in Computer Science (LMCS)*, 8(3:13), 2012. An extended abstract that announced some of the results appeared in the proceedings of Logic in Computer Science (LICS'10).

18 Manuel Bodirsky, Peter Jonsson, and Timo von Oertzen. Essential convexity and complexity of semi-algebraic constraints. *Logical Methods in Computer Science*, 8(4), 2012. An extended abstract about a subset of the results has been published under the title *Semilinear Program Feasibility* at ICALP'10.

19 Manuel Bodirsky and Jan Kára. The complexity of equality constraint languages. *Theory of Computing Systems*, 3(2):136–158, 2008. A conference version appeared in the proceedings of Computer Science Russia (CSR'06).

20 Manuel Bodirsky and Jan Kára. The complexity of temporal constraint satisfaction problems. *Journal of the ACM*, 57(2):1–41, 2009. An extended abstract appeared in the Proceedings of the Symposium on Theory of Computing (STOC).

21 Manuel Bodirsky, Dugald Macpherson, and Johan Thapper. Constraint satisfaction tractability from semi-lattice operations on infinite sets. *Transaction of Computational Logic (ACM-TOCL)*, 14(4):1–30, 2013.

22 Manuel Bodirsky and Marcello Mamino. Max-closed semilinear constraints. In *Proceedings of CSR*, pages 88–101, 2016.

23 Manuel Bodirsky and Antoine Mottet. Reducts of finitely bounded homogeneous structures, and lifting tractability from finite-domain constraint satisfaction. In *Proceedings of LICS'16*, pages 623–632, 2016. Preprint available under ArXiv:1601.04520.

24 Manuel Bodirsky, Antoine Mottet, and Barnaby Martin. Constraint satisfaction problems over the integers with successor. In *Proceedings of ICALP*, pages 256–267, 2015. ArXiv:1503.08572.

25 V. G. Bodnarčuk, L. A. Kalužnin, V. N. Kotov, and B. A. Romov. Galois theory for Post algebras, part I and II. *Cybernetics*, 5:243–539, 1969.

26 Andrei A. Bulatov and Peter Jeavons. Algebraic structures in combinatorial problems. Technical report MATH-AL-4-2001, Technische Universität Dresden, 2001.

27 Peter J. Cameron. Transitivity of permutation groups on unordered sets. *Mathematische Zeitschrift*, 148:127–139, 1976.

28 John Canny. Some algebraic and geometric computations in PSPACE. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing (STOC)*, pages 460–467, New York, NY, USA, 1988. ACM.

29 T.-W. J. Chou and G. E. Collins. Algorithms for the solution of systems of linear Diophantine equations. *SIAM Journal on Computing*, 11:687–708, 1982.

30 George E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decompostion. In *Automata Theory and Formal Languages (2nd GI Conference Kaiserslautern)*, pages 134–183. Springer Berlin, Heidelberg, 1975.

31 Anne Condon. The complexity of stochastic games. *Information and Computation*, 96(2):203–224, 1992. doi:10.1016/0890-5401(92)90048-K.

32 Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms*. MIT Press, 2009. Third edition.

33 Mike Develin and Bernd Sturmfels. Tropical convexity. *Documenta Mathematica*, 9:1–27, 2004.

34 Andrzej Ehrenfeucht and Jan Mycielski. Positional strategies for mean payoff games. *International Journal of Game Theory*, 8(2):109–113, 1979.

35 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*, 28:57–104, 1999.

36 Jeanne Ferrante and Charles Rackoff. A decision procedure for the first order theory of real addition with order. *SIAM Journal on Computing*, 4(1):69–76, 1975.

37 Oliver Friedmann. A subexponential lower bound for zadeh's pivoting rule for solving linear programs and games. In *Integer Programming and Combinatoral Optimization: 15th International Conference (IPCO, New York)*, pages 192–206. Springer Berlin Heidelberg, 2011.

38 Oliver Friedmann, Thomas Dueholm Hansen, and Uri Zwick. Subexponential lower bounds for randomized pivoting rules for the simplex algorithm. In *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing (STOC)*, pages 283–292, New York, NY, USA, 2011. ACM. doi:10.1145/1993636.1993675.

39 Joseph A. Gallian. *Contemporary Abstract Algebra.* Brooks/Core, 2006.

40 M. R. Garey, R. L. Graham, and D. S. Johnson. Some NP-complete geometric problems. In *Eighth Annual ACM Symposium on Theory of Computing (Hershey, Pa., 1976)*, pages 10–22. Assoc. Comput. Mach., New York, 1976.

41 D. Gillette. Stochastic games with zero probabilities. *Contributions to the Theory of Games*, 3:179–187, 1957.

42 Christian Glaßer, Peter Jonsson, and Barnaby Martin. Circuit satisfiability and constraint satisfaction around skolem arithmetic. In *Pursuit of the Universal – 12th Conference on Computability in Europe, CiE 2016, Paris, France, June 27 – July 1, 2016, Proceedings*, pages 323–332, 2016.

43 Donald Goldfarb and William Y. Sit. Worst case behavior of the steepest edge simplex method. *Discrete Applied Mathematics*, 1(4):277–285, 1979. doi:10.1016/0166-218X(79)90004-0.

44 Dima Grigoriev and Vladimir V. Podolskii. Tropical effective primary and dual Nullstellensätze. In *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4-7, 2015, Garching, Germany*, pages 379–391, 2015.

45 V. A. Gurvich, A. V. Karzanov, and L. G. Khachiyan. Cyclic games and finding minimax mean cycles in digraphs. *Zh. Vychisl. Mat. i Mat. Fiz.*, 28(9):1407–1417, 1439, 1988.

46 Nir Halman. Simple stochastic games, parity games, mean payoff games and discounted payoff games are all LP-type problems. *Algorithmica*, 49(1):37–50, 2007. doi:10.1007/s00453-007-0175-3.

47 Thomas Dueholm Hansen and Uri Zwick. An improved version of the random-facet pivoting rule for the simplex algorithm. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 209–218, 2015.

48 J. William Helton and Jiawang Nie. Sufficient and necessary conditions for semidefinite representability of convex hulls and sets. *SIAM Journal on Optimization*, 20(2):759–791, 2009.

49 J. William Helton and Jiawang Nie. Semidefinite representation of convex sets. *Math. Program.*, 122(1):21–64, 2010.

50 J. William Helton and Victor Vinnikov. Linear matrix inequality representation of sets. *Communications on Pure and Applied Mathematics*, 60(5), 2007.

51 Dorit S. Hochbaum and Joseph Naor. Simple and fast algorithms for linear and integer programs with two variables per inequality. *SIAM Journal on Computing*, 23(6):1179–1192, 1994.

52 Wilfrid Hodges. *Model theory.* Cambridge University Press, 1993.

53 Peter Jeavons, David Cohen, and Marc Gyssens. Closure properties of constraints. *Journal of the ACM*, 44(4):527–548, 1997.

54 P. G. Jeavons and M. C. Cooper. Tractable constraints on ordered domains. *Artificial Intelligence*, 79(2):327–339, 1995.

55 R. G. Jeroslow. The simplex algorithm with the pivot rule of maximizing criterion improvement. *Discrete Mathematics*, 4(4):367–377, 1973.

56 J. P. Jones. Universal diophantine equation. *Journal of Symbolic Logic*, 47(3):549–571, 1982.

57 Peter Jonsson and Christer Bäckström. A unifying approach to temporal constraint reasoning. *Artificial Intelligence*, 102(1):143–155, 1998.

58 Peter Jonsson and Tomas Lööw. Computation complexity of linear constraints over the integers. *Artificial Intelligence*, 195:44–62, 2013.

59 Peter Jonsson and Johan Thapper. Constraint satisfaction and semilinear expansions of addition over the rationals and the reals. *Journal of Computer and System Sciences*, 82(5):912–928, 2016. `doi:10.1016/j.jcss.2016.03.002`.

60 Gil Kalai. A subexponential randomized simplex algorithm (extended abstract). In *Proceedings of the Twenty-fourth Annual ACM Symposium on Theory of Computing (STOC)*, pages 475–482, New York, NY, USA, 1992. ACM. `doi:10.1145/129712.129759`.

61 Ravindran Kannan and Achim Bachem. Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix. *SIAM J. Comput.*, 8(4):499–507, 1979.

62 N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984. `doi:10.1007/BF02579150`.

63 Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations*, pages 85–103, Boston, MA, 1972. Springer US.

64 L. Khachiyan. A polynomial algorithm in linear programming. *Doklady Akademii Nauk SSSR*, 244:1093–1097, 1979.

65 Victor Klee and George J. Minty. How good is the simplex algorithm? In *Inequalities, III (Proc. Third Sympos., Univ. California, Los Angeles, Calif., 1969; dedicated to the memory of Theodore S. Motzkin)*, pages 159–175. Academic Press, New York, 1972.

66 Igor Klep. An exact duality theory for semidefinite programming based on sums of squares. *Mathematics of Operations Research*, 38:569–590, 2013.

67 Pascal Koiran. Hilbert's Nullstellensatz is in the polynomial hierarchy. *J. Complexity*, 12(4):273–286, 1996.

68 Pascal Koiran. Hilbert's Nullstellensatz is in the polynomial hierarchy. Technical Report DIMACS Technical report 96-27, Center for Discrete Mathematics and Theoretical Computer Science, July 1996.

69 M. Koubarakis. Tractable disjunctions of linear constraints: Basic results and applications to temporal reasoning. *Theoretical Computer Science*, 266:311–339, 2001.

70 J. Kratochvil and J. Matousek. Intersection graphs of segments. *Journal of Combinatorial Theory, Series B*, 62(2):289–315, 1994. `doi:10.1006/jctb.1994.1071`.

71 J. C. Lagarias. The computational complexity of simultaneous Diophantine approximation problems. *SIAM Journal on Computing*, 14(1):196–209, 1985.

72 H. W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, 1983.

73 Thomas M. Liggett and Steven A. Lippman. Stochastic games with perfect information and time average payoff. *SIAM Review*, 11(4):604–607, 1969.

74 Kenneth L. Manders and Leonard Adleman. NP-complete decision problems for binary quadratics. *Journal of Computer and System Sciences*, 16(2):168–184, 1978.

75 David Marker, Yaacov Peterzil, and Anand Pillay. Additive reducts of real closed fields. *Journal of Symbolic Logic*, 57(1):109–117, 1992.

76 David Marker and Anand Pillay. Reducts of $(C, +, *)$ which contain $+$. *J. Symb. Log.*, 55(3):1243–1251, 1990.

77 Gary A. Martin. Definability in reducts of algebraically closed fields. *Journal of Symbolic Logic*, 53(1):188–199, 1988.

78 Ju. V. Matijasevic. Some purely mathematical results inspired by mathematical logic. In *Foundations of Mathematics and Computability Theory*, pages 121–127. Reidel, Dordrecht, 1977.

79 Yuri Matiyasevich. Enumerable sets are Diophantine. *Doklady Akademii Nauk SSSR*, 191:279–282, 1970.

80 Yuri V. Matiyasevich. *Hilbert's Tenth Problem.* MIT Press, Cambridge, Massachusetts, 1993.

81 J. Matoušek, M. Sharir, and E. Welzl. A subexponential bound for linear programming. *Algorithmica*, 16(4):498–516, 1996. `doi:10.1007/BF01940877`.

82 Nimrod Megiddo. Towards a genuinely polynomial algorithm for linear programming. *SIAM Journal on Computing*, 12(2):347–353, 1983.

83 Nimrod Megiddo. Linear programming in linear time when the dimension is fixed. *J. ACM*, 31(1):114–127, January 1984. `doi:10.1145/2422.322418`.

84 Rolf H. Möhring, Martin Skutella, and Frederik Stork. Scheduling with and/or precedence constraints. *SIAM Journal on Computing*, 33(2):393–415, 2004.

85 Bernhard Nebel and Hans-Jürgen Bürckert. Reasoning about temporal relations: A maximal tractable subclass of Allen's interval algebra. *Journal of the ACM*, 42(1):43–66, 1995.

86 Ya'acov Peterzil. Reducts of some structures over the reals. *J. Symb. Log.*, 58(3):955–966, 1993.

87 Motakuri V. Ramana. An exact duality theory for semidefinite programming and its complexity implications. *Mathematical Programming*, 77:129–162, 1997.

88 James Renegar. On the computational complexity and geometry of the first-order theory of the reals. Part I: Introduction. Preliminaries. The geometry of semi-algebraic sets. The decision problem for the existential theory of the reals. *Journal of Symbolic Computation*, 13(3):255–299, 1992. `doi:10.1016/S0747-7171(10)80003-3`.

89 James Renegar. On the computational complexity and geometry of the first-order theory of the reals. Part II: The general decision problem. Preliminaries for quantifier elimination. *Journal of Symbolic Computation*, 13(3):301–327, 1992. `doi:10.1016/S0747-7171(10)80004-5`.

90 James Renegar. On the computational complexity and geometry of the first-order theory of the reals. Part III: Quantifier elimination. *Journal of Symbolic Computation*, 13(3):329–352, 1992. `doi:10.1016/S0747-7171(10)80005-7`.

91 Marcus Schaefer. Realizability of graphs and linkages. In *Thirty Essays on Geometric Graph Theory*, pages 461–482. Springer New York, 2013.

92 Marcus Schaefer and Daniel Štefankovič. Fixed points, Nash equilibria, and the existential theory of the reals. *Theory of Computing Systems*, pages 1–22, 2015.

93 Claus Scheiderer. Semidefinite representation for convex hulls of real algebraic curves, 2012. Preprint, arXiv:1208.3865.

94 Alexander Schrijver. *Theory of Linear and Integer Programming.* Wiley – Interscience Series in Discrete Mathematics and Optimization, 1998.

95 Alexandra Shlapentokh. *Hilbert's tenth problem: Diophantine classes and extensions to global fields*, volume 7 of *New Mathematical Monographs.* Cambridge University Press, 2007.

96 Sergey P. Tarasov and Mikhail N. Vyalyi. Semidefinite programming and arithmetic circuit evaluation. *Discrete Applied Mathematics*, 156(11):2070–2078, 2008. In Memory of Leonid Khachiyan (1952–2005).

97 Éva Tardos. A strongly polynomial algorithm to solve combinatorial linear programs. *Operations Research*, 34(2):250–256, 1986. URL: `http://www.jstor.org/stable/170819`.

98 Alfred Tarski. A decision method for elementary algebra and geometry: Prepared for publication with the assistance of J. C. C. McKinsey. Technical report, RAND Corporation, Santa Monica, CA, 1951.

99 Lou van den Dries. *Tame topology and o-minimal structures*, volume 248 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, Cambridge, 1998. `doi:10.1017/CBO9780511525919`.

100 Peter van Emde Boas. Another NP-complete partition problem and the complexity of computing short vectors in a lattice. Technical Report Technical Report 81-04, Mathematische Instituut, University of Amsterdam, 1981.

101 Marc Vilain, Henry Kautz, and Peter van Beek. Constraint propagation algorithms for temporal reasoning: A revised report. *Reading in Qualitative Reasoning about Physical Systems*, pages 373–381, 1989.

102 Sergei Vorobyov. Cyclic games and linear programming. *Discrete Applied Mathematics*, 156(11):2195–2231, 2008. In Memory of Leonid Khachiyan (1952–2005). `doi:10.1016/j.dam.2008.04.012`.

103 Margaret H. Wright. The interior-point revolution in optimization: history, recent developments, and lasting consequences. *Bull. Amer. Math. Soc.*, 42:39–56, 2005.

104 Uri Zwick and Mike Paterson. The complexity of mean payoff games on graphs. *Theor. Comput. Sci.*, 158(1&2):343–359, 1996.

Hybrid Tractable Classes of Constraint Problems*

Martin C. Cooper[1] **and Stanislav Živný**[2]

1 **IRIT, University of Toulouse III, Toulouse, France**
cooper@irit.fr
2 **Dept. of Computer Science, University of Oxford, Oxford, UK**
standa.zivny@cs.ox.ac.uk

Abstract

We present a survey of complexity results for *hybrid* constraint satisfaction problems (CSPs) and valued constraint satisfaction problems (VCSPs). These are classes of (V)CSPs defined by restrictions that are not exclusively language-based or structure-based.

1998 ACM Subject Classification F.4.1 Mathematical Logic, G.2.1 Combinatorics, G.2.2 Graph Theory, F.1.3 Complexity Measures and Classes, G.1.6 Optimization

Keywords and phrases Constraint satisfaction problems, Optimisation, Tractability

Digital Object Identifier 10.4230/DFU.Vol7.15301.113

1 Introduction

A fundamental challenge in computer science is to map out the frontier of the complexity class P, the class of decision problems that can be solved in polynomial time. The constraint satisfaction problem (CSP) is a generic combinatorial problem which includes in a natural way many important NP-complete problems such as SAT or graph-colouring. The valued constraint satisfaction problem (VCSP) can be seen as an even richer language than the CSP since it provides a general framework in which to express both constraint satisfaction problems and constrained optimisation problems. The identification of tractable classes of generic problems, such as the (V)CSP, has led not only to a deeper understanding of tractability, but also to wider application areas of well-known algorithms. Indeed, recent research has shown that very few algorithmic techniques suffice to solve all tractable language-based classes of (V)CSPs [1, 35, 40].

Many real-world problems can be modelled as classical and well-studied NP-complete problems, such as SAT, CSP or VCSP. This has the advantage that generic solvers exist which are efficient on many instances, but has the disadvantage of not taking into account specificities of the particular problem which could perhaps guarantee the existence of a polynomial-time algorithm. Obvious specificities include the type of constraints or cost functions that can occur or the structure of the hypergraph of constraint scopes. Much research effort has been devoted to identifying tractable language-based classes [2, 5] or tractable structural classes [32], with many notable successes. However, it is natural to ask whether interesting classes of instances can be defined in other ways.

* The authors were supported by EPSRC grant EP/L021226/1. Stanislav Živný was supported by a Royal Society University Research Fellowship. This work was partly done while the second author was visiting the Simons Institute for the Theory of Computing at UC Berkeley. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 714532). The paper reflects only the authors' views and not the views of the ERC or the European Commission. The European Union is not liable for any use that may be made of the information contained therein.

One way of defining classes of instances is by simultaneously placing restrictions both on the language of possible constraints (or cost functions) and on the structure of the hypergraph of constraint (cost function) scopes. Such classes are known as hybrid classes, and by extension all classes which are not exclusively language-based nor structure-based are also known as hybrid [24]. This larger meaning is the one we apply in this chapter.

As an example of a hybrid tractable class, consider a company which wishes to give bonuses to its employees (chosen from a finite set of possible amounts). Each employee has a grade, with higher grades corresponding to more important posts. Some, but not all, employees have an immediate boss to whom they report. In this case, the employee and the immediate boss must receive bonuses such that the sum of the bonuses of the employee and the boss is bounded above and below by a specified amount. On the other hand, if an employee has no immediate boss, then the rule is that they must not receive a bigger bonus than anyone at a higher grade. We will see, in Example 1, that this bonus-assignment problem falls in a hybrid tractable class.

As another example, consider the same company which now wants to assign staff to a project, minimising total salary costs while respecting constraints concerning the minimum number of personnel from each section, the maximum total number of staff on the project, as well as the availability of each member of staff. Again, we will see, in Example 4, that this problem falls into a hybrid tractable class

An important way to classify work on hybrid tractability is how classes of instances are defined. We consider the following ways of defining a class of instances of (V)CSPs:

- independent restrictions on both the language of constraints (or cost functions) and on the structure of the instance.
- excluding generic sub-instances (known as forbidden patterns).
- properties that are required to hold *after* a preprocessing operation, such as establishing a certain level of consistency, has been performed.
- graph properties of the (weighted) microstructure of the instance.
- instances which are so strongly constrained that this implies a polynomial bound on search-tree size (or, on the contrary, so weakly constrained that there is always a solution).

Historically, different hybrid classes have been identified by attempting to generalise language or structural restrictions, or to determine a large class of instances solved by a particular algorithmic technique, or to translate known results from another field, usually graph theory, to (V)CSPs. The field of hybrid tractability has not yet reached maturity and is notably lacking a general theory which would allow us to express all the above types of restrictions, together with language and structural restrictions, in a common language. Such a unified language would no doubt lead to a greater understanding and new applications.

2 Independent Language and Structure Restrictions

A *constraint satisfaction problem* (CSP) instance I is given by a triple $\langle X, D, C\rangle$, where $X = \{X_1, \ldots, X_n\}$ is a finite set of variables, D is a finite set of values, and C is a finite set of constraints. Each constraint $c \in C$ is a pair $\langle \mathbf{v}, R\rangle$, where $\mathbf{v}$, the constraint *scope*, is a list of k variables from X and $R \subseteq D^k$, the constraint *relation*, is a k-ary relation on D. We call k the *arity* of the constraint. We note that different constraints within the same instance can have different arity. The question is whether there is an assignment of values to the variables that satisfies all the constraints. More formally, to decide whether there is an assignment $s : X \to D$ such that for every $c \in C$ with $c = (\mathbf{v}, R)$ and $\mathbf{v} = (X_{i_1}, \ldots, X_{i_k})$, we have $\langle s(X_{i_1}), \ldots, s(X_{i_k})\rangle \in R$.

In language-based classes of CSPs, one restricts the set of constraint relations R on D that can appear in any instance. A finite set of relations on a fixed finite set D is called a *constraint language* and we denote by $\mathrm{CSP}(\Gamma)$ the class of CSP instances in which all constraint relations belong to Γ.

In structure-based classes of CSPs, one restricts the type of interactions of the constraint scopes s (by restricting the hypergraph of the constraint scopes s on X) that can appear in any instance.

We remark that an equivalent definition of CSPs is the following: given two relational structures A and B, is there a homomorphism from A to B? Language-based CSPs correspond to fixing B to a single relational structure (corresponding to a constraint language) whereas structure-based CSPs correspond to requiring $A \in \mathcal{A}$ for some (infinite) class $\mathcal{A}$ of relational structures [34].

In this section we will discuss known results on the complexity of CSPs that impose independent restrictions on the structure of the instance and on the constraint language.

2.1 Planarity

A constraint language is called *Boolean* if the domain D is equal to $\{0, 1\}$.

The *incidence graph* of a CSP instance $I = \langle X, D, C\rangle$ has $X \cup C$ as its vertex set and (X_i, c) is an edge, for $X_i \in X$ and $c \in C$, if $X_i \in \mathbf{v}$ where $c = (\mathbf{v}, R)$.

For a Boolean constraint language Γ, we denote by $\mathrm{CSP}_p(\Gamma)$ the set of CSP instances from $\mathrm{CSP}(\Gamma)$ with *planar* incidence graphs and with the condition that, for each constraint in the instance, the variables in the scope of the constraint appear in the clockwise order (in some fixed planar embedding).

The complexity of Boolean planar language-restricted CSPs has recently been established. First, it was shown that, apart from Boolean constraint languages Γ where $\mathrm{CSP}(\Gamma)$ is tractable (and thus also $\mathrm{CSP}_p(\Gamma)$ is tractable), $\mathrm{CSP}_p(\Gamma)$ is intractable unless Γ is an *even Δ-matroid* [26]. Secondly, the tractability of $\mathrm{CSP}_p(\Gamma)$ for even Δ-matroids was shown [38]. In order to define even Δ-matroids we need some further notation.

We use $\oplus$ for the exclusive or. For a tuple t over $\{0, 1\}$, we denote by $\bar{t}$ the tuple obtained from t by flipping all values; i.e., $\bar{t} = t \oplus (1, \ldots, 1)$. We say that a relation R is *self-complementary* if for every $t \in R$ we have $\bar{t} \in R$. For a self-complementary R, we denote by dR the relation $\{(t_1 \oplus t_2, t_2 \oplus t_3, \ldots, t_k \oplus t_1) \mid (t_1, \ldots, t_k) \in R\}$. We say that Γ is self-complementary if every $R \in \Gamma$ is self-complementary. We define $d\Gamma = \{dR \mid R \in \Gamma\}$. A set $M \subseteq \{0, 1\}^k$ is a *Δ-matroid* if for all $\langle t_1, \ldots, t_k\rangle, \langle t'_1, \ldots, t'_k\rangle \in M$ and every $1 \leq i \leq k$ with $t_i \neq t'_i$ there is $1 \leq j \leq k$ with $t_j \neq t'_j$ such that t with t_i and t_j flipped also belongs to M. (We also allow the case of $i = j$, in which case the new tuple that is required to belong to M is obtained from t by flipping the ith position.) A Δ-matroid M is called *even* if all tuples in M have the same parity of the number of 1s. (Note that in this case i and j cannot be the same in the definition of Δ-matroids as this would change the parity of 1s.) Finally, a Boolean constraint language is called an even Δ-matroid if Γ is self-complementary and each relation in $d\Gamma$ is an even Δ-matroid.

We now give an example of a constraint language that can be used to model the perfect matching problem in graphs as a planar CSP [26]. Let $M_k \subseteq \{0, 1\}^k$ be the relation containing the k-tuples in which precisely one coordinate is set to one (and all others are set to zero). It is easy to check that M_k is an even Δ-matroid for every k. Let Γ_{pm} be the constraint language on $\{0, 1\}$ with $d\Gamma = \cup_{k \geq 1} M_k$. It can be checked that Γ is self-complementary and by definition $d\Gamma$ is an even Δ-matroid. Hence Γ_{pm} is an even Δ-matroid. For a graph $G = (V, E)$, we construct an instance $I_G = \langle E, \{0, 1\}, C\rangle$ of Boolean $\mathrm{CSP}(\Gamma_{\mathsf{pm}})$ with a constraint $\langle\langle e_1, \dots, e_k\rangle, M_k\rangle \in C$ for every degree-k vertex of G incident to edges $e_1, \dots, e_k$. It is clear that perfect matchings in G are in 1-to-1 correspondence with satisfying assignments to I_G.

2.2 Bounded Occurrence

Another natural restriction is that of bounded occurrence of variables in the constraints. We denote by $\mathrm{CSP}_k(\Gamma)$ the class of instances of $\mathrm{CSP}(\Gamma)$ in which every variable appears in at most k constraints. Feder showed that for Boolean constraint languages that contain constants, $\mathrm{CSP}_3(\Gamma)$ is as hard as $\mathrm{CSP}(\Gamma)$ [27]. Here a constant is a unary singleton relation. On the Boolean domain $D = \{0, 1\}$, there are only two constants $c_0 = \{(0)\}$ and $c_1 = \{(1)\}$.

Boolean CSPs in which every variable appears *exactly* in two constraints are called *edge* CSPs in [38] but have been long known as *Holant* problems in the counting community. We denote by $\mathrm{CSP}_e(\Gamma)$ the set of CSP instances from $\mathrm{CSP}(\Gamma)$ in which every variable appears in exactly two constraints. (The case of a variable appearing in *at most* two constraints, $\mathrm{CSP}_2(\Gamma)$, can be reduced to this case [27]). Feder showed that if Γ is a Boolean constraint language including constants such that $\mathrm{CSP}(\Gamma)$ is intractable then $\mathrm{CSP}_e(\Gamma)$ is intractable unless Γ is a Δ-matroid. Tractability has been shown for several special classes of Δ-matroids [27, 21] and most recently for even Δ-matroids [38] (where the reader can find more references). For instance, $\mathrm{CSP}_e(\Gamma_{\mathsf{pm}})$ captures precisely the perfect matching problem in graphs.

2.3 Lifted Languages

Two recent papers [39, 52] study certain hybrid classes of CSPs in which the algebraic machinery developed for the computational complexity of constraint languages is (partially) applicable. In particular, it has been shown that the complexity of the class of $\mathrm{CSP}(\Gamma)$ instances in which the structure (hypergraph of constraint scopes) is closed under inverse homomorphisms can be shifted to the analysis of the complexity of a $\mathrm{CSP}(\Gamma')$ instance, where Γ' is a new, “lifted” language [39, 52]. A class of structures $\mathcal{H}$ is closed under inverse homomorphisms if whenever there is a homomorphism from R' to $R \in \mathcal{H}$ then $R' \in \mathcal{H}$. Examples of such structures include classes of acyclic or k-colourable graphs. Non-examples include classes of planar or perfect graphs.

3 Forbidden Patterns

The notion of forbidden pattern is based on the idea that local properties of an instance may guarantee tractability and that a natural local property is the exclusion of those sub-instances which are obstructions to polynomial-time solution algorithms. Classifying graphs by excluding substructures is classical in graph theory and a CSP instance can be represented by a labelled graph, known as its microstructure, so this approach has the advantage that it sometimes allows us to use known results from graph theory.

For the moment, the study of forbidden patterns has been mostly limited to binary CSPs with no structure on the variables or domain values except possibly a total order. In this section, we begin with some formal definitions before presenting certain tractable classes defined by forbidden patterns. We then go on to consider extensions to non-binary CSPs and explore classes defined by excluding patterns as topological minors.

3.1 Definitions

A *binary CSP instance* requires the assignment of values from some specified finite domain D to a finite set X of variables $\{X_1, \ldots, X_n\}$. Each variable X_i has its own domain of possible values $\mathcal{D}(X_i) \subseteq D$. Without loss of generality, in this section we assume that each pair of variables, $X_i, X_j \in X$ is constrained by a constraint relation R_{ij}. A constraint is *non-trivial* if it is not the Cartesian product of the domains of the two variables. A *solution* to a binary CSP instance is an assignment s of values to variables, such that, for each constraint R_{ij}, $\langle s(X_i), s(X_j)\rangle \in R_{ij}$.

The *constraint graph* of an instance I is $G_I = \langle V_I, E_I\rangle$, where $V_I = X$ is the set of variables of I and E_I is the set of pairs $\{X_i, X_j\}$ for which R_{ij} is non-trivial. The instance I is *arc consistent* if $\forall i \neq j \in \{1, \ldots, n\}$, $\forall a \in \mathcal{D}(X_i)$, $\exists b \in \mathcal{D}(X_j)$ such that $\langle a, b\rangle \in R_{ij}$. The instance I is *directional arc consistent* if $\forall i < j \in \{1, \ldots, n\}$, $\forall a \in \mathcal{D}(X_i)$, $\exists b \in \mathcal{D}(X_j)$ such that $\langle a, b\rangle \in R_{ij}$.

One possible presentation of a binary CSP instance is as a labelled graph whose vertices are the set of possible variable-value assignments. This (labelled) graph is known as the (coloured) *microstructure* [37, 10, 50, 6, 43]. An n-variable binary CSP instance I in this microstructure presentation is an n-partite graph $\langle A_1, \ldots, A_n, E^+\rangle$, where the ith part A_i corresponds to the set of possible assignments $\langle X_i, a\rangle$ to variable X_i and there is an edge in E^+ between $\langle X_i, a\rangle \in A_i$ and $\langle X_j, b\rangle \in A_j$ if and only if $(a, b) \in R_{ij}$. We refer to individual variable-value assignments, such as $\langle X_i, a\rangle$ (i.e., the vertices of this n-partite graph) as *points*. If v is some variable of the instance we use the notation A_v to represent the set of possible assignments to v. (Sets A_v are sometimes called *potatoes*.) Thus A_i and A_{X_i} are synonyms.

An instance I can also be presented as a negative microstructure which is the n-partite labelled graph $\langle A_1, \ldots, A_n, E^-\rangle$, where there is an edge in E^- between points $p \in A_i$ and $q \in A_j$ (for $i \neq j$) if and only if there is no edge between p and q in E^+.

We refer to edges in E^+ as *positive* edges and edges in E^- as *negative* edges. We now generalise the notion of microstructure and negative microstructure to obtain *patterns*: a pattern is a labelled n-partite graph which has a set of positive edges, E^+, and a set of negative edges, E^-.

A binary CSP instance can be seen as a special kind of pattern where the parts correspond to the variables of the instance and there is exactly one positive or negative edge between each pair of possible assignments to each pair of distinct variables. Positive edges connect assignments that are allowed by the constraint on the corresponding pair of variables, and negative edges connect assignments that are disallowed by this constraint.

A *pattern* $P = \langle A_1, \ldots, A_n, E^+, E^-\rangle$ is a partially specified instance: there may be pairs of points $p \in A_i, q \in A_j$ (with $i \neq j$) such that there is neither a positive edge nor a negative edge between p and q. A point is said to be *isolated* if it does not belong to any edges in E^+ or E^-. We do not specifically disallow the possibility that two points in a pattern are joined by both positive and negative edges: such patterns cannot occur as a subpattern in an instance but can occur as a topological minor (c.f. Section 3.6).

In order that pattern exclusion be natural we define a pattern P as occurring in another pattern Q if, after arbitrary renaming and then possible merging of points, we get a substructure of Q.

A pattern $P' = \langle A'_1, \ldots, A'_n, E'^+, E'^-\rangle$ is a *homomorphic image* of a pattern $P = \langle A_1, \ldots, A_n, E^+, E^-\rangle$ if there exists a surjective mapping $f : \bigcup_{i=1}^n A_i \to \bigcup_{i=1}^n A'_i$ such that

- $\forall p, q \in \bigcup_{k=1}^n A_k$, p and q belong to the same part A_i if and only if $f(p)$ and $f(q)$ belong to the same part A'_j,

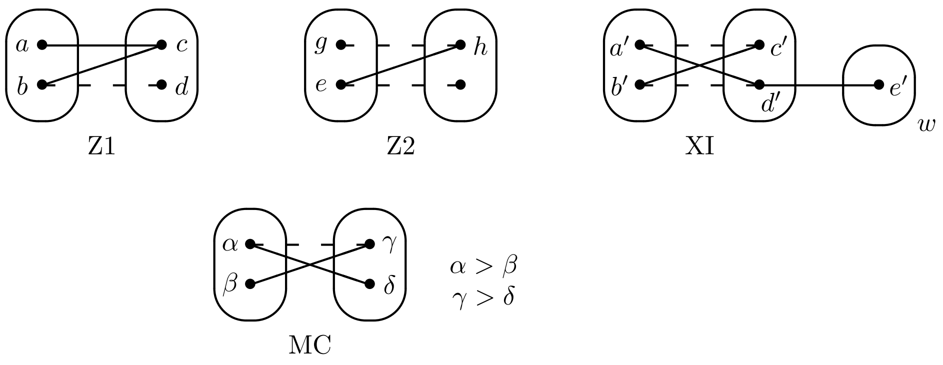


Figure 1 Four patterns.

- $\forall i, j \in \{1, \ldots, n\}$ with $i \neq j$, $\forall p \in A_i$, $\forall q \in A_j$: $\{p, q\} \in E^+ \Rightarrow \{f(p), f(q)\} \in E'^+$ and $\{p, q\} \in E^- \Rightarrow \{f(p), f(q)\} \in E'^-$.

Note that forming a homomorphic image of a pattern allows the parts to be renamed, and points p, q within the same part to be merged (provided there is no third point r such that $\{p, r\}$ and $\{q, r\}$ are different types of edges).

We will say that a pattern P *occurs as a sub-pattern* of a pattern Q if Q can be transformed into a homomorphic image of P by a sequence of the following *substructure operations*:

- removal of (positive or negative) edges,
- removal of isolated points, and
- removal of empty parts.

A binary CSP instance is a pattern in which a part A_i corresponds to the set of assignments to a variable; hence elimination of a part corresponds to the elimination of a variable.

Consider the three patterns Z1, Z2 and XI shown in Figure 1. Points are represented by bullets, and points representing assignments to the same variable v are grouped together within an oval representing A_v. Solid lines represent positive edges (i.e., compatibility of the corresponding pair of points) and dashed lines negative (i.e., incompatibility) edges. For example, the pattern Z1 consists of 4 points $a, b \in A_{v_0}$, $c, d \in A_{v_1}$, two positive edges $E^+ = \{ac, bd\}$ and one negative edge $E^- = \{bd\}$. Z1 occurs in Z2 since Z2 can be transformed into a homomorphic image of Z1 by removal of edge gh and point g (under a homomorphism that maps a and b to the same point e in Z2). The pattern Z2 occurs in XI (as a subpattern) since XI can be transformed into a homomorphic image of Z2 by removal of the edges $a'd'$ and $d'e'$ followed by the removal of the (then) isolated point e' together with the (then) empty part w. By transitivity of the occurrence relation, Z1 also occurs in XI.

We also consider patterns with structure in the sense of relations between the points in a pattern. In this case the homomorphism f in the definition of homomorphic image, above, must preserve the structure of the pattern. This structure may be, for example, an order on the parts (i.e., variables) or an order on points within a part (i.e., domain values). Patterns (such as Z1, Z2 and XI in Figure 1) without any such structure are known as *flat* patterns [6]. The pattern MC in Figure 1 has the structure consisting of the partial order: $\alpha > \beta$, $\gamma > \delta$. Since XI is a flat pattern, MC does not occur in XI since this partial order clearly cannot be preserved by a homomorphism from MC to XI. On the other hand, Z1 occurs in MC (via a homomorphism which maps both a and b to α, c to δ and d to γ) since Z1 has no structure to be preserved.

A class of binary CSP instances can be defined by *forbidding* a pattern P. We use the notation $\mathrm{CSP}_{\overline{\mathrm{SP}}}(P)$ to represent the set of binary CSP instances in which the pattern P does

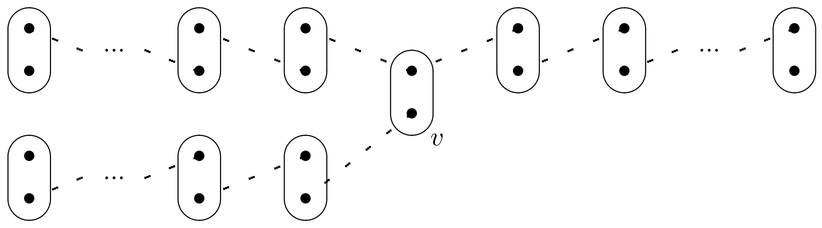


Figure 2 The negative pattern Pivot(k), where the number of edges in each of the three branches leaving the central variable v is k.

not occur as a subpattern. We say that a pattern P is *tractable* if there is a polynomial-time algorithm to solve $\mathrm{CSP}_{\overline{\mathrm{SP}}}(P)$ and *intractable* if $\mathrm{CSP}_{\overline{\mathrm{SP}}}(P)$ is NP-hard. The pattern MC is tractable since $\mathrm{CSP}_{\overline{\mathrm{SP}}}(\mathrm{MC})$ is the class of binary max-closed instances [36]. On the other hand, we know that Z2 is intractable since it does not satisfy a necessary condition for tractability described in Section 3.2. If P occurs as a subpattern of Q, then $\mathrm{CSP}_{\overline{\mathrm{SP}}}(P) \subseteq \mathrm{CSP}_{\overline{\mathrm{SP}}}(Q)$ and hence P is tractable if Q is tractable [6]. Thus we can immediately deduce that Z1 is tractable (since it occurs in MC) and that XI is intractable (since Z2 occurs in XI).

An important point is that applying any reduction operation which eliminates domain elements, such as arc consistency, SAC (Singleton Arc Consistency) or neighbourhood substitution [24], cannot introduce a pattern in an instance. On the other hand, reduction operations, such as 3-consistency, which modify constraints may introduce patterns.

A pattern P (such as pattern Z1 in Figure 1) is *mergeable* if there exists some pattern Q (such as the pattern Z1 without the edge ac or the point a) such that Q is a homomorphic image of P but P is not a homomorphic image of Q; otherwise, P is *unmergeable.* A point p (such as e' in pattern XI in Figure 1) is called *dangling* if p belongs to at most one positive edge $\{p, q\}$ and no negative edges (and p belongs to no other relation, such as a partial order, in the case of patterns with structure). The corresponding *dangling reduction* consists in removing both the edge $\{p, q\}$ and the point p (together with the part to which p belonged if this part becomes empty after removal of p). Dangling points provide no information in arc-consistent instances, in the sense that P occurs in an instance I if and only if the pattern P' occurs in I where P' is the result of applying a dangling reduction to P. Thus, using the fact that establishing arc consistency cannot introduce patterns, we have that P is tractable if and only if P' is tractable.

An unmergeable pattern with no dangling points is called *irreducible.* In Figure 1, pattern Z1 is mergeable, whereas patterns Z2 and XI are unmergeable. The point e' in pattern XI is dangling (as is the point a in the pattern Z1) but not β and δ in pattern MC (because of the partial order relation on these points). Thus of the four patterns in Figure 1, only Z2 and MC are irreducible.

3.2 Characterising Tractable Patterns

The theoretical tools necessary to provide a complete characterisation of tractable patterns have yet to be discovered. Indeed, characterising tractable patterns would appear to be, in general, even more difficult than characterising tractable constraint languages or tractable constraint-hypergraph structures. Nevertheless, certain characterisation results have been proved. One important result concerns the negative edges in a tractable unmergeable pattern. It has been shown that the "skeleton" of a tractable unmergeable pattern, consisting of just

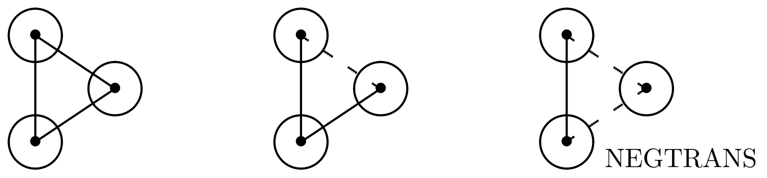


Figure 3 Tractable triangle patterns.

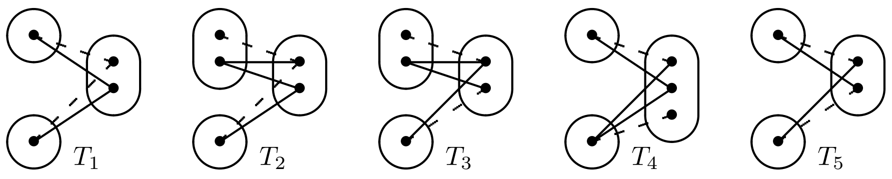


Figure 4 The five tractable irreducible flat patterns on 3 variables and 2 constraints.

the negative edges, must occur as a subpattern of (possibly multiple copies of) the pattern Pivot(k), shown in Figure 2, for some constant k [6]. Unfortunately, very little is known about the positive edges that can be added to such skeletons of negative edges.

Given that a general characterisation seems, for the moment, out of reach, certain special cases have been studied, such as triangle patterns and 2-constraint patterns [19, 15]. Figure 3 shows the three tractable flat patterns on a triangle of 3 points and 3 edges [19]. Although the first two patterns define fairly trivial classes, the third one, called NEGTRANS, is interesting since $\mathrm{CSP}_{\overline{\mathrm{SP}}}$(NEGTRANS) includes non-trivial instances composed of arbitrary unary constraints and non-overlapping All-Different constraints [48, 55]. The class of binary CSP instances satisfying this negative-transitivity property has been generalised to a large tractable class of optimisation problems involving cost functions of arbitrary arity which we will discuss in Section 7 [19]. Figure 4 shows the five tractable irreducible flat patterns on 3 variables and 2 constraints [15]. $\mathrm{CSP}_{\overline{\mathrm{SP}}}(T_4)$ is interesting since it includes all binary CSP instances with zero-one-all constraint relations [13] (which can be seen as a generalisation of 2SAT to non-Boolean domains).

Given the relatively modest successes in defining new and useful tractable classes by forbidding flat patterns, it is natural to consider possible extensions of patterns by studying structured patterns, non-binary patterns and other forms of occurrence than subpattern occurrence. These three extensions are the subject of the remainder of this section.

3.3 Partially-Ordered Patterns

The pattern BTP shown in Figure 5 is known as a broken triangle (since the positive edges can be said to form a triangle which is broken at variable z). Forbidding this pattern on all triples of variables $x < y < z$ defines a tractable class $\mathrm{CSP}_{\overline{\mathrm{SP}}}$(BTP) [16]. This class includes all binary CSP instances whose constraint graph is a tree $\mathcal{T}$ since, ordering the variables according to a pre-order of $\mathcal{T}$, each variable z is constrained by at most one variable $y < z$, its parent in $\mathcal{T}$, and hence the broken-triangle pattern cannot occur. If a variable ordering exists such that the broken-triangle pattern does not occur, then this order can be found in polynomial time: it suffices to establish arc consistency and then successively eliminate variables v which are not the right-hand variable z of a broken triangle (since we know that such a variable v can be the last variable in the ordering among the remaining variables).

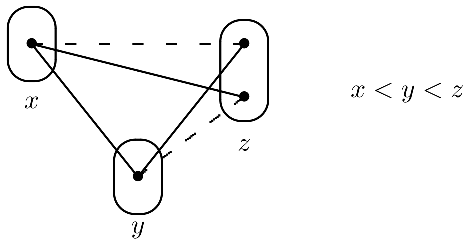


Figure 5 A binary CSP instance satisfies the broken triangle property if this pattern (known as a broken triangle or BTP) does not occur in the instance.

$\mathrm{CSP}_{\overline{\mathrm{SP}}}(\mathrm{BTP})$ is solved by arc consistency since the BTP is exactly the obstruction which prevents an arc-consistent instance being backtrack-free. Indeed, even if the variable order is unknown, MAC (Maintaining Arc Consistency) solves $\mathrm{CSP}_{\overline{\mathrm{SP}}}(\mathrm{BTP})$ [16]. Thus, most CSP solvers will automatically solve in polynomial time all instances in $\mathrm{CSP}_{\overline{\mathrm{SP}}}(\mathrm{BTP})$.

▶ **Example 1.** Consider a company which wishes to give bonuses to its n employees. Each employee $i \in \{1,\ldots,n\}$ has a grade $grade_i$, with higher grades corresponding to more important posts. Some, but not all, employees have an immediate boss to whom they report. The company wants to assign bonuses so that each employee's bonus is a multiple of 50 euros between 5% and 20% of their salary. If an employee i has an immediate boss b_i, then the sum of the bonuses of i and b_i must be no less than 10% and no more than 30% of the salary of b_i. On the other hand, if an employee i has no immediate boss then the rule is that they must not receive a bigger bonus than anyone at a higher grade.

Let the variable x_i be the bonus assigned to employee i. We assume that employees are numbered so that $(grade_i > grade_j) \Rightarrow (i < j)$. Thus, for example, employee number 1 is the CEO of the company. The domain of x_i is the multiples of 50 between 5% and 20% of the salary sal_i of employee i. If employee i has a boss b_i, then there is a binary constraint $0.1sal_{b_i} \leq x_i + x_{b_i} \leq 0.3sal_{b_i}$. Indeed, this is the only constraint between x_i and the variables x_j $(j < i)$. If employee i has no boss, then there are binary constraints $x_i \leq x_j$ for each $j \in \{1,\ldots,i-1\}$ such that $grade_j > grade_i$. In either case, it is easy to verify that x_i cannot be the rightmost variable in the broken triangle pattern shown in Figure 5. Thus, this problem falls in $\mathrm{CSP}_{\overline{\mathrm{SP}}}(\mathrm{BTP})$ and is solved by arc consistency. The constraint graph is of unbounded tree-width: for example, if no-one has a boss but everyone is at a different grade, then the constraint graph is the complete graph. Furthermore, the language of constraints is NP-hard. Thus, this bonus-assignment problem defines a truly hybrid tractable class.

We have seen that broken-triangle free instances are solved by arc consistency. Arc consistency is also a decision procedure for $\mathrm{CSP}_{\overline{\mathrm{SP}}}(\mathrm{EMC})$ where EMC (Extended Max-Closed) is the pattern shown in the top left of Figure 6. EMC is particularly interesting because $\mathrm{CSP}_{\overline{\mathrm{SP}}}(\mathrm{EMC})$ is a strict generalisation of binary max-closed CSPs [36] (since the pattern MC shown in Figure 1 is a subpattern of EMC).

▶ **Example 2.** Consider a binary CSP instance I with integer domains and in which all binary constraints are of the following form:

$$aX_i + bX_j \;\geq\; c$$

where a, b, c are non-zero constants. We say that X_i occurs *positively* (respectively, negatively) if $a > 0$ $(a < 0)$. These constraints are max-closed if and only if at least one of the variables

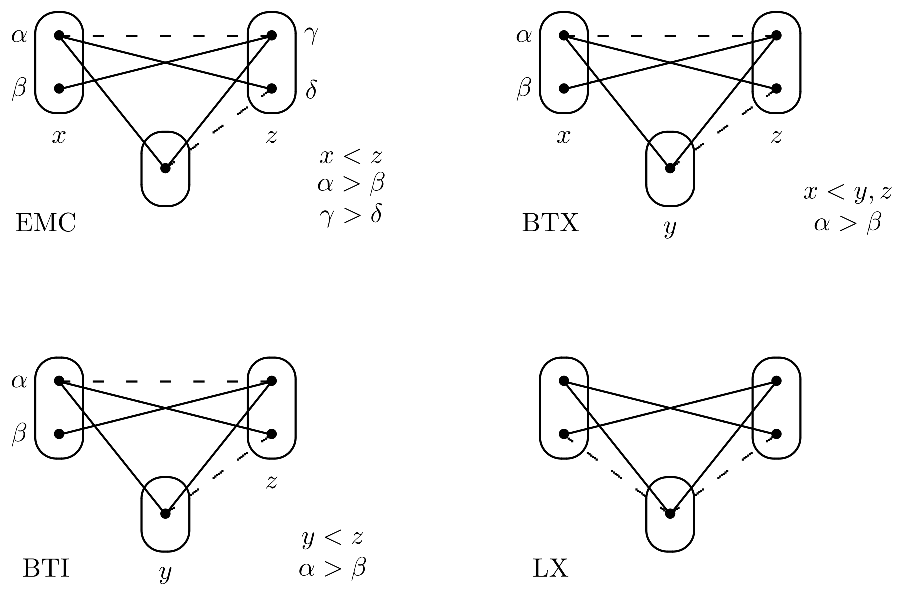


Figure 6 Partially-ordered patterns that are solved by arc consistency.

X_i, X_j occurs positively [36]. Suppose that in I, for all constraints on a pair of variables $X_i < X_j$ in which both variables occur negatively, the variable X_j only occurs negatively in other constraints. Then $I \in \mathrm{CSP}_{\overline{\mathrm{SP}}}(\mathrm{EMC})$, and hence is solved by arc consistency.

In fact, if we consider only unmergeable patterns to which we then add a partial order to the variables and/or the domains, then there are just five patterns P such that arc consistency is a decision procedure for $\mathrm{CSP}_{\overline{\mathrm{SP}}}(P)$: BTP and the patterns EMC, BTX, BTI and LX shown in Figure 6 [20]. Given a fixed total order of the domain, there is a polynomial-time algorithm to find a total variable ordering such that any one of these patterns does not occur in an instance (or to determine that no such ordering exists). However, if the domain and variable orders are both unknown, then EMC, BTX and BTI become NP-complete to detect [20].

3.4 Non-Binary CSPs

Most work on forbidden-pattern tractability has been restricted to binary CSPs. Indeed, notions such as microstructure and forbidden pattern do not (yet) have a widely-accepted generalisation to non-binary constraints. Nonetheless, the notion of arbitrary-arity patterns can be said to be already present in language classes defined by a polymorphism [11]. A polymorphism can be viewed as a forbidden pattern in each constraint relation R of the instance. The forbidden pattern corresponding to a polymorphism $f : D^r \to D$ consists of a set of r positive tuples and one negative tuple. The positive tuples $t_1, \dots, t_r \in R$ are consistent assignments to the same variables and the negative tuple $f(t_1, \dots, t_r) \notin R$ is the assignment to the same variables resulting from the pointwise application of f to the $t_1, \dots, t_r$. By *forbidding* the pattern $(t_1, \dots, t_r \in R,\ f(t_1, \dots, t_r) \notin R)$, we impose the well-known polymorphism condition $t_1, \dots, t_r \in R \implies f(t_1, \dots, t_r) \in R$ [11].

As we have seen in Section 3.3, in the binary CSP, the broken-triangle property defines a tractable class $\mathrm{CSP}_{\overline{\mathrm{SP}}}(\mathrm{BTP})$. Although the definition of tractable classes by forbidden

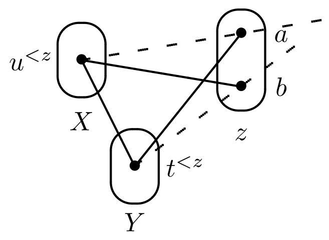


Figure 7 Illustration of a directional general-arity broken triangle.

patterns in general-arity CSPs is an area which remains largely unexplored, a generalisation of the broken-triangle class $\text{CSP}_{\overline{\text{SP}}}(\text{BTP})$ to general-arity CSPs has recently been given [14].

Purely for notational convenience we assume that a CSP instance I is given in the form of a set of negative (incompatible) tuples NoGoods(I), where a tuple is a set of variable-value assignments, and that the predicate Good(I, t) is true iff the tuple t does not contain any pair of distinct assignments to the same variable and $\nexists t' \subseteq t$ such that $t' \in \text{NoGoods}(I)$. We write Good as a predicate whereas Nogoods(I) is a set to emphasize the asymmetry between the notions of positive and negative tuples. This asymmetry is not evident in the case of binary CSPs nor in the case of polymorphisms.

We suppose that a total ordering $<$ of the variables of a CSP instance I is given. We write $t^{<x}$ to represent the subset of the tuple t consisting of assignments to variables occurring before x in the order $<$, and $Vars(t)$ to denote the set of all variables assigned by t.

▶ **Definition 3.** A *directional general-arity broken triangle* (DGABTP) on assignments a, b to variable z in a CSP instance I is a pair of tuples t, u (containing no assignments to variable z) satisfying the following conditions:

1. $t^{<z}$ and $u^{<z}$ are non-empty,
2. $\text{Good}(I, t^{<z} \cup u^{<z}) \quad \wedge \quad \text{Good}(I, t^{<z} \cup \{\langle z, a\rangle\}) \quad \wedge \quad \text{Good}(I, u^{<z} \cup \{\langle z, b\rangle\})$,
3. $t \cup \{\langle z, b\rangle\} \in \text{NoGoods}(I) \quad \wedge \quad u \cup \{\langle z, a\rangle\} \in \text{NoGoods}(I)$,
4. $\exists t'$ s.t. $Vars(t') = Vars(t) \quad \wedge \quad (t')^{<z} = t^{<z} \quad \wedge \quad t' \cup \{\langle z, a\rangle\} \notin \text{NoGoods}(I)$,
5. $\exists u'$ s.t. $Vars(u') = Vars(u) \quad \wedge \quad (u')^{<z} = u^{<z} \quad \wedge \quad u' \cup \{\langle z, b\rangle\} \notin \text{NoGoods}(I)$.

I satisfies the *directional general-arity broken-triangle property (DGABTP)* according to the variable ordering $<$ if no directional general-arity broken triangle occurs on any pair of values a, b for any variable z.

Points (1), (2) and (3) of Definition 3 are illustrated by Figure 7. This figure is similar to Figure 5 except that X,Y are sets of variables and $t^{<z}$,$u^{<z}$ are tuples. Note that the sets $X = Vars(u^{<z})$ and $Y = Vars(t^{<z})$ may overlap. Solid lines now represent partial solutions (i.e., consistent assignments to subsets of variables). The two dashed lines represent nogoods (i.e., tuples not in the constraint relation on its variables) $u \cup \{\langle z, a\rangle\}$ and $t \cup \{\langle z, b\rangle\}$ which possibly involve assignments to variables $w > z$. In the case of binary CSPs, a directional general-arity broken triangle is equivalent to a broken triangle as shown in Figure 5 (since nogoods, being binary, involve no other variables $w > z$ and the sets X, Y are necessarily singletons). Points (4) and (5) of Definition 3 are technical conditions (which always hold if a weak form of directional consistency holds) ensuring that the DGABTP can be tested in polynomial time for a given order whether constraints are given as tables of satisfying assignments or as nogoods.

Any instance I satisfying the DGABTP can be solved in polynomial time by repeatedly applying the following two operations: (i) merge two values in the last remaining variable (according to the order $<$); (ii) eliminate this variable when its domain becomes a singleton. *Merging* values $a, b \in \mathcal{D}(z)$ in a general-arity CSP instance I consists of replacing a, b in $\mathcal{D}(z)$ by a new value c which is compatible with all variable-value assignments compatible with at least one of the assignments $\langle z, a\rangle$ or $\langle z, b\rangle$, thus producing an instance I' with the new set of nogoods defined as follows:

$$\begin{aligned}
\text{NoGoods}(I') \;=\;& \{t \in \text{NoGoods}(I) \mid \langle z,a\rangle, \langle z,b\rangle \notin t\} \\
\cup\;& \{t \cup \{\langle z,c\rangle\} \mid t \cup \{\langle z,a\rangle\} \in \text{NoGoods}(I) \;\wedge \\
& \qquad \exists t' \in \text{NoGoods}(I) \text{ s.t. } t' \subseteq t \cup \{\langle z,b\rangle\}\} \\
\cup\;& \{t \cup \{\langle z,c\rangle\} \mid t \cup \{\langle z,b\rangle\} \in \text{NoGoods}(I) \;\wedge \\
& \qquad \exists t' \in \text{NoGoods}(I) \text{ s.t. } t' \subseteq t \cup \{\langle z,a\rangle\}\}\,.
\end{aligned}$$

In general, merging a pair of values in an instance I may produce an instance I' which is satisfiable even though I was not, but forbidding directional general-arity broken triangles prevents this from happening. Eliminating a variable z whose domain is a singleton $\{a\}$ consists in making the assignment $\langle z, a\rangle$ and eliminating $\langle z, a\rangle$ from all nogoods.

Unfortunately, when the variable order is not given, testing the existence of a variable ordering for which a CSP instance satisfies the DGABTP is NP-complete in the general-arity case [14]. This can be contrasted with the binary case in which this test is polytime.

Note that the set of general-arity CSP instances whose dual instance satisfies the BTP, denoted by DBTP, also defines a tractable class which can be recognised in polynomial time even if the ordering of the variables in the dual instance is unknown [44]. This DBTP class is incomparable with DGABTP (which is equivalent to BTP in binary CSP) since DBTP is known to be incomparable with the BTP class already in the special case of binary CSP [44]. A general-arity broken triangle can be said to be centred on a pair of *values* in the domain of a variable whereas a broken triangle in the dual instance is centred on a pair of *tuples* in a constraint relation. One consequence of this is that eliminating tuples from constraint relations cannot introduce broken triangles in the dual instance, whereas the DGABTP is only invariant under elimination of domain values. On the other hand, the DGABTP is invariant under adding a complete constraint (i.e., whose relation is the direct product of the domains of the variables in its scope) whereas this operation can introduce broken triangles in the dual instance. Another important difference is that DGABTP depends on an order on the variables whereas DBTP depends on an order on the constraints.

The generalisation of forbidden patterns to non-binary constraints is a largely unexplored area of research, but the generalisation of BTP to DGABTP has highlighted the asymmetry between positive and negative tuples when constraints are non-binary.

3.5 Quantified Patterns

The notion of forbidding patterns has also been extended to rules based on applying a sequence of quantifiers to the variables and values in a pattern. This has led to the discovery of novel variable-elimination or value-elimination techniques [7, 12]. As an example, consider the broken triangle pattern shown in Figure 5. It is known that we can eliminate variables which are not the right-hand variable of a broken triangle: the resulting instance is satisfiable if and only if the original instance was satisfiable [16]. This variable-elimination rule can be strictly generalised to the following rule illustrated in Figure 8: a variable z can be eliminated from an instance I without changing the satisfiability of I if for all other variables y, $\forall a \in \mathcal{D}(y)$, $\exists b \in \mathcal{D}(z)$ with ab a positive edge in I such that no broken triangle exists including this edge ab [12]. The class of binary CSP instances, known as $\forall\exists$BTP, which are

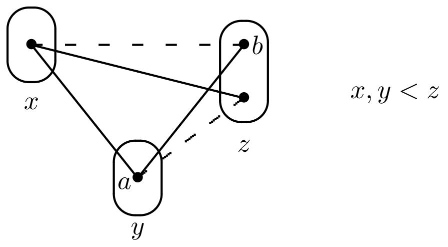


Figure 8 In the ∀∃BTP class, for all pairs of variables $y < z$, $\forall a \in \mathcal{D}(y)$, $\exists b \in \mathcal{D}(z)$ such that for all variables $x < z$, the broken triangle pattern shown does not occur.

such that all variables can be eliminated according to this rule strictly generalises the tractable class $\mathrm{CSP}_{\overline{\mathrm{SP}}}(\mathrm{BTP})$, since the BTP imposes the same condition but for *all* $b \in \mathcal{D}(X_k)$.

Let I be a binary arc-consistent CSP instance in the ∀∃BTP class and let s be a solution to the instance obtained by eliminating the last variable z from I. We will give a sketch proof that s can be extended to a solution for I. (We refer the reader to [12] for full details.) By assumption, $\forall y \in X \setminus \{z\}$, $\forall a \in \mathcal{D}(y)$, $\exists b^y_a \in \mathcal{D}(z)$ with ab^y_a a positive edge such that $\forall x \in X \setminus \{y, z\}$, $\forall c \in \mathcal{D}(x)$ with ac a positive edge and $b^y_a c$ a negative edge, $\forall d \in \mathcal{D}(z)$ with cd a positive edge, ad is also a positive edge. For $v \in X \setminus \{z\}$, let $\mathrm{Im}(v) := \{d \in \mathcal{D}(z) \mid s(v)d$ a positive edge$\}$, where $s(v)$ is the value assigned to variable v by s. If $x, y \in X \setminus \{z\}$ are such that $s(x)b^y_{s(y)}$ is a negative edge, then the ∀∃BTP property implies that $\mathrm{Im}(x) \subsetneq \mathrm{Im}(y)$ [12]. Now choose some $y \in X \setminus \{z\}$ such that $\mathrm{Im}(y)$ is minimal for inclusion among the sets $\mathrm{Im}(v)$ ($v \in X \setminus \{z\}$). Then the assignment $\langle z, b^y_{s(y)}\rangle$ is compatible with all the assignments $s(x)$ ($x \in X \setminus \{y, z\}$), otherwise we would have $\mathrm{Im}(x) \subsetneq \mathrm{Im}(y)$ (contradicting the minimality of $\mathrm{Im}(y)$). Therefore, s can be extended to a solution to I, by making the assignment $s(z) = b_{s(y)}$.

3.6 Topological Minor Patterns

We now present a new operation on patterns which allows us to define the notion of a topological minor of a pattern (and hence of a binary CSP instance). This new operation is analogous to the operation of eliminating subdivisions (vertices of degree 2) that is used to define a topological minor of a graph [25]. However, since patterns contain two kinds of edges, the definition is slightly more complicated.

This new operation, path reduction, will sometimes lead to the introduction of edges in $E^+ \cap E^-$ in a coloured microstructure $\langle A_1, \dots, A_n, E^+, E^-\rangle$. This is why we do not impose the restriction that E^+ and E^- be disjoint in a pattern.

In a pattern $P = \langle A_1, \dots, A_n, E^+, E^-\rangle$, we say that two parts A_i, A_j are *directly connected* if there is at least one (positive or negative) edge $\{p, q\} \in E^+ \cup E^-$ with $p \in A_i$ and $q \in A_j$.

If A_i, A_j are not directly connected and A_k is directly connected only to A_i and A_j, then the following operation can be performed, which is known as *path reduction*:

1. $\forall p \in A_i$, $\forall q \in A_j$: if $\exists r \in A_k$ such that $\{p, r\}, \{r, q\} \in E^+$, then introduce a new positive edge $\{p, q\}$,
2. $\forall p \in A_i$, $\forall q \in A_j$: if $\exists r, s \in A_k$ such that $\{p, r\}, \{s, q\} \in E^-$, then introduce a new negative edge $\{p, q\}$,
3. remove the part A_k and all edges containing points in A_k.

This operation is illustrated in Figure 9. Positive and negative edges are treated differently in this definition; this is because for $p \in A_i$ and $q \in A_j$ to be part of a solution to the

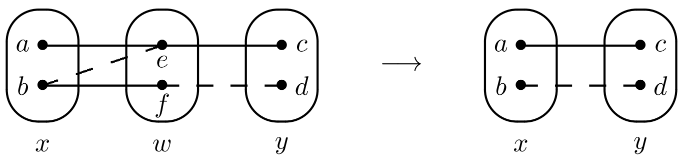


Figure 9 Path reduction removes the part w.

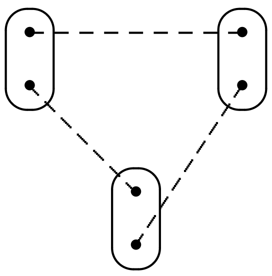


Figure 10 A pattern which defines the class of acyclic binary CSP instances when forbidden as a topological minor.

sub-instance on variables X_i, X_j, X_k, the points p and q must both be compatible with some common point $r \in A_k$, whereas p and q may be incompatible if they are each incompatible with some point in A_k, not necessarily the same point. In Figure 9, after the path reduction operation which eliminates w, we have a positive edge ac (thanks to the edges ae and ec), but no positive edge bc. We also have a negative edge bd (thanks to the edges be and fd). As in the case of non-binary patterns (c.f. Section 3.4), it is essential to introduce an asymmetry between positive and negative edges in order to obtain a useful notion.

A pattern P occurs as a *topological minor* of a pattern Q if Q can be transformed into a homomorphic image of P by a sequence of substructure operations (listed above in Section 3.1) and path reductions.

We use the notation $\mathrm{CSP}_{\overline{\mathrm{TM}}}(P)$ to represent the set of binary CSP instances in which the pattern P does not occur as a topological minor. For each pattern P there are therefore two distinct notions of tractability: a pattern P is *sub-pattern tractable* if there is a polynomial-time algorithm to solve $\mathrm{CSP}_{\overline{\mathrm{SP}}}(P)$; a pattern P is *topological-minor tractable* if there is a polynomial-time algorithm to solve $\mathrm{CSP}_{\overline{\mathrm{TM}}}(P)$. A pattern which is sub-pattern tractable is topological-minor tractable since any pattern that occurs as a sub-pattern will also occur as a topological minor.

One important tractable class of binary CSP instances is the class of instances whose constraint graph is acyclic [28]. However, this class cannot be defined by a finite set of forbidden sub-patterns [9]. On the other hand, it is straightforward to characterise the class of acyclic instances by forbidding a single pattern as a topological minor. Forbidding the pattern shown in Figure 10 as a topological minor exactly defines the class of binary CSP instances whose constraint graph is acyclic. Indeed, this idea can easily be extended to any of the tractable classes of binary CSP instances defined by imposing any fixed bound on the treewidth of the constraint graph [29] using the graph minor theorem [49]. However, it remains to be seen whether the notion of patterns occurring as topological minors can be used to define a practically useful and genuinely novel tractable class.

4 Classes Requiring a Level of Consistency

Some hybrid tractable classes have been defined which guarantee global consistency if some local property holds after establishing a certain level of local consistency. One example is that the constraints can be decomposed into the join of arity-r constraints after establishing strong $d(r-1)+1$ consistency, where d is the maximum domain size [23]. Of course, in general, establishing this level of consistency introduces constraints of order $d(r-1)$, so the assumption that constraints are of arity r is very strong. This class has been generalised to the class of arity-r CSP instances which are strongly $((m+1)(r-1)+1)$-consistent, where given an r-ary constraint and an instantiation of $r-1$ of the variables that participate in the constraint, the parameter m (called the tightness) is an upper bound on the number of instantiations of the rth variable that satisfy the constraint in the case that this is not the whole domain [54].

Naanaa [46] has proposed a generalisation of m-tightness. Let E be a finite set and let $\{E_i\}_{i\in I}$ be a finite family of subsets of E. The family $\{E_i\}_{i\in I}$ is said to be *independent* if and only if for all $J \subsetneq I$,

$$\bigcap_{i\in I} E_i \subsetneq \bigcap_{j\in J} E_j \,.$$

In particular, observe that $\{E_i\}_{i\in I}$ cannot be independent if $\exists j \neq j' \in I$ such that $E_j \subseteq E_{j'}$, since in this case and with $J = I \setminus \{j'\}$ we would have

$$\bigcap_{i\in I} E_i = \bigcap_{j\in J} E_j \,.$$

Let I be a CSP instance whose variables are totally ordered by $<$. Let $\langle\sigma, R\rangle$ be an r-ary constraint whose scope σ contains a variable x and let t be a tuple that instantiates the $r-1$ remaining variables of σ. Denote by $R_x(t)$ the set of values in $\mathcal{D}(x)$ that can extend t to form a tuple in the relation R. The *directional extension* of tuple t to variable x w.r.t. R and $<$ is defined to be $R_x(t)$ if x is the last (w.r.t. the order $<$) variable in σ, and $\mathcal{D}(x)$ otherwise. A family of extensions of tuples $t \in T$ is said to be consistent if and only if the tuple formed by the join $\bowtie_{t\in T} t$ of the corresponding tuples is consistent. With respect to the ordering $<$, the *directional rank* of x in I is the size of the largest independent and consistent family of directional extensions to x, and the *directional rank* κ of I is the maximum directional rank over all its variables. If I is a CSP instance with constraints of arity no greater than r which has directional rank no greater than κ and is directional strong $(\kappa(r-1)+1)$-consistent, then I is globally consistent [46]. In general, establishing this level of consistency introduces constraints of arity $\kappa(r-1)$ which is no greater than r only if $\kappa = 1$ or $(\kappa = 2 \ \wedge \ r = 2)$.

There is an interesting link between directional rank and forbidden patterns. Directional rank 1 is equivalent to the broken triangle property (i.e., forbidding as a subpattern the pattern shown in Figure 5) and directional rank $\kappa > 1$ subsumes an extension of the broken triangle property known as $(\kappa+1)$-BTP [17]. A binary CSP instance satisfies k-BTP if for all variables z and for all sets S of k variables occurring before z in the variable ordering, $\exists x, y \in S$ such that there are no broken triangles on variables x, y, z. We can see that 2-BTP corresponds exactly to the broken triangle property. If a binary CSP instance satisfies 3-BTP after establishment of directional strong 3-consistency, then it has directional rank $\kappa = 2$ and is directional strong $(\kappa(r-1)+1)$-consistent (since $r = 2$), and hence is globally consistent [46]. Directional rank 2 strictly subsumes 3-BTP since it is equivalent to forbidding (as a subpattern) the pattern shown in Figure 11. This is a natural generalisation

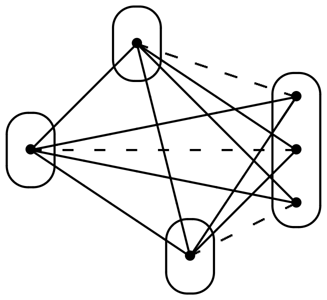


Figure 11 A binary CSP instance has directional rank 2 if this pattern does not occur in the instance.

of the broken-triangle pattern shown in Figure 5, but, unfortunately, we also require strong directional 3-consistency to obtain a tractable class and establishing this level of consistency may introduce the pattern.

5 Microstructure-Based Classes

We recall the definition of microstructure. We have seen that a binary CSP instance on variables $X_1, \ldots, X_n$ can be represented by the domain $\mathcal{D}(X_i)$ of each variable X_i and a binary relation R_{ij} for each pair of variables X_i, X_j $(i \neq j)$ consisting of all possible consistent assignments to this pair of variables. If I is a binary CSP instance, then its *microstructure* is a graph $\langle A, E\rangle$ where $A = \{(X_i, a) \mid a \in \mathcal{D}(X_i)\}$ is the set of possible variable-value assignments and $E = \{\{(X_i, a), (X_j, b)\} \mid (a, b) \in R_{ij}\}$ [37]. The microstructure relies on both the structure and the relations of the instance I and so is a natural place to look for hybrid tractable classes. In this section we study properties of the graph $\langle A, E\rangle$ in which A is not partitioned into parts corresponding to variables (as was the case in patterns, studied in Section 3). Ignoring variable information has the obvious disadvantage that we lose possibly valuable information, but has the advantage that deep theorems from graph theory can be directly applied.

The *complement* of a graph $G = \langle V, E\rangle$ is the graph with vertices V and whose edges are the non-edges of G. The *microstructure complement* is the complement of the microstructure. Solutions to I are in one-to-one correspondence with the n-cliques of the microstructure of I and with the size-n independent sets of the microstructure complement of I.

The *chromatic number* of a graph is the smallest number of colours required to colour its vertices so that no two adjacent vertices have the same colour. A graph G is *perfect* if for every induced subgraph H of G, the chromatic number of H is equal to the size of the largest clique contained in H. Since a maximum clique in a perfect graph can be found in polynomial time [33], the class of binary CSP instances with a perfect microstructure is tractable as a direct consequence, as observed in [50]. Perfect graphs can also be recognized in polynomial time [3].

For a class of graphs $\mathcal{C}$, a graph G is $\mathcal{C}$-free if no induced subgraph of G is isomorphic to any graph in $\mathcal{C}$. The *cycle of order* k is the graph with vertices $v_1, \ldots, v_k$ and edges $\{v_k, v_1\}$ and $\{v_i, v_{i+1}\}$ for $i = 1, \ldots, k-1$. A *hole* is a cycle of length $k \geq 5$. An *antihole* is the complement of a hole. An alternative definition of perfect graphs is that a graph is perfect if and only if it is (odd-hole,odd-antihole)-free [4]. Chordal graphs are examples of perfect graphs. Interesting examples of binary CSP instances whose microstructure is perfect are

- instances with unary constraints together with a global AllDifferent constraint [50],
- instances which are arc consistent and max-closed after independent (and possibly unknown) permutations of each domain [31].

6 Weakly or Strongly Constrained Instances

One way to define a tractable class is to only allow a small number of weak constraints, in order to guarantee that the instance is always satisfiable. For example, if each variable is in the scope of at most t constraints and in each constraint relation the proportion of tuples that are disallowed is strictly less than $1/e(r(t-1)+1)$, where e is the base of natural logarithms and r the arity of the constraint, then the instance is necessarily satisfiable [47].

Another way to define a tractable class is to consider only instances which are sufficiently strongly constrained so that there is necessarily only a small number of partial solutions examined during search (and hence a small number of solutions). A simple example of a condition that guarantees a polynomial number of solutions is functionality. A constraint $\langle\sigma, R\rangle$ is *functional* on variable $X_i \in \sigma$ if the relation R contains no two assignments differing only at variable X_i. A CSP instance is *functional with root set of size* k if there exists a variable ordering $X_1 < \ldots < X_n$ such that, for all $i \in \{k+1, \ldots, n\}$, there is some constraint $\langle\sigma, R\rangle$ with $X_i \in \sigma \subseteq \{X_1, \ldots, X_i\}$ that is functional on X_i. (Note that this implies tractability.) In the case of binary CSP instances, a minimum root set can be found in polynomial time [22]. Unfortunately, determining the size of a minimum root set is NP-hard for ternary CSP instances [8].

Another condition that guarantees a backtracking search tree of polynomial size (assuming domain size bounded by a constant) is the k-Turan property [8] which we now define. Indeed, this property is very strong since it guarantees a polynomial-size search tree for all variable orderings. We say that a subset of variables S *represents* another set T if $S \subseteq T$. An (n,k)-*Turan system* is a pair $\langle X, B\rangle$ where B is a collection of subsets of the n-element set X such that every k-element subset of X is represented by some set in B. For example, let $\mathcal{C}_{4\text{Turan}}$ be the class of binary CSP instances over a set of variables X, each with a Boolean domain, in which each constraint is equivalent to a 3SAT clause and for each quadruple of distinct variables X_i, X_j, X_k, X_ℓ there is at least one ternary constraint whose scope is a subset of these variables. In this example, every 4-element subset of the set of n variables X is represented by the scope of some ternary constraint, and hence $\langle X, S\rangle$, where S is the set of constraint scopes, is an $(n,4)$-Turan system. An n-variable CSP instance over domain D and variables X is k-*Turan* if $\langle X, B\rangle$ forms an (n,k)-Turan system where B is the set of the scopes of the constraints $\langle\sigma, R\rangle$ for which

$$\forall a, b \in D,\ \{a,b\}^{|\sigma|} \nsubseteq R\,.$$

This condition says that at least one tuple is disallowed by the constraint over each Boolean subdomain $\{a,b\}$ of D. In the class $\mathcal{C}_{4\text{Turan}}$, all constraints satisfy this condition and hence $\mathcal{C}_{4\text{Turan}}$ is tractable since all instances in this class satisfy the 4-Turan property. Generalising this example, the class of k-SAT instances where every k'-tuple of variables, where $k' \geq k$, is restricted by a clause is k'-Turan and hence tractable.

It is an open question whether the k-Turan property can be relaxed in a way that guarantees a polynomial-size search tree for just one variable ordering, rather than all variable orderings, while imposing a weaker condition than functionality.

7 Valued CSPs

CSPs are inherently decision problems. In this section we discuss hybrid classes of valued CSPs, which is a generalisation of CSPs to problem that capture both decision and optimisation problems (and their combinations).

We denote by $\overline{\mathbb{Q}} = \mathbb{Q} \cup \{\infty\}$ the set of extended rationals. A *valued constraint satisfaction problem* (VCSP) instance I is given by a triple $\langle X, D, C\rangle$, where $X = \{X_1, \ldots, X_n\}$ is a finite set of variables, D is a finite set of values, and $C : D^n \to \overline{\mathbb{Q}}$ is an objective function expressed as a sum of valued constraints, i.e., $C(X_1, \ldots, X_n) = \sum_{i=1}^{q} \gamma_i(\mathbf{v}_i)$, where $\gamma_i : D^{k_i} \to \overline{\mathbb{Q}}$ is a *cost function* of arity k_i and $\mathbf{v}_i \in X^{k_i}$ is the scope of the valued constraint $\gamma_i(\mathbf{v}_i)$. The question is to find an assignment of values to the variables that minimises the objective function C.

The class of VCSP instances with $\{0, \infty\}$-valued cost functions corresponds to the class of CSP instances. VCSP instances with $\{0, 1\}$-valued cost functions are known as Min-CSPs. VCSP instances with $\mathbb{Q}$-valued cost functions are known as finite-valued CSPs [53].

Similarly to the case of CSPs, language-restricted VCSPs parameterised by the set of allowed cost functions in the instance have been studied [42]. In this section we will mention known results on the complexity of hybrid classes of VCSPs.

The idea of lifted languages briefly discussed in Section 2.3 has also been applied to certain VCSPs [39, 52].

7.1 JWP and Generalisations

The study of hybrid classes of VCSPs was initiated in [18], where an interesting hybrid class called the *joint winner property* (JWP) was discovered. A class of *binary* VCSPs satisfies the JWP if for any three variable-value assignments (to three distinct variables), the multiset of pairwise costs imposed by the binary valued constraints does not have a unique minimum. If there is no valued constraint with the scope, say, $\mathbf{v} = \langle X_i, X_j\rangle$, then we view it as a 0-valued constraint $\gamma(\mathbf{v})$, where $\gamma : D^2 \to \overline{\mathbb{Q}}$ is the constant-0 binary cost function. Note that the unary valued constraints in a VCSP that satisfies the JWP can be arbitrary. JWP generalises the tractable pattern NEGTRANS discussed in Section 3.2, as the NEGTRANS pattern precisely forbids the combination of one 0 cost and two ∞ costs.

Following the discovery of JWP, Cooper and Živný classified classes of binary CSPs, Min-CSPs, finite-valued CSPs, and VCSPs parametrised by the allowed types of costs in triples of variable-value assignments (called triangles) [19]. In all studied cases, JWP was essentially the only interesting tractable case. Moreover, [19] generalised JWP to the tractable class of VCSPs with the *cross-free convex* (CFC) property.

A function $g : \{0, \ldots, s\} \to \overline{\mathbb{Q}}$ is called *convex on the interval* $[l, u]$ if g is $\mathbb{Q}$-valued on the interval $[l, u]$ and the derivative of g is nondecreasing on $[l, u]$, that is, $g(m+2) - g(m+1) \geq g(m+1) - g(m)$ for all $m = l, \ldots, u-2$.

Sets $A_1, \ldots, A_r \subseteq A$ are called *cross-free* if for all $1 \leq i, j \leq r$, either $A_i \subseteq A_j$, or $A_i \supseteq A_j$, or $A_i \cap A_j = \emptyset$, or $A_i \cup A_j = A$ [51].

We interpret a solution $s : X \to D$ to a VCSP instance $I = \langle X, D, C\rangle$ as its set of variable-value assignments $\{\langle X_i, s(X_i)\rangle \mid i = 1, \ldots, n\}$. If A_i is a set of variable-value assignments of a VCSP instance I and s a solution to I, then we use the notation $|s \cap A_i|$ to represent the number of variable-value assignments in the solution s that lie in A_i.

Finally we have everything we need to define the CFC property. Let I be a VCSP instance. Let $A_1, \ldots, A_r$ be cross-free sets of variable-value assignments of I. Let s_i be the number of distinct variables occurring in the set A_i. Instance I satisfies the *cross-free convexity* *property* if the objective function of I is $g(s) = g_1(|s \cap A_1|) + \ldots + g_r(|s \cap A_r|)$, where each $g_i : [0, s_i] \to \overline{\mathbb{Q}}$ $(i = 1, \ldots, r)$ is convex on an interval $[l_i, u_i] \subseteq [0, s_i]$ and $g_i(z) = \infty$ for $z \in [0, l_i - 1] \cup [u_i + 1, s_i]$.

We remark that the functions g_i above are not the cost functions associated with the valued constraints. Note that similarly to JWP, the addition of any unary cost function cannot destroy the cross-free convexity property because for each variable-value assignment $\langle X_i, a\rangle$ we can add the singleton $A_i = \{\langle X_i, a\rangle\}$, which is necessarily either disjoint from or a subset of any other set A_j (and furthermore the corresponding function $g_i : \{0, 1\} \to \overline{\mathbb{Q}}$ is trivially convex).

A special case of the CFC property are global cardinality constraints [48, 56] on cross-free sets.

▶ **Example 4.** To give a concrete example, consider a company which needs to assign staff to a project, minimising total salary cost while respecting constraints concerning the minimum number of personnel from each section, the maximum total number of staff on the project, as well as the availability of each member of staff. We can code this as a VCSP with a boolean variable X_i for each member of staff with X_i assigned the value true if the person in question is assigned to the project. The availability of each member of staff, as well as his/her salary, can be coded as a unary cost function. Assuming each member of staff belongs to a single section of the company, the remaining constraints are global cardinality constraints on cross-free sets.

The employees, numbered from 1 to n, are partitioned into sections S_j $(j = 1, \ldots, t)$. Let $A_j = \{\langle X_i, true\rangle \mid i \in S_j\}$ $(j = 1, \ldots, t)$, $A_0 = \bigcup_{j=1}^{t} A_j$, and $A_{t+i} = \{\langle X_i, true\rangle\}$ $(i = 1, \ldots, n)$. The sets A_i $(i = 0, \ldots, t + n)$ are cross-free. Let $g_{\le m} : \{0, \ldots, n\} \to \overline{\mathbb{Q}}$ be the function given by $g_{\le m}(x) = 0$ if $x \le m$ and $g_{\le m}(x) = \infty$ if $x > m$, with $g_{\ge m}$ defined similarly. For $i = 1, \ldots, n$, let $h_i : \{0, 1\} \to \overline{\mathbb{Q}}$ be the function given by $h_i(0) = 0$ and $h_i(1)$ equal to the salary of employee i if he/she is available to work on the project, ∞ if not. Suppose that the project requires at least n_i employees from section i, for each $i = 1, \ldots, t$, making a total of at most N staff members. Then the objective function is

$$g_{\le N}(|s \cap A_0|) \; + \; \sum_{j=1}^{t} g_{\ge n_i}(|s \cap A_j|) \; + \; \sum_{i=1}^{n} h_i(|s \cap A_{t+i}|)\,.$$

The functions $g_{\le N}$ and $g_{\ge n_i}$ are convex (indeed constant) on the interval on which they are finite, and each h_i is trivially convex on the interval $[0, 1]$.

It has recently been shown[1] that the class of convex cross-free VCSPs is a special case of M$^\sharp$-convex functions studied in [45] and that JWP precisely captures binary M$^\sharp$-completable functions.

7.2 Planarity

Similarly to the planar CSPs discussed in Section 2.1, one can define planar VCSPs. Fulla and Živný gave necessary conditions on the tractability of planar VCSPs [30]. In particular, they showed that if Γ is a Boolean valued constraint language such that VCSP(Γ) is intractable then VCSP$_p$(Γ) is intractable unless Γ is self-complementary in the valued sense; i.e., for every $\gamma \in \Gamma$ and every tuple t, $\gamma(t) = \gamma(\bar{t})$. This is a generalisation of the self-complementarity

[1] Personal communication with Yuni Iwamasa and Kazuo Murota.

condition for planar CSPs from [26] discussed in Section 2.1. Furthermore, [30] obtained a dichotomy for planar VCSPs for conservative language (i.e., languages containing all $\{0,1\}$-valued unary cost functions) over arbitrary finite domains. As it turns out the planarity restriction does not give any new tractable languages in this setting, and the classification from [30] sharpens the classification of conservative valued constraint languages obtained in [41].

References

1 Libor Barto and Marcin Kozik. Constraint satisfaction problems solvable by local consistency methods. *J. ACM*, 61(1):3:1–3:19, 2014. `doi:10.1145/2556646`.

2 Andrei Bulatov, Andrei Krokhin, and Peter Jeavons. Classifying the Complexity of Constraints using Finite Algebras. *SIAM Journal on Computing*, 34(3):720–742, 2005. `doi:10.1137/S0097539700376676`.

3 Maria Chudnovsky, Gérard Cornuéjols, Xinming Liu, Paul Seymour, and Kristina Vuskovic. Recognizing Berge graphs. *Combinatorica*, 25(2):143–186, 2005.

4 Maria Chudnovsky, Neil Robertson, Paul Seymour, and Robin Thomas. The strong perfect graph theorem. *Annals of Math.*, 164(1):51–229, 2006.

5 David A. Cohen, Martin C. Cooper, Páidí Creed, Peter Jeavons, and Stanislav Živný. An algebraic theory of complexity for discrete optimisation. *SIAM Journal on Computing*, 42(5):915–1939, 2013. URL: `http://zivny.cz/publications/cccjz13sicomp-preprint.pdf`, `doi:10.1137/130906398`.

6 David A. Cohen, Martin C. Cooper, Páidí Creed, Dániel Marx, and András Z. Salamon. The tractability of CSP classes defined by forbidden patterns. *J. Artif. Intell. Res. (JAIR)*, 45:47–78, 2012. `doi:10.1613/jair.3651`.

7 David A. Cohen, Martin C. Cooper, Guillaume Escamoche, and Stanislav Živný. Variable and value elimination in binary constraint satisfaction via forbidden patterns. *Journal of Computer and System Sciences*, 81(7):1127–1143, 2015. URL: `http://zivny.cz/publications/ccez15jcss-preprint.pdf`, `doi:10.1016/j.jcss.2015.02.001`.

8 David A. Cohen, Martin C. Cooper, Martin J. Green, and Dániel Marx. On guaranteeing polynomially bounded search tree size. In Jimmy Ho-Man Lee, editor, *CP 2011*, volume 6876 of *Lecture Notes in Computer Science*, pages 160–171. Springer, 2011. `doi:10.1007/978-3-642-23786-7_14`.

9 David A. Cohen, Martin C. Cooper, Peter G. Jeavons, and Stanislav Živný. Tractable classes of binary CSPs defined by excluded topological minors. In Qiang Yang and Michael Wooldridge, editors, *IJCAI 2015*, pages 1945–1951. AAAI Press, 2015. URL: `http://ijcai.org/papers15/Abstracts/IJCAI15-276.html`.

10 David A. Cohen, Peter Jeavons, Christopher Jefferson, Karen E. Petrie, and Barbara M. Smith. Symmetry definitions for constraint satisfaction problems. *Constraints*, 11(2-3):115–137, 2006. `doi:10.1007/s10601-006-8059-8`.

11 David A. Cohen and Peter G. Jeavons. The complexity of constraint languages. In Francesca Rossi, Peter van Beek, and Toby Walsh, editors, *Handbook of Constraint Programming*, pages 245–280. Elsevier, 2006.

12 Martin C. Cooper. Beyond consistency and substitutability. In Barry O'Sullivan, editor, *CP 2014*, volume 8656 of *Lecture Notes in Computer Science*, pages 256–271. Springer, 2014. `doi:10.1007/978-3-319-10428-7_20`.

13 Martin C. Cooper, David A. Cohen, and Peter Jeavons. Characterising tractable constraints. *Artif. Intell.*, 65(2):347–361, 1994. `doi:10.1016/0004-3702(94)90021-3`.

14 Martin C. Cooper, Aymeric Duchein, Achref El Mouelhi, Guillaume Escamocher, Cyril Terrioux, and Bruno Zanuttini. Broken triangles: From value merging to a tractable class of general-arity constraint satisfaction problems. *Artificial Intelligence*, 234:196–218, 2016.

15 Martin C. Cooper and Guillaume Escamocher. Characterising the complexity of constraint satisfaction problems defined by 2-constraint forbidden patterns. *Discrete Applied Mathematics*, 184:89–113, 2015. doi:10.1016/j.dam.2014.10.035.

16 Martin C. Cooper, Peter G. Jeavons, and András Z. Salamon. Generalizing constraint satisfaction on trees: Hybrid tractability and variable elimination. *Artif. Intell.*, 174(9-10):570–584, 2010. doi:10.1016/j.artint.2010.03.002.

17 Martin C. Cooper, Philippe Jégou, and Cyril Terrioux. A microstructure-based family of tractable classes for CSPs. In Gilles Pesant, editor, *CP 2015*, volume 9255 of *Lecture Notes in Computer Science*, pages 74–88. Springer, 2015. doi:10.1007/978-3-319-23219-5_6.

18 Martin C. Cooper and Stanislav Živný. Hybrid tractability of valued constraint problems. *Artificial Intelligence*, 175(9-10):1555–1569, 2011. URL: http://zivny.cz/publications/cz11aij-preprint.pdf, doi:10.1016/j.artint.2011.02.003.

19 Martin C. Cooper and Stanislav Živný. Tractable triangles and cross-free convexity in discrete optimisation. *J. Artif. Intell. Res. (JAIR)*, 44:455–490, 2012. doi:10.1613/jair.3598.

20 Martin C. Cooper and Stanislav Živný. The power of arc consistency for CSPs defined by partially-ordered forbidden patterns. In *31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS*, 2016.

21 Víctor Dalmau and Daniel K. Ford. Generalized satisfiability with limited occurrences per variable: A study through delta-matroid parity. In *Proceedings of the 28th International Symposium on Mathematical Foundations of Computer Science (MFCS'03)*, volume 2747 of *Lecture Notes in Computer Science*, pages 358–367. Springer, 2003. doi:10.1007/978-3-540-45138-9_30.

22 Philippe David. Using pivot consistency to decompose and solve functional CSPs. *J. Artif. Intell. Res. (JAIR)*, 2:447–474, 1995. doi:10.1613/jair.167.

23 Rina Dechter. From local to global consistency. *Artif. Intell.*, 55(1):87–108, 1992. doi:10.1016/0004-3702(92)90043-W.

24 Rina Dechter. *Constraint processing*. Elsevier Morgan Kaufmann, 2003. URL: http://www.elsevier.com/wps/find/bookdescription.agents/678024/description.

25 Reinhard Diestel. *Graph Theory*. Springer, fourth edition, 2010.

26 Zdeněk Dvořák and Martin Kupec. On Planar Boolean CSP. In *Proceedings of the 42nd International Colloquium on Automata, Languages and Programming (ICALP'15)*, volume 9134 of *Lecture Notes in Computer Science*, pages 432–443. Springer, 2015.

27 Tomás Feder. Fanout limitations on constraint systems. *Theoretical Computer Science*, 255(1-2):281–293, 2001. doi:10.1016/S0304-3975(99)00288-1.

28 Eugene C. Freuder. A sufficient condition for backtrack-free search. *J. ACM*, 29(1):24–32, 1982.

29 Eugene C. Freuder. A sufficient condition for backtrack-bounded search. *J. ACM*, 32:755–761, 1985.

30 Peter Fulla and Stanislav Živný. On Planar Valued CSPs. In *Proceedings of the 41st International Symposium on Mathematical Foundations of Computer Science (MFCS'16)*, volume 58 of *LIPIcs*, pages 39:1–39:14. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2016. Full version arXiv:1602.06323. doi:10.4230/LIPIcs.MFCS.2016.39.

31 Martin J. Green and David A. Cohen. Domain permutation reduction for constraint satisfaction problems. *Artif. Intell.*, 172(8-9):1094–1118, 2008.

32 Martin Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *J. ACM*, 54(1):1–24, March 2007.

33 Martin Grötschel, Laszlo Lovasz, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169–198, 1981.

34 Pavol Hell and Jaroslav Nešetřil. Colouring, constraint satisfaction, and complexity. *Computer Science Review*, 2(3):143–163, 2008. `doi:10.1016/j.cosrev.2008.10.003`.

35 Pawel M. Idziak, Petar Markovic, Ralph McKenzie, Matthew Valeriote, and Ross Willard. Tractability and learnability arising from algebras with few subpowers. *SIAM J. Comput.*, 39(7):3023–3037, 2010. `doi:10.1137/090775646`.

36 Peter G. Jeavons and Martin C. Cooper. Tractable constraints on ordered domains. *Artif. Intell.*, 79(2):327–339, 1995.

37 Philippe Jégou. Decomposition of domains based on the micro-structure of finite constraint-satisfaction problems. In Richard Fikes and Wendy G. Lehnert, editors, *Proceedings of the 11th National Conference on Artificial Intelligence (AAAI'93)*, pages 731–736. AAAI Press / The MIT Press, 1993. URL: `http://www.aaai.org/Library/AAAI/1993/aaai93-109.php`.

38 Alexandr Kazda, Vladimir Kolmogorov, and Michal Rolínek. Even delta-matroids and the complexity of planar Boolean CSPs. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'17)*, 2017.

39 Vladimir Kolmogorov, Michal Rolínek, and Rustem Takhanov. Effectiveness of structural restrictions for hybrid CSPs. In *Proceedings of the 26th International Symposium on Algorithms and Computation (ISAAC'15)*, volume 9472 of *Lecture Notes in Computer Science*, pages 566–577. Springer, 2015. `doi:10.1007/978-3-662-48971-0_48`.

40 Vladimir Kolmogorov, Johan Thapper, and Stanislav Živný. The power of linear programming for general-valued csps. *SIAM J. Comput.*, 44(1):1–36, 2015. `doi:10.1137/130945648`.

41 Vladimir Kolmogorov and Stanislav Živný. The complexity of conservative valued CSPs. *Journal of the ACM*, 60(2), 2013. Article No. 10. URL: `http://zivny.cz/publications/kz13jacm-preprint.pdf`, `doi:10.1145/2450142.2450146`.

42 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 229–261. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2017. `doi:10.4230/DFU.Vol7.15301.229`.

43 Achref El Mouelhi, Philippe Jégou, and Cyril Terrioux. Different classes of graphs to represent microstructures for CSPs. In Madalina Croitoru, Sebastian Rudolph, Stefan Woltran, and Christophe Gonzales, editors, *Graph Structures for Knowledge Representation and Reasoning – Third International Workshop, GKR 2013*, volume 8323 of *Lecture Notes in Computer Science*, pages 21–38. Springer, 2013. `doi:10.1007/978-3-319-04534-4_3`.

44 Achref El Mouelhi, Philippe Jégou, and Cyril Terrioux. A hybrid tractable class for non-binary CSPs. *Constraints*, 20(4):383–413, 2015. `doi:10.1007/s10601-015-9185-y`.

45 Kazuo Murota. *Discrete Convex Analysis.* SIAM, 2003.

46 Wady Naanaa. Unifying and extending hybrid tractable classes of CSPs. *J. Exp. Theor. Artif. Intell.*, 25(4):407–424, 2013.

47 Justin K. Pearson and Peter G. Jeavons. A survey of tractable constraint satisfaction problems. Technical Report CSD-TR-97-15, Royal Holloway, University of London, July 1997.

48 Jean-Charles Régin. A filtering algorithm for constraints of difference in CSPs. In *12th National Conference on Artificial Intelligence (AAAI'94)*, volume 1, pages 362–367, 1994.

49 Neil Robertson and Paul D. Seymour. Graph minors. XX. Wagner's conjecture. *J. Comb. Theory, Ser. B*, 92(2):325–357, 2004. `doi:10.1016/j.jctb.2004.08.001`.

50 András Z. Salamon and Peter G. Jeavons. Perfect constraints are tractable. In Peter J. Stuckey, editor, *CP 2008*, volume 5202 of *Lecture Notes in Computer Science*, pages 524–528. Springer, 2008. doi:10.1007/978-3-540-85958-1_35.

51 Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24 of *Algorithms and Combinatorics*. Springer, 2003.

52 Rustem Takhanov. Hybrid (V)CSPs and algebraic reductions. *CoRR*, abs/1506.06540, 2015. URL: http://arxiv.org/abs/1506.06540.

53 Johan Thapper and Stanislav Živný. The complexity of finite-valued CSPs. *Journal of the ACM*, 63(4), 2016. Article No. 37. URL: http://zivny.cz/publications/tz16jacm-preprint.pdf, doi:10.1145/2974019.

54 Peter van Beek and Rina Dechter. Constraint tightness and looseness versus local and global consistency. *J. ACM*, 44(4):549–566, 1997. doi:10.1145/263867.263499.

55 Willem Jan van Hoeve and Irit Katriel. Global constraints. In Francesca Rossi, Peter van Beek, and Toby Walsh, editors, *Handbook of Constraint Programming*, chapter 6, pages 169–208. Elsevier, 2006.

56 Willem Jan van Hoeve, Gilles Pesant, and Louis-Martin Rousseau. On global warming: Flow-based soft global constraints. *Journal of Heuristics*, 12(4-5):347–373, 2006. doi:10.1007/s10732-006-6550-4.

Backdoor Sets for CSP*

Serge Gaspers[1], Sebastian Ordyniak[2], and Stefan Szeider[3]

1 **UNSW Australia (The University of New South Wales), Sydney, Australia; and Data61 (formerly: NICTA), CSIRO, Australia**
sergeg@cse.unsw.edu.au

2 **Algorithms and Complexity Group, TU Wien, Vienna, Austria**
ordyniak@ac.tuwien.ac.at

3 **Algorithms and Complexity Group, TU Wien, Vienna, Austria**
szeider@ac.tuwien.ac.at

Abstract

A backdoor set of a CSP instance is a set of variables whose instantiation moves the instance into a fixed class of tractable instances (an island of tractability). An interesting algorithmic task is to find a small backdoor set efficiently: once it is found we can solve the instance by solving a number of tractable instances. Parameterized complexity provides an adequate framework for studying and solving this algorithmic task, where the size of the backdoor set provides a natural parameter. In this survey we present some recent parameterized complexity results on CSP backdoor sets, focusing on backdoor sets into islands of tractability that are defined in terms of constraint languages.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Backdoor sets, Constraint satisfaction problems, Parameterized complexity, Polymorphisms

Digital Object Identifier 10.4230/DFU.Vol7.15301.137

1 Introduction

The Constraint Satisfaction Problem (CSP) is a central and generic computational problem which provides a common framework for many theoretical and practical applications [32]. An instance of CSP consists of a collection of variables that must be assigned values subject to constraints, where each constraint is given in terms of a relation whose tuples specify the allowed combinations of values for specified variables. The problem was originally formulated by Montanari [47], and has been found equivalent to the homomorphism problem for relational structures [19] and the problem of evaluating conjunctive queries on databases [37]. In general, CSP is NP-complete. A central line of research is concerned with the identification of classes of instances for which CSP can be solved in polynomial time. Such classes are often called "islands of tractability" [37, 38].

A prominent way of defining islands of tractability for CSP is to restrict the relations that may occur in the constraints to a fixed set Γ, called a *constraint language*. A finite constraint language is *tractable* if CSP restricted to instances using only relations from Γ, denoted

* NICTA was funded by the Australian Government through the Department of Communications and the Australian Research Council (ARC) through the ICT Centre of Excellence Program. Serge Gaspers is the recipient of an ARC Future Fellowship (FT140100048) and acknowledges support under the ARC's Discovery Projects funding scheme (DP150101134). Stefan Szeider acknowledges the support by the Austrian Science Fund (FWF), grant reference P26696.

CSP(Γ), can be solved in polynomial time. Schaefer's famous Dichotomy Theorem [52] identifies all islands of tractability in terms of tractable constraint languages over the two-element domain. Since then, many extensions and generalizations of this result have been obtained [11, 34, 39, 53]. The Dichotomy Conjecture of Feder and Vardi [18] states that for every finite constraint language Γ, CSP(Γ) is either NP-complete or solvable in polynomial time. Schaefer's Dichotomy Theorem shows that the conjecture holds for two-element domains; more recently, Bulatov [4] showed the conjecture to be true for three-element domains.

If a CSP instance does not belong to a known island of tractability, none of the above tractability results apply. What if an instance does not belong to an island, but is "close" to an island in a certain way? Can we exploit this closeness algorithmically and possibly scale the island's tractability to the considered instance? In order to answer this question one needs to provide a definition for the distance of a CSP instance from an island of tractability (or, more generally, from a class $\mathcal{H}$ of CSP instances). The notion of (strong or weak) backdoor sets, introduced by Williams et al. [55], provides natural distance measures. A *strong backdoor set* of a CSP instance $\mathcal{I}$ into a class $\mathcal{H}$ of CSP instances is a set B of variables of $\mathcal{I}$ such that for all instantiations of the variables in B, the reduced instance belongs to $\mathcal{H}$ (we provide a more formal definition in Section 2.5). The set B is a *weak backdoor set* if for at least one instantiation the reduced instance is satisfiable and belongs to $\mathcal{H}$.

Once we know a strong backdoor set of size k into an island of tractability $\mathcal{H}$ for a CSP instance $\mathcal{I}$ over a finite domain of size d, we can solve $\mathcal{I}$ by solving at most d^k many tractable instances that arise by all the possible instantiations of the backdoor set ($\mathcal{I}$ is satisfiable if and only if at least one of the reduced instances is). Similarly, if we know a weak backdoor set of size k, we can actually find a satisfying assignment again by solving all reduced instances that arise from instantiating the backdoor set and belong to $\mathcal{H}$. The size of a smallest backdoor set provides a notion of distance of the instance from $\mathcal{H}$.

Overall, if we can solve an instance $\mathcal{I}$ of a class $\mathcal{H}$ in polynomial time, say in time $p(|\mathcal{I}|)$, then we can solve an instance for which we know a strong backdoor set of size k into $\mathcal{H}$ in time $d^k p(|\mathcal{I}|)$. This is an exponential running time with the special feature that it is exponential not in the instance size $|\mathcal{I}|$, but in the domain size and backdoor set size only. Problems that admit solution of this type are called *fixed-parameter tractable* [13]. In fact, fixed-parameter tractability provides a desirable way of scaling with the parameter (in this case, the backdoor set size), much preferred over a scaling of the form $|\mathcal{I}|^k$, where the order of the polynomial depends on k.

The *backdoor set approach* for CSP consistes of two parts, first finding a possibly small backdoor set and second using the backdoor set to solve the CSP instance.

This brings up the question: under which circumstances one can efficiently detect a weak or strong backdoor set of size at most k, if one exists? Stated more specifically: under which circumstances is the problem of detecting a weak or strong backdoor set fixed-parameter tractable when parameterized by the size of a smallest backdoor set?

The systematic study of the parameterized complexity of this backdoor set detection problem in the context of propositional satisfiability (SAT) was initiated by Nishimura et al. [48], and has since then received a lot of attention (see the survey paper [29]). Over the last few years, this research has been extended to the area of CSP and the present article provides a survey for some of the results in this direction. Namely we will focus on strong backdoor sets into classes of CSP instances defined via restrictions on the allowed constraint languages.

This survey is structured as follows: In Section 2 we provide the preliminaries about CSP, parameterized complexity, and the backdoor set approach. We also show the first very general results about the application of the backdoor sets approach to CSP that will provide the skeleton for the results in the remaining sections. Sections 3–6 cover the main results of this survey, i.e., they cover the application of strong backdoor sets to CSP using more and more general and evolved base classes defined via restrictions on the constraint language. Starting with base classes defined via a single constraint language in Section 3 the exposition goes on to cover bases classes defined via finite and infinite sets of constraint languages in Sections 4 and 5. The results presented in Sections 3 and 4 are based on [28] and the results from Section 5 are based on [26]. Section 5.1 then gives an overview how the results in Section 5 can be applied for Valued CSP and are based on [25]. Section 6 is based on recent work [27] and outlines how even large backdoor sets can be exploited, as long as the backdoor set induces a graph with a sufficiently simple structure. Section 7 is devoted to a brief exposition of related work. We conclude in Section 8.

2 Preliminaries

2.1 Constraint Satisfaction

Let $\mathcal{V}$ be an infinite set of variables and $\mathcal{D}$ a finite set of values. A *constraint of arity ρ over $\mathcal{D}$* is a pair (S, R) where $S = (x_1, \dots, x_\rho)$ is a sequence of variables from $\mathcal{V}$ and $R \subseteq \mathcal{D}^\rho$ is a ρ-ary relation. The set $\text{var}(C) = \{x_1, \dots, x_\rho\}$ is called the *scope* of C. A *value assignment* (or *assignment*, for short) $\alpha : X \to \mathcal{D}$ is a mapping defined on a set $X \subseteq \mathcal{V}$ of variables. An assignment $\alpha : X \to \mathcal{D}$ *satisfies* a constraint $C = ((x_1, \dots, x_\rho), R)$ if $\text{var}(C) \subseteq X$ and $(\alpha(x_1), \dots, \alpha(x_\rho)) \in R$. For a set of constraints $\mathcal{I}$ we write $\text{var}(\mathcal{I}) = \bigcup_{C\in\mathcal{I}} \text{var}(C)$ and $\text{rel}(\mathcal{I}) = \{\, R : (S, R) \in C, C \in \mathcal{I} \,\}$.

A finite set $\mathcal{I}$ of constraints is *satisfiable* if there exists an assignment that satisfies all the constraints in $\mathcal{I}$. The *Constraint Satisfaction Problem* (CSP, for short) asks, given a finite set $\mathcal{I}$ of constraints, whether $\mathcal{I}$ is satisfiable. Therefore we refer to a finite set of constraints also as a *CSP instance*.

Let $\alpha : X \to \mathcal{D}$ be an assignment. For a ρ-ary constraint $C = (S, R)$ with $S = (x_1, \dots, x_\rho)$ we denote by $C|_\alpha$ the constraint (S', R') obtained from C as follows. R' is obtained from R by (i) deleting all tuples $(d_1, \dots, d_\rho)$ from R for which there is some $1 \leq i \leq \rho$ such that $x_i \in X$ and $\alpha(x_i) \neq d_i$, and (ii) removing from all remaining tuples all coordinates d_i with $x_i \in X$. S' is obtained from S by deleting all variables x_i with $x_i \in X$. For a set $\mathcal{I}$ of constraints we define $\mathcal{I}|_\alpha$ as $\{\, C|_\alpha : C \in \mathcal{I} \,\}$.

A *constraint language* (or *language*, for short) Γ over a finite domain $\mathcal{D}$ is a set Γ of relations (of possibly various arities) over $\mathcal{D}$. By CSP(Γ) we denote CSP restricted to instances $\mathcal{I}$ with $\text{rel}(\mathcal{I}) \subseteq \Gamma$. A constraint language Γ is *globally tractable* if there is a polynomial-time algorithm solving any CSP instance $\mathcal{I} \in \text{CSP}(\Gamma)$ in polynomial time. A constraint language Γ is *efficiently recognizable* if there is an algorithm, which, for any CSP instance $\mathcal{I}$, determines whether $\text{rel}(\mathcal{I}) \subseteq \Gamma$ in time polynomial in $|\mathcal{I}|$. Clearly, every finite constraint language is efficiently recognizable.

2.2 Polymorphisms

Here we introduce the concept of polymorphism, one of the most common ways to define infinite (tractable) constraint languages. Let $\mathcal{D}$ be a finite set of values, ρ and n be natural numbers, let $R \subseteq \mathcal{D}^\rho$, and let $t \in R$. We denote by $t[i]$ the i-th coordinate of t, where i is a natural number with $1 \leq i \leq \rho$. A *n-ary operation over $\mathcal{D}$* is a function from $\mathcal{D}^n$ to $\mathcal{D}$. We say R is *closed* under some n-ary operation φ over $\mathcal{D}$ if R contains the tuple

$\langle \varphi(t_1[1], \dots, t_n[1]), \dots, \varphi(t_1[\rho], \dots, t_n[\rho]) \rangle$ for every sequence $t_1, \dots, t_n$ (of not necessarily distinct) tuples in R.

For a sequence $t_1, \dots, t_n$ of tuples of a relation R we will often write $\varphi[t_1, \dots, t_n]$ to denote the tuple $\langle \varphi(t_1[1], \dots, t_n[1]), \dots, \varphi(t_1[\rho], \dots, t_n[\rho]) \rangle$. The operation φ is also said to be a *polymorphism of R*.

Let φ be an n-ary operation over $\mathcal{D}$. We denote by $\Gamma(\varphi)$ the constraint language over $\mathcal{D}$ consisting of all relations that are closed under φ and we write $\mathrm{CSP}(\varphi)$ as an abbreviation for $\mathrm{CSP}(\Gamma(\varphi))$. We say that the operation φ is *tractable* if $\Gamma(\varphi)$ is globally tractable. We extend the notion of closedness under the polymorphism φ from relations to constraints and entire CSP instances in the natural way, i.e., we say that a constraint $C = (S, R)$ of a CSP instance $\mathcal{I}$ is closed under the operation φ if $R \in \Gamma(\varphi)$ and we say that the same applies to the CSP instance $\mathcal{I}$ if $\mathcal{I} \in \mathrm{CSP}(\varphi)$.

2.3 Base Classes

As base classes for the backdoor set approach we use classes of CSP instances that are defined via (possibly singular) sets of constraint languages. We will consider two basic types of constraint languages, finite and infinite constraint languages. Whereas finite constraint languages will always be represented explicitly, we will characterize infinite constraint languages via polymorhisms and sets of infinite constraint languages via types of polymorphisms.

The following are well-known types of operations.

- An operation $\varphi : D \to D$ is *constant* if there is a $d \in D$ such that for every $d' \in D$, it holds that $\varphi(d') = d$;
- An operation $\varphi : D^n \to D$ is *idempotent* if for every $d \in D$ it holds that $\varphi(d, \dots, d) = d$;
- An operation $\varphi : D^n \to D$ is *conservative* if for every $d_1, \dots, d_n \in D$ it holds that $\varphi(d_1, \dots, d_n) \in \{d_1, \dots, d_n\}$;
- An operation $\varphi : D^2 \to D$ is a *min/max* operation if there is an ordering of the elements of D such that for every $d, d' \in D$, it holds that $\varphi(d, d') = \varphi(d', d) = \min\{d, d'\}$ or $\varphi(d, d') = \varphi(d', d) = \max\{d, d'\}$, respectively;
- An operation $\varphi : D^3 \to D$ is a *majority* operation if for every $d, d' \in D$ it holds that $\varphi(d, d, d') = \varphi(d, d', d) = \varphi(d', d, d) = d$;
- An operation $\varphi : D^3 \to D$ is an *minority* operation if for every $d, d' \in D$ it holds that $\varphi(d, d, d') = \varphi(d, d', d) = \varphi(d', d, d) = d'$;
- An operation $\varphi : D^3 \to D$ is a *Mal'cev* operation if for every $d, d' \in D$ it holds that $\varphi(d, d, d') = \varphi(d', d, d) = d'$.

We denote by VAL, MINMAX, MAJ, AFF, and MAL the classes of CSP instances $\mathcal{I}$, which are closed under some constant, min/max, majority, minority, or Mal'cev operation, respectively. These are some of the most well-known classes of tractable CSP instances and are closely related to (generalizations of) the well-known Schaefer languages [52].

When applying the backdoor set approach to base classes defined via sets of infinite constraint languages, it will become convenient to define more general types of operations than the standard ones introduced above. Namely, we will define predicates of operations (called tractable polymorphism predicates) allowing us to employ the backdoor set approach.

Let $\mathcal{P}(\varphi)$ be a predicate for the operation φ. We call $\mathcal{P}(\varphi)$ a *tractable polymorphism predicate* if the following conditions hold.

N1. There is a constant $c(\mathcal{P})$ such that for all finite domains D, all operations φ over D for which $\mathcal{P}$ holds are of arity at most $c(\mathcal{P})$.

N2. Given a operation φ and a domain D, one can check in polynomial time whether $\mathcal{P}(\varphi)$ holds on all of the at most $|D|^{c(\mathcal{P})}$ tuples over D,

N3. Every operation for which $\mathcal{P}$ holds is tractable.

For a tractable polymorphism predicate $\mathcal{P}$ we define $\Delta(\mathcal{P})$ to be the set of all constraint languages that are closed under some operation for which $\mathcal{P}$ holds. Note that the classes VAL, MINMAX, MAJ, AFF, and MAL as well as combination of these classes can easily be defined via tractable polymorphism predicates. Moreover also much more general types of operations such as semilattice operations (sometimes called ACI operation) [34] (a generalization of min/max), k-ary near unanimity operations [35, 19] (a generalization of majority), k-ary edge operations [33] (a generalization of Mal'cev), and the two operations of arities three and four [40] that capture the bounded width property [1] (a generalization of semilattice and near unanimity operations) can be defined via tractable polymorphism predicates. Finally, we would like to note here that the property of belonging to a tractable algebraic variety [5] is an example of a tractable polymorphism predicate.

2.4 Parameterized Complexity

A *parameterized problem* Π is a problem whose instances are tuples $(\mathcal{I}, k)$, where $k \in \mathbb{N}$ is called the parameter. We say that a parameterized problem is *fixed parameter tractable* (FPT in short) if it can be solved by an algorithm which runs in time $f(k) \cdot |\mathcal{I}|^{\mathcal{O}(1)}$ for some computable function f; algorithms with a running time of this form are called FPT algorithms. FPT also denotes the class of all fixed-parameter tractable decision problems. The notions of W[i]-*hardness* (for $i \in \mathbb{N}$) are frequently used to show that a parameterized problem is not likely to be FPT. The W[i] classes and FPT are closed under parameterized reductions, which are FPT algorithms reducing any instance $(\mathcal{I}, k)$ of a parameterized problem Π to an instance $(\mathcal{I}', k')$ of a parameterized problem Π' such that $(\mathcal{I}, k)$ is a yes-instance for Π if and only if $(\mathcal{I}', k')$ is a yes-instance for Π', and k' is upper bounded by a function of k. An FPT algorithm for a W[i]-hard problem would imply that the Exponential Time Hypothesis fails [9]. We refer the reader to other sources [14, 23] for an in-depth introduction to parameterized complexity.

We also consider parameterized problems with multiple parameters $k_1, \dots, k_\ell$ or parameterized by a set $T = \{k_1, \dots, k_\ell\}$ of natural numbers, whose instances are tuples $(\mathcal{I}, T)$, where $k_i \in \mathbb{N}$, $1 \le i \le \ell$, are the parameters. Such parameterized problems are equivalent to parameterized problems with a single parameter $k_1 + \cdots + k_\ell$.

2.5 Backdoors

Let $\mathcal{I}$ be an instance of CSP over $\mathcal{D}$ and let $\mathcal{H}$ be a class of CSP instances. A set B of variables of $\mathcal{I}$ is called a *strong backdoor set* into $\mathcal{H}$ (or shortly *strong $\mathcal{H}$-backdoor set*) if for every assignment $\alpha : B \to \mathcal{D}$ it holds that $\mathcal{I}|_\alpha \in \mathcal{H}$. Notice that if we are given a strong backdoor set B of size k into a class $\mathcal{H}$ of CSP instances that can be solved in polynomial time, then it is possible to solve the entire instance in time $|\mathcal{D}|^k \cdot |\mathcal{I}|^{\mathcal{O}(1)}$. In general for a class $\mathcal{H}$ of CSP instances the application of the backdoor set approach usually requires the solution to the following two subproblems.

STRONG $\mathcal{H}$-BACKDOOR SET DETECTION
Input: A CSP instance $\mathcal{I}$ over the same domain as $\mathcal{H}$ and a non-negative integer k.
Question: Find a strong $\mathcal{H}$-backdoor set for $\mathcal{I}$ of cardinality at most k, or determine that no such strong backdoor set exists.

STRONG $\mathcal{H}$-BACKDOOR SET EVALUATION
Input: A CSP instance $\mathcal{I}$ over the same domain as $\mathcal{H}$ and a strong $\mathcal{H}$-backdoor set for $\mathcal{I}$.
Question: Determine whether $\mathcal{I}$ is satisfiable.

We will consider the parameterized complexity of the above problems depending on various base classes $\mathcal{H}$ defined via restrictions on the allowed constraint languages as well as the following parameters:

- `arity` denoting the maximum arity of the given CSP instance,
- `dom` denoting the maximum domain of the given CSP instance, and
- `bd-size` denoting the bound on the backdoor set size given as an input to the STRONG $\mathcal{H}$-BACKDOOR SET DETECTION problem or the size of the backdoor set given as an input to the STRONG $\mathcal{H}$-BACKDOOR SET EVALUATION problem, respectively.
- $\texttt{bd-size}_{\mathcal{H}}$ denoting the smallest size of a strong $\mathcal{H}$-backdoor set for the provided CSP instance.

As it turns out for all the results surveyed here, the complexity of the above problems solely depends on the following two properties of the base class $\mathcal{H}$. For a set $T \subseteq \{\texttt{arity}, \texttt{dom}, \texttt{bd-size}\}$, we say that a class of CSP instances $\mathcal{H}$ is:

- *T-tractable* if there is an FPT-algorithm parameterized by T that solves every CSP instance in $\mathcal{H}$.
- *T-detectable* if there is an FPT algorithm parameterized by T, denoted by $\mathcal{A}_{\mathcal{H}}$, that, given a CSP instance $\mathcal{I}$ and a set $B \subseteq \mathrm{var}(\mathcal{I})$, determines whether B is a strong $\mathcal{H}$-backdoor set of $\mathcal{I}$, and if not, outputs a set $Q \subseteq \mathrm{var}(\mathcal{I}) \setminus B$ whose size can be bounded by a function of T, such that every strong $\mathcal{H}$-backdoor set of $\mathcal{I}$ containing B contains at least one variable in Q.

▶ **Theorem 1.** STRONG $\mathcal{H}$-BACKDOOR SET EVALUATION *is fixed-parameter tractable parameterized by* $T \cup \{\texttt{dom}, \texttt{bd-size}\}$ *for every T-tractable class $\mathcal{H}$ of CSP instances.*

Proof. We solve such an instance $\mathcal{I} \in \mathcal{H}$ by going over all of the at most $\texttt{dom}^{\texttt{bd-size}}$ assignments of the variables in the given backdoor set and checking for each of those whether the reduced instance is satisfiable by using the algorithm implied because $\mathcal{H}$ is T-tractable. ◀

▶ **Theorem 2.** STRONG $\mathcal{H}$-BACKDOOR SET DETECTION *is fixed-parameter tractable parameterized by* $T \cup \{\texttt{bd-size}\}$ *for every T-detectable class $\mathcal{H}$ of CSP instances.*

Proof. We will employ a branching algorithm that employs $\mathcal{A}_{\mathcal{H}}$ as a subroutine. The main ingredient of the algorithm is a recursive function, which is called with a set B of at most k variables representing a partial backdoor set. The algorithm simply returns the value of the recursive function called with the empty set of variables and the recursive function consists of the following steps.

1. The function executes the algorithm $\mathcal{A}_{\mathcal{H}}$ on $\mathcal{I}$ and B.
2. If Step 1 concludes that B is a strong $\mathcal{H}$-backdoor set, then the function returns YES,
3. otherwise, let Q be the set of variables returned by the algorithm $\mathcal{A}_{\mathcal{H}}$ in Step (1). Then,
   - if $|B| = k$, the function returns NO,
   - otherwise the function branches on all variables in Q, i.e., for every variable $q \in Q$, the function calls itself on the set $B \cup \{q\}$. If any of these calls returns YES, then the function returns YES, otherwise it returns NO.

This concludes the description of the algorithm. Its correctness follows from the properties of algorithm $\mathcal{A}_{\mathcal{H}}$. Because of the properties of the algorithm $\mathcal{A}_{\mathcal{H}}$ the number of times that the recursive function calls itself recursively is bounded by a function of T. Moreover since the recursive function does not call itself when $|B| = k$, the depth of the recursion is at most k. It follows that the total number of calls to the recursive function is bounded by a function of $T \cup \{k\}$. Finally, the time required for each call of the recursive function is dominated by the time required by $\mathcal{A}_{\mathcal{H}}$. This shows that SBD($\mathcal{H}$) is fixed-parameter tractable parameterized by $T \cup \{\texttt{bd-size}\}$. ◀

3 Basic Results

In this section we consider the backdoor set approach for classes $\mathcal{H}$ of CSP instances defined by a single constraint language Γ. It is easy to see that any such class $\mathcal{H}$ is $\emptyset$-tractable if and only if the defining constraint language Γ is globally tractable. Hence we obtain from Theorem 1 that if Γ is globally tractable, then Strong $\mathcal{H}$-Backdoor Set Evaluation is fixed-parameter tractable parameterized by $\{\texttt{dom}, \texttt{bd-size}\}$. We will now show that if Γ is additionally efficiently recognizable, then Strong $\mathcal{H}$-Backdoor Set Detection is fixed-parameter tractable parameterized by $\{\texttt{arity}, \texttt{bd-size}\}$. Because of Theorem 2 it is sufficient to show that $\mathcal{H}$ is $\{\texttt{arity}\}$-detectable.

▶ **Lemma 3.** CSP(Γ) *is* $\{\texttt{arity}\}$*-detectable for every efficiently recognizable constraint language* Γ*.*

Proof. Let $\mathcal{H} = \text{CSP}(\Gamma)$. It is sufficient to give the algorithm $\mathcal{A}_{\mathcal{H}}$. Let $\mathcal{I}$ be the given CSP instance, B be the given set of variables of $\mathcal{I}$, let m and t be the number of constraints and the maximum number of tuples in any constraint of $\mathcal{I}$, respectively. Observe that B is a strong $\mathcal{H}$-backdoor set if and only if for every constraint $C = (S, R)$ of $\mathcal{I}$ it holds that $C|_\alpha \in \mathcal{H}$ for every assignment α of the variables in $B \cap S$. By ordering the tuples in R according to the assignments of the variables in $B \cap S$, this can be checked in time $O(t \log t)$ times the time required to determine whether $\mathcal{I} \in \mathcal{H}$. Hence executing this for every constraint requires $O(m \cdot t \log t \cdot |\mathcal{I}|^{O(1)})$ time. If $C|_\alpha \in \mathcal{H}$ for every constraint $C = (S, R)$ and every such assignment of the variables in $B \cap S$, then the algorithm returns Yes. Otherwise there is a constraint $C = (S, R)$ and an assignment α of the variables in $B \cap S$ such that $C|_\alpha \notin \mathcal{H}$. In this case B is not a strong $\mathcal{H}$-backdoor set and the algorithm returns the set $S \setminus B$, which we claim satisfies the properties of the set Q given in the statement of the algorithm. Towards showing this, assume for a contradiction that this is not the case, i.e., there is a strong $\mathcal{H}$-backdoor set B' with $B \subseteq B'$ that does not contain a variable in $S \setminus B$. Note that because $B \subseteq B'$ and B' does not contain any variable in $S \setminus B$, it holds that $B' \cap S = B \cap S$. Hence the assignment α as given above contradicts our assumption that B' is a strong $\mathcal{H}$-backdoor set, because $C|_\alpha \notin \mathcal{H}$. ◀

As an immediate consequence of the above lemma and Theorem 2 we obtain the following.

▶ **Theorem 4.** Strong CSP(Γ)-Backdoor Set Detection *is fixed-parameter tractable parameterized by* $\{\texttt{arity}, \texttt{bd-size}\}$ *for every efficiently recognizable constraint language* Γ*.*

This leads to our main result for base classes $\mathcal{H}$ defined via single constraint languages.

▶ **Corollary 5.** *CSP is fixed-parameter tractable parameterized by* $\{\texttt{arity}, \texttt{dom}, \texttt{bd-size}_{\mathcal{H}}\}$ *for every efficiently recognizable and globally tractable constraint language* Γ*.*

If the constraint language Γ is finite, then the above result can even be improved to fixed-parameter tractability with respect to the single parameter backdoor set size. This is because any finite constraint language has bounded domain and bounded arity, i.e., bounded by some fixed constants say D and R, respectively. Hence any input CSP instance with domain larger than D or with arity at least $R + k$ (where k is the size of the backdoor set) can immediately be identified as a No-instance and thus the above algorithm only needs to be applied to CSP instances of domain at most D and arity at most $R + k$. Since all finite constraint languages are efficiently recognizable, we obtain the following corollary.

▶ **Corollary 6.** *CSP is fixed-parameter tractable parameterized by* $\texttt{bd-size}_{\mathrm{CSP}(\Gamma)}$ *for every globally tractable finite constraint language* Γ.

In the following we show that when considering infinite constraint languages (in particular constraint languages defined via polymorphisms), then it is not possible to drop the arity parameter. In particular, we will show that Strong CSP(φ)-Backdoor Set Detection is not fixed-parameter tractable parameterized by the size of the backdoor set alone for any tractable idempotent operation φ.

▶ **Theorem 7.** Strong CSP(φ)-Backdoor Set Detection *is fixed-parameter intractable (*W[2]*-hard) parameterized by* `bd-size` *for every tractable idempotent operation* φ*, even for CSP instances over the Boolean domain.*

Proof. We start by introducing what we call "Boolean barriers" of operations since they form the basis of the proof. Let $\varphi : D^n \to D$ be an n-ary operation over D. We say a set λ of $r(\lambda)$-ary tuples over $\{0, 1\}$ is a *Boolean barrier* for φ if there is a sequence $\langle t_1, \ldots, t_n\rangle$ of (not necessarily distinct) tuples in λ such that $\varphi(t_1, \ldots, t_n) \notin \lambda$. We call a Boolean barrier λ of φ *minimal* if $|\lambda|$ is minimal over all Boolean barriers of φ. For an operation φ, we denote by $\lambda(\varphi)$ a minimal Boolean barrier of φ. For our reduction below, we will employ the fact that every tractable operation φ has a non-empty Boolean barrier of finite size. The reason that a Boolean barrier must exist is simply because if φ would not have a Boolean barrier then every Boolean CSP instance would be closed under φ and thus tractable, which unless P = NP is not possible. To see that $\lambda(\varphi)$ is finite first note that $|\lambda(\varphi)|$ is at most as large as the arity r_φ of φ. Moreover, $r(\lambda) \leq 2^{r_\varphi}$ because there are at most 2^{r_φ} distinct r_φ-ary tuples over $\{0, 1\}$. Hence the size of $\lambda(\varphi)$ is at most $r_\varphi \cdot 2^{r_\varphi}$.

We are now ready to show the theorem via a parameterized reduction from the well-known W[2]-hard Hitting Set problem [13]. Let $\langle U, \mathcal{F}, k\rangle$ be an instance of Hitting Set, where U is a set (often refered to as the universe), $\mathcal{F}$ is a familly of subsets of U and k is a non-negative integer. Note that the Hitting Set problem asks whether there is a subsets $H \subseteq U$ of the universe U with cardinality and most k such that $H \cap F \neq \emptyset$ for every $F \in \mathcal{F}$. We construct a CSP instance $\mathcal{I}$ such that $(U, \mathcal{F})$ has a hitting set of size at most k if and only if $\mathcal{I}$ has a strong CSP(φ)-backdoor set of size at most k.

In the following let $\lambda(\varphi) = \{t_1, \ldots, t_n\}$ and r denote the arity of the tuples in $\lambda(\varphi)$. The variables of $\mathcal{I}$ are $\{\, x_u : u \in U \,\} \cup \{\, o_1(F), \ldots, o_r(F) : F \in \mathcal{F} \,\}$. Furthermore, for every $F \in \mathcal{F}$ with $F = \{u_1, \ldots, u_{|F|}\}$, C contains a constraint $R(F)$ with scope $\langle o_1(F), \ldots, o_r(F), x_{u_1}, \ldots, x_{u_{|F|}}\rangle$ whose relation contains the row

$$t_i[1], \ldots, t_i[r], \underbrace{\langle i \bmod 2, \ldots, i \bmod 2\rangle}_{|F| \text{ times}}$$

for every i in $1 \leq i \leq n$. This completes the construction of $\mathcal{I}$. Suppose that $\mathcal{F}$ has a hitting set B of size at most k. We claim that $B_u = \{\, x_u : u \in B \,\}$ is a strong CSP(φ)-backdoor set of $\mathcal{I}$. Let α be an assignment of the variables in B. We claim that $\mathcal{I}|_\alpha$ is closed under φ and hence B_u is a strong CSP(φ)-backdoor set of $\mathcal{I}$. Note that because φ is idempotent every relation containing only a single tuple is closed under φ, it holds that $|\lambda(\varphi)| > 1$. Because B

is a hitting set of H, it follows that every relation of $\mathcal{I}|_\alpha$ contains at least one tuple less than the corresponding relation in $\mathcal{I}$. Hence any relation of $\mathcal{I}|_\alpha$ contains less tuples than $|\lambda(\varphi)|$, which is the minimal size of any boolean barrier for φ, which implies that $\mathcal{I}|_\alpha$ is closed under φ, as required.

For the reverse direction, suppose that $\mathcal{I}$ has a strong CSP(φ)-backdoor set B of size at most k. Because no constraint of $\mathcal{I}$ is closed under φ, we obtain that B has to contain at least one variable from every constraint of $\mathcal{I}$. Since the only variables that are shared between $R(F)$ and $R(F')$ for distinct $F, F' \in \mathcal{F}$ are the variables in $\{\, x_u : u \in U \,\}$, it follows that B is a hitting set of size at most k for $\mathcal{F}$, as required. ◀

4 Heterogeneous Base Classes

In the previous section, we considered so-called *homogeneous* base classes defined via a single globally tractable constraint language. In this section we introduce a more general form of base classes (called *heterogeneous* base classes) that are defined via a set of globally tractable constraint languages. In particular, given a set of globally tractable constraint languages Δ, let CSP(Δ) be the class of all CSP instances in $\bigcup_{\Gamma \in \Delta} \text{CSP}(\Gamma)$. The size of a backdoor set into a heterogeneous base class can be much smaller than the minimum size of a backdoor set into any of its homogeneous base classes. Therefore, backdoor sets into heterogeneous base classes are considerably more powerful but also more complicated to handle. Even the evaluation of backdoor sets into heterogeneous base classes needs to be handled carefully. Namely, because the given set Δ can be infinite the class CSP(Δ) of CSP instances is not a priory $\emptyset$-tractable even if the considered constraint languages are globally tractable.

We start by showing that backdoor sets into heterogeneous base classes can be arbitrarily smaller than backdoor sets into their homogeneous counterparts. For the construction of the example let φ_{min} be the *min*-type operation and let φ_{maj} be the *majority*-type operation both defined on the ordered domain $(0,1)$.

▶ **Proposition 8.** *For every natural number n, there is a CSP instance $\mathcal{I}_n$ such that $\mathcal{I}_n$ has a strong* CSP$(\{\varphi_{\text{min}}, \varphi_{\text{maj}}\})$*-backdoor set of size one but every strong* CSP$(\{\varphi_{\text{min}}\})$*-backdoor set and every strong* CSP$(\{\varphi_{\text{maj}}\})$*-backdoor set of $\mathcal{I}_n$ has size at least n.*

Proof. Let MAJ$[a,b,c,d]$ be the constraint with scope (a,b,c,d) and whose relation contains all possible tuples that set d to 0 except for the tuple $(1,1,1,0)$. Then MAJ$[a,b,c,d]$ is not closed under φ_{maj} but happens to be closed under φ_{min}. Similarly, let MIN$[a,b,c]$ be the constraint with scope (a,b,c) and whose relation contains the tuples $(0,1,1)$ and $(1,0,1)$. Then MIN$[a,b,c]$ is not closed under φ_{min} but happens to be closed under φ_{maj}. We claim that the CSP instance $\mathcal{I}_n$ with variables $\{y_1,\dots,y_{3n}\} \cup \{z_1,\dots,z_{2n}\} \cup \{x\}$ and constraints MAJ$_i$ = MAJ$[y_{3i+1}, y_{3i+2}, y_{3i+3}, x]$ and MIN$_i$ = MIN$[z_{2i+1}, z_{2i+2}, x]$ for every i with $0 \leq i \leq n-1$, satisfies the claim of the proposition. Towards showing this first note that $\{x\}$ is a strong CSP$(\{\varphi_{\text{min}}, \varphi_{\text{maj}}\})$-backdoor set of $\mathcal{I}_n$ of size 1. This is because for the assignment α with $\alpha(x) = 0$, it holds that $\mathcal{I}_n|_\alpha \in$ CSP(φ_{min}) and for the assignment α with $\alpha(x) = 1$, it holds that $\mathcal{I}_n|_\alpha \in$ CSP(φ_{maj}). Moreover it is straightforward to verify that every strong CSP(φ_{min})-backdoor set of $\mathcal{I}_n$ has to contain at least one variable from every constraint MIN$_i$ that is not x and similarly every strong CSP(φ_{maj})-backdoor set of $\mathcal{I}_n$ has to contain at least one variable from every constraint MAJ$_i$ that is not x. Hence the size of every strong CSP(φ_{min})-backdoor set as well as any strong CSP(φ_{maj})-backdoor set is at least n. ◀

The following auxiliary lemma provides a useful property for detecting backdoor sets into heterogeneous base classes.

▶ **Lemma 9.** *Let Δ be a set of constraint languages, $\mathcal{I}$ a CSP instance, and B a set of variables of $\mathcal{I}$. If there is an assignment α of the variables in B such that $\mathcal{I}|_\alpha \notin \mathrm{CSP}(\Delta)$, then any strong $\mathrm{CSP}(\Delta)$-backdoor set B' with $B \subseteq B'$ contains at least one variable from the set $Q = (\bigcup_{\Gamma\in\Delta} \mathrm{var}(C^\Gamma)) \setminus B$, where for any $\Gamma \in \Delta$, C^Γ is a constraint of $\mathcal{I}|_\alpha$ such that $C^\Gamma \notin \mathrm{CSP}(\Gamma)$.*

Proof. Let $\mathcal{H} = \mathrm{CSP}(\Delta)$. Assume for a contradiction that this is not the case, i.e., there is a strong $\mathcal{H}$-backdoor set B' for $\mathcal{I}$ with $B \subseteq B'$ and B' does not contain any variable from Q. Because B is not a strong $\mathcal{H}$-backdoor set there is an assignment α such that $\mathcal{I}|_\alpha \notin \mathcal{H}$. Let α' be any assignment of the variables in B' that agrees with α on the variables in B. Because B' is a strong $\mathcal{H}$-backdoor set, it follows that there is a constraint language $\Gamma \in \Delta$ such that $\mathcal{I}|_{\alpha'} \in \mathrm{CSP}(\Gamma)$. We claim that B' contains at least one variable from every constraint $C \in \mathcal{I}|_\alpha$ with $C \notin \mathrm{CSP}(\Gamma)$. For if not, then let C be such a constraint with an empty intersection with B'. It follows that $C \in \mathcal{I}|_{\alpha'}$ but because $C \notin \mathrm{CSP}(\Gamma)$ this contradicts our assumption that $\mathcal{I}|_{\alpha'} \in \mathrm{CSP}(\Gamma)$. ◀

In the following we will give our first example of a heterogeneous base class, which can be employed for the backdoor set approach. Namely, we will show this for any finite set Δ of finite globally tractable constraint languages. We start by showing that $\mathrm{CSP}(\Delta)$ is $\emptyset$-tractable, which, because of Theorem 1, implies that STRONG $\mathrm{CSP}(\Delta)$-BACKDOOR SET EVALUATION is fixed-parameter tractable parameterized by `bd-size` (because the domain is finite).

▶ **Proposition 10.** $\mathrm{CSP}(\Delta)$ *is $\emptyset$-tractable for every finite set Δ of finite and globally tractable constraint languages.*

Proof. Let $\mathcal{H} = \mathrm{CSP}(\Delta)$. We can solve a CSP instance $\mathcal{I} \in \mathcal{H}$ by going over all constraint languages $\Gamma \in \Delta$, checking whether $\mathcal{I}$ is in $\mathrm{CSP}(\Gamma)$ and if so solving $\mathcal{I}$ in polynomial time. This can be achieved in polynomial time because there are only finitely many constraint languages in Δ and each of them can be recognized in polynomial time (because it is finite). ◀

The next lemma shows that for any such set Δ, $\mathrm{CSP}(\Delta)$ is also {`bd-size`}-detectable.

▶ **Lemma 11.** $\mathrm{CSP}(\Delta)$ *is {`bd-size`}-detectable for every finite set Δ of finite constraint languages.*

Proof. It is sufficient to give the algorithm $\mathcal{A}_{\mathrm{CSP}(\Delta)}$. Let $\mathcal{I}$ be a CSP instance and let B be the given set of variables of $\mathcal{I}$. Because Δ contains only finitely many finite constraint languages, it holds that the maximum domain value D as well as the maximum arity R of any of its languages is also finite. Hence w.l.o.g. we can assume that the maximum domain value of $\mathcal{I}$ is at most D and similarly the maximum arity of $\mathcal{I}$ is at most $R + k$, since otherwise we can simply return NO. To determine whether B is a strong $\mathrm{CSP}(\Delta)$-backdoor set, we test for every assignment α of the variables in B whether $\mathcal{I}|_\alpha \in \mathrm{CSP}(\Gamma)$ for some constraint language $\Gamma \in \Delta$. Because there are at most $D^{|B|}$ assignments of the variables in B, there are finitely many constraint languages in Δ, and verifying whether $\mathcal{I}|_\alpha \in \mathrm{CSP}(\Gamma)$ for some finite constraint language can be done in polynomial time, the total time required by this step of the algorithm is $O(D^{|B|}|\mathcal{I}|^{O(1)})$. If the above test holds for every assignment of the variables in B, then the algorithm returns YES. Otherwise there is an assignment α of the variables in B such that $\mathcal{I}|_\alpha \notin \text{CSP}(\Delta)$. Hence we obtain from Lemma 9 that any strong CSP(Δ)-backdoor set B' with $B \subseteq B'$ contains at least one variable from the set Q as given in the statement of Lemma 9. Because the size of this set Q is at most $|\Delta|(R+k) \in O(k)$ and the set Q can be computed within the running time of the first step of the algorithm, the lemma follows. ◀

As an immediate consequence of the above lemma and Theorem 2, we obtain the following.

▶ **Theorem 12.** STRONG CSP(Δ)-BACKDOOR SET DETECTION *is fixed-parameter tractable parameterized by* `bd-size` *for every finite set* Δ *of finite constraint languages.*

The above discussion leads directly to our main result for heterogeneous base classes defined via finite constraint languages.

▶ **Corollary 13.** *CSP is fixed-parameter tractable parameterized by* $\texttt{bd-size}_{CSP(\Delta)}$ *for every finite set* Δ *of globally tractable finite constraint languages.*

This concludes our discussion for heterogeneous base classes defined via finite constraint languages. In the following we will consider sets of infinite constraint languages defined via a tractable polymorphism predicate $\mathcal{P}$. Namely, let $\mathcal{H} = \text{CSP}(\Delta(\mathcal{P}))$ for a tractable polymorphism predicate $\mathcal{P}$. We will show that CSP is fixed-parameter tractable parameterized by `arity`, `dom`, and the size of a smallest strong $\mathcal{H}$-backdoor set. We will also show that none of these three parameters can be dropped without sacrificing fixed-parameter tractability. Crucial to this result is the fact that even though a tractable polymorphism predicate holds for a potentially infinite set of operations, the number of operations is bounded for a fixed domain.

▶ **Lemma 14.** *Let* $\mathcal{P}$ *be a tractable polymorphism predicate and* D *be a finite set. Then there are at most* $d^{d^{c(\mathcal{P})}}$ *operations on* D *that satisfy* $\mathcal{P}$*, and computing all these operations is fixed-parameter tractable parameterized by* $|D|$*.*

Proof. This follows because there are at most $d^{d^{c(\mathcal{P})}}$ $c(\mathcal{P})$-ary operations on D and because of Property N2 for each of those one can test in polynomial time whether it satisfies $\mathcal{P}$. ◀

We show next that STRONG $\mathcal{H}$-BACKDOOR SET EVALUATION is fixed-parameter tractable parameterized by $\{$`dom`, `bd-size`$\}$. Because of Theorem 1 it is sufficient to show that $\mathcal{H}$ is $\{$`dom`$\}$-tractable.

▶ **Lemma 15.** CSP($\Delta(\mathcal{P})$) *is* $\{$`dom`$\}$*-tractable for every tractable polymorphism predicate* $\mathcal{P}$*.*

Proof. We first compute the set P of all operations on the domain D of $\mathcal{I}$ using Lemma 14. We then check for every such operation $\varphi \in P$ whether $\mathcal{I}$ is closed under φ in time $O(m \cdot t^{c(\mathcal{P})})$, where m is the number of constraints of $\mathcal{I}$ and t is the maximum number of tuples occurring in any constraint of $\mathcal{I}$. If it is we use the fact that φ is tractable to solve $\mathcal{I}$ in polynomial time. ◀

We now turn to the detection of backdoor sets into $\mathcal{H}$.

▶ **Lemma 16.** CSP($\Delta(\mathcal{P})$) *is* $\{$`arity`, `dom`, `bd-size`$\}$*-detectable for every tractable polymorphism predicate* $\mathcal{P}$*.*

Proof. Let $\mathcal{H} = \text{CSP}(\Delta(\mathcal{P}))$. It is sufficient to give the algorithm $\mathcal{A}_\mathcal{H}$. Let $\mathcal{I}$ be the given CSP instance over $\mathcal{D}$ and let B be the given set of variables of $\mathcal{I}$.

We first compute the set P of all operations on $\mathcal{D}$ for which $\mathcal{P}$ holds by employing Lemma 14. Observe that a set B of variables of $\mathcal{I}$ is a strong $\mathcal{H}$-backdoor set if and only if it is a strong CSP(P)-backdoor set, where $\mathrm{CSP}(P) = \bigcup_{\varphi \in P} \mathrm{CSP}(\varphi)$.

To determine whether B is a strong CSP(P)-backdoor set, we test for every assignment α of the variables in B whether $\mathcal{I}|_\alpha \in \mathrm{CSP}(P)$. Because there are at most $\mathtt{dom}^{\mathtt{bd\text{-}size}}$ assignments of the variables in B, at most $\mathtt{dom}^{\mathtt{dom}^{c(\mathcal{P})}}$ operations in P, and verifying whether $\mathcal{I}|_\alpha \in \mathrm{CSP}(\varphi)$ for any $\varphi \in P$ can be achieved in polynomial time, the total time required by this step of the algorithm is at most $O(\mathtt{dom}^{\mathtt{bd\text{-}size}}\mathtt{dom}^{\mathtt{dom}^{c(\mathcal{P})}}|\mathcal{I}|^{O(1)})$. If the above holds for every assignment of the variables in B, then the algorithm returns YES. Otherwise there is an assignment α such that $\mathcal{I}|_\alpha \notin \mathrm{CSP}(P)$. Hence we obtain from Lemma 9 that any strong $\mathcal{H}$-backdoor set B' with $B \subseteq B'$ contains at least one variable from the set Q given in the statement of the lemma. Because the size of this set Q is at most $|P| \cdot \mathtt{arity} \leq \mathtt{dom}^{\mathtt{dom}^{c(\mathcal{P})}} \cdot \mathtt{arity}$ and the set Q can be computed within the running time of the first step of the algorithm, the lemma follows. ◀

The following theorem follows immediately from the above lemma and Theorem 2.

▶ **Theorem 17.** STRONG CSP($\Delta(\mathcal{P})$)-BACKDOOR SET DETECTION *is fixed-parameter tractable parameterized by* $\{\mathtt{arity}, \mathtt{dom}, \mathtt{bd\text{-}size}\}$ *for every tractable polymorphism predicate* $\mathcal{P}$.

The above results naturally lead to our main result of this section.

▶ **Corollary 18.** *CSP is fixed-parameter tractable parameterized by* $\{\mathtt{arity}$, $\mathtt{dom}$, $\mathtt{bd\text{-}size}_{\mathrm{CSP}(\Delta(\mathcal{P}))}\}$ *for every tractable polymorphism predicate* $\mathcal{P}$.

It turns out that we cannot avoid to parameterize by both the maximum domain value and the arity in the above theorem. We have already seen in Theorem 7 that even if we consider just a single idempotent tractable operation φ, then STRONG CSP(φ)-BACKDOOR SET DETECTION is fixed-parameter intractable parameterized by $\{\mathtt{bd\text{-}size}, \mathtt{dom}\}$ (even for boolean domain). Because it is easy to define a tractable polymorphism predicate that only holds for a single idempotent operation, this result generalizes to tractable polymorphism predicates. Hence it only remains to consider the case where we parameterize only by the maximum arity and backdoor set size. The proof of the following theorem can be found in [28, Theorem 12]. Note that since the reduction employed in the hardness result does strongly depend on the considered tractable polymorphism predicate it is difficult to obtain a general result as in Theorem 7, but it can be stated to include some of the arguably most prominent types of operations.

▶ **Theorem 19.** STRONG $\mathcal{H}$-BACKDOOR SET DETECTION *is fixed-parameter intractable (*W[2]*-hard) parameterized by* $\{\mathtt{arity}, \mathtt{bd\text{-}size}\}$ *for every* $\mathcal{H} \in \{$MINMAX, MAJ, AFF, MAL$\}$, *even for CSP instances with arity* 2.

5 Scattered Base Classes

Thus far we have considered a setting in which the reduced CSP instance for each assignment of the backdoor set variables belonged entirely to a single tractable constraint language. To ensure that the reduced CSP instance is tractable, which is sufficient for an application of the backdoor set approach, there is however a natural and more general possibility: Instead of belonging entirely to a single tractable constraint language, the CSP instance could consist of a disjoint union of (pairwise variable disjoint) CSP instances, each belonging to some tractable constraint language. This type of tractable base class is particularly interesting in combination with the backdoor set approach, since now the variables of the backdoor set can be naturally employed to separate the CSP instance into parts only interacting with each other through the variables in the backdoor set.

More specifically, a CSP instance $\mathcal{I}$ is *connected* if either it consists of at most one constraint, or for each partition of $\mathcal{I}$ into nonempty sets $\mathcal{I}_1$ and $\mathcal{I}_2$, there exists at least one constraint $c_1 \in \mathcal{I}_1$ and one constraint $c_2 \in \mathcal{I}_2$ that share at least one variable. A *connected component* of $\mathcal{I}$ is a maximal connected subinstance $\mathcal{I}'$ of $\mathcal{I}$. These notions naturally correspond to the connectedness and connected components of standard graph representations of CSP instances.

Now, given a set Δ of constraint languages, we define the scattered class $\oplus(\Delta)$ of CSP instances $\mathcal{I}$ where each connected component $\mathcal{I}'$ of $\mathcal{I}$ belongs to $\mathrm{CSP}(\Gamma)$ for some $\Gamma \in \Delta$.

We start by showing that backdoor sets into scattered base classes can be arbitrarily smaller than backdoor sets into their heterogeneous counterparts. For the construction of the example let $\varphi_{\min}$ be the *min*-type operation and let φ_{maj} be the *majority*-type operation both defined on the ordered domain $\{0, 1\}$.

▶ **Proposition 20.** *For every natural number n, there is a CSP instance $\mathcal{I}_n$ such that $\mathcal{I}_n$ has a strong $\oplus(\Delta(\{\varphi_{\min}, \varphi_{\mathrm{maj}}\}))$-backdoor set of size zero but every strong $\mathrm{CSP}(\{\varphi_{\min}, \varphi_{\mathrm{maj}}\})$-backdoor set of $\mathcal{I}_n$ has size at least n.*

Proof. Let $\mathrm{MAJ}[a, b, c]$ be the constraint with scope (a, b, c) and whose relation contains all possible tuples except for the tuple $(1, 1, 1)$. Then $\mathrm{MAJ}[a, b, c]$ is not closed under φ_{maj} but it is closed under $\varphi_{\min}$. Similarly, let $\mathrm{MIN}[a, b]$ be the constraint with scope (a, b) and whose relation contains the tuples $(0, 1)$ and $(1, 0)$. Then $\mathrm{MIN}[a, b]$ is not closed under $\varphi_{\min}$ but it is closed under φ_{maj}. We claim that the CSP instance $\mathcal{I}_n$ with variables $\{y_1, \dots, y_{3n}\} \cup \{z_1, \dots, z_{2n}\}$ and constraints $\mathrm{MAJ}_i = \mathrm{MAJ}[y_{3i+1}, y_{3i+2}, y_{3i+3}]$ and $\mathrm{MIN}_i = \mathrm{MIN}[z_{2i+1}, z_{2i+2}]$ for every i with $0 \leq i \leq n - 1$, satisfies the claim of the proposition. Towards showing this first note that the empty set is a strong $\oplus(\Delta((\{\varphi_{\min}, \varphi_{\mathrm{maj}}\}))$-backdoor set of $\mathcal{I}_n$ of size 0. This is because $\mathcal{I}_n$ is the disjoint union of variable disjoint constraints, which are either closed under $\varphi_{\min}$ or under φ_{maj}. However it is straightforward to verify that every strong $\mathrm{CSP}(\{\varphi_{\min}, \varphi_{\mathrm{maj}}\})$-backdoor set of $\mathcal{I}_n$ has to either contain at least one variable from every constraint MIN_i or at least one variable from every constraint MAJ_i. Hence the size of any strong $\mathrm{CSP}(\{\varphi_{\min}, \varphi_{\mathrm{maj}}\})$-backdoor set is at least n. ◀

Below we will present an algorithm that detects scattered base classes. For this algorithm it is convenient to deal only with constraint languages that are closed under assignments (as we will define next). This has the advantage that there is no danger that a partially constructed backdoor set becomes invalidated by adding another variable to the backdoor set. In fact, in the orginal paper which introduced backdoor sets [55], this property is even part of the definition. We will also assume that the considered constraint languages contain a redundant constraint which can be used within the algorithm to artificially connect parts of the instance.

A constraint language Γ is *closed under assignments* if for every $C = (S, R)$ such that $R \in \Gamma$ and every assignment α, it holds that $R' \in \Gamma$ where $C|_\alpha = (S', R')$. The lemma below shows that languages closed under assignments are closely related to semi-conservative languages. For a constraint language Γ over a domain $\mathcal{D}$ we denote by Γ^* the smallest constraint language over $\mathcal{D}$ that contains $\Gamma \cup \{\mathcal{D}^2\}$ and is closed under assignments; notice that Γ^* is uniquely determined by Γ. For a set Δ of constraint languages we denote by Δ^* the set $\{\, \Gamma^* : \Gamma \in \Delta \,\}$

▶ **Lemma 21.** *If Δ is a set of globally tractable semi-conservative constraint languages, then $\oplus(\Delta^*)$ is also globally tractable.*

Proof. Evidently, if a semi-conservative language Γ is globally tractable, then so is Γ^*: first, all constraints of the form $(S, \mathcal{D}^2|_\alpha)$ can be detected in polynomial time and removed from the instance without changing the solution, and then each constraint $C' = (S', R')$ with $R' \in \Gamma^* \setminus \Gamma$ can be expressed in terms of the conjunction of a constraint $C = (S, R)$ with $R \in \Gamma$ and unary constraints over variables in $\text{var}(C) \setminus \text{var}(C')$. Now, if each $\Gamma^* \in \Delta^*$ is globally tractable, also $\oplus(\Delta^*)$ is globally tractable, as we can solve connected components independently. ◀

The main technical result for scattered base classes is the next lemma.

▶ **Theorem 22.** *Let Δ be a finite set of finite constraint languages. Then there is an FPT-algorithm that, given a CSP instance $\mathcal{I}$ and a parameter k, either finds a strong* $\text{CSP}(\oplus(\Delta^*))$*-backdoor set of size at most k or correctly decides that none exists.*

We sketch the main ingredients of the algorithm and refer for the ArXiv version of the original paper [26] for details.

The algorithm uses the technique of iterative compression [50] to transform the problem into a structured subproblem. In this technique, the idea is to start with a sub-instance and a trivial solution for this sub-instance and iteratively expand the sub-instances while compressing the solutions till we solve the problem on the original instance. Specifically, for backdoor detection we are given additional information about the desired solution in the input: we receive an 'old' strong backdoor set which is slightly bigger than our target size, along with information about how this old backdoor set interacts with our target solution.

Further more, the algorithm considers only solutions for instances of the iterative compression problem which have a certain inseparability property and uses an FPT algorithm to test for the presence of such solutions. To be more precise, the algorithm only looks for solutions which leave the omitted part of the old strong backdoor set in a single connected component. Interestingly, even this base case requires the extension of state of the art separator techniques to a CSP setting.

Finally, the general instances of the iterative compression problem are handled using a new pattern replacement technique, which is somehow similar to the protrusion replacement technique [3] but allows the preservation of a much larger set of structural properties (such as containment of disconnected forbidden structures and connectivity across the boundary). This pattern replacement procedure is interleaved with the technique of important separator sequences [44] as well as the above algorithm for inseparable instances.

▶ **Corollary 23.** *Let Δ be a finite set of globally tractable semi-conservative finite constraint languages. Then* $\text{CSP}(\oplus(\Delta))$ *is fixed-parameter tractable parameterized by the backdoor set size.*

Proof. Let $\mathcal{I}$ be the given CSP instance over domain D and k a parameter such that $\mathcal{I}$ has a strong $\text{CSP}(\oplus(\Delta))$-backdoor set of size $\leq k$. Since $\text{CSP}(\oplus(\Delta)) \subseteq \text{CSP}(\oplus(\Delta^*))$, $\mathcal{I}$ has also a strong $\text{CSP}(\oplus(\Delta^*))$-backdoor set of size $\leq k$, and we compute this backdoor set using Theorem 22. Because of Lemma 21, $\text{CSP}(\oplus(\Delta^*))$ is globally tractable. Hence we can use the backdoor set by solving all the instances that arise by instantiating the backdoor set variables. As the considered languages are finite, they are over a finite domain D, and so the number of tractable instances to consider is at most $|D|^k$. ◀

It might be possible to generalize Corollary 23 to sets of constraint languages characterized by tractable polymorphism predicates. Since the algorithm is already quite complicated for the finite case, this has not yet been checked in detail.

5.1 Extension to Valued CSP

Valued CSP (or VCSP for short) is a powerful framework that entails among others the problems CSP and MAX-CSP as special cases [56]. A VCSP instance consists of a finite set of cost functions over a finite set of variables which range over a domain D, and the task is to find an instantiation of these variables that minimizes the sum of the cost functions. The VCSP framework is robust and has been studied in different contexts in computer science. In its full generality, VCSP considers cost functions that can take as values the rational numbers and positive infinity. CSP (feasibility) and Max-CSP (optimization) arise as special cases by limiting the values of cost functions to $\{0, \infty\}$ and $\{0, 1\}$, respectively. Clearly VCSP is in general intractable. Over the last decades much research has been devoted into the identification of tractable VCSP subproblems. An important line of this research (see, e.g., [36, 39, 54]) is the characterization of tractable VCSPs in terms of restrictions on the underlying *valued constraint language* Γ, i.e., a set Γ of cost functions that guarantees polynomial-time solvability of all VCSP instances that use only cost functions from Γ. The VCSP restricted to instances with cost functions from Γ is denoted by VCSP$[\Gamma]$.

The definitions of strong backdoor sets, scattered base classes, etc., generalize straightforwardly from the CSP setting to VCSP, and we omit the details. A valued constraint language is *conservative* if it contains all unary cost functions [39].

Theorem 23 generalizes to VCSP in the following way:

▶ **Theorem 24.** *Let Δ be a finite set of globally tractable conservative valued constraint languages of bounded domain size and bounded arity. Then* VCSP$(\oplus(\Delta))$ *is fixed-parameter tractable parameterized by the backdoor set size.*

Here it is important to note that a valued constraint language of bounded domain and arity can be infinite. Hence the technique used for establishing Theorem 23 does not directly apply. However, it turns out that one can transform the backdoor set detection problem from a general scattered class VCSP$(\oplus(\Delta))$ to a scattered class VCSP$(\oplus(\Delta'))$ over a finite set Δ' of *finite* valued constraint languages. The reduction does not preserve VCSP solutions, but preserves backdoor sets. Once the backdoor set is found, one applies it to the original VCSP instance.

6 Backdoors of Small Treewidth

In this section we outline a new concept that allows us to algorithmically exploit a backdoor set even if it is large, as long as it induces a graph with a sufficiently simple structure. More specifically, we associate with the backdoor set a certain *torso graph* and measure its structure in terms of the widely used graph invariant treewidth. Minimizing this treewidth over all strong backdoor sets X of a CSP instance $\mathcal{I}$ into CSP(Γ) (for a fixed constraint language Γ) gives as a new “hybrid” parameter, the *backdoor-treewidth.*

Let $\mathcal{I}$ be a CSP instance and X a subset of its variables. We define the *torso graph* of $\mathcal{I}$ with respect to X, denoted $\mathsf{torso}_{\mathcal{I}}(X)$, as follows. The vertex set of $\mathsf{torso}_{\mathcal{I}}(X)$ is X, and the graph contains an edge $\{x, y\}$ if x and y appear together in the scope of a constraint or x and y are in the scopes of constraints that belong to the same connected component of $\mathcal{I} - X$ (see Section 5). Here $\mathcal{I} - X$ denotes the CSP instance obtained from $\mathcal{I}$ by deleting the variables $x \in X$ from all constraint scopes and deleting the corresponding entries from the constraint relations.

Let $G = (V, E)$ be a graph. A tree decomposition of G is a pair $(T, \mathcal{X} = \{X_t\}_{t \in V(T)})$ where T is a tree and $\mathcal{X}$ is a collection of subsets of V such that: (i) for each edge $\{u, v\} \in E$ there exists a node t of T such that $\{u, v\} \subseteq X_t$, and (ii) for each $v \in V$, the set $\{t \mid v \in X_t\}$ induces in T a nonempty connected subtree. The width of $(T, \mathcal{X})$ is equal to $\max\{|X_t| - 1 \mid t \in V(T)\}$ and the *treewidth* of G, denoted $\mathsf{tw}(G)$, is the minimum width over all tree decompositions of G.

Let $\mathcal{I}$ be a CSP instance and X a strong backdoor set of $\mathcal{I}$ into CSP(Γ). The *width* of X is the treewidth of the torso graph $\mathsf{torso}_{\mathcal{I}}(X)$, and the *backdoor-treewidth* of $\mathcal{I}$ with respect to Γ is the smallest width over all strong backdoor sets X of $\mathcal{I}$ into CSP(Γ).

▶ **Theorem 25.** *For each finite constraint language Γ, there is an FPT-algorithm that, given a CSP instance $\mathcal{I}$ and a parameter k, either finds a strong* CSP(Γ)*-backdoor set of with at most k or correctly decides that none exists.*

The proof of the theorem makes use of a new notion of "boundaried CSP instances" defined in the spirit of boundaried graphs, and uses a new replacement framework inspired by the graph replacement tools dating back to the results of Fellows and Langston [20], combined with the so-called recursive-understanding technique [31]

Once a backdoor set of small with is found, one can apply standard dynamic programming techniques to solve CSP and #CSP.

▶ **Corollary 26.** *For each finite (#-)tractable constraint language Γ,* CSP *(or* #CSP*, respectively) is fixed-parameter tractable parameterized by the backdoor treewidth with respect to Γ.*

7 Related Work

Related work on backdoor sets for CSP includes a paper by Bessière et at. [2] who consider so-called *partition backdoor sets* which are less general than the backdoor sets we considered in Section 4 (see [28, Section 7]); they also provide some initial empirical results which show that this concept has practical potential. A further work on CSP backdoor sets is a paper by Carbonnel et al. [7] who show W[2]-hardness for strong backdoor set detection when parameterized by the size of the backdoor set, even for CSP-instances with only one constraint (however with unbounded domain and unbounded arity). They also give a fixed-parameter algorithm for strong backdoor set detection parameterized by the size of the backdoor set and the maximum arity of any constraint, if the base class is "h-Helly" for a fixed integer h and under the additional assumption that the domain is a finite subset of the natural numbers, which comes with a fixed ordering (see also the recent survey by Carbonnel and Cooper [8]).

As mentioned at the beginning of this survey, there is much work on backdoor sets in the context of SAT, most of it is covered in the survey paper [29]. More recent additions include the detection of strong backdoor sets with respect to the base class of CNF formulas whose incidence graph is of bounded treewidth [24, 30]. We would like to note that in SAT the application of a partial assignment to a CNF formula results in the deletion of all satisfied clauses, which provides an additional power for strong backdoor sets, and provides an additional challenge for their detection.

The parameterized complexity of finding and using backdoor sets has been studied for several problems besides SAT and CSP, including the satisfiability of quantified Boolean formulas [51], disjunctive answer set programming [21, 22], abductive reasoning [49], abstract argumentation [15], planning [41, 42], and linear temporal logic [46].

Finally, we would like to mention that in the area of graph algorithms the notion of *modulators* [6] is closely related to the concept of backdoor sets. A modulator of a graph G into a fixed graph class $\mathcal{C}$ is a set M of vertices of G such that deleting M from G moves G into the class $\mathcal{C}$. By considering modulators into graph classes $\mathcal{C}$ where certain NP-hard graph problems can be solved in polynomial time, one can often lift the tractability of the problem to fixed-parameter tractability for general graphs, parameterized by the size of the modulator. Therefore the fixed-parameter tractability of the detection of modulators (parameterized by modulator size) is of interest. Important results include modulators to bipartite graphs [43, 50], to chordal graphs [45], and to forests [12]. Recently alternative parameterizations of modulators that are more general than their size have been explored [16, 17].

8 Conclusion

We presented parameterized complexity results on CSP backdoor sets into bases classes defined via restrictions on the constraint languages. The presented results show that the notion of CSP backdoor sets provides an interesting area of research, which on one side builds upon and extends classical CSP-tractability results (in form of the considered base classes), and on the other side uses advanced algorithmic methods from the area of parameterized complexity. There are plenty of questions that are mostly unexplored, such as CSP backdoor sets into base classes defined by structural properties, base classes defined by "hybrid" concepts like forbidden patterns [10], or CSP backdoor sets for instances with global constraints. Another promising direction of future research is to parameterize backdoor sets not by their size but by structural properties of the backdoor set and how it interacts with the rest of the instance, similar to the parameters that have been considered for modulators [16, 17]; we have outlined first results into this direction in Section 6. We hope that this survey stimulates further research on CSP backdoor sets.

References

1 Libor Barto and Marcin Kozik. Constraint satisfaction problems solvable by local consistency methods. *J. of the ACM*, 61(1):3:1–3:19, 2014.

2 Christian Bessiére, Clément Carbonnel, Emmanuel Hebrard, George Katsirelos, and Toby Walsh. Detecting and exploiting subproblem tractability. In Francesca Rossi, editor, *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*. IJCAI/AAAI, 2013.

3 Hans L. Bodlaender, Fedor V. Fomin, Daniel Lokshtanov, Eelko Penninkx, Saket Saurabh, and Dimitrios M. Thilikos. (Meta) kernelization. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2009)*, pages 629–638. IEEE Computer Soc., Los Alamitos, CA, 2009.

4 Andrei A. Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-element set. *J. of the ACM*, 53(1):66–120, 2006.

5 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.

6 Leizhen Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters*, 58(4):171–176, 1996.

7 Clément Carbonnel, Martin C. Cooper, and Emmanuel Hebrard. On backdoors to tractable constraint languages. In *Principles and Practice of Constraint Programming – 20th International Conference, CP 2014, Lyon, France, September 8-12, 2014. Proceedings*, volume 8656 of *Lecture Notes in Computer Science*, pages 224–239. Springer Verlag, 2014.

8 Clément Carbonnel and Martin C. Cooper. Tractability in constraint satisfaction problems: a survey. *Constraints*, pages 1–30, 2015. doi:10.1007/s10601-015-9198-6.

9 Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. Strong computational lower bounds via parameterized complexity. *J. of Computer and System Sciences*, 72(8):1346–1367, 2006.

10 David A. Cohen, Martin C. Cooper, Páidí Creed, Dániel Marx, and András Z. Salamon. The tractability of CSP classes defined by forbidden patterns. *J. Artif. Intell. Res.*, 45:47–78, 2012.

11 Nadia Creignou. A dichotomy theorem for maximum generalized satisfiability problems. *J. of Computer and System Sciences*, 51(3):511–522, 1995.

12 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 150–159. IEEE Computer Society, 2011. doi:10.1109/FOCS.2011.23.

13 R. G. Downey and M. R. Fellows. *Parameterized Complexity.* Monographs in Computer Science. Springer Verlag, New York, 1999.

14 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity.* Texts in Computer Science. Springer, 2013.

15 Wolfgang Dvorák, Sebastian Ordyniak, and Stefan Szeider. Augmenting tractable fragments of abstract argumentation. *Artificial Intelligence*, 186:157–173, 2012.

16 Eduard Eiben, Robert Ganian, and Stefan Szeider. Meta-kernelization using well-structured modulators. In Thore Husfeldt and Iyad A. Kanj, editors, *Parameterized and Exact Computation – 10th International Symposium, IPEC 2014, Patras, Greece, September 16-18, 2015. Revised Selected Papers*, volume 43 of *LIPIcs*, pages 114–126, 2015. doi:10.4230/LIPIcs.IPEC.2015.114.

17 Eduard Eiben, Robert Ganian, and Stefan Szeider. Solving problems on graphs of high rank-width. In *Algorithms and Data Structures Symposium (WADS 2015), August 5-7, 2015, University of Victoria, BC, Canada*, LNCS, pages 314–326. Springer Verlag, 2015. URL: http://arxiv.org/abs/1507.05463, doi:10.1007/978-3-319-21840-3_26.

18 Tomás Feder and Moshe Y. Vardi. Monotone monadic snp and constraint satisfaction. In S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal, editors, *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, pages 612–622. ACM, 1993.

19 Tomás Feder and Moshe Y. Vardi. The computational structure of monotone monadic snp and constraint satisfaction: A study through datalog and group theory. *SIAM J. Comput.*, 28(1):57–104, 1998.

20 Michael R. Fellows and Michael A. Langston. An analogue of the myhill-nerode theorem and its use in computing finite-basis characterizations (extended abstract). In *30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October – 1 November 1989*, pages 520–525. IEEE Computer Society, 1989.

21 Johannes Klaus Fichte and Stefan Szeider. Backdoors to normality for disjunctive logic programs. *ACM Trans. Comput. Log.*, 17(1), 2015. doi:10.1145/2818646.

22 Johannes Klaus Fichte and Stefan Szeider. Backdoors to tractable answer set programming. *Artificial Intelligence*, 220:64–103, March 2015. doi:10.1016/j.artint.2014.12.001.

23 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*, volume XIV of *Texts in Theoretical Computer Science. An EATCS Series.* Springer Verlag, Berlin, 2006.

24 Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, M. S. Ramanujan, and Saket Saurabh. Solving d-SAT via backdoors to small treewidth. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 630–641, 2015.

25 Robert Ganian, M. S. Ramanujan, and Stefan Szeider. Backdoors to valued constraint satisfaction. In *Proceedings of CP 2016, the 22nd International Conference on Principles and Practice of Constraint Programming Toulouse, France 5-9 September 2016*, Lecture Notes in Computer Science. Springer Verlag, 2016. to appear.

26 Robert Ganian, M. S. Ramanujan, and Stefan Szeider. Discovering archipelagos of tractability for constraint satisfaction and counting. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1670–1681, 2016. Full version to appear in the *ACM Transactions on Algorithms*. `doi:10.1137/1.9781611974331.ch114`.

27 Robert Ganian, M. S. Ramanujan, and Stefan Szeider. Combining treewidth and backdoors for CSP. In *Proceedings of STACS 2017, the 34th International Symposium on Theoretical Aspects of Computer Science*, 2017. To appear. URL: `https://arxiv.org/abs/1610.03298`.

28 Serge Gaspers, Neeldhara Misra, Sebastian Ordyniak, Stefan Szeider, and Stanislav Živný. Backdoors into heterogeneous classes of SAT and CSP. *J. of Computer and System Sciences*, 2015. To appear, a preliminary version is available from https://arxiv.org/abs/1509.05725.

29 Serge Gaspers and Stefan Szeider. Backdoors to satisfaction. In Hans L. Bodlaender, Rod Downey, Fedor V. Fomin, and Dániel Marx, editors, *The Multivariate Algorithmic Revolution and Beyond – Essays Dedicated to Michael R. Fellows on the Occasion of His 60th Birthday*, volume 7370 of *Lecture Notes in Computer Science*, pages 287–317. Springer Verlag, 2012.

30 Serge Gaspers and Stefan Szeider. Strong backdoors to bounded treewidth SAT. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 489–498. IEEE Computer Society, 2013.

31 Martin Grohe, Ken-ichi Kawarabayashi, Dániel Marx, and Paul Wollan. Finding topological subgraphs is fixed-parameter tractable. In Lance Fortnow and Salil P. Vadhan, editors, *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 479–488. ACM, 2011.

32 Pavol Hell and Jaroslav Nesetril. Colouring, constraint satisfaction, and complexity. *Computer Science Review*, 2(3):143–163, 2008.

33 Pawel M. Idziak, Petar Markovic, Ralph McKenzie, Matthew Valeriote, and Ross Willard. Tractability and learnability arising from algebras with few subpowers. *SIAM J. Comput.*, 39(7):3023–3037, 2010.

34 Peter Jeavons, David Cohen, and Marc Gyssens. Closure properties of constraints. *J. of the ACM*, 44(4):527–548, 1997.

35 Peter Jeavons, David A. Cohen, and Martin C. Cooper. Constraints, consistency and closure. *Artificial Intelligence*, 101(1-2):251–265, 1998.

36 Peter Jeavons, Andrei A. Krokhin, and Stanislav Živný. The complexity of valued constraint satisfaction. *Bulletin of the European Association for Theoretical Computer Science*, 113, 2014.

37 Phokion G. Kolaitis. Constraint satisfaction, databases, and logic. In *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, pages 1587–1595. Morgan Kaufmann, 2003.

38 Phokion G. Kolaitis and Moshe Y. Vardi. A logical approach to constraint satisfaction. In *Finite model theory and its applications*, Texts Theoret. Comput. Sci. EATCS Ser., pages 339–370. Springer Verlag, 2007.

39 Vladimir Kolmogorov and Stanislav Živný. The complexity of conservative valued CSPs. *J. of the ACM*, 60(2):Art. 10, 38, 2013.

40 Marcin Kozik, Andrei Krokhin, Matt Valeriote, and Ross Willard. Characterizations of several Maltsev Conditions. *Algebra Universalis*, 73(3-4):205–224, 2015.

41 Martin Kronegger, Sebastian Ordyniak, and Andreas Pfandler. Backdoors to planning. In Carla E. Brodley and Peter Stone, editors, *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27-31, 2014, Québec City, Québec, Canada.*, pages 2300–2307. AAAI Press, 2014.

42 Martin Kronegger, Sebastian Ordyniak, and Andreas Pfandler. Variable-deletion backdoors to planning. In Blai Bonet and Sven Koenig, editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 3305–3312. AAAI Press, 2015.

43 Daniel Lokshtanov, N. S. Narayanaswamy, Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. Faster parameterized algorithms using linear programming. *ACM Trans. Algorithms*, 11(2):15:1–15:31, 2014. `doi:10.1145/2566616`.

44 Daniel Lokshtanov and M. S. Ramanujan. Parameterized tractability of multiway cut with parity constraints. In *ICALP 2012, Automata, languages, and programming. Part I*, volume 7391 of *Lecture Notes in Computer Science*, pages 750–761. Springer Verlag, 2012.

45 Dániel Marx. Chordal deletion is fixed-parameter tractable. *Algorithmica*, 57(4):747–768, 2010.

46 Arne Meier, Sebastian Ordyniak, M. S. Ramanujan, and Irena Schindler. Backdoors for linear temporal logic. In *11th International Symposium on Parameterized and Exact Computation, IPEC 2016, August 24-26, 2016, Aarhus, Denmark*, Leibniz International Proceedings in Informatics (LIPIcs), pages 23:1–23:17. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2016. `doi:10.4230/LIPIcs.IPEC.2016.23`.

47 Ugo Montanari. Networks of constraints: fundamental properties and applications to picture processing. *Information Sciences*, 7:95–132, 1974.

48 Naomi Nishimura, Prabhakar Ragde, and Stefan Szeider. Detecting backdoor sets with respect to Horn and binary clauses. In *Proceedings of SAT 2004 (Seventh International Conference on Theory and Applications of Satisfiability Testing, 10–13 May, 2004, Vancouver, BC, Canada)*, pages 96–103, 2004.

49 Andreas Pfandler, Stefan Rümmele, and Stefan Szeider. Backdoors to abduction. In *Proceedings of IJCAI 2013, the 23th International Joint Conference on Artificial Intelligence, August 3-9, 2013, Beijing, China*, 2013.

50 Bruce A. Reed, Kaleigh Smith, and Adrian Vetta. Finding odd cycle transversals. *Oper. Res. Lett.*, 32(4):299–301, 2004. `doi:10.1016/j.orl.2003.10.009`.

51 Marko Samer and Stefan Szeider. Backdoor sets of quantified Boolean formulas. *Journal of Automated Reasoning*, 42(1):77–97, 2009. URL: `https://www.ac.tuwien.ac.at/files/pub/SamerSzeider09a.pdf`.

52 Thomas J. Schaefer. The complexity of satisfiability problems. In *Conference Record of the Tenth Annual ACM Symposium on Theory of Computing (San Diego, Calif., 1978)*, pages 216–226. ACM, 1978.

53 Johan Thapper and Stanislav Živný. The complexity of finite-valued CSPs. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 695–704. ACM, 2013.

54 Johan Thapper and Stanislav Živný. Necessary conditions for tractability of valued CSPs. *SIAM J. Discrete Math.*, 29(4):2361–2384, 2015.

55 Ryan Williams, Carla Gomes, and Bart Selman. Backdoors to typical case complexity. In Georg Gottlob and Toby Walsh, editors, *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, IJCAI 2003*, pages 1173–1178. Morgan Kaufmann, 2003.

56 Stanislav Živný. *The complexity of valued constraint satisfaction problems.* Cognitive Technologies. Springer Verlag, 2012.

On the Complexity of Holant Problems

Heng Guo[1] **and Pinyan Lu**[2]

1 **School of Mathematical Sciences, Queen Mary University of London, London, UK**
`h.guo@qmul.ac.uk`

2 **Institute for Theoretical Computer Science, School of Information Management and Engineering, Shanghai University of Finance and Economics, Shanghai, China**
`lu.pinyan@mail.shufe.edu.cn`

Abstract

In this article we survey recent developments on the complexity of Holant problems. We discuss three different aspects of Holant problems: the decision version, exact counting, and approximate counting.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, G.2.1 Combinatorics.

Keywords and phrases Computational complexity, Counting complexity, Dichotomy theorems, Approximate counting, Holant problems

Digital Object Identifier 10.4230/DFU.Vol7.15301.159

1 Introduction

Ladner's theorem [53] states that if P $\neq$ NP then there is an infinite hierarchy of intermediate problems that are not polynomial time interreducible. For certain restrictions of these classes, however, dichotomy theorems can be achieved. For NP a dichotomy theorem would state that any problem in the restricted subclass of NP is either in P or NP-complete (or both, in the eventuality that NP equals P.)

The restrictions for which dichotomy theorems are known can be framed in terms of local constraints, most importantly, Constraint Satisfaction Problems (CSP) [58, 28, 4, 5, 6, 33, 38, 27, 37], and Graph Homomorphism Problems [34, 42, 8]. Explicit dichotomy results, where available, manifest a total understanding of the class of computation in question, within polynomial time reduction, and modulo the collapse of the class.

In this article we survey dichotomies in a framework for characterizing local properties that is more general than those mentioned in the previous paragraph, namely the so-called Holant framework [18, 19]. A particular problem in this framework is characterized by a set of signatures as defined in the theory of Holographic Algorithms [65, 64]. The CSP framework can be viewed as a special case of the Holant framework in which equality relations of any arity are always assumed to be available in addition to the stated constraints. The extension from CSP to Holant problems enables us to express certain important problems such as graph matchings, which escape CSP [40] but are expressible in the Holant framework. Moreover, for the same constrain language, Holant problems contain potentially more structure than CSP. Indeed, in the Holant framework, new tractable cases emerge, the most notable among which is holographic algorithms [65].

The graph matching problem plays an important role in the studies of complexity theory, leading to many exciting developments. Here is an (incomplete) list of related complexity and algorithmic results:

- There is a polynomial time algorithm to decide if a given graph has a perfect matching or not. This is the remarkable blossom algorithm by Edmonds [35]. In fact, in that paper, Edmonds proposed the complexity class P as the class of tractable problems.
- Counting perfect matchings is #P-Complete. This is proved by Valiant [60], right after he defined the class #P [61]. This problem is interesting since it shows that counting version may be much harder than the decision version for the same problem.
- Counting Perfect Matchings for planar graph is polynomial time solvable. This is the famous FKT algorithm [48, 59, 49]. It also serves as the computational primitive for Holographic algorithms [65].
- There is a polynomial time algorithm to compute the parity of the number of (perfect) matchings. The algorithm utilizes the fact that the value of permanent and determinate is the same modular 2. Thus matching problems have an interesting complexity transition from P and $\oplus$P to #P.
- There is a *fully polynomial-time randomized approximation scheme* (FPRAS) for approximately counting matchings. This is one of the first canonical examples of approximate counting [45]. The known algorithm is randomized. Deterministic algorithm is known for bounded degree graphs but open for general graphs [1].
- There is a FPRAS for approximately counting perfect matchings for bipartite graphs. The same algorithm can be used to approximate the permanent of nonnegative matrixes [47]. However it is a long-standing open question to generalize this algorithm to arbitrary graphs (or to show the impossibility for such an algorithm to exist).

From this list, we can see that the graph matching problem often sits right at the boundary between tractability and intractability. In order to understand the boundary of polynomial-time computation through the lens of dichotomy theorems, it is intrinsically important to include matching problems into consideration. Hence the natural framework to express matching problems, namely Holant problems, are more desirable (and more challenging at the same time) to understand than the conventional CSP framework. In this survey, we summarize results for both decision version and counting version of Holant problems with a focus on counting problems, since there is a lot of great progress in the last several years.

The Holant framework is strongly influenced by the development of holographic algorithms and holographic reductions [65, 64, 16, 18]. Indeed, holographic reductions are developed and applied as one of the primary techniques, which has not been used in the study of counting CSP (#CSP) previously. One advantage of the Holant framework is its flexibility. The conventional #CSP can be viewed as a special sub-framework of Holant by assuming that all equality functions are freely available. It is natural to assume other freely available classes of functions, such as the set of unary functions. When all unary functions are present, the framework becomes similar to the "conservative" case of CSP. We emphasize that the Holant framework is more flexible as a priori it assumes less freely available functions than the CSP.

In this survey, we put our attention mainly on Holant problems that cannot be expressed in the CSP framework. We will assume some familiarity to CSP as well as basic and classical dichotomy theorems for the CSP framework. Hence we can focus on new and interesting phenomena which are unique for the Holant framework.

Organization of the Survey

In Section 2, we formally define the framework of Holant Problems and some other basic notations. Section 3 summarizes some results for the decision version of Holant problems. Section 4 is the main section. We carefully discuss complexity dichotomies for the (exact) counting version of the Holant framework. Section 5 surveys some approximate counting results.

2 Definitions and Background

A *signature grid* $\Omega = (G, \mathcal{F}, \pi)$ is a tuple, where $G = (V, E)$ is a graph, $\mathcal{F}$ is a set of functions, and π is a mapping from the vertex set V to $\mathcal{F}$. A function $f \in \mathcal{F}$ with arity k is a mapping $[q]^k \to \mathbb{C}$, and the mapping π satisfies that the arity of $\pi(v)$ (which is a function $f \in \mathcal{F}$) is the same as the degree of v for any $v \in V$. Here we may consider any function with the range of a ring rather than just $\mathbb{C}$, but we choose $\mathbb{C}$ in this survey for clarity. Let $f_v := \pi(v)$ be the function on v. An assignment σ of edges is a mapping $E \to [q]$. The weight of σ is the evaluation $\prod_{v \in V} f_v(\sigma\mid_{E(v)})$, where $E(v)$ denotes the set of incident edges of v. The (counting version of) Holant problem on the instance Ω is to compute the sum of weights of all assignments; namely,

$$\mathrm{Holant}_\Omega = \sum_{\sigma} \prod_{v \in V} f_v(\sigma\mid_{E(v)}). \tag{1}$$

We also write $\mathrm{Holant}(\Omega; \mathcal{F})$ when we want to emphasize the function set $\mathcal{F}$.

The term Holant was first coined by Valiant in [65] to denote an exponential sum of the above form. Cai, Xia and Lu first formally introduced this framework of counting problems in [18, 19]. We can view each function f_v as a truth table, and then we can represent it by a vector in $\mathbb{C}^{q^{d(v)}}$, or a tensor in $(\mathbb{C}^q)^{\otimes d(v)}$. The vector or the tensor is called the *signature* of a function. When we say "function", we put a slight emphasis on that it is a mapping. When we say "signature", we put a slight emphasis on that it is ready to go through linear transformations. However most of the time in this survey, we use the two terms "function" and "signature" interchangeably without special attention.

A Holant problem is parameterized by a set of functions.

▶ **Definition 1.** Given a set of functions $\mathcal{F}$, we define a counting problem $\mathrm{Holant}(\mathcal{F})$:
Input: A *signature grid* $\Omega = (G, \mathcal{F}, \pi)$;
Output: Holant_Ω.

We will use $\text{Pl-Holant}(\mathcal{F})$ to denote the problem where the input graph is planar.

The main goal here is to characterize what kind of function set $\mathcal{F}$ makes the problem $\mathrm{Holant}(\mathcal{F})$ tractable (or hard).

The main focus of this survey is for functions over the Boolean domain $\{0, 1\}$, which we call Boolean functions. We use the following notations to denote some special functions. Let $=_k$ denote the equality function of arity k. Let Δ_s denote the constant unary function which gives value 1 on inputs $s \in [q]$, and 0 on all other inputs. Let $\textsc{ExactOne}_k$ denote the function that is one if the input has Hamming weight 1 and zero otherwise. Let $\mathcal{EO}$ be the set of $\textsc{ExactOne}_k$ functions for all integers k. Then $\mathrm{Holant}(\mathcal{EO})$ is the same as the problem of counting perfect matchings.

A function is symmetric iff its function value is preserved under any permutation of its inputs. A symmetric function f on Boolean variables can be expressed by a compact signature $[f_0, f_1, \ldots, f_k]$, where f_i is the value of f on inputs of Hamming weight i. For the

Boolean domain $[2]=\{0,1\}$, $=_k$ function has the signature $[1,0,\dots,0,1]$ with $k+1$ entries and Δ_0 has $[1,0]$. Moreover, $\textsc{ExactOne}_k$ has signature $[0,1,0,\dots,0]$ of $k+1$ entries.

Multiplying a signature $f\in\mathcal{F}$ by a scaler $c\neq 0$ will not change the complexity of $\operatorname{Holant}(\mathcal{F})$. So we always view f and cf as the same signature. In other words, we consider the projective space of vectors or tensors.

Another important property of signatures is degeneracy.

▶ **Definition 2.** A signature is called degenerate iff it can be decomposed into a tensor product of unary signatures.

In particular, a symmetric signature over a Boolean domain is degenerate iff it can be expressed as $\lambda[x,y]^{\otimes k}$.

We use $\operatorname{Holant}(\mathcal{F}\mid\mathcal{G})$ to denote the Holant problem over signature grids with a bipartite graph $H=(U,V,E)$, where each vertex in U or V is assigned a signature in $\mathcal{F}$ or $\mathcal{G}$, respectively. Signatures in $\mathcal{F}$ are considered as row vectors (or covariant tensors); signatures in $\mathcal{G}$ are considered as column vectors (or contravariant tensors) (see, for example [30]). In this setting we sometimes write the Holant sum as $\operatorname{Holant}(\Omega;\mathcal{F}\mid\mathcal{G})$ for input Ω. Let $\operatorname{Pl-Holant}(\mathcal{F}\mid\mathcal{G})$ denote the Holant problem over signature grids with a planar bipartite graph.

2.1 Holographic Reductions

One key technique for Holant problems is holographic reductions. To introduce the idea, it is convenient to consider bipartite graphs. For a general graph, we can always transform it into a bipartite graph while preserving the Holant value as follows. For each edge in the graph, we replace it by a path of length two. (This operation is called the *2-stretch* of the graph and yields the edge-vertex incidence graph.) Each new vertex is assigned the binary $\textsc{Equality}$ signature $(=_2)=[1,0,1]$. Recall that $\operatorname{Holant}(\mathcal{F}\mid\mathcal{G})$ denotes the Holant problem over signature grids with a bipartite graph $H=(U,V,E)$, where each vertex in U or V is assigned a signature in $\mathcal{F}$ or $\mathcal{G}$, respectively. Hence we have that $\operatorname{Holant}(\mathcal{F})\equiv_T\operatorname{Holant}(=_2\mid\mathcal{F})$.

For a 2-by-2 matrix T and a signature set $\mathcal{F}$, define $T\mathcal{F}=\{g\mid\exists f\in\mathcal{F}$ of arity n, $g=T^{\otimes n}f\}$, and similarly for $\mathcal{F}T$. Whenever we write $T^{\otimes n}f$ or $T\mathcal{F}$, we view the signatures as column vectors; similarly for $fT^{\otimes n}$ or $\mathcal{F}T$ as row vectors. In the special case that $T=\left[\begin{smallmatrix}1&1\\1&-1\end{smallmatrix}\right]$, we use $\widehat{\mathcal{F}}$ to denote $T\mathcal{F}$.

Let T be an invertible 2-by-2 matrix. The holographic transformation defined by T is the following operation: given a signature grid $\Omega=(H,\pi)$ of $\operatorname{Holant}(\mathcal{F}\mid\mathcal{G})$, for the same bipartite graph H, we get a new grid $\Omega'=(H,\pi')$ of $\operatorname{Holant}\big(\mathcal{F}T\mid T^{-1}\mathcal{G}\big)$ by replacing each signature in $\mathcal{F}$ or $\mathcal{G}$ with the corresponding signature in $\mathcal{F}T$ or $T^{-1}\mathcal{G}$.

▶ **Theorem 3** (Valiant's Holant Theorem [65]). *If $T\in\mathbb{C}^{2\times 2}$ is an invertible matrix, then we have* $\operatorname{Holant}(\Omega;\mathcal{F}\mid\mathcal{G})=\operatorname{Holant}(\Omega';\mathcal{F}T\mid T^{-1}\mathcal{G})$.

Therefore, an invertible holographic transformation does not change the complexity of the Holant problem in the bipartite setting. Furthermore, there is a special kind of holographic transformation, the orthogonal transformation, that preserves the binary equality and thus can be used freely in the standard setting.

▶ **Theorem 4** (Theorem 2.6 in [20]). *If $T\in\mathbf{O}_2(\mathbb{C})$ is an orthogonal matrix (i.e. $TT^{T}=I_2$), then* $\operatorname{Holant}(\Omega;\mathcal{F})=\operatorname{Holant}(\Omega';T\mathcal{F})$.

We frequently apply a holographic transformation defined by the matrix $Z=\frac{1}{\sqrt{2}}\left[\begin{smallmatrix}1&1\\i&-i\end{smallmatrix}\right]$ (or sometimes without the nonzero factor of $\frac{1}{\sqrt{2}}$ since this does not affect the complexity). This matrix has the property that the binary EQUALITY signature $(=_2) = [1,0,1]$ is transformed to $[1,0,1]Z^{\otimes 2} = [0,1,0] = (\neq_2)$, the binary DISEQUALITY signature.

By Theorem 3, we have that

$$\begin{aligned} \text{Holant}(\mathcal{F}) &\equiv \text{Holant}\left([1,0,1]T^{\otimes 2} \mid T^{-1}\mathcal{F}\right) \\ \text{Pl-Holant}(\mathcal{F}) &\equiv \text{Pl-Holant}\left([1,0,1]T^{\otimes 2} \mid T^{-1}\mathcal{F}\right), \end{aligned}$$

where $T \in \mathbf{GL}_2(\mathbb{C})$ is nonsingular. This leads to the notion of $\mathcal{C}$-*transformable*.

▶ **Definition 5.** Let $\mathcal{F}$ and $\mathcal{C}$ be two sets of signatures. We say $\mathcal{F}$ is $\mathcal{C}$-transformable if there exists a $T \in \mathbf{GL}_2(\mathbb{C})$ such that $[1,0,1]T^{\otimes 2} \in \mathcal{C}$ and $\mathcal{F} \subseteq T\mathcal{C}$.

The following lemma is immediate.

▶ **Lemma 6.** *If $\mathcal{F}$ is $\mathcal{C}$-transformable, then we have the following reductions.*

$$\begin{aligned} \text{Holant}(\mathcal{F}) &\le_T \text{Holant}(\mathcal{C}); \\ \text{Pl-Holant}(\mathcal{F}) &\le_T \text{Pl-Holant}(\mathcal{C}). \end{aligned}$$

Clearly, if Holant$(\mathcal{C})$ or Pl-Holant$(\mathcal{C})$ is tractable, then Holant$(\mathcal{F})$ or Pl-Holant$(\mathcal{F})$ is tractable for any $\mathcal{C}$-transformable set $\mathcal{F}$.

2.2 Counting Constraint Satisfaction Problems

An instance of counting constraint satisfaction problems (#CSP$(\mathcal{F})$) has the following bipartite view. We have a set of vertices standing for variables and another set for functions (or constraints). Connect a variable vertex to a constraint vertex if the variable appears in the constraint. This bipartite graph is also known as the *constraint graph*. Moreover, each variable can be viewed as an EQUALITY function, as it forces the same value for all adjacent edges. Under this view, we see that

$$\#\text{CSP}(\mathcal{F}) \equiv_T \text{Holant}\left(\mathcal{EQ} \mid \mathcal{F}\right),$$

where $\mathcal{EQ} = \{=_1, =_2, =_3, \dots\}$ is the set of EQUALITY signatures of all arities.

The relationship between #CSP and Holant problems is the following:

$$\begin{aligned} \#\text{CSP}(\mathcal{F}) &\equiv_T \text{Holant}(\mathcal{EQ} \cup \mathcal{F}); \\ \text{Pl-}\#\text{CSP}(\mathcal{F}) &\equiv_T \text{Pl-Holant}(\mathcal{EQ} \cup \mathcal{F}). \end{aligned}$$

Reductions from left to right are trivial. For the other direction, we take a signature grid Ω for the problem on the right and create a bipartite signature grid Ω' for the problem on the left such that both signature grids have the same Holant value. We simply create the equivalent bipartite grid Ω'' of Ω by replace each edge with a path of length 2 with $=_2$ in the middle point, as described earlier. Then we contract all EQUALITY signatures that are connected with each other, resulting in Ω' where EQUALITY signatures are on one side and signatures from $\mathcal{F}$ on the other.

3 Decision Version

In the decision version, we focus on the functions taking values in $\{0,1\}$ and ask the question if the Holant value (as defined in (1)) is zero or not, or equivalently ask if there exists an assignment to satisfy all constraints or not. In this case, a function is a relation and it can also be viewed as a subset of all the possible assignments.

For the decision version of the CSP framework, it is a long standing open question to prove a dichotomy in general. But if we restrict to the Boolean domain, a classification is given by Schaefer [58] as one of the first computational complexity dichotomy theorems. The same dichotomy holds even if we restrict to the instances where each variable appears in at most three constraints. On the other hand, the decision version of Holant is equivalent to CSP where each variable appears at most twice. Its complexity classification, even for the Boolean domain, is still wide open and very interesting. To see why this is challenging, note that the perfect matching problem is a Holant problem defined by the $\textsc{ExactOne}_k$ function (with signature $[0, 1, 0 \ldots, 0]$). Deciding the existence of a perfect matching is polynomial time solvable due to Edmonds's remarkable blossom algorithm [36]. However Edmonds's algorithm is highly non-trivial. Indeed it is much more complicated than any of the tractable cases in Schaefer's dichotomy for the CSP framework, and utilizes a lot of special structures intrinsic to the problem. It is hard to rule out the possibility of other similar tractable problems. The following family of Δ-matroid relations turn out to be the main obstacle. Let e_i denote the unit vector which is 0 on all indices other than i, on which its entry is 1.

▶ **Definition 7.** Let M be a subset of $\{0,1\}^d$. It is called a Δ-*matroid* if for any pair of vectors $x, y \in M$ that differ on some index i, either $x \oplus e_i \in M$, or there exists another index $j \neq i$ on which x and y also differ and $x \oplus e_i \oplus e_j \in M$.

We say a relation R (or a $\{0,1\}$ valued function f) is a Δ-matroid if the set of allowed assignments of R (or the set of inputs x such that $f(x) = 1$) is a Δ-matroid.

It is easy to verify that the perfect matching function ($\textsc{ExactOne}_k$ function) is a Δ-matroid according to the definition, but there are many more functions that are Δ-matroids. In particular, it was shown that there are functions in this family which cannot be expressed by composition of $\textsc{ExactOne}_k$ functions. Feder showed the following hardness result: unless all relations in F are Δ-matroids, the decision Holant($\mathcal{F}$) has the same complexity as CSP($\mathcal{F}$) [39]. Based on this hardness result, we have the following classification result.

▶ **Theorem 8.** *All decision* Holant($\mathcal{F}$) *problems are divided into three classes according to $\mathcal{F}$:*

1. *Every function in $\mathcal{F}$ is a Δ-matroid;*
2. *If CSP($\mathcal{F}$) is tractable according to the dichotomy classification of Schaefer [58], then* Holant($\mathcal{F}$) *is also tractable;*
3. *Otherwise,* Holant($\mathcal{F}$)*is NP-complete.*

The only remaining open case is the complexity of Δ-matroid functions in the Holant framework. A number of tractable classes of Δ-matroids have been identified [26, 39, 29, 41, 50], but it seems to be still quite challenging to settle the complexity for the whole class. A recurring theme is the connection between Δ-matroids and matching problems. Here we mention two known tractable classes of Δ-matroids.

For a symmetric relation $[f_0, f_1, \ldots, f_k]$ where $f_i \in \{0,1\}$, the number of consecutive 0's between two 1's is called a gap. For example, $[0,1,0,0,1,1,0,0,0,1,0,0]$ have two gaps with length 2 and 3 respectively, while the one 0 in the beginning and two 0's at the end are not viewed as gaps. It is not difficulty to verify that, a symmetric relation is a Δ-matroid if and only if it has no gap with length larger than one. For all symmetric Δ-matroid relations, the decision Holant problem is tractable [26].

▶ **Theorem 9.** *If all the relations in $\mathcal{F}$ are symmetric and Δ-matroid, i.e. the largest gap of any relation in $\mathcal{F}$ has length* 1*, then there is a polynomial time algorithm to decide* Holant($\mathcal{F}$)*.*

Another broad family of tractable functions is called even Δ-matroid relations, shown recently by Kazda, Kolmogorov, and Rolínek [50]. A Δ-matroid relation M is called even if all vectors in M have the same parity of their Hamming weights; that is, they all have even Hamming weights or all have odd Hamming weights.

▶ **Theorem 10.** *If F contains only even Δ-matroid relations, then decision* Holant$(\mathcal{F})$ *can be solved in polynomial time.*

This tractability result for even Δ-matroid relations leads to a complete complexity dichotomy of Boolean CSP on planar graphs (see [31, 50]).

4 Exact Counting

There has been a lot of progress in understanding the complexity of computing the Holant sum exactly. In particular, we have a thorough understanding of Holant problems defined by Boolean symmetric functions, even if the input is restricted to planar graphs and the weights are complex.

The following theorem is a combination of [11, 9], the culminating results from a long line of research [23, 19, 21, 51, 52, 15, 14, 13, 44]. We are interested in general input graphs as well as planar graphs. The Holant framework was first proposed to systematically study the power of Valiant's holographic algorithms [65], which was designed to solve counting problems in planar graphs. When inputs are planar graphs, the problem is denoted by Pl-Holant$(\mathcal{F})$.

▶ **Theorem 11.** *Let $\mathcal{F}$ be a set of Boolean symmetric functions with complex weights.* Holant$(\mathcal{F})$ *either has a polynomial time algorithm, or is* $\#\mathbf{P}$*-hard to compute. This dichotomy also holds for* Pl-Holant$(\mathcal{F})$ *(but the tractable criterion is different).*

In fact, we know more than merely that the dichotomy holds. (This is non-trivial due to Ladner's Theorem [53]) We have a complete explicit criteria for tractable sets of functions in Theorem 11. In order to describe the criteria, we will first introduce some families of functions that appear as tractable cases in the dichotomy theorem.

4.1 Tractable Families

We summarize several known sets of tractable Boolean functions with complex weights. The first one is very simple. If all signatures are degenerate or binary, then the problem is tractable.

For a binary signature, define its matrix as

$$M_f := \begin{bmatrix} f(00) & f(01) \\ f(10) & f(11) \end{bmatrix}. \tag{2}$$

Connecting f to g via one edge gives another signature h with the matrix $M_h = M_f M_g$.

▶ **Lemma 12.** *Let $\mathcal{F}$ be a set of complex weighted symmetric signatures in Boolean variables. Then* Holant$(\mathcal{F})$ *is computable in polynomial time if all non-degenerate signatures in $\mathcal{F}$ are of arity at most 2.*

Proof. We first replace degenerate signatures by a bunch of equivalent unary signatures. Then any instance of Holant$(\mathcal{F})$ can be decomposed into paths and cycles. The Holant is a product of all paths and cycles.

For a path, we remove the two endpoints, leaving a binary signature f composed by a series of binary signatures. Compute the signature matrix M_f of f by multiplying all binary signatures along the path. Then the Holant is $vM_f u^{\mathtt{T}}$, where v and u are the two unary signatures at endpoints.

For a cycle, we arbitrarily break an edge getting a path with two dangling edges. Similar to the above case, we multiply matrices of all binary signatures along this path, getting M. The trace of M is the Holant. ◀

We further note that for a binary signature f and $T \in \mathbb{C}^{2\times 2}$, let $g = fT^{\otimes 2}$. Then

$$M_g = TM_fT^{\mathtt{T}}. \tag{3}$$

This can be seen by viewing T as a binary, and then treating g as connecting T, f, and $T^{\mathtt{T}}$ sequentially.

4.1.1 Affine Signatures

▶ **Definition 13** (Definition 3.1 in [25]). A k-ary function $f(x_1, \dots, x_k)$ is *affine* if it has the form

$$\lambda \cdot \chi_{Ax=0} \cdot i^{\sum_{j=1}^{n} \langle \mathbf{v}_j, x\rangle},$$

where $\lambda \in \mathbb{C}$, $x = (x_1, x_2, \dots, x_k, 1)^{\mathtt{T}}$, A is a matrix over $\mathbb{F}_2$, $\mathbf{v}_j$ is a vector over $\mathbb{F}_2$, and χ is a 0-1 indicator function such that $\chi_{Ax=0}$ is 1 if and only if $Ax = 0$. Note that the dot product $\langle \mathbf{v}_j, x\rangle$ is calculated over $\mathbb{F}_2$, while the summation $\sum_{j=1}^{n}$ on the exponent of $i = \sqrt{-1}$ is evaluated as a sum mod 4 of 0-1 terms. We use $\mathscr{A}$ to denote the set of all affine functions.

The matrix A defines an affine space which is the support of the signature f (and hence the name). Notice that there is no restriction on the number of rows in the matrix A. It is permissible that A is the zero matrix so that $\chi_{Ax=0} = 1$ holds for all x. An equivalent way to express the exponent of i is as a quadratic polynomial where all cross terms have an even coefficient (cf. [7]).

It is known that the set of non-degenerate symmetric signatures in $\mathscr{A}$ is precisely the nonzero signatures ($\lambda \neq 0$) in $\mathcal{F}_1 \cup \mathcal{F}_2 \cup \mathcal{F}_3$ with arity at least 2, where $\mathcal{F}_1$, $\mathcal{F}_2$, and $\mathcal{F}_3$ are three families of signatures defined as

$$\begin{aligned}
\mathcal{F}_1 &= \left\{\lambda\left([1,0]^{\otimes k} + i^r[0,1]^{\otimes k}\right) \mid \lambda \in \mathbb{C}, k = 1,2,\dots, r = 0,1,2,3\right\},\\
\mathcal{F}_2 &= \left\{\lambda\left([1,1]^{\otimes k} + i^r[1,-1]^{\otimes k}\right) \mid \lambda \in \mathbb{C}, k = 1,2,\dots, r = 0,1,2,3\right\},\\
\mathcal{F}_3 &= \left\{\lambda\left([1,i]^{\otimes k} + i^r[1,-i]^{\otimes k}\right) \mid \lambda \in \mathbb{C}, k = 1,2,\dots, r = 0,1,2,3\right\}.
\end{aligned} \tag{4}$$

We explicitly list these signatures up to an arbitrary constant multiple from $\mathbb{C}$, see Table 1.

The tractability of $\mathscr{A}$ is first shown in [25]. It was later generalized to arbitrary domain size using Gauss sums [7]. Together with Lemma 6, we have the following.

▶ **Lemma 14.** *Let $\mathcal{F}$ be any set of symmetric, complex-valued signatures in Boolean variables. If $\mathcal{F}$ is $\mathscr{A}$-transformable, then* Holant$(\mathcal{F})$ *is computable in polynomial time.*

4.1.2 Product-Type Signatures

▶ **Definition 15** (Definition 3.3 in [25]). A function is of *product type* if it can be expressed as a product of unary functions, binary equality functions ($[1,0,1]$), and binary disequality functions ($[0,1,0]$). We use $\mathscr{P}$ to denote the set of product-type functions.

Table 1 List of all non-degenerate affine signatures.

1.	$[1, 0, \dots, 0, \pm 1]$;	$(\mathscr{F}_1, r = 0, 2)$
2.	$[1, 0, \dots, 0, \pm i]$;	$(\mathscr{F}_1, r = 1, 3)$
3.	$[1, 0, 1, 0, \dots, 0 \text{ or } 1]$;	$(\mathscr{F}_2, r = 0)$
4.	$[1, -i, 1, -i, \dots, (-i) \text{ or } 1]$;	$(\mathscr{F}_2, r = 1)$
5.	$[0, 1, 0, 1, \dots, 0 \text{ or } 1]$;	$(\mathscr{F}_2, r = 2)$
6.	$[1, i, 1, i, \dots, i \text{ or } 1]$;	$(\mathscr{F}_2, r = 3)$
7.	$[1, 0, -1, 0, 1, 0, -1, 0, \dots, 0 \text{ or } 1 \text{ or } (-1)]$;	$(\mathscr{F}_3, r = 0)$
8.	$[1, 1, -1, -1, 1, 1, -1, -1, \dots, 1 \text{ or } (-1)]$;	$(\mathscr{F}_3, r = 1)$
9.	$[0, 1, 0, -1, 0, 1, 0, -1, \dots, 0 \text{ or } 1 \text{ or } (-1)]$;	$(\mathscr{F}_3, r = 2)$
10.	$[1, -1, -1, 1, 1, -1, -1, 1, \dots, 1 \text{ or } (-1)]$.	$(\mathscr{F}_3, r = 3)$

An alternate definition for $\mathscr{P}$, implicit in [22], is the tensor closure of signatures with support on two complementary bit vectors. It is easily shown (cf. Lemma A.1 in the full version of [44]) that if f is a symmetric signature in $\mathscr{P}$, then f is degenerate, binary DISEQUALITY $\neq_2$, or $[a, 0, \dots, 0, b]$ for some $a, b \in \mathbb{C}$.

The tractability of $\mathscr{P}$ is due to a straightforward propagation algorithm (see, for example [25]). Together with Lemma 6, we have the following.

▶ **Lemma 16.** *Let $\mathcal{F}$ be any set of symmetric, complex-valued signatures in Boolean variables. If $\mathcal{F}$ is $\mathscr{P}$-transformable, then* Holant$(\mathcal{F})$ *is computable in polynomial time.*

4.1.3 Vanishing Signatures

Vanishing signatures define Holant problems where the Holant sum is always 0.

▶ **Definition 17.** A set of signatures $\mathcal{F}$ is called *vanishing* if $\text{Holant}(\Omega; \mathcal{F}) = 0$ for every signature grid Ω. A signature f is called *vanishing* if the singleton set $\{f\}$ is vanishing.

A useful way to understand vanishing signatures is via a low rank tensor decomposition. To state these decompositions, we use the following definition.

▶ **Definition 18.** Let S_n be the symmetric group of degree n. Then for positive integers t and n with $t \le n$ and unary signatures $v, v_1, \dots, v_{n-t}$, we define

$$\text{Sym}_n^t(v; v_1, \dots, v_{n-t}) = \sum_{\pi \in S_n} \bigotimes_{k=1}^{n} u_{\pi(k)},$$

where the ordered sequence $(u_1, u_2, \dots, u_n) = (\underbrace{v, \dots, v}_{t \text{ copies}}, v_1, \dots, v_{n-t})$.

With this notation we can define the vanishing degree.

▶ **Definition 19.** A nonzero symmetric signature f of arity n has *positive vanishing degree* $k \ge 1$, denoted by $\text{vd}^+(f) = k$, if $k \le n$ is the largest positive integer such that there exists $n - k$ unary signatures $v_1, \dots, v_{n-k}$ such that

$$f = \text{Sym}_n^k([1, i]; v_1, \dots, v_{n-k}).$$

If f cannot be expressed as such a symmetrization form, we define $\text{vd}^+(f) = 0$. If f is the all zero signature, define $\text{vd}^+(f) = n + 1$.

We define *negative vanishing degree* vd^- similarly, using $-i$ instead of i.

It is possible that both $\mathrm{vd}^+(f)$ and $\mathrm{vd}^-(f)$ are nonzero. For example, $\mathrm{vd}^+(=_2) = \mathrm{vd}^-(=_2) = 1$.

The following theorem completely characterizes symmetric vanishing signatures. It is proved in [11]. For $\sigma \in \{+, -\}$, let $\mathcal{V}^\sigma := \{f \mid 2\,\mathrm{vd}^\sigma(f) > \mathrm{arity}(f)\}$.

▶ **Theorem 20.** *Let $\mathcal{F}$ be a set of symmetric signatures. Then $\mathcal{F}$ is vanishing if and only if $\mathcal{F} \subseteq \mathcal{V}^+$ or $\mathcal{F} \subseteq \mathcal{V}^-$.*

Obviously, vanishing signatures define tractable Holant problems. The algorithm is simple – just output 0! However, vanishing signatures can be combined with other functions and remain tractable. The following two lemma are shown in [11].

▶ **Lemma 21.** *Let $\sigma = +$ or $-$. Let $\mathcal{F}$ be a set of complex weighted symmetric signatures in Boolean variables. Then* $\mathrm{Holant}(\mathcal{F})$ *is computable in polynomial time if $\mathcal{F} \subseteq \mathcal{V}^\sigma \cup \{f \mid \mathrm{vd}^\sigma(f) \geq 1 \& \,\mathrm{arity}(f) = 2\}$.*

▶ **Lemma 22.** *Let $\sigma = +$ or $-$. Let $\mathcal{F}$ be a set of complex weighted symmetric signatures in Boolean variables. Then* $\mathrm{Holant}(\mathcal{F})$ *is computable in polynomial time if any non-degenerate signature $f \in \mathcal{F}$ satisfies that* $\mathrm{vd}^\sigma(f) \geq \mathrm{arity}(f) - 1$.

Lemma 21 can be understood as putting vanishing signatures and certain kind of binary signatures together remain tractable. Lemma 22 can be understood as highly vanishing signatures ($\mathrm{vd}^\sigma(f) \geq \mathrm{arity}(f) - 1$) can be put together with all unary signatures and remain tractable, since unary signatures automatically satisfy the condition $\mathrm{vd}^\sigma(f) \geq \mathrm{arity}(f) - 1 = 0$.

4.1.4 Matchgate Signatures

Matchgates were introduced by Valiant [63, 62] to give polynomial-time algorithms for a collection of counting problems over planar graphs. As the name suggests, problems expressible by matchgates can be reduced to computing a weighted sum of perfect matchings. The latter problem is tractable over planar graphs by Kasteleyn's algorithm [49]. Historically the algorithm was first found by Temperley and Fisher for $\mathbb{Z}^2$ [59] and independently by Kasteleyn [48]. It was later generalized to general planar graphs by Kastelyn [49]. Hence sometimes it is also called the FKT algorithm. These counting problems are naturally expressed in the Holant framework using *matchgate signatures*, denoted by $\mathcal{M}$. Thus Pl-Holant($\mathcal{M}$) is tractable.

Formally, recall that $\mathcal{EO}$ is the set of $\textsc{ExactOne}_k$ functions for all integers k. Let $\mathcal{WEO}$ be the set of weighted $\textsc{ExactOne}_k$ functions for all k. Then $\mathcal{M}$ contains signatures that can be realized as an $\mathcal{WEO}$-gate (realizable by functions in the set $\mathcal{WEO}$). Holographic transformations extend the reach of the FKT algorithm even further by Lemma 6, as stated below.

▶ **Lemma 23.** *Let $\mathcal{F}$ be any set of symmetric, complex-valued signatures in Boolean variables. If $\mathcal{F}$ is $\mathcal{M}$-transformable, then* Pl-Holant($\mathcal{F}$) *is computable in polynomial time.*

Matchgate signatures are characterized by the matchgate identities (for an up-to-date treatment, see [10] for the identities and a self-contained proof). Any matchgate signature f must satisfy the *parity condition*, which asserts that the support of f has to contain entries of only even or odd Hamming weights, but not both. For symmetric matchgates, they have 0 for every other entry and form a geometric progression with the remaining entries. We explicitly list all the symmetric signatures in $\mathcal{M}$ (see [10]).

▶ **Proposition 24.** *Let f be a symmetric signature in $\mathcal{M}$. Then there exists $a, b \in \mathbb{C}$ and $n \in \mathbb{N}$ such that f takes one of the following forms:*

1. $[a^n, 0, a^{n-1}b, 0, \ldots, 0, ab^{n-1}, 0, b^n]$ *(of arity* $2n \geq 2$*);*
2. $[a^n, 0, a^{n-1}b, 0, \ldots, 0, ab^{n-1}, 0, b^n, 0]$ *(of arity* $2n+1 \geq 1$*);*
3. $[0, a^n, 0, a^{n-1}b, 0, \ldots, 0, ab^{n-1}, 0, b^n]$ *(of arity* $2n+1 \geq 1$*);*
4. $[0, a^n, 0, a^{n-1}b, 0, \ldots, 0, ab^{n-1}, 0, b^n, 0]$ *(of arity* $2n+2 \geq 2$*).*

In the last three cases with $n = 0$, the signatures are $[1, 0]$, $[0, 1]$, and $[0, 1, 0]$. Any multiple of these is also a matchgate signature.

Note that perfect matching signatures, $[0, 1, 0, \cdots, 0]$, and their reversal are special cases when $b = 0$ or $a = 0$ in the last two cases.

Similar to vanishing signatures, signatures in $\mathcal{M}$ have low rank decompositions as well.

▶ **Proposition 25.** *Let f be a symmetric signature in $\mathcal{M}$ of arity n. Then there exist $a, b, \lambda \in \mathbb{C}$ such that f takes one of the following forms:*

1. $[a, b]^{\otimes n} + [a, -b]^{\otimes n} = \begin{cases} 2[a^n, 0, a^{n-2}b^2, 0, \ldots, 0, b^n] & n \text{ is even,} \\ 2[a^n, 0, a^{n-2}b^2, 0, \ldots, 0, ab^{n-1}, 0] & n \text{ is odd;} \end{cases}$
2. $[a, b]^{\otimes n} - [a, -b]^{\otimes n} = \begin{cases} 2[0, a^{n-1}b, 0, a^{n-3}b^3, 0, \ldots, 0, ab^{n-1}, 0] & n \text{ is even,} \\ 2[0, a^{n-1}b, 0, a^{n-3}b^3, 0, \ldots, 0, b^n] & n \text{ is odd;} \end{cases}$
3. $\lambda \operatorname{Sym}_n^{n-1}([1, 0]; [0, 1]) = [0, \lambda, 0, \ldots, 0]$*;*
4. $\lambda \operatorname{Sym}_n^{n-1}([0, 1]; [1, 0]) = [0, \ldots, 0, \lambda, 0]$.

The understanding of matchgates was further developed in [17], which characterized, for every symmetric signature, the set of holographic transformations under which the transformed signature becomes a matchgate signature.

4.1.5 An Extra Planar Tractable Case

In [9], towards a complete planar dichotomy theorem, a new tractable case was found for planar graphs.

Recall that $\mathcal{EO}$ is the set of functions $\textsc{ExactOne}_k$ for all arities k, and $Z = \left[\begin{smallmatrix} 1 & 1 \\ i & -i \end{smallmatrix}\right]$. Let $\mathcal{EO}'$ be the set of inverses of $\textsc{ExactOne}_k$ for all arities k. Namely, $f \in \mathcal{EO}'$ requires the input to have hamming weight exactly (arity -1). Let $\mathcal{WEQ}$ denotes the set of weighted equality functions. Moreover, let $\mathcal{F}^*$ denote $\mathcal{F}$ with all degenerate signatures $[a, b]^{\otimes m}$ replaced by unary $[a, b]$. Then we have the following lemma [9].

▶ **Lemma 26.** *Let $\mathcal{F}$ be a set of symmetric Boolean functions. If $\mathcal{F} \subseteq Z\mathcal{P} \cup Z(\mathcal{EO})$ or $\mathcal{F} \subseteq Z\mathcal{P} \cup Z(\mathcal{EO}')$, and the greatest common divisor of the arities of the signatures in $\mathcal{F}^* \cap Z(\mathcal{WEQ})$ is at least 5, then* $\operatorname{Holant}(\mathcal{F})$ *is tractable.*

The algorithm of this special case is a recursive procedure to find edges that either have to be a particular value, or do not have satisfying assignments. We can show that either these edges show up in the graph, or the instance falls into one of the tractable cases above. The existence of these edges (namely, when the instance is not solvable by previous cases) is due to the degree rigidity of a planar graph. (For example, the average degree of a planar graph cannot be more than 6.)

4.2 The Full Dichotomy

After introducing tractable families, we can finally state the dichotomy theorem in full detail.

▶ **Theorem 27.** *Let $\mathcal{F}$ be any set of symmetric, complex-valued functions in Boolean variables. Then* Pl-Holant($\mathcal{F}$) *is* #**P**-*hard unless $\mathcal{F}$ satisfies one of the following conditions:*

1. *All non-degenerate signatures in $\mathcal{F}$ are of arity at most 2;*
2. *$\mathcal{F}$ is $\mathscr{A}$-transformable;*
3. *$\mathcal{F}$ is $\mathscr{P}$-transformable;*
4. $\mathcal{F} \subseteq \mathscr{V}^\sigma \cup \{f \mid \mathrm{vd}^\sigma(f) \geq 1 \& \operatorname{arity}(f) = 2\}$ *for some* $\sigma \in \{+, -\}$*;*
5. *Any non-degenerate signature $f \in \mathcal{F}$ satisfies* $\operatorname{arity}(f) - \mathrm{vd}^\sigma(f) \leq 1$ *for some* $\sigma \in \{+, -\}$.
6. *$\mathcal{F}$ is $\mathscr{M}$-transformable;*
7. $\mathcal{F} \subseteq Z\mathscr{P} \cup Z(\mathcal{EO})$ *or* $\mathcal{F} \subseteq Z\mathscr{P} \cup Z(\mathcal{EO}')$*, and the greatest common divisor of the arities of the signatures in* $\mathcal{F}^* \cap Z(\mathcal{WEQ})$ *is at least* 5.

In each exceptional case, Pl-Holant($\mathcal{F}$) *is computable in polynomial time. If $\mathcal{F}$ satisfies conditions 1 to 5, then* Holant($\mathcal{F}$) *is computable in polynomial time without planarity; otherwise* Holant($\mathcal{F}$) *is* #**P**-*hard.*

4.3 Beyond Boolean and Symmetric Functions

In full generality, we would like to understand the complexity of Holant problems defined by any set of functions, rather than just symmetric Boolean functions. However, the understanding of those Holant problems is far from complete.

Still in the Boolean domain, the best dichotomy result we know of regarding asymmetric functions is [22]. A crucial constraint for the result of [22] is that it requires unary functions to be available freely. This corresponds to the "conservative" case in the study of CSP problems. We use Holant$^*(\mathcal{F})$ to denote these problems.

For asymmetric functions, we need to be careful to state the result. Let $\langle\mathcal{F}\rangle$ of a set $\mathcal{F}$ denote its tensor closure; namely, $\langle\mathcal{F}\rangle$ is the minimum set containing $\mathcal{F}$, closed under tensor product. This closure exists, being the set of all functions obtained by taking a finite sequence of tensor products from $\mathcal{F}$.

Let $\mathcal{T}$ be the set of all unary and binary functions. Let $\mathcal{E}$ be the set of all functions f such that f is zero except on two inputs $(x_1, \ldots, x_n)$ and $(1 - x_1, \ldots, 1 - x_n)$. In other words, $f \in \mathcal{E}$ iff its support is contained in a pair of complementary points. We think of $\mathcal{E}$ as a generalized form of equality functions. Let $\mathcal{M}$ be the set of all functions f such that f is zero except on $n + 1$ inputs whose Hamming weight is at most 1, where n is the arity of f. We think of $\mathcal{M}$ as a generalized form of matchings.

Recall that $Z = \left[\begin{smallmatrix} 1 & 1 \\ i & -i \end{smallmatrix}\right]$. Let $Z' = \left[\begin{smallmatrix} 1 & 1 \\ -i & i \end{smallmatrix}\right]$. Then we have the following theorem [22].

▶ **Theorem 28.** *Let $\mathcal{F}$ be any set of complex valued functions in Boolean variables. The problem* Holant$^*(\mathcal{F})$ *is polynomial time computable, if (1) $F \subseteq \langle\mathcal{T}\rangle$, or (2) there exists an orthogonal matrix H such that $\mathcal{F} \subseteq H$, or (3) $\mathcal{F} \subseteq \langle Z\mathcal{E}\rangle$ or $\mathcal{F} \subseteq \langle Z'\mathcal{E}\rangle$, or (4) $\mathcal{F} \subseteq \langle Z\mathcal{M}\rangle$ or $\mathcal{F} \subseteq \langle Z'\mathcal{E}\rangle$. In all other cases,* Holant$^*(\mathcal{F})$ *is* #**P**-*hard. The dichotomy is still true even if the inputs are restricted to planar graphs.*

Going beyond the Boolean domain, [24] gives a dichotomy theorem regarding Holant* problems defined by a single ternary symmetric function over a domain of size 3. The statement is rather technical and we refer the interested readers to [24] for details.

For even larger domain sizes, we know the complexity of counting k-edge-colourings over (planar) d-regular graphs for any pair of integers (k, d) [12]. Edge colourings are special cases of Holant problems where the domain size is k and the constraint on the vertex requires that all inputs are distinct. Following this path a dichotomy is known for Holant problems defined by ternary functions and with certain high symmetry [12]. This symmetry requirement is inspired by the All-Distinct constraint of edge colorings. For simplicity here we only state the edge coloring result and refer the reader to [12] for the more technical dichotomy theorem.

▶ **Theorem 29.** *Counting k-edge-colourings is $\#\mathbf{P}$-hard over planar d-regular (multi-)graphs if $k \geq d \geq 3$.*

Note that if $d \leq 2$ the problem is trivial, and if $k < d$ there is no such colourings.

5 Approximate Counting

In this last section, we study the approximation version of counting problems. For any given parameter $\epsilon > 0$, the algorithm outputs a number $\hat{Z}$ such that $(1-\epsilon)Z \leq \hat{Z} \leq (1+\epsilon)Z$, where Z is the accurate Holant summation of the input instance. We also require that the running time of the algorithm is bounded by $poly(n, 1/\epsilon)$, where n is the number of vertices of the given graph. This is called a fully polynomial-time approximation scheme (FPTAS). The randomized relaxation of FPTAS is called fully polynomial-time randomized approximation scheme (FPRAS), which uses random bits in the algorithm and requires that the final output is within the range $[(1-\epsilon)Z, (1+\epsilon)Z]$ with high probability.

Recall that we may view Holant problems as a CSP where each variable appears at most twice. The CSP problem with a degree bound is not necessarily of the same computational complexity as the problem without the degree bound even if the degree bound is larger than 2. New and interesting tractable families show up. For degree bounds larger than 2, a partial classification was known (see, for example [32]). On the other hand, the situation of Holant (bounded degree 2) is wide open, and it seems that there are many more tractable problems. Here we list a number of interesting ones.

Matching. There is an FPRAS for counting the number of matchings, even with weights [45].

Parity Function. A parity function is a symmetric function of form $[a, b, a, b, \cdots]$. If the constraint in each vertex is a parity function, there is an FPRAS for computing the partition function for any weighted graphs [46]. By transforming to this Holant problem (which was called the "subgraph world" problem in [46]) of parity functions, an FPRAS for ferromenaginic Ising model was given by Jerrum and Sinclair [46]

SAT. For SAT instances where each variable appears in at most two clauses, there is an FPRAS to count the number of satisfying assignments [3].

Not-All-Equal. Let $\textsc{NotAllEqual}_k$ be the symmetric function that is 0 if the input has Hamming weights 0 or k, and 1 otherwise; namely its signature is $[0, 1, 1, \cdots, 1, 0]$ with $k+1$ entries. Let $\mathcal{NAE}$ be the set of $\textsc{NotAllEqual}_k$ functions for all integers k. There is an FPRAS for Holant($\mathcal{NAE}$) [57].

5.1 Winding

One powerful approach to design approximate counting algorithms is Markov Chain Monte Carlo (MCMC). The key step is to prove that the Markov chain is rapidly mixing, namely, it is very close to the stationary distribution after polynomial number of steps. Canonical paths argument, introduced by [45, 46] is one of the two main tools (the other one is coupling) to prove rapid mixing of the Markov chain. To make use of canonical paths, one needs to design paths between each pair of states for the Markov chain and prove that the overall congestion at each transition of the Markov chain is low. However, it is typically a very difficult task to come up with a low congestion routing, especially because there are usually exponentially many states corresponding to the Markov chain.

There are some successful examples such as the matching problem mentioned above. The symmetric difference of two matchings of a graph is a disjoint union of paths and cycles. Then, the natural and successful canonical paths for matchings is "(un-)winding" the edges one by one following an arbitrary order of these paths and cycles. Another important successful example is the "subgraph world" problem (or the parity function problem, as described in the last section) transformed from ferromagnetic Ising model [46]. For this problem, the symmetric difference of two configurations can be any graphs. The key observation is that we may utilize the path-cycle decomposition. Jerrum and Sinclair's canonical paths simply do an arbitrary path-cycle decomposition and unwind edges following these paths and cycles. Since the constraint in each vertex for that problem is simply the parity function, one can prove that these canonical paths indeed have low congestion.

In an unpublished manuscript [57], McQuillan proposed a beautiful generalization of this path-cycle decomposition idea called winding. The idea was further developed in [43]. Here is the definition of windable functions.

▶ **Definition 30.** For any finite set J and any configuration $x \in \{0,1\}^J$, define $\mathcal{M}_x$ to be the set of partitions of $\{i \mid x_i = 1\}$ into pairs and at most one singleton. A function $f : \{0,1\}^J \to \mathbb{R}^+$ is **windable** if there exist values $B(x,y,M) \geq 0$ for all $x,y \in \{0,1\}^J$ and all $M \in \mathcal{M}_{x\oplus y}$ satisfying:
1. $f(x)f(y) = \sum_{M \in \mathcal{M}_{x\oplus y}} B(x,y,M)$ for all $x,y \in \{0,1\}^J$, and
2. $B(x,y,M) = B(x \oplus S, y \oplus S, M)$ for all $x,y \in \{0,1\}^J$ and all $S \in M \in \mathcal{M}_{x\oplus y}$.
Here $x \oplus S$ denotes the vector obtained by changing x_i to $1 - x_i$ for the one or two elements i in S.[1]

To get a rapidly mixing Markov chain, we assign two values to the two half-edges of each edge. We call it consistent if the two values are the same. A normal edge assignment is therefore an assignment of half-edges without inconsistency. We call these assignments perfect. A near-perfect assignment is one where there are two inconsistencies along edges. Windable functions admit a rapidly mixing Markov chain, by moving between perfect assignments and near-perfect assignments. Due to the design of the algorithm, the number of inconsistencies cannot be one.

However since we enlarge the state space slightly, merely a rapidly mixing Markov chain is not sufficient to guarantee a polynomial time algorithm. We also need to be able to hit perfect assignments with at least inverse polynomial probability. This can be stated as a bound between the Holant sum of all perfect assignments and that of all near-perfect assignments. More precisely, recall (1). The weight of an assignment can be naturally extend to near-perfect assignments, and their Holant sum is to simply add all weights up. Denote by Z_0 the Holant of all perfect assignments, and Z_2 that of all near-perfect assignments. Then we need to ensure that Z_2/Z_0 is bounded above by a polynomial for the MCMC algorithm to run in polynomial time.

▶ **Theorem 31.** *There exists an FPRAS to compute the partition function of* Holant$(\mathcal{F})$ *if all the functions in $\mathcal{F}$ are windable and Z_2/Z_0 is bounded above by a polynomial of the input size.*

One can verify that the matching constraint [45] and the parity function [46] are indeed windable. Thus both of these two FPRASes can be viewed as special cases of Theorem 31.

[1] This definition is taken from [43], which simplifies the original definition from [57].

However for a general function, it is still quite difficulty to tell whether it is windable or not. A clear characterization was given for all symmetric functions in [43] by solving a set of linear equations. With this powerful approach and characterization in hand, one can design a number of new FPRAS for approximate counting by simply verifying that the local constraint functions are windable. One such example is counting b-matchings, which is a natural generalization of matchings. A subset of edges for a graph is called a b-matching if every vertex is incident to at most b edges in the set. Hence 1-matching is the conventional definition of matching for a graph. Huang, Lu, and Zhang [43] showed that there exists an FPRAS to count b-matchings when $b \leq 7$ for any graphs.

Another problem one can resolve using this approach is a generalization of the edge cover problem. A subset of edges for a graph is called an edge cover if every vertex is incident to at least one edge in the set. Previously, MCMC based approximation algorithm for counting edge covers was only known for 3-regular graphs [2] by Bezáková and Rummler. In fact, they also used canonical paths to get rapid mixing and used path-cycle decompositions to construct canonical paths. However, due to the lack of a systematic approach, Bezáková and Rummler stopped at the special case of 3-regular graphs. Using the winding approach and the systematic characterization of windable functions, one can show that there exist a convex combination of path-cycle decompositions which works for general graphs [43]. Moreover, one can generalize it to b-edge-covers by requiring that every vertex is incident to at least b edges in the set. This approach yields an FPRAS to count b-edge-covers for $b \leq 2$ [43]. We note that FPTAS based on the correlation decay technique for counting edge covers for general graphs was known [54, 55]. However, it seems that the correlation decay approach have intrinsic difficulties for 2-edge-covers.

It is still open whether there exists an FPRAS for counting b-matchings for $b > 7$ or counting b-edge-covers for $b > 2$.

5.2 Fibonacci Functions

Correlation decay is another idea based on which one may design approximate counting algorithms. This approach has the advantage of yielding deterministic algorithms, namely FPTAS. Here we present FPTAS for a family of functions called Fibonacci Functions.

Fibonacci Functions by themselves are tractable, as they are $\mathcal{P}$-transformable (see Lemma 16). We extend the framework a bit by allowing edge weights. An edge-weighted Holant instance $\Omega = (G, \{f_v | v \in V\}, \{\lambda_e | e \in E\})$ is a tuple defined as follows. $G = (V, E)$ is a graph. f_v is a function with arity d_v: $\{0,1\}^{d_v} \to \mathbb{R}^+$, where d_v is the degree of v and $\mathbb{R}^+$ denotes non-negative real numbers. Edge weight λ_e is a mapping $\{0,1\} \to \mathbb{R}^+$. A configuration σ of edges is a mapping $E \to \{0,1\}$ and has a weight

$$w_\Omega(\sigma) = \prod_{e \in E} \lambda_e(\sigma(e)) \prod_{v \in V} f_v(\sigma \mid_{E(v)}),$$

where $E(v)$ denotes the set of incident edges of v. The counting problem on the instance Ω is to compute the partition function (or the Holant sum):

$$Z(\Omega) = \sum_\sigma \left(\prod_{e \in E} \lambda_e(\sigma(e)) \prod_{v \in V} f_v(\sigma \mid_{E(v)}) \right).$$

We use Holant$(\mathcal{F}, \Lambda)$ to denote the problem of computing the above quantity, where all functions are from $\mathcal{F}$ and all edge weights are from the set Λ. This version and its complexity are slightly different from the decision version and exactly counting as described in the previous two sections.

A Fibonacci function f is a symmetric function $[f_0, f_1, \ldots, f_k]$, satisfying that $f_i = cf_{i-1} + f_{i-2}$ for some constant c. For example, the parity function $[a, b, a, b, \ldots]$ is a special Fibonacci function with $c = 0$. If there is no edge weights (or equivalently all the weights are equal to 1) and all the node functions are Fibonacci functions with the same parameter c, we have a polynomial time algorithm to compute the partition function exactly [18]. If we allow edges to have non-trivial weights or different functions to have different parameters in Fibonacci gates, then the exact counting problem becomes #P-hard [19, 11]. Thus, it is interesting to study the problem in the approximation setting. Indeed, these edge-weighted Holant problems have connections with ferromagnetic 2-spin systems. For more details, see [56].

We use $\mathcal{F}_c^{p,q}$ to denote a subfamily of $\mathcal{F}_c$ such that $f_{i+1} \geq pf_i$ and $f_{i+1} \leq qf_i$ for all $i = 0, 1, \cdots, d-1$. When the upper bound q is not given, we simply write $\mathcal{F}_c^p$. We use $\mathcal{F}_{c_1,c_2}^{p,q}$ to denote $\bigcup_{c_1 \leq c \leq c_2} \mathcal{F}_c^{p,q}$. We use $\Lambda_{\lambda_1,\lambda_2}$ to denote the set of edge weights λ_e such that $\lambda_1 \leq \lambda_e \leq \lambda_2$.

Lu, Wang and Zhang [56] give the following algorithms.

▶ **Theorem 32.** *For any $c > 0$ and $p > 0$, there exists $\lambda_1(p, c) < 1$ and $\lambda_2(p, c) > 1$ such that there is an FPTAS for* $\text{Holant}(\mathcal{F}_c^p, \Lambda_{\lambda_1(p,c),\lambda_2(p,c)})$.

▶ **Theorem 33.** *Let $p > 0$. Then there is an FPTAS for* $\text{Holant}(\mathcal{F}_{1.17,+\infty}^p, \Lambda_{1,+\infty})$.

References

1 Mohsen Bayati, David Gamarnik, Dimitriy Katz, Chandra Nair, and Prasad Tetali. Simple deterministic approximation algorithms for counting matchings. In *Proceedings of STOC*, pages 122–127, 2007.

2 Ivona Bezáková and William A Rummler. Sampling edge covers in 3-regular graphs. In *Mathematical Foundations of Computer Science 2009*, pages 137–148. Springer, 2009.

3 Russ Bubley and Martin Dyer. Graph orientations with no sink and an approximation for a hard case of no. sat. Technical report, Association for Computing Machinery, New York, NY (United States), 1997.

4 Andrei A. Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-element set. *J. ACM*, 53(1):66–120, 2006. `doi:10.1145/1120582.1120584`.

5 Andrei A. Bulatov. The complexity of the counting constraint satisfaction problem. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP (1)*, volume 5125 of *Lecture Notes in Computer Science*, pages 646–661. Springer, 2008. `doi:10.1007/978-3-540-70575-8_53`.

6 Andrei A. Bulatov and Víctor Dalmau. Towards a dichotomy theorem for the counting constraint satisfaction problem. In *FOCS*, pages 562–571. IEEE Computer Society, 2003. URL: `http://csdl.computer.org/comp/proceedings/focs/2003/2040/00/20400562abs.htm`.

7 Jin-Yi Cai, Xi Chen, Richard J. Lipton, and Pinyan Lu. On tractable exponential sums. In *FAW*, pages 148–159. Springer Berlin Heidelberg, 2010.

8 Jin-Yi Cai, Xi Chen, and Pinyan Lu. Graph homomorphisms with complex values: A dichotomy theorem. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *ICALP (1)*, volume 6198 of *Lecture Notes in Computer Science*, pages 275–286. Springer, 2010. `doi:10.1007/978-3-642-14165-2_24`.

9 Jin-Yi Cai, Zhiguo Fu, Heng Guo, and Tyson Williams. A Holant dichotomy: Is the FKT algorithm universal? In *FOCS*, pages 249–260, 2015.

10 Jin-Yi Cai and Aaron Gorenstein. Matchgates revisited. *Theory Comput.*, 10(7):167–197, 2014.

11 Jin-Yi Cai, Heng Guo, and Tyson Williams. A complete dichotomy rises from the capture of vanishing signatures. In *STOC*, pages 249–260, 2013.

12 Jin-Yi Cai, Heng Guo, and Tyson Williams. The complexity of counting edge colorings and a dichotomy for some higher domain Holant problems. In *FOCS*, pages 601–610, 2014.

13 Jin-Yi Cai, Sangxia Huang, and Pinyan Lu. From Holant to #CSP and back: Dichotomy for Holant^c problems. *Algorithmica*, 64(3):511–533, 2012.

14 Jin-Yi Cai and Michael Kowalczyk. Spin systems on k-regular graphs with complex edge functions. *Theor. Comput. Sci.*, 461:2–16, 2012.

15 Jin-Yi Cai and Michael Kowalczyk. Partition functions on k-regular graphs with $\{0,1\}$-vertex assignments and real edge functions. *Theor. Comput. Sci.*, 494(0):63–74, 2013.

16 Jin-Yi Cai and Pinyan Lu. Holographic algorithms: From art to science. *J. Comput. Syst. Sci.*, 77(1):41–61, 2011. `doi:10.1016/j.jcss.2010.06.005`.

17 Jin-Yi Cai and Pinyan Lu. Holographic algorithms: From art to science. *J. Comput. Syst. Sci.*, 77(1):41–61, 2011.

18 Jin-Yi Cai, Pinyan Lu, and Mingji Xia. Holographic algorithms by fibonacci gates and holographic reductions for hardness. In *FOCS'08: Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science*, Washington, DC, USA, 2008. IEEE Computer Society.

19 Jin-Yi Cai, Pinyan Lu, and Mingji Xia. Holant problems and counting CSP. In Michael Mitzenmacher, editor, *STOC*, pages 715–724. ACM, 2009. `doi:10.1145/1536414.1536511`.

20 Jin-Yi Cai, Pinyan Lu, and Mingji Xia. Computational complexity of Holant problems. *SIAM J. Comput.*, 40(4):1101–1132, 2011.

21 Jin-Yi Cai, Pinyan Lu, and Mingji Xia. A computational proof of complexity of some restricted counting problems. *Theor. Comput. Sci.*, 412(23):2468–2485, 2011.

22 Jin-Yi Cai, Pinyan Lu, and Mingji Xia. Dichotomy for Holant* problems of boolean domain. In *SODA'11: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, 2011.

23 Jin-Yi Cai, Pinyan Lu, and Mingji Xia. Holographic reduction, interpolation and hardness. *Computational Complexity*, 21(4):573–604, 2012.

24 Jin-Yi Cai, Pinyan Lu, and Mingji Xia. Dichotomy for Holant* problems with domain size 3. In *SODA*, pages 1278–1295, 2013.

25 Jin-Yi Cai, Pinyan Lu, and Mingji Xia. The complexity of complex weighted Boolean #CSP. *J. Comput. System Sci.*, 80(1):217–236, 2014.

26 Gérard Cornuéjols. General factors of graphs. *Journal of Combinatorial Theory, Series B*, 45(2):185–198, 1988.

27 N. Creignou, S. Khanna, and M. Sudan. *Complexity classifications of boolean constraint satisfaction problems.* SIAM Monographs on Discrete Mathematics and Applications, 2001.

28 Nadia Creignou and Miki Hermann. Complexity of generalized satisfiability counting problems. *Inf. Comput.*, 125(1):1–12, 1996.

29 Victor Dalmau and Daniel K Ford. Generalized satisfiability with limited occurrences per variable: A study through delta-matroid parity. In *International Symposium on Mathematical Foundations of Computer Science*, pages 358–367. Springer, 2003.

30 C. T. J. Dodson and T. Poston. *Tensor Geometry.* Graduate Texts in Mathematics 130. Springer-Verlag, New York, 1991.

31 Zdeněk Dvořák and Martin Kupec. On planar boolean csp. In *International Colloquium on Automata, Languages, and Programming*, pages 432–443. Springer, 2015.

32 Martin E. Dyer, Leslie Ann Goldberg, Markus Jalsenius, and David Richerby. The complexity of approximating bounded-degree Boolean #CSP. *Inf. Comput.*, 220:1–14, 2012. `doi:10.1016/j.ic.2011.12.007`.

33 Martin E. Dyer, Leslie Ann Goldberg, and Mark Jerrum. The complexity of weighted boolean #CSP. *CoRR*, abs/0704.3683, 2007. URL: `http://arxiv.org/abs/0704.3683`.

34 Martin E. Dyer, Leslie Ann Goldberg, and Mike Paterson. On counting homomorphisms to directed acyclic graphs. *J. ACM*, 54(6), 2007. `doi:10.1145/1314690.1314691`.

35 Jack Edmonds. Maximum matching and a polyhedron with $0,1$ vertices. *J. of Res. the Nat. Bureau of Standards*, 69 B:125–130, 1965.

36 Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of mathematics*, 17(3):449–467, 1965.

37 John Faben. The complexity of counting solutions to generalised satisfiability problems modulo k. *CoRR*, abs/0809.1836, 2008.

38 T. Feder and M. 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, 1999.

39 Tomás Feder. Fanout limitations on constraint systems. *Theoretical Computer Science*, 255(1):281–293, 2001.

40 M. Freedman, L. Lovász, and A. Schrijver. Reflection positivity, rank connectivity, and homomorphism of graphs. *J. AMS*, 20:37–51, 2007.

41 James F Geelen, Satoru Iwata, and Kazuo Murota. The linear delta-matroid parity problem. *Journal of Combinatorial Theory, Series B*, 88(2):377–398, 2003.

42 Leslie Ann Goldberg, Martin Grohe, Mark Jerrum, and Marc Thurley. A complexity dichotomy for partition functions with mixed signs. *SIAM J. Comput.*, 39(7):3336–3402, 2010. `doi:10.1137/090757496`.

43 Lingxiao Huang, Pinyan Lu, and Chihao Zhang. Canonical paths for mcmc: from art to science. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 514–527. SIAM, 2016.

44 Sangxia Huang and Pinyan Lu. A dichotomy for real weighted Holant problems. In *IEEE Conference on Computational Complexity*, pages 96–106. IEEE Computer Society, 2012.

45 Mark Jerrum and Alistair Sinclair. Approximating the permanent. *SIAM J. Comput.*, 18(6):1149–1178, 1989. `doi:10.1137/0218077`.

46 Mark Jerrum and Alistair Sinclair. Polynomial-time approximation algorithms for the ising model. *SIAM Journal on Computing*, 22(5):1087–1116, 1993.

47 Mark Jerrum, Alistair Sinclair, and Eric Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *Journal of the ACM*, 51:671–697, July 2004. `doi:10.1145/1008731.1008738`.

48 P. W. Kasteleyn. The statistics of dimers on a lattice. *Physica*, 27:1209–1225, 1961.

49 P. W. Kasteleyn. Graph theory and crystal physics. In (F. Harary, editor, *Graph Theory and Theoretical Physics*, pages 43–110. Academic Press, London, 1967.

50 Alexandr Kazda, Vladimir Kolmogorov, and Michal Rolínek. Even delta-matroids and the complexity of planar boolean CSPs. *arXiv preprint arXiv:1602.03124*, 2016.

51 Michael Kowalczyk. Classification of a class of counting problems using holographic reductions. In Hung Q. Ngo, editor, *COCOON*, volume 5609 of *Lecture Notes in Computer Science*, pages 472–485. Springer, 2009. `doi:10.1007/978-3-642-02882-3_47`.

52 Michael Kowalczyk and Jin-Yi Cai. Holant problems for regular graphs with complex edge functions. *In the proceeding of STACS*, 2010.

53 Richard E. Ladner. On the structure of polynomial time reducibility. *J. ACM*, 22(1):155–171, 1975.

54 Chengyu Lin, Jingcheng Liu, and Pinyan Lu. A simple FPTAS for counting edge covers. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 341–348, 2014. `doi:10.1137/1.9781611973402.25`.

55 Jingcheng Liu, Pinyan Lu, and Chihao Zhang. FPTAS for counting weighted edge covers. In *Algorithms – ESA 2014 – 22th Annual European Symposium, Wroclaw, Poland, September 8-10, 2014. Proceedings*, pages 654–665, 2014.

56 Pinyan Lu, Menghui Wang, and Chihao Zhang. FPTAS for weighted Fibonacci gates and its applications. In *ICALP*, pages 787–799, 2014.

57 Colin McQuillan. Approximating holant problems by winding. *arXiv preprint arXiv:1301.2880*, 2013.

58 T. J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, page 226. ACM, 1978.

59 H. N. V. Temperley and M. E. Fisher. Dimer problem in statistical mechanics-an exact result. *Philosophical Magazine*, 6:1061–1063, 1961.

60 Leslie G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979.

61 Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM J. Comput.*, 8(3):410–421, 1979.

62 Leslie G. Valiant. Expressiveness of matchgates. *Theor. Comput. Sci.*, 289(1):457–471, 2002.

63 Leslie G. Valiant. Quantum circuits that can be simulated classically in polynomial time. *SIAM J. Comput.*, 31(4):1229–1254, 2002. URL: `http://epubs.siam.org/sam-bin/dbq/article/37702`.

64 Leslie G. Valiant. Accidental algorthims. In *FOCS'06: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 509–517, Washington, DC, USA, 2006. IEEE Computer Society. `doi:10.1109/FOCS.2006.7`.

65 Leslie G. Valiant. Holographic algorithms. *SIAM J. Comput.*, 37(5):1565–1594, 2008. `doi:10.1137/070682575`.

Parameterized Constraint Satisfaction Problems: a Survey

Gregory Gutin[*1] **and Anders Yeo**[2]

1 Royal Holloway, University of London, Egham, Surrey, UK; and University of Haifa, Mount Carmel, Haifa, Israel
gutin@cs.rhul.ac.uk

2 Singapore University of Technology and Design, Singapore; and University of Johannesburg, Auckland Park, South Africa
anders.yeo.work@gmail.com

Abstract

We consider constraint satisfaction problems parameterized above or below guaranteed values. One example is MaxSat parameterized above $m/2$: given a CNF formula F with m clauses, decide whether there is a truth assignment that satisfies at least $m/2 + k$ clauses, where k is the parameter. Among other problems we deal with are MaxLin2-AA (given a system of linear equations over $\mathbb{F}_2$ in which each equation has a positive integral weight, decide whether there is an assignment to the variables that satisfies equations of total weight at least $W/2+k$, where W is the total weight of all equations), Max-r-Lin2-AA (the same as MaxLin2-AA, but each equation has at most r variables, where r is a constant) and Max-r-Sat-AA (given a CNF formula F with m clauses in which each clause has at most r literals, decide whether there is a truth assignment satisfying at least $\sum_{i=1}^{m}(1 - 2^{r_i}) + k$ clauses, where k is the parameter, r_i is the number of literals in clause i, and r is a constant). We also consider Max-r-CSP-AA, a natural generalization of both Max-r-Lin2-AA and Max-r-Sat-AA, order (or, permutation) constraint satisfaction problems parameterized above the average value and some other problems related to MaxSat. We discuss results, both polynomial kernels and parameterized algorithms, obtained for the problems mainly in the last few years as well as some open questions.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Constraint satisfaction problems, Fixed-parameter tractability

Digital Object Identifier 10.4230/DFU.Vol7.15301.179

1 Introduction

While the main body of papers in the area of parameterized algorithms and complexity deals with problems on graphs and hypergraphs, in this paper we will consider parameterized constraint satisfaction problems (CSPs). This article is an update of survey paper [27] on the topic. We provide basic terminology and notation on parameterized algorithms and complexity in Section 2.

To the best of our knowledge, the first study of parameterized CSPs was almost twenty years ago by Cai and Chen [7] on standard parameterization of MaxSat. In MaxSat, we are given a CNF formula F with m clauses and asked to determine the maximum number of clauses of F that can be satisfied simultaneously by a truth assignment. In the standard parametrization of MaxSat, denoted by k-MaxSat, we are to decide whether there is a

[*] G. Gutin's research was partially supported by Royal Society Wolfson Research Merit Award.

truth assignment which satisfies at least k clauses of F, where k is the parameter. However, in the next paper on the topic Mahajan and Raman [41] already observed that the standard parameterization of MaxSat is not in the spirit of parameterized complexity. Indeed, it is well-known (and shown below, in Section 6) that there exists a truth assignment to the variables of F which satisfies at least $m/2$ clauses. Thus, for $k \leq m/2$ every instance of k-MaxSat is positive and thus only for $k > m/2$ the problem is of any interest. However, then the parameter k is quite large "in which range the fixed-parameter tractable algorithms are infeasible" [41].

Also it is easy to see that k-MaxSat has a kernel with a linear number of clauses. Indeed, consider an instance I of k-MaxSat. As we mentioned above, if $k \leq m/2$ then I is a positive instance. Otherwise, we have $k > m/2$ and $m \leq 2k - 1$. Such a kernel should be viewed as *large* rather than *small* as the bound $2k - 1$ might suggest at the first glance.

The bound $m/2$ is tight as we can satisfy only half clauses in the instances consisting of pairs $(x), (\bar{x})$ of clauses. This suggest the following parameterization of MaxSat *above tight bound* introduced by Mahajan and Raman [41]: decide whether there is a truth assignment which satisfies at least $m/2 + k$ clauses of F, where k is the parameter.

To the best our knowledge, [41] was the first paper on problems parameterized above or below tight bounds. Since then a large number of papers have appeared on the topic, some on graph and hypergraph problems and others on CSPs. In this survey paper, we will overview results on CSPs parameterized above or below tight bounds, as well as some methods used to obtain these results. Since some graph problems can also be viewed as those on CSPs, we will mention some results initially proved for graphs. While not going into details of the proofs, we will discuss some ideas behind the proofs. We will also consider some open problems in the area.

In the remainder of this section we give an overview of the paper and its organization.

In the next section we provide basics on parameterized algorithms and complexity. In Section 3, we describe the Strictly-Above-Below-Expectation Method (SABEM) introduced in [26]. The method uses some tools from Probabilistic Method and Harmonic Analysis. A relatively simple example illustrates the method.

Another example for SABEM is given in Section 4, which is devoted to the Maximum r-CSPs parameterized above the average value, where r is a positive integral constant. In general, the Maximum r-CSP is given by a set V of n Boolean variables and a set of m Boolean formulas; each formula is assigned an integral positive weight and contains at most r variables from V. The aim is to find a truth assignment which maximizes the weight of satisfied formulas. Averaging over all truth assignments, we can find the average value A of the weight of satisfied formulas. It is easy to show that we can always find a truth assignment to the variables of V which satisfied formulas of total weight at least A. Thus, a natural parameterized problem is whether there exists a truth assignment that satisfies formulas of total weight at least $A + k$, where k is the parameter (k is a nonnegative integer). We denote such a problem by Max-r-CSP-AA.

In Subsection 4.1, we consider the Max-r-Lin2-AA problem, which is a special case of Max-r-CSP-AA when every formula is a linear equation over $\mathbb{F}_2$ with at most r variables. For Max-r-Lin2-AA, we have $A = W/2$, where W is the total weight of all equations. It is well-known that, in polynomial time, we can find an assignment to the variables that satisfies equations of total weight at least $W/2$, but, for any $\epsilon > 0$ it is NP-hard to decide whether there is an assignment satisfying equations of total weight at least $W(1 + \epsilon)/2$ [31]. We will prove a result by Gutin, Kim, Szeider and Yeo [26] that Max-r-Lin2-AA has a kernel of quadratic size. We will mention some other results, in particular, a result of Crowston et al. [11] that Max-r-Lin2-AA has a kernel with at most $(2k - 1)r$ variables.

In Subsection 4.2, we give a proof scheme of a result by Alon et al. [2] that Max-r-CSP-AA has a kernel of polynomial size. The main idea of the proof is to reduce Max-r-CSP-AA to Max-r-Lin2-AA and use results on Max-r-Lin2-AA and a lemma on bikernels given in the next section. The result of Alon et al. [2] solves an open question of Mahajan, Raman and Sikdar [42] not only for Max-r-Sat-AA but for the more general problem Max-r-CSP-AA. The problem Max-r-Sat-AA is a special case of Max-r-CSP-AA when every formula is a clause with at most r variables. For Max-r-Sat-AA, the reduction to Max-r-Lin2-AA can be complemented by a reduction from Max-r-Lin2-AA back to Max-r-Sat-AA, which yields a kernel of quadratic size.

(Note that while the size of the kernel for Max-r-CSP-AA is polynomial, any bound on the degree of the polynomial is unknown so far.)

Section 5 is devoted to two parameterizations of MaxLin2. The first is MaxLin2-AA, which is the same problem as Max-r-Lin2-AA, but the number of variables in an equation is not bounded. Thus, MaxLin2-AA is a generalization of Max-r-Lin2-AA. We present a scheme of a proof by Crowston et al. [11] that MaxLin2-AA is fixed-parameter tractable (FPT) and has a kernel with polynomial number of variables. This result solved an open question of Mahajan et al. [42] of whether MaxLin2-AA is FPT, but we still do not know whether MaxLin2-AA has a kernel of polynomial size and we present only partial results on the topic. The second parameterization of MaxLin2 is as follows. Let W be the total weight of all equations in MaxLin2. We are to decide whether there is an assignment satisfying equations of total weight at least $W - k$, where k is a nonnegative parameter. This problem was proved to be W[1]-hard by Crowston et al. [14]. Following [14] we will discuss special cases of this problem giving its classification into fixed-parameter tractable and W[1]-hard cases.

In Section 6 we consider several parameterizations of Max-Sat different from Max-r-Sat-AA. Subsection 6.1 is devoted to MaxSat-A($m/2$), where given a CNF formula F with m clauses, we are to decide whether there is a truth assignment with satisfies at least $m/2 + k$ clauses of F, where k is the parameter.

In Subsection 6.2 we consider Max-$r(n)$-Sat-AA, which is the same problem as Max-r-Sat-AA, but $r(n)$ now depends on n. We discuss bounds on $r(n)$, which make Max-$r(n)$-Sat-AA either fixed-parameter tractable or not fixed-parameter tractable under the assumption that the Exponential Time Hypothesis (ETH) holds (we introduce ETH in the next section).

Results on MaxSat parameterized above or below various other tight bounds are discussed in Subsection 6.3. We will consider the above-mentioned parameterization of MaxSat above $m/2$ and some "stronger" parameterizations of MaxSat introduced or inspired by Mahajan and Raman [41]. The stronger parameterizations are based on the notion of a t-satisfiable CNF formula (a formula in which each set of t clauses can be satisfied by a truth assignment) and asymptotically tight lower bounds on the maximum number of clauses of a t-satisfiable CNF formula satisfied by a truth assignment for $t = 2$ and 3. We will describe linear-variable kernels obtained for both $t = 2$ and 3. We will also consider the parameterization of 2-Sat below the upper bound m, the number of clauses. This problem was proved to be fixed-parameter tractable by Razgon and O'Sullivan [48]. Raman et al. [47] and Cygan et al. [17] designed faster parameterized algorithms. for the problem.

In Section 7 we discuss parameterizations above average for Ordering CSPs. An Ordering CSP of arity r is defined by a set $V = \{x_1, \dots, x_n\}$ of variables and a set of constraints. Each constraint is a disjunction of clauses of the form $x_{i_1} < x_{i_2} < \cdots < x_{i_r}$. A linear ordering α of V satisfies such a constraint if one of the clauses in the disjunction agrees with α. The objective of the problem is to find an ordering of V which satisfies the maximum number of constraints. For the only nontrivial Ordering CSP of arity 2, 2-Linear Ordering, Guruswami, Manokaran and Raghavendra [22] proved that it is impossible to find, in polynomial time, an ordering that satisfies at least $|C|(1+\epsilon)/2$ constraints for every $\epsilon > 0$ provided the Unique Games Conjecture (UGC) of Khot [35] holds. (Note that $|C|/2$ is the expected number of constraints satisfied by a random uniformly-distributed ordering of V.) Similar approximation resistant results were proved for all Ordering CSPs of arity 3 by Charikar, Guruswami and Manokaran [8] and for Ordering CSPs of any arity by Guruswami et al. [21].

Thus it makes sense to consider Ordering CSPs parameterized above average. It was proved by Gutin, Kim, Szeider and Yeo [26] that 2-Linear Ordering parameterized above average is fixed-parameter tractable. Gutin, Iersel, Mnich and Yeo [23] showed that all Ordering CSPs of arity 3 parameterized above average are fixed-parameter tractable and conjectured the same results for every arity $r \geq 2$. Recently, Makarychev, Makarychev and Zhou [43] proved the conjecture. All the results can be proved using SABEM. This is already illustrated in Subsection 4.1 for 2-Linear Ordering parameterized above average. In Section 7, we provide a proof scheme for Betweenness parameterized above average by Gutin, Kim, Szeider and Yeo [25] who solved an open question of Benny Chor stated in Niedermeier's monograph [44]. This scheme was used also by Makarychev, Makarychev and Zhou [43], but their proof involves significantly more involved tools from Harmonic Analysis and we only provide some general remarks on their proof. We will also briefly discuss an interesting generalization of the result of Makarychev, Makarychev and Zhou [43].

We complete the paper with Section 8, where we briefly discuss two open problems.

2 Basics on Parameterized Algorithms and Complexity

A parameterized problem Π can be considered as a set of pairs (I, k) where I is the *problem instance* and k (usually a nonnegative integer) is the *parameter*. Π is called *fixed-parameter tractable (FPT)* if membership of (I, k) in Π can be decided by an algorithm of runtime $O(f(k)|I|^c)$, where $|I|$ is the size of I, $f(k)$ is an arbitrary function of the parameter k only, and c is a constant independent from k and I. Such an algorithm is called an *FPT* algorithm. Let Π and Π' be parameterized problems with parameters k and k', respectively. An *FPT-reduction R from* Π *to* Π' is a many-to-one transformation from Π to Π', such that (i) $(I, k) \in \Pi$ if and only if $(I', k') \in \Pi'$ with $k' \leq g(k)$ for a fixed computable function g, and (ii) R is of complexity $O(f(k)|I|^c)$.

If the nonparameterized version of Π (where k is just part of the input) is NP-hard, then the function $f(k)$ must be superpolynomial provided P$\neq$NP. Often $f(k)$ is "moderately exponential," which makes the problem practically feasible for small values of k. Thus, it is important to parameterize a problem in such a way that the instances with small values of k are of real interest.

When the decision time is replaced by the much more powerful $|I|^{O(f(k))}$, we obtain the class XP, where each problem is polynomial-time solvable for any fixed value of k. There is a number of parameterized complexity classes between FPT and XP (for each integer $t \geq 1$, there is a class W[t]) and they form the following tower:

$$\text{FPT} \subseteq \text{W}[1] \subseteq \text{W}[2] \subseteq \cdots \subseteq \text{W}[\text{P}] \subseteq \text{XP}.$$

Here W[P] is the class of all parameterized problems (I, k) that can be decided in $f(k)|I|^{O(1)}$ time by a nondeterministic Turing machine that makes at most $f(k)\log|I|$ nondeterministic steps for some function f. For the definition of classes W[t], see, e.g., [16, 18] (we do not use these classes in the rest of the paper).

Π is in *para-NP* if membership of (I, k) in Π can be decided in nondeterministic time $O(f(k)|I|^c)$, where $|I|$ is the size of I, $f(k)$ is an arbitrary function of the parameter k only, and c is a constant independent from k and I. Here, nondeterministic time means that we can use nondeterministic Turing machine. A parameterized problem Π' is *para-NP-complete* if it is in para-NP and for any parameterized problem Π in para-NP there is an FPT-reduction from Π to Π'.

Given a pair Π, Π' of parameterized problems, a *bikernelization from* Π *to* Π' is a polynomial-time algorithm that maps an instance (I, k) to an instance (I', k') (the *bikernel*) such that (i) $(I, k) \in \Pi$ if and only if $(I', k') \in \Pi'$, (ii) $k' \leq f(k)$, and (iii) $|I'| \leq g(k)$ for some functions f and g. The function $g(k)$ is called the *size* of the bikernel. A *kernelization* of a parameterized problem Π is simply a bikernelization from Π to itself. Then (I', k') is a *kernel*. The term bikernel was coined by Alon et al. [2]; in [4] a bikernel is called a generalized kernel.

It is well-known that a parameterized problem Π is fixed-parameter tractable if and only if it is decidable and admits a kernelization [16, 18]. This result can be extended as follows: A decidable parameterized problem Π is fixed-parameter tractable if and only if it admits a bikernelization from itself to a decidable parameterized problem Π' [2].

Due to applications, low degree polynomial size kernels are of main interest. Unfortunately, many fixed-parameter tractable problems do not have kernels of polynomial size unless the polynomial hierarchy collapses to the third level [4, 5, 19]. For further background and terminology on parameterized complexity we refer the reader to the monographs [16, 18].

The following lemma of Alon et al. [2] inspired by a lemma from [5] shows that polynomial bikernels imply polynomial kernels.

▶ **Lemma 1.** *Let* Π, Π' *be a pair of decidable parameterized problems such that the nonparameterized version of* Π' *is in NP, and the nonparameterized version of* Π *is NP-complete. If there is a bikernelization from* Π *to* Π' *producing a bikernel of polynomial size, then* Π *has a polynomial-size kernel.*

Recently many lower bound results for parameterized complexity were proved under the assumption that the Exponential Time Hypothesis (EHT) (see [16]) holds. ETH claims that 3-SAT cannot be solved in $O(2^{\delta n})$ time for some $\delta > 0$, where n is the number of variables in the CNF formula of 3-SAT.

Henceforth $[n]$ stands for the set $\{1, 2, \ldots, n\}$.

3 Strictly Above/Below Expectation Method

This section briefly describes basics of the method.

Let us start by outlining the very basic principles of the probabilistic method which will be implicitly used later. Given random variables $X_1, \ldots, X_n$, the fundamental property known as *linearity of expectation* states that $\mathbb{E}(X_1 + \ldots + X_n) = \mathbb{E}(X_1) + \ldots + \mathbb{E}(X_n)$. The *averaging argument* utilizes the fact that there is a point for which $X \geq \mathbb{E}(X)$ and a point for which $X \leq \mathbb{E}(X)$ in the probability space. Also a positive probability $\mathbb{P}(A) > 0$ for some event A means that there is at least one point in the probability space which belongs to A. For example, $\mathbb{P}(X \geq k) > 0$ tells us that there exists a point for which $X \geq k$.

A random variable is *discrete* if its distribution function has a finite or countable number of positive increases. A random variable X is *symmetric* if $-X$ has the same distribution function as X. If X is discrete, then X is symmetric if and only if $\mathbb{P}(X = a) = \mathbb{P}(X = -a)$ for each real a. Let X be a symmetric variable for which the first moment $\mathbb{E}(X)$ exists. Then $\mathbb{E}(X) = \mathbb{E}(-X) = -\mathbb{E}(X)$ and, thus, $\mathbb{E}(X) = 0$. The following easy to prove [26] result, is the simplest tool of the Strictly Above/Below Expectation method as it allows us sometimes to show that a certain random variable takes values (significantly) above/below its expectation.

▶ **Lemma 2.** *If X is a symmetric random variable and $\mathbb{E}(X^2)$ is finite, then*

$$\mathbb{P}(\ X \geq \sqrt{\mathbb{E}(X^2)}\ ) > 0.$$

We will illustrate the usefulness of the lemma using the 2-Linear Ordering Above Average problem. Let $D = (V, A)$ be a digraph on n vertices with no loops or parallel arcs in which every arc ij has a positive integral weight w_{ij}. Consider an ordering $\alpha : V \to [n]$ and the subdigraph $D_\alpha = (V, \{ij \in A : \ \alpha(i) < \alpha(j)\})$ of D. Note that D_α is acyclic.

2-Linear Ordering Above Average (2-Linear Ordering-AA)
Instance: A digraph $D = (V, A)$, each arc ij has an integral positive weight w_{ij}, and a positive integer κ.
Parameter: The integer κ.
Question: Is there a subdigraph D_α of D of weight at least $W/2 + \kappa$, where $W = \sum_{ij \in A} w_{ij}$?

Mahajan, Raman, and Sikdar [42] asked whether 2-Linear Ordering-AA is FPT for the special case when all arcs are of weight 1. Gutin et al. [26] solved the problem by obtaining a quadratic kernel for the problem. In fact, the problem can be solved using the following result of Alon [1]: there exists an ordering α such that D_α has weight at least $(\frac{1}{2} + \frac{1}{16|V|})W$. However, the proof in [1] uses a probabilistic approach for which a derandomization is not known yet and, thus, we cannot find the appropriate α deterministically. Moreover, the probabilistic approach in [1] is quite specialized. Thus, we will briefly describe a solution from [26]. Consider the following reduction rule:

▶ **Reduction Rule 1.** *Assume D has a directed 2-cycle iji; if $w_{ij} = w_{ji}$ delete the cycle, if $w_{ij} > w_{ji}$ delete the arc ji and replace w_{ij} by $w_{ij} - w_{ji}$, and if $w_{ji} > w_{ij}$ delete the arc ij and replace w_{ji} by $w_{ji} - w_{ij}$.*

It is easy to check that the answer to 2-Linear Ordering-AA for a digraph D is Yes if and only if the answer to 2-Linear Ordering-AA is Yes for a digraph obtained from D using the reduction rule as long as possible. Note that applying Rule 1 as long as possible results in an *oriented graph*, i.e., a digraph with no directed 2-cycle.

▶ **Theorem 3** ([26]). 2-Linear Ordering-AA *has a kernel with $O(\kappa^2)$ arcs.*

Proof. Consider a random ordering: $\alpha : V \to [n]$ and a random variable $X(\alpha)$ defined by $X(\alpha) = \frac{1}{2}\sum_{ij \in A} x_{ij}(\alpha)$, where $x_{ij}(\alpha) = w_{ij}$ if $\alpha(i) < \alpha(j)$ and $x_{ij}(\alpha) = -w_{ij}$, otherwise. It is easy to see that $X(\alpha) = \sum\{w_{ij} : \ ij \in A, \alpha(i) < \alpha(j)\} - W/2$. Thus, the answer to 2-Linear Ordering-AA is Yes if and only if there is an ordering $\alpha : V \to [n]$ such that $X(\alpha) \geq \kappa$.

By Rule 1, we may assume that the input of 2-Linear Ordering-AA is an oriented graph $D = (V, A)$. Let $\alpha : V \to [n]$ be a random ordering. Since $X(-\alpha) = -X(\alpha)$, where $-\alpha(i) = n + 1 - \alpha(i)$, X is a symmetric random variable and, thus, we can apply Lemma 2. It was proved in [26] that $\mathbb{E}(X^2) \geq |A|/12$. By this inequality and Lemma 2, we have

$\mathbb{P}(\ X \geq \sqrt{|A|/12}\ ) > 0$. Thus, if $\sqrt{|A|/12} \geq \kappa$, there is an ordering $\beta : V \to [n]$ such that $X(\beta) \geq k$ and so the answer to 2-Linear Ordering-AA is Yes. Otherwise, $\sqrt{|A|/12} \geq \kappa$ implying $|A| \leq 12\kappa^2$ and we are done. ◀

By deleting isolated vertices (if any), we can obtain a kernel with $O(\kappa^2)$ arcs and vertices. Kim and Williams [36] proved that 2-Linear Ordering has a kernel with a linear number of variables.

If a random variable X is not symmetric then the following lemma can be used instead of Lemma 2.

▶ **Lemma 4** (Alon et al. [2]). *Let X be a real random variable and suppose that its first, second and fourth moments satisfy $\mathbb{E}[X] = 0$, $\mathbb{E}[X^2] = \sigma^2 > 0$ and $\mathbb{E}[X^4] \leq c\mathbb{E}[X^2]^2$, respectively, for some constant c. Then $\mathbb{P}(X > \frac{\sigma}{2\sqrt{c}}) > 0$.*

To check whether $\mathbb{E}[X^4] \leq c\mathbb{E}[X^2]^2$ we often can use the following well-known inequality whose proof can be found in [16] and [45].

▶ **Lemma 5** (Hypercontractive Inequality [6]). *Let $f = f(x_1, \dots, x_n)$ be a polynomial of degree r in n variables $x_1, \dots, x_n$ each with domain $\{-1, 1\}$. Define a random variable X by choosing a vector $(\epsilon_1, \dots, \epsilon_n) \in \{-1, 1\}^n$ uniformly at random and setting $X = f(\epsilon_1, \dots, \epsilon_n)$. Then $\mathbb{E}[X^4] \leq 9^r\mathbb{E}[X^2]^2$.*

If $f = f(x_1, \dots, x_n)$ is a polynomial in n variables $x_1, \dots, x_n$ each with domain $\{-1, 1\}$, then it can be written as $f = \sum_{I \subseteq [n]} c_I \prod_{i \in S} x_i$, where $[n] = \{1, \dots, n\}$ and c_I is a real for each $I \subseteq [n]$.

The following dual, in a sense, form of the Hypercontractive Inequality was proved by Gutin and Yeo [28]; for a weaker result, see [26].

▶ **Lemma 6.** *Let $f = f(x_1, \dots, x_n)$ be a polynomial in n variables $x_1, \dots, x_n$ each with domain $\{-1, 1\}$ such that $f = \sum_{I \subseteq [n]} c_I \prod_{i \in S} x_i$. Suppose that no variable x_i appears in more than ρ monomials of f. Define a random variable X by choosing a vector $(\epsilon_1, \dots, \epsilon_n) \in \{-1, 1\}^n$ uniformly at random and setting $X = f(\epsilon_1, \dots, \epsilon_n)$. Then $\mathbb{E}[X^4] \leq (2\rho + 1)\mathbb{E}[X^2]^2$.*

The following lemma is easy to prove, cf. [26]. In fact, the equality there is a special case of Parseval's Identity in Harmonic Analysis, cf. [45].

▶ **Lemma 7.** *Let $f = f(x_1, \dots, x_n)$ be a polynomial in n variables $x_1, \dots, x_n$ each with domain $\{-1, 1\}$ such that $f = \sum_{I \subseteq [n]} c_I \prod_{i \in I} x_i$. Define a random variable X by choosing a vector $(\epsilon_1, \dots, \epsilon_n) \in \{-1, 1\}^n$ uniformly at random and setting $X = f(\epsilon_1, \dots, \epsilon_n)$. Then $\mathbb{E}[X^2] = \sum_{i \in I} c_I^2$.*

We will give a relatively simple application of Lemmas 5 and 7 in Subsection 4.1. Another application is in Subsection 7.2.

4 Boolean Max-r-CSPs Above Average

Throughout this section, r is a positive integral constant. Recall that the problem Max-r-CSP-AA is given by a set V of n Boolean variables and a set of m Boolean formulas; each formula is assigned an integral positive weight and contains at most r variables from V. Averaging over all truth assignments, we can find the average value A of the weight of satisfied formulas. We wish to decide whether there exists a truth assignment that satisfies formulas of total weight at least $A + k$, where k is the parameter (k is a nonnegative integer).

Recall that the problem Max-r-Lin2-AA is a special case of Max-r-CSP-AA when every formula is a linear equation over $\mathbb{F}_2$ with at most r variables and that Max-Lin2-AA is the extension of Max-r-Lin2-AA when we do not bound the number of variables in an equation. We will see that for both Max-r-CSP-AA and Max-Lin2-AA, $A = W/2$, where W is the total weight of all equations.

Subsection 4.1 is devoted to parameterized complexity results on Max-r-Lin2-AA which are not only of interest by themselves, but also as tools useful for Max-r-CSP-AA. In particular, we will prove that Max-r-Lin2-AA has a kernel of quadratic size. Since some basic results on Max-r-Lin2-AA hold also for Max-Lin2-AA, in general, we will show them for Max-Lin2-AA.

In Subsection 4.2, we give a proof scheme of a result by Alon et al. [2] that Max-r-CSP-AA has a a kernel of polynomial size. The main idea of the proof is to reduce Max-r-CSP-AA to Max-r-Lin2-AA and use the above kernel result on Max-r-Lin2-AA and Lemma 1. This shows the existence of a polynomial-size kernel, but does not allow us to obtain a bound on the degree of the polynomial. We complete the section, by pointing out that for Max-r-Sat-AA, the reduction to Max-r-Lin2-AA can be complemented by a reduction from Max-r-Lin2-AA back to Max-r-Sat-AA and so we obtain a quadratic kernel for Max-r-Sat-AA.

4.1 Max-r-Lin-AA

In the Max-Lin2-AA problem, we are given a system S consisting of m linear equations in n variables over $\mathbb{F}_2$ in which each equation is assigned a positive integral weight. If we add the requirement that every equation has at most r variables then we get Max-r-Lin2-AA. Let us write the system S as $\sum_{i\in I} z_i = b_I$, $I \in \mathcal{F}$, and let w_I denote the weight of an equation $\sum_{i\in I} z_i = b_I$. Clearly, $m = |\mathcal{F}|$. Let $W = \sum_{I\in\mathcal{F}} w_I$ and let $\text{sat}(S)$ be the maximum total weight of equations that can be satisfied simultaneously.

For each $i \in [n]$, set $z_i = 1$ with probability $1/2$ independently of the rest of the variables. Then each equation is satisfied with probability $1/2$ and the expected weight of satisfied equations is $W/2$ (as our probability distribution is uniform, $W/2$ is also the average weight of satisfied equations). Hence $W/2$ is a lower bound; to see its tightness consider a system of pairs of equations of the form $\sum_{i\in I} z_i = 0$, $\sum_{i\in I} z_i = 1$ of weight 1. The aim in both Max-Lin2-AA and Max-r-Lin2-AA is to decide whether for the given system S, $\text{sat}(S) \geq W/2 + k$, where k is the parameter. It is well-known that, in polynomial time, we can find an assignment to the variables that satisfies equations of total weight at least $W/2$, but, for any $\epsilon > 0$ it is NP-hard to decide whether there is an assignment satisfying equations of total weight at least $W(1+\epsilon)/2$ [31].

Henceforth, it will often be convenient for us to consider linear equations in their multiplicative form, i.e., instead of an equation $\sum_{i\in I} z_i = b_I$ with $z_i \in \{0,1\}$, we will consider the equation $\prod_{i\in I} x_i = (-1)^{b_I}$ with $x_i \in \{-1,1\}$. Clearly, an assignment $z^0 = (z_1^0, \dots, z_n^0)$ satisfies $\sum_{i\in I} z_i = b_I$ if and only if the assignment $x^0 = (x_1^0, \dots, x_n^0)$ satisfies $\prod_{i\in I} x_i = (-1)^{b_I}$, where $x_i^0 = (-1)^{z_i^0}$ for each $i \in [n]$.

Let $\varepsilon(x) = \sum_{I\in\mathcal{F}} w_I (-1)^{b_I} \prod_{i\in I} x_i$ (each $x_i \in \{-1,1\}$) and note that $\varepsilon(x^0)$ is the difference between the total weight of satisfied and falsified equations when $x_i = x_i^0$ for each $i \in [n]$. We will call $\varepsilon(x)$ the *excess* and the maximum possible value of $\varepsilon(x)$ the *maximum excess.* The following claim is easy to check.

▶ **Lemma 8.** *Observe that the answer to* Max-Lin2-AA *and* Max-r-Lin2-AA *is* Yes *if and only if the maximum excess is at least* $2k$*.*

Let A be the matrix over $\mathbb{F}_2$ corresponding to the set of equations in S, such that $a_{ji} = 1$ if $i \in I_j$ and 0, otherwise.

Consider two reduction rules for Max-Lin2-AA introduced by Gutin et al. [26].

▶ **Reduction Rule 2.** *If we have, for a subset I of $[n]$, an equation $\prod_{i\in I} x_i = b'_I$ with weight w'_I, and an equation $\prod_{i\in I} x_i = b''_I$ with weight w''_I, then we replace this pair by one of these equations with weight $w'_I + w''_I$ if $b'_I = b''_I$ and, otherwise, by the equation whose weight is bigger, modifying its new weight to be the difference of the two old ones. If the resulting weight is 0, we delete the equation from the system.*

▶ **Reduction Rule 3.** *Let $t = \text{rank}A$ and suppose columns $a^{i_1}, \ldots, a^{i_t}$ of A are linearly independent. Then delete all variables not in $\{x_{i_1}, \ldots, x_{i_t}\}$ from the equations of S.*

▶ **Lemma 9** ([26]). *Let S' be obtained from S by Rule 2 or 3. Then the maximum excess of S' is equal to the maximum excess of S. Moreover, S' can be obtained from S in time polynomial in n and m.*

▶ **Definition 10.** If we cannot change a weighted system S using Rules 2 and 3, we call it *irreducible*.

Now we are read to prove the following result.

▶ **Theorem 11** ([26]). *The problem* Max-*r*-Lin2-AA *admits a kernel with at most $O(k^2)$ variables and equations.*

Proof. Let the system S be irreducible. Consider the excess

$$\varepsilon(x) = \sum_{I\in\mathcal{F}} w_I(-1)^{b_I} \prod_{i\in I} x_i. \tag{1}$$

Let us assign value -1 or 1 to each x_i with probability 1/2 independently of the other variables. Then $X = \varepsilon(x)$ becomes a random variable. By Lemma 7, we have $\mathbb{E}(X^2) = \sum_{I\in\mathcal{F}} w_I^2$. Therefore, by Lemmas 4 and 5,

$$\mathbb{P}[\ X \geq \sqrt{m}/(2\cdot 3^r)\ ] \geq \mathbb{P}\left[\ X \geq \sqrt{\sum_{I\in\mathcal{F}} w_I^2}/(2\cdot 3^r)\ \right] > 0.$$

Hence by Remark 8, if $\sqrt{m}/(2\cdot 3^r) \geq 2k$, then the answer to Max-*r*-Lin2-AA is Yes. Otherwise, $m = O(k^2)$ and, by Rule 3, we have $n \leq m = O(k^2)$. ◀

The bound on the number of variables can be improved and it was done by Crowston et al. [12] and Kim and Williams [36]. The best known improvement is by Crowston et al. [11]:

▶ **Theorem 12.** *The problem* Max-*r*-Lin2-AA *admits a kernel with at most $(2k-1)r$ variables.*

Theorem 12 implies the following:

▶ **Corollary 13.** *There is an algorithm of runtime $2^{O(k)} + m^{O(1)}$ for* Max-*r*-Lin2-AA.

Kim and Williams [36] proved that the last result is best possible, in a sense, if the Exponential Time Hypothesis (ETH) holds.

▶ **Theorem 14** ([36]). *If* Max-3-Lin2-AA *can be solved in $O(2^{\epsilon k}2^{\epsilon m})$ time for every $\epsilon > 0$, then ETH does not hold.*

4.2 Max-r-CSP-AA and Max-r-Sat-AA

Consider first a detailed formulation of Max-r-CSP-AA. Let $V = \{v_1, \ldots, v_n\}$ be a set of variables, each taking values -1 (True) and 1 (False). We are given a set Φ of Boolean functions, each involving at most r variables, and a collection $\mathcal{F}$ of m Boolean functions, each $f \in \mathcal{F}$ being a member of Φ, each with a positive integral weight and each acting on some subset of V. We are to decide whether there is a truth assignment to the n variables such that the total weight of satisfied functions is at least $A + k$, where A is the average weight (over all truth assignments) of satisfied functions and k is the parameter.

Note that A is a tight lower bound for the problem, whenever the family Φ is closed under replacing each variable by its complement, since if we apply any Boolean function to all 2^r choices of literals whose underlying variables are any fixed set of r variables, then any truth assignment to the variables satisfies exactly the same number of these 2^r functions.

Note that if Φ consists of clauses, we get Max-r-Sat-AA. In Max-r-Sat-AA, $A = \sum_{j=1}^m w_j(1 - 2^{-r_j})$, where w_j and r_j are the weight and the number of variables of Clause j, respectively. Clearly, A is a tight lower bound for Max-r-Sat.

Following [3], for a Boolean function f of weight $w(f)$ and on $r(f) \le r$ Boolean variables $x_{i_1}, \ldots, x_{i_{r(f)}}$, we introduce a polynomial $h_f(x)$, $x = (x_1, \ldots, x_n)$ as follows. Let $S_f \subset \{-1,1\}^{r(f)}$ denote the set of all satisfying assignments of f. Then

$$h_f(x) = w(f)2^{r-r(f)} \sum_{(v_1,\ldots,v_{r(f)}) \in S_f} [\prod_{j=1}^{r(f)} (1 + x_{i_j} v_j) - 1].$$

Let $h(x) = \sum_{f \in \mathcal{F}} h_f(x)$. It is easy to see (cf. [2]) that the value of $h(x)$ at some x^0 is precisely $2^r(U - A)$, where U is the total weight of the functions satisfied by the truth assignment x^0. Thus, the answer to Max-r-CSP-AA is Yes if and only if there is a truth assignment x^0 such that $h(x^0) \ge k2^r$.

Algebraic simplification of $h(x)$ will lead us the following (Fourier expansion of $h(x)$, cf. [45]):

$$h(x) = \sum_{S \in \mathcal{F}} c_S \prod_{i \in S} x_i, \tag{2}$$

where $\mathcal{F} = \{\emptyset \neq S \subseteq [n] : \ c_S \neq 0, |S| \le r\}$. Thus, $|\mathcal{F}| \le n^r$. The sum $\sum_{S \in \mathcal{F}} c_S \prod_{i \in S} x_i$ can be viewed as the excess of an instance of Max-r-Lin2-AA and, thus, we can reduce Max-r-CSP-AA into Max-r-Lin2-AA in polynomial time (since r is fixed, the algebraic simplification can be done in polynomial time and it does not matter whether the parameter of Max-r-Lin2-AA is k or $k' = k2^r$). By Theorem 11, Max-r-Lin2-AA has a kernel with $O(k^2)$ variables and equations. This kernel is a bikernel from Max-r-CSP-AA to Max-r-Lin2-AA. Thus, by Lemma 1, we obtain the following theorem of Alon et al. [2].

▶ **Theorem 15.** Max-r-CSP-AA *admits a polynomial-size kernel.*

Applying a reduction from Max-r-Lin2-AA to Max-r-Sat-AA in which each monomial in (2) is replaced by 2^{r-1} clauses, Alon et al. [2] obtained the following:

▶ **Theorem 16.** Max-r-Sat-AA *admits a kernel with $O(k^2)$ clauses and variables.*

Using also Theorem 12, it is easy to improve this theorem with respect to the number of variables in the kernel. Note that this result was first obtained by Kim and Williams [36].

▶ **Theorem 17.** Max-r-Sat-AA *admits a kernel with $O(k)$ variables.*

4.3 Max-r-CSP-AA with global cardinality constraint

Recall the formulation of Max-r-CSP-AA: $V = \{v_1, \ldots, v_n\}$ is a set of variables, each taking values from $\{-1, 1\}$. We are given m Boolean formulas, each with an integral positive weight. We wish to decide if we can satisfy clauses with a total weight of k more than the average weight (if every variables is assigned -1 or 1 with equal probability).

Chen and Zhou [9] consider the unweighted case of Max-r-CSP-AA but allow for a global cardinality constraint. That is, we can restrict the number of 1's (or -1's) to be a given fraction of the total number of variables. Consider the sum $\sum_{i=1}^{n} v_i$ and note that this is an integer between $-n$ and n. We now consider the case when $\sum_{i=1}^{n} v_i = \alpha n$ and $-p_0 \leq \alpha \leq p_0$ for some fixed integer $p_0 < 1$. Note that α may depend on n, but has to be bounded by constants $-p_0$ and p_0 (p_0 does not depend on n and will be considered a constant in the complexity). We now formally describe the problem.

Let $0 \leq p_0 < 1$ be a constant and for every n, let α_n satisfy $-p_0 \leq \alpha_n \leq p_0$. We consider all instances that satisfy the following:

$$\sum_{i=1}^{n} v_i = \alpha_n n\,.$$

For example if $\alpha_n = 0$ then we require to be equally many 1's and -1's in the assignments to $v_1, v_2, \ldots, v_n$ (this is called a *bisection constraint*). If $\alpha_n = 1/2$ then we require exactly one quarter of all variables $v_1, v_2, \ldots, v_n$ to be assigned -1 and three quarters to be assigned 1.

One application of the global cardinality constraint can be found in the MaxBisection problem where we are given a graph G and want to partition the vertices into two equal-size sets such that we have the maximum possible number of edges between the two sets. Let $v_1, v_2, \ldots, v_n$ be the vertices of the graph G and, with abuse of notation, also the variables in our instance of Max-r-CSP-AA. If v_iv_j is an edge in G then add the constraint $v_i \neq v_j$. Adding the global cardinality constraint $\sum_{i=1}^{n} v_i = 0$ now gives us an instance of Max-r-CSP-AA (with global cardinality constraint) which has a solution k above the average if and only if MaxBisection has a solution with k more edges in the cut than an average cut (given that both partite sets are equally large).

If we do not require both partite sets to be equally large we get the problem MaxCut, in which every edge has probability 1/2 of belonging to a random cut. However for the MaxBisection problem this probability is slightly higher. Consider en edge v_iv_j in the graph G. Without loss of generality, let v_i be assigned 1. Of the remaining $n-1$ vertices $n/2-1$ will be assigned 1 and $n/2$ will be assigned -1. Therefore the probability that v_iv_j is in the cut will be as follows.

$$\frac{n/2}{n-1} = \frac{1}{2} + \frac{1}{2(n-1)}\,.$$

Therefore in the MaxBisection-AA we are looking to decide if there is a solution with at least $m\left(\frac{1}{2} + \frac{1}{2(n-1)}\right) + k$ edges in the cut, where k is the parameter.

MaxBisection-AA was shown to be FPT and have a $O(k^2)$ kernel by Chen and Zhou [9]. This significantly improves a result by Gutin and Yeo [29] who showed a similar result when looking for a solution with at least $m/2 + k$ edges, where k is the parameter.

In fact, in [9] it is proved that each unweighted Max-r-CSP-AA with a global cardinality constraint is FPT and has a kernel of size $O(k^2)$.

▶ **Theorem 18** ([9]). *For every p_0 with $0 < p_0 \leq 1$ there exists a kernel of size $O(k^2)$ for the unweighted* Max-r-CSP-AA *with global cardinality constraint (given by p_0).*

If $m \in n^{O(r)}$ (which is the case if we do not repeat constraints), then in Theorem 18 the polynomial algorithm that produces the kernel has runtime $n^{O(r)}$ (where r was the number of variables in each constraint). If $m \notin n^{O(r)}$, it is not difficult to obtain a runtime of $m^{O(r)}$.

The size of the kernel in Theorem 18 depends on p_0 and r, but in the theorem these are constants. Theorem 18 furthermore implies the following result (where p_0 and r again are considered constants).

▶ **Theorem 19** ([9]). *For every $0 < p_0 \leq 1$ and unweighted instance* Max-r-CSP-AA *with global cardinality constraint (given by p_0) and $m \in n^{O(1)}$ there exists an algorithm with runtime $n^{O(1)} + 2^{O(k^2)}$ that decides if there is a solution satisfying k constraints more than the average.*

The proofs of the above results are deep and beyond the scope of this survey. They use a version of the hypercontractive inequality where the probability space is given by all assignments satisfying the global cardinality constraint. Therefore the variables are not independent, which complicates matters compared to previous proofs of the ordinary hypercontractive inequality. The proof of this new hypercontractive inequality relies on the analysis of the eigenvalues of several $n^{O(r)} \times n^{O(r)}$ set-symmetric matrices.

5 Parameterizations of MaxLin2

In the Max-Lin2 problem, we are given a system S of m linear equations in n variables over $\mathbb{F}_2$ in which each equation is assigned a positive integral weight. Our aim is to find am assignment to the variables that maximizes the total weight of satisfied equations. In this section, we will consider the following two parameterizations of Max-Lin2 :

- MaxLin2-AA is the same problem as Max-r-Lin2-AA, but the number of variables in an equation is not bounded. Thus, MaxLin2-AA is a generalization of Max-r-Lin2-AA. In Subsection 5.1 we present a scheme of a recent proof by Crowston et al. [11] that MaxLin2-AA is FPT and has a kernel with polynomial number of variables. This result finally solved an open question of Mahajan et al. [42]. Still, we do not know whether MaxLin2-AA has a kernel of polynomial size and we are able to give only partial results on the topic.
- Let W be the total weight of all equations in S. In Subsection 5.2 we consider the following parameterized version of MaxLin2: decide whether there is an assignment satisfying equations of total weight at least $W - k$, where k is a nonnegative parameter.

5.1 MaxLin2-AA

Let S be an irreducible system of Max-Lin2-AA (recall Definition 10). Consider the following algorithm introduced in [12]. We assume that, in the beginning, no equation or variable in S is marked.

```
Algorithm $\mathcal{H}$
While the system $S$ is nonempty do the following:
1. Choose an equation $\prod_{i\in I} x_i = b$ and mark a variable $x_l$ such that $l \in I$.
2. Mark this equation and delete it from the system.
3. Replace every equation $\prod_{i\in I'} x_i = b'$ in the system containing $x_l$ by $\prod_{i\in I\Delta I'} x_i = bb'$, where $I\Delta I'$ is the symmetric difference of $I$ and $I'$ (the weight of the equation is unchanged).
4. Apply Reduction Rule 2 to the system.
```

The *maximum $\mathcal{H}$-excess* of S is the maximum possible total weight of equations marked by $\mathcal{H}$ for S taken over all possible choices in Step 1 of $\mathcal{H}$. The following lemma indicates the potential power of $\mathcal{H}$.

▶ **Lemma 20** ([12]). *Let S be an irreducible system. Then the maximum excess of S equals its maximum $\mathcal{H}$-excess.*

This lemma gives no indication on how to choose equations in Step 1 of Algorithm $\mathcal{H}$. As the problem Max-Lin2-AA is NP-hard, we cannot hope to obtain an polynomial-time procedure for optimal choice of equations in Step 1 and, thus, have to settle for a good heuristic. For the heuristic we need the following notion first used in [12]. Let K and M be sets of vectors in $\mathbb{F}_2^n$ such that $K \subseteq M$. We say K is *M-sum-free* if no sum of two or more distinct vectors in K is equal to a vector in M. Observe that K is M-sum-free if and only if K is linearly independent and no sum of vectors in K is equal to a vector in $M \backslash K$.

The following lemma was proved implicitly in [12]; we provide a short proof of this result.

▶ **Lemma 21.** *Let S be an irreducible system of* Max-Lin2-AA *and let A be the matrix corresponding to S. Let M be the set of rows of A (viewed as vectors in $\mathbb{F}_2^n$) and let K be an M-sum-free set of k vectors. Let $w_{\min}$ be the minimum weight of an equation in S. Then, in time in $(nm)^{O(1)}$, we can find an assignment to the variables of S that achieves excess of at least $w_{\min} \cdot k$.*

Proof. Let $\{e_{j_1}, \dots, e_{j_k}\}$ be the set of equations corresponding to the vectors in K. Run Algorithm $\mathcal{H}$, choosing at Step 1 an equation of S from $\{e_{j_1}, \dots, e_{j_k}\}$ each time, and let S' be the resulting system. Algorithm $\mathcal{H}$ will run for k iterations of the while loop as no equation from $\{e_{j_1}, \dots, e_{j_k}\}$ will be deleted before it has been marked.

Indeed, suppose that this is not true. Then for some e_{j_l} and some other equation e in S, after applying Algorithm $\mathcal{H}$ for at most $l-1$ iterations e_{j_l} and e contain the same variables. Thus, there are vectors $v_j \in K$ and $v \in M$ and a pair of nonintersecting subsets K' and K'' of $K \setminus \{v, v_j\}$ such that $v_j + \sum_{u \in K'} u = v + \sum_{u \in K''} u$. Thus, $v = v_j + \sum_{u \in K' \cup K''} u$, a contradiction to the definition of K. ◀

The main result of this subsection is the following theorem whose proof is based on Theorems 23 and 25.

▶ **Theorem 22** ([11]). *The problem* MaxLin2-AA *has a kernel with at most $O(k^2 \log k)$ variables.*

▶ **Theorem 23** ([12]). *Let S be an irreducible system of* MaxLin2-AA *and let $k \geq 2$. If $k \leq m \leq 2^{n/(k-1)} - 2$, then the maximum excess of S is at least k. Moreover, we can find an assignment with excess of at least k in time $m^{O(1)}$.*

This theorem can easily be proved using Lemma 21 and the following lemma.

▶ **Lemma 24** ([12]). *Let M be a set in $\mathbb{F}_2^n$ such that M contains a basis of $\mathbb{F}_2^n$, the zero vector is in M and $|M| < 2^n$. If k is a positive integer and $k+1 \leq |M| \leq 2^{n/k}$ then, in time $|M|^{O(1)}$, we can find an M-sum-free subset K of M with at least $k+1$ vectors.*

▶ **Theorem 25** ([11]). *There exists an $n^{2k}(nm)^{O(1)}$-time algorithm for* MaxLin2-AA *that returns an assignment of excess of at least $2k$ if one exists, and returns* No *otherwise.*

The proof of this theorem in [11] is based on constructing a special depth-bounded search tree.

Now we will present the proof of Theorem 22 from [11].

Proof of Theorem 22. Let $\mathcal{L}$ be an instance of MaxLin2-AA and let S be the system of $\mathcal{L}$ with m equations and n variables. We may assume that S is irreducible. Let the parameter k be an arbitrary positive integer.

If $m < 2k$ then $n < 2k = O(k^2 \log k)$. If $2k \le m \le 2^{n/(2k-1)} - 2$ then, by Theorem 23 and Remark 8, the answer to $\mathcal{L}$ is Yes and the corresponding assignment can be found in polynomial time. If $m \ge n^{2k} - 1$ then, by Theorem 25, we can solve $\mathcal{L}$ in polynomial time.

Finally we consider the case $2^{n/(2k-1)} - 2 \le m \le n^{2k} - 2$. Hence, $n^{2k} \ge 2^{n/(2k-1)}$. Therefore, $4k^2 \ge 2k + n/\log n \ge \sqrt{n}$ and $n \le (2k)^4$. Hence, $n \le 4k^2 \log n \le 4k^2 \log(16k^4) = O(k^2 \log k)$.

Since S is irreducible, $m < 2^n$ and thus we have obtained the desired kernel. ◀

Now let us consider some cases where we can prove that MaxLin2-AA has a polynomial-size kernel. Consider first the case when each equation in S has odd number of variables. Then we have the following theorem proved by Gutin et al. [26] using the Strictly Above/Below Expectation Method (in particular, Lemmas 2 and 7).

▶ **Theorem 26.** *The following special case of* MaxLin2-AA *admits a kernel with at most $4k^2$ variables and equations: there exists a subset U of variables such that each equation in $Ax = b$ has odd number of variables from U.*

Let us turn to results on MaxLin2-AA that do not require any parity conditions. One such result is Theorem 11. Gutin et al. [26] also proved the following 'dual' theorem.

▶ **Theorem 27.** *Let $\rho \ge 1$ be a fixed integer. Then* MaxLin2-AA *restricted to instances where no variable appears in more than ρ equations, admits a kernel with $O(k^2)$ variables and equations.*

The proof is similar to that of Theorem 11, but Lemma 6 (in fact, its weaker version obtained in [26]) is used instead of Lemma 5.

5.2 MaxLin2-B

In 2011, Arash Rafiey asked to determine the parameterized complexity of the following parameterized problem denoted MaxLin2-B. Let W be the total weight of all equations in S. Let us we consider the following parameterized version of MaxLin2: decide whether there is an assignment satisfying equations of total weight at least $W - k$, where k is a nonnegative parameter.

Crowston, Gutin, Jones and Yeo [14] proved that MaxLin2-B is W[1]-hard. This hardness result prompts us to investigate the complexity of MaxLin2-B in more detail by considering special cases of this problem. Let Max-($\le r$,$\le s$)-Lin2 (Max-($= r$,$= s$)-Lin2, respectively) denote the problem MaxLin2 restricted to instances, which have at most (exactly, respectively) r variables in each equation and at most (exactly) s appearances of any variable in all equations. In the special case when each equation has weight 1 and there are no two equations with the same left-hand side, MaxLin2-B will be denoted by MaxLin2-B[m]. Crowston et al. [14] proved that MaxLin2-B remains hard even after significant restrictions are imposed on it, namely, even Max-($= 3$,$= 3$)-Lin2-B[m] is W[1]-hard.

No further improvement of this result is possible unless FPT=W[1] as Crowston et al. [14] proved that Max-(≤ 2,*)-Lin2-B is fixed-parameter tractable, where symbol * indicates that no restriction is imposed on the number of appearances of a variable in the equations. Moreover, they showed that the nonparameterized problem Max-(*,≤ 2)-Lin2 is polynomial time solvable, where symbol * indicates that no restriction is imposed on the number of variables in any equation.

6 Parameterizations of MaxSat

In the well-known problem MaxSat, we are given a CNF formula F with m clauses and asked to determine the maximum number of clauses of F that can be satisfied by a truth assignment. In this section, we overview results on various parameterizations of MaxSat apart from Max-r-Sat-AA, where r is a constant (see Theorems 16 and 17 for this parameterized problem).

6.1 MaxSat above $m/2$

Let us assign True to each variable of F with probability $1/2$ and observe that the probability of a clause to be satisfied is at least $1/2$ and thus, by linearity of expectation, the expected number of satisfied clauses in F is at least $m/2$. Thus, by the averaging argument, there exists a truth assignment to the variables of F which satisfies at least $m/2$ clauses of F.

Let us denote by $\mathrm{sat}(F)$ the maximum number of clauses of F that can be satisfied by a truth assignment. The lower bound $\mathrm{sat}(F) \geq m/2$ is tight as we have $\mathrm{sat}(H) = m/2$ if $H = (x_1) \wedge (\bar{x}_1) \wedge \cdots \wedge (x_{m/2}) \wedge (\bar{x}_{m/2})$. Consider the following parameterization of MaxSat above tight lower bound introduced by Mahajan and Raman [41].

MaxSat-A$(m/2)$
Instance: A CNF formula F with m clauses (clauses may appear several times in F) and a nonnegative integer k.
Parameter: k.
Question: $\mathrm{sat}(F) \geq m/2 + k$?

Mahajan and Raman [41] proved that MaxSat-A$(m/2)$ admits a kernel with at most $6k+3$ variables and $10k$ clauses. Crowston et al. [15] improved this result, by obtaining a kernel with at most $4k$ variables and $(2\sqrt{5}+4)k$ clauses. The improved result is a simple corollary of a new lower bound on $\mathrm{sat}(F)$ obtained in [15], which is significantly stronger than the simple bound $\mathrm{sat}(F) \geq m/2$. We give the new lower bound below, in Theorem 34.

For a variable x in F, let $m(x)$ denote the number of pairs of unit of clauses $(x), (\bar{x})$ that have to be deleted from F such that F has no pair $(x), (\bar{x})$ any longer. Let $\mathrm{var}(F)$ be the set of all variables in F and let $\ddot{m} = \sum_{x \in \mathrm{var}(F)} m(x)$. The following is a stronger lower bound on $\mathrm{sat}(F)$ than $m/2$.

▶ **Theorem 28.** *For a CNF formula F, we have $\mathrm{sat}(F) \geq \ddot{m}/2 + \hat{\phi}(m - \ddot{m})$, where $\hat{\phi} = (\sqrt{5}-1)/2 \approx 0.618$.*

6.2 Max-$r(n)$-Sat-AA

Max-$r(n)$-Sat-AA is a generalization of Max-r-Sat-AA, where $r(n)$ is no longer just a constant, it depends on n. The following results for Max-$r(n)$-Sat-AA were obtained by Crowston et al. [13].

▶ **Theorem 29.** Max-$r(n)$-Sat-AA *is para-NP-complete for $r(n) = \lceil \log n \rceil$.*

Assuming ETH, one can prove the following stronger result.

▶ **Theorem 30.** *Assuming ETH,* Max-$r(n)$-Sat-AA *is not FPT for any $r(n) \geq \log\log n + \phi(n)$, where $\phi(n)$ is any unbounded strictly increasing function of n.*

The following theorem shows that Theorem 30 provides a bound on $r(n)$ which is not far from optimal.

▶ **Theorem 31** ([13]). Max-$r(n)$-Sat-AA *is FPT for* $r(n) \leq \log\log n - \log\log\log n - \phi(n)$, *for any unbounded strictly increasing function* $\phi(n)$.

6.3 Parameterizations for MaxSat with t-Satisfiable CNF Formulas

A CNF formula F is *t-satisfiable* if for any t clauses in F, there is a truth assignment which satisfies all of them. It is easy to check that F is 2-satisfiable if and only if $\ddot{m} = 0$ and clearly Theorem 28 is equivalent to the assertion that if F is 2-satisfiable then $\text{sat}(F) \geq \hat{\phi}m$. The proof of this assertion by Lieberherr and Specker [38] is quite long; Yannakakis [51] gave the following short probabilistic proof. For $x \in \text{var}(F)$, let the probability of x being assigned True be $\hat{\phi}$ if (x) is in F, $1-\hat{\phi}$ if $(\bar{x})$ is in F, and 1/2, otherwise, independently of the other variables. Let us bound the probability $p(C)$ of a clause C to be satisfied. If C contains only one literal, then, by the assignment above, $p(C) = \hat{\phi}$. If C contains two literals, then, without loss of generality, $C = (x \vee y)$. Observe that the probability of x assigned False is at most $\hat{\phi}$ (it is $\hat{\phi}$ if $(\bar{x})$ is in F). Thus, $p(C) \geq 1 - \hat{\phi}^2$. If C containes more than two literals then it is easy to see that $p(C) \geq 1 - \hat{\phi}^2$. It remains to observe that $1 - \hat{\phi}^2 = \hat{\phi}$. Now to obtain the bound $\text{sat}(F) \geq \hat{\phi}m$ apply linearity of expectation and the averaging argument.

Note that $\hat{\phi}m$ is an *asymptotically* tight lower bound: for each $\epsilon > 0$ there are 2-satisfiable CNF formulae F with $\text{sat}(F) < m(\hat{\phi}+\epsilon)$ [38]. Thus, the following problem stated by Mahajan and Raman [41] is natural.

Max-2S-Sat-A($\hat{\phi}m$)
Instance: A 2-satisfiable CNF formula F with m clauses (clauses may appear several times in F) and a nonnegative integer k.
Parameter: k.
Question: $\text{sat}(F) \geq \hat{\phi}m + k$?

Mahajan and Raman [41] conjectured that Max-2S-Sat-A($\hat{\phi}m$) is FPT. Crowston et al. [15] solved this conjecture in the affirmative; moreover, they obtained a kernel with at most $(7+3\sqrt{5})k$ variables. This result is an easy corollary from a lower bound on $\text{sat}(F)$ given in Theorem 34, which, for 2-satisfiable CNF formulas, is stronger than the one in Theorem 28. The main idea of [15] is to obtain a lower bound on $\text{sat}(F)$ that includes the number of variables as a factor. It is clear that for general CNF formula F such a bound is impossible. For consider a formula containing a single clause c containing a large number of variables. We can arbitrarily increase the number of variables in the formula, and the maximum number of satisfiable clauses will always be 1. We therefore need a reduction rule that cuts out ‘excess’ variables. Our reduction rule is based on the notion of an expanding formula given below. Lemma 32 and Theorem 33 show the usefulness of this notion.

A CNF formula F is called *expanding* if for each $X \subseteq \text{var}(F)$, the number of clauses containing at least one variable from X is at least $|X|$ [20, 50]. The following lemma and its parts were proved by many authors, see, e.g., Fleischner et al. [20], Lokshtanov [40] and Szeider [50].

▶ **Lemma 32.** *Let F be a CNF formula and let V and C be its sets of variables and clauses. There exists a subset $C^* \subseteq C$ that can be found in polynomial time, such that the formula F' with clauses $C \setminus C^*$ and variables $V \setminus V^*$, where $V^* = \text{var}(C^*)$, is expanding. Moreover,* $\text{sat}(F) = \text{sat}(F') + |C^*|$.

The following result was shown by Crowston et al. [15]. The proof is nontrivial and consists of a deterministic algorithm for finding the corresponding truth assignment and a detailed combinatorial analysis of the algorithm.

▶ **Theorem 33.** *Let F be an expending 2-satisfiable CNF formula with n variables and m clauses. Then* $\mathrm{sat}(F) \geq \hat{\phi}m + n(2-3\hat{\phi})/2$.

Lemma 32 and Theorem 33 imply the following:

▶ **Theorem 34.** *Let F be a 2-satisfiable CNF formula and let V and C be its sets of variables and clauses. There exists a subset $C^* \subseteq C$ that can be found in polynomial time, such that the formula F' with clauses $C \setminus C^*$ and variables $V \setminus V^*$, where $V^* = \mathrm{var}(C^*)$, is expanding. Moreover, we have*

$$\mathrm{sat}(F) \geq \hat{\phi}m + (1-\hat{\phi})m^* + (n-n^*)(2-3\hat{\phi})/2,$$

where $m = |C|$, $m^* = |C^*|$, $n = |V|$ and $n^* = |V^*|$.

Let us turn now to 3-satisfiable CNF formulas. If F is 3-satisfiable then it is not hard to check that the forbidden sets of clauses are pairs of the form $\{x\}, \{\bar{x}\}$ and triplets of the form $\{x\}, \{y\}, \{\bar{x}, \bar{y}\}$ or $\{x\}, \{\bar{x}, y\}, \{\bar{x}, \bar{y}\}$, as well as any triplets that can be derived from these by switching positive literals with negative literals.

Lieberherr and Specker [39] and, later, Yannakakis [51] proved the following: if F is 3-satisfiable then $\mathrm{sat}(F) \geq \frac{2}{3}w(\mathcal{C}(F))$. This bound is also asymptotically tight. Yannakakis [51] gave a probabilistic proof which is similar to his proof for 2-satisfiable formulas, but requires consideration of several cases and, thus, not as short as for 2-satisfiable formulas. For details of his proof, see, e.g., Gutin, Jones and Yeo [24] and Jukna [33] (Theorem 20.6). Yannakakis's approach was extended by Gutin, Jones and Yeo [24] to prove the following theorem using a quite complicated probabilistic distribution for a random truth assignment.

▶ **Theorem 35.** *Let F be an expanding 3-satisfiable CNF formula with n variables and m clauses. Then $\mathrm{sat}(F) \geq \frac{2}{3}m + \rho n$, where $\rho (> 0.0019)$ is a constant.*

This theorem and Lemma 32 imply the following:

▶ **Theorem 36.** *Let F be a 3-satisfiable CNF formula and let V and C be its sets of variables and clauses. There exists a subset $C^* \subseteq C$ that can be found in polynomial time, such that the formula F' with clauses $C \setminus C^*$ and variables $V \setminus V^*$, where $V^* = \mathrm{var}(C^*)$, is expanding. Moreover, we have*

$$\mathrm{sat}(F) \geq \frac{2}{3}m + \frac{1}{3}m^* + \rho(n-n^*),$$

where $\rho (> 0.0019)$ is a constant, $m = |C|$, $m^* = |C^*|$, $n = |V|$ and $n^* = |V^*|$.

Using this theorem it is easy to obtain a linear-in-number-of-variables kernel for the following natural analog of Max-2S-Sat-A($\hat{\phi}m$), see [24] for details.

Max-3S-Sat-A($\frac{2}{3}m$)
Instance: A 3-satisfiable CNF formula F with m clauses and a nonnegative integer k.
Parameter: k.
Question: $\mathrm{sat}(F) \geq \frac{2}{3}m + k$?

Now let us consider the following important parameterization of r-Sat below the tight upper bound m:

r-SAT-B(m)
Instance: An r-CNF formula F with m clauses (every clause has at most r literals) and a nonnegative integer k.
Parameter: k. *Question:* $\mathrm{sat}(F) \geq m - k$?

Since MAX-r-SAT is NP-hard for each fixed $r \geq 3$, r-SAT-B(m) is not FPT unless P=NP. However, the situation changes for $r = 2$: Razgon and O'Sullivan [48] proved that 2-SAT-B(m) is FPT. The algorithm in [48] is of complexity $O(15^k km^3)$ and, thus, MAX-2-SAT-B(m) admits a kernel with at most $15^k k$ clauses. Raman et al. [47] and Cygan et al. [17] designed algorithms for 2-SAT-B(m) of runtime $9^k(km)^{O(1)}$ and $4^k(km)^{O(1)}$, respectively. Kratsch and Wahlström [37] proved that 2-SAT-B(m) admits a randomized kernel with a polynomial number of variables. The existence of a deterministic polynomial kernel 2-SAT-B(m) is an open problem.

7 Ordering CSPs

In this section we will discuss recent results in the area of *Ordering Constraint Satisfaction Problems (Ordering CSPs)* parameterized above average. Ordering CSPs include several well-known problems such as BETWEENNESS, CIRCULAR ORDERING and ACYCLIC SUBDIGRAPH (which is equivalent to 2-LINEAR ORDERING). These three problems have applications in circuit design and computational biology [10, 46], in qualitative spatial reasoning [32], and in economics [49], respectively. Our main interest are Ordering CSPs parameterized above average, Ordering CSPs-AA.

2-LINEAR ORDERING-AA was already considered in Subsection 3. In the next subsection, we give some basic definitions and results on Ordering CSPs. In Subsection 7.2, we give a proof scheme that BETWEENNESS-AA is fixed-parameter tractable. The proof uses SABEM supplemented by additional approaches. In Subsection 7.3, we discuss how to combine fixed-parameter tractability of 2-LINEAR ORDERING-AA and BETWEENNESS-AA to show that 3-LINEAR ORDERING-AA is fixed-parameter tractable. Finally, in Subsection 7.4, we briefly discuss the recent paper of Makarychev, Makarychev and Zhou [43], where it was proved that any Ordering CSP-AA is fixed-parameter tractable. Moreover, the authors of [43] extended their result to a linear programming generalization of Ordering CSPs-AA.

7.1 Basic Definitions and Results

Let us define Ordering CSPs of arity 3. The reader can easily generalize it to any arity $r \geq 2$ and we will do it below for LINEAR ORDERING of arity r. Let V be a set of n variables and let

$$\Pi \subseteq \mathcal{S}_3 = \{(123), (132), (213), (231), (312), (321)\}$$

be arbitrary. A *constraint set over* V is a multiset $\mathcal{C}$ of *constraints*, which are permutations of three distinct elements of V. A bijection $\alpha : V \to [n]$ is called an *ordering* of V. For an ordering $\alpha : V \to [n]$, a constraint $(v_1, v_2, v_3) \in \mathcal{C}$ is Π-*satisfied by* α if there is a permutation $\pi \in \Pi$ such that $\alpha(v_{\pi(1)}) < \alpha(v_{\pi(2)}) < \alpha(v_{\pi(3)})$. Thus, given Π the problem Π-CSP, is the problem of deciding if there exists an ordering of V that Π-satisfies all the constraints. Every such problem is called an Ordering CSP of arity 3. We will consider the maximization version of these problems, denoted by MAX-Π-CSP, parameterized above the average number of constraints satisfied by a random ordering of V (which can be shown to be a tight bound).

Table 1 Ordering CSPs of arity 3 (after symmetry considerations).

$\Pi \subseteq \mathcal{S}_3$	Name	Complexity
$\Pi_0 = \{(123)\}$	LINEAR ORDERING-3	polynomial
$\Pi_1 = \{(123), (132)\}$		polynomial
$\Pi_2 = \{(123), (213), (231)\}$		polynomial
$\Pi_3 = \{(132), (231), (312), (321)\}$		polynomial
$\Pi_4 = \{(123), (231)\}$		NP-comp.
$\Pi_5 = \{(123), (321)\}$	BETWEENNESS	NP-comp.
$\Pi_6 = \{(123), (132), (231)\}$		NP-comp.
$\Pi_7 = \{(123), (231), (312)\}$	CIRCULAR ORDERING	NP-comp.
$\Pi_8 = \mathcal{S}_3 \setminus \{(123), (231)\}$		NP-comp.
$\Pi_9 = \mathcal{S}_3 \setminus \{(123), (321)\}$	NON-BETWEENNESS	NP-comp.
$\Pi_{10} = \mathcal{S}_3 \setminus \{(123)\}$		NP-comp.

Guttmann and Maucher [30] showed that there are in fact only 13 distinct Π-CSP's of arity 3 up to symmetry, of which 11 are nontrivial. They are listed in Table 1 together with their complexity. Note that if $\Pi = \{(123), (321)\}$ then we obtain the BETWEENNESS problem and if $\Pi = \{(123)\}$ then we obtain 3-LINEAR ORDERING.

Gutin et al. [23] proved that all 11 nontrivial MAX-Π-CSP problems are NP-hard (even though four of the Π-CSP are polynomial).

Now observe that given a variable set V and a constraint multiset $\mathcal{C}$ over V, for a random ordering α of V, the probability of a constraint in $\mathcal{C}$ being Π-satisfied by α equals $\frac{|\Pi|}{6}$. Hence, the expected number of satisfied constraints from $\mathcal{C}$ is $\frac{|\Pi|}{6}|\mathcal{C}|$, and thus there is an ordering α of V satisfying at least $\frac{|\Pi|}{6}|\mathcal{C}|$ constraints (and this bound is tight). A derandomization argument leads to $\frac{|\Pi_i|}{6}$-approximation algorithms for the problems MAX-Π_i-CSP [8]. No better constant factor approximation is possible assuming the Unique Games Conjecture [8].

We will study the parameterization of MAX-Π_i-CSP above tight lower bound:

Π-ABOVE AVERAGE (Π-AA)

Input:	A finite set V of variables, a multiset $\mathcal{C}$ of ordered triples of distinct variables from V and an integer $\kappa \geq 0$.
Parameter:	κ.
Question:	Is there an ordering α of V such that at least $\frac{|\Pi|}{6}|\mathcal{C}| + \kappa$ constraints of $\mathcal{C}$ are Π-satisfied by α?

In [23] it is shown that all 11 nontrivial Π-CSP-AA problems admit kernels with $O(\kappa^2)$ variables. This is shown by first reducing them to 3-LINEAR ORDERING-AA (or 2-LINEAR ORDERING-AA), and then finding a kernel for this problem, which is transformed back to the original problem. The first transformation is easy due to the following:

▶ **Proposition 37** ([23]). *Let Π be a subset of $\mathcal{S}_3$ such that $\Pi \notin \{\emptyset, \mathcal{S}_3\}$. There is a polynomial time transformation f from Π-AA to* 3-LINEAR ORDERING-AA *such that an instance $(V, \mathcal{C}, k)$ of* Π-AA *is a* YES*-instance if and only if $(V, \mathcal{C}_0, k) = f(V, \mathcal{C}, k)$ is a* YES*-instance of* 3-LINEAR ORDERING-AA.

Proof. From an instance $(V, \mathcal{C}, k)$ of Π-AA, construct an instance $(V, \mathcal{C}_0, k)$ of 3-LINEAR

Ordering-AA as follows. For each triple $(v_1, v_2, v_3) \in \mathcal{C}$, add $|\Pi|$ triples $(v_{\pi(1)}, v_{\pi(2)}, v_{\pi(3)})$, $\pi \in \Pi$, to $\mathcal{C}_0$.

Observe that a triple $(v_1, v_2, v_3) \in \mathcal{C}$ is Π-satisfied if and only if exactly one of the triples $(v_{\pi(1)}, v_{\pi(2)}, v_{\pi(3)})$, $\pi \in \Pi$, is satisfied by 3-Linear Ordering. Thus, $\frac{|\Pi|}{6}|\mathcal{C}| + k$ constraints from $\mathcal{C}$ are Π-satisfied if and only if the same number of constraints from $\mathcal{C}_0$ are satisfied by 3-Linear Ordering. It remains to observe that $\frac{|\Pi|}{6}|\mathcal{C}| + k = \frac{1}{6}|\mathcal{C}_0| + k$ as $|\mathcal{C}_0| = |\Pi| \cdot |\mathcal{C}|$. ◀

r-Linear Ordering ($r \geq 2$) can be defined as follows. An instance of such a problem consists of a set of variables V and a multiset of constraints, which are ordered r-tuples of distinct variables of V (note that the same set of r variables may appear in several different constraints). The objective is to find an ordering α of V that maximizes the number of constraints whose order in α follows that of the constraint (we say that these constraints are satisfied). It is well-known that 2-Linear Ordering is NP-hard (it follows immediately from the fact proved by Karp [34] that the feedback arc set problem is NP-hard). It is easy to extend this hardness result to all r-Linear Ordering problems (for each fixed $r \geq 2$). Note that in r-Linear Ordering Above Average (r-Linear Ordering-AA), given a multiset $\mathcal{C}$ of constraints over V we are to decide whether there is an ordering of V that satisfies at least $|\mathcal{C}|/r! + \kappa$ constraints.

7.2 Betweenness-AA

Let $V = \{v_1, \ldots, v_n\}$ be a set of variables and let $\mathcal{C}$ be a multiset of m *betweenness* constraints of the form $(v_i, \{v_j, v_k\})$. For an ordering $\alpha :\ V \to [n]$, a constraint $(v_i, \{v_j, v_k\})$ *is satisfied* if either $\alpha(v_j) < \alpha(v_i) < \alpha(v_k)$ or $\alpha(v_k) < \alpha(v_i) < \alpha(v_j)$. In the Betweenness problem, we are asked to find an ordering α satisfying the maximum number of constraints in $\mathcal{C}$. Betweenness is NP-hard as even the problem of deciding whether all betweenness constraints in $\mathcal{C}$ can be satisfied by an ordering α is NP-complete [46].

Let $\alpha :\ V \to [n]$ be a random ordering and observe that the probability of a constraint in $\mathcal{C}$ to be satisfied is 1/3. Thus, the expected number of satisfied constraints is $m/3$. A triple of betweenness constraints of the form $(v, \{u, w\}), (u, \{v, w\}), (w, \{v, u\})$ is called a *complete triple.* Instances of Betweenness consisting of complete triples demonstrate that $m/3$ is a tight lower bound on the maximum number of constraints satisfied by an ordering α. Thus, the following parameterization is of interest:

Betweenness Above Average (Betweenness-AA)
Instance: A multiset $\mathcal{C}$ of m betweenness constraints over variables V and an integer $\kappa \geq 0$.
Parameter: The integer κ.
Question: Is there an ordering $\alpha : V \to [n]$ that satisfies at least $m/3 + \kappa$ constraints from $\mathcal{C}$?

In order to simplify instances of Betweenness-AA we introduce the following reduction rule.

▶ **Reduction Rule 4.** *If $\mathcal{C}$ has a complete triple, delete it from $\mathcal{C}$. Delete from V all variables that appear only in the deleted triple.*

Benny Chor's question (see [44, p. 43]) to determine the parameterized complexity of Betweenness-AA was solved by Gutin, Kim, Mnich and Yeo [25] who proved that Betweenness-AA admits a kernel with $O(\kappa^2)$ variables and constraints (in fact, [25] considers only the case when $\mathcal{C}$ is a set, not a multiset, but the proof for the general case is the same [23]). Below we briefly describe the proof in [25].

Suppose we define a random variable $X(\alpha)$ just as we did for 2-Linear Ordering. However such a variable is not symmetric and therefore we would need to use Lemma 7 on $X(\alpha)$. The problem is that α is a permutation and in Lemma 7 we are looking at polynomials, $f = f(x_1, x_2 \dots, x_n)$, over variables $x_1, \dots, x_n$ each with domain $\{-1, 1\}$. In order to get around this problem the authors of [25] considered a different random variable $g(Z)$, which they defined as follows.

Let $Z = (z_1, z_2, \dots, z_{2n})$ be a set of $2n$ variables with domain $\{-1, 1\}$. These $2n$ variables correspond to n variables $z_1^*, z_2^*, \dots, z_n^*$ such that z_{2i-1} and z_{2i} form the binary representation of z_i^*. That is, z_i^* is 0, 1, 2 or 3 depending on the value of $(z_{2i-1}, z_{2i}) \in \{(-1,-1), (-1,1), (1,-1), (1,1)\}$. An ordering: $\alpha : V \to [n]$ complies with Z if for every $\alpha(i) < \alpha(j)$ we have $z_i^* \leq z_j^*$. We now define the value of $g(Z)$ as the average number of constraints satisfied over all orderings which comply with Z. Let $f(Z) = g(Z) - m/3$, and by Lemma 38 we can now use Lemma 7 on $f(Z)$ as it is a polynomial over variables whose domain is $\{-1, 1\}$. We consider variables z_i as independent uniformly distributed random variables and then $f(Z)$ is also a random variable. In [25] it is shown that the following holds if Reduction Rule 4 has been exhaustively applied.

▶ **Lemma 38.** *The random variable $f(Z)$ can be expressed as a polynomial of degree 6. We have $\mathbb{E}[f(Z)] = 0$. Finally, if $f(Z) \geq \kappa$ for some $Z \in \{-1, 1\}^{2n}$ then the corresponding instance of* Betweenness-AA *is a* Yes-*instance.*

▶ **Lemma 39** ([23]). *For an irreducible (by Reduction Rule 4) instance we have $\mathbb{E}[f(Z)^2] \geq \frac{11}{768}m$.*

▶ **Theorem 40** ([23]). Betweenness-AA *has a kernel of size $O(\kappa^2)$.*

Proof. Let $(V, \mathcal{C})$ be an instance of Betweenness-AA. We can obtain an irreducible instance $(V', \mathcal{C}')$ such that $(V, \mathcal{C})$ is a Yes-instance if and only if $(V', \mathcal{C}')$ is a Yes-instance in polynomial time. Let $m' = |\mathcal{C}'|$ and let $f(Z)$ be the random variable defined above. Then $f(Z)$ is expressible as a polynomial of degree 6 by Lemma 38; hence it follows from Lemma 5 that $\mathbb{E}[f(Z)^4] \leq 2^{36}\mathbb{E}[f(Z)^2]^2$. Consequently, $f(Z)$ satisfies the conditions of Lemma 4, from which we conclude that $\mathbb{P}\left(f(Z) > \frac{1}{4 \cdot 2^{18}}\sqrt{\frac{11}{768}m'}\right) > 0$, by Lemma 39. Therefore, by Lemma 38, if $\frac{1}{4 \cdot 2^{18}}\sqrt{\frac{11}{768}m'} \geq \kappa$ then $(V', \mathcal{C}')$ is a Yes-instance for Betweenness-AA. Otherwise, we have $m' = O(\kappa^2)$. This concludes the proof of the theorem. ◀

By deleting variables not appearing in any constraint, we obtain a kernel with $O(\kappa^2)$ constraints and variables.

7.3 3-Linear Ordering-AA

In this subsection, we will give a short overview of the proof in [23] that 3-Linear Ordering has a kernel with at most $O(\kappa^2)$ variables and constraints.

Unfortunately, approaches which we used for the 2-Linear Ordering-AA problem and the Betweenness-AA problem do not work for this problem. In fact, if we wanted to remove subsets of constraints where only the average number of constraints can be satisfied such that after these removals we are guaranteed to have more than the average number of constraints satisfied, then, in general case, an *infinite* number of reduction rules would be needed [23].

However, we can reduce an instance of 3-Linear Ordering-AA to instances of Betweenness-AA *and* 2-Linear Ordering-AA as follows. With an instance $(V, \mathcal{C})$ of 3-Linear Ordering-AA, we associate an instance $(V, \mathcal{B})$ of Betweenness-AA and two instances (V, A') and (V, A'') of 2-Linear Ordering-AA such that if $C_p = (u, v, w) \in \mathcal{C}$, then add $B_p = (v, \{u, w\})$ to $\mathcal{B}$, $a'_p = (u, v)$ to A', and $a''_p = (v, w)$ to A''.

Let α be an ordering of V and let $\mathsf{dev}(V, \mathcal{C}, \alpha)$ denote the number of constraints satisfied by α minus the average number of satisfied constraints in $(V, \mathcal{C})$, where $(V, \mathcal{C})$ is an instance of 3-Linear Ordering-AA, Betweenness-AA or 2-Linear Ordering-AA.

▶ **Lemma 41** ([23]). *Let (V, C, κ) be an instance of* 3-Linear Ordering-AA *and let α be an ordering of V. Then*

$$\mathsf{dev}(V, \mathcal{C}, \alpha) = \frac{1}{2}\left[\mathsf{dev}(V, A', \alpha) + \mathsf{dev}(V, A'', \alpha) + \mathsf{dev}(V, \mathcal{B}, \alpha)\right].$$

Therefore, we want to find an ordering satisfying as many constraints as possible from both of our new type of instances (note that we need to use the same ordering for all the problems).

Suppose we have a No-instance of 3-Linear Ordering-AA. As above, we replace it by three instances of Betweenness-AA and 2-Linear Ordering-AA. Now we apply the reduction rules for Betweenness-AA and 2-Linear Ordering-AA introduced above as well as the proof techniques described in the previous sections in order to show that the total number of variables and constraints left in any of our instances is bounded by $O(\kappa^2)$. We then transform these reduced instances back into an instance of 3-Linear Ordering-AA as follows. If $\{v, \{u, w\}\}$ is a Betweenness constraint then we add the 3-Linear Ordering-AA constraints (u, v, w) and (w, v, u) and if (u, v) is an 2-Linear Ordering-AA constraint then we add the 3-Linear Ordering-AA constraints (u, v, w), (u, w, v) and (w, u, v) (for any $w \in V$). As a result, we obtain a kernel of 3-Linear Ordering-AA with at most $O(\kappa^2)$ variables and constraints.

This result has been partially improved by Kim and Williams [36] who showed that 3-Linear Ordering-AA has a kernel with at most $O(\kappa)$ variables.

7.4 Ordering CSPs AA

Recall that an Ordering CSP of arity r is defined by a set $V = \{x_1, \ldots, x_n\}$ of variables and set of constraints. Each constraint is a disjunction of clauses of the form $x_{i_1} < x_{i_2} < \cdots < x_{i_r}$. A linear ordering α of V satisfies such a constraint if one of the clauses in the disjunction agrees with α.

Gutin, Iersel, Mnich and Yeo [23] conjectured that all Ordering CSPs parameterized above average are fixed-parameter tractable. One of the difficulties in proving this conjecture is that, as we mentioned in the previous subsection, we may need an infinite number of reduction rules. The approach of the previous section will not work as it is designed for the case when some Ordering CSPs of the same arity have already proved to be fixed-parameter tractable. Recently, Makarychev, Makarychev and Zhou [43] proved the conjecture. Their proof uses SABEM together the idea to define variables x_i not on a discrete domain, but on the continuous interval $[-1, 1]$. Such a domain allows to order all the variables with ties being almost impossible.

Makarychev, Makarychev and Zhou [43] also use the Efron-Stein decomposition instead of the (standard) Fourier Analysis on $[-1, +1]^n$ since "we have no control over the Fourier coefficients of the functions we need to analyze." For terminology and results on the Efron-Stein decomposition, see [43] and for a more detailed account [45]. Here we will only give some very basic definitions. Let (Ω, μ) be a probability space and consider the product probability space (Ω^n, μ^n). Let $f : \Omega^n \to \mathbb{R}$ be a function (random variable). Informally, the Efron-Stein decomposition of f is $f = \sum_{S\subseteq[n]} f_S$, where f_S depends only on variables x_i, $i \in S$. The functions f_S have some very useful properties such as $\mathbb{E}[f_S f_T] = 0$ if $S \neq T$ (this implies that the variance of f equals the sum of the variances of f_S in the decomposition). To use the Efron-Stein decomposition for SABEM, Makarychev, Makarychev and Zhou [43] obtained the following Hypercontractive Inequality for functions defined on arbitrary product probability spaces:

▶ **Theorem 42.** *Consider $f \in L_2(\Omega^n, \mu^n)$. Let $f = \sum_{S\subseteq[n]} f_S$ be the Efron-Stein decomposition of f and let $d = \max\{t :\ f_S \neq 0$ and $|S| = d\}$. Assume that for every S_1, S_2, S_3, S_4,*

$$\mathbb{E}[f_{S_1} f_{S_2} f_{S_3} f_{S_4}] \leq C(\mathbb{E}[f^2_{S_1}]\mathbb{E}[f^2_{S_1}]\mathbb{E}[f^2_{S_1}]\mathbb{E}[f^2_{S_1}])^{1/2}\,. \tag{3}$$

Then

$$\mathbb{E}[f(X_1, \dots, X_n)^4] \leq 81^d C\mathbb{E}[f(X_1, \dots, X_n)^2]^2\,. \tag{4}$$

Note that Condition (3) is usually much easier to verify than Inequality (4 [43].

The idea to use a continuous domain allows one to define various systems of linear inequalities rather than just those of the form $x_{i_1} < x_{i_2} < \cdots < x_{i_r}$. For example, we can require that x_4 is to the left of the average of x_1, x_2 and x_3, which corresponds to the system $3x_4 - x_1 - x_2 - x_3 < 0$. Makarychev, Makarychev and Zhou [43] define the following generalization of Ordering CSPs. An (r, b)-LP CSP is defined by a set $V = \{x_1, \dots, x_n\}$ of variables taking values in $[-1, 1]$ and set of constraints. Each constraint is a disjunction of clauses of the form $Ax < c$, where A is a matrix with integral entries in the range $[-b, b]$ and there are at most r non-zero columns in A, and c is a vector with integral entries in the range $[-b, b]$. The aim is to assign distinct real values to variables x_i so as to maximize the number of satisfied constraints. Makarychev, Makarychev and Zhou [43] proved that every (r, b)-LP CSP above average is also fixed-parameter tractable.

8 Two Open Problems

Many results described in the previous sections were obtained in order to solve open problems. Two problems stated a while ago remain unsolved. The first is whether r-SAT-B(m), usually called ALMOST 2-SAT, admits a (deterministic) polynomial kernel. It seems it widely believed to be the case, but no proof is obtained. The second is whether MAX-2-LIN-AA admits a polynomial kernel (in the number of constraints). For the second problem, the possible answer is unclear.

References

1 N. Alon, Voting paradoxes and digraphs realizations, Advances in Applied Math. 29:126–135, 2002.

2 N. Alon, G. Gutin, E. J. Kim, S. Szeider, and A. Yeo, Solving MAX-r-SAT above a tight lower bound. Algorithmica 61(3):638-655, 2011.

3 N. Alon, G. Gutin and M. Krivelevich. Algorithms with large domination ratio. J. Algorithms 50:118–131, 2004.

4 H. L. Bodlaender, R. G. Downey, M. R. Fellows, and D. Hermelin, On problems without polynomial kernels. J. Comput. Syst. Sci. 75(8):423–434, 2009.

5 H. L. Bodlaender, S. Thomassé, and A. Yeo, Kernel bounds for disjoint cycles and disjoint paths. Theor. Comput. Sci. 412(35): 4570–4578, 2011.

6 A. Bonami, Étude des coefficients de Fourier des fonctions de $L^p(G)$. Ann. Inst. Fourier, 20(2):335–402, 1970.

7 L. Cai and J. Chen, On fixed-parameter tractability and approximation of NP optimization problems, J. Comput. Syst. Sci. 54:465–474, 1997.

8 M. Charikar, V. Guruswami, and R. Manokaran, Every permutation CSP of arity 3 is approximation resistant. *Proc. Computational Complexity 2009*, 62–73.

9 X. Chen and Y. Zhou, Parameterized Algorithms for Constraint Satisfaction Problems Above Average with Global Cardinality Constraints. Manuscript

10 B. Chor and M. Sudan. A geometric approach to betweenness. SIAM J. Discrete Math., ll(4):511-523, 1998.

11 R. Crowston, M. Fellows, G. Gutin, M. Jones, F. Rosamond, S. Thomassé and A. Yeo, Satisfying more than half of a system of linear equations over GF(2): A multivariate approach. J. Comput. Syst. Sci. 80(4): 687-696 (2014).

12 R. Crowston, G. Gutin, M. Jones, E. J. Kim, and I. Ruzsa. Systems of linear equations over $\mathbb{F}_2$ and problems parameterized above average. *Proc. SWAT 2010*, Lect. Notes Comput. Sci. 6139: 164–175, 2010.

13 R . Crowston, G. Gutin, M. Jones, V. Raman, and S. Saurabh, Parameterized Complexity of MaxSat Above Average. Theor. Comput. Sci. 511:77-84, 2013.

14 R. Crowston, G. Gutin, M. Jones and A. Yeo, Parameterized Complexity of Satisfying Almost All Linear Equations over $\mathbb{F}_2$, Theory Comput. Syst. 52(4):719–728, 2013.

15 R. Crowston, G. Gutin, M. Jones, and A. Yeo, A new lower bound on the maximum number of satisfied clauses in Max-SAT and its algorithmic applications. Algorithmica 64(1): 56-68 (2012).

16 M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. Parameterized Algorithms. Springer-Verlag, 2015.

17 M. Cygan, M. Pilipczuk, M. Pilipczuk, and J.O. Wojtaszczyk, On Multiway Cut parameterized above lower bounds. ACM Trans. Comput. Theory 5(1):1–11, 2013.

18 R. G. Downey and M. R. Fellows, Fundamentals of Parameterized Complexity. Springer, 2013.

19 H. Fernau, F.V. Fomin, D. Lokshtanov, D. Raible, S. Saurabh, and Y. Villanger, Kernel(s) for problems with no kernel: On out-trees with many leaves. *Proc. STACS 2009*, 421–432.

20 H. Fleischner, O. Kullmann and S. Szeider, Polynomial-time recognition of minimal unsatisfiable formulas with fixed clause-variable difference. *Theoret. Comput. Sci.*, 289(1):503–516, 2002.

21 V. Guruswami, J. Håstad, R. Manokaran, P. Raghavendra, and M. Charikar, Beating the random ordering is hard: Every ordering CSP is approximation resistant. SIAM J. Comput. 40(3): 878–914 (2011).

22 V. Guruswami, R. Manokaran, and P. Raghavendra, Beating the random ordering is hard: Inapproximability of maximum acyclic subgraph. *Proc. FOCS 2008*, 573–582.

23 G. Gutin, L. van Iersel, M. Mnich, and A. Yeo, Every ternary permutation constraint satisfaction problem parameterized above average has a kernel with a quadratic number of variables, J. Comput. Syst. Sci., in press, `doi:10.1016/j.jcss.2011.01.004`.

24 G. Gutin, M. Jones, D. Scheder, and A. Yeo, A New Bound for 3-Satisfiable MaxSat and its Algorithmic Application. Inf. Comput. 231:117-124, 2013.

25 G. Gutin, E. J. Kim, M. Mnich, and A. Yeo. Betweenness parameterized above tight lower bound. J. Comput. Syst. Sci., 76:872–878, 2010.

26 G. Gutin, E. J. Kim, S. Szeider, and A. Yeo. A probabilistic approach to problems parameterized above tight lower bound. J. Comput. Syst. Sci. 77: 422–429, 2011.

27 G. Gutin and A. Yeo, Constraint satisfaction problems parameterized above or below tight bounds: a survey. In Fellows Festschrift, Lect. Notes Comput. Sci. 7370 (2012), 257–286.

28 G. Gutin and A. Yeo, Hypercontractive inequality for pseudo-boolean functions of bounded Fourier width. Discrete Appl. Math. 160 (2012), 2323–2328.

29 G. Gutin and A. Yeo, Note on maximal bisection above tight lower bound. Information Proc. Letters. 110 (2010) 966–969.

30 W. Guttmann and M. Maucher. Variations on an ordering theme with constraints. *Proc. 4th IFIP International Conference on Theoretical Computer Science-TCS 2006*, pp. 77–90, Springer.

31 J. Håstad, Some optimal inapproximability results. *J. ACM* 48: 798–859, 2001.

32 A. Isli and A. G. Cohn. A new approach to cyclic ordering of 2D orientations using ternary relation algebras. Artif. Intelligence, 122(1-2):137–187, 2000.

33 S. Jukna, Extremal Combinatorics With Applications in Computer Science, Springer-Verlag, 2001.

34 R. M. Karp, Reducibility among combinatorial problems, *Proc. Complexity of Computer Computations*, Plenum Press, 1972.

35 S. Khot, On the power of unique 2-prover 1-round games. *Proc. STOC 2002*, 767–775.

36 E. J. Kim and R. Williams, Improved parameterized algorithms for constraint satisfaction. *Proc. IPEC 2011*, Lect. Notes Comput. Sci. 7112 (2011) 118–131.

37 S. Kratsch and M. Wahlström, Representative Sets and Irrelevant Vertices: New Tools for Kernelization. In 54th Annual Symposium on Foundations of Computer Science (FOCS), 450–459, 2012.

38 K. J. Lieberherr and E. Specker, Complexity of partial satisfaction. J. ACM, 28(2):411-421, 1981.

39 K. J. Lieberherr and E. Specker, Complexity of partial satisfaction, II. Tech. Report 293, Dept. of EECS, Princeton Univ., 1982.

40 D. Lokshtanov, New Methods in Parameterized Algorithms and Complexity, PhD thesis, Bergen, 2009.

41 M. Mahajan and V. Raman. Parameterizing above guaranteed values: MaxSat and Max-Cut. J. Algorithms, 31(2):335–354, 1999. Preliminary version in Electr. Colloq. Comput. Complex. (ECCC), TR-97-033, 1997.

42 M. Mahajan, V. Raman, and S. Sikdar. Parameterizing above or below guaranteed values. J. Computer System Sciences, 75(2):137–153, 2009. Preliminary version in *Proc. IWPEC 2006*, Lect. Notes Comput. Sci. 4169: 38–49, 2006.

43 K. Makarychev, Y. Makarychev and Y. Zhou, Satisfiability of Ordering CSPs Above Average Is Fixed-Parameter Tractable. In FOCS 2015, 975–993.

44 R. Niedermeier. Invitation to Fixed-Parameter Algorithms. Oxford University Press, 2006.

45 R. O'Donnell, Analysis of Boolean Functions, Cambridge UP, 2014.

46 J. Opatrný, Total ordering problem. SIAM J. Comput., 8: 111–114, 1979.

47 V. Raman, M. S. Ramanujan and S. Saurabh, Paths, Flowers and Vertex Cover. *Proc. ESA 2011*, Lect. Notes Comput. Sci. 6942: 382–393, 2011.

48 I. Razgon and B. O'Sullivan. Almost 2-SAT is fixed-parameter tractable. J. Comput. Syst. Sci. 75(8):435–450, 2009.

49 G. Reinelt, The linear ordering problem: Algorithms and applications, Heldermann Verlag, 1985.

50 S. Szeider, Minimal unsatisfiable formulas with bounded clause-variable difference are fixed-parameter tractable. J. Comput. Syst. Sci., 69(4):656–674, 2004.

51 M. Yannakakis, On the approximation of maximum satisfiability. J. Algorithms, 17:475–502, 1994.

Counting Constraint Satisfaction Problems*

Mark Jerrum

School of Mathematical Sciences, Queen Mary, University of London, London, UK
`m.jerrum@qmul.ac.uk`

Abstract

This chapter surveys counting Constraint Satisfaction Problems (counting CSPs, or #CSPs) and their computational complexity. It aims to provide an introduction to the main concepts and techniques, and present a representative selection of results and open problems. It does not cover holants, which are the subject of a separate chapter.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Approximation algorithms, Computational complexity, Constraint satisfaction problems, Counting problems, Partition functions

Digital Object Identifier 10.4230/DFU.Vol7.15301.205

1 Introduction

In this chapter, we shall be working within the usual CSP framework, or natural extensions of it, but our interest will be in *counting* assignments that satisfy all the constraints, rather than just determining whether one exists. We can make a swift entry into the topic by minimally adapting the classical CSP framework. Having done this, we shall briefly discuss the limitations of this simple-minded approach, and how the model can be refined to encompass a wider range of situations.

A Constraint Satisfaction Problem (CSP) typically has a finite *domain* D, which we identify with $\{0,\dots,q-1\}$, or possibly $\{1,\dots,q\}$, for some positive integer q. Keeping close to classical decision CSPs, a *constraint language* Γ is a set of relations of various arities on D. Given a finite constraint language Γ, an *instance* of #CSP(Γ), the counting CSP with constraint language Γ, is specified by a set of *variables* $X=\{x_1,\dots,x_n\}$ and a set $C=\{(R_1,\mathbf{x}_1),\dots,(R_m,\mathbf{x}_m)\}$ of *constraints*. Each constraint is a pair: a relation $R_i\in\Gamma$ of some arity k, and a *scope* $\mathbf{x}_i=(x_{s_{i,1}},\dots,x_{s_{i,k}})$, which is a k-tuple of variables from X. An *assignment* σ is a mapping from X to D. The assignment σ is said to be *satisfying*, or to *satisfy the instance* (X,C), if the scope of every constraint is mapped to a tuple that is in the corresponding relation, that is to say, σ satisfies the formula $\bigwedge_{i=1}^m(\sigma(\mathbf{x}_i)\in R_i)$, where $\sigma(\mathbf{x}_i)=(\sigma(x_{s_{i,1}}),\dots,\sigma(x_{s_{i,k}}))$.

Given an instance (X,C) of a CSP with constraint language Γ, the *decision problem* CSP(Γ) asks us to determine whether any assignment σ exists that satisfies (X,C). The *counting problem* #CSP(Γ) asks us to determine the *number* of assignments that satisfy (X,C).

By varying the constraint language Γ we obtain infinite families of decision problems CSP(Γ) and counting problems #CSP(Γ). We wish to classify these problems according to their computational complexity. A simple observation is that the counting CSP cannot be easier than its decision twin, but can be harder. For example, consider the binary relation on

* This work was partially supported by the EPSRC grant EP/N004221/1.

$\{0,1\}$ defined by NAND $= \{(0,0),(0,1),(1,0)\}$, and the constraint language $\Gamma = \{\text{NAND}\}$ consisting of this single relation. Since the relation NAND is symmetric, we can view an instance of CSP(Γ) or #CSP(Γ) as an undirected graph G whose vertices represent variables and whose edges represent scopes. The decision problem CSP(Γ) asks whether G contains an independent set; this decision problem is trivial, since every graph has the empty set of vertices as an independent set. In contrast, #CSP(Γ) is the problem of counting independent sets in G, which is #P-complete. As we shall see, even estimating the number of independent sets in G within relative error $1 \pm \varepsilon$ is computational intractable, assuming $\mathsf{RP} \neq \mathsf{NP}$.

Many of the motivating examples for counting CSPs come from statistical physics. Variables represent states of atoms, say, and constraints represent interactions between pairs, triples, etc., of atoms. These interactions are not usually "hard" constraints that can be modelled by relations. Because of this, even more than in the case of valued CSPs (VCSPs), there is a strong motivation to extend the above setting for counting CSPs to include weights. With this in mind, we replace the set Γ of relations by a set of functions $\mathcal{F}$. Each function $f \in \mathcal{F}$ is of the form $f : D^k \to \mathbf{R}$, where $\mathbf{R}$ is a commutative semiring: common choices for $\mathbf{R}$ are $\mathbb{C}$, $\mathbb{R}$, $\mathbb{R}_{\geq 0}$, or some computationally tractable subrings of these, such as $\mathbb{Q}$ or the algebraic reals. An instance of a (weighted) counting CSP is specified by a set of variables X and a collection of functional "constraints" $\big\{(f_1, \mathbf{x}_1), \ldots, (f_m, \mathbf{x}_m)\big\}$, where $f_i \in \mathcal{F}$ is a function of arity k, and $\mathbf{x}_i = (x_{s_{i,1}}, \ldots, x_{s_{i,k}})$ is a scope. Then the required output for the counting CSP is the quantity

$$Z(X, C) = \sum_{\sigma : X \to D} \prod_{i=1}^{m} f_i(\sigma(\mathbf{x}_i)).$$

Relative to usual decision CSPs we have replaced relations by functions, conjunction by multiplication and existential quantification by summation. This is a strict extension of classical decision CSPs, which we can recover by setting $\mathbf{R}$ to be the semiring $(\{0,1\}, \vee, \wedge)$. Even VCSPs can be viewed in this light by taking $\mathbf{R} = (\mathbb{R}_{\geq 0}, \min, +)$. However, although some techniques can be traded between these CSP variants, they maintain strong identities of their own.

There is already an extensive body of work on counting CSPs, and it is not possible to cover all of it here. The most significant omission is surely holants, which are, roughly speaking, read-twice counting CSPs. Holants generalise counting CSPs; for example, the generating function of perfect matchings in a graph (the dimer model in statistical physics) can be expressed as a holant, but not as a counting CSP. Holants have a special flavour and an extensive literature of their own, and are the subject of a separate chapter in this volume.

In deciding how to organise the material in this survey, a number of different attributes can be taken into account. CSPs on a two-element domain have a special prominence and are often easier to deal with. The same is true of conservative CSPs in which unary functions are given free. Bijunctive CSPs (all functions in $\mathcal{F}$ have arity at most 2), and more specifically CSPs with one symmetric binary function are not only potentially easier to handle, but have a particular importance because of their connection with spin models in statistical physics. However, it seems to me that the clearest division in terms of the flavour of the results and the techniques employed is between exact and approximate computation. I have therefore decided to make this the high level split. Within these two parts, the material will be organised as far as is possible into special cases, such as Boolean, conservative, partition functions, etc.

2 Exact Computation

A combinatorial (i.e., unweighted) counting problem specifies a function $\Sigma^* \to \mathbb{N}$ that maps problem instances, encoded over an alphabet Σ, to natural numbers. A function $f : \Sigma^* \to \mathbb{N}$ is in the complexity class $\#\mathsf{P}$ if there exists a nondeterministic Turing machine M such that, for all $x \in \Sigma^*$, the number of accepting computations of M on input $x \in \Sigma^*$ is equal to $f(x)$. The relative complexity of counting problems is customarily investigated using Turing reductions. Thus, a problem f is said to be $\#\mathsf{P}$*-hard* if every problem $g \in \#\mathsf{P}$ is polynomial-time Turing reducible to f. If, in addition, $f \in \#\mathsf{P}$, then the problem f is said to be $\#\mathsf{P}$*-complete*. It is clear that $\#\mathrm{CSP}(\Gamma)$ is in $\#\mathsf{P}$ for any finite set of relations Γ, since we can construct a Turing machine M that, given an instance (X, C) of $\#\mathrm{CSP}(\Gamma)$, nondeterministically chooses an assignment $\sigma : X \to D$ to the variables X, and accepts if all constraints C are satisfied by σ. The complexity class $\#\mathsf{P}$ was introduced by Valiant, who also made the initial exploration of the phenomenon of $\#\mathsf{P}$-completeness [54].

To make sense of a counting CSP with real or complex weights as a computational problem, we need to restrict the real or complex numbers to some suitable subfield, say rational, algebraic or polynomial-time computable. A weighted counting CSP is no longer a member of $\#\mathsf{P}$, for the banal reason that it does not in general produce integer outputs. Nevertheless, at least if we restrict attention to finite sets of functions $\mathcal{F}$ taking rational or algebraic values, it will be the case that $\#\mathrm{CSP}(\mathcal{F}) \in \mathsf{FP}^{\#\mathsf{P}}$, i.e., that $\#\mathrm{CSP}(\mathcal{F})$ is solvable by a polynomial-time deterministic Turing machine with a $\#\mathsf{P}$-oracle. Also, we can still expect $\#\mathrm{CSP}(\mathcal{F})$ to be $\#\mathsf{P}$-hard in many cases, thus locating the computational complexity up to polynomial-time Turing reductions. For some comments on the complexity of counting problems with rational weights (in the context of the Tutte polynomial) see [37]. We will ignore the issue of representing real and complex numbers in the remainder of the survey.

2.1 Boolean #CSPs

Just as with decision CSPs, the first step historically in the exploration of counting CSPs was the resolution of the Boolean case. This was achieved by Creignou and Hermann [15]. It turns out to be helpful to identify the 2-element domain D with the 2-element field $\mathbb{F}_2$. Then we can say that a relation $R \subseteq \{0,1\}^k$ is *affine* if and only if it is the solution set to a system of linear equations over $\mathbb{F}_2$.

▶ **Theorem 1.** *Let* Γ *be a finite set of relations on the domain* $\{0,1\}$*. If every relation in* Γ *is affine then* $\#\mathrm{CSP}(\Gamma)$ *is in* FP*; otherwise* $\#\mathrm{CSP}(\Gamma)$ *is* $\#\mathsf{P}$*-complete.*

This result is pessimistic when compared to Schaefer's dichotomy for classical (decision) CSPs. Recall that a relation is bijunctive if it is equivalent to a conjunction of clauses with at most two literals, 0-valid (respectively, 1-valid) if it is empty or contains the all-0 (respectively, all-1) tuple, and Horn (respectively, dual-Horn) if it is equivalent to a conjunction of Horn (respectively, dual-Horn) clauses. Any of the conditions affine, bijunctive, 0/1-valid, Horn or dual-Horn is sufficient to ensure tractability of the decision problem. For counting, only affine will do. The constraint language $\Gamma = \{\mathrm{IMP}\}$ consisting solely of the implies relation $\mathrm{IMP} = \{(0,0),(0,1),(1,1)\}$ neatly illustrates the point: IMP is bijunctive, 0-valid, 1-valid, Horn and dual-Horn and yet $\#\mathrm{CSP}(\Gamma)$ is $\#\mathsf{P}$-complete, being essentially equivalent to counting downsets in a partial order [51].

In broad brushstrokes, Creignou and Hermann's proof of Theorem 1 runs as follows. Denote by OR the relation $\mathrm{OR} = \{(0,1),(1,0),(1,1)\}$. If Γ is affine then any instance of $\#\mathrm{CSP}(\Gamma)$ defines an affine relation. Thus the set of solutions to the instance (X, C) forms an affine subspace of $\mathbb{F}_2^X$, whose dimension d can be computed by standard linear algebra techniques. The required output for the instance is then 2^d. If Γ is not affine, then one of the relations NAND, OR or IMP can (in a suitable sense) be implemented in terms of relations in Γ. Since all of #CSP({NAND}), #CSP({OR}) and #CSP({IMP}) are #P-complete, this completes the proof. It is an interesting exercise to translate the proof into the language of clones and Post's lattice, and also an instructive one, as it hints at what survives of the universal algebra approach, and what does not, in the passage from decision to counting. We'll sketch how this works once suitable concepts and notation have been introduced.

The next step is to add positive real weights. So now $\mathcal{F}$ is a finite set of functions of the form $\{0,1\}^k \to \mathbb{R}_{\geq 0}$ (with the arity k possibly varying), and we are interested in the complexity of #CSP($\mathcal{F}$). A dichotomy similar to Theorem 1 continues to hold. Denote by $\mathcal{P}$ the set of all functions that can be expressed as a product of nullary and unary functions, binary equality functions and binary disequality functions; these functions are said to be of *product type*. Denote by $\mathcal{A}$ the set of functions whose support is an affine relation, and whose range is a subset of $\{0,b\}$ for some $b \in \mathbb{R}_{\geq 0}$; these functions are said to be *pure affine*. In other words, to get a pure affine function we interpret an affine relation as a 0,1-function, then multiply that function by a positive constant. Dyer, Goldberg and Jerrum [21] show that #CSP($\mathcal{F}$) is in FP if $\mathcal{F} \subset \mathcal{P}$ or $\mathcal{F} \subset \mathcal{A}$. In all other cases, #CSP($\mathcal{F}$) is #P-hard.

Notice that the addition of non-negative real weights did not significantly change the statement of the dichotomy, only its proof. The first indication that something new and interesting happens when we extend the domain to negative real numbers comes with the following example. Denote by $H_2 : \{0,1\}^2 \to \mathbb{R}$ the function defined by $H_2(x,y) = -1$ if $x = y = 1$ and $H_2(x,y) = +1$ otherwise. We can interpret #CSP($\{H_2\}$) as the problem of counting induced subgraphs of a graph G which have an even (or odd) number of edges. To see this, view an instance (X,C) of #CSP($\{H_2\}$) as an undirected graph G, with vertex set X and with edges determined by the scopes of the constraints in C. An assignment to variables in X can be interpreted as the indicator function of a vertex subset $U \subseteq V(G)$ of G. If the subgraph $G[U]$ induced by U has as even (respectively, odd) number of edges then it contributes an additive $+1$ (respectively, -1) to the solution of the instance (X,C). Given that the total number of induced subgraphs is $2^{|V(G)|}$, the solution to the counting CSP easily yields the number of induced subgraphs with an even (or odd) number of edges.

It transpires that #CSP($\{H_2\}$) $\in$ FP. As we saw, the required output for an instance (X,C) is a sum of $2^{|X|}$ terms, each of them ± 1. Letting $X = \{x_1,\dots,x_n\}$, consider the quadratic form over $\mathbb{F}_2$ defined by $Q(X) = \sum_{\{i,j\}\in S} x_i x_j$, where S is the set of all scopes of constraints in C. Note that $Q(X) = 0$ if the assignment to variables corresponds to a $+1$ term and $Q(X) = 1$ otherwise. So the number of positive terms in the sum is equal to the number of solutions to $Q(X) = 0$, and once we know the number of positive terms, we know the sum itself. Now any quadratic form over $\mathbb{F}_2$ is equivalent under linear substitutions of variables to a quadratic form over a possibly smaller number of variables, in canonical form [48, Thm 6.30]. This canonical form allows easy calculation of the number of solutions. This tractable example was first noted in the context of counting CSPs by Goldberg, Grohe, Jerrum and Thurley [35], and it, and others like it, substantially complicate the classification programme when negative or complex weights are involved.

Nevertheless, the challenges were incrementally overcome, and the Boolean #CSP dichotomy was extended to arbitrary real weights by Bulatov, Dyer, Goldberg, Jalsenius and Richerby [3], and then extended further to arbitrary complex weights by Cai, Lu and Xia [13]. Let $\mathcal{A}$ denote the set of complex functions $f(x_1,\dots,x_k)$ that are the product of a pure affine function (defined as above, but with $b \in \mathbb{C}$ now a complex number) and a certain rotation

$\omega(x_1,\dots,x_k)$, which takes values in the set of fourth roots of unity. Specifically, identify the Boolean domain with the field $\mathbb{F}_2$, and denote by $\mathbf{x}'$ the vector $(x_1,\dots,x_k,1) \in \mathbb{F}_2^{k+1}$ of arguments to f, extended to the right by the constant 1. Then there are vectors $\mathbf{a}_1,\dots,\mathbf{a}_m \in \mathbb{F}_2^{k+1}$, such that the value of the rotation $\omega(x_1,\dots,x_k)$ is given by $i^{L_1(\mathbf{x}')+L_2(\mathbf{x}')+\cdots+L_n(\mathbf{x}')}$, where $i = \sqrt{-1}$, and each $L_j(\mathbf{x}')$ is the indicator function for $\mathbf{a}_j \cdot \mathbf{x}' = 1$. Note that the dot product is computed over $\mathbb{F}_2$, whereas the sum $L_1(\mathbf{x}') + L_2(\mathbf{x}') + \cdots + L_n(\mathbf{x}')$ is computed over $\mathbb{Z}$ or, equivalently, over $\mathbb{Z}_4$. This completes the definition of $\mathcal{A}$. As before let $\mathcal{P}$ be the set of all functions which can be expressed as a product of nullary and unary functions, binary equality functions and binary disequality functions (now extended to complex functions). Cai, Lu and Xia [13] proved the following dichotomy.

▶ **Theorem 2.** *Suppose $\mathcal{F}$ is a finite set of functions mapping Boolean inputs to complex numbers. If $\mathcal{F} \subset \mathcal{A}$ or $\mathcal{F} \subset \mathcal{P}$, then $\#\mathrm{CSP}(\mathcal{F})$ is in* FP*. Otherwise, $\#\mathrm{CSP}(\mathcal{F})$ is* #P*-hard.*

This wraps up the Boolean case as far as the complexity of exact computation is concerned. We not only have a dichotomy, but one that takes an explicit form which is quite easy to comprehend, even in its most general statement (Theorem 2). Unfortunately, this happy state of affairs will not continue as we move to domains of cardinality greater than two.

2.2 Graph Homomorphisms and Partition Functions

Another natural subclass of counting CSPs, and one that has important connections to other fields, is obtained by restricting the constraint language Γ to a single binary symmetric relation. It is natural to view this relation as an undirected graph H, and the instance also as an undirected graph G whose vertices correspond to variables, and whose edges correspond to the scopes of the constraints. A (graph) homomorphism from G to H is a function $\phi : V(G) \to V(H)$ such that $\{\phi(u), \phi(v)\} \in E(H)$ whenever $\{u, v\} \in E(H)$. Thus, the problem $\#\mathrm{CSP}(\Gamma)$ is equivalent to counting homomorphisms from G to H. This correspondence gives rise to the alternative names *H-homomorphisms* or *H-colourings* for this restriction of counting CSPs. In the latter case, we are thinking of the vertices of H as "colours", and viewing a homomorphism from G to H as a colouring of the vertices of G in which the colours of adjacent vertices of G are constrained by the adjacency relation of H. We see that CSPs with one symmetric relation generalise usual graph colouring, in the same way that Boolean CSPs generalise the usual CNF satisfiability problem.

It is customary to allow the fixed graph H to have loops but not parallel edges; in other words, we do not assume that the single relation in Γ is irreflexive. To focus on the irreflexive situation would exclude some interesting and natural examples. For a graph H, possibly with loops, but without parallel edges, denote by $\#H$-Col is the following problem: given a graph G, return the number of graph homomorphisms from G to H. By way of example, if K_2' is the connected graph on two vertices with a loop on one of them, and K_3 is the complete graph on three vertices with no loops, then $\#K_2'$-Col asks for the number of independent sets, in G, and $\#K_3$-Col the number of (proper, vertex) 3-colourings. The first step in the study of the complexity of $\#H$-Col was made by Dyer and Greenhill [23] who proved the following dichotomy. Say that a graph is *reflexive* if all its vertices have loops and *irreflexive* if it is loop-free.

▶ **Theorem 3.** *If every connected component of H is a reflexive complete graph or an irreflexive complete bipartite graph, then $\#H$-Col is in* FP*. Otherwise, $\#H$-Col is* #P*-complete.*

The next step is to add weights. A weighted graph H on q vertices, can be thought of as a weighted adjacency matrix, which is a symmetric $q \times q$ matrix $A = (a_{ij} : 0 \leq i, j < q)$ with non-negative real entries. In statistical physics terminology, the matrix A defines a *spin model.* We'll refer to the matrix A as the interaction matrix of the model. For an instance G, which is an undirected graph, the *partition function* Z_A of this model is defined as follows:

$$Z_A(G) = \sum_{\sigma:V(G)\to[q]} \prod_{\{u,v\}\in E(G)} a_{\sigma(u),\sigma(v)}, \tag{1}$$

where $[q] = \{0, \ldots, q-1\}$. By way of example, consider the following matrices

$$A^{\lambda}_{\text{Ising}} = \begin{pmatrix} \lambda & 1 \\ 1 & \lambda \end{pmatrix} \quad \text{and} \quad A_{\text{BIS}} = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}. \tag{2}$$

(The second matrix may look a little mysterious, but it provides an illuminating test case, and will play an important role in the second part of this survey.) These interaction matrices yield, respectively, the partition functions of the classical Ising model (ferromagnetic in the case $\lambda > 1$ and antiferromagnetic when $\lambda < 1$) and the independent set (or hard-core) model on a bipartite graph. This correspondence between, on the one hand, partition functions of spin models in statistical physics and, on the other, weighted counting CSPs with a single symmetric binary function has led to this special case of counting CSPs being extensively studied.

As before, the generalisation to non-negative weights goes smoothly. Bulatov and Grohe [4] showed that an analogue of Theorem 3 holds for the computation of Z_A, with the role of the reflexive complete graph being taken by a rank-1 interaction matrix A, and that of the irreflexive bipartite graph being taken by an adjacency matrix in block form $\left(\begin{smallmatrix} 0 & B \\ B^{\top} & 0 \end{smallmatrix}\right)$, where B is of rank 1. Technically, we must also replace the conclusion of "#P-completeness" by "#P-hardness", as the required output is no longer an integer. Note that, unfortunately, all non-trivial spin systems, including the ones specified by A_{BIS} and $A^{\lambda}_{\text{Ising}}$ (with $\lambda \neq 1$) have hard-to-compute partition functions.

Generalising to arbitrary real weights, as we saw earlier, significantly increases the complexity. The 2×2 matrix H_2 we encountered in the previous section is one of an infinite sequence of matrices leading to tractable partition functions. Other Hadamard matrices yield tractable functions on larger domains. However, this is far from the end of the story: not every Hadamard matrix yields a tractable counting CSP, and not every tractable Boolean counting CSP arises directly from a Hadamard matrix. Nevertheless, Goldberg, Grohe, Jerrum and Thurley [35] showed that there is a dichotomy into FP and #P-hard. The characterisation is too complicated to describe here, though it is decidable. Cai, Chen and Lu [11] made the final step, generalising to complex weights. Again there is a dichotomy which is decidable (in fact in polynomial time) but too complex to describe here. It seems at this point that we have come a long way, but we'll see in Section 2.4 that we can go a quite a bit further yet.

2.3 Send in the Clones

Before studying general counting CSPs it is worth taking time to digress into functional clones, the counting analogue of relational clones, or co-clones.

Let D be a finite domain, $\mathcal{U}_k$ be the set of functions $D^k \to \mathbb{R}_{\geq 0}$, and $\mathcal{U} = \cup_{k=0}^{\infty} \mathcal{U}_k$. Denote by EQ the equality function defined by $\text{EQ}(x, y) = 1$ if $x = y$ and $\text{EQ}(x, y) = 0$ otherwise.

A set of functions $\mathcal{F} \in \mathcal{U}$ is a *functional clone* of it contains equality, and is closed under variable introduction, variable renaming, product, and summation over a variable. If $\mathcal{F} \subseteq \mathcal{U}$ is some (usually finite) set of functions, then $\langle \mathcal{F} \rangle_\#$ denotes the *functional clone generated by* $\mathcal{F}$, that is to say, the minimal functional clone containing $\mathcal{F}$. We say that a function $f \in \mathcal{U}$ is *pps-definable over* $\mathcal{F}$ if $f \in \langle \mathcal{F} \rangle_\#$. Relative to classical pp-definability, we have replaced conjunction by product and existential quantification by summation over a variable. However, the idea is the same: a function f is pps-definable over $\mathcal{F}$ if it can be "implemented" in terms of functions in $\mathcal{F}$, in just the same way as a relation is pp-definable over Γ if it can be "implemented" in terms of relations in Γ. Also, just as in the relational case, if $\langle \mathcal{F} \rangle_\# = \langle \mathcal{F}' \rangle_\#$ then $\#\mathrm{CSP}(\mathcal{F})$ and $\#\mathrm{CSP}(\mathcal{F}')$ are of equivalent computational complexity. For details, refer to [8], but note that there the subscript # is dropped from the notation for functional clones, since functional clones are the main object of study in that article. Note also that a more inclusive notion of functional clone is defined there which requires closure under taking limits of sequences of functions of the same arity.

Observe that even if we start in the unweighted or relational situation with a finite set of functions with range $\{0,1\}$, then pps-definablity will soon generate more general functions and take us into the weighted situation. Thus, with $D = \{0,1\}$,

$$g(x,y) = \sum_{z \in \{0,1\}} \mathrm{IMP}(x,z)\,\mathrm{IMP}(z,y),$$

defines the function g taking the values $g(0,0) = g(1,1) = 1$, $g(1,0) = 0$ and $g(0,1) = 2$. As an aside, the rationale for introducing limits is the following. It is fairly easy to check that the functional clone $\langle \mathcal{U}_0 \cup \{\mathrm{IMP}\} \rangle_\#$ contains, for all $n \in \mathbb{N}$, the unary function $f(x)$ defined by $f(0) = 2^{-n}$ and $f(1) = 1$, but not the function defined by $f(0) = 0$ and $f(1) = 1$. This is a trivial example, but it suggests that in some situations it may be reasonable to include limits.

Both the utility of functional clones and their limitations can be appreciated by reproving Theorem 1 using this technology. First a little notation. For a function $f : D^k \to \mathbb{R}_{\geq 0}$, define $\operatorname{supp} f \subseteq \{0,1\}^k$ to be the relation $\operatorname{supp} f = \{x \in D^k : f(x) > 0\}$. Extend this notation to sets of functions $\mathcal{F} \subseteq \mathcal{U}$ via $\operatorname{supp} \mathcal{F} = \{R : R = \operatorname{supp} f, \text{ for some } f \in \mathcal{F}\}$. If $\mathcal{F}$ is a functional clone then $\operatorname{supp} \mathcal{F}$ is a relational clone, and, moreover, for any set of functions $\mathcal{F} \subseteq \mathcal{U}$ we have $\operatorname{supp}\langle \mathcal{F} \rangle_\# = \langle \operatorname{supp} \mathcal{F} \rangle$, where $\langle \Gamma \rangle$ denotes the relational clone generated by a set of relations Γ. To appreciate this fact, consider the homomorphism $\varphi : (\mathbb{R}_{\geq 0}, +, \times) \to (\{0,1\}, \vee, \wedge\}$ defined by $\varphi(x) = 0$ if $x = 0$ and $\varphi(x) = 1$ if $x > 0$, and its action on the closure operations of pps-definability; alternatively, refer to [8].

So to the proof of Theorem 1. We specialise the above definitions to the Boolean domain $\{0,1\}$. In making use of Post's lattice, we refer to Böhler, Creignou, Reith and Vollmer [1] and Creignou, Kolaitis and Zanuttini [16], and employ their terminology. So suppose Γ is a finite subset of affine relations, that is to say $\Gamma \subset \mathsf{IL}_2$. As we noted earlier, the set of satisfying assignments to an instance of $\#\mathrm{CSP}(\Gamma)$ can be expressed as the solution set to a system of linear equations over $\mathbb{F}_2$, and so $\#\mathrm{CSP}(\Gamma)$ is in FP.

Now suppose $\Gamma \not\subseteq \mathsf{IL}_2$. For a relation $R \subseteq \{0,1\}^k$, denote by $\operatorname{fun} R$ the derived function $f : \{0,1\}^k \to \mathbb{R}_{\geq 0}$ defined by $f(x) = 1$ if $x \in R$ and $f(x) = 0$ otherwise. Extend fun to sets of relations via $\operatorname{fun} \Gamma = \{\operatorname{fun} R : R \in \Gamma\}$. Then the relational clone C generated by Γ is

$$C = \langle \Gamma \rangle = \langle \operatorname{supp} \operatorname{fun} \Gamma \rangle = \operatorname{supp}\langle \operatorname{fun} \Gamma \rangle_\#. \tag{3}$$

We know that C is not IL_2, nor any relational clone that lies below IL_2 in Post's lattice of relational clones. Inspecting Figure 2 of [1], we see that this implies that C contains one of relational clones IS_1^2, IS_0^2, IM or IN. In the first, second and third cases cases (see [16, Table 2]), C contains NAND, OR and IMP, respectively. By (3), this is turn implies that $\langle \mathrm{fun}\,\Gamma\rangle_\#$ contains a function f such that $\operatorname{supp} f$ is one of those three relations. But we know, from routine direct arguments, or from [4], that $\#\mathrm{CSP}(\{f\})$ is $\#\mathsf{P}$-hard in any of those three cases. In this context, note that the interaction matrix

$$A = \begin{pmatrix} f(0,0) & f(0,1) \\ f(1,0) & f(1,1) \end{pmatrix}$$

associated with f is necessarily of rank 2. The final case, $C \supseteq \mathsf{IN}$ presents a slight fly in the ointment. The relational clone IN contains precisely the relations that are 0-valid, 1-valid and "complementive", i.e., invariant under interchange of 0 and 1. In this case, $\langle \mathrm{fun}\,\Gamma\rangle_\#$ contains a function f such that $\operatorname{supp} f$ is the relation $\{0,1\}^3 \setminus \{(1,1,0),(0,0,1)\}$. Although one or other of the binary functions $f(x,y,z)\,\mathrm{EQ}(y,z)$ or $\sum_{z\in\{0,1\}} f(x,y,z)$ may fail the rank-2 test, it may be verified that at least one will pass. This demonstrates that $\Gamma \not\subseteq \mathsf{IL}_2$ implies $\#\mathrm{CSP}(\Gamma)$ is $\#\mathsf{P}$-hard, and completes the proof.

This argument could be carried through without using functional clones, using a theorem of Bulatov and Dalmau's [7, Thm 2], but part of the motivation for writing down the proof in the language of functional clones was to introduce a concept that will be important later. Even at this stage it is possible to appreciate the limitations of the utility of functional clones. We may very well be interested in sets $\mathcal{F}$ of functions, all of which have the complete relation as their supports. In that case, the above line of argument yields nothing. Bulatov's "max-co-clones" [6] may be viewed partly as a response to this failing.

2.4 #CSPs in General

The Feder-Vardi conjecture for decision CSPs [27] is famously open in its original form but, remarkably, has been resolved positively for counting CSPs by Bulatov [5]. The original proof was streamlined by Dyer and Richerby [24], who reduced the dependence on universal algebra and showed that the dichotomy is decidable. In order to state the dichotomy in the form given by Dyer and Richerby, we require some definitions. A matrix is said to be a *rank-one block matrix* if it can be transformed (by row and column permutations) into block diagonal form, such that every block has rank one. A ternary relation $R \subseteq A_1 \times A_2 \times A_3$ is *balanced* if the *balance matrix*

$$M(x,y) = |\{z \in A_3 : (x,y,z) \in R\}|, \quad \text{for all } x \in A_1 \text{ and } y \in A_2$$

is a rank-one block matrix. A set of relations Γ over domain D is *strongly balanced* if every ternary relation that is pp-definable over Γ is balanced.

▶ **Theorem 4.** *Suppose Γ is a finite set of relations over a finite domain D. If Γ is strongly balanced then $\#\mathrm{CSP}(\Gamma)$ is in FP. Otherwise, $\#\mathrm{CSP}(\Gamma)$ is $\#\mathsf{P}$-complete. Moreover, the dichotomy is decidable.*

It is possible to offer some hints towards the proof. First, some definitions. A binary relation $B \subseteq A_1 \times A_2$ is *rectangular* if $(a,c),(a,d),(b,c) \in B$ implies $(b,d) \in B$ for all $a,b \in A_1$ and $c,d \in A_2$. Suppose $R \subseteq D^n$ is a relation of arity $n \geq 2$. For each non-trivial partition of $[n]$ into blocks of size k and $n-k$ there is a natural isomorphism between D^n and $D^k \times D^{n-k}$ under which R can be viewed as a binary relation on $D^k \times D^{n-k}$. We say that R is rectangular if every expression of R as a binary relation on $D^k \times D^{n-k}$, for $1 \leq k < n$, is rectangular. A constraint language Γ is *strongly rectangular* if every relation in $\langle\Gamma\rangle$ of arity at least 2 is rectangular. Finally a relation $R \subseteq D^n$ is strongly rectangular if $\langle\{R\}\rangle$ is strongly rectangular. Dyer and Richerby [24] show that if Γ is strongly balanced then Γ is strongly rectangular, but that the converse does not hold. They also show that if Γ is not strongly balanced then #CSP(Γ) is #P-complete; this strengthens a result of Bulatov and Dalmau [7] that if Γ is not strongly rectangular then #CSP(Γ) is #P-complete.

The really difficult part of the proof is demonstrating tractability in the case that Γ is strongly balanced. For this we need the concept of a frame, which provides is a compact representation for strongly rectangular relations. Say that a set $D' \subseteq D$ is *i-equivalent in a relation* R if R contains tuples which agree on their first $i-1$ elements and whose ith elements are exactly the members of D'. A *frame* for a relation $R \subseteq D^n$ is a relation $F \subseteq R$ satisfying two properties: (i) whenever R contains a tuple whose ith component is a then F also contains such a tuple, and (ii) for $1 < i \leq n$, any set that is i-equivalent in R must also be i-equivalent in F. It can be shown that every strongly rectangular relation $R \subseteq D^n$ has a small frame, specifically, one of cardinality $n\,|D|$.

In the decision world, Bulatov and Dalmau [2] showed (though expressed in different terminology) that CSP(Γ) $\in$ FP for every strongly rectangular constraint language Γ. This result cannot translate to counting CSPs unless #P $\subseteq$ FP. However, it can be reproved with the technology of frames, giving a pointer as to how to proceed. Suppose (X, C) is an instance of CSP(Γ) with $|X| = n$. Note that the n-ary relation R defined by (X, C) is strongly rectangular, since $\langle\{R\}\rangle \subseteq \langle\Gamma\rangle$ and Γ is strongly rectangular. We can construct a small frame for R iteratively, starting with a frame for the complete relation $R_0 = D^n$ Let $R_0 \supseteq R_1 \supseteq \cdots \supseteq R_{|C|} = R$ be a sequence of relations in which R_i is obtained from R_{i-1} by removing the tuples that violate the ith constraint. At each step the frame is updated so that it represents the current relation. The process ends with a frame for R. It can be shown that a frame is empty iff the relation it represents is empty, so this process yields a polynomial-time algorithm for the decision problem CSP(Γ) in the case that Γ is strongly rectangular.

Dyer and Richerby demonstrate that frames can be used to count solutions, under the stronger assumption that Γ is strongly balanced. As before, they construct a frame for the relation R. Their approach is then one of dynamic programming based on a carefully selected set of subproblems. For $1 \leq i < j \leq n$, let $N_{i,j}(a)$ be the number of prefixes $(u_1, \ldots, u_i) \in D^i$ such that there is a tuple $(u_1, \ldots, u_n) \in R$ with $u_j = a$. The key step of the iteration is computing $N_{i,j}(\cdot)$ for each $j > i$, given $N_{i-1,j}(\cdot)$ for each $j \geq i$, and it turns out that this can be achieved using the property of strong balance. At the end of the process, summing $N_{n-1,n}(a)$ over $a \in D$ gives $|R|$.

The universal algebra doesn't go away, but is reduced to an easy to digest and intuitively appealing fragment. A Mal'tsev operation is a ternary operation $\varphi : D^3 \to D$ satisfying $\varphi(a, a, b) = \varphi(b, a, a) = b$ for all $a, b \in D$. An important fact proved in [24] is that a constraint language is strongly rectangular if and only if it has a Mal'tsev polymorphism. This fact has two important consequences. First, it allows an efficient implementation of membership testing in a strongly rectangular relation R given only a frame for R. Second, it allows an efficient (in NP) test for strong rectangularity. (Note that the definition of strong rectangularity in itself does not even imply decidability.) Testing strong rectangularity is the first step in testing strong balance. It transpires that deciding strong balance (and hence the dichotomy itself) is in NP.

The resolution of the counting version of the Feder-Vardi conjecture is a major achievement. One might ask how it is that the counting version has been resolved while the original decision version has not. Of course, this is a vague, possibly nonsensical question. However, it is difficult to avoid the thought that tractability results are generally harder to prove than intractability results, and $\#\text{CSP}(\Gamma)$ simply has fewer tractable cases than $\text{CSP}(\Gamma)$.

Perhaps as remarkable as the dichotomy for relational constraint languages itself is the fact that it has been extended to the weighted case. The first step, to non-negative real weights, was taken by Cai, Chen and Lu [10]. As we have seen already, the extension of dichotomies in counting CSPs to arbitrary real weights adds new possibilities for tractable cases that must be taken into account, and the further extension to complex weights provides further complications. This line of work culminated with Cai and Chen [9], who proved the existence of a dichotomy in the complex weighted case. They provide three rather clean conditions on a set of complex functions $\mathcal{F}$ – block orthogonality, Mal'tsev and type partition – that taken together imply $\#\text{CSP}(\mathcal{F}) \in \mathsf{FP}$. If any of the conditions fail, then $\#\text{CSP}(\mathcal{F})$ is #P-hard. Unfortunately, the conditions are not currently known to be decidable.

Although the conditions of block orthogonality, Mal'tsev and type partition, are really quite clean, it would take too much space to define them here. Nor is it feasible to give a sketch of the proof techniques. For those things the reader should consult the really lucid exposition of the definitions and proof sketch to be found in the conference version of Cai and Chen's paper [9]. Suffice it to say that the main ingredients of Dyer and Richerby's work survive, namely the compact representation of relations and the Mal'tsev polymorphism that allows information to be extracted from it, but completely new ideas need to be added, particularly in the definition and application of the type partition condition.

In summary, there has been massive progress in our understanding of the computational complexity of counting CSPs. In fact, the main questions in the basic setting have been all but answered. That is not to say that there is not much work to do: for example, with the notable exception of the extensive literature on read-twice #CSPs or holants, there is not a great deal of work on restricted instances, e.g., planar [44], and perhaps none at all on infinite domains.

3 Approximate Computation

We saw in Section 2 that certain non-trivial counting CSPs are exactly solvable using interesting algorithmic techniques, such as reduction to a system of linear equations over a finite field or to a quadratic form over $\mathbb{F}_2$. However, the general picture is gloomy, with intractability results dominating. This observation has prompted the search for approximation algorithms. An encouraging sign is that the partition function of a ferromagnetic Ising system (i.e., an instance of the two-spin model specified by the interaction matrix $A^{\lambda}_{\text{Ising}}$, with $\lambda > 1$) can be computed with small relative error in polynomial time [45]. Note that the interaction matrix $A^{\lambda}_{\text{Ising}}$ has rank two, so the partition function is #P-hard to compute exactly.

Before embarking on the study of specific counting CSPs, we need to say a something about the computational complexity of approximate counting problems in general. There is a well-established framework for this. We provide only an informal description here and direct the reader to Dyer, Goldberg, Greenhill and Jerrum [19] for precise definitions.

The standard notion of efficient approximation algorithm is *Fully Polynomial Randomised Approximation Scheme*, or FPRAS. This is a randomised algorithm that is required to produce a solution within relative error specified by a tolerance $\varepsilon > 0$, in time polynomial in the instance size and ε. Under some mild condition, an efficient algorithm that provides only very weak approximations can be boosted to the achieve the quality of approximation demanded by an FPRAS. As a consequence, in the context of counting problems, there is just one notion of approximation algorithm, namely FPRAS.[1] In this aspect, counting problems provide a contrast with optimisation problems, which exhibit a hierarchy of possible degrees of approximability.

Evidence for the non-existence of an FPRAS for a problem Π can be obtained through *Approximation-Preserving* (or AP-) *reductions*. These are polynomial-time Turing reductions that preserve (closely enough) the error tolerance. The key feature of the definition is that the class of problems admitting an FPRAS is closed under AP-reducibility. Every problem in $\#\mathsf{P}$ is AP-reducible to #SAT, so #SAT is complete for $\#\mathsf{P}$ with respect to AP-reductions. In the other direction, we know, using the bisection technique of Valiant and Vazirani [55, Corollary 3.6], that #SAT can be approximated (in the FPRAS sense) by a polynomial-time probabilistic Turing machine equipped with an oracle for the decision problem SAT. Thus, the counting version of any NP-complete decision problem is complete for $\#\mathsf{P}$ with respect to AP-reductions. Note the contrast with exact computation, where there may exist NP-complete decision problems whose counting analogue is not $\#\mathsf{P}$-hard under classical Turing reducibility.

We can summarise the situation as follows. Assuming we restrict attention to counting problems in $\#\mathsf{P}$ (and this includes all problems of the form $\#\mathrm{CSP}(\Gamma)$), the hardest problems Π are those that are complete for $\#\mathsf{P}$ with respect to AP-reducibility. Since such problems are AP-interreducible with #SAT, we will use the shorthand "Π is #SAT-equivalent", omitting the qualification "with respect to AP-reducibility" for brevity. We know that these problems do not have an FPRAS unless $\mathsf{RP} = \mathsf{NP}$. At the risk of overemphasising the point, in the context of approximate computation, the complexity of a problem that is AP-interreducible with #SAT lies only a little above NP (formally in the class $\mathsf{FP}^{\mathsf{NP}}$) and presumably far below $\#\mathsf{P}$.

Sometimes we have to settle for weaker evidence of computational intractability. The problem of counting independent sets in a bipartite graph is denoted by #BIS. The problem #BIS appears to be of intermediate complexity: on the one hand, there is no known FPRAS for #BIS (and it is generally believed that none exists) but, on the other hand, there is no known AP-reduction from #SAT to #BIS. The fact that #BIS is complete for a certain complexity class $\#\mathsf{RH}\Pi_1$ with respect to AP-reducibility [19], can be interpreted as evidence for the special status of #BIS and the problems AP-interreducible with it.

If there is an AP-reduction from #BIS to Π, we say that Π is #BIS-*hard*. We conjecture that no FPRAS for #BIS exists, in which case the same is true for all #BIS-hard problems. If there exists an AP-reduction from Π to #BIS, we say that Π is #BIS-*easy*; if Π is #BIS-hard and #BIS-easy then we say that Π is #BIS-*equivalent*. Many problems are in this last class, including counting downsets in a partial order [19], estimating the partition function of the Widom-Rowlinson model [19] or of the ferromagnetic Ising model with an external field [36]. In the absence of NP-hardness, the claim of #BIS-equivalence is currently almost the strongest one can make for an approximate counting problem Π, in that it locates the complexity of Π quite precisely.

3.1 Boolean #CSPs

As usual, restricting attention to Boolean #CSPs, i.e., those with domain-size two, allows us to make a brisk start. Let us further simplify matters by considering the unweighted

[1] This is not quite accurate. Another sweet spot is occupied by algorithms that provide an additive approximation to the *logarithm* of the solution, within $\pm\varepsilon n$, where n is the instance size.

case. Recall that IL_2 is the clone of affine relations, i.e., relations that can be expressed as the solution set of a system of linear equations over $\mathbb{F}_2$. Define the relational clone IM_2 by $\mathsf{IM}_2 = \langle \mathrm{IMP}, \delta_0, \delta_1 \rangle$, where δ_0 and δ_1 are the unary relations $\delta_0 = \{(0)\}$ and $\delta_1 = \{(1)\}$. Dyer, Goldberg and Jerrum [22] prove the following.

▶ **Theorem 5.** *Let Γ be a Boolean constraint language. If every relation in Γ is in IL_2, then $\#\mathrm{CSP}(\Gamma)$ is in FP. Otherwise, if every relation in Γ is in IM_2 then $\#\mathrm{CSP}(\Gamma)$ is $\#\mathrm{BIS}$-equivalent. Otherwise, $\#\mathrm{CSP}(\Gamma)$ is $\#\mathrm{SAT}$-equivalent.*

Using the language of functional clones and Post's lattice, it is possible to hint at a proof. For example, letting $C = \langle \Gamma \rangle$, any constraint language covered by the final part of the theorem will satisfy $C \not\subseteq \mathsf{IL}_2$ and $C \not\subseteq \mathsf{IM}_2$. Consulting Post's lattice of relational clones [1, Fig. 2], we find that C contains one of IS_1^2, IS_0^2 or IN. In the first two cases we can find an AP-reduction from the problem of counting independent sets in a general graph to $\#\mathrm{CSP}(\Gamma)$, and in the third case an AP-reduction from the problem of evaluating the partition function of the antiferromagnetic Ising model. Both the independent set and antiferromagnetic Ising problems are #SAT-equivalent, showing that $\#\mathrm{CSP}(\Gamma)$ is also. This sketch can be completed to a proof of the final part of the theorem.

Assuming that there is no FPRAS for #BIS, Theorem 5 is a discouraging result, as it says that the only Boolean counting CSPs that are efficiently approximable are the affine ones, which we already know to be exactly solvable. So relaxing the problem specification appears to have gained us nothing. Although we shall not be discussing restricted problem instances extensively in this survey, it is worth pointing out that Dyer, Goldberg, Jalsenius and Richerby [20] have shown that the hardness results in Theorem 5 continue to hold for instances of degree at most six, where the *degree* of a CSP instance is the maximum number of occurrences of any variable in the instance.

The next step is to introduce weights. We restrict attention to non-negative real weights, as this situation seems to give the greatest scope for positive results. (If negative weights are allowed, it is likely that we will be required to compute a small quantity that is the difference of two much larger quantities, and it will be hard to achieve small relative error.) Recall the material on functional clones from Section 2.3. No generally applicable theory of polymorphisms for functional clones exists. However, some interesting functional clones can be defined by operations reminiscent of multimorphisms or fractional polymorphisms in the study of valued CSPs (VCSPs).

Denote by $\mathcal{B}_k$ the set of functions $\{0,1\}^k \to \mathbb{R}_{\geq 0}$, and write $\mathcal{B} = \cup_{k=0}^{\infty} \mathcal{B}_k$. A function $f \in \mathcal{B}_n$ is *log-supermodular* (lsm) if

$$f(\mathbf{x} \vee \mathbf{y}) f(\mathbf{x} \wedge \mathbf{y}) \geq f(\mathbf{x}) f(\mathbf{y}), \quad \text{for all } \mathbf{x}, \mathbf{y} \in \mathcal{B}_n, \tag{4}$$

where $\wedge$ and $\vee$ denote meet and join in the Boolean lattice, which are equivalent to pointwise min and max. The terminology is justified by the observation that $f \in \mathcal{B}_n$ is lsm if and only if $\log f$ is supermodular. Note the similarity to multimorphisms in the study of VCSPs, but with multiplication replacing addition. We denote by $\mathsf{LSM} \subset \mathcal{B}$ the class of all lsm functions.

In the weighted situation we need to work with functional clones that are closed under taking limits of sequences of functions; as our functions are defined on a finite domain we don't need to be specific about the notion of convergence. The clone generated by a set of functions $\mathcal{F}$ with the additional limiting operation is denoted $\langle \mathcal{F} \rangle_{\#,\omega}$ There is no general result to the effect that sets of functions defined by conditions such as (4) are clones; each case must be handled individually. In this instance we do have a clone [8, Lemma 4.2].

▶ **Lemma 6.** $\langle \mathsf{LSM} \rangle_{\#,\omega} = \mathsf{LSM}$.

The non-trivial part of the proof lies in showing that LSM is closed under the operation of summing over a variable. It turns out that this requirement can be viewed as a special case of the Ahlswede-Daykin four-functions theorem.

As usual, we can say more about the conservative case, where all unary functions $\mathcal{B}_1$ are given free. Bulatov, Dyer, Goldberg, Jerrrum and McQuillan show [8].

▶ **Theorem 7.** *Suppose $\mathcal{F} \subseteq \mathcal{B}$.*

- *If $\mathcal{F} \not\subseteq \langle \mathrm{NEQ}, \mathcal{B}_1\rangle_\#$ then $\langle \mathrm{IMP}, \mathcal{B}_1\rangle_{\#,\omega} \subseteq \langle \mathcal{F}, \mathcal{B}_1\rangle_{\#,\omega}$.*
- *If, in addition, $\mathcal{F} \not\subseteq \mathsf{LSM}$ then $\langle \mathcal{F}, \mathcal{B}_1\rangle_{\#,\omega} = \mathcal{B}$.*

Informally, every non-trivial functional clone contains $\langle \mathrm{IMP}, \mathcal{B}_1\rangle_{\#,\omega}$ and any non-trivial clone containing a non-lsm function is in fact $\mathcal{B}$. In other words, all the interesting action takes place between $\langle \mathrm{IMP}, \mathcal{B}_1\rangle_{\#,\omega}$ and LSM.

Care is needed to obtain computational consequences from Theorem 7. In particular, it is necessary to introduce computationally efficient versions of $\mathcal{B}$ and of the closure operation $\langle\,\cdot\,\rangle_{\#,\omega}$. These are needed so that we can compute efficiently with functions in $\mathcal{B}$, and so that we can utilise the limiting operation in the proofs. This programme can in fact be carried out (see [8] for details), resulting in the following classification theorem, in which we assume that the necessary efficient versions of concepts are used.

▶ **Theorem 8.** *Suppose $\mathcal{F}$ is a finite subset of $\mathcal{B}$.*

1. *If $\mathcal{F} \subseteq \langle \mathrm{NEQ}, \mathcal{B}_1\rangle_\#$ then there is an FPRAS for $\#\mathrm{CSP}(\mathcal{F})$.*
2. *Otherwise,*
   (a) *there is a finite subset S of $\mathcal{B}_1$ such that $\#\mathrm{CSP}(\mathcal{F}, S)$ is #BIS-hard, and*
   (b) *if $\mathcal{F} \not\subseteq \mathsf{LSM}$ then there is a finite subset S of $\mathcal{B}_1$ such that $\#\mathrm{CSP}(\mathcal{F}, S)$ is #SAT-equivalent.*

The polynomial-time algorithm guaranteed in the first part of the theorem needs to compute sufficiently accurate approximations to functions in $\mathcal{F} \cap \mathcal{B}_1$; it is for this reason only that we specify an FPRAS and not an exact algorithm. The second part of the theorem may be conveniently illustrated with reference to the Ising model. The ferromagnetic Ising model with a field is covered by part (2a) of the theorem, and hence its partition function is #BIS-hard. (In fact the partition function is #BIS-equivalent, as can be seen from [8, Lemma 7.1] or Theorem 11(1b).) The antiferromagnetic Ising model with a field is covered by part (2b), and hence its partition function is #SAT-equivalent. These special cases were known earlier ([36] and [45]), but Theorem 8 places these isolated intractability results in a general setting. The Ising model will be discussed at greater length in §3.3.2.

It is natural to ask if Theorem 7 can be strengthed to a strict trichotomy. Unfortunately the answer is no. Consider the function $g : \{0,1\}^4 \to \mathbb{R}_{\geq 0}$ defined by

$$g(x_1,x_2,x_3,x_4) = \begin{cases} 4, & \text{if } x_1+x_2+x_3+x_4 = 4; \\ 2, & \text{if } x_1+x_2+x_3+x_4 = 3; \quad \text{and} \\ 1, & \text{otherwise.} \end{cases}$$

The function g is in LSM but not in $\langle \mathrm{IMP}, \mathcal{B}_1\rangle_{\#,\omega}$ [8, Lemma 11.9]. Nevertheless, it is entirely possible that $\#\mathrm{CSP}(\{g\} \cup \mathcal{B}_1)$ is #BIS-easy, since AP-reduction is a more liberal notion than pps-definabilty.

Theorem 8 encapsulates most of what is known about the computational complexity of general conservative Boolean counting CSPs. When we go beyond conservative, we know rather little. We do not even have a complete understanding of $\#\mathrm{CSP}(\mathcal{F})$ when $\mathcal{F} \subset \mathcal{B}_2$.

The problem is that the boundary between tractable and intractable becomes intimately tied up with the existence of phase transition in spin systems. However, much attention has been directed to this issue, and the restriction to the case where $\mathcal{F}$ contains a single *symmetric* binary functions is now well understood, as we shall see in the next section.

3.2 Graph Homomorphisms

We turn to the case of a single binary relation, which can be viewed as an undirected graph H, possibly with loops. As before, we look first at the conservative case, which means that arbitrary unary relations are available in addition to the binary relation H. In the graph theory community, this situation is described as *list H-colouring.* Formally, the problem #List-H-Col is defined as follows. An instance is a graph G and a collection of colour sets $\mathbf{S} = \{S_v \subseteq Q : v \in V(G)\}$, where $Q = V(H)$. The required output is the number of list H-colourings of $(G, \mathbf{S})$, i.e., the number of mappings $\sigma : V(G) \to Q$ such that $\sigma(v) \in S_v$ for all $v \in V(G)$, and $(\sigma(u), \sigma(v)) \in E(H)$ for all $(u, v) \in E(G)$.

A class of graphs is *hereditary* if it is closed under taking induced subgraphs; the class of bipartite graphs is a simple example. A moment's thought reveals that any maximal class of graphs $\mathcal{H}$ for which #List-H-Col is tractable for $H \in \mathcal{H}$ must be hereditary. On the basis of that general consideration, we expect hereditary graph classes to feature in any complexity classification of #List-H-Col. Two graph classes turn out to be important here. There are many equivalent definitions of these, but the matrix characterisation is perhaps easiest to grasp. Say that a 0,1-matrix $A = (a_{i,j} : 0 \leq i < n, 0 \leq j < m)$ has *staircase form* if the 1s in each row are contiguous and the following condition is satisfied: letting $\alpha_i = \min\{j : a_{i,j} = 1\}$ and $\beta_i = \max\{j : a_{i,j} = 1\}$, we require that the sequences (α_i) and (β_i) are non-decreasing. A graph is a *bipartite permutation graph* if the rows and columns of its biadjacency matrix can be (independently) permuted so that the resulting biadjacency matrix has staircase form. A graph is a *proper interval graph* if the rows and columns of its adjacency matrix can be (simultaneously) permuted so that the resulting adjacency matrix has staircase form. The decision version of #List-H-Col was studied by Feder, Hell and Hwang [26], who established a dichotomy between FP and NP-complete. Goldberg and Jerrum prove the following trichotomy for the counting version [30]. Recall that a graph H is reflexive if every vertex has a loop and irreflexive if no vertex has a loop.

▶ **Theorem 9.** *Suppose H is a connected undirected graph, possibly with loops.*
- *If H is an irreflexive complete bipartite graph or a reflexive complete graph then* #List-H-Col *is in* FP.
- *Otherwise, if H is an irreflexive bipartite permutation graph or a reflexive proper interval graph then* #List-H-Col *is* #BIS-*equivalent.*
- *Otherwise,* #List-H-Col *is* #SAT-*equivalent.*

The most interesting part of the proof lies in demonstrating #SAT-hardness in the final case of the theorem. Here, alternative "excluded subgraph" characterisations of the hereditary classes are useful. For example, a graph that is not a bipartite permutation graph must contain either an induced cycle of length other than four, or one of three special graphs. It is enough, then, to verify that each of these possible subgraphs corresponds to a hard list-colouring problem.

In the non-conservative situation, that is to say, the straight graph homomorphism counting problem called #H-Col, the situation becomes more complicated. Formally, #H-Col is defined as follows. An instance is a graph G and the required output is the number of H-colourings of G, i.e., the number of mappings $\sigma : V(G) \to V(H)$ such that

$(\sigma(u), \sigma(v)) \in E(H)$ for all $(u, v) \in E(G)$. For #H-Col we do not have a general complexity classification or even a plausible conjecture. We do, however, have the following complexity lower bound for non-trivial graphs H, due to Galanis, Goldberg and Jerrum [29].

▶ **Theorem 10.** *Let H be a graph, possibly with self-loops but without parallel edges. If every connected component of H is non-trivial (i.e., neither a reflexive complete graph nor an irreflexive complete bipartite graph), then #H-Col is #BIS-hard.*

The proof extends ideas from earlier work of Goldberg, Kelk and Paterson [42] concerning the related problem of *sampling* H-colourings.

There are some quite small graphs, two of them with as few as four vertices, that get in the way of a neat classification in the style of Theorem 9. Take, for example, the reflexive 4-cycle C_4^*. We know from Theorem 10 that #C_4^*-Col is #BIS-hard, but that is all; the problem #C_4^*-Col is not known either to be #BIS-easy or to be #SAT-hard. An extensive exploration of the complexity of #H-Col, undertaken by Kelk [46], suggests a potentially rich classification.

3.3 Partition Functions

By a partition function we mean a counting CSP, #CSP($\mathcal{F}$), where $\mathcal{F}$ is a single binary function, usually, but not necessarily, symmetric. We will assume in this section that the function is symmetric unless explicitly stated otherwise. Note that in the symmetric case, the problem instance can be viewed as an undirected graph. Despite being very restrictive, this special case is important because it covers partition function of spin models in statistical physics. In view of this, we'll use the term *spin model* as a shorthand for a counting CSP of the above form. Recall that a spin model with q spins can be represented by a $q \times q$ interaction matrix A. We say that A is *irreducible* if, for every pair $i, j \in [q]$, there exists an integer t such that $(A^t)_{ij} > 0$. If A is not irreducible then the domain $[q]$ can be partitioned into equivalence classes of interacting spins, and the partition function (1) decomposed into a sum of component partition functions, one for each equivalence class. (This assumes that the instance graph G is connected; the modification for disconnected G is easy.)

3.3.1 The Conservative Case

We look first at the conservative case, which translates to unary functions being freely available. In terms of spin models in physics, "conservative" corresponds to the existence of an applied field.

Let $A = (a_{ij} : 0 \le i, j < q)$ be an $q \times q$ matrix of non-negative reals. Given a graph G and an assignment $\mathbf{h} = (h_v : v \in V(G))$ of unary non-negative real functions to the vertices of G, we are interested in computing the extended partition function

$$Z_A(G, \mathbf{h}) = \sum_{\sigma : V(G) \to [q]} \prod_{\{u,v\} \in E(G)} a_{\sigma(u),\sigma(v)} \prod_{v \in V(G)} h_v(\sigma(v)). \tag{5}$$

Specifically, we would like to know the computational complexity of the following problem.

Name. EvalZ$_c(A)$.
Instance. A graph G and an assignment of unary functions $\mathbf{h} = (h_v : v \in V(G))$ to the vertices of G.
Output. $Z_A(G, \mathbf{h})$, where Z_A is the extended partition function (5).

The subscript "c" in the problem name is intended to indicate "conservative". In the conservative situation, we can restrict attention to irreducible interaction matrices A, since the complexity of computing the partition function Z_A is determined by the maximum complexity of computing $Z_{A'}$ for any block A' of A.

A certain class of spins models have to be treated separately. These are ones in which the spins can be partitioned into two blocks such that two spins can only be adjacent if they occur in different blocks. Such a spin model is called *imprimitive*, while the others are *primitive*. The interaction matrix of an imprimitive model can be written in block form:

$$A = \begin{pmatrix} 0 & B \\ B^{\mathsf{T}} & 0 \end{pmatrix}. \tag{6}$$

The following result is due to by Goldberg and Jerrum [39]. Although we'll be encountering a more general result later, this one has the advantage of providing an explicit and clearly effective characterisation. We say that matrix A is *log-supermodular* if every 2×2 submatrix has non-negative determinant.

▶ **Theorem 11.**

1. *Suppose A is primitive.*
   - **(a)** *If A has rank 1, then* $\textsc{EvalZ}_c(A) \in \mathsf{FP}$
   - **(b)** *Otherwise, if there is a simultaneous permutation of the rows and columns of A that renders A log-supermodular, then* $\textsc{EvalZ}_c(A)$ *is* #BIS*-equivalent.*
   - **(c)** *Otherwise,* $\textsc{EvalZ}_c(A)$ *is* #SAT*-equivalent.*
4. *Now suppose A is imprimitive. Write A in the form (6).*
   - **(a)** *If B has rank 1, then* $\textsc{EvalZ}_c(A) \in \mathsf{FP}$.
   - **(b)** *Otherwise, if there are independent permutations of the rows and columns of B that render B log-supermodular, then* $\textsc{EvalZ}_c(A)$ *is* #BIS*-equivalent.*
   - **(c)** *Otherwise,* $\textsc{EvalZ}_c(A)$ *is* #SAT*-equivalent.*

The log-supermodularity conditions in Theorem 11 are natural generalisations to the weighted situation of the graph-theoretic conditions in Theorem 9. However, it is not the case that one theorem is a generalisation of the other. It is true that Theorem 11 covers a wider range of interaction matrices, but at the same time it permits a wider range of unary functions $\mathbf{h}$ in the problem instance. In fact, Theorem 11 no longer holds if the functions in $\mathbf{h}$ are resticted to take values in $\{0, 1\}$, which is the situation in Theorem 9 [30].

3.3.2 Boolean Domain

As usual, we can say more about domain size two. As we are considering symmetric interactions, the interaction matrix can, after suitable normalisation, be written as $A = \left(\begin{smallmatrix} \beta & 1 \\ 1 & \gamma \end{smallmatrix}\right)$ with $\beta, \gamma \in \mathbb{R}_{\geq 0}$. Also, the problem instance is just an undirected graph G. As well as the weights for pairs of spins given by A, it is quite common to introduce weights for individual spins: 1 for spin 0 and λ for spin 1. The quantity we wish to study is the extended partition function (5) with $A = \left(\begin{smallmatrix} \beta & 1 \\ 1 & \gamma \end{smallmatrix}\right)$, and with h_v given by $h_v(0) = 1$ and $h_v(1) = \lambda$, for all $v \in V(G)$. For future convenience, we define the problem of interest in the general q setting. So letting the domain or set of spins be $Q = [q]$, we model the external field as a function $h : Q \to \mathbb{R}_{\geq 0}$. As usual, A is a $q \times q$ matrix of non-negative reals.

Name. $\textsc{EvalZ}(A, h)$.
Instance. A graph G.
Output. $Z_A(G, \mathbf{h})$, where Z_A is the extended partition function (5), and $h_v = h$ for all $v \in V(G)$.

We employ the following conventions: the function $h : Q \to \mathbb{R}_{\geq 0}$ will be specified as a column vector whose ith entry is $h(i)$; also, if h is the all-1 vector, then we omit h from the problem name. With this notation, the problem of immediate interest is $\textsc{EvalZ}\big(\left(\begin{smallmatrix}\beta & 1\\ 1 & \gamma\end{smallmatrix}\right), \left(\begin{smallmatrix}1\\ \lambda\end{smallmatrix}\right)\big)$. Although this problem formulation does not fit the CSP framework exactly, it is natural when viewed from the perspective of spin models with an external field.

Up to this point in our study of the complexity of approximating counting CSPs, the only tractable examples have been trivial. The situation now changes. Jerrum and Sinclair [45] presented an FPRAS for the partition function of the ferromagnetic Ising model, i.e., for $\textsc{EvalZ}\big(\left(\begin{smallmatrix}\beta & 1\\ 1 & \gamma\end{smallmatrix}\right), \left(\begin{smallmatrix}1\\ \lambda\end{smallmatrix}\right)\big)$ in the case $\beta = \gamma \geq 1$ and $\lambda = 1$ In fact, the algorithm they presented works also in the presence of an external field, i.e., for $\textsc{EvalZ}\big(\left(\begin{smallmatrix}\beta & 1\\ 1 & \gamma\end{smallmatrix}\right), \left(\begin{smallmatrix}1\\ \lambda\end{smallmatrix}\right)\big)$, with $\lambda \neq 1$. The reader may wonder how this result may be squared with Theorem 11. The matrix A, after all, has rank 2 and is log-supermodular, so Theorem 11 classifies the partition function as #BIS-equivalent. To resolve the paradox, note that the #BIS-equivalence result relates to a setting in which different functions h_v can be assigned to different vertices of the instance G. A varying field can be accommodated by the algorithm of [45] provided either spin 0 is always favoured, or spin 1 always favoured. Intractability apparently arises when 0- and 1-favouring fields are mixed. This phenomenon had been investigated earlier: see [36].

For the rest of the section we concentrate on the complexity of $\textsc{EvalZ}\big(\left(\begin{smallmatrix}\beta & 1\\ 1 & \gamma\end{smallmatrix}\right), \left(\begin{smallmatrix}1\\ \lambda\end{smallmatrix}\right)\big)$. An early investigation was carried out by Goldberg, Jerrum and Paterson [41], who mapped out some easy and hard regions in "phase space" $(\beta, \gamma, \lambda) \in \mathbb{R}_{\geq 0}$, but left quite a bit unclassified. To describe the more refined results that followed, we need to introduce a further parameter Δ, which is a uniform upper bound on the degrees of vertices of the instance graph G. We start our survey with the independent set or "hard-core" model, whose interaction matrix is $A_{\mathrm{IS}} = \left(\begin{smallmatrix}1 & 1\\ 1 & 0\end{smallmatrix}\right)$. After the Ising model, it is perhaps the most intensively studied spin model. Note that the partition function we are required to evaluate is $Z^{\lambda}_{\mathrm{IS}}(G) = \sum_{\sigma} \lambda^{|\sigma|}$, where the sum ranges over all independent sets σ of G, and $|\sigma| = |\sigma^{-1}(1)|$ denotes the size of the independent set σ.

Weitz [57] proved the following surprising and very influential result.

▶ **Theorem 12.** *Let* $\lambda_c = (\Delta-1)^{\Delta-1}/(\Delta-2)^{\Delta}$. *There is an FPRAS for* $\textsc{EvalZ}\big(\left(\begin{smallmatrix}1 & 1\\ 1 & 0\end{smallmatrix}\right), \left(\begin{smallmatrix}1\\ \lambda\end{smallmatrix}\right)\big)$ *restricted to graphs of maximum degree* Δ, *when* $\lambda < \lambda_c$.

To appreciate the result, it is important to understand the significance of the critical value λ_c. Given a finite graph G, there is a natural probability distribution on independent sets on G that assigns probability $\lambda^{|\sigma|}/Z^{\lambda}_{\mathrm{IS}}(G)$ to each independent set σ. Let $\mathbb{T}_{\Delta,\ell}$ denote the Δ-regular tree with root r and depth ℓ. For each ℓ fix some boundary configuration $\tau_\ell : \partial\,\mathbb{T}_{\Delta,\ell} \to \{0,1\}$ on the leaves $\partial\,\mathbb{T}_{\Delta,\ell}$ of $\mathbb{T}_{\Delta,\ell}$. If $\lambda < \lambda_c$ then $\Pr(\sigma(r) = 1)$ (i.e., the probability that the root r of the tree is in the independent set σ) tends to a limit, as $\ell \to \infty$, independently of the sequence of boundary conditions $(\tau_\ell : \ell \in \mathbb{N})$. If $\lambda > \lambda_c$, then the limit does not exist.

Since the ideas used to prove Theorem 12 have been influential, we provide a sketch of Weitz's approach here. Unlike previous approaches via Markov chain simulation, his approach leads to a *deterministic* approximation algorithm, technically a *Fully Polynomial-Time Approximation Scheme* or FPTAS. The formal definition of FPRAS is similar to that of FPRAS, except that the algorithm is deterministic, and the result is always within relative error $1 \pm \varepsilon$, rather than merely with high probability. Weitz's FPTAS for estimating the partition function $Z^{\lambda}_{\mathrm{IS}}(G)$ is based on an ingenious recursive algorithm for computing the probability that vertex v is occupied in a randomly chosen independent set in G. If this probability p_v can be estimated to sufficient accuracy then the partition function $Z^{\lambda}_{\mathrm{IS}}(G)$ can be estimated recursively, by estimating the partition function $Z_{\mathrm{IS}}^{\lambda}(G-v)$ of the graph G with vertex v and incident edges removed, and multiplying that quantity by $(1-p_v)^{-1}$. Note that $p_v \le \lambda/(\lambda+1)$, so this multiplicative factor is not too sensitive to errors in the evaluation of p_v.

Now we look at the same computation in a different way. We define a self-avoiding walk tree $T_{\mathrm{saw}}(G,v)$ whose vertices correspond to self-avoiding walks in G starting at vertex v. The root r of the tree corresponds to the self-avoiding walk of length 0, and the edges of the tree to extensions of a walk of length ℓ by one edge to a walk of length $\ell+1$. Since G is finite, so is the tree $T_{\mathrm{saw}}(G,v)$. Also, the degrees of vertices in the tree are bounded by Δ. Another ingenious ingredient in this approach is the rule for setting the boundary condition at the leaves of $T_{\mathrm{saw}}(G,v)$. A leaf arises when a self-avoiding walk loops back on itself, and the boundary condition in some sense encodes the cycle structure of G.

The probability that the root r is occupied in a randomly chosen independent set in $T_{\mathrm{saw}}(G,v)$ is easily computed using a simple recursive algorithm based on the inductive structure of the tree. The crucial observation is that, provided the boundary condition for $T_{\mathrm{saw}}(G,v)$ is set correctly, this recursive algorithm on the tree goes through exactly the same sequence of operations as the more complex recursive algorithm on the graph G alluded to earlier. The upshot is that we can compute the occupation probability p_v for vertex v in the graph G by computing the occupation probability of the root r of the tree $T_{\mathrm{saw}}(G,v)$.

We are not done, because the number of self-avoiding walks in G starting from v is exponential in $n=|V(G)|$. So although the self-avoiding walk tree is finite, it is nevertheless exponentially large in n. At this point we use the fact that $\lambda<\lambda_c$. When this condition holds, correlations in the tree decay exponentially fast, and the influence of vertices at depth greater than $c\ln n$ becomes small enough to be ignored, without altering the computed occupation probability of the root by too much. As a consequence, the recursive procedure for evaluating the occupation probability can be truncated at depth $O(\log n)$, while retaining adequate accuracy. This description necessarily skates over all the details, and even omits completely some critical issues.

One of those issues is the distinction between weak and strong spatial mixing. It is sufficiently important that we need to give some brief notes here. Earlier, we informally described a property that the sequence of trees $\mathbb{T}_{\Delta,\ell}$ might possess, namely the occupation probability of the root tends to a limit as $\ell\to\infty$, independently of the sequence of boundary conditions τ_ℓ. This property is *weak spatial mixing.* Roughly speaking, the property of *strong spatial mixing* obtains if the limit continues to exist even if the configuration σ (in this case an independent set) is fixed on some of the internal vertices of the trees.

Weitz's technique was extended by other authors. Sinclair, Srivastava and Thurley [52] considered the antiferromagnetic Ising model with a constant field on a graph of maximum degree Δ. Formally, they were interested in approximating $\textsc{EvalZ}\big(\big(\begin{smallmatrix}\beta & 1\\ 1 & \beta\end{smallmatrix}\big),\big(\begin{smallmatrix}1\\ \lambda\end{smallmatrix}\big)\big)$ when $\beta<1$ and $\lambda>0$, and the instance graph G has maximum degree Δ. For some critical value $\lambda_c(\beta,\Delta)$, we say that of β and λ are in the uniqueness region of the regular tree of degree Δ if either $\beta\ge\frac{\Delta-2}{\Delta}$, or $\beta<\frac{\Delta-2}{\Delta}$ and $\max\{\lambda,\lambda^{-1}\}>\lambda_c(\beta,\Delta)$. The critical value λ_c is determined by the existence of a fixed point to a certain recursion. (Determining λ_c is a contribution on the paper.) In the interior of the uniqueness region, the trees $(\mathbb{T}_{\Delta,\ell}:\ell\in\mathbb{N})$ with degree Δ exhibit the decay of correlations phenomenon known as weak spatial mixing, which we saw earlier in the case of the independent set model. (Outside of the uniqueness region, decay of correlations does not occur.) An important step in the argument is showing that weak spatial mixing implies strong spatial mixing. Then Weitz's self avoiding tree leads to:

▶ **Theorem 13.** *If $\beta < 1$ and $\lambda > 0$ are in the interior of the uniqueness region of the infinite regular tree of degree Δ, then there is a FPTAS for* $\textsc{EvalZ}\big(\left(\begin{smallmatrix}\beta & 1\\ 1 & \beta\end{smallmatrix}\right), \left(\begin{smallmatrix}1\\ \lambda\end{smallmatrix}\right)\big)$ *restricted to graphs of degree at most Δ.*

The algorithms in Theorems 12 and 13 have rather natural limits of validity, and it is reasonable to ask whether matching intractability results can be found. Outside of the uniqueness region, we do not have decay of correlations, which leaves open the possibility that we can construct gadgets of maximum degree Δ in which the spins are correlated at the global (or "macroscopic") level. Consider a regular bipartite graph B_Δ of degree Δ that is locally tree-like, a natural choice being a uniform random such graph. If $\Delta = 6$ then $\lambda_c < 1$ and we are outside the tree uniqueness region when $\lambda = 1$. This observation suggests that B_Δ may exhibit correlation at a global level. What we expect to happen is that a typical independent set will be asymmetric: a definite majority of the vertices in the independent set will accumulate on the left or right side of the bipartition of B_Δ. We can then plausibly use B_Δ as a bistable gadget in a reduction from an NP-hard decision or optimisation problem, to the problem $\textsc{EvalZ}(\left(\begin{smallmatrix}1 & 1\\ 1 & 0\end{smallmatrix}\right))$ (evaluating the partition function of the independent set model at $\lambda = 1$). In a rather basic form, this programme was carried through by Dyer, Frieze and Jerrum [18], to show that approximating $\textsc{EvalZ}(\left(\begin{smallmatrix}1 & 1\\ 1 & 0\end{smallmatrix}\right))$ is #SAT-equivalent when $\Delta \geq 25$. In other words, there is no FPRAS for counting independent sets in a graph of maximum degree 25, unless $\mathsf{RP} = \mathsf{NP}$.

Of course, 25 is a long way from 6. Using much more delicate arguments, Mossel, Weitz and Wormald [50] proved a negative result for λ just above the critical value λ_c of Theorem 12; specifically they showed that local Markov chain Monte Carlo algorithms for evaluating $\textsc{EvalZ}\big(\left(\begin{smallmatrix}1 & 1\\ 1 & 0\end{smallmatrix}\right), \left(\begin{smallmatrix}1\\ \lambda\end{smallmatrix}\right)\big)$ have exponential mixing time. Developing this theme, Sly and Sun [53] (see also Galanis, Štefankovič, Vigoda [28]), proved a general intractability result (i.e., one not restricted to a particular algorithmic technique, but conditional on standard complexity theoretic assumptions).

▶ **Theorem 14.** *The problem* $\textsc{EvalZ}\big(\left(\begin{smallmatrix}\beta & 1\\ 1 & \gamma\end{smallmatrix}\right), \left(\begin{smallmatrix}1\\ \lambda\end{smallmatrix}\right)\big)$ *restricted to graphs of maximum degree Δ is* #SAT*-equivalent in either of the following cases:*

- [The independent set model.] *$\beta = 1$, $\gamma = 0$ and $\lambda > \lambda_c = (\Delta - 1)^{\Delta-1}/(\Delta - 2)^{\Delta}$.*
- [The antiferromagnetic Ising model.] *$\beta = \gamma < 1$, and β and λ are outside of the uniqueness region of the Δ-regular tree.*

The complexity classification of antiferromagnetic two-spin systems, i.e., satisfying $\beta\gamma < 1$, culminates with the work of Li, Lu and Yin [47]. They show the following result, where, by convention, $\Delta = \infty$ indicates that there is no upper bound on vertex degree.

▶ **Theorem 15.** *Suppose $\beta\gamma < 1$ and $\Delta \geq 3$ or $\Delta = \infty$. Suppose also that, for all $\Delta' \leq \Delta$, the parameters (β, γ, λ) lie in the interior of the uniqueness region of the infinite Δ'-regular tree. Then there exists an FPTAS for the problem* $\textsc{EvalZ}\big(\left(\begin{smallmatrix}\beta & 1\\ 1 & \gamma\end{smallmatrix}\right), \left(\begin{smallmatrix}1\\ \lambda\end{smallmatrix}\right)\big)$ *restricted to graphs of maximum degree at most Δ.*

Combined with the negative results of Sly and Sun [53], this essentially completes the analysis of antiferromagnetic two-spin models, except at the boundary of the uniqueness region. We have a dichotomy between models that admit a FPTAS and those which are #SAT-equivalent, and everything is down to the uniqueness condition on regular trees of the appropriate degrees. It should be remembered, however, that we have restricted our attention to symmetric models, i.e., ones where the instance is an *undirected* graph, and the interaction matrix is symmetric. The non-symmetric situation is currently too complex to analyse completely.

In the absence of an external field (i.e., when $\lambda = 1$), the complexity of ferromagnetic models (i.e., those with $\beta\gamma \geq 1$), is easy to describe: they all admit an FPRAS by reduction to the ferromagnetic Ising model with a consistent field [41, 45]. However, if $\beta > \gamma$ and $\lambda > 1$ (or $\beta < \gamma$ and $\lambda < 1$) then a tension arises between the interactions between sites, which tend to pull in one direction, and the action of the field, which tends to pull in the other. How this tension resolves itself is not completely understood, but Liu, Lu and Zhang [49] and Guo and Lu [43] have extracted a great deal of information. It is reasonable to conjecture that there is a dichotomy, with all spin models either admitting an FPRAS or being #BIS-equivalent.

Finally, there in another way in which essentially ferromagnetic models can arise which exhibit the tension alluded to above, namely by restricting an antiferromagnetic model to a bipartite graph. Although we could in principle treat these by inverting the role 0 and 1 in one side of the bipartition, we would then lose symmetry, which, as we observed, is currently fatal. In fact, there is a dichotomy for bipartite antiferromagnetic models between spin models that admit an FPRAS and those that are #BIS-equivalent, as was shown by Cai, Galanis, Goldberg, Guo, Jerrum, Štefankovič and Vigoda [12].

3.3.3 Domain Size Greater Than Two

We have covered the conservative situation. So now suppose a symmetric $q \times q$ interaction matrix A is given, and we want to know the complexity of approximating $\textsc{EvalZ}(A)$, i.e, the complexity of computing an approximation to partition function $Z_A(G)$ defined in (1). In the Boolean case, there is a natural distinction between ferromagnetic ($\beta\gamma > 1$) and antiferromagnetic ($\beta\gamma < 1$) models. When $q > 2$ it is less clear what these terms should mean. Since A is symmetric, we know its eigenvalues are real. Suppose further that A is irreducible. By the Perron-Frobenius theorem, A has at least one positive eigenvalue. Galanis, Štefankovič and Vigoda say that a model is antiferromagnetic if all the other eigenvalues are negative.

The q state Potts model with interaction matrix

$$A^{q,B}_{\mathrm{Potts}} = \begin{pmatrix} B & 1 & \cdots & 1 \\ 1 & B & & 1 \\ \vdots & & \ddots & \\ 1 & 1 & & B \end{pmatrix} \in \mathbb{R}^{q\times q}_{\geq 0},$$

is antiferromagnetic under this, or any other reasonable definition of the term, when $B < 1$. $\textsc{EvalZ}(A^{q,B}_{\mathrm{Potts}})$ is #SAT-hard by a rather direct reduction from maximum q-way cut in a graph, which is an NP-hard optimisation problem. However, we can discuss, as we did in the case $q = 2$, the computational complexity of approximating $\textsc{EvalZ}(A^{q,B}_{\mathrm{Potts}})$, for restricted instances of degree at most Δ. Galanis, Štefankovič and Vigoda [32] prove the following.

▶ **Theorem 16.** *Suppose $q \geq 4$ is even, $\Delta > q$ and $0 \leq B < (\Delta - q)/\Delta$. Then $\textsc{EvalZ}(A^{q,B}_{\mathrm{Potts}})$, restricted to graphs of degree at most Δ, is #SAT-equivalent.*

The reduction employed in proving this result again employs random Δ-regular bipartite graphs as bistable gadgets. The condition for these gadgets to have distinguishable "phases" relates to a certain threshold in an infinite regular tree of degree Δ. However the picture is more complicated than in the case $q = 2$, and there is more than one critical value of B. The specific threshold that is relevant to Theorem 16 is the uniqueness threshold for "semi-translation-invariant measures". These are invariant measures on an infinite regular tree of degree Δ that are invariant under automorphisms of the tree that move the root a distance of two. Proving that the gadgets have the appropriate bistability property below the threshold is challenging. Some ingenious devices are introduced to simplify the technical details of the proof, but the paper still runs to 60 pages.

Theorem 16 provides a natural boundary beyond which the partition function of the antiferromagnetic Potts model is hard to approximate. Unlike the $q = 2$ case, we don't know whether we can approach the boundary arbitrary closely from the other side. This is because Weitz's approach has not so far been generalised to $q > 2$. As an illustration of the gap, in the special case $B = 0$, we have intractability when $q < \Delta$, but the best general positive result requires $q > \frac{11}{6}\Delta$ [56].

Galanis, Štefankovič, Vigoda and Yang [31] say that a model is ferromagnetic if the interaction matrix A is positive definite. An example is, of course, the ferromagnetic Potts model defined by the interaction matrix $A^{q,B}_{\text{Potts}}$ with $q \geq 2$ and $B > 1$. When $q = 2$, we know that an FPRAS exists [45]. In contrast, Goldberg and Jerrum [38] provide evidence of computational intractability when $q > 2$.

▶ **Theorem 17.** $\textsc{EvalZ}(A^{q,B}_{\text{Potts}})$ *is* #BIS*-hard, for all* $q \geq 3$ *and* $B > 1$*.*

What is the essential difference between the $q = 2$ and $q > 2$ situations that explains apparent switch from tractability to intractability? In both situations, there is a phase transition from a disordered to an ordered phase as B increases. However, the nature of that transition is different when $q > 2$ than when $q \leq 2$. This difference can be appreciated by looking at typical configurations of the Potts model on a complete graph when B is a little below and a little above the critical value, which we'll call B_o. Configurations are assignments $\sigma : V(G) \to [q]$ of spins to the vertices of G, and they occur with probability implicitly given by (1). Suppose we observe the fraction of vertices that are assigned the majority spin. For $B < B_o$, this fraction is roughly q^{-1} (the "disordered phase") but when $B > B_o$ it is strictly greater (the "ordered phase").

If we plot the fraction of majority spins as a function of B, we find a discontinuity at B_o: a discontinuity of the derivative when $q = 2$ and of the function itself when $q > 2$. A phase transition of the latter kind is called "first-order". At a first-order phase transition, the disordered and ordered phases coexist, and it is this that allows us to construct a bistable gadget, the two phases coding true and false. It appears that we cannot use such a gadget to code an NP-hard problem, but we can code the problem #BIS [38]. When $q = 2$, the phase transition is "second-order", and does not permit gadget construction.

Actually, using the random cluster formulation of the Potts model, we can make sense of the Potts partition function for non-integer q; with this interpretation, Theorem 17 holds for all $q > 2$. Note that this is best possible, as we noted earlier.

Galanis, Štefankovič, Vigoda and Yang [31] greatly strengthen this result so that it applies to bounded degree graphs. Suppose $q \geq 3$ and $\Delta \geq 3$. Define $B_o = B_o(q, \Delta) = (q-2)/[(q-1)^{1-2/\Delta} - 1]$; the significance of B_o is that it is the point of coexistence of ordered and disordered phases in the infinite regular tree of degree Δ.

▶ **Theorem 18.** $\textsc{EvalZ}(A^{q,B}_{\text{Potts}})$ *is* #BIS*-hard, for all* $q \geq 3$*,* $\Delta \geq 3$ *and* $B > B_o(q, \Delta)$*.*

The gadgets used in this proof are again random regular graphs. There are substantial technical hurdles to overcome, particularly in describing the phase transition in a very precise way, and proving rigorously that the description is correct.

The majority of spin models are neither ferromagnetic nor antiferromagnetic in the the sense described above, i.e., the number of negative eigenvalues is in the range $[1, q-2]$. What then? As a test case, we can take the interaction matrix associated with the Widom-Rowlinson model, namely

$$A_{\mathrm{WR}} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}.$$

The eigenvalues of this matrix are 1 and $1 \pm \sqrt{2}$, so the model is neither ferromagnetic nor antiferromagnetic in the technical sense. This matrix A_{WR} fits the second part of Theorem 11, so computing $\textsc{EvalZ}_c(A_{\mathrm{WR}})$ is #BIS-equivalent. Evidently, the model does not allow us to encode a hard partitioning problem, such as maximum cut in a graph, and so does not feel "antiferromagnetic".

On the other hand, if we replace the off-diagonal 1s by 2s, to get the modified matrix A'_{WR} then the eigenvalues are 1 and $1 \pm 2\sqrt{2}$ which is still indeterminate. (Replacing each 1 on the diagonal by $1+\varepsilon$ would also work.) We are now in situation of the third part of Theorem 11, so that $\textsc{EvalZ}_c(A'_{\mathrm{WR}})$ is #SAT-equivalent. Indeed it is not too difficult to see that the external fields are not really required, so that $\textsc{EvalZ}(A'_{\mathrm{WR}})$ is also #SAT-equivalent. (We just need to extract the latent antiferromagnetic Ising model embedded in the top-left 2×2 submatrix of A'_{WR}, which can be done with standard gadgetry.) This model feels genuinely antiferromagnetic. In summary, if the number of negative eigenvalues of A is in the range $[1, q-2]$ then the spin model with interaction matrix A may exhibit either ferromagnetic or antiferromagnetic characteristics.

3.4 #CSPs in General

Progress has been made towards classifying the complexity of approximating general counting CSPs, but only in the conservative case. Fix a finite domain D, and recall that $\mathcal{U}_k$, for all $k \in \mathbb{N}$, is the class of all functions $D^k \to \mathbb{R}_{\geq 0}$, and that $\mathcal{U} = \cup_{k=0}^{\infty} \mathcal{U}_k$. In particular, $\mathcal{U}_1$ is the set of unary functions, which are given free in the conservative case. Recall also the class of functions LSM that is defined in the case $|D| = 2$ by (4).

To state the main result concerning general counting CSPs, we require some further definitions. Recall the notion of functional clone from §2.3. A set of functions $\mathcal{F}$ is *weakly log-modular* if, for all binary functions $f \in \langle \mathcal{F} \rangle_\#$ and elements $a, b \in D$,

$$f(a,a)f(b,b) = f(a,b)f(b,a) \quad \text{or} \quad f(a,a) = f(b,b) = 0 \quad \text{or} \quad f(a,b) = f(b,a) = 0;$$

$\mathcal{F}$ is *weakly log-supermodular* if, for all binary functions $f \in \langle \mathcal{F} \rangle_\#$ and elements $a, b \in D$,

$$f(a,a)f(b,b) \geq f(a,b)f(b,a) \quad \text{or} \quad f(a,a) = f(b,b) = 0.$$

Finally, a problem Π is LSM-*easy* if there is a finite set $\mathcal{G} \subset \mathsf{LSM}$ of log-supermodular functions (over the Boolean domain) such that Π is AP-reducible to $\#\mathrm{CSP}(\mathcal{G})$.

Chen, Dyer, Goldberg, Jerrum, Lu, McQuillan and Richerby [14] studied general counting CSPs and found the following classification.

▶ **Theorem 19.** *Let $\mathcal{F} \subseteq \mathcal{U}$ be a set of functions that includes all unary functions $\mathcal{U}_1$.*

- *If $\mathcal{F}$ is weakly log-modular then $\#\mathrm{CSP}(\mathcal{G})$ is in* FP *for every finite $\mathcal{G} \subset \mathcal{F}$.*
- *If $\mathcal{F}$ is weakly log-supermodular but not weakly log-modular, then $\#\mathrm{CSP}(\mathcal{G})$ is* LSM-*easy for every finite $\mathcal{G} \subset \mathcal{F}$ and* #BIS-*hard for some such $\mathcal{G}$.*
- *If $\mathcal{F}$ is weakly log-supermodular but not weakly log-modular and consists of functions of arity at most two, then $\#\mathrm{CSP}(\mathcal{G})$ is* #BIS-*easy for every finite $\mathcal{G} \subset \mathcal{F}$ and* #BIS-*equivalent for some such $\mathcal{G}$.*
- *If $\mathcal{F}$ is not weakly log-supermodular, then $\#\mathrm{CSP}(\mathcal{G})$ is* #SAT-*easy for every finite $\mathcal{G} \subset \mathcal{F}$ and* #SAT-*equivalent for some such $\mathcal{G}$.*

This theorem is clearly more general than Theorem 11, but the latter provides more insight into the particular counting CSPs (i.e., partition functions) that it covers. Indeed, it is not obvious that the classification provided by Theorem 19 is decidable. However there is a kind of multimorphism underlying weak log-submodularity that can be tested fairly directly, and weak-modularity is essentially equivalent to another condition, known as "balance", that was already known to be decidable.

We saw already (see the comments following Theorem 11) that Theorem 19 does not in general establish a trichotomy. However, it does in the "bijunctive" case where all functions have arity at most 2.

4 Esoterica

Faben and Jerrum [25] considered the complexity of the problem $\oplus H$-Col of computing the parity of the number of H-colourings of a graph. This can be viewed as a counting CSP over $\mathbb{F}_2$, of the form $\#\mathrm{CSP}(\{f\})$, where $f : D^2 \to \mathbb{F}_2$ is a symmetric binary function. Define $\oplus$P to be the class of functions $\Sigma^* \to \{0,1\}$ that can be expressed as the number of accepting computations of a polynomial-time nondeterministic Turing machine, reduced modulo 2. It is tempting to conjecture that $\oplus H$-Col exhibits a dichotomy between FP and $\oplus$P-complete. However, the dichotomy here, if it exists, has a very different flavour to conventional counting CSPs.

In order to understand the possible nature of the dichotomy, we introduce a reduction system on undirected graphs in which a single transition has the following form. Suppose H is an undirected graph, possibly with loops. If π is an involution of H (automorphism of order 2), remove from H all vertices that are moved by π and denote the resulting graph by H^π. Then $H \to H^\pi$ is a possible transition of the system. If H has no involution, then no transition from H is possible; in this case, H is a normal form. This reduction system is confluent, that is to say, for each H there is a unique normal form H_0 such that $H \to^* H_0$. where $\to^*$ is the transitive closure of the reduction relation $\to$. Call a graph trivial if it has zero vertices, one vertex (with or without a loop), or two disconnected vertices, one with a loop and one without. Suppose H is a graph and H_0 is its normal form. It is easy to show that $\oplus H$-Col in FP if H_0 is trivial. Faben and Jerrum conjecture that $\oplus H$-Col is $\oplus$P-complete if H_0 is not trivial, and confirm the conjecture in the special case that H is a tree.

The conjecture for general graphs is still open. However, Göbel, Goldberg and Richerby confirm the conjecture for cactus graphs [33] and square-free graphs [34]. A graph is a *cactus* if every edge is in at most one (simple) cycle. Note that trees are a special case of cactus graphs. A graph is *square-free* if it contains no (not necessarily induced) 4-cycle.

Finally, one can study variants of $\#\mathrm{CSP}(\Gamma)$ in which only minimal (or maximal) satisfying assignments are to be counted. Durand and Hermann consider the problem of "propositional circumscription" [17]. Fix the domain to be $D = \{0,1\}$. A circumscription problem is defined as usual by a constraint language Γ of relations of various arities over D. An instance (X, C) is specified by a set of variables X and constraints C. Instead of counting all satisfying assignments, we are required to count just the minimal such assignments. A satisfying assignment $\sigma : X \to \{0,1\}$ is *mininal* if there does not exist a satisfying assignment $\sigma' \neq \sigma$ such that $\sigma'(x) \leq \sigma(x)$ for all $x \in X$.

The first thing to note is that we are (apparently) no longer working within the complexity class #P. A non-deterministic polynomial-time Turing machine can guess an assignment $\sigma : X \to \{0,1\}$ and decide whether it is satisfying, but it cannot in general decide whether a satisfying assignment is minimal. Indeed, Durand and Hermann show that circumscription in general is $\# \cdot$coNP-complete and hence, presumably, not in #P. (Roughly, a problem is in $\# \cdot$coNP if it is a witness counting problem for which witness checking is in coNP. In this case, deciding whether a satisfying assignment σ is minimal is clearly in coNP.)

However, Durand and Hermann prove that certain circumscription problems are in fact #P-complete: examples include ones whose constraint language Γ that are bijunctive (all relations in Γ have arity at most two), or that are affine or dual Horn. In contrast, the circumscription problems deriving from constraint languages that are Horn, or that are both affine and bijunctive, are in FP (trivially, in the former case).

Within a similar framework, Goldberg and Jerrum [40] consider the problem of counting satisfying assignments that are locally maximal. The crucial difference with Durand and Hermann lies in the "locally" and not in the "maximal". A satisfying assignment σ is *locally maximal* if any assignment σ' that can be obtained from σ by flipping a single 0 to a 1 is unsatisfying. Local maximality can easily be tested in polynomial time, so we find ourselves again working within the complexity class #P.

It turns out that counting locally maximal satisfying assignments can sometimes be easier than counting all satisfying assignments, but never harder. One kind of constraint language Γ that is trivially tractable in this variant is one in which all relations $R \in \Gamma$ are monotone (increasing). A relation R of arity k is *monotone* if for all $(x_1, \dots, x_k) \in R$ and all $i \in [k]$, it is the case that $(x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_k) \in R$. Actually, this definition can be relaxed slightly to essentially monotone, while retaining tractability. Let $Z \subseteq [n]$ be the set of indices for which R forces $x_i = 0$; that is, $i \in Z$ if $(x_1, \dots, x_k) \in R$ implies $x_i = 0$. Then R is *essentially monotone* if it is locally monotone when restricted to the variables $\{x_i : i \in [k] \setminus Z\}$ (and with the variables in Z set to 0).

Goldberg and Jerrum [40] show that the dichotomy for exact counting (Theorem 1) and the trichotomy for approximate counting (Theorem 5) carry over to locally maximal CSPs provided we add an additional case asserting tractability in the case that every relation in Γ is essentially monotone.

Acknowledgements. My understanding of the topic, such as it is, was acquired through collaborations with many people. I therefore thank my coauthors: Andrei Bulatov, Jin-Yi Cai, Xi Chen, Martin Dyer, John Faben, Alan Frieze, Andreas Galanis, Leslie Ann Goldberg, Catherine Greenhill, Martin Grohe, Heng Guo, Pinyan Lu, Colin McQuillan, Mike Paterson, David Richerby, Alistair Sinclair, Daniel Štefankovič, Mark Thurley and Eric Vigoda. Any misunderstandings are my own.

References

1 Elmar Böhler, Nadia Creignou, Steffen Reith, and Heribert Vollmer. Playing with Boolean blocks, part II: Constraint satisfaction problems. *ACM SIGACT Newsletter*, 35:22–35, 2004.

2 Andrei Bulatov and Víctor Dalmau. A simple algorithm for Mal′tsev constraints. *SIAM J. Comput.*, 36(1):16–27 (electronic), 2006. doi:10.1137/050628957.

3 Andrei Bulatov, Martin Dyer, Leslie Ann Goldberg, Markus Jalsenius, and David Richerby. The complexity of weighted Boolean #CSP with mixed signs. *Theoret. Comput. Sci.*, 410(38-40):3949–3961, 2009. doi:10.1016/j.tcs.2009.06.003.

4 Andrei Bulatov and Martin Grohe. The complexity of partition functions. *Theoret. Comput. Sci.*, 348(2-3):148–186, 2005. doi:10.1016/j.tcs.2005.09.011.

5 Andrei A. Bulatov. The complexity of the counting constraint satisfaction problem. *J. ACM*, 60(5):Art 34, 41, 2013. doi:10.1145/2528400.

6 Andrei A. Bulatov. Boolean max-co-clones. *Algebra Universalis*, 74(1-2):139–162, 2015. doi:10.1007/s00012-015-0336-1.

7 Andrei A. Bulatov and Víctor Dalmau. Towards a dichotomy theorem for the counting constraint satisfaction problem. *Inform. and Comput.*, 205(5):651–678, 2007. doi:10.1016/j.ic.2006.09.005.

8 Andrei A. Bulatov, Martin Dyer, Leslie Ann Goldberg, Mark Jerrum, and Colin McQuillan. The expressibility of functions on the Boolean domain, with applications to counting CSPs. *J. ACM*, 60(5):Art. 32, 36, 2013. doi:10.1145/2528401.

9 Jin-Yi Cai and Xi Chen. Complexity of counting CSP with complex weights. In *STOC'12 – Proceedings of the 2012 ACM Symposium on Theory of Computing*, pages 909–919. ACM, New York, 2012. doi:10.1145/2213977.2214059.

10 Jin-Yi Cai, Xi Chen, and Pinyan Lu. Non-negatively weighted #CSP: an effective complexity dichotomy. In *26th Annual IEEE Conference on Computational Complexity*, pages 45–54. IEEE Computer Soc., Los Alamitos, CA, 2011.

11 Jin-Yi Cai, Xi Chen, and Pinyan Lu. Graph homomorphisms with complex values: a dichotomy theorem. *SIAM J. Comput.*, 42(3):924–1029, 2013. doi:10.1137/110840194.

12 Jin-Yi Cai, Andreas Galanis, Leslie Ann Goldberg, Heng Guo, Mark Jerrum, Daniel Štefankovič, and Eric Vigoda. #BIS-hardness for 2-spin systems on bipartite bounded degree graphs in the tree non-uniqueness region. *Journal of Computer and System Sciences*, 82(5):690–711, 2016. doi:http://dx.doi.org/10.1016/j.jcss.2015.11.009.

13 Jin-Yi Cai, Pinyan Lu, and Mingji Xia. The complexity of complex weighted Boolean #CSP. *J. Comput. System Sci.*, 80(1):217–236, 2014. doi:10.1016/j.jcss.2013.07.003.

14 Xi Chen, Martin Dyer, Leslie Ann Goldberg, Mark Jerrum, Pinyan Lu, Colin McQuillan, and David Richerby. The complexity of approximating conservative counting CSPs. *J. Comput. System Sci.*, 81(1):311–329, 2015. doi:10.1016/j.jcss.2014.06.006.

15 Nadia Creignou and Miki Hermann. Complexity of generalized satisfiability counting problems. *Inform. and Comput.*, 125(1):1–12, 1996. doi:10.1006/inco.1996.0016.

16 Nadia Creignou, Phokion Kolaitis, and Bruno Zanuttini. Structure identification of Boolean relations and plain bases for co-clones. *J. Comput. System Sci.*, 74(7):1103–1115, 2008. doi:10.1016/j.jcss.2008.02.005.

17 Arnaud Durand and Miki Hermann. On the counting complexity of propositional circumscription. *Inform. Process. Lett.*, 106(4):164–170, 2008. doi:10.1016/j.ipl.2007.11.006.

18 Martin Dyer, Alan Frieze, and Mark Jerrum. On counting independent sets in sparse graphs. *SIAM J. Comput.*, 31(5):1527–1541, 2002. doi:10.1137/S0097539701383844.

19 Martin Dyer, Leslie Ann Goldberg, Catherine Greenhill, and Mark Jerrum. The relative complexity of approximate counting problems. *Algorithmica*, 38(3):471–500, 2004. Approximation algorithms. doi:10.1007/s00453-003-1073-y.

20 Martin Dyer, Leslie Ann Goldberg, Markus Jalsenius, and David Richerby. The complexity of approximating bounded-degree Boolean #CSP. *Inform. and Comput.*, 220/221:1–14, 2012. doi:10.1016/j.ic.2011.12.007.

21 Martin Dyer, Leslie Ann Goldberg, and Mark Jerrum. The complexity of weighted Boolean #CSP. *SIAM J. Comput.*, 38(5):1970–1986, 2008/09. doi:10.1137/070690201.

22 Martin Dyer, Leslie Ann Goldberg, and Mark Jerrum. An approximation trichotomy for Boolean #CSP. *J. Comput. System Sci.*, 76(3-4):267–277, 2010. doi:10.1016/j.jcss.2009.08.003.

23 Martin Dyer and Catherine Greenhill. The complexity of counting graph homomorphisms. *Random Structures Algorithms*, 17(3-4):260–289, 2000. doi:10.1002/1098-2418(200010/12)17:3/4<260::AID-RSA5>3.3.CO;2-N.

24 Martin E. Dyer and David Richerby. An effective dichotomy for the counting constraint satisfaction problem. *SIAM J. Comput.*, 42(3):1245–1274, 2013. doi:10.1137/100811258.

25 John Faben and Mark Jerrum. The complexity of parity graph homomorphism: an initial investigation. *Theory Comput.*, 11:35–57, 2015. doi:10.4086/toc.2015.v011a002.

26 Tomas Feder, Pavol Hell, and Jing Huang. Bi-arc graphs and the complexity of list homomorphisms. *J. Graph Theory*, 42(1):61–80, 2003. doi:10.1002/jgt.10073.

27 Tomás Feder and Moshe Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: a study through Datalog and group theory. *SIAM J. Comput.*, 28(1):57–104 (electronic), 1999. doi:10.1137/S0097539794266766.

28 Andreas Galanis, Qi Ge, Daniel Štefankovič, Eric Vigoda, and Linji Yang. Improved inapproximability results for counting independent sets in the hard-core model. *Random Structures Algorithms*, 45(1):78–110, 2014. doi:10.1002/rsa.20479.

29 Andreas Galanis, Leslie Ann Goldberg, and Mark Jerrum. Approximately Counting H-Colorings is #BIS-Hard. *SIAM J. Comput.*, 45(3):680–711, 2016. doi:10.1137/15M1020551.

30 Andreas Galanis, Leslie Ann Goldberg, and Mark Jerrum. A complexity trichotomy for approximately counting list H-colourings. *CoRR*, abs/1602.03985, 2016. Extended abstract to appear in Proc. International Colloquium for Automata, Languages and Programming (ICALP), 2016. URL: http://arxiv.org/abs/1602.03985.

31 Andreas Galanis, Daniel Štefankovič, Eric Vigoda, and Linji Yang. Ferromagnetic Potts Model: Refined #BIS-hardness and Related Results. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2014, September 4-6, 2014, Barcelona, Spain*, volume 28 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 677–691. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2014. doi:10.4230/LIPIcs.APPROX-RANDOM.2014.677.

32 Andreas Galanis, Daniel Štefankovič, and Eric Vigoda. Inapproximability for antiferromagnetic spin systems in the tree nonuniqueness region. *J. ACM*, 62(6):50:1–50:60, December 2015. doi:10.1145/2785964.

33 Andreas Göbel, Leslie Ann Goldberg, and David Richerby. The complexity of counting homomorphisms to cactus graphs modulo 2. *ACM Trans. Comput. Theory*, 6(4):Art. 17, 29, 2014. doi:10.1145/2635825.

34 Andreas Göbel, Leslie Ann Goldberg, and David Richerby. Counting homomorphisms to square-free graphs, modulo 2. *ACM Trans. Comput. Theory*, 8(3):12:1–12:29, May 2016. doi:10.1145/2898441.

35 Leslie Ann Goldberg, Martin Grohe, Mark Jerrum, and Marc Thurley. A complexity dichotomy for partition functions with mixed signs. *SIAM J. Comput.*, 39(7):3336–3402, 2010. doi:10.1137/090757496.

36 Leslie Ann Goldberg and Mark Jerrum. The complexity of ferromagnetic Ising with local fields. *Combin. Probab. Comput.*, 16(1):43–61, 2007. doi:10.1017/S096354830600767X.

37 Leslie Ann Goldberg and Mark Jerrum. Inapproximability of the Tutte polynomial. *Inform. and Comput.*, 206(7):908–929, 2008. doi:10.1016/j.ic.2008.04.003.

38 Leslie Ann Goldberg and Mark Jerrum. Approximating the partition function of the ferromagnetic Potts model. *J. ACM*, 59(5):Art. 25, 31, 2012. doi:10.1145/2371656.2371660.

39 Leslie Ann Goldberg and Mark Jerrum. A complexity classification of spin systems with an external field. *Proceedings of the National Academy of Sciences*, 112(43):13161–13166, 2015. doi:10.1073/pnas.1505664112.

40 Leslie Ann Goldberg and Mark Jerrum. The complexity of counting locally maximal satisfying assignments of Boolean CSPs. *Theoret. Comput. Sci.*, 634:35–46, 2016. doi:10.1016/j.tcs.2016.04.008.

41 Leslie Ann Goldberg, Mark Jerrum, and Mike Paterson. The computational complexity of two-state spin systems. *Random Structures Algorithms*, 23(2):133–154, 2003. doi:10.1002/rsa.10090.

42 Leslie Ann Goldberg, Steven Kelk, and Mike Paterson. The complexity of choosing an H-coloring (nearly) uniformly at random. *SIAM J. Comput.*, 33(2):416–432 (electronic), 2004. doi:10.1137/S0097539702408363.

43 Heng Guo and Pinyan Lu. Uniqueness, spatial mixing, and approximation for ferromagnetic 2-spin systems. *CoRR*, abs/1511.00493, 2015. URL: http://arxiv.org/abs/1511.00493.

44 Heng Guo and Tyson Williams. The complexity of planar Boolean #CSP with complex weights. In *Automata, languages, and programming. Part I*, volume 7965 of *Lecture Notes in Comput. Sci.*, pages 516–527. Springer, Heidelberg, 2013. doi:10.1007/978-3-642-39206-1_44.

45 Mark Jerrum and Alistair Sinclair. Polynomial-time approximation algorithms for the Ising model. *SIAM J. Comput.*, 22(5):1087–1116, 1993. doi:10.1137/0222066.

46 Steven Kelk. *On the relative complexity of approximately counting H-colourings.* PhD thesis, Warwick University, 2003.

47 Liang Li, Pinyan Lu, and Yitong Yin. Correlation decay up to uniqueness in spin systems. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 67–84. SIAM, Philadelphia, PA, 2012. Full version available at arXiv:1111.7064.

48 Rudolf Lidl and Harald Niederreiter. *Finite fields*, volume 20 of *Encyclopedia of Mathematics and its Applications.* Cambridge University Press, Cambridge, second edition, 1997. With a foreword by P. M. Cohn.

49 Jingcheng Liu, Pinyan Lu, and Chihao Zhang. The complexity of ferromagnetic two-spin systems with external fields. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2014, September 4-6, 2014, Barcelona, Spain*, volume 28 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 843–856. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2014. doi:10.4230/LIPIcs.APPROX-RANDOM.2014.843.

50 Elchanan Mossel, Dror Weitz, and Nicholas Wormald. On the hardness of sampling independent sets beyond the tree threshold. *Probab. Theory Related Fields*, 143(3-4):401–439, 2009. doi:10.1007/s00440-007-0131-9.

51 J. Scott Provan and Michael O. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM J. Comput.*, 12(4):777–788, 1983. doi:10.1137/0212053.

52 Alistair Sinclair, Piyush Srivastava, and Marc Thurley. Approximation algorithms for two-state anti-ferromagnetic spin systems on bounded degree graphs. *J. Stat. Phys.*, 155(4):666–686, 2014. doi:10.1007/s10955-014-0947-5.

53 Allan Sly and Nike Sun. Counting in two-spin models on d-regular graphs. *Ann. Probab.*, 42(6):2383–2416, 2014. doi:10.1214/13-AOP888.

54 Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM J. Comput.*, 8(3):410–421, 1979. doi:10.1137/0208032.

55 Leslie G. Valiant and Vijay V. Vazirani. NP is as easy as detecting unique solutions. *Theoret. Comput. Sci.*, 47(1):85–93, 1986. doi:10.1016/0304-3975(86)90135-0.

56 Eric Vigoda. Improved bounds for sampling colorings. *J. Math. Phys.*, 41(3):1555–1569, 2000. doi:10.1063/1.533196.

57 Dror Weitz. Counting independent sets up to the tree threshold. In *STOC'06: Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pages 140–149. ACM, New York, 2006. doi:10.1145/1132516.1132538.

The Complexity of Valued CSPs*

Andrei Krokhin[1] and Stanislav Živný[2]

1 School of Engineering and Computing Sciences, University of Durham, UK
andrei.krokhin@durham.ac.uk

2 Department of Computer Science, University of Oxford, UK
standa.zivny@cs.ox.ac.uk

Abstract

We survey recent results on the broad family of problems that can be cast as valued constraint satisfaction problems (VCSPs). We discuss general methods for analysing the complexity of such problems, give examples of tractable cases, and identify general features of the complexity landscape.

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes, G.1.6 Optimization

Keywords and phrases Constraint satisfaction problems, Optimisation, Tractability

Digital Object Identifier 10.4230/DFU.Vol7.15301.233

1 Introduction

Computational problems from many different areas involve finding an assignment of values to a set of variables, where that assignment must satisfy some specified feasibility conditions and optimise some specified objective function. In many such problems the objective function can be represented as a sum of functions, each of which depends on some subset of the variables. Examples include: Gibbs energy minimisation, Markov Random Fields (MRF), Conditional Random Fields (CRF), Min-Sum Problems, Minimum Cost Homomorphism, Constraint Optimisation Problems (COP) and Valued Constraint Satisfaction Problems (VCSP) [11, 30, 87, 110, 114, 116].

We focus in this article on a generic framework for such problems that captures their general form. Bringing all such problems into a common framework draws attention to common aspects that they all share, and allows a very general algebraic approach for analysing their complexity to be developed. The primary motivation for this line of research is to understand the general picture of complexity within this general framework, rather than to develop specialised techniques for specific applications.

We will give an overview of this algebraic approach, and the results that have been obtained by using it. We will focus on algorithms for solving problems to optimality and on the computational complexity of such problems. Although there is a survey from two years ago [61], the recent massive progress [43, 77, 78, 105, 72] justifies a new survey.

The generic framework we use is the *valued constraint satisfaction problem* (VCSP), defined formally as follows. Throughout the paper, let D be a fixed finite set and let $\overline{\mathbb{Q}} = \mathbb{Q} \cup \{\infty\}$ denote the set of rational numbers with (positive) infinity.

* Stanislav Živný was supported by a Royal Society University Research Fellowship.This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 714532). The paper reflects only the authors' views and not the views of the ERC or the European Commission. The European Union is not liable for any use that may be made of the information contained therein.

▶ **Definition 1.** We denote the set of all functions $\phi : D^m \rightarrow \overline{\mathbb{Q}}$ by $\mathbf{\Phi}_D^{(m)}$ and let $\mathbf{\Phi}_D = \bigcup_{m\geq 1} \mathbf{\Phi}_D^{(m)}$. We will often call the functions in $\mathbf{\Phi}_D$ *cost functions* over D.

Let $V = \{x_1, \dots, x_n\}$ be a set of variables. A *valued constraint* over V is an expression of the form $\phi(\mathbf{x})$ where $\mathbf{x} \in V^m$ and $\phi \in \mathbf{\Phi}_D^{(m)}$. The number m is called the *arity* of the constraint, the function ϕ is called the *constraint function*, and the tuple $\mathbf{x}$ the *scope* of the constraint.

We will call the elements of D *labels* (for variables), and say that the cost functions in $\mathbf{\Phi}_D$ take *values*.

▶ **Definition 2.** An instance of the *valued constraint satisfaction problem* (VCSP) is specified by a finite set $V = \{x_1, \dots, x_n\}$ of variables, a finite set D of labels, and an *objective function* Φ expressed as follows:

$$\Phi(x_1, \dots, x_n) = \sum_{i=1}^{q} \phi_i(\mathbf{x}_i) \tag{1}$$

where each $\phi_i(\mathbf{x}_i)$, $1 \leq i \leq q$, is a valued constraint over V. Each constraint can appear multiple times in Φ. The goal is to find an *assignment* of labels to the variables (or *labelling*) that minimises Φ.

Note that the value of the function Φ for any assignment of labels to the variables in V is given by the sum of the values taken by the constraints; this value will sometimes be called the *cost* of the assignment. An infinite value for any constraint indicates an infeasible assignment.

If the constraint functions in some VCSP instance are finite-valued, i.e., take only finite values, then every assignment is feasible, and the problem is to identify an assignment with minimum possible cost (i.e., we need to deal only with the optimisation issue). On the other hand, if each constraint function in an instance takes only two values: one finite value (possibly specific to the constraint) and ∞, then all feasible assignments are equally good, and so the only question is whether any such assignment exists (i.e., we need to deal only with the feasibility issue). If we have neither of the above cases then we need to deal with both feasibility and optimisation. To emphasise where this can happen we sometimes call VCSPs general-valued.

In Section 2 we give examples to show that many standard combinatorial optimisation problems can be conveniently expressed in the VCSP framework. In Section 3 we define certain algebraic properties of the constraints that can be used to identify tractable cases. Section 4 describes the basics of a general algebraic theory for analysing the complexity of different forms of valued constraints. In Sections 5 and 6 we use this algebraic theory to identify several tractable and intractable cases. In Section 7 we discuss the oracle model for representing the objective function. Finally, Section 8 gives a brief summary and identifies some open problems.

2 Problems and Frameworks Captured by the VCSP

In this section we will give examples of specific problems and previously studied frameworks that can be expressed as VCSPs with restricted forms of constraints.

▶ **Definition 3.** Any set $\Gamma \subseteq \mathbf{\Phi}_D$ is called a *valued constraint language* over D, or simply a *language*. We will denote by VCSP(Γ) the class of all VCSP instances in which the constraint functions are all contained in Γ.

Valued constraint languages may be infinite, but it will be convenient to follow [17, 25] and define the complexity of a valued constraint language in terms of its finite subsets. We assume throughout that $\mathrm{P} \neq \mathrm{NP}$.

▶ **Definition 4.** A valued constraint language Γ is called *tractable* if $\mathrm{VCSP}(\Gamma')$ can be solved (to optimality) in polynomial time for every finite subset $\Gamma' \subseteq \Gamma$, and Γ is called *NP-hard* if $\mathrm{VCSP}(\Gamma)$ is NP-hard for some finite $\Gamma' \subseteq \Gamma$.

One advantage of defining tractability in terms of finite subsets is that the tractability of a valued constraint language is independent of whether the cost functions are represented explicitly (say, via full tables of values, or via tables for the finite-valued parts) or implicitly (via oracles).

▶ **Example 5** (1-in-3-Sat)**.** Let $D = \{0, 1\}$ and let $\Gamma_{\textsf{1-in-3}}$ be the language that contains just the single ternary cost function $\phi_{\textsf{1-in-3}} : D^3 \to \overline{\mathbb{Q}}$ defined by

$$\phi_{\textsf{1-in-3}}(x, y, z) \overset{\text{def}}{=} \begin{cases} 0 & \text{if exactly one of } x, y, z \text{ is } 1 \\ \infty & \text{otherwise} \end{cases} .$$

The problem $\mathrm{VCSP}(\Gamma_{\textsf{1-in-3}})$ is exactly the 1-in-3-Sat problem. This problem is NP-hard [95, 44], so $\Gamma_{\textsf{1-in-3}}$ is NP-hard.

▶ **Example 6** (NAE-Sat)**.** Let $D = \{0, 1\}$ and let $\Gamma_{\textsf{nae}}$ be the language that contains just the single ternary cost function $\phi_{\textsf{nae}} : D^3 \to \overline{\mathbb{Q}}$ defined by

$$\phi_{\textsf{nae}}(x, y, z) \overset{\text{def}}{=} \begin{cases} \infty & \text{if } x = y = z \\ 0 & \text{otherwise} \end{cases} .$$

The problem $\mathrm{VCSP}(\Gamma_{\textsf{nae}})$ is exactly the Not-All-Equal-Sat problem, also known as the 3-Uniform Hypergraph 2-Colourability problem. This problem is NP-hard [44], so $\Gamma_{\textsf{nae}}$ is NP-hard.

▶ **Example 7** (Max-k-Cut)**.** Let $\Gamma_{\textsf{xor}}$ be the language that contains just the single binary cost function $\phi_{\textsf{xor}} : D^2 \to \overline{\mathbb{Q}}$ defined by

$$\phi_{\textsf{xor}}(x, y) \overset{\text{def}}{=} \begin{cases} 1 & \text{if } x = y \\ 0 & \text{if } x \neq y \end{cases} .$$

The problem $\mathrm{VCSP}(\Gamma_{\textsf{xor}})$ corresponds to the problem of minimising the number of monochromatic edges in a k-colouring (where $k = |D|$) of the graph G formed by the scopes of the constraints. This problem is known as the Maximum k-Cut problem (or simply Max-Cut when $|D| = 2$), and is NP-hard [44].

Hence, for any choice of D with $|D| \geq 2$, the language $\Gamma_{\textsf{xor}}$ is NP-hard.

▶ **Example 8** (Potts model)**.** Let $\Gamma_{\textsf{Potts}}$ be the language that contains all unary cost functions and the single binary cost function $\phi_{\textsf{Potts}}$: $D^2 \to \overline{\mathbb{Q}}$ defined by

$$\phi_{\textsf{Potts}}(x, y) \overset{\text{def}}{=} \begin{cases} 0 & \text{if } x = y \\ 1 & \text{if } x \neq y \end{cases} .$$

The problem $\mathrm{VCSP}(\Gamma_{\textsf{Potts}})$ corresponds to finding the minimum energy state of the Potts model (with external field) from statistical mechanics [93]. This model is also used as the basis for a standard Markov Random Field approach to a wide variety of problems in machine vision [11]. For $|D| = 2$, the function $\phi_{\textsf{Potts}}$ is submodular (see Example 19) and we will show that this implies that $\Gamma_{\textsf{Potts}}$ is tractable. For $|D| > 2$, $\Gamma_{\textsf{Potts}}$ is NP-hard as it includes, as a special case, the multiway cut problem, which is NP-hard [34].

▶ **Example 9** ((s,t)-Min-Cut)**.** Let $G = (V, E)$ be a directed weighted graph such that for every $(u, v) \in E$ there is a weight $w(u, v) \in \mathbb{Q}_{\geq 0}$ and let $s, t \in V$ be distinguished source and target nodes. Recall that an (s,t)*-cut* C is a subset of vertices V such that $s \in C$ but $t \notin C$. The weight, or the size, of an (s,t)-cut C is defined as $\sum_{(u,v)\in E, u\in C, v\notin C} w(u, v)$. The (s,t)-Min-Cut problem consists in finding a minimum-weight (s,t)-cut in G. We can formulate the search for a minimum-weight (s,t)-cut in G as a VCSP instance as follows.

Let $D = \{0, 1\}$. For any label $d \in D$ and cost $c \in \overline{\mathbb{Q}}$, we define

$$\eta_d^c(x) \overset{\text{def}}{=} \begin{cases} 0 & \text{if } x = d \\ c & \text{if } x \neq d \end{cases}.$$

For any weight $w \in \mathbb{Q}_{\geq 0}$, we define

$$\phi_{\mathsf{cut}}^w(x, y) \overset{\text{def}}{=} \begin{cases} w & \text{if } x = 0 \text{ and } y = 1 \\ 0 & \text{otherwise} \end{cases}.$$

We denote by Γ_{cut} the set $\{\eta_0^\infty, \eta_1^\infty\} \cup \{\phi_{\mathsf{cut}}^w \mid w \in \mathbb{Q}_{\geq 0}\}$. A minimum-weight (s,t)-cut in a graph G with set of nodes $V = \{x_1, \ldots, x_n\}$ corresponds to the set of variables assigned the label 0 in a minimal cost assignment to the VCSP instance defined by

$$\Phi(x_1, \ldots, x_n) \overset{\text{def}}{=} \eta_0^\infty(s) + \eta_1^\infty(t) + \sum_{(x_i,x_j)\in E} \phi_{\mathsf{cut}}^{w(x_i,x_j)}(x_i, x_j).$$

The unary constraints ensure that the source and target nodes must be assigned the labels 0 and 1, respectively, in any minimal cost assignment.

Furthermore, it is an easy exercise to show that any instance of VCSP(Γ_{cut}) on n variables can be solved in $O(n^3)$ time by a reduction to (s,t)-Min-Cut and then using the standard algorithm [45]. Hence Γ_{cut} is tractable.

▶ **Example 10** (Minimum Vertex Cover)**.** The Minimum Vertex Cover problem asks for a minimum size set W of vertices in a given graph $G = (V, E)$ such that each edge in E has at least one endpoint in W. This problem is NP-hard [44].

Let $D = \{0, 1\}$. We define

$$\phi_{\mathsf{vc}}(x, y) \overset{\text{def}}{=} \begin{cases} \infty & \text{if } x = y = 0 \\ 0 & \text{otherwise} \end{cases}.$$

We denote by Γ_{vc} the language $\{\phi_{\mathsf{vc}}, \eta_0^1\}$, where η_0^1 is the function defined in Example 9 that imposes unit cost for any variable assigned the label 1. A minimum vertex cover in a graph G with set of vertices $V = \{x_1, \ldots, x_n\}$ corresponds to the set of vertices assigned the label 1 in some minimum cost assignment to the VCSP(Γ_{vc}) instance defined by

$$\Phi(x_1, \ldots, x_n) \overset{\text{def}}{=} \sum_{x_i\in V} \eta_0^1(x_i) + \sum_{(x_i,x_j)\in E} \phi_{\mathsf{vc}}(x_i, x_j).$$

The binary constraints ensure that in any minimal cost assignment at least one endpoint of each edge belongs to the vertex cover.

Furthermore, it is easy to convert any instance of VCSP(Γ_{vc}) to an equivalent instance of Minimum Vertex Cover by repeatedly assigning the label 1 to all variables which do not appear in the scope of any unary constraints and removing these variables and all constraints involving them. Hence Γ_{vc} is NP-hard.

We will now show how several broad frameworks previously studied in the literature can be expressed as special cases of the VCSP with restricted languages. We will discuss algorithms and complexity classifications for them in Sections 5 and 6.

▶ **Example 11** (CSP)**.** The standard constraint satisfaction problem (CSP) over any fixed set of possible labels D can be seen as the special case of the VCSP where all cost functions take only the values 0 or ∞, representing allowed (satisfying) and disallowed tuples, respectively. Such constraints and cost functions are sometimes called *crisp*. In other words, the CSP can be seen as VCSP(Γ_{crisp}), where Γ_{crisp} is the language consisting of all cost functions on some fixed set D with range $\{0, \infty\}$. Note that the CSP can also be cast as the homomorphism problem for relational structures [37] (cf. Example 12).

Since the CSP includes many known NP-hard problems, such as 1-in-3-Sat (Example 5) and Graph-3-Colouring, the language Γ_{crisp} is clearly NP-hard. However, many tractable subsets of Γ_{crisp} have been identified [95, 62, 37, 17, 13, 18, 57, 6], mostly through an algebraic approach whose extension we discuss in Section 4. There are many surveys on the complexity of the CSP, see the books [33, 32, 80], and also [22, 51, 2].

Feder and Vardi conjectured that the CSP exhibits a *dichotomy*: that is, every finite language $\Gamma \subseteq \Gamma_{\mathsf{crisp}}$ is either tractable or NP-hard [37], thus excluding problems of intermediate complexity, as given by Ladner's Theorem (assuming P≠NP) [83]. The *Algebraic Dichotomy* conjecture, which we state formally and discuss in Section 6, specifies the precise boundary between tractable and NP-hard crisp languages [17].

▶ **Example 12** (Digraph Homomorphism)**.** Given two digraphs $G = (V(G), E(G))$ and $H = (V(H), E(H))$, a mapping $f : V(G) \to V(H)$ is a *homomorphism* from G to H if f preserves edges, that is, $(u, v) \in E(G)$ implies $(f(u), f(v)) \in E(H)$.

The problem whether an input digraph G admits a homomorphism to a fixed digraph H is also known as the H-Colouring problem and has been actively studied in graph theory [50, 51], see also [84].

For any digraph H, let $D = V(H)$ and let Γ_H be the language that contains just the single binary cost function $\phi_H : D^2 \to \overline{\mathbb{Q}}$ defined by

$$\phi_H(x, y) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } (x, y) \in E(H) \\ \infty & \text{otherwise} \end{cases} .$$

For any digraph H, the problem VCSP(Γ_H), which is a special case of the CSP (Example 11), corresponds to the H-colouring problem, where the input digraph G is given by the scopes of the constraints. If we add all unary crisp functions to Γ_H then the resulting VCSP is known as List H-Colouring [50, 51].

It is known that both the Feder-Vardi conjecture and the Algebraic Dichotomy conjecture are equivalent to their restrictions to the H-colouring problem [20, 37].

▶ **Example 13** (Max-CSP)**.** An instance of the (weighted) maximum constraint satisfaction problem (Max-CSP) is an instance of the CSP where the goal is to maximise the (weighted) number of satisfied constraints.

When seeking the optimal solution, maximising the number of satisfied constraints is the same as minimising the number of unsatisfied constraints. Hence for any instance Φ of the Max-CSP, we can define a corresponding VCSP instance Φ' in which each constraint c of Φ is associated with a constraint over the same scope in Φ' which assigns cost 0 to tuples allowed by c, and cost 1 to tuples disallowed by c. It follows that Max-CSP is equivalent to VCSP(Γ_{Max}), where Γ_{Max} is the language consisting of cost functions whose values are restricted to zero and one.

For $D = \{0, 1\}$, the complexity of all subsets of Γ_{Max} has been completely classified in [70]. Initial results for languages over arbitrary finite sets appeared in [21].

▶ **Example 14** (Min-Cost-Hom). Here and in the following examples let Γ_{unary} consist of all unary cost functions and let $\Gamma_{\mathsf{mc}} = \Gamma_{\mathsf{crisp}} \cup \Gamma_{\mathsf{unary}}$ (where Γ_{crisp} is defined in Example 11). Problems of the form VCSP(Γ) with $\Gamma \subseteq \Gamma_{\mathsf{mc}}$ have been studied under the name of the Minimum-Cost Homomorphism problem (or Min-Cost-Hom) [49, 52, 100, 101, 110, 111]. Note that the first three of these papers assume that $\Gamma_{\mathsf{unary}} \subseteq \Gamma$, while the last three do not. In [49, 52] Γ is assumed to be of the form $\{\phi_H\} \cup \Gamma_{\mathsf{unary}}$, where ϕ_H is a binary crisp cost function, as in Example 12.

In any instance of VCSP(Γ_{mc}), the crisp constraints specify the CSP part, i.e., the feasibility aspect of the problem, while the unary constraints specify the optimisation aspect. More precisely, the unary constraints specify the costs of assigning labels to individual variables. Complexity classifications for special cases of Min-Cost-Hom will be discussed in Section 6.

▶ **Example 15** (Min-Ones). An instance of the Boolean Minimum Ones (Min-Ones) problem is an instance of the CSP over $D = \{0, 1\}$ where the goal is to satisfy all constraints and minimise the number of variables assigned the label 1. Such instances correspond to Min-Cost-Hom instances over $\{0, 1\}$ in which all unary constraints are of the form η_0^1 as defined in Example 9 (which impose a unit cost for any variables assigned the label 1). A classification of the complexity of all subsets of this language was obtained in [70, 33].

▶ **Example 16** (Min-Sol). The Minimum Solution problem (Min-Sol) [67, 65] is a generalisation of Min-Ones from Example 15 to D being a larger set of non-negative integers, where the only allowed unary cost function is a particular finite-valued injective function, namely $u(x) = wx$ for some positive $w \in \mathbb{Q}$. Thus, this problem is also a subproblem of Min-Cost-Hom. A complexity classification for Min-Sol problems will be discussed in Section 6.

3 Polymorphisms and Fractional Polymorphisms

To develop general tools to classify the complexity of different valued constraint languages, we will now define certain algebraic properties of cost functions.

A function $f : D^k \to D$ is called a k-ary *operation* on D. The k-ary *projections*, defined for all $1 \leq i \leq k$, are the operations $\mathrm{e}_i^{(k)}$ such that $\mathrm{e}_i^{(k)}(x_1, \ldots, x_k) = x_i$. For any tuples $\mathbf{x_1}, \ldots, \mathbf{x_k} \in D^m$, we denote by $f(\mathbf{x_1}, \ldots, \mathbf{x_k})$ the tuple in D^m obtained by applying f to $\mathbf{x_1}, \ldots, \mathbf{x_k}$ componentwise.

For a cost function $\phi : D^m \to \overline{\mathbb{Q}}$, we denote by $\mathrm{Feas}(\phi) = \{\mathbf{x} \in D^m \mid \phi(\mathbf{x}) \text{ is finite}\}$ the *feasibility relation* of ϕ. We will view Feas(ϕ) both as a relation and as a $\{0, \infty\}$-valued cost function. Recall from Example 11 that $\{0, \infty\}$-valued cost functions are called *crisp*.

Any valued constraint language Γ defined on D can be associated with a set of operations on D, known as the polymorphisms of Γ, and defined as follows.

▶ **Definition 17** (Polymorphism). Let $\phi : D^m \to \overline{\mathbb{Q}}$ be a cost function. We say that a k-ary operation $f : D^k \to D$ is a *polymorphism* of ϕ if, for any $\mathbf{x_1}, \ldots, \mathbf{x_k} \in \mathrm{Feas}(\phi)$ we have that $f(\mathbf{x_1}, \ldots, \mathbf{x_k}) \in \mathrm{Feas}(\phi)$.

For any valued constraint language Γ over a set D, we denote by Pol(Γ) the set of all operations on D which are polymorphisms of all $\phi \in \Gamma$. We denote by $\mathrm{Pol}^{(k)}(\Gamma)$ the k-ary operations in Pol(Γ).

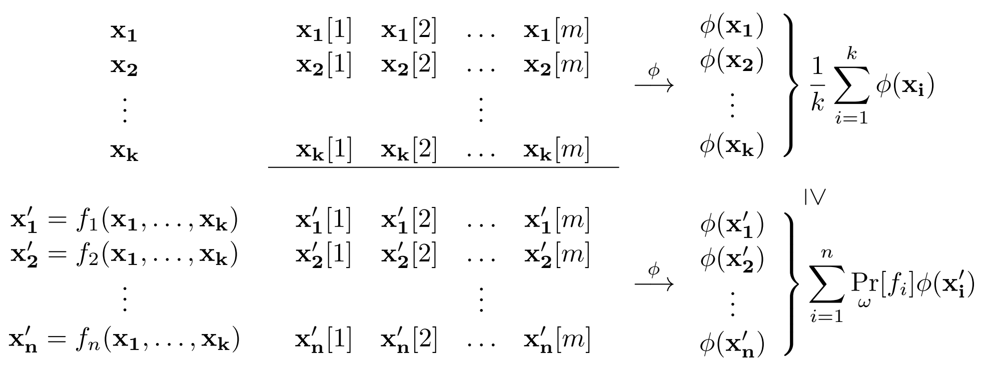


Figure 1 Probabilistic definition of a fractional polymorphism.

Note that trivially the projections are polymorphisms of all valued constraint languages.

For $\{0,\infty\}$-valued cost functions (relations) this notion of polymorphism corresponds precisely to the standard notion of polymorphism for relations [10, 62]. This notion of polymorphism has played a key role in the analysis of complexity for the CSP [62, 17]. However, for the analysis of the VCSP we need a more flexible notion that assigns weights to polymorphisms [23].

▶ **Definition 18** (Fractional Polymorphism). Let $\phi : D^m \to \overline{\mathbb{Q}}$ be a cost function. A function $\omega : \mathrm{Pol}^{(k)}(\phi) \to \mathbb{Q}_{\geq 0}$ is called a k-ary *fractional polymorphism* of ϕ if it satisfies the following conditions:

- $\sum_{f \in \mathrm{Pol}^{(k)}(\phi)} \omega(f) = 1$;
- for any $\mathbf{x_1}, \ldots, \mathbf{x_k} \in \mathrm{Feas}(\phi)$

$$\sum_{f \in \mathrm{Pol}^{(k)}(\phi)} \omega(f)\phi(f(\mathbf{x_1}, \ldots, \mathbf{x_k})) \;\leq\; \frac{1}{k}\sum_{i=1}^{k} \phi(\mathbf{x_i})\,. \tag{2}$$

We define $\mathrm{supp}(\omega) = \{f \mid \omega(f) > 0)$ to be the *support* of ω.

▶ Remark. The definition of a fractional polymorphism can be re-stated in probabilistic terms, as follows. Any fractional polymorphism ω can be seen as a probability distribution over $\mathrm{Pol}^{(k)}(\phi)$. We can then re-write Inequality (2) as follows:

$$\mathbb{E}_{f\sim\omega}[\phi(f(\mathbf{x_1}, \ldots, \mathbf{x_k}))] \;\leq\; \mathrm{avg}\{\phi(\mathbf{x_1}), \ldots, \phi(\mathbf{x_k})\}. \tag{3}$$

This is illustrated in Figure 1, which should be read from left to right. Let ω be a probability distribution on $\mathrm{Pol}^{(k)}(\phi)$ and let $\mathrm{supp}(\omega) = \{f_1, \ldots, f_n\}$. Starting with the m-tuples $\mathbf{x_1}, \ldots, \mathbf{x_k}$, we first apply operations $f_1, \ldots, f_n$ to these tuples componentwise, thus obtaining the m-tuples $\mathbf{x'_1}, \ldots, \mathbf{x'_n}$. Inequality 3 amounts to comparing the average of the values of ϕ applied to the tuples $\mathbf{x_1}, \ldots, \mathbf{x_k}$ with the weighted sum of the values of ϕ applied to the tuples $\mathbf{x'_1}, \ldots, \mathbf{x'_n}$, which is the expected value of $\phi(f(\mathbf{x_1}, \ldots, \mathbf{x_k}))$ as f is drawn from ω.

Examples of fractional polymorphisms include *fractional projections*: a k-ary fractional projection ω is defined by $\omega(\mathrm{e}_1^{(k)}) = \ldots = \omega(\mathrm{e}_k^{(k)}) = \frac{1}{k}$. In this case, Inequality 2 holds trivially with equality. Hence, fractional projections are trivially fractional polymorphisms of all valued constraint languages.

If ω is a fractional polymorphism of ϕ, then we say that ϕ *admits* ω as a fractional polymorphism. We say that a language Γ admits a fractional polymorphism ω if ω is a fractional polymorphism of every cost function $\phi \in \Gamma$. It is easy to see that in this case ω is also a fractional polymorphism of any function Φ arising as an instance of VCSP(Γ). The intuition behind the notion of fractional polymorphism is that it allows one to combine several feasible assignments for an instance of VCSP(Γ), in a randomised way, into a new feasible assignment so that the expected value of the new assignment (non-strictly) improves the average value of the original assignments.

Note that if Γ consists of crisp functions then the k-ary fractional polymorphisms of Γ are all possible probability distributions on $\mathrm{Pol}^{(k)}(\Gamma)$.

A more restricted form of fractional polymorphism was introduced earlier in [25] and is known as a *multimorphism*. This is essentially a k-ary fractional polymorphism where the value of $\omega(f)$ is of the form ℓ/k, where $\ell \in \mathbb{N}$, for every $f \in \mathrm{supp}(\omega)$.

One can specify a k-ary multimorphism as a k-tuple $\mathbf{f} = \langle f_1, \ldots, f_k \rangle$ of k-ary operations f_i on D, where each operation f with $\omega(f) = \ell/k$ for some $\ell > 0$ appears ℓ times, and then the definition simplifies as follows: for all $\mathbf{x_1}, \ldots, \mathbf{x_k} \in \mathrm{Feas}(\phi)$,

$$\sum_{i=1}^{k} \phi(f_i(\mathbf{x_1}, \ldots, \mathbf{x_k})) \ \leq \ \sum_{i=1}^{k} \phi(\mathbf{x_i})\,. \tag{4}$$

Fractional polymorphisms (including the special cases of multimorphisms) have proved to be a valuable tool for identifying tractable valued constraint languages, as we will illustrate in this section. Closely related algebraic objects, known as *weighted polymorphisms*, will be discussed in Section 4.

▶ **Example 19** (Submodularity). For any finite set V, a rational-valued function h defined on subsets of V is called a *set function*. A set function h is called *submodular* if for all subsets S and T of V,

$$h(S \cap T) + h(S \cup T) \ \leq \ h(S) + h(T). \tag{5}$$

Submodular functions are a key concept in operational research and combinatorial optimisation (see, e.g. [38, 96, 109] for extensive information about them). They are often considered to be a discrete analogue of convex functions. Examples of submodular functions include cuts in graphs, matroid rank functions, and entropy functions. There are combinatorial algorithms for minimising submodular functions in polynomial time (see [96, 38], and also [59]).

If we set $D = \{0, 1\}$, then any set function h on V can be associated with a ($|V|$-ary) cost function ϕ defined on the characteristic vectors of subsets of V. The intersection and union operations on subsets correspond to the Min and Max operations on the associated characteristic vectors. Hence h is submodular if and only if the associated cost function ϕ satisfies the following inequality:

$$\phi(\mathrm{Min}(\mathbf{x}_1, \mathbf{x}_2)) + \phi(\mathrm{Max}(\mathbf{x}_1, \mathbf{x}_2)) \ \leq \ \phi(\mathbf{x}_1) + \phi(\mathbf{x}_2)\,.$$

But this means that ϕ admits the 2-ary fractional polymorphism ω_{sub}, defined by $\omega_{sub}(\mathrm{Min}) = \omega_{sub}(\mathrm{Max}) = \frac{1}{2}$. This is equivalent to saying that ϕ admits $\langle \mathrm{Min}, \mathrm{Max} \rangle$ as a multimorphism.

▶ **Example 20** (Generalised Submodularity). Let D be a finite *lattice*, i.e., a partially ordered set, where each pair of elements $\{a, b\}$ has a least upper bound, $\vee(a, b)$, and a greatest lower bound, $\wedge(a, b)$. We denote by Γ_{sub} the set of all cost functions over D that admit $\langle \vee, \wedge \rangle$ as a multimorphism. Using a polynomial-time strongly combinatorial algorithm for minimising submodular functions, it was shown in [25] that Γ_{sub} is tractable when D is a totally ordered lattice (i.e., a *chain*). More general lattices will be discussed in Section 5 and Section 7.

▶ **Example 21** (Max)**.** We denote by Γ_{max} the set of all cost functions (over some fixed finite totally ordered set D) that admit $\langle \mathrm{Max}, \mathrm{Max}\rangle$ as a multimorphism, where $\mathrm{Max} : D^2 \to D$ is the binary operation returning the larger of its two arguments. Note that Γ_{max} includes all monotonic decreasing finite-valued cost functions, as well as some non-monotonic crisp cost functions [25]. It was shown in [25] that Γ_{max} is tractable.

▶ **Example 22** (Min)**.** We denote by Γ_{min} the set of all cost functions (over some fixed finite totally ordered set D) that admit $\langle \mathrm{Min}, \mathrm{Min}\rangle$ as a multimorphism, where $\mathrm{Min} : D^2 \to D$ is the binary operation returning the smaller of its two arguments. The tractability of Γ_{min} was established in [25].

▶ **Example 23** (Bisubmodularity)**.** For a given finite set V, bisubmodular functions are functions defined on pairs of disjoint subsets of V with a requirement similar to Inequality 5 (see [38, 92] for the precise definition). Examples of bisubmodular functions include rank functions of delta-matroids [38].

A property equivalent to bisubmodularity can be defined on cost functions on the set $D = \{0, 1, 2\}$. We define two binary operations Min_0 and Max_0 as follows:

$$\mathrm{Min}_0(x, y) \overset{\text{def}}{=} \begin{cases} 0 & \text{if } 0 \neq x \neq y \neq 0 \\ \mathrm{Min}(x, y) & \text{otherwise} \end{cases},$$

$$\mathrm{Max}_0(x, y) \overset{\text{def}}{=} \begin{cases} 0 & \text{if } 0 \neq x \neq y \neq 0 \\ \mathrm{Max}(x, y) & \text{otherwise} \end{cases}.$$

We denote by Γ_{bis} the set of finite-valued cost functions that admit $\langle \mathrm{Min}_0, \mathrm{Max}_0\rangle$ as a multimorphism. The language Γ_{bis} can be shown to be tractable using the results of [92] (see also [38]).

The definitions of Min_0 and Max_0 still make sense when $D = \{0, 1, 2 \ldots , k\}$, $k \geq 3$. In that case, functions on D that admit $\langle \mathrm{Min}_0, \mathrm{Max}_0\rangle$ as a multimorphism are called *k-submodular*; they were introduced in [54]. The tractability of k-submodular general-valued constraint languages was shown in [73] and will be discussed in Section 5.

▶ **Example 24** (Skew Bisubmodularity)**.** Let $D = \{0, 1, 2\}$. Recall the definition of operations Min_0 and Max_0 from Example 23. We define

$$\mathrm{Max}_1(x, y) \overset{\text{def}}{=} \begin{cases} 1 & \text{if } 0 \neq x \neq y \neq 0 \\ \mathrm{Max}(x, y) & \text{otherwise} \end{cases}.$$

A function $\phi \colon D^m \to \overline{\mathbb{Q}}$ is called *α-bisubmodular* [55], for some real $0 < \alpha \leq 1$, if ϕ admits the fractional polymorphism ω defined by $\omega(\mathrm{Min}_0) = \frac{1}{2}$, $\omega(\mathrm{Max}_0) = \frac{\alpha}{2}$, $\omega(\mathrm{Max}_1) = \frac{1-\alpha}{2}$. Note that 1-bisubmodular functions are (ordinary) bisubmodular functions as defined in Example 23. It is shown in [55] that each distinct value of α is associated with a distinct class of α-bisubmodular functions. The tractability of α-bisubmodular valued constraint languages was shown in [73] and will be discussed in Section 5.

▶ **Example 25** ((Symmetric) Tournament Pair)**.** A binary operation $f : D^2 \to D$ is called a *tournament* operation if (i) f is commutative, i.e., $f(x, y) = f(y, x)$ for all $x, y \in D$; and (ii) f is conservative, i.e., $f(x, y) \in \{x, y\}$ for all $x, y \in D$. The *dual* of a tournament operation is the unique tournament operation g satisfying $x \neq y \Rightarrow g(x, y) \neq f(x, y)$.

A *tournament pair* is a pair $\langle f, g\rangle$, where both f and g are tournament operations. A tournament pair $\langle f, g\rangle$ is called *symmetric* if g is the dual of f.

Let Γ be an arbitrary language that admits a symmetric tournament pair as a multimorphism. It was shown in [24], by a reduction to the minimisation problem for submodular functions (cf. Example 20), that any such Γ is tractable. A different proof of tractability of Γ will be discussed in Section 5. It is shown in [73] that any finite-valued language that admits a symmetric tournament pair multimorphism also admits the submodularity multimorphism with respect to some totally ordered lattice on D (cf. Example 20).

Now let Γ be an arbitrary language that admits any tournament pair as a multimorphism. It was shown in [24], by a reduction to the symmetric tournament pair case, that any such Γ is also tractable. Again, the tractability of Γ will be discussed in more detail in Section 5.

▶ **Example 26** (Tournament in the support)**.** Let Γ be an arbitrary language that admits a binary fractional polymorphism ω such that $\mathrm{supp}(\omega)$ contains a tournament operation, as defined in Example 25. The tractability of Γ was shown in [105], thus generalising Example 25, and will be discussed in Section 5.

▶ **Example 27** (1-Defect)**.** Let b and c be two distinct elements of D and let $(D; <)$ be a partial order which relates all pairs of elements except for b and c. We call $\langle f, g\rangle$, where $f, g : D^2 \to D$ are two binary operations, a *1-defect* if f and g are both commutative and satisfy the following conditions:

- If $\{x, y\} \neq \{b, c\}$, then $f(x, y) = \mathrm{Min}(x, y)$ and $g(x, y) = \mathrm{Max}(x, y)$.
- If $\{x, y\} = \{b, c\}$, then $\{f(x, y), g(x, y)\} \cap \{x, y\} = \emptyset$, and $f(x, y) < g(x, y)$.

The tractability of languages that admit a 1-defect multimorphism was shown in [66], and was used in the classification of the Max-CSP over a four-element set (see Section 5).

▶ **Example 28** (Majority)**.** A ternary operation $f : D^3 \to D$ is called a majority operation if $f(x, x, y) = f(x, y, x) = f(y, x, x) = x$ for all $x, y \in D$.

Let $\mathbf{f} = \langle f_1, f_2, f_3\rangle$ be a triple of ternary operations such that f_1, f_2 and f_3 are all majority operations. Let $\phi : D^m \to \overline{\mathbb{Q}}$ be an m-ary cost function that admits $\mathbf{f}$ as a multimorphism. By Inequality (4), for all $\mathbf{x}, \mathbf{y} \in D^m$, $3\phi(\mathbf{x}) \leq \phi(\mathbf{x}) + \phi(\mathbf{x}) + \phi(\mathbf{y})$ and $3\phi(\mathbf{y}) \leq \phi(\mathbf{y}) + \phi(\mathbf{y}) + \phi(\mathbf{x})$. Therefore, if both $\phi(\mathbf{x})$ and $\phi(\mathbf{y})$ are finite, then we have $\phi(\mathbf{x}) \leq \phi(\mathbf{y})$ and $\phi(\mathbf{y}) \leq \phi(\mathbf{x})$, and hence $\phi(\mathbf{x}) = \phi(\mathbf{y})$. In other words, the range of ϕ is $\{c, \infty\}$, for some finite $c \in \overline{\mathbb{Q}}$.

Let Γ_{Mjty} be the set of all cost functions that admit as a multimorphism some triple $\mathbf{f} = \langle f_1, f_2, f_3\rangle$ of arbitrary ternary majority operations. The tractability of Γ_{Mjty} was shown in [25].

▶ **Example 29** (Minority)**.** A ternary operation $f : D^3 \to D$ is called a minority operation if $f(x, x, y) = f(x, y, x) = f(y, x, x) = y$ for all $x, y \in D$. Let Γ_{Mnty} be the set of cost functions that admit as a multimorphism some triple $\mathbf{f} = \langle f_1, f_2, f_3\rangle$ of arbitrary ternary minority operations. A similar argument to the one in Example 28 shows that the cost functions in Γ_{Mnty} have range $\{c, \infty\}$, for some finite $c \in \overline{\mathbb{Q}}$. The tractability of Γ_{Mnty} was shown in [25].

▶ **Example 30** (MJN)**.** Let $\mathbf{f} = \langle f_1, f_2, f_3\rangle$ be three ternary operations such that f_1 and f_2 are majority operations, and f_3 is a minority operation. Let Γ_{MJN} be the set of cost functions that admit $\mathbf{f}$ as a multimorphism. The tractability of Γ_{MJN} was shown in [74], generalising an earlier tractability result for a specific $\mathbf{f}$ of this form from [25].

▶ **Example 31** (Majority in the support)**.** Let Γ be an arbitrary language admitting a ternary fractional polymorphism ω such that $\mathrm{supp}(\omega)$ contains a majority operation. The tractability of Γ was shown in [105], thus generalising Examples 28 and 30, and will be discussed in Section 5.

Other tractable valued constraint languages defined by fractional polymorphisms include the so-called $L^{\#}$-convex languages [38], as well as the weakly and strongly tree-submodular languages defined in [71]. Hirai [53] recently introduced a framework of submodular functions on modular semilattices (defined by a type of fractional polymorphism) that generalises many examples given above, including standard submodularity, k-submodularity, skew bisubmodularity, and tree submodularity. See [53] for the natural (but somewhat technical) definition of this very general framework.

4 A General Algebraic Theory of Complexity

We have seen in the previous section that many tractable cases of the VCSP can be defined by having a particular fractional polymorphism. The algebraic theory developed in [23, 77, 78, 43] establishes that, in fact, every tractable valued constraint language can be exactly characterised by an associated set of algebraic objects known as weighted polymorphisms, which are different but equivalent to fractional polymorphisms. This extends (parts of) the algebraic theory previously developed for the CSP [19, 17, 62] that has led to significant advances in understanding the landscape of complexity for the CSP over the last 10 years (e.g., [1, 6, 13, 14, 16, 18, 57]). In this section, we will give a brief overview of the main results of this algebraic theory for the VCSP. We refer the reader to [23, 77, 78, 43] for full details and proofs.

First we consider the effect of extending a valued constraint language $\Gamma \subseteq \mathbf{\Phi}_D$ to a possibly larger valued constraint language. We first define and study a notion of *expressibility* for valued constraint languages. This notion has played a key role in the analysis of complexity for the CSP and VCSP [17, 62, 25, 116].

▶ **Definition 32.** We say that an m-ary cost function ϕ is *expressible* over a constraint language Γ if there exists an instance $\Phi \in \text{VCSP}(\Gamma)$ with variables $V = \{x_1, \ldots, x_n, y_1, \ldots, y_m\}$, such that

$$\phi(y_1, \ldots, y_m) = \min_{x_1,\ldots,x_n} \Phi(x_1, \ldots, x_n, y_1, \ldots, y_m) .$$

For a cost function ϕ, we denote by $\text{Opt}(\phi) = \{\mathbf{x} \in \text{Feas}(\phi) \mid \forall \mathbf{y} : \phi(\mathbf{x}) \leq \phi(\mathbf{y})\}$ the *optimality* relation, which contains the tuples on which ϕ is *minimised*. Similarly to $\text{Feas}(\phi)$, we will view $\text{Opt}(\phi)$ both as a relation and as a crisp cost function.

▶ **Definition 33.** A valued constraint language $\Gamma \subseteq \mathbf{\Phi}_D$ is called a *weighted relational clone* if it contains the binary equality relation and the unary empty relation; is closed under expressibility, scaling by non-negative rational constants, addition of rational constants, and operators Feas and Opt. We define $\text{wRelClone}(\Gamma)$ to be the smallest weighted relational clone containing Γ.

▶ **Example 34.** Let $D = \{0, 1\}$ and let u_1 be the unary crisp cost function defined by $u_1(0) = \infty$ and $u_1(1) = 0$. We will show that $\phi_{\textsf{1-in-3}} \in \text{wRelClone}(\{\phi_{\textsf{xor}}, u_1\})$, where $\phi_{\textsf{1-in-3}}$ is from Example 5 and $\phi_{\textsf{xor}}$ is from Example 7.

First observe that $\phi(x, y, z) = \phi_{\textsf{xor}}(x, y) + \phi_{\textsf{xor}}(x, z) + \phi_{\textsf{xor}}(y, z)$ satisfies $\phi(0, 0, 0) = \phi(1, 1, 1) = 3$ and $\phi(x, y, z) = 1$ otherwise. Next observe that $\phi'(x, y, z) = \min_{w \in D}(\phi(x, y, z) + u_1(w) + \phi_{\textsf{xor}}(x, w) + \phi_{\textsf{xor}}(y, w) + \phi_{\textsf{xor}}(z, w))$ satisfies $\phi'(0, 0, 0) = 3$, $\phi'(1, 1, 1) = 6$, $\phi'(x, y, z) = 2$ if there is exactly one 1 among $\{x, y, z\}$ and $\phi'(x, y, z) = 3$ otherwise. Thus, $\phi_{\textsf{1-in-3}} = \text{Opt}(\phi')$ and we have $\phi_{\textsf{1-in-3}} \in \text{wRelClone}(\{\phi_{\textsf{xor}}, u_1\})$.

▶ **Theorem 35** ([23, 43]). *A valued constraint language* Γ *is tractable if* $\mathrm{wRelClone}(\Gamma)$ *is tractable and NP-hard if* $\mathrm{wRelClone}(\Gamma)$ *is NP-hard.*

▶ Remark. For weighted relational clones that are not finitely generated, the rational values are replaced by real values in Definition 1, Definition 33 requires topological closedness, and Theorem 35 holds only up to an arbitrary additive error; we refer the reader to [43] for more details.

We now develop tools that will allow an alternative characterisation of any weighted relational clone. We first recall some basic terminology from universal algebra [10, 99]. We denote by $\mathbf{O}_D$ the set of all finitary operations on D and by $\mathbf{O}_D^{(k)}$ the k-ary operations in $\mathbf{O}_D$. Let $f \in \mathbf{O}_D^{(k)}$ and $g_1, \ldots, g_k \in \mathbf{O}_D^{(\ell)}$. The *superposition* of f and $g_1, \ldots, g_k$ is the ℓ-ary operation $f[g_1, \ldots, g_k]$ such that $f[g_1, \ldots, g_k](x_1, \ldots, x_\ell) = f(g_1(x_1, \ldots, x_\ell), \ldots, g_k(x_1 \ldots, x_\ell))$.

A set $F \subseteq \mathbf{O}_D$ is called a *clone* of operations if it contains all the projections on D and is closed under superposition. It is easy to verify that the set of operations $\mathrm{Pol}(\Gamma)$ is a clone. Clones are actively studied in universal algebra; for example, all (countably many) clones on $D = \{0, 1\}$ are known, but the situation is known to be much more complicated for larger sets D (see, e.g., [10, 99]). We denote by $\mathbf{J}_D$ the clone of all projections on D.

▶ **Definition 36.** A k-ary *weighting* is a function $\omega : \mathbf{O}_D^{(k)} \to \mathbb{Q}$ such that $\omega(f) < 0$ only if f is a projection and $\sum_{f \in \mathbf{O}_D^{(k)}} \omega(f) = 0$.

We denote by $\mathbf{W}_D$ the set of all possible weightings on D and by $\mathbf{W}_D^{(k)}$ the set of k-ary weightings in $\mathbf{W}_D$.

Since a weighting is simply a rational-valued function satisfying certain linear inequalities it can be scaled by any non-negative rational to obtain a new weighting. Similarly, any two weightings of the same arity can be added to obtain a new weighting.

The notion of superposition can also be extended to weightings in a natural way, by forming a superposition with each argument of the weighting, as follows.

▶ **Definition 37.** For any $\omega \in \mathbf{W}_D^{(k)}$ and any $g_1, \ldots, g_k \in \mathbf{O}_D^{(\ell)}$, we define the *superposition* of ω and $g_1, \ldots, g_k$, to be the function $\omega[g_1, \ldots, g_k] : \mathbf{O}_D^{(\ell)} \to \mathbb{Q}$ defined by

$$\omega[g_1, \ldots, g_k](f') \stackrel{\text{def}}{=} \sum_{\substack{f \in \mathbf{O}_D^{(k)} \\ f[g_1, \ldots, g_k] = f'}} \omega(f) \,. \tag{6}$$

It follows immediately from the definition of superposition that the sum of the weights in any superposition $\omega[g_1, \ldots, g_k]$ is equal to the sum of the weights in ω, which is zero, by Definition 36. However, it is not always the case that an arbitrary superposition satisfies the other condition in Definition 36, that negative weights are only assigned to projections. Hence we make the following definition:

▶ **Definition 38.** If the result of a superposition is a valid weighting, then that superposition will be called a *proper* superposition.

For a weighting $\omega \in \mathbf{W}_D^{(k)}$, we denote $\mathrm{supp}(\omega) = \{f \in \mathbf{O}_D^{(k)} \mid \omega(f) > 0\}$.

▶ **Definition 39.** Let W be a non-empty set of weightings on a fixed domain D. We define $\mathrm{supp}(W) = \mathbf{J}_D \cup \bigcup_{\omega \in W} \mathrm{supp}(\omega)$.

We call W a *weighted clone* if it is closed under non-negative scaling, addition of weightings of equal arity, and proper superposition with operations from $\mathrm{supp}(W)$.

For any weighted clone W, $\text{supp}(W)$ is a clone, which we call the *support clone* of W. This easy but important fact has been observed in [77, 78, 105, 43]. We remark that the inclusion of the Opt operator in the definition of weighted relational clones [43] ensures that there is no need for distinguishing the support clone from the positive support clone of weighted clones [77, 78].

▶ **Example 40.** For any clone, C, the set $\mathbf{W}_C$ containing all possible weightings of C is a weighted clone with support clone C.

▶ **Example 41.** For any clone, C, the set $\mathbf{W}_C^0$ containing all *zero-valued* weightings of C is a weighted clone with support clone C. $\mathbf{W}_C^0$ contains exactly one weighting of each possible arity, which assigns the value 0 to all operations in C of that arity.

▶ Remark. For weighted clones that are not finitely generated, the rational weights are replaced by real weights in Definition 36 and Definition 39 requires topological closedness; again, we refer the reader to [43] for more details.

We link weightings and cost functions by the concept of weighted polymorphisms.

▶ **Definition 42** (Weighted Polymorphism). Let $\phi : D^m \to \overline{\mathbb{Q}}$ be a cost function and let ω be a k-ary weighting on D. We call ω a *weighted polymorphism* of ϕ if $\text{supp}(\omega) \subseteq \text{Pol}(\phi)$ and for any $\mathbf{x_1}, \ldots, \mathbf{x_k} \in \text{Feas}(\phi)$

$$\sum_{f \in \text{supp}(\omega)} \omega(f)\phi(f(\mathbf{x_1}, \ldots, \mathbf{x_k})) \ \leq \ 0\,. \tag{7}$$

For a set $W \subseteq \mathbf{W}_D$ which may contain weightings with different support clones over D, we can extend each of these weightings with zeros, as necessary, so that they are weightings of the same support clone C, where C is the smallest clone containing all the clones that are support clones of weightings in W. For any set $W \subseteq \mathbf{W}_D$, we define $\text{wClone}(W)$ to be the smallest weighted clone containing this set of extended weightings obtained from W.

▶ **Definition 43.** For any $W \subseteq \mathbf{W}_D$, we denote by $\text{Imp}(W)$ the set of all cost functions in $\mathbf{\Phi}_D$ which admit all weightings $\omega \in W$ as weighted polymorphisms.[1]

It follows immediately from the definition of a Galois connection [10] that, for any set D, the mappings wPol and Imp form a Galois connection between $\mathbf{W}_D$ and $\mathbf{\Phi}_D$, as illustrated in Figure 2. A characterisation of this Galois connection for finite sets D is given by the following theorem from [23, 43]:

▶ **Theorem 44** (Galois Connection for Valued Constraint Languages [23, 43]).
1. *For any finite D, and any finite $\Gamma \subseteq \mathbf{\Phi}_D$,* $\text{Imp}(\text{wPol}(\Gamma)) = \text{wRelClone}(\Gamma)$.
2. *For any finite D and any finite $W \subseteq \mathbf{W}_D$,* $\text{wPol}(\text{Imp}(W)) = \text{wClone}(W)$.

▶ Remark (fractional vs. weighted polymorphisms). Fractional and weighted polymorphisms are effectively the same things, just written differently. A k-ary fractional polymorphism ω can be transformed into a weighted polymorphism if, in Inequality 2, we move the right-hand side to the left and write $\phi(\mathbf{x_i})$ as $\phi(\mathrm{e}_i^{(k)}(\mathbf{x_1}, \ldots, \mathbf{x_i}, \ldots, \mathbf{x_k}))$. A similar simple manipulation transforms a k-ary weighted polymorphism into a fractional one.

[1] The name Imp is chosen to suggest that such cost functions are *improved* by weightings in W in the sense of the remark and discussion after Definition 18.

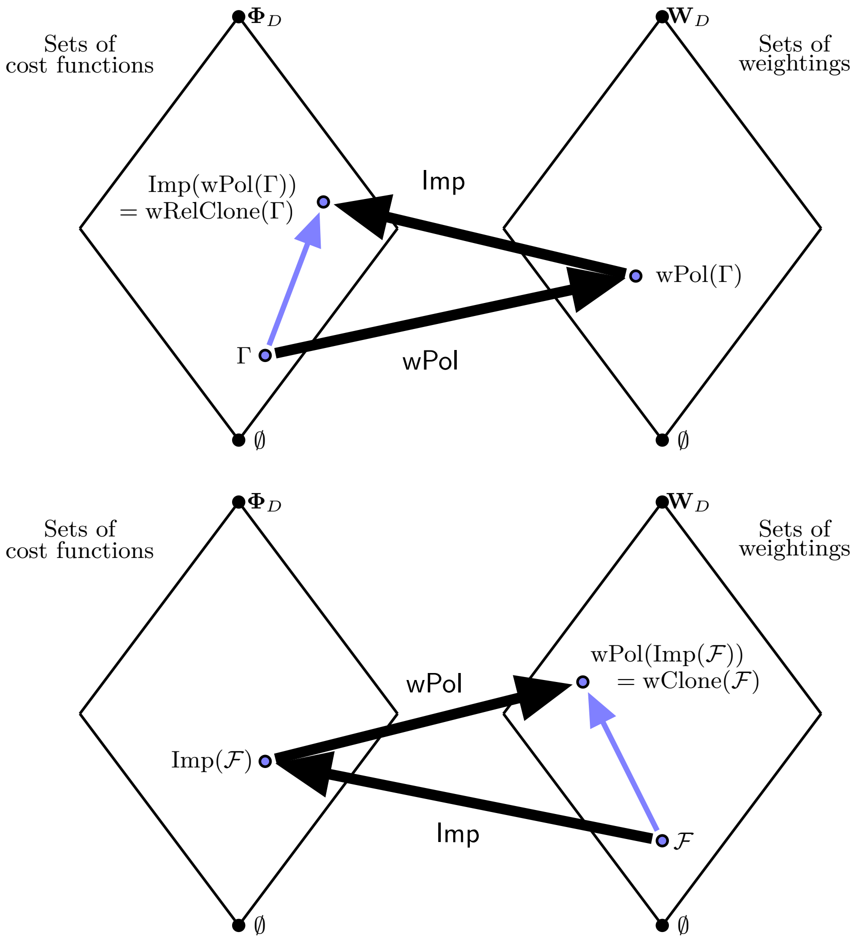


Figure 2 Galois connection between $\mathbf{\Phi}_D$ and $\mathbf{W}_D$.

Each of the two concepts is more convenient to work with depending on the context. Fractional polymorphisms are more convenient to work with when describing properties of operations in the support of fractional polymorphisms, and this relates to algorithmic and complexity consequences discussed in Sections 5 and 6. On the other hand, weighted polymorphisms are more convenient to work with when building an algebraic theory [23, 43, 77, 78].

It follows from Theorem 44 that to identify all tractable valued constraint languages on a finite set D it is sufficient to study the possible weighted clones of weighted polymorphisms on D. This provides an algebraic approach to the identification of tractable cases. In particular, it follows that weighted polymorphisms completely determine the computational complexity of valued constraint languages (the same way polymorphisms determine the computational complexity of crisp languages [62]).

For a valued constraint language Γ we denote by supp(Γ) the support clone of the weighted clone of weighted polymorphisms of Γ; i.e., supp(Γ) = supp(wPol(Γ)).

▶ **Definition 45.** A valued constraint language Γ is called a *core* if all unary operations in supp(Γ) are bijections. Moreover, Γ is called a *rigid core* if the only unary operation in supp(Γ) is the identity operation.

For $d \in D$, let u_d be a unary function on D such that $u_d(d) = 0$ and $u_d(x) = \infty$ if $x \neq d$. Rigid cores are important due to the following result.

▶ **Theorem 46** ([77, 78]).
1. *For any valued constraint language* Γ *there exists a valued constraint language* Γ' *which is a core such that* $\mathrm{VCSP}(\Gamma)$ *and* $\mathrm{VCSP}(\Gamma')$ *are polynomial-time equivalent.*
2. *For any core valued constraint language* Γ *there exists a valued constraint language* Γ' *which is a rigid core such that* $\mathrm{VCSP}(\Gamma)$ *and* $\mathrm{VCSP}(\Gamma')$ *are polynomial-time equivalent.*

The core language Γ' in Theorem 46 (1) is built from Γ as follows. Let $u \in \mathrm{supp}(\Gamma)$ be a unary operation with the minimum size of the set range $U = u(D)$ among all such operations. Then $\Gamma_{|U} = \{\phi_{|U} \mid \phi \in \Gamma\}$ is a core. The rigid core language Γ' in Theorem 46 (2) is $\Gamma' = \Gamma_{|U} \cup \{u_d \mid d \in U\}$, with domain U.

A k-ary operation $f : D^k \to D$ is called *idempotent* if $f(x, \ldots, x) = x$ for every $x \in D$. It is easy to show that Γ is a rigid core if and only if every operation from $\mathrm{supp}(\Gamma)$ is idempotent.

Thus, the intuition behind moving to the rigid core is that (a) one removes labels from the domain that can always be (uniformly) replaced in any solution to an instance without increasing its value, and (b) algebras with only idempotent operation are known to have much more structure (than the general case), leading to the applicability of more powerful algebraic results.

The Galois connection described in Theorem 44 implies the following result.

▶ **Theorem 47.** *Let* Γ *and* Γ' *be two valued constraint languages on the same domain and of finite size. If* $\mathrm{wPol}(\Gamma) \subseteq \mathrm{wPol}(\Gamma')$ *then* $\mathrm{VCSP}(\Gamma')$ *polynomial-time reduces to* $\mathrm{VCSP}(\Gamma)$.

This is a generalisation of the analogous result for the CSP: if $\mathrm{Pol}(\Gamma) \subseteq \mathrm{Pol}(\Gamma')$ for crisp languages Γ and Γ' then $\mathrm{CSP}(\Gamma')$ polynomial-time reduces to $\mathrm{CSP}(\Gamma)$ [62].

The algebraic theory of the CSP extends beyond clones to finite algebras and varieties of algebras (see [19, 17, 85], see also the surveys in [32]), where a variety is a class of algebras over the same signature defined by a set of identities. This extension explains why the complexity of a (crisp) language is determined by the identities, i.e., universally quantified equations, satisfied by its polymorphisms, which is why we usually define the relevant operations by identities. This extension was instrumental in obtaining most state-of-the-art results in this area (e.g. [1, 4, 6, 13, 14, 16, 18, 57, 85]). The basics of this theory were extended to VCSPs in [77, 78], introducing weighted algebras and weighted varieties, which produced a hardness result and a tractability conjecture for VCSPs – we discuss them in Section 6.

While weighted clones were introduced primarily for the understanding of the computational complexity of valued constraint languages, there have been several studies of the purely algebraic structure of weighted clones [113, 112, 104, 63].

5 Algorithms

A curious feature of research into the tractability of constraint languages is that all languages known to be tractable have been shown tractable by using very few algorithmic techniques.

Despite many tractability results concerning crisp languages (i.e., the CSP), only two algorithmic techniques have so far been sufficient, and the applicability of each of them individually has been characterised by specific algebraic conditions.

The first technique is based on enforcing local consistency, which is a natural algorithm for dealing with (crisp) constraints. There are several closely related variants of this algorithm, we will describe one of them. Fix parameters $1 \leq \kappa \leq \ell$. For a given CSP instance, this algorithm starts by adding a new constraint for each subset of variables of size ℓ, the new constraints initially allowing all tuples. Then the algorithm repeatedly discards (i.e., disallows) tuples of labels in the constraints as follows. For every set W of at most κ variables and every pair of constraints ϕ_1, ϕ_2 whose scopes contain W, discard all assignments for ϕ_1 whose restriction on W is inconsistent with the restriction of ϕ_2 to W. Eventually, either all assignments (for at least one constraint) are discarded or else local consistency is established; this procedure takes polynomial time for any fixed D and any fixed κ, ℓ. The former case implies no feasible assignments. One says that a CSP has *relational width* (κ, ℓ) if the latter case implies the existence of a feasible assignment. A CSP has bounded relational width if it has relational width (κ, ℓ) for some κ, ℓ. The power of local consistency, i.e., a precise characterisation of crisp languages that give rise to VCSP instances solvable by some form of local consistency, has recently been established [3, 6, 14, 86] (see also [15, 75]).

▶ **Theorem 48** (Bounded Width [3, 6, 14, 86]). *Let Γ be a crisp language. Then the following are equivalent:*
1. $\mathrm{VCSP}(\Gamma)$ *has bounded relational width;*
2. $\mathrm{VCSP}(\Gamma)$ *has relational width (2,3);*
3. *For all but finitely many k,* $\mathrm{Pol}(\Gamma)$ *contains a k-ary operation satisfying, for all $x, y \in D$,*

$$f(y, x, x, \ldots, x) = f(x, y, x, x, \ldots, x) = f(x, x, \ldots, x, y); \tag{8}$$

4. *For every $k \geq 3$,* $\mathrm{Pol}(\Gamma)$ *contains a k-ary operation satisfying (8);*

A k-ary ($k \geq 2$) idempotent operation satisfying (8) is called a *weak near-unanimity* operation. We remark that that there are many algebraic conditions equivalent to $\mathrm{Pol}(\Gamma)$ containing weak near-unanimity operations of all but finitely many arities. One such condition is that $\mathrm{Pol}(\Gamma)$ contains two weak near-unanimity operations, 3-ary w_3 and 4-ary w_4, such that $w_3(x, x, y) = w_4(x, x, x, y)$ [76].

The second standard algorithmic technique for the CSP is based on the property of having a polynomial-sized representation (a generating set) for the solution set of any instance [16, 57]. Roughly, the algorithm works by starting from the empty set and adding the constraints of the instance one by one while maintaining (in polynomial time) a small enough representation of the current solution set (of feasible assignments). At the end (i.e., after all constraints have been added), either this representation is non-empty and contains a solution to the instance or else there is no solution. In a way, this technique is a generalisation of Gaussian elimination. This algorithm is often called "few subpowers" because it is related to a certain algebraic property to do with the number of subalgebras in powers of an algebra. The power of this algorithm was established in [57]. A k-ary ($k \geq 3$) operation $f : D^k \to D$ is called an *edge* operation if, for all $x, y \in D$,

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

and, for all $4 \leq i \leq k$,

$$f(x, \ldots, x, y, x, \ldots, x) = x \text{ where } y \text{ is in position } i.$$

▶ **Theorem 49** (Few Subpowers [57]). *Let Γ be a crisp language. Then* $\mathrm{VCSP}(\Gamma)$ *is solvable by the few subpowers algorithm if* $\mathrm{Pol}(\Gamma)$ *contains an edge operation.*

The converse to this theorem is true in the following sense: the absence of edge operations from $\mathrm{Pol}(\Gamma)$ implies that the presence of small enough representations is not guaranteed, see [57] for details. Interestingly, the few subpowers algorithm makes use of the actual edge operations in its work (in contrast with bounded width, where the weak near-unanimity operations are used only to guarantee correctness).

For the general VCSP another algorithm, based on linear programming, has been the most thoroughly investigated. Every VCSP instance has a natural integer linear programming (ILP) formulation. Let Φ be a VCSP instance defined by $\Phi(\mathbf{x}) = \sum_{i=1}^{q} \phi_i(\mathbf{x}_i)$, with a set of variables V. For each i, let S_i be the set of variables appearing in $\mathbf{x}_i$; assignments to $\mathbf{x}_i$ are naturally associated with elements in D^{S_i}.

The ILP formulation involves variables $\mu_x(a)$, where $x \in V$ and $a \in D$, and $\lambda_i(\mathbf{s})$, where $1 \le i \le q$ and $\mathbf{s} \in D^{S_i}$. All variables take values in $\{0,1\}$. The intuition is that $\mu_x(a) = 1$ in a solution to the ILP formulation if x is assigned label a in the corresponding solution to Φ. The ILP formulation includes constraints (9c) that ensure that, for any $x \in V$, exactly one variable $\mu_x(a)$ is assigned 1. Similarly, $\lambda_i(\mathbf{s}) = 1$ corresponds to $\mathbf{s}$ being assigned to $\mathbf{x}_i$. The ILP formulation also includes constraints (9b) enforcing consistency between the two types of variables in the ILP. The ILP instance BILP(Φ) associated with Φ is defined as follows:

$$\mathrm{BILP}(\Phi) \overset{\mathrm{def}}{=} \min \quad \sum_{i=1}^{q} \sum_{\mathbf{s} \in D^{S_i}} \phi_i(\mathbf{s})\, \lambda_i(\mathbf{s}) \tag{9a}$$

$$\text{s.t.} \quad \sum_{\mathbf{s} \in D^{S_i} \,\mid\, \mathbf{s}(x)=a} \lambda_i(\mathbf{s}) = \mu_x(a), \quad 1 \le i \le q,\ x \in S_i,\ a \in D \tag{9b}$$

$$\sum_{a \in D} \mu_x(a) = 1, \quad x \in V \tag{9c}$$

$$\lambda_i(\mathbf{s}) = 0, \quad 1 \le i \le q,\ \phi_i(\mathbf{s}) = \infty \tag{9d}$$

Note that terms in (9a) corresponding to (9d) are assumed to be equal to 0.

If we allow the variables in BILP(Φ) to take arbitrary real values in the interval $[0,1]$, we obtain a relaxation called the *basic LP relaxation* (BLP) of Φ, denoted by BLP(Φ). The variables can then be seen as probability distributions on D and D^{S_i}, respectively. The marginalization constraints (9b) impose that μ_x is the marginal of $\lambda_i(\mathbf{s})$, for each constraint and each variable x in the scope of that constraint.

We remark that an LP relaxation of the VCSP, similar or closely related to (9), has been proposed independently by many authors; we refer the reader to [73] and the references therein.

Given a VCSP instance Φ, the optimal value of BLP(Φ), which can be found in polynomial time, is always a lower bound for the optimal value of Φ. We say that BLP *solves* Φ if the optimal value of BLP(Φ) is equal to the optimal value of Φ. Moreover, we say that BLP solves a valued constraint language Γ if BLP solves every instance $\Phi \in \mathrm{VCSP}(\Gamma)$. It is shown in [73] that in all cases where BLP solves Γ, a standard self-reduction method can be used to obtain an assignment that minimises any Φ in VCSP(Γ) in polynomial time. For $d \in D$, let u_d be a unary function on D such that $u(d) = 0$ and $u(x) = \infty$ if $x \neq d$. For rigid cores, the self-reduction method goes through the variables in some order, finding $d \in D$ for the current variable v such that instances Φ and $\Phi + u_d(v)$ have the same optimal value (which can be checked by BLP), updating $\Phi := \Phi + u_d(v)$, and moving to the next variable. At the end, the instance will have a unique feasible assignment whose value is the optimum of the original instance. Hence if BLP solves Γ, then Γ is tractable.

The power of BLP for valued constraint languages was fully characterised in [103]. To state this result, we first introduce some further terminology about operations. A k-ary operation $f : D^k \to D$ is called *symmetric* if for every permutation π on $\{1,\ldots,k\}$, $f(x_1,\ldots,x_k) = f(x_{\pi(1)},\ldots,x_{\pi(k)})$. A fractional polymorphism ω is called symmetric if

$\mathrm{supp}(\omega)$ is non-empty and contains symmetric operations only. Finally, we say that an operation f is *generated* from a set of operations $F \subseteq \mathbf{O}_D$ if $f \in \mathrm{Clone}(F)$.

▶ **Theorem 50** (Power of BLP for Arbitrary Languages [103]). *Let Γ be a valued constraint language. Then the following are equivalent:*

1. *BLP solves Γ;*
2. *For every $k \geq 2$, Γ admits a k-ary symmetric fractional polymorphism;*
3. *For every $k \geq 2$, Γ admits a fractional polymorphism (not necessarily k-ary) ω_k such that supp(ω_k) generates a symmetric k-ary operation.*

Condition (3) has turned out to be very useful for proving the tractability of many valued constraint languages. A binary operation $f : D^2 \to D$ is called a *semilattice* operation if f is associative, commutative, and idempotent. Since any semilattice operation trivially generates symmetric operations of all arities, Theorem 50 shows that any valued constraint language with a binary fractional polymorphism whose support includes a semilattice operation is solvable using the BLP. This immediately implies that all of the following cases are solvable using the BLP, and hence tractable: languages with a (generalised) submodular multimorphism (Example 20), a bisubmodular multimorphism (Example 23), a k-submodular multimorphism for any k (Example 23), a symmetric tournament pair multimorphism (Example 25), or a skew bisubmodular fractional polymorphism (Example 24), or the fractional polymorphisms describing submodularity on modular semilattices [53]. Moreover, a not very difficult argument can be used to show that languages with a 1-defect multimorphism (Example 27) also satisfy condition (3) of Theorem 50 [103], and thus are tractable.

Examples of problems VCSP(Γ) that are tractable, but not solvable by BLP, include (the crisp) 2-Sat problem and 3-Lin-k, the (crisp) problem of solving system of linear equations modulo k with 3 variables per equation, and some languages with a tournament pair multimorphism (Example 25).

Recall that, for any crisp language Γ, any probability distribution on k-ary polymorphisms of Γ is a fractional polymorphism of Γ. Thus, Theorem 50 can be re-stated for crisp languages as follows.

▶ **Theorem 51** (Power of BLP for Crisp Languages [82]). *Let Γ be a crisp constraint language. Then the following are equivalent:*

1. *BLP solves Γ;*
2. *For every $k \geq 2$, $\mathrm{Pol}(\Gamma)$ contains a k-ary symmetric polymorphism;*

It is unknown whether condition (2) in Theorem 51 is decidable, hence the same can be said about the conditions in Theorem 50.

Recent work identified a sufficient instance-based condition for binary CSPs admitting a cyclic polymorphism (cf. Section 6) that are solvable by BLP [12].

For valued constraint languages where the cost functions take only finite values, Theorem 50 has been strengthened further [73].

▶ **Theorem 52** (Power of BLP for Finite-Valued Languages [73]). *Let Γ be a valued constraint language where every cost function takes only finite values. Then the following are equivalent:*

1. *BLP solves Γ;*
2. *For every $k \geq 2$, Γ admits a k-ary symmetric fractional polymorphism;*
3. *For some $k \geq 2$, Γ admits a k-ary symmetric fractional polymorphism;*
4. *Γ admits a binary symmetric fractional polymorphism;*
5. *Γ admits a fractional polymorphism ω such that* $\mathrm{supp}(\omega)$ *generates a symmetric operation.*

In contrast with Theorems 50 and 51, condition (4) in Theorem 52 is decidable because deciding whether Γ admits a fixed-arity symmetric fractional polymorphism (by its definition) amounts to solving a linear program.

The Sherali-Adams hierarchy [97] provides successively tighter LP relaxations of an integer LP. Higher levels of this hierarchy can potentially solve more VCSPs than BLP does. Fix parameters $1 \leq \kappa \leq l$. For a VCSP instance $\Phi(\mathbf{x}) = \sum_{i=1}^{q} \phi_i(\mathbf{x}_i)$, with a set of variables V, its Sherali-Adams relaxation $\mathrm{SA}_{\kappa,\ell}(\Phi)$ is defined as follows. First, we ensure that every non-empty $S \subseteq V$ with $|S| \leq \ell$ appears in some term $\phi_i(\mathbf{x}_i)$, possibly by adding 0-valued constraints (this is similar to how we start the local consistency algorithm described above). The variables are $\lambda_i(\mathbf{s})$ for every $1 \leq i \leq q$ and every tuple $\mathbf{s} \in D^{S_i}$, they take values in the interval $[0, 1]$.

$$\mathrm{SA}_{\kappa,\ell}(\Phi) \overset{\text{def}}{=} \min \quad \sum_{i=1}^{q} \sum_{\mathbf{s} \in D^{S_i}} \phi_i(\mathbf{s})\,\lambda_i(\mathbf{s}) \tag{10a}$$

$$\text{s.t.} \quad \lambda_j(\mathbf{t}) = \sum_{\mathbf{s} \in D^{S_i}\,:\,\mathbf{s}_{|S_j} = \mathbf{t}} \lambda_i(\mathbf{s}), \quad 1 \leq i, j \leq q,\ S_j \subseteq S_i,\ |S_j| \leq \kappa,\ \mathbf{t} \in S_j \tag{10b}$$

$$\sum_{\mathbf{s} \in D^{S_i}} \lambda_i(\mathbf{s}) = 1, \quad 1 \leq i \leq q \tag{10c}$$

$$\lambda_i(\mathbf{s}) = 0, \quad 1 \leq i \leq q,\ \phi_i(\mathbf{s}) = \infty \tag{10d}$$

As with BLP, terms in (10a) corresponding to (10d) are assumed to be equal to 0.

Say that a valued language Γ has *valued relational width* (κ, ℓ) if the optimum value of $\mathrm{SA}_{\kappa,\ell}(\Phi)$ is equal to the optimal value of Φ for every instance Φ of VCSP(Γ). Also, say that Γ has *bounded* valued relational width if it has valued relational width (κ, ℓ) for some κ, ℓ.

▶ **Theorem 53** (Power of SA for Arbitrary Languages [105, 108]). *Let Γ be a valued constraint language. Then the following are equivalent:*
1. *Γ has bounded valued relational width;*
2. *Γ has valued relational width (2,3);*
3. *For every $k \geq 3$, $\mathrm{supp}(\Gamma)$ contains a k-ary (not necessarily idempotent) operation satisfying the weak near-unanimity identities described in (8).*

We remark that condition (3) in the above theorem plays a role not only here and in Theorem 48, it also characterize the so-called robust approximability of CSPs [5].

Examples of languages solvable by SA but *not* by BLP include (the crisp) 2-Sat problem and certain languages admitting a tournament pair multimorphism (cf. Example 25), see [73, Example 5] for more details. The problem 3-Lin-k mentioned above is an example of a tractable problem (solvable by few subpowers) not solvable by SA.

Finally, we also remark that it has recently been shown that if a valued constraint language Γ does not have bounded valued relational width then not only is VCSP(Γ) not solved by a constant level of the Sherali-Adams hierarchy but actually VCSP(Γ) is not solved even by linear levels of the Lasserre semidefinite programming relaxation [107].

6 Complexity Classifications

As mentioned before, the ultimate goal of the research direction that we survey is to obtain complexity classifications: clear descriptions of which VCSPs are tractable and which are not. The usual way to approach this task has been to first restrict constraint languages under consideration to rigid cores, this can be done without loss of generality [77, 78]. After that, one proves an algebraic dichotomy theorem, which states that every rigid core language either expresses some NP-hard problem and therefore is NP-hard itself, or else it has polymorphisms with some nice properties (usually in the form of identities).

We mentioned above that the tractability of constraint languages seems to arise from very few algorithmic techniques. Interestingly, the hardness of constraint languages seems to arise from a single specific problem 1-in-3-Sat (see Example 5)!

A *Taylor* operation is a k-ary ($k \geq 2$) idempotent operation f such that, for each $1 \leq i \leq k$, it satisfies an identity of the form

$$f(\Box_1, \Box_2, \ldots, \Box_k) = f(\triangle_1, \triangle_2, \ldots, \triangle_k) \tag{11}$$

where $\Box_i = x, \triangle_i = y$ and $\Box_s, \triangle_t \in \{x, y\}$ for all s, t. (Note that such identities are the weakest identities that prevent f from being a projection.) The following theorem can be derived from [102] (see also [17]).

▶ **Theorem 54.** *For a crisp constraint language* Γ *on* D *that is a rigid core,*
- *either* $\mathrm{Pol}(\Gamma)$ *contains a Taylor operation,*
- *or else* $\mathrm{wRelClone}(\Gamma)$ *contains a crisp function* ϕ *such that*

$$\mathrm{Opt}(\phi) = \{(x, y, z) \in A^3 \mid \phi_{\text{1-in-3}}(g(x), g(y), g(z)) = 0\} \tag{12}$$

for some $A \subseteq D$ *and some function* $g : A \to \{0, 1\}$.

It follows from Theorem 35 that if a crisp rigid core Γ has no Taylor polymorphism then $\mathrm{VCSP}(\Gamma)$ is NP-hard.

For a crisp language Γ that is a rigid core, having a Taylor polymorphism is equivalent to any one of the following conditions:

1. Γ has a weak near-unanimity polymorphism of some arity $k \geq 2$ [19, 90];
2. Γ has a *cyclic* polymorphism [4], i.e. a k-ary ($k \geq 2$) idempotent polymorphism f such that
$$f(x_1, x_2, \ldots, x_k) = f(x_2, \ldots, x_k, x_1); \tag{13}$$
3. Γ has a 6-ary *Siggers* polymorphism [98], i.e. a 6-ary idempotent polymorphism f that satisfies identities
$$f(x, x, x, x, y, y) = f(x, y, x, y, x, x),$$
$$f(y, y, x, x, x, x) = f(x, x, y, x, y, x);$$
One can visualise these identities by thinking of a three-element complete graph (triangle) whose vertices are $\begin{pmatrix} x \\ x \end{pmatrix}, \begin{pmatrix} x \\ y \end{pmatrix}, \begin{pmatrix} y \\ x \end{pmatrix}$. Then the pairs of vertices that appear in the 6 coordinates (first coordinate on the right and the first coordinate on the left, and so on) give a complete list of edges of the triangle, each edge being a pair of directed edges in opposite directions.
4. Γ has a 4-ary polymorphism f [69] (sometimes also called a Siggers polymorphism) satisfying the identity[2]
$$f(y, x, y, z) = f(x, y, z, x),$$

[2] Using different variables, $f(r, a, r, e) = f(a, r, e, a)$ – mnemonic due to Ryan O'Donnell.

The Algebraic CSP Dichotomy conjecture (originally stated in [17] in a different, but equivalent, form) mentioned in Example 11 is the following.

▶ **Conjecture 55** (Algebraic CSP Dichotomy Conjecture)**.** *A crisp language* Γ *that is a rigid core is tractable if* Γ *has a Taylor polymorphism (or, equivalently, satisfies one of the four conditions above), and it is NP-hard otherwise.*

The hardness part is known, as explained above, and it is the tractability part that is the conjecture. This conjecture refines the original Feder-Vardi dichotomy conjecture [37] by specifying the boundary in algebraic terms. Conjecture 55 was confirmed in many special cases, for example, for crisp languages over two-element sets [95] and three-element sets [13] and for crisp languages containing all unary crisp functions [18, 1]. The conjecture is still open, but widely believed to hold in full generality.

Obviously, a complete classification of VCSPs would include a complete classification for crisp languages. It turns out, however, that the latter classification implies the former one as we now discuss.

A fractional operation ω is said to be cyclic if all operations in $\text{supp}(\omega)$ are cyclic. The following lemma is contained in the proof of Theorem 50 in [78].

▶ **Lemma 56.** *Let* Γ *be a rigid core on a set* D*. Then the following are equivalent:*
1. $\text{supp}(\Gamma)$ *contains a Taylor operation of arity at least 2;*
2. Γ *has a cyclic fractional polymorphism of (some) arity at least 2;*
3. Γ *has a cyclic fractional polymorphism of every prime arity* $p > |D|$*.*

The following theorem is Corollary 51 from [78].

▶ **Theorem 57** ([78])**.** *Let* Γ *be a valued constraint language that is a rigid core. If* $\text{supp}(\Gamma)$ *does not contain a Taylor operation then* Γ *is NP-hard.*

It is actually shown in [78] that if $\text{supp}(\Gamma)$ does not contain a Taylor operation then $\text{wRelClone}(\Gamma)$ contains a (not necessarily crisp) function ϕ satisfying condition (12).

Kozik and Ochremiak state a conjecture (which they attribute to L. Barto) that the above theorem describes all NP-hard valued constraint languages, and all other languages are tractable. Using Lemma 56, we restate their original conjecture via cyclic fractional polymorphisms.

▶ **Conjecture 58** (Algebraic VCSP Dichotomy Conjecture [77, 78])**.** *Let* Γ *be a valued constraint language that is a rigid core. If* Γ *has a cyclic fractional polymorphism of arity at least 2, then* Γ *is tractable, and it is NP-hard otherwise.*

Note that, for fixed D, the problem of checking whether a given finite rigid core Γ has a cyclic fractional polymorphism of some arity can be solved in polynomial time. Indeed, if $p > |D|$ is some fixed prime number, then it is sufficient to check for a cyclic fractional polymorphism of arity p. Such polymorphisms, by definition, are solutions to a system of linear inequalities. Since the number of cyclic operations of arity p on D is constant (because we assume that D is a fixed finite set), the system will have size polynomial in Γ and its feasibility can be decided by linear programming.

Recall that, for a (possibly infinite) crisp language, any probability distribution on polymorphisms (of the same arity) is a fractional polymorphism. Then Theorem 57 is a direct generalisation of the above-mentioned corresponding result for crisp languages. Moreover, Conjecture 58, when restricted to crisp languages, gives precisely Conjecture 55.

For a constraint language Γ, let $\text{Feas}(\Gamma) = \{\text{Feas}(\phi) \mid \phi \in \Gamma\}$. Thus, CSP(Feas($\Gamma$)) is the problem of deciding whether an given instance of VCSP(Γ) has a feasible solution. It is obvious that, for VCSP(Γ) to be tractable, CSP(Feas(Γ)) must also be tractable.

► **Theorem 59** (Classification of General-Valued Languages [72]). *Let Γ be a valued constraint language over domain D that is a rigid core. If the following conditions hold then* VCSP(Γ) *is tractable:*
1. Γ *has a cyclic fractional polymorphism of arity at least 2, and*
2. Feas(Γ) *is tractable.*
Otherwise, Γ is not tractable.

Notice that the above theorem shows that a classification for crisp languages implies the classification for all languages, whether or not the boundary is as predicted by the Algebraic CSP Dichotomy Conjecture. In particular, it follows that Conjecture 55 implies Conjecture 58. Moreover, if the Algebraic CSP Dichotomy Conjecture holds then condition (2) can be removed from Theorem 59, since it would be implied by condition (1).

Theorem 59 implies classification within any class of languages Γ such that the classification for the class of corresponding languages Feas(Γ) is known. For example, this is the case for the class of languages containing all unary crisp functions [18, 1] or for the class of languages over a two- or three-element domain [95, 13].

The necessity of conditions (1-2) in Theorem 59 is clear. To explain why they are sufficient, we need to give a definition.

Let Φ be a VCSP instance over variables V. For each variable $v \in V$, let $D_v = \{d \in D \mid d = \sigma(v)$ for some feasible solution σ for $\Phi\}$. Then the $(1, \infty)$-*minimal instance* $\bar{\Phi}$ associated with Φ is the VCSP instance obtained from Φ by adding, for each $v \in V$, the constraint $u_{D_v}(x_v)$, where u_{D_v} is a crisp function such that $u_{D_v}(d) = 0$ if and only if $d \in D_v$. Note that if Γ is a rigid core and the problem CSP(Feas(Γ)) is tractable, then, for any instance Φ of VCSP(Γ), one can construct the associated $(1, \infty)$-minimal instance in polynomial time. Indeed, to find out whether a given $d \in D$ is in D_v, one only needs to decide whether the CSP instance obtained from Feas(Φ) by adding the constraint $u_d(x_v)$ is satisfiable. Since Γ is a rigid core, one can assume that $u_d \in \Gamma$, so the latter instance is also an instance of CSP(Feas(Γ)).

If Γ is a rigid core satisfying the conditions in Theorem 59 then, for every instance Φ of VCSP(Γ), the optimal value of $BLP(\bar{\Phi})$ is the same as the optimal value of Φ, as proved in [72]. This algorithm (first computing $\bar{\Phi}$ and then finding its optimal value) allows one to find the optimal value of any instance in polynomial time and then find an optimal solution via self-reduction, as discussed earlier.

We would like to point out two surprising features of Theorem 59. The first one is that the algorithm described above that solves all tractable cases uses feasibility checking only as a black-box. The second one is that the proof of Theorem 59 does not involve structural universal algebra used for CSP classifications and also in the proof of Theorem 57.

Tighter tractability conditions (than those given in Theorem 59) are known for a number of important special cases.

For finite-valued languages, condition (2) in Theorem 59 is trivial and can be removed, while condition (1) can be replaced by a much stronger condition.

► **Theorem 60** (Classification of Finite-Valued Languages [106]). *Let Γ be a finite-valued constraint language that is a core. Either Γ has a binary symmetric fractional polymorphism (and hence is solvable by BLP), or else Γ is NP-hard.*

It is shown in [106] that, for any NP-hard finite-valued Γ, $\mathrm{wRelClone}(\Gamma)$ contains a binary function ϕ such that $\mathrm{minarg}(\phi) = \{(a,b),(b,a)\}$ for some distinct $a, b \in D$ (which can be obtained from Γ even without using the operator Opt), thus simulating Max-Cut (see Example 7). We have seen in Example 34 that, on domain $\{0,1\}$, $\mathrm{wRelClone}(\{\phi_{\mathsf{xor}}, u_0, u_1\})$ contains $\phi_{\mathsf{1\text{-}in\text{-}3}}$. Since weighted relational clones are closed under scaling by non-negative rational constants and addition of rational constants, we have $\phi_{\mathsf{1\text{-}in\text{-}3}} \in \mathrm{wRelClone}(\{\phi, u_0, u_1\})$.

Theorem 60 generalises several previous classification results for finite-valued languages. Tractability in these earlier results was often characterised by (more) specific binary symmetric fractional polymorphisms:

- A core $\{0,1\}$-valued language[3] over a two-element set [70, 33], or over a three-element set [64], or including all unary $\{0,1\}$-valued functions [36] is tractable if it is submodular on a chain (cf. Examples 19 and 20), and NP-hard otherwise.
- A core $\{0,1\}$-valued language over a four-element set [66] is tractable if it is submodular on some lattice (cf. Example 20) or 1-defect (cf. Example 27) and NP-hard otherwise.
- A core finite-valued language over a two-element set [25] is tractable if it is submodular (cf. Example 19) and NP-hard otherwise.
- A core finite-valued language over a three-element set [55] is tractable if it is submodular on a chain (cf. Example 20) or skew bisubmodular (cf. Example 24) and NP-hard otherwise.
- A finite-valued language containing all $\{0,1\}$-valued unary cost functions [74] is tractable if it is submodular on a chain (cf. Example 25) and NP-hard otherwise.

Theorem 60 also implies a classification of the so-called Min-0-Ext problems [53].

A tight complete complexity classification for valued constraint languages over a two-element set was established in [25]. Note that on a two-element set there is precisely one majority operation, as defined in Example 28, which we will denote by Mjrty, and precisely one minority operation, as defined in Example 29, which we will denote by Mnrty. There are also precisely two constant operations, which will be denoted Const_0 and Const_1.

▶ **Theorem 61** (Classification of Boolean Languages [25]). *A valued constraint language Γ on $D = \{0,1\}$ is tractable if it admits at least one of the following eight multimorphisms. Otherwise* $\mathrm{wRelClone}(\Gamma)$ *contains $\phi_{\mathsf{1\text{-}in\text{-}3}}$ and Γ is NP-hard.*

1. $\langle \mathrm{Const}_0 \rangle$
2. $\langle \mathrm{Const}_1 \rangle$
3. $\langle \mathrm{Min}, \mathrm{Min} \rangle$,
4. $\langle \mathrm{Max}, \mathrm{Max} \rangle$,
5. $\langle \mathrm{Min}, \mathrm{Max} \rangle$,
6. $\langle \mathrm{Mjrty}, \mathrm{Mjrty}, \mathrm{Mjrty} \rangle$,
7. $\langle \mathrm{Mnrty}, \mathrm{Mnrty}, \mathrm{Mnrty} \rangle$,
8. $\langle \mathrm{Mjrty}, \mathrm{Mjrty}, \mathrm{Mnrty} \rangle$.

Let us compare Theorem 61 with a classification of crisp Boolean languages, originally established by Schaefer in [95] and restated here using polymorphisms (see, e.g. [22]): A crisp constraint language on $D = \{0,1\}$ is tractable if it admits one of the following six polymorphisms: Const_0, Const_1, Min, Max, Mjrty, Mnrty; otherwise it is NP-hard. These six tractable cases are covered by cases (1-4), (6), and (7) in Theorem 61. The six cases correspond to sets of Boolean relations that are 0-valid, or 1-valid, or expressible by Horn

[3] $\{0,1\}$-valued languages correspond to Max-CSPs, cf. Example 13.

clauses, dual Horn clauses, 2-clauses, or linear equations over the field with 2 elements, respectively.

The hardness part of Theorem 61 can be rederived using the algebraic theory described in Section 4; see [31, 23] for details. We remark that if we restrict to core Boolean valued constraint languages, the first two cases in Theorem 61 disappear as those languages are not cores (and in fact are solvable trivially). The original proof of Theorem 61 identified ϕ_{nae} (Example 6) and ϕ_{xor} (Example 7) as sources of hardness [25]. However, for rigid cores we have seen in Example 34 how to obtain $\phi_{\mathsf{1\text{-}in\text{-}3}}$ from ϕ_{xor} and it is well known that $\phi_{\mathsf{1\text{-}in\text{-}3}}$ and ϕ_{nae} can express each other in this case (see, e.g. [95]).

Another general complexity classification result concerns languages that contain all $\{0,1\}$-valued unary cost functions. Note that a fractional polymorphism ω is called *conservative* if $f(x_1,\ldots,x_k) \in \{x_1,\ldots,x_k\}$ for all $f \in \operatorname{supp}(\omega)$.

▶ **Theorem 62** (Classification of Conservative Languages [74]). *Let Γ be a valued constraint language on a set D such that Γ contains all $\{0,1\}$-valued unary cost functions on D. Then either Γ admits a conservative binary multimorphism $\langle f_1, f_2\rangle$ and a conservative ternary multimorphism $\langle f_1', f_2', f_3'\rangle$ and there is a family M of 2-element subsets of D, such that:*

1. *for every $\{a,b\} \in M$, $\langle f_1, f_2\rangle$ restricted to $\{a,b\}$ is a symmetric tournament pair (see Example 25), and*
2. *for every $\{a,b\} \notin M$, $\langle f_1', f_2', f_3'\rangle$ restricted to $\{a,b\}$ is an MJN multimorphism (see Example 30),*

in which case Γ is tractable, or else Γ is NP-hard.

It is shown in [108] that the tractable cases in Theorem 62 can be equivalently characterised by the condition that $\operatorname{supp}(\Gamma)$ contains a majority operation (see also Example 31).

The original algorithm for solving the tractable cases identified in Theorem 62 was similar to (and in fact inspired) the general algorithm for tractable VCSPs: after establishing some sort of local consistency, any instance admits a symmetric tournament pair multimorphism [74] and is thus solvable using BLP. It was shown in [108] that all tractable languages identified in Theorem 62 in fact have valued relational width (2,3).

Recall the Min-Cost-Hom and Min-Sol problems discussed in Examples 14 and 16 respectively. Recall that a Min-Cost-Hom problem corresponds to VCSP(Γ) for some language Γ containing only crisp cost functions and unary cost functions. We now briefly describe the classification results so far obtained for these problems that give more information than the general Theorem 59. It may seem that the Min-Cost-Hom framework is rather more restrictive than the (general-valued) VCSP. However, it was shown in [26], by adapting the main result of [20], that for every problem VCSP(Γ), where Γ is finite, there is a polynomial-time equivalent Min-Cost-Hom problem, VCSP(Γ'), where Γ' contains only a single crisp binary function and a single finite-valued unary function. This mirrors a similar reduction from the general CSP (Example 11) to the digraph homomorphism problem (Example 12) which was first established in [37].

The complexity classification for Min-Cost-Hom for languages containing all unary cost functions was established in [100]. The tractable case can be reduced, after a preprocessing step using local consistency techniques, to a certain problem on perfect graphs known to be solvable in polynomial time using linear programming [48]. For the special case of digraphs (i.e., when the only non-unary cost function allowed is a single binary crisp cost function), a complexity classification in graph-theoretic terms was obtained in [52]. The classification of Min-Cost-Hom for languages containing all unary crisp cost functions was initially studied in [101] and fully established in [110].

The complexity of Min-Cost-Hom for all languages over a three-element set was classified in [111]. The only tractable cases either admit a fractional polymorphism with a semilattice operation in its support or a certain type of tournament pair. The former case is tractable using BLP by Theorem 50 and the latter case is tractable using a reduction to the result in [100] discussed above.

Min-Sol problems are Min-Cost-Hom problems where the only unary cost function in Γ is a specific injective and finite-valued cost function. The classification of Min-Sol problems for various special cases was established in [70, 65, 68, 110]. It was shown in [108] that every Min-Sol problem satisfies either the conditions of Theorem 57 or the conditions of Theorem 53, thus providing a full dichotomy for such problems. That dichotomy, as well as Theorem 62, are corollaries of the following result, which does not follow from Theorem 59.

▶ **Theorem 63** (Classification of General-Valued Languages with an Injective Function [108]). *Let Γ be a valued constraint language Γ on D that is a rigid core. Assume that Γ can express a unary finite-valued cost function $u : D \to \mathbb{Q}$ that is injective, i.e. $u(a) \neq u(b)$ for any $a \neq b \in D$. Then either Γ has bounded valued relational width (and hence is solvable by SA), or Γ is NP-hard.*

7 The Oracle Model

In this paper we have assumed that the objective function in our problem is represented as a sum of functions each defined on some subset of the variables. There is a rich tradition in combinatorial optimisation of studying problems where the objective function is represented instead by a value-giving *oracle*. In this model a problem is tractable if it can be solved in polynomial time using only polynomially many queries to the oracle (where the polynomial is in the number of variables). Note that any query to the oracle can be simulated in linear time in the VCSP model. Hence, a tractability result (for a class of functions) in the oracle model automatically transfers to the VCSP model, while hardness results automatically transfer in the opposite direction.

One class of functions that has received particular attention in the oracle model is the class of submodular functions (cf. Example 19). There are several known algorithms for minimising a (finite-valued) submodular function using only a polynomial number of calls to a value-giving oracle (see [58, 59, 88, 96]).

The fastest general (strongly polynomial) algorithm [88] runs in $O(n^3 \log^2 n \cdot EO + n^4 \log^{O(1)} n)$ time, where EO is the time for function evaluation by oracle. However, for some submodular valued constraint languages Γ, VCSP(Γ) can be solved more efficiently than by using these general approaches. For example, the (submodular) language Γ_{cut} defined in Example 9 can be solved in cubic time using the Min-Cut-based algorithm described in Example 9. A similar efficient approach can be used for all languages that are expressible over Γ_{cut}. However, it was shown in [115, 117] that not all submodular functions are expressible over Γ_{cut}, so this approach cannot be directly extended to solve arbitrary submodular VCSP instances. It is currently an open question whether the minimisation problem for submodular functions defined by sums of bounded arity submodular functions in the VCSP model is easier than general submodular function minimisation in the oracle model.

Other classes of finite-valued functions that can be efficiently minimised in the oracle model include bisubmodular and α-bisubmodular functions (Examples 23 and 24) [39, 41, 92, 42, 56], functions with a 1-defect multimorphism (Example 27) [66], and functions that are submodular on certain lattices (Example 20) [40, 79, 81]. The complexity of submodular function minimisation in the oracle model over arbitrary non-distributive lattices is still unknown (in the VCSP model, all such language are tractable, by Theorem 50).

The following general problem was mentioned in [55, 66, 103]: which fractional/weighted polymorphisms ω are sufficient to guarantee an efficient minimization algorithm, in the value-oracle model, for the class of functions $\mathrm{Imp}(\omega)$? This problem can be asked only for classes of finite-valued functions, or for classes of general-valued functions – in the latter case, some representation of feasible tuples should be part of input. Natural candidates for which the question is open include the k-submodularity multimorphism for $k \geq 3$ from Example 23 and submodularity multimorphisms on many lattices from Example 20.

8 Conclusions and Future Directions

We have shown that the valued constraint satisfaction problem is a powerful general framework that can be used to express many standard combinatorial optimisation problems. The general problem is NP-hard, but there are many special cases that have been shown to be tractable. In particular, by considering restrictions on the cost functions we allow in problem instances, we have identified a range of different sets of cost functions that ensure tractability.

These restricted sets of cost functions are referred to as valued constraint languages, and we have described in Section 4 the very general algebraic techniques now being developed to classify the complexity of these languages.

This classification is still incomplete. In fact, even in the special case of the CSP (Example 11), where all cost functions take only the values 0 or ∞, there is still no complete classification of complexity for the corresponding constraint languages. This problem has been studied for many years, beginning with the seminal work of Feder and Vardi who conjectured that any such language will be either tractable or NP-complete [37]. This conjecture is still unresolved. However, the Algebraic Dichotomy conjecture (see Conjecture 55) specifies the boundary between tractable and NP-hard languages, and it has been proved in many important cases. It was stated in [61] that "it is desirable to develop the algebraic theory of VCSPs to the point where one could make a credible algebraic dichotomy conjecture for the VCSP, in order to have a specific target to aim at." Now, only two years after [61], this algebraic theory has been developed, the VCSP dichotomy conjecture stated (see Conjecture 58) and proved to be equivalent to Conjecture 55 (see Theorem 59).

It is an interesting open question to find tight(er) conditions characterising tractable cases, both for the general case and for important special cases. For example, the tractability condition for finite-valued languages (see Theorem 60), is considerably tighter then the condition from Conjecture 58. It could probably be made tighter still: for $|D| = 2, 3$, tight (i.e., irreducible) descriptions are given in [25, 55], respectively.

Another interesting open question is to improve the efficiency of the general algorithm for tractable VCSPs described after Theorem 59. The current algorithm involves solving the feasibility problem for a given instance many (specifically, $O(|V| \cdot |D|)$) times, perhaps this can be improved.

In this survey, we looked only at solving VCSPs to optimality. There is plenty of literature on (in)approximability of CSP-related problems (see, e.g., the recent survey [89]), but many problems about the approximability of VCSPs are still open: for example, the open problems from [35], stated there for $\{0, 1\}$-valued CSPs, make perfect sense for arbitrary VCSPs. Note that the equivalence of maximisation and minimisation as described in Example 13 does not work when dealing with approximation properties. Fixed-parameter approximability for $\{0, 1\}$-valued CSPs was considered in [9]. An interesting application of VCSP classifications to study fixed-parameter tractability appeared in [60].

In this survey, we looked at VCSPs over *finite* domains. There is a significant line of research dealing with CSPs over *infinite* domains – see [94, 7] and also [8]. The complexity of VCSPs over infinite domains is almost unexplored beyond CSPs.

The algebraic theory of the VCSP presented in Section 4 is based on the notion of a weighted clone. Not much is known about weighted clones, and this direction is wide open for purely algebraic investigation. Some specific open problems include the (possible) description of weighted clones for $D = \{0, 1\}$, the identification of minimal weighted clones, and the investigation of classes of weighted clones supported by a given ordinary clone. Some partial results are presented in two MSc theses [113, 112] and in [104, 63].

In this survey we have focused on the complexity of valued constraint satisfaction problems with restricted constraint languages. It is also possible to ensure tractability by restricting the structure of the constraint scopes – so-called *structural* restrictions [46, 47, 91]. Combining structural restrictions with language restrictions leads to so-called *hybrid* restrictions, and these provide a promising source of new tractable cases [27, 28] which has so far been very little explored – see survey [29].

Acknowledgements. We would like to thank Pete Jeavons for his useful comments that improved the presentation of this paper.

References

1 Libor Barto. The dichotomy for conservative constraint satisfaction problems revisited. In *Proceedings of the 26th IEEE Symposium on Logic in Computer Science (LICS'11)*, pages 301–310. IEEE Computer Society, 2011. doi:10.1109/LICS.2011.25.

2 Libor Barto. Constraint satisfaction problem and universal algebra. *ACM SIGLOG News*, 1(2):14–24, 2014. doi:10.1145/2677161.2677165.

3 Libor Barto. The collapse of the bounded width hierarchy. *J. Log. Comput.*, 26(3):923–943, 2016. doi:10.1093/logcom/exu070.

4 Libor Barto and Marcin Kozik. Absorbing Subalgebras, Cyclic Terms, and the Constraint Satisfaction Problem. *Logical Methods in Computer Science*, 8(1), 2012. doi:10.2168/LMCS-8(1:7)2012.

5 Libor Barto and Marcin Kozik. Robust satisfiability of constraint satisfaction problems. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19-22, 2012*, pages 931–940, 2012.

6 Libor Barto and Marcin Kozik. Constraint Satisfaction Problems Solvable by Local Consistency Methods. *Journal of the ACM*, 61(1), 2014. Article No. 3. doi:10.1145/2556646.

7 Manuel Bodirsky. Complexity classification in infinite-domain constraint satisfaction. *CoRR*, abs/1201.0856, 2012. URL: http://arxiv.org/abs/1201.0856.

8 Manuel Bodirsky. Constraint Satisfaction Problems over Numeric Domains. In Andrei Krokhin and Stanislav Živný, editors, *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*, pages 79–111. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2017. doi:10.4230/DFU.Vol7.15301.79.

9 Édouard Bonnet, László Egri, and Dániel Marx. Fixed-parameter approximability of Boolean MinCSPs. *CoRR*, abs/1601.04935, 2016.

10 Ferdinand Börner. Basics of Galois connections. In *Complexity of Constraints*, volume 5250 of *Lecture Notes in Computer Science*, pages 38–67. Springer, 2008.

11 Yuri Boykov, Olga Veksler, and Ramin Zabih. Markov Random Fields with Efficient Approximations. In *1998 Conference on Computer Vision and Pattern Recognition (CVPR'98)*, pages 648–655, 1998. doi:10.1109/CVPR.1998.698673.

12 Jonah Brown-Cohen and Prasad Raghavendra. Correlation Decay & Tractability of CSPs. In *Proceedings of the 43rd International Colloquium on Automata, Languages and Programming (ICALP'16)*, 2016.

13 Andrei Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-element set. *Journal of the ACM*, 53(1):66–120, 2006. `doi:10.1145/1120582.1120584`.

14 Andrei Bulatov. Bounded relational width. Unpublished manuscript, 2009.

15 Andrei Bulatov. Graphs of relational structures: restricted colours. In *LICS'16*, 2016. To appear.

16 Andrei Bulatov and Víctor Dalmau. A simple algorithm for Mal'tsev constraints. *SIAM Journal on Computing*, 36(1):16–27, 2006. `doi:10.1137/050628957`.

17 Andrei Bulatov, Andrei Krokhin, and Peter Jeavons. Classifying the Complexity of Constraints using Finite Algebras. *SIAM Journal on Computing*, 34(3):720–742, 2005. `doi:10.1137/S0097539700376676`.

18 Andrei A. Bulatov. Complexity of conservative constraint satisfaction problems. *ACM Transactions on Computational Logic*, 12(4), 2011. Article 24. `doi:10.1145/1970398.1970400`.

19 Andrei A. Bulatov and Peter G. Jeavons. Algebraic structures in combinatorial problems. Technical Report MATH-AL-4-2001, Technische Universität Dresden, 2001.

20 Jakub Bulin, Dejan Delic, Marcel Jackson, and Todd Niven. A finer reduction of constraint problems to digraphs. *Logical Methods in Computer Science*, 11(4), 2015. `doi:10.2168/LMCS-11(4:18)2015`.

21 David Cohen, Martin Cooper, Peter Jeavons, and Andrei Krokhin. Supermodular Functions and the Complexity of MAX-CSP. *Discrete Applied Mathematics*, 149(1-3):53–72, 2005. `doi:10.1016/j.dam.2005.03.003`.

22 David Cohen and Peter Jeavons. The complexity of constraint languages. In F. Rossi, P. van Beek, and T. Walsh, editors, *The Handbook of Constraint Programming.* Elsevier, 2006.

23 David A. Cohen, Martin C. Cooper, Páidí Creed, Peter Jeavons, and Stanislav Živný. An algebraic theory of complexity for discrete optimisation. *SIAM Journal on Computing*, 42(5):915–1939, 2013. URL: `http://zivny.cz/publications/cccjz13sicomp-preprint.pdf`, `doi:10.1137/130906398`.

24 David A. Cohen, Martin C. Cooper, and Peter G. Jeavons. Generalising submodularity and Horn clauses: Tractable optimization problems defined by tournament pair multimorphisms. *Theoretical Computer Science*, 401(1-3):36–51, 2008. `doi:10.1016/j.tcs.2008.03.015`.

25 David A. Cohen, Martin C. Cooper, Peter G. Jeavons, and Andrei A. Krokhin. The Complexity of Soft Constraint Satisfaction. *Artificial Intelligence*, 170(11):983–1016, 2006. `doi:10.1016/j.artint.2006.04.002`.

26 David A. Cohen, Martin C. Cooper, Peter G. Jeavons, Andrei A. Krokhin, Robert Powell, and Stanislav Živný. Binarisation for Valued Constraint Satisfaction Problems, August 2016. arXiv:1608.01628. URL: `http://arxiv.org/abs/1608.01628`.

27 Martin C. Cooper and Stanislav Živný. Hybrid tractability of valued constraint problems. *Artificial Intelligence*, 175(9-10):1555–1569, 2011. URL: `http://zivny.cz/publications/cz11aij-preprint.pdf`, `doi:10.1016/j.artint.2011.02.003`.

28 Martin C. Cooper and Stanislav Živný. Tractable triangles and cross-free convexity in discrete optimisation. *Journal of Artificial Intelligence Research*, 44:455–490, 2012. URL: `http://zivny.cz/publications/cz12jair-preprint.pdf`, `doi:10.1613/jair.3598`.

29 Martin C. Cooper and Stanislav Živný. Hybrid Tractable Classes of Constraint Problems. In Andrei Krokhin and Stanislav Živný, editors, *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*, pages 113–135.

Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2017. `doi:10.4230/DFU.Vol7.15301.113`.

30 Yves Crama and Peter L. Hammer. *Boolean Functions – Theory, Algorithms, and Applications*. Cambridge University Press, 2011.

31 Páidí Creed and Stanislav Živný. On minimal weighted clones. In *Proceedings of the 17th International Conference on Principles and Practice of Constraint Programming (CP'11)*, volume 6876 of *Lecture Notes in Computer Science*, pages 210–224. Springer, 2011. URL: `http://zivny.cz/publications/cz11cp-mwc-preprint`, `doi:10.1007/978-3-642-23786-7_18`.

32 Nadaia Creignou, Phokion G. Kolaitis, and Heribert Vollmer, editors. *Complexity of Constraints: An Overview of Current Research Themes*, volume 5250 of *Lecture Notes in Computer Science*. Springer, 2008. `doi:10.1007/978-3-540-92800-3`.

33 Nadia Creignou, Sanjeev Khanna, and Madhu Sudan. *Complexity Classification of Boolean Constraint Satisfaction Problems*, volume 7 of *SIAM Monographs on Discrete Mathematics and Applications*. SIAM, 2001.

34 Elias Dahlhaus, David S. Johnson, Christos H. Papadimitriou, Paul D. Seymour, and Mihalis Yannakakis. The Complexity of Multiterminal Cuts. *SIAM Journal on Computing*, 23(4):864–894, 1994. `doi:10.1137/S0097539792225297`.

35 Víctor Dalmau and Andrei Krokhin. Robust satisfiability for CSPs: Hardness and algorithmic results. *TOCT*, 5(4):15, 2013.

36 Vladimir Deineko, Peter Jonsson, Mikael Klasson, and Andrei Krokhin. The approximability of Max CSP with fixed-value constraints. *Journal of the ACM*, 55(4), 2008. Article 16. `doi:10.1145/1391289.1391290`.

37 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. `doi:10.1137/S0097539794266766`.

38 Satoru Fujishige. *Submodular Functions and Optimization*, volume 58 of *Annals of Discrete Mathematics*. North-Holland, Amsterdam, 2nd edition, 2005.

39 Satoru Fujishige and Satoru Iwata. Bisubmodular Function Minimization. *SIAM Journal on Discrete Mathematics*, 19(4):1065–1073, 2005. `doi:10.1137/S0895480103426339`.

40 Satoru Fujishige, Tamás Király, Kazuhisa Makino, Kenjiro Takazawa, and Shin-ichi Tanigawa. Minimizing submodular functions on diamonds via generalized fractional matroid matchings. Technical Report RIMS-1812, Kyoto University, 2015.

41 Satoru Fujishige and Shin-ichi Tanigawa. Polynomial combinatorial algorithms for skew-bisubmodular function minimization. Technical Report RIMS-1837, Kyoto University, 2015.

42 Satoru Fujishige, Shin-ichi Tanigawa, and Yuichi Yoshida. Generalized skew bisubmodularity: A characterization and a min-max theorem. *Discrete Optimization*, 12:1–9, 2014. `doi:10.1016/j.disopt.2013.12.001`.

43 Peter Fulla and Stanislav Živný. A Galois Connection for Valued Constraint Languages of Infinite Size. *ACM Transactions on Computation Theory*, 8(3), 2016. Article No. 9. URL: `http://www.cs.ox.ac.uk/Stanislav.Zivny/homepage/publications/fz16toct-preprint.pdf`, `doi:10.1145/2898438`.

44 Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, 1979.

45 Andrew V. Goldberg and Robert Endre Tarjan. A New Approach to the Maximum Flow Problem. *Journal of the ACM*, 35(4):921–940, 1988. `doi:10.1145/48014.61051`.

46 Georg Gottlob, Gianluigi Greco, and Francesco Scarcello. Tractable Optimization Problems through Hypergraph-Based Structural Restrictions. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming (ICALP'09), Part II*, volume 5556 of *Lecture Notes in Computer Science*, pages 16–30. Springer, 2009. doi:10.1007/978-3-642-02930-1_2.

47 Martin Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *Journal of the ACM*, 54(1):1–24, 2007. doi:10.1145/1206035.1206036.

48 Martin Grötschel, Laszlo Lovasz, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer, 1988.

49 Gregory Gutin, Pavol Hell, Arash Rafiey, and Anders Yeo. A dichotomy for minimum cost graph homomorphisms. *European Journal of Combinatorics*, 29(4):900–911, 2008. doi:10.1016/j.ejc.2007.11.012.

50 Pavol Hell and Jaroslav Nešetřil. *Graphs and Homomorphisms*. Oxford University Press, 2004.

51 Pavol Hell and Jaroslav Nešetřil. Colouring, constraint satisfaction, and complexity. *Computer Science Review*, 2(3):143–163, 2008. doi:10.1016/j.cosrev.2008.10.003.

52 Pavol Hell and Arash Rafiey. The Dichotomy of Minimum Cost Homomorphism Problems for Digraphs. *SIAM Journal on Discrete Mathematics*, 26(4):1597–1608, 2012. doi:10.1137/100783856.

53 Hiroshi Hirai. Discrete convexity and polynomial solvability in minimum 0-extension problems. *Math. Program.*, 155(1-2):1–55, 2016. doi:10.1007/s10107-014-0824-7.

54 Anna Huber and Vladimir Kolmogorov. Towards Minimizing k-Submodular Functions. In *Proceedings of the 2nd International Symposium on Combinatorial Optimization (ISCO'12)*, volume 7422 of *Lecture Notes in Computer Science*, pages 451–462. Springer, 2012. doi:10.1007/978-3-642-32147-4_40.

55 Anna Huber, Andrei Krokhin, and Robert Powell. Skew bisubmodularity and valued CSPs. *SIAM Journal on Computing*, 43(3):1064–1084, 2014. doi:10.1137/120893549.

56 Anna Huber and Andrei A. Krokhin. Oracle tractability of skew bisubmodular functions. *SIAM J. Discrete Math.*, 28(4):1828–1837, 2014. doi:10.1137/130936038.

57 Pawel M. Idziak, Petar Markovic, Ralph McKenzie, Matthew Valeriote, and Ross Willard. Tractability and learnability arising from algebras with few subpowers. *SIAM Journal on Computing*, 39(7):3023–3037, 2010. doi:10.1137/090775646.

58 Satoru Iwata. Submodular Function Minimization. *Mathematical Programming*, 112(1):45–64, 2008. doi:10.1007/s10107-006-0084-2.

59 Satoru Iwata and James B. Orlin. A Simple Combinatorial Algorithm for Submodular Function Minimization. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'09)*, pages 1230–1237, 2009. doi:10.1145/1496770.1496903.

60 Yoichi Iwata, Magnus Wahlström, and Yuichi Yoshida. Half-integrality, LP-branching and FPT algorithms. *CoRR*, abs/1310.2841, 2014. URL: http://arxiv.org/abs/1310.2841v2.

61 Peter Jeavons, Andrei Krokhin, and Stanislav Živný. The complexity of valued constraint satisfaction. *Bulletin of the European Association for Theoretical Computer Science (EATCS)*, 113:21–55, 2014. URL: http://zivny.cz/publications/jkz14.pdf.

62 Peter G. Jeavons, David A. Cohen, and Marc Gyssens. Closure Properties of Constraints. *Journal of the ACM*, 44(4):527–548, 1997. doi:10.1145/263867.263489.

63 Peter G. Jeavons, Andrius Vaicenavičius, and Stanislav Živný. Minimal weighted clones with Boolean support. In *Proceedings of the 46th IEEE International Symposium on Multiple-Valued Logic (ISMVL'16)*. IEEE, 2016. URL: http://zivny.cz/publications/jvz15ismvl-preprint.pdf.

64 Peter Jonsson, Mikael Klasson, and Andrei Krokhin. The Approximability of Three-valued MAX CSP. *SIAM Journal on Computing*, 35(6):1329–1349, 2006. doi:10.1137/S009753970444644X.

65 Peter Jonsson, Fredrik Kuivinen, and Gustav Nordh. MAX ONES Generalized to Larger Domains. *SIAM Journal on Computing*, 38(1):329–365, 2008. `doi:10.1137/060669231`.

66 Peter Jonsson, Fredrik Kuivinen, and Johan Thapper. Min CSP on Four Elements: Moving Beyond Submodularity. In *Proceedings of the 17th International Conference on Principles and Practice of Constraint Programming (CP'11)*, volume 6876 of *Lecture Notes in Computer Science*, pages 438–453. Springer, 2011. `doi:10.1007/978-3-642-23786-7_34`.

67 Peter Jonsson and Gustav Nordh. Introduction to the MAXIMUM SOLUTION Problem. In *Complexity of Constraints*, volume 5250 of *Lecture Notes in Computer Science*, pages 255–282. Springer, 2008. `doi:10.1007/978-3-540-92800-3_10`.

68 Peter Jonsson and Johan Thapper. Approximability of the maximum solution problem for certain families of algebras. In *Proceedings of the 4th International Computer Science Symposium in Russia (CSR'09)*, volume 5675 of *Lecture Notes in Computer Science*, pages 215–226. Springer, 2009. `doi:10.1007/978-3-642-03351-3_21`.

69 Keith Kearnes, Petar Marković, and Ralph McKenzie. Optimal strong Mal'cev conditions for omitting type 1 in locally finite varieties. *Algebra Universalis*, 72(1):91–100, 2014. `doi:10.1007/s00012-014-0289-9`.

70 Sanjeev Khanna, Madhu Sudan, Luca Trevisan, and David Williamson. The approximability of constraint satisfaction problems. *SIAM Journal on Computing*, 30(6):1863–1920, 2000. `doi:10.1137/S0097539799349948`.

71 Vladimir Kolmogorov. Submodularity on a tree: Unifying $L^\sharp$-convex and bisubmodular functions. In *Proceedings of the 36th International Symposium on Mathematical Foundations of Computer Science (MFCS'11)*, volume 6907 of *Lecture Notes in Computer Science*, pages 400–411. Springer, 2011. `doi:10.1007/978-3-642-22993-0_37`.

72 Vladimir Kolmogorov, Andrei Krokhin, and Michal Rolinek. The complexity of general-valued CSPs. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 1246–1258, 2015. `doi:10.1109/FOCS.2015.80`.

73 Vladimir Kolmogorov, Johan Thapper, and Stanislav Živný. The power of linear programming for general-valued CSPs. *SIAM Journal on Computing*, 44(1):1–36, 2015. `doi:10.1137/130945648`.

74 Vladimir Kolmogorov and Stanislav Živný. The complexity of conservative valued CSPs. *Journal of the ACM*, 60(2), 2013. Article No. 10. URL: `http://zivny.cz/publications/kz13jacm-preprint.pdf`, `doi:10.1145/2450142.2450146`.

75 Marcin Kozik. Weaker consistency notions for all the CSPs of bounded width. In *LICS'16*, 2016. To appear. Full version is available as arXiv:1605.00565.

76 Marcin Kozik, Andrei Krokhin, Matt Valeriote, and Ross Willard. Characterizations of several Maltsev conditions. *Algebra Universalis*, 73(3):205–224, 2015. `doi:10.1007/s00012-015-0327-2`.

77 Marcin Kozik and Joanna Ochremiak. Algebraic properties of valued constraint satisfaction problem. In *Proceedings of the 42nd International Colloquium on Automata, Languages and Programming (ICALP'15)*, volume 9134 of *Lecture Notes in Computer Science*, pages 846–858. Springer, 2015. `doi:10.1007/978-3-662-47672-7_69`.

78 Marcin Kozik and Joanna Ochremiak. Algebraic properties of valued constraint satisfaction problem. *CoRR*, abs/1403.0476, 2015.

79 Andrei Krokhin and Benoit Larose. Maximizing Supermodular Functions on Product Lattices, with Application to Maximum Constraint Satisfaction. *SIAM Journal on Discrete Mathematics*, 22(1):312–328, 2008. `doi:10.1137/060669565`.

80 Andrei Krokhin and Stanislav Živný, editors. *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*. Schloss Dagstuhl

– Leibniz-Zentrum fuer Informatik, 2017. URL: http://www.dagstuhl.de/dagpub/978-3-95977-003-3.

81 Fredrik Kuivinen. On the complexity of submodular function minimisation on diamonds. *Discrete Optimization*, 8(3):459–477, 2011. doi:10.1016/j.disopt.2011.04.001.

82 Gábor Kun, Ryan O'Donnell, Suguru Tamaki, Yuichi Yoshida, and Yuan Zhou. Linear programming, width-1 CSPs, and robust satisfaction. In *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, pages 484–495, 2012.

83 Richard E. Ladner. On the Structure of Polynomial Time Reducibility. *Journal of the ACM*, 22:155–171, 1975. doi:10.1145/321864.321877.

84 Benoît Larose. Algebra and the Complexity of Digraph CSPs: a Survey. In Andrei Krokhin and Stanislav Živný, editors, *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*, pages 267–285. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2017. doi:10.4230/DFU.Vol7.15301.267.

85 Benoit Larose and Pascal Tesson. Universal Algebra and Hardness Results for Constraint Satisfaction Problems. *Theoretical Computer Science*, 410(18):1629–1647, 2009. doi:10.1016/j.tcs.2008.12.048.

86 Benoit Larose and László Zádori. Bounded width problems and algebras. *Algebra Universalis*, 56(3-4):439–466, 2007. doi:10.1007/s00012-007-2012-6.

87 Steffen L. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

88 Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 1049–1065, 2015. Full version is available at arXiv:1508.04874.

89 Konstantin Makarychev and Yuri Makarychev. Approximation Algorithms for CSPs. In Andrei 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 fuer Informatik, Dagstuhl, Germany, 2017. doi:10.4230/DFU.Vol7.15301.287.

90 Miklós Maróti and Ralph McKenzie. Existence theorems for weakly symmetric operations. *Algebra Universalis*, 59(3-4):463–489, 2008. doi:10.1007/s00012-008-2122-9.

91 Dániel Marx. Tractable hypergraph properties for constraint satisfaction and conjunctive queries. *Journal of the ACM*, 60(6), 2013. Article No. 42. doi:10.1145/2535926.

92 S. Thomas McCormick and Satoru Fujishige. Strongly polynomial and fully combinatorial algorithms for bisubmodular function minimization. *Mathematical Programming*, 122(1):87–120, 2010. doi:10.1007/s10107-008-0242-9.

93 Marc Mezard and Andrea Montanari. *Information, Physics, and Computation*. Oxford University Press, 2009.

94 Michael Pinsker. Algebraic and model theoretic methods in constraint satisfaction. *CoRR*, abs/1507.00931, 2015. URL: http://arxiv.org/abs/1507.00931.

95 Thomas J. Schaefer. The Complexity of Satisfiability Problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing (STOC'78)*, pages 216–226. ACM, 1978. doi:10.1145/800133.804350.

96 Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24 of *Algorithms and Combinatorics*. Springer, 2003.

97 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. Discrete Math.*, 3(3):411–430, 1990. doi:10.1137/0403036.

98 Mark H. Siggers. A strong Mal'cev condition for locally finite varieties omitting the unary type. *Algebra Universalis*, 64(1):15–20, 2010. doi:10.1007/s00012-010-0082-3.

99 A. Szendrei. *Clones in Universal Algebra*, volume 99 of *Seminaires de Mathematiques Superieures*. University of Montreal, 1986.

100 Rustem Takhanov. A Dichotomy Theorem for the General Minimum Cost Homomorphism Problem. In *Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science (STACS'10)*, pages 657–668, 2010. `doi:10.4230/LIPIcs.STACS.2010.2493`.

101 Rustem Takhanov. Extensions of the Minimum Cost Homomorphism Problem. In *Proceedings of the 16th International Computing and Combinatorics Conference (COCOON'10)*, volume 6196 of *Lecture Notes in Computer Science*, pages 328–337. Springer, 2010. `doi:10.1007/978-3-642-14031-0_36`.

102 Walter Taylor. Varieties obeying homotopy laws. *Canadian Journal of Mathematics*, 29(3):498–527, 1977.

103 Johan Thapper and Stanislav Živný. The power of linear programming for valued CSPs. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS'12)*, pages 669–678. IEEE, 2012. URL: `http://zivny.cz/publications/tz12focs-preprint.pdf`, `doi:10.1109/FOCS.2012.25`.

104 Johan Thapper and Stanislav Živný. Necessary Conditions on Tractability of Valued Constraint Languages. *SIAM Journal on Discrete Mathematics*, 29(4):2361–2384, 2015. URL: `http://zivny.cz/publications/tz15sidma-preprint.pdf`, `doi:10.1137/140990346`.

105 Johan Thapper and Stanislav Živný. Sherali-Adams relaxations for valued CSPs. In *Proceedings of the 42nd International Colloquium on Automata, Languages and Programming (ICALP'15)*, volume 9134 of *Lecture Notes in Computer Science*, pages 1058–1069. Springer, 2015. URL: `http://zivny.cz/publications/tz15icalp-preprint.pdf`, `doi:10.1007/978-3-662-47672-7_86`.

106 Johan Thapper and Stanislav Živný. The complexity of finite-valued CSPs. *Journal of the ACM*, 63(4), 2016. Article No. 37. URL: `http://zivny.cz/publications/tz16jacm-preprint.pdf`, `doi:10.1145/2974019`.

107 Johan Thapper and Stanislav Živný. The limits of SDP relaxations for general-valued CSPs, December 2016. arXiv:1612.01147. URL: `http://arxiv.org/abs/1612.01147`.

108 Johan Thapper and Stanislav Živný. The power of Sherali-Adams relaxations for general-valued CSPs. Technical report, arXiv:1606.02577, 2016.

109 Donald Topkis. *Supermodularity and Complementarity*. Princeton University Press, 1998.

110 Hannes Uppman. The Complexity of Three-Element Min-Sol and Conservative Min-Cost-Hom. In *Proceedings of the 40th International Colloquium on Automata, Languages, and Programming (ICALP'13)*, volume 7965 of *Lecture Notes in Computer Science*, pages 804–815. Springer, 2013. `doi:10.1007/978-3-642-39206-1_68`.

111 Hannes Uppman. Computational Complexity of the Extended Minimum Cost Homomorphism Problem on Three-Element Domains. In *Proceedings of the 31st International Symposium on Theoretical Aspects of Computer Science (STACS'14)*, volume 25, pages 651–662, 2014. `doi:10.4230/LIPIcs.STACS.2014.651`.

112 Andrius Vaicenavičius. A study of weighted clones. Master's thesis, Mathematical Institute, University of Oxford, 2014.

113 Jiří Vančura. Weighted Clones. Master's thesis, Department of Algebra, Charles University, 2014.

114 Martin J. Wainwright and Michael I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008. `doi:10.1561/2200000001`.

115 Stanislav Živný. *The Complexity and Expressive Power of Valued Constraints*. PhD thesis, University of Oxford, 2009. URL: `http://ora.ouls.ox.ac.uk/objects/uuid:63facf22-7c2b-4d4a-8b6f-f7c323759ca0`.

116 Stanislav Živný. *The complexity of valued constraint satisfaction problems.* Cognitive Technologies. Springer, 2012. `doi:10.1007/978-3-642-33974-5`.

117 Stanislav Živný, David A. Cohen, and Peter G. Jeavons. The Expressive Power of Binary Submodular Functions. *Discrete Applied Mathematics*, 157(15):3347–3358, 2009. URL: `http://zivny.cz/publications/zcj09dam-preprint.pdf`, `doi:10.1016/j.dam.2009.07.001`.

Algebra and the Complexity of Digraph CSPs: a Survey

Benoît Larose

LACIM, Université du Québec à Montréal, Montréal, Canada
blarose@lacim.ca

Abstract

We present a brief survey of some of the key results on the interplay between algebraic and graph-theoretic methods in the study of the complexity of digraph-based constraint satisfaction problems.

1998 ACM Subject Classification A.1 Introductory and Survey, D.3.2 Constraint and Logic Languages, F.2.2 Computations on Discrete Structures

Keywords and phrases Constraint satisfaction problems, Polymorphisms, Digraphs

Digital Object Identifier 10.4230/DFU.Vol7.15301.267

1 Introduction

At the expense of elegance and tradition, and since in all likelyhood the reader is already acquainted with some or all aspects of the topic at hand, we shall spare her the customary high-level introductory paragraphs, and refer to [10, 24, 75, 51] for more detailed accounts on motivation, background, and history of the field; in particular, we highly recommend the paper [60] for an excellent overview of algebraic methods in the study of digraph CSPs, as well as a wealth of interesting examples. In brief: the constraint satisfaction problem is a natural, flexible framework which encompasses several decision problems which are ubiquitous and fundamental in computer science; the introduction of powerful algebraic techniques in the pioneering work of [46], [61], [62] and [20] has led to a much deeper understanding of the algorithmic complexity of fixed-template CSPs; precise conjectures have been formulated linking the algorithmic and descriptive complexity of these CSPs to the algebraic properties of the fixed template. Very roughly, the paradigm underlying this theory is the following: if the template supports structure-preserving operations (*polymorphisms*) that obey "nice enough" identities, then the associated decision problem should be well-behaved from an algorithmic point of view; and if no such operations are present, then the problem is hard for some well-known complexity class. The present paper, concerned with the interplay of algebraic and graph-theoretic techniques in the study of these conjectures, focuses on CSPs whose fixed template is a digraph, possibly with some extra unary constraints. These are known in the literature under various names, such as *graph* or *digraph homomorphism problems*, *digraph list homomorphism problems*, *digraph with constants problem*, *digraph retraction problem*, *one-or-all list homomorphism problems*, and so on.

Why digraphs? Obviously these structures offer a good source of examples to test conjectures because they are simple, natural and we have powerful representation techniques for digraphs (in other words, we can draw them). One can sometimes hope to explicitly describe combinatorial properties that turn out to characterise the complexity. On the other hand, digraphs are flexible enough to encode complex problems. Secondly, various natural conditions can be imposed on digraphs to obtain subfamilies such as simple graphs, graphs with loops, posets, tournaments, acyclic digraphs, etc. as to make testing some difficult conjectures more amenable; in this respect, the considerable literature on graph theory is a powerful tool. Thirdly, and perhaps more interestingly, the proofs of some strong general theorems on CSPs rely on results specifically on digraphs, see for instance [12]. It should also be noted that the complexity of CSPs on digraphs has been studied well before algebraic tools were introduced; in fact the concept of graph homomorphism can be traced back to the mid 20th century, and its complexity-theoretic aspects at least as far back as the late 70's.

To streamline the presentation, and in order to make this survey accessible to both mathematician and computer scientist alike, we shall avoid more involved universal algebraic concepts and results concerning varieties, tame congruence theory and such, and try to rephrase the relevant results only in terms of polymorphisms of structures whenever possible; for a more detailed account we refer the gentle reader to [10, 24]. The small price to pay for this approach is that some of the terminology we use is slightly non-standard (namely Definitions 6 and 7). Similarly, we avoid most technicalities concerning complexity issues and refer the reader to [2, 86] for these.

We make no claim at comprehensiveness, and obviously certain editorial choices have been made as to the inclusion or not of certain results. The literature on the complexity of digraph homomorphism is quite vast, so we shall focus mainly on those results involving algebraic techniques. Fixed-template constraint satisfaction problem involving digraphs that appear in the literature can be roughly classified into one of four different categories: (i) the "straightforward" $CSP(\mathbb{H})$ where $\mathbb{H}$ is a digraph; (ii) the so-called CSP with constants, or retraction problem, or one-or-all list homomorphism problem $CSP(\mathbb{H}^{+c})$ where $\mathbb{H}^{+c}$ is the structure consisting of the digraph $\mathbb{H}$ together with all unary singleton relations $\{h\}$ with $h \in H$; (iii) the list homomorphism or conservative constraint satisfaction problem $CSP(\mathbb{H}^{+l})$ where $\mathbb{H}^{+l}$ is the structure consisting of the digraph $\mathbb{H}$ together with all non-empty unary relations S with $S \subseteq H$; (iv) constraint satisfaction problems of one of the above forms but with various restrictions on the inputs (bounded degree, connected lists, bipartite inputs, etc.) Since this last case has not been much investigated with the use of algebraic tools we focus mainly on cases (i), (ii), (iii): in the sequel we shall refer to CSPs of one of these forms as *digraph CSPs*.

Some of the questions we wish to address in this paper are the following: are the dichotomy conjectures proved for a particular class of digraphs? Is there a combinatorial characterisation of the digraphs admitting such and such polymorphisms? Is there a combinatorial description of the digraphs whose CSP is solvable with such and such restriction on a given resource? Is there some collapse (of Mal'tsev conditions or complexity) for such and such class of digraphs?

Here is a brief outline of the paper: in section 2 we first introduce the basic notation and concepts in algebra, (descriptive) complexity and graphs that we require in the sequel. We shall then state in section 3 the important general results on CSPs we need later on, as well as the conjectures that will orient our presentation. In section 4 we present results on CSPs on digraphs and consider various subfamilies of digraphs; sections 5 and 6 deal with the variants of the CSP alluded to earlier, namely digraphs with constants and the list homomorphism problem. Section 7 closes the paper with a series of open problems.

2 Preliminaries

Where we set our notation and define our terms.

2.1 Relational Structures and Digraphs

A (finite) relational structure is a tuple $\mathbb{H} = \langle H; \theta_1, \cdots, \theta_r \rangle$ where H is a non-empty, finite set, and for each $1 \leq i \leq r$, θ_i is a relation of arity ρ_i on H, i.e. $\theta_i \subseteq H^{\rho_i}$; the *signature* of $\mathbb{H}$ is the sequence $(\rho_1, \dots, \rho_r)$. In the sequel, all structures will be assumed finite, and equipped with finitely many basic relations. Let $\mathbb{H}_i = \langle H_i; \mu_1^i, \cdots, \mu_r^i \rangle$, $i = 1, 2$ be two structures of the same signature $\rho_1, \dots, \rho_r$. The *product* of $\mathbb{H}_1$ and $\mathbb{H}_2$, is the relational structure $\mathbb{H}_1 \times \mathbb{H}_2 = \langle H_1 \times H_2; \mu_1, \cdots, \mu_r \rangle$ of the same signature as the $\mathbb{H}_i$ where for every $j = 1, \dots, r$, $((x_1, y_1), \dots, (x_{\rho_j}, y_{\rho_j})) \in \mu_j$ if $(x_1, \dots, x_{\rho_j}) \in \mu_j^1$ and $(y_1, \dots, y_{\rho_j}) \in \mu_j^2$. This extends naturally to any number of factors, and we use the notation $\mathbb{H}^k$ to denote the product of k copies of the structure $\mathbb{H}$.

Let $\mathbb{G} = \langle G; \mu_1, \cdots, \mu_r \rangle$ be a structure with the same signature as $\mathbb{H}$. A function $f : G \to H$ is a *homomorphism* from $\mathbb{G}$ to $\mathbb{H}$, and we write $f : \mathbb{G} \to \mathbb{H}$, if for every $1 \leq i \leq r$, $(f(x_1), \dots, f(x_{\rho_i})) \in \theta_i$ whenever $(x_1, \dots, x_{\rho_i}) \in \mu_i$. If there exist homomorphisms from $\mathbb{G}$ to $\mathbb{H}$ and from $\mathbb{H}$ to $\mathbb{G}$, we say that $\mathbb{G}$ and $\mathbb{H}$ are *homomorphically equivalent.* A structure $\mathbb{H}$ is a *core* if every homomorphism $f : \mathbb{H} \to \mathbb{H}$ is a bijection. It is well-known and easy to verify that every finite relational structure is homomorphically equivalent to a core which is unique up to isomorphism; hence we may speak of *the* core of a structure.

▶ **Definition 1.** Let $\mathbb{H}$ be a relational structure. The set of structures that admit a homomorphism to $\mathbb{H}$ is denoted by $CSP(\mathbb{H})$.

Viewed as a decision problem, $CSP(\mathbb{H})$ consists in determining on input $\mathbb{G}$ whether there exists a homomorphism from $\mathbb{G}$ to $\mathbb{H}$. Obviously if $\mathbb{H}'$ and $\mathbb{H}$ are homomorphically equivalent then $CSP(\mathbb{H}') = CSP(\mathbb{H})$; in particular, for every structure $\mathbb{H}$, we have that $CSP(\mathbb{H}) = CSP(\mathbb{H}')$ where $\mathbb{H}'$ is the core of $\mathbb{H}$.

▶ **Definition 2.** Let $\mathbb{H} = \langle H; \theta_1, \dots, \theta_r \rangle$ be a structure.

- Let $\mathbb{H}^{+c} = \langle H; \theta_1, \dots, \theta_r, \{h\}\, (h \in H) \rangle$ be the structure obtained from $\mathbb{H}$ by adding every singleton unary relation $\{h\}\, (h \in H)$ to $\mathbb{H}$ as a basic relation.
- Let $\mathbb{H}^{+l} = \langle H; \theta_1, \dots, \theta_r, S\, (\emptyset \subset S \subseteq H) \rangle$ be the structure obtained from $\mathbb{H}$ by adding every non-empty subset $S \subseteq H$ to $\mathbb{H}$ as a basic relation.

Viewed as a decision problem, $CSP(\mathbb{H}^{+c})$ takes as input a structure $\mathbb{G}$ of the same signature as $\mathbb{H}$ where certain elements of $\mathbb{G}$ have been "pre-coloured" by some value in H; one must decide if there exists a homomorphism from $\mathbb{G}$ to $\mathbb{H}$ that respects the pre-colouring. This problem is known in the literature as the *homomorphism extension problem* [78], or *one-or-all list homomorphism problem* [40], and can easily seen to be equivalent to the so-called *retraction problem* [40]. On the other hand, $CSP(\mathbb{H}^{+l})$ takes as input a structure $\mathbb{G}$ of the same signature as $\mathbb{H}$ where certain elements of $\mathbb{G}$ are assigned a list of prescribed values in H; one must decide if a homomorphism f exists from $\mathbb{G}$ to $\mathbb{H}$ such that $f(x)$ belongs to the list assigned to x. This is known as the *list homomorphism problem*, and such problems are also known as *conservative CSPs.* Notice that the structures $\mathbb{H}^{+c}$ and $\mathbb{H}^{+l}$ are cores.

2.2 Digraphs

A *digraph* is a relational structure $\mathbb{H} = \langle H, \theta \rangle$ with a single binary relation θ; the members of H are the *vertices* of $\mathbb{H}$ and the elements of θ are called *arcs.* If (h, h') is an arc we say that h is an *in-neighbour* of h' and h' is an *out-neighbour* of h; we also say that h and h' are neighbours. The digraph $\mathbb{G} = \langle G; \rho \rangle$ is an *induced subdigraph* of $\mathbb{H}$ if $G \subseteq H$ and $\rho = \theta \cap G^2$. A digraph $\mathbb{H}$ is *connected (strongly connected)* if for every distinct $h, h' \in H$ there exists a sequence $h = x_0, \dots, x_n = h'$ of vertices of $\mathbb{H}$ such that x_i and x_{i+1} are neighbours ((x_i, x_{i+1}) is an arc, respectively) for all $0 \leq i \leq n-1$. A *connected component (strong connected component)* of $\mathbb{H}$ is a connected (strongly connected respectively) induced subdigraph of $\mathbb{H}$ maximal with respect to inclusion. A digraph $\mathbb{H} = \langle H; \theta \rangle$ is *bipartite* if $H = A \cup B$ with A and B disjoint such that $\theta \subseteq A \times B \cup B \times A$.

An arc of the form (h, h) is a *loop*; the digraph $\mathbb{H}$ is *reflexive* if θ contains all loops, and is *symmetric* if $(h, h') \in \theta$ implies $(h', h) \in \theta$; symmetric digraphs are often called *graphs*, and a *simple graph* is a graph without loops. The *underlying graph* of a digraph $\mathbb{H}$ is the graph obtained from $\mathbb{H}$ by replacing every arc by a symmetric edge. A digraph is *antisymmetric* if $(h, h'), (h', h) \in \theta$ implies $h = h'$, and it is *transitive* if $(h, h'') \in \theta$ whenever $(h, h'), (h', h'') \in \theta$. A *poset* is a reflexive, antisymmetric, transitive digraph.

An *oriented path* is a digraph with vertex set $\{x_0, \dots, x_n\}$ ($n \geq 1$) such that, for every $i = 0, \dots, n-1$, precisely one of (x_i, x_{i+1}) or (x_{i+1}, x_i) is an arc, and there are no other arcs; an *oriented cycle* is a digraph with vertex set $\{x_0, \dots, x_n\}$ ($n \geq 1$) such that, for every $i = 0, \dots, n-1$, precisely one of (x_i, x_{i+1}) or (x_{i+1}, x_i) is an arc, precisely one of (x_0, x_n) or (x_n, x_0) is an arc and there are no other arcs. The *net length* (or *algebraic length)* of an oriented cycle is the number of forward arcs minus the number of backward arcs according to some fixed traversal of the cycle. An oriented cycle is *balanced* if it has net length 0, and *unbalanced* otherwise. An n-vertex oriented cycle of net length n we call a *directed cycle (or circle).* An *oriented tree* is an antisymmetric digraph whose underlying graph is a tree, i.e. an acyclic connected graph.

2.3 Polymorphisms

Let $\mathbb{H}$ be a relational structure. A *polymorphism* of $\mathbb{H}$ of *arity* k is a homomorphism from $\mathbb{H}^k$ to $\mathbb{H}$. If f is a polymorphism of $\mathbb{H}$ we also say that $\mathbb{H}$ *admits* f, or that $\mathbb{H}$ is *invariant under* f. A polymorphism is *idempotent* if it satisfies $f(x, x, \dots, x) = x$ for all $x \in H$, and is *conservative* if $f(x_1, \dots, x_n) \in \{x_1, \dots, x_n\}$ for all $x_i \in H$.

We use the convenient notation $f(x_1, \dots, x_k) \approx g(y_1, \dots, y_n)$ to indicate equality where all variables are universally quantified, and call such an expression a *linear identity.*

A *semilattice* operation is an associative, idempotent, commutative binary operation. Let $k \geq 2$; a k-ary operation f is *cyclic* if it obeys

$$f(x_1, \dots, x_k) \approx f(x_k, x_1, \dots, x_{k-1});$$

it is *symmetric* if, for every permutation σ of the set $\{1, \dots, k\}$, it obeys the identity

$$f(x_1, \dots, x_k) \approx f(x_{\sigma(1)}, \dots, x_{\sigma(k)});$$

and finally call f *totally symmetric (TS)* if

$$f(x_1, \dots, x_k) \approx f(y_1, \dots, y_k)$$

whenever $\{x_1, \dots, x_k\} = \{y_1, \dots, y_k\}$.

For $k \geq 3$, the operation f is a *near-unanimity (NU) operation* if it obeys the identity

$$f(x, \dots, x, y, x, \dots, x) \approx x$$

for any position of the lone y. A 3-ary NU operation is called a *majority* operation. For $k \geq 2$, the idempotent operation f is a *weak near-unanimity (WNU) operation* if it obeys the identities

$$f(x, \dots, x, y, x, \dots, x) \approx f(x, \dots, x, y, x, \dots, x)$$

for any positions of the lone y's.

A 3-ary operation f is *Mal'tsev* if it obeys the identities

$$f(y,y,x) \approx f(x,y,y) \approx x.$$

A 4-ary, idempotent operation f is *Siggers* if it satisfies the identity

$$f(a,r,e,a) \approx f(r,a,r,e).$$

We now gather some well-known implications involving the special polymorphisms defined here; as some of these results are folklore, we give a general reference only [24] (see also [66].)

▶ **Proposition 3.** *If a structure admits a (conservative) semilattice polymorphism then it admits (conservative) idempotent k-ary TS polymorphisms for all $k \geq 2$. A structure admits a Siggers polymorphism if and only if it admits a WNU polymorphism. If a structure admits an idempotent polymorphism f which is cyclic, symmetric, TS, NU or Mal'tsev then it admits a WNU polymorphism; moreover, in each case, if f is conservative, so is the WNU polymorphism.*

2.4 Datalog

Many naturally occurring tractable CSPs fall within one of two families of CSPs, namely problems of *bounded width* and those with *few subpowers*. The first family consists of problems solvable by local consistency methods; the CSPs in the second family are those that are solvable by an algorithm that shares many characteristics with Gaussian elimination; both classes of problems are characterised by identities [13], [18, 58]. As far as we can tell very little is known about digraph problems with few subpowers which do not have bounded width.

It is convenient for us to describe problems of bounded width with the use of the logic programming language Datalog; for more details see for instance [75]. A *Datalog program* is a finite set of rules of the form

$$T_0 :- T_1, \ldots, T_n$$

where each T_i is an atomic formula $R(x_{i_1}, \ldots, x_{i_k})$ from some fixed signature. Then T_0 is called the *head* of the rule, and the sequence $T_1, \ldots, T_n$ the *body* of the rule. There are two kinds of relational predicates occurring in the program: predicates R that occur at least once in the head of a rule are called *intensional database predicates* (IDBs) and are not part of τ. The other predicates which occur only in the body of a rule are called *extensional database predicates* and must all lie in τ. One special IDB, which is 0-ary, is the *goal predicate* of the program. Each Datalog program is a recursive specification of the IDBs, with semantics obtained via least fixed-points of monotone operators. The goal predicate is initially set to `false`, and the Datalog program *accepts* a structure $\mathbb{G}$ if its goal predicate evaluates to `true` on $\mathbb{G}$.

A Datalog program is *linear* if each of its rules has at most one occurrence of an IDB in its body. Given a rule t of the form

$$I :- J, T_1, \ldots, T_n$$

of a linear Datalog program where I and J are IDB's, its *symmetric* complement t_s is the rule

$$J :- I, T_1, \ldots, T_n;$$

if t has no IDB in the body then we let $t_s = t$. A linear program is said to be *symmetric* if it contains the symmetric complement of each of its rules. Finally, a Datalog program is *non-recursive* if the body of every rule contains only EDB's.

▶ **Definition 4.** A class $\mathcal{C}$ of structures is said to be *definable in (linear, symmetric, non-recursive) Datalog* if there is a (linear, symmetric, non-recursive) Datalog program which accepts precisely the structures from $\mathcal{C}$.

By their nature, Datalog programs define homomorphism closed classes of structures, hence in the context of CSPs a Datalog program accepts precisely the structures that *do not* map to the target structure; for instance it is a nice exercise to write up a symmetric Datalog program that recognises precisely non-bipartite graphs. To simplify the presentation we shall just say that $CSP(\mathbb{H})$ is definable in (linear, symmetric, non-recursive) Datalog rather than introduce extra notation. CSPs definable in Datalog are said to be of *bounded width*; CSPs definable in non-recursive Datalog are precisely those that are first-order definable; this was first proved in [3]; a slightly more refined result is Theorem 2 of [23]. $CSP(\mathbb{H})$ has *width 1* if it is recognised by a Datalog program whose IDBs are at most unary; a structure $\mathbb{H}$ has this property precisely if it admits a *set polymorphism*, or equivalently, if it admits TS polymorphisms of all arities [38, 33].

If $CSP(\mathbb{H})$ is definable in Datalog, then it is tractable; if it is definable in linear (symmetric) Datalog then it is solvable in non-deterministic (deterministic) logspace (see [75]). Combining results from [9], [83] and [66], the following characterises CSPs of bounded width.

▶ **Theorem 5.** *Let $\mathbb{H}$ be a core structure. Then the following are equivalent:*

1. *$CSP(\mathbb{H})$ has bounded width;*
2. *there is some N such that $\mathbb{H}$ admits k-ary WNU polymorphisms for all $k \geq N$;*
3. *$\mathbb{H}$ admits idempotent polymorphisms v and w satisfying*

$$\begin{aligned} v(x,x,y) &\approx v(x,y,x) \approx v(y,x,x) \\ w(x,x,x,y) &\approx w(x,x,y,x) \approx w(x,y,x,x) \approx w(y,x,x,x) \\ v(y,x,x) &\approx w(y,x,x,x). \end{aligned}$$

3 General Results

3.1 Three Conjectures and Some Results

It follows from deep results in universal algebra [57] that the existence of certain well-behaved polymorphisms on a structure $\mathbb{H}$ is equivalent to the impossibility of obtaining from $\mathbb{H}$ certain "minimal" structures via so-called *pp-interpretations*. It turns out that the CSPs associated to these minimal structures are logspace reducible to the original CSP [20, 75]; and hence the non-existence of the polymorphisms gives rise to natural hardness results, which are presented in Theorem 9 below. Conversely, it is believed (at least by some ...) that the presence of these polymorphisms should give complexity upper bounds (Conjecture 8). For completeness' sake we now define the polymorphisms in question.

▶ **Definition 6.** Let H be a structure, and $n \geq 2$. We say that $\mathbb{H}$ is *n-permutable* if there exist 3-ary polymorphisms $\{f_1, \ldots, f_{n-1}\}$ of $\mathbb{H}$ that satisfy for $i \leq n-2$ the identities

$$\begin{aligned} x &\approx f_1(x,y,y) \\ f_i(x,x,y) &\approx f_{i+1}(x,y,y) \\ f_{n-1}(x,x,y) &\approx y. \end{aligned}$$

In particular a structure is 2-permutable precisely when it admits a Mal'tsev polymorphism.

Let t be a k-ary operation on the set H and let A be a $k \times k$ matrix with entries in H. We write $t[A]$ to denote the $k \times 1$ matrix whose entry on the i-th row is the value of f applied to row i of A.

▶ **Definition 7** ([66, 47]). Let H be a structure. We say that $\mathbb{H}$ is *join semidistributive* if there exists a k-ary idempotent polymorphism of $\mathbb{H}$ satisfying $t[A] = t[B]$ where A and B are $k \times k$ matrices with entries in $\{x, y\}$ such that $a_{ii} = x$ for all i, $a_{ij} = b_{ij} = x$ for all $i > j$ and $b_{ii} = y$ for all i.

▶ **Conjecture 8.** *Let $\mathbb{H}$ be a core structure.*

1. [20] *If $\mathbb{H}$ admits a WNU polymorphism then $CSP(\mathbb{H})$ is tractable.*
2. [75] *If $\mathbb{H}$ is join-semidistributive then $CSP(\mathbb{H})$ is definable in linear Datalog (and hence is solvable in non-deterministic logspace).*
3. [75] *If $CSP(\mathbb{H})$ has bounded width and $\mathbb{H}$ is n-permutable for some $n \geq 2$ then $CSP(\mathbb{H})$ is definable in symmetric Datalog (and hence is solvable in logspace).*

The first of these conjectures is known as the *algebraic dichotomy conjecture*; the "converse" of all three conjectures holds:

▶ **Theorem 9.** *Let $\mathbb{H}$ be a core structure.*

1. [20] *If $\mathbb{H}$ admits no WNU polymorphism then $CSP(\mathbb{H})$ is NP-complete.*
2. [75] *If $\mathbb{H}$ is not join-semidistributive then $CSP(\mathbb{H})$ is not expressible in linear Datalog and is P-hard.*
3. [75] *If $\mathbb{H}$ is not n-permutable for any n then $CSP(\mathbb{H})$ is not expressible in symmetric Datalog and is NL-hard.*

We gather in the next theorem some special cases of the conjectures that are known to hold.

▶ **Theorem 10.** *Let $\mathbb{H}$ be a core structure.*

1. [16] *If $\mathbb{H}$ admits an NU polymorphism then $CSP(\mathbb{H})$ is definable in linear Datalog;*
2. [32] *If $CSP(\mathbb{H})$ has bounded width and $\mathbb{H}$ is 2-permutable then $CSP(\mathbb{H})$ is definable in symmetric Datalog;*
3. [65] *If $CSP(\mathbb{H})$ is definable in linear Datalog and $\mathbb{H}$ is n-permutable for some $n \geq 2$ then $CSP(\mathbb{H})$ is definable in symmetric Datalog.*

Since join semidistributive structures automatically satisfy the equivalent conditions of Theorem 5 (see [66]), statement (3) in the previous result reduces the symmetric Datalog conjecture to the linear Datalog conjecture.
First-order definable CSPs are in a sense the "easiest" of all non-trivial CSPs. It is known that CSPs that are not first-order definable are logspace-hard [75], and hence there are no fixed template CSPs with complexity strictly between AC^0 and L. Furthermore, first-order definable CSPs cannot be characterised in a purely algebraic way in the sense of the above conjectures: indeed, adding the equality relation to a structure's basic relations does not change its polymorphisms but will make the CSP trivially logspace-hard. On the other hand, there exists a fairly simple combinatorial description of first-order definable CSPs via a dismantling algorithm [71], which in many special cases allows an explicit description of the underlying structures, see section 6 for some examples. It is known that the core of a structure $\mathbb{H}$ with first-order definable CSP admits an NU polymorphism [71], and by Theorem 9 it must also be k-permutable for some $k \geq 2$; furthermore, it follows from [85] that the CSP has tree duality, and hence the core of $\mathbb{H}$ must also admit idempotent TS polymorphisms of all arities.

3.2 Reductions to Digraph Problems

Digraph CSPs are, in full generality, as difficult as CSPs on more general templates. In fact, this remains true even for restricted families of digraphs. We say that two problems $\mathcal{A}$ and $\mathcal{B}$ are *poly-time (logspace, first-order) equivalent* if there exists polynomial time (logspace, first-order) reductions both from $\mathcal{A}$ to $\mathcal{B}$ and from $\mathcal{B}$ to $\mathcal{A}$.

▶ **Theorem 11.** *Let $\mathbb{H}$ be a relational structure.*
1. [46] *There exists a digraph $D(\mathbb{H})$ such that $CSP(D(\mathbb{H}))$ and $CSP(\mathbb{H})$ are poly-time equivalent.*
2. [46] *There exists a bipartite graph $B(\mathbb{H})$ such that $CSP(B(\mathbb{H})^{+c})$ and $CSP(\mathbb{H})$ are poly-time equivalent.*
3. [46] *There exists a poset $P(\mathbb{H})$ such that $CSP(P(\mathbb{H})^{+c})$ and $CSP(\mathbb{H})$ are poly-time equivalent.*
4. [40] *There exists a reflexive graph $R(\mathbb{H})$ such that $CSP(R(\mathbb{H})^{+c})$ and $CSP(\mathbb{H})$ are poly-time equivalent.*

Feder and Vardi [46] actually refine results (1)-(3) by imposing various stringent conditions on the digraphs, bipartite graphs and posets. Unfortunately, the reductions given do not seem to behave well with respect to polymorphisms. The next result handles this situation, and guarantees that all interesting polymorphism identities will be preserved, with the notable exception of Malt'sev polymorphisms; indeed, Kazda [63] has shown that every digraph that admits a Mal'tsev polymorphism also admits a majority polymorphism, a property which does not hold for structures in general.

Let $\mathbb{Z} = \langle Z; \theta \rangle$ be the digraph with $Z = \{0, 1, 2, 3\}$ and $\theta = \{(0, 1), (2, 1), (2, 3)\}$. A linear identity $f(x_1, \dots, x_k) \approx g(y_1, \dots, y_n)$ is *balanced* if the variables appearing on each side are the same, i.e. $\{x_1, \dots, x_k\} = \{y_1, \dots, y_n\}$. Call a set Γ of linear identities *idempotent* if for each operation symbol f appearing in some identity of Γ the identity $f(x, \dots, x) \approx x$ is in Γ. We say that a structure $\mathbb{H}$ *obeys* or *satisfies* Γ if for each operation symbol f appearing in Γ it admits a polymorphism $f_{\mathbb{H}}$ such that the set $\{f_{\mathbb{H}}\}$ satisfies the identities in Γ.

▶ **Theorem 12** ([26])**.** *Let $\mathbb{H}$ be a relational structure. There exists a digraph $\mathcal{D}(\mathbb{H})$ such that the following hold:*
1. *The problems $CSP(\mathcal{D}(\mathbb{H}))$ and $CSP(\mathbb{H})$ are logspace equivalent;*
2. *$\mathbb{H}$ is a core if and only if $\mathcal{D}(\mathbb{H})$ is a core;*
3. *If Γ is an idempotent set of linear identities such that*
   (a) *$\mathbb{Z}$ satisfies Γ,*
   (b) *every identity in Γ is either balanced or contains at most two variables,*
   then $\mathbb{H}$ satisfies Γ if and only if $\mathcal{D}(\mathbb{H})$ satisfies Γ.

We remark that condition (a) is not very restrictive since $\mathbb{Z}$ satisfies all interesting identities in the present context, with the exception of 2-permutability (but it is 3-permutable) [26].

4 $CSP(\mathbb{H})$

Obviously if the digraph $\mathbb{H}$ has a loop the problem $CSP(\mathbb{H})$ is trivial as any digraph $\mathbb{G}$ then admits a constant homomorphism to $\mathbb{H}$; consequently in this section all digraphs are assumed to have no loops. We begin with a classic result of Hell and Nešetřil's, reformulated in its stronger form that shows the algebraic dichotomy conjecture holds.

▶ **Theorem 13** ([50, 21])**.** *Let $\mathbb{H}$ be a symmetric digraph. If $\mathbb{H}$ is bipartite then $CSP(\mathbb{H})$ is tractable; otherwise $\mathbb{H}$ admits no WNU polymorphism and hence $CSP(\mathbb{H})$ is NP-complete.*

If $\mathbb{H}$ is a non-trivial bipartite graph then its core is an edge, and hence the problem $CSP(\mathbb{H})$ is in fact logspace-complete [1]. Other algebraic proofs of Hell and Nešetřil's result can be found in [12] and [93].

Arguably the simplest digraphs are oriented paths and cycles; the classification of the complexity of their associated CSPs was completed by Feder [39]. The case of balanced cycles is settled by a reduction to so-called bipartite boolean constraint-satisfaction problems that are shown to be either tractable or NP-complete, but the polymorphism behaviour is not quite transparent in the proof.

▶ **Theorem 14** ([56, 48, 39]). *Let $\mathbb{H}$ be a digraph.*
1. *If $\mathbb{H}$ is an oriented path, then it admits (conservative) majority and semilattice polymorphisms;*
2. *If $\mathbb{H}$ is an unbalanced oriented cycle, then it admits a (conservative) majority polymorphism;*
3. *If $\mathbb{H}$ is a balanced oriented cycle, then $CSP(\mathbb{H})$ is either tractable or NP-complete.*

There are known oriented trees with NP-complete CSP [48], [49]; the smallest known example is a 33-vertex triad [14]: a *polyad* is an oriented tree whose underlying graph has a unique vertex of degree greater than 2; a *triad* is a polyad with a unique vertex of degree 3. The algebraic dichotomy conjecture has been verified for a restricted family of triads called *special triads* [14], generalised to *special polyads* [11], and then to *special oriented trees* [25]. It turns out that the tractable CSPs on special oriented trees all have bounded width, and Bulín conjectures this holds for all oriented trees [25].

A *semi-complete* digraph is a digraph (without loops) such that there is at least one arc between every two vertices; this family includes complete graphs and tournaments as special cases. A dichotomy was first proved for semi-complete digraphs in [5]; the polymorphism behaviour of these digraphs is completely described in [60], Theorem 8.1. A digraph is *locally semi-complete* if if for every vertex v of $\mathbb{H}$, both the sets of in- (out-) neighbours of v induce semicomplete digraphs. A dichotomy for CSPs on connected locally semi-complete digraphs is proved in [6], Theorem 6.1; it turns out that the dividing line between tractability and NP-completeness is exactly the same for the list homomorphism problem on these digraphs ([6], Theorem 6.2).

A digraph is *smooth* if it has no sinks or sources, i.e. if every vertex has both in- and out-degree at least 1. The next result proves the algebraic dichotomy conjecture for CSPs on smooth digraphs, as well as confirming a conjecture of Bang-Jensen and Hell [4]:

▶ **Theorem 15** ([15]). *Let $\mathbb{H}$ be a smooth digraph. If each connected component of the core of $\mathbb{H}$ is a circle, then $CSP(\mathbb{H})$ is tractable, otherwise $\mathbb{H}$ admits no WNU polymorphism (and hence $CSP(\mathbb{H})$ is NP-complete).*

5 $CSP(\mathbb{H}^{+c})$

If we add to $\mathbb{H}$ all unary singleton relations as possible constraints, we obtain the problem $CSP(\mathbb{H}^{+c})$ which is in general harder than $CSP(\mathbb{H})$, unless $\mathbb{H}$ is itself a core, in which case the problems are logspace equivalent [62]. Notice that the polymorphisms of $\mathbb{H}^{+c}$ are precisely the *idempotent* polymorphisms of $\mathbb{H}$. For instance, if $\mathbb{H}$ is the symmetric 6-cycle, then its core is the symmetric edge and hence $CSP(\mathbb{H})$ is logspace-complete; on the other hand, $\mathbb{H}$ admits no idempotent polymorphisms other than projections (exercise), and hence by Theorem 9 above the problem $CSP(\mathbb{H}^{+c})$ is NP-complete. In the other direction, any complexity upper bound on the list homomorphism problem for $\mathbb{H}$ also applies to $CSP(\mathbb{H}^{+c})$;

notice also that the polymorphisms of $\mathbb{H}^{+l}$ are precisely the *conservative* polymorphisms of $\mathbb{H}$.

One of the interesting aspects of the decision problem $CSP(\mathbb{H}^{+c})$ from a combinatorial point of view is that, since $\mathbb{H}^{+c}$ is a core for any digraph $\mathbb{H}$, we may consider digraphs with possible loops. We start by examining a few results on mixed digraphs, then consider results on mixed undirected graphs, and then finally move on to reflexive digraphs. We shall use the following terminology: A *reflexive oriented path (cycle, tree)* is an oriented path (cycle, tree) where all loops have been added.

5.1 $CSP(\mathbb{H}^{+c})$ for Mixed Digraphs

An antisymmetric semi-complete digraph $\mathbb{H}$ is called a *tournament*, i.e. for every pair of distinct vertices u and v exactly one of $(u, v), (v, u)$ is an arc of $\mathbb{H}$. A *tournament of mixed type* is obtained from a tournament by adding some (perhaps not all) loops; more generally, we use the same terminology and talk about *mixed (di)graphs*, etc.

▶ **Theorem 16** ([92]). *Let $\mathbb{H}$ be a tournament of mixed type. Then either $CSP(\mathbb{H}^{+c})$ has bounded width or $\mathbb{H}$ admits no idempotent WNU polymorphism, and hence $CSP(\mathbb{H}^{+c})$ is NP-complete.*

A *strongly bipartite digraph* is a digraph $\mathbb{H} = \langle H; \theta \rangle$ where H is the disjoint union of two non-empty sets A and B and $\theta \subseteq A \times B$; equivalently, a digraph is strongly bipartite if every vertex is a source or a sink. The retraction problems for these digraphs exhibit a sharp collapse: indeed, by Theorem 10 the presence of an NU polymorphism guarantees that $CSP(\mathbb{H}^{+c})$ is definable in linear Datalog; in the present case we actually get all the way down to non-recursive Datalog:

▶ **Theorem 17** ([45]). *Let $\mathbb{H}$ be a connected strongly bipartite digraph. Then the following are equivalent:*
1. *$\mathbb{H}$ admits an NU polymorphism;*
2. *$CSP(\mathbb{H}^{+c})$ is first-order definable.*

5.2 $CSP(\mathbb{H}^{+c})$ for Mixed Undirected Graphs

A *(mixed) pseudotree* is a connected undirected graph that contains at most one cycle (other than loops, which are permitted.) The complexity of $CSP(\mathbb{H}^{+c})$ for pseudotrees is characterised in the next result, although the polymorphism behaviour of the tractable case is not transparent in the proof.

▶ **Theorem 18** ([43]). *Let $\mathbb{H}$ be a mixed pseudotree. If the loops of $\mathbb{H}$ induce a disconnected graph, or $\mathbb{H}$ contains an induced cycle of length at least 5, or a reflexive 4-cycle or an irreflexive 3-cycle, then $CSP(\mathbb{H}^{+c})$ is NP-complete. Otherwise $CSP(\mathbb{H}^{+c})$ is tractable.*

The special case of mixed undirected cycles is worth delineating. We note that the result invokes [90] which classifies the complexity of $CSP(\mathbb{H}^{+c})$ for all mixed undirected graphs with at most 4 vertices.

▶ **Theorem 19** ([43]). *Let $\mathbb{H}$ be a mixed undirected cycle on $n \geq 3$ vertices. If $n = 3$ and $\mathbb{H}$ has at least one loop, or if $n = 4$, $\mathbb{H}$ has at least one non-loop and the loops of $\mathbb{H}$ induce a connected graph, then $CSP(\mathbb{H}^{+c})$ is tractable. Otherwise, $CSP(\mathbb{H}^{+c})$ is NP-complete.*

Analogous to Theorem 17 above, retraction problems on bipartite graphs also exhibit some collapse, although not quite as sharp as in the strongly bipartite case. Notice that a non-trivial bipartite graph without loops cannot admit an idempotent binary TS polymorphism and hence its retraction problem cannot be first-order definable.

▶ **Theorem 20** ([70]). *Let $\mathbb{H}$ be a connected, irreflexive bipartite graph. If $\mathbb{H}$ admits an NU polymorphism then $CSP(\mathbb{H}^{+c})$ is definable in symmetric Datalog, and hence solvable in logspace.*

Combined with Theorem 9, it follows that a bipartite graph with an NU polymorphism must be k-permutable for some $k \geq 2$. R. Willard has verified that the converse holds if $k \leq 5$, however there exists a 6-permutable bipartite graph that admits no NU polymorphism [91].

5.3 $CSP(\mathbb{H}^{+c})$ for Reflexive Digraphs

A digraph is *intransitive* if, whenever $(u, v), (v, w), (u, w)$ are arcs then $|\{u, v, w\}| \leq 2$. Notice that any digraph of girth at least 4 (i.e. whose underlying graph contains no induced cycle of size 3 or less) is intransitive, in particular oriented trees as well as oriented cycles on 4 or more vertices are intransitive. [1]

▶ **Theorem 21** ([69]). *Let $\mathbb{H}$ be an intransitive reflexive digraph. Then the following are equivalent:*
1. *$\mathbb{H}$ admits a WNU polymorphism;*
2. *$\mathbb{H}$ admits a majority polymorphism;*
3. *$\mathbb{H}$ is a disjoint union of oriented trees;*

if any of these conditions hold, $CSP(\mathbb{H}^{+c})$ is definable in linear Datalog; otherwise $CSP(\mathbb{H}^{+c})$ is NP-complete.

The fact that reflexive oriented trees admit a majority polymorphism is from [42] (the majority operation defined in the undirected case in Corollary 2.58 of [51] respects orientations.)

Reflexive trees also admit a semilattice polymorphism [88]; hence the problem $CSP(\mathbb{H}^{+c})$ has width 1. We note in passing that there exist reflexive graphs whose retraction problem has width 1 but admit no semilattice polymorphism [55]; in fact, M. Siggers has recently found examples of such reflexive graphs that even admit a majority polymorphism [89]. A stronger statement than the last theorem holds for reflexive oriented cycles with at least 4 vertices, even allowing symmetric edges, which are in fact *idempotent trivial*, i.e. their only idempotent polymorphisms are projections [69]. The theorem as well as this last result are proved using a natural topological structure underlying reflexive digraphs; topological methods have also been used to study polymorphisms on digraphs in [30], [34], [74], [79].

Gumm terms characterise the important property of congruence modularity in varieties of algebras; by a result of Barto [7], a digraph $\mathbb{H}$ admits Gumm polymorphisms exactly when it admits edge or cube terms, i.e. when $CSP(\mathbb{H}^{+c})$ has few subpowers (and hence is tractable.) In general the existence of Gumm polymorphisms does not imply the existence of NU polymorphisms (although the converse is true); the next result shows that for reflexive digraphs these conditions are actually equivalent. This result generalises [84] and [77] which previously proved it for bounded posets and general posets respectively. The fact that width 1 is implied by the presence of an NU polymorphism was first proved for posets in [78].

[1] For completeness' sake we note briefly the behaviour of the remaining reflexive cycles: the directed cycle $\mathbb{H} = \langle\{0, 1, 2\}; \theta\rangle$ with $\theta = \{(0, 0), (1, 1), (2, 2), (0, 1), (1, 2), (2, 0)\}$ is idempotent trivial, and all other 3-cycles admit either a majority or a semilattice polymorphism.

▶ **Theorem 22** ([81]). *Let $\mathbb{H}$ be a connected reflexive digraph. Then the following are equivalent:*
1. *$\mathbb{H}$ admits Gumm polymorphisms;*
2. *$\mathbb{H}$ admits an NU polymorphism.*

If these conditions hold, then $\mathbb{H}$ admits idempotent TS polymorphisms of all arities, and hence $CSP(\mathbb{H}^{+c})$ has width 1.

Combined with Theorem 42 of [27], it follows from the last statement that reflexive digraphs admitting a majority polymorphism are precisely those for which $CSP(\mathbb{H}^{+c})$ has so-called *path duality*.

Reflexive digraphs whose retraction problem is first-order definable have a nice description:

▶ **Theorem 23** ([73]). *Let $\mathbb{H}$ be a connected reflexive digraph. If $\mathbb{H}$ is k-permutable for some $k \geq 2$ then it is strongly connected. Consequently, the following are equivalent:*
1. *$CSP(\mathbb{H}^{+c})$ is first-order definable;*
2. *$\mathbb{H}$ is strongly connected and admits an NU polymorphism.*

The *sum* $\mathbb{G} \oplus \mathbb{H}$ of two reflexive digraphs $\mathbb{G}$ and $\mathbb{H}$ is the digraph obtained from the disjoint union of the digraphs, adding all arcs of the form (g, h) with $g \in G$ and $h \in H$. A reflexive digraph is *series-parallel* if it can be obtained from copies of the one element digraph using disjoint unions and sums. It is easy to see that such a digraph is in fact a poset. Equivalently, a poset is series-parallel if it is *N-free*, i.e. it does not contain the digraph $\mathbb{Z}$ (defined in section 3.2) as an induced subdigraph. The following result describes a dichotomy for series-parallel posets; the tractable posets are also characterised by a finite list of forbidden retracts, as well as a simple topological condition.

▶ **Theorem 24** ([30]). *Let $\mathbb{H}$ be a connected, series-parallel poset. Then the following are equivalent:*
1. *$\mathbb{H}$ admits a WNU polymorphism;*
2. *$\mathbb{H}$ admits idempotent TS polymorphisms of all arities $k \geq 2$.*

If these conditions hold, then $CSP(\mathbb{H}^{+c})$ has width 1, otherwise it is NP-complete.

There is a similar characterisation of series-parallel posets admitting an NU polymorphism [76]. For other work on the study of polymorphisms on posets the reader may consult the references of [84], [79] and [77]. There also has been extensive work on the study of polymorphisms on reflexive graphs, but most results relevant to this survey can be obtained as special cases of the above results of reflexive digraphs; for instance the analogs of Theorems 22 and 23 were first proved for reflexive graphs in [72] and [31]. [80] contains various interesting examples, [44] describes explicit generators for the variety of reflexive graphs, [19] studies NU polymorphisms on reflexive graphs, and [55, 87] investigate semilattice and lattice polymorphisms on these same graphs. [17] studies the idempotent polymorphisms of digraphs with at most 5 vertices.

6 $CSP(\mathbb{H}^{+l})$

Recall that the polymorphisms of the structure $\mathbb{H}^{+l}$ are precisely the conservative polymorphisms of $\mathbb{H}$. The proof of the algebraic dichotomy conjecture for the conservative case is due to Bulatov (see [8] for an alternative proof):

▶ **Theorem 25** ([22]). *Let $\mathbb{H}$ be a structure. If $\mathbb{H}$ admits a conservative WNU polymorphism then $CSP(\mathbb{H}^{+l})$ is tractable, otherwise it is NP-complete.*

It turns out that a stronger result holds for structures whose basic relations are at most binary:

▶ **Theorem 26** ([64]). *Let $\mathbb{H}$ be a structure whose basic relations are at most binary. If $\mathbb{H}$ admits a conservative WNU polymorphism then $CSP(\mathbb{H}^{+l})$ has bounded width.*

Hell and Rafiey had obtained this result earlier in the case of digraph CSPs (i.e. for a single binary relation) [52], as a by-product of a graph-theoretic description of the tractable cases, in terms of *digraph asteroidal triples (DAT)*; because the definition of a DAT is rather involved and technical we do not give it here. In a very recent paper [54], Hell and Rafiey have characterised digraphs admitting a conservative semilattice polymorphism; the following result is implicit in their proof, and shows that there is quite a bit of collapse for digraphs in the conservative case. Note that the equivalence of the last two conditions does not hold for general structures, see example 99 in [67].

▶ **Theorem 27** ([54]). *Let $\mathbb{H}$ be a digraph. Then the following are equivalent:*
1. *$\mathbb{H}$ admits a conservative semilattice polymorphism;*
2. *$\mathbb{H}$ admits conservative cyclic polymorphisms of all arities;*
3. *$\mathbb{H}$ admits conservative symmetric polymorphisms of all arities;*
4. *$\mathbb{H}$ admits conservative TS polymorphisms of all arities, i.e. $CSP(\mathbb{H}^{+l})$ has width 1.*

The logspace conjecture (Conjecture 8 (3)) has been verified for at most binary structures [29]; here we state the graph-theoretic description of the digraphs with $CSP(\mathbb{H}^{+l})$ definable in symmetric Datalog which is from [36].

Let $\mathbb{H}$ be a digraph, and let $x, y \in H$. We say that (x, y) is an *edge* if either (x, y) or (y, x) is an arc of $\mathbb{H}$. A sequence of vertices $x_0, \dots, x_n$, $(n \geq 0)$ in $\mathbb{H}$ such that (x_i, x_{i+1}) is an edge for all $0 \leq i \leq n-1$ is called a *walk* in $\mathbb{H}$ from x_0 to x_n; we call the pair (x_i, x_{i+1}) a *forward (backward) edge* if (x_i, x_{i+1}) ((x_{i+1}, x_i) respectively) is an arc. Two walks $P = x_0, \dots, x_n$ and $Q = y_0, \dots, y_n$ in $\mathbb{H}$ are *congruent*, if they follow the same pattern of forward and backward edges, i.e., when (x_i, x_{i+1}) is an arc if and only if (y_i, y_{i+1}) is an arc. Suppose P, Q and $R = z_0, \dots, z_n$ are pairwise congruent walks. We say that (x_i, y_{i+1}) is a *faithful edge from P to Q* if it is an edge of $\mathbb{H}$ in the same direction (forward or backward) as (x_i, x_{i+1}). We say that P *avoids* Q in $\mathbb{H}$ if there is no faithful edge from P to Q; R *protects* Q from P if the existence of faithful edges (x_i, z_{i+1}) and (z_j, y_{j+1}) implies that $j \leq i$. The digraph $\mathbb{H}$ *contains a circular N* if there exist vertices $x, y \in H$, congruent walks P from x to x, Q from y to y and R from y to x such that P avoids Q and R protects Q from P.

▶ **Theorem 28** ([29, 36]). *Let $\mathbb{H}$ be a digraph. Then the following are equivalent:*
1. *$\mathbb{H}$ does not contain a circular N;*
2. *$\mathbb{H}$ is k-permutable for some $k \geq 2$;*
3. *$CSP(\mathbb{H}^{+l})$ is definable in symmetric Datalog.*

If one of these conditions holds then $CSP(\mathbb{H}^{+l})$ is solvable in logspace, otherwise it is NL hard.

[35] contains related results on oriented trees; digraphs that admit a conservative semilattice polymorphism are characterised in [53]. The digraphs with first-order definable list homomorphism problem also admit a nice graph-theoretic description [59]: two arcs (x_1, y_1) and (x_2, y_2) of a digraph $\mathbb{H}$ are said to be *separated* if neither (x_1, y_2) nor (x_2, y_1) is an arc of $\mathbb{H}$. A *hindering bicycle in* $\mathbb{H}$ is a subset $\{x_0, \dots, x_n, y_0, \dots, y_n\}$ of vertices of $\mathbb{H}$ $(n \geq 0)$ such that (i) (x_i, x_{i+1}), (y_i, y_{i+1}) and (x_i, y_{i+1}) are arcs of $\mathbb{H}$ for all $i = 0, \dots, n$ (indices modulo $n + 1$) and (ii) (x_{i+1}, y_i) is not an arc of $\mathbb{H}$ for any $i = 0, \dots, n$ (indices modulo $n + 1$).

▶ **Theorem 29** ([59]). *Let $\mathbb{H}$ be a digraph. Then the following are equivalent:*
1. *$\mathbb{H}$ contains no separated arcs nor any hindering bicycle;*
2. *$CSP(\mathbb{H}^{+l})$ is first-order definable.*

The special case of graphs (with loops allowed) in interesting in its own right. The algebraic dichotomy conjecture for list homomorphism problems has a very neat dividing line in this context: the graphs such that $CSP(\mathbb{H}^{+l})$ is tractable are the so-called *bi-arc graphs* [41], which are precisely the graphs that admit a conservative majority polymorphism [19]. In [70] it is shown that among these, the graphs whose list homomorphism problem has width 1 are the bi-arc graphs that do not have a loopless edge; equivalently, these are the graphs that admit a *binary* conservative WNU polymorphism.

Since the presence of a majority polymorphism guarantees the CSP is definable in linear Datalog, Conjecture 8 (2) for the list homomorphism problem on graphs follows from the above; the proof of Conjecture 8 (3) in this special case can be found in [37]; an explicit description by finitely many forbidden subgraphs is given for the graphs $\mathbb{H}$ such that $CSP(\mathbb{H}^{+l})$ is definable in symmetric Datalog.

7 Open Problems and Further Discussion

We list, in no particular order, some open questions and problems, as well as further discussion of the results presented earlier.

1. If $\mathbb{H}$ is an oriented tree such that $CSP(\mathbb{H})$ is tractable, does it also have bounded width [25]?
2. There is very little known about digraphs admitting (conservative) cube or edge terms, i.e. such that the problems $CSP(\mathbb{H})$, $CSP(\mathbb{H}^{+c})$ and $CSP(\mathbb{H}^{+l})$ have few subpowers. Investigate.
3. Characterise those digraphs $\mathbb{H}$ whose list homomorphism problem is definable in linear Datalog and confirm Conjecture 8 (2) in this case.
4. Give a (simple?) graph-theoretic characterisation of digraphs that admit a conservative NU polymorphism.
5. Which posets admit a semilattice polymorphism? Does there exist a poset that admits TS polymorphisms of all arities, or even an NU polymorphism, but no semilattice polymorphism?
6. There exist posets whose retraction problem is tractable but does not have bounded width [68] [2], but the ones that are known are quite large. Find small examples of such posets. Same question for reflexive graphs.
7. There exists an acyclic digraph $\mathbb{H}$ such that $CSP(\mathbb{H})$ is tractable but does not have bounded width [3] but it is quite large; find some amenable examples.
8. M. Maróti [82] has analysed small reflexive digraphs by computer and obtained several 6-element examples whose retraction problem has bounded width but not width 1; there are no such examples for posets nor reflexive graphs of size at most 8. Investigate.
9. Topological methods would appear promising in the analysis of polymorphisms on reflexive digraphs, but there has been only preliminary work in this direction. For instance, is there a characterisation of reflexive digraphs admitting a WNU polymorphism via homotopy groups of idempotent subalgebras analogous to the case of posets? See the remarks after Corollary 4.5 in [77], but see also Proposition 1.3 of [72].

[2] L. Barto has verified that the example there is indeed tractable without the assumption that $P \neq NP$.

10. The complexity of deciding if a relational structure admits such and such "nice" polymorphism has been investigated in [28]. For many identities, the hardness results for general structures are still valid for structures with at most binary basic relations; however, for a single binary relation, i.e. a digraph, the problem often turns out to be better behaved. Investigate.

Acknowledgements. The author wishes to thank Gary MacGillivray, Miklós Maróti, Mark Siggers, Matt Valeriote and Ross Willard for useful discussions.

References

1 Eric Allender, Michael Bauland, Neil Immerman, Henning Schnoor, and Heribert Vollmer. The complexity of satisfiability problems: refining Schaefer's theorem. In *Mathematical foundations of computer science 2005*, volume 3618 of *Lecture Notes in Comput. Sci.*, pages 71–82. Springer, Berlin, 2005. `doi:10.1007/11549345_8`.

2 Sanjeev Arora and Boaz Barak. *Computational complexity.* Cambridge University Press, Cambridge, 2009. A modern approach. `doi:10.1017/CBO9780511804090`.

3 Albert Atserias. On digraph coloring problems and treewidth duality. *European J. Combin.*, 29(4):796–820, 2008. `doi:10.1016/j.ejc.2007.11.004`.

4 Jørgen Bang-Jensen and Pavol Hell. The effect of two cycles on the complexity of colourings by directed graphs. *Discrete Appl. Math.*, 26(1):1–23, 1990. `doi:10.1016/0166-218X(90)90017-7`.

5 Jørgen Bang-Jensen, Pavol Hell, and Gary MacGillivray. The complexity of colouring by semicomplete digraphs. *SIAM J. Discrete Math.*, 1(3):281–298, 1988. `doi:10.1137/0401029`.

6 Jørgen Bang-Jensen, Gary MacGillivray, and Jacobus Swarts. The complexity of colouring by locally semicomplete digraphs. *Discrete Math.*, 310(20):2675–2684, 2010. `doi:10.1016/j.disc.2010.03.033`.

7 L. Barto. Finitely related algebras in congruence modular varieties have few subpowers. to appear in JEMS.

8 Libor Barto. The dichotomy for conservative constraint satisfaction problems revisited. In *26th Annual IEEE Symposium on Logic in Computer Science – LICS 2011*, pages 301–310. IEEE Computer Soc., Los Alamitos, CA, 2011.

9 Libor Barto. The collapse of the bounded width hierarchy. *Journal of Logic and Computation*, 2014. `doi:10.1093/logcom/exu070`.

10 Libor Barto. The constraint satisfaction problem and universal algebra. *Bull. Symb. Log.*, 21(3):319–337, 2015. `doi:10.1017/bsl.2015.25`.

11 Libor Barto and Jakub Bulín. CSP dichotomy for special polyads. *Internat. J. Algebra Comput.*, 23(5):1151–1174, 2013. `doi:10.1142/S0218196713500215`.

12 Libor Barto and Marcin Kozik. Absorbing subalgebras, cyclic terms, and the constraint satisfaction problem. *Log. Methods Comput. Sci.*, 8(1):1:07, 27, 2012. `doi:10.2168/LMCS-8(1:7)2012`.

13 Libor Barto and Marcin Kozik. Constraint satisfaction problems solvable by local consistency methods. *J. ACM*, 61(1):3, 2014.

14 Libor Barto, Marcin Kozik, Miklós Maróti, and Todd Niven. CSP dichotomy for special triads. *Proc. Amer. Math. Soc.*, 137(9):2921–2934, 2009. `doi:10.1090/S0002-9939-09-09883-9`.

15 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, 2008/09. `doi:10.1137/070708093`.

16 Libor Barto, Marcin Kozik, and Ross Willard. Near unanimity constraints have bounded pathwidth duality. In *Proceedings of the 2012 27th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 125–134. IEEE Computer Soc., Los Alamitos, CA, 2012. doi:10.1109/LICS.2012.24.

17 Libor Barto and David Stanovský. Polymorphisms of small digraphs. *Novi Sad J. Math.*, 40(2):95–109, 2010.

18 J. Berman, P. Idziak, P. Markovic, R. McKenzie, M. Valeriote, and R. Willard. Varieties with few subalgebras of powers. *Transactions of the American Mathematical Society*, 362(3):1445–1473, 2010.

19 Richard C. Brewster, Tomas Feder, Pavol Hell, Jing Huang, and Gary MacGillivray. Near-unanimity functions and varieties of reflexive graphs. *SIAM J. Discrete Math.*, 22(3):938–960, 2008. doi:10.1137/S0895480103436748.

20 A. Bulatov, P. Jeavons, and A. Krokhin. Classifying the Complexity of Constraints using Finite Algebras. *SIAM Journal on Computing*, 34(3):720–742, 2005.

21 Andrei A. Bulatov. H-coloring dichotomy revisited. *Theoret. Comput. Sci.*, 349(1):31–39, 2005. doi:10.1016/j.tcs.2005.09.028.

22 Andrei A. Bulatov. Complexity of conservative constraint satisfaction problems. *ACM Trans. Comput. Log.*, 12(4):24, 2011.

23 Andrei A. Bulatov, Andrei A. Krokhin, and Benoit Larose. Dualities for constraint satisfaction problems. In *Complexity of Constraints – An Overview of Current Research Themes (Result of a Dagstuhl Seminar)*, volume 5250 of *Lecture Notes in Computer Science*, pages 93–124. Springer, 2008. doi:10.1007/978-3-540-92800-3_5.

24 Andrei A. Bulatov and Matthew Valeriote. Recent results on the algebraic approach to the CSP. In *Complexity of Constraints – An Overview of Current Research Themes (Result of a Dagstuhl Seminar)*, volume 5250 of *Lecture Notes in Computer Science*, pages 68–92. Springer, 2008. doi:10.1007/978-3-540-92800-3_4.

25 Jakub Bulín. On the complexity of h-coloring for special oriented trees. arXiv:1407.1779v2, 2014.

26 Jakub Bulín, Dejan Delić, Marcel Jackson, and Todd Niven. A finer reduction of constraint problems to digraphs. *Log. Methods Comput. Sci.*, 11(4):4:18, 33, 2015.

27 Catarina Carvalho, Víctor Dalmau, and Andrei Krokhin. Two new homomorphism dualities and lattice operations. *J. Logic Comput.*, 21(6):1065–1092, 2011. doi:10.1093/logcom/exq030.

28 Hubie Chen and Benoit Larose. Asking the metaquestions in constraint tractability. arXiv:1604.00932, 2016.

29 Víctor Dalmau, László Egri, Pavol Hell, Benoit Larose, and Arash Rafiey. Descriptive complexity of list h-coloring problems in logspace: A refined dichotomy. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015*, pages 487–498, 2015. doi:10.1109/LICS.2015.52.

30 Víctor Dalmau, Andrei Krokhin, and Benoit Larose. Retractions onto series-parallel posets. *Discrete Math.*, 308(11):2104–2114, 2008. doi:10.1016/j.disc.2006.08.010.

31 Víctor Dalmau, Andrei A. Krokhin, and Benoit Larose. First-order definable retraction problems for posets and reflexive graph. In *19th IEEE Symposium on Logic in Computer Science (LICS 2004), 14-17 July 2004, Turku, Finland, Proceedings*, pages 232–241, 2004. doi:10.1109/LICS.2004.1319617.

32 Víctor Dalmau and Benoit Larose. Maltsev + datalog –> symmetric datalog. In *Proceedings of the Twenty-Third Annual IEEE Symposium on Logic in Computer Science, LICS 2008, 24-27 June 2008, Pittsburgh, PA, USA*, pages 297–306, 2008. doi:10.1109/LICS.2008.14.

33 Victor Dalmau and Justin Pearson. Closure Functions and Width 1 Problems. In *CP 1999*, pages 159–173, 1999.

34 C. Delhommé. Projection properties and reflexive binary relations. *Algebra Universalis*, 41(4):255–281, 1999. doi:10.1007/s000120050115.

35 Lászlo Egri. Space complexity of list h-coloring revisited: the case of oriented trees. arXiv:1510.07124, 2015.

36 László Egri, Pavol Hell, Benoit Larose, and Arash Rafiey. Space complexity of list H-colouring: a dichotomy. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 349–365, 2014. doi:10.1137/1.9781611973402.26.

37 László Egri, Andrei Krokhin, Benoit Larose, and Pascal Tesson. The complexity of the list homomorphism problem for graphs. *Theory Comput. Syst.*, 51(2):143–178, 2012. doi:10.1007/s00224-011-9333-8.

38 T. Feder and M. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory. *SIAM Journal on Computing*, 28:57–104, 1999.

39 Tomás Feder. Classification of homomorphisms to oriented cycles and of k-partite satisfiability. *SIAM J. Discrete Math.*, 14(4):471–480 (electronic), 2001. doi:10.1137/S0895480199383353.

40 Tomas Feder and Pavol Hell. List homomorphisms to reflexive graphs. *J. Combin. Theory Ser. B*, 72(2):236–250, 1998. doi:10.1006/jctb.1997.1812.

41 Tomas Feder, Pavol Hell, and Jing Huang. Bi-arc graphs and the complexity of list homomorphisms. *J. Graph Theory*, 42(1):61–80, 2003. doi:10.1002/jgt.10073.

42 Tomas Feder, Pavol Hell, and Jing Huang. List homomorphisms and retractions to reflexive digraphs. Manuscript, 2007.

43 Tomás Feder, Pavol Hell, Peter Jonsson, Andrei Krokhin, and Gustav Nordh. Retractions to pseudoforests. *SIAM J. Discrete Math.*, 24(1):101–112, 2010. doi:10.1137/080738866.

44 Tomás Feder, Pavol Hell, Benoît Larose, Cynthia Loten, Mark Siggers, and Claude Tardif. Graphs admitting k-NU operations. Part 1: The reflexive case. *SIAM J. Discrete Math.*, 27(4):1940–1963, 2013. doi:10.1137/120894312.

45 Tomás Feder, Pavol Hell, Benoît Larose, Mark Siggers, and Claude Tardif. Graphs admitting k-NU operations. Part 2: The irreflexive case. *SIAM J. Discrete Math.*, 28(2):817–834, 2014. doi:10.1137/130914784.

46 Tomás Feder and Moshe Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: a study through Datalog and group theory. *SIAM J. Comput.*, 28(1):57–104 (electronic), 1999. doi:10.1137/S0097539794266766.

47 Ralph Freese and Ralph McKenzie. Maltsev families of varieties closed under join or maltsev product. Preprint, 2015.

48 Wolfgang Gutjahr, Emo Welzl, and Gerhard Woeginger. Polynomial graph-colorings. *Discrete Appl. Math.*, 35(1):29–45, 1992. doi:10.1016/0166-218X(92)90294-K.

49 P. Hell, J. Nešetřil, and X. Zhu. Complexity of tree homomorphisms. *Discrete Appl. Math.*, 70(1):23–36, 1996. doi:10.1016/0166-218X(96)00099-6.

50 Pavol Hell and Jaroslav Nešetřil. On the complexity of H-coloring. *J. Combin. Theory Ser. B*, 48(1):92–110, 1990. doi:10.1016/0095-8956(90)90132-J.

51 Pavol Hell and Jaroslav Nešetřil. *Graphs and homomorphisms*, volume 28 of *Oxford Lecture Series in Mathematics and its Applications*. Oxford University Press, Oxford, 2004. doi:10.1093/acprof:oso/9780198528173.001.0001.

52 Pavol Hell and Arash Rafiey. The dichotomy of list homomorphisms for digraphs. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1703–1713. SIAM, Philadelphia, PA, 2011.

53 Pavol Hell and Arash Rafiey. Monotone proper interval digraphs and min-max orderings. *SIAM J. Discrete Math.*, 26(4):1576–1596, 2012. doi:10.1137/100783844.

54 Pavol Hell and Arash Rafiey. Bi-arc digraphs and conservative polymorphisms. arXiv:1608.03368, 2016.

55 Pavol Hell and Mark Siggers. Semilattice polymorphisms and chordal graphs. *European J. Combin.*, 36:694–706, 2014. `doi:10.1016/j.ejc.2013.10.007`.

56 Pavol Hell and Xu Ding Zhu. The existence of homomorphisms to oriented cycles. *SIAM J. Discrete Math.*, 8(2):208–222, 1995. `doi:10.1137/S0895480192239992`.

57 D. Hobby and R. McKenzie. *The Structure of Finite Algebras*, volume 76 of *Contemporary Mathematics*. American Mathematical Society, 1988.

58 P. Idziak, P. Markovic, R. McKenzie, M. Valeriote, and R. Willard. Tractability and learnability arising from algebras with few subpowers. *SIAM J. Comput.*, 39(7):3023–3037, 2010.

59 Adrien Lemaître. *Complexité des homomorphismes de graphes avec listes.* PhD thesis, Université de Montréal (Canada), 2012.

60 Marcel Jackson, Tomasz Kowalski, and Todd Niven. Complexity and polymorphisms for digraph constraint problems under some basic constructions. to appear in Internat. J. Algebra Comput. (arXiv:1304.4986), 2016.

61 P. Jeavons, D. Cohen, and M. Cooper. Constraints, consistency, and closure. *Artificial Intelligence*, 101(1-2):251–265, 1998.

62 Peter Jeavons. On the algebraic structure of combinatorial problems. *Theoretical Computer Science*, 200:185–204, 1998.

63 Alexandr Kazda. Maltsev digraphs have a majority polymorphism. *European Journal of Combinatorics*, 32(3):390–397, 2011.

64 Alexandr Kazda. CSP for binary conservative relational structures. *Algebra Universalis*, 75(1):75–84, 2016. `doi:10.1007/s00012-015-0358-8`.

65 Alexandr Kazda. n-permutability and linear datalog implies symmetric datalog. arXiv:1508.05766v1, 2016.

66 Marcin Kozik, Andrei Krokhin, Matt Valeriote, and Ross Willard. Characterizations of several Maltsev conditions. *Algebra Universalis*, 73(3-4):205–224, 2015. `doi:10.1007/s00012-015-0327-2`.

67 Gábor Kun and Mario Szegedy. A new line of attack on the dichotomy conjecture. *European J. Combin.*, 52(part B):338–367, 2016. `doi:10.1016/j.ejc.2015.07.011`.

68 B. Larose and L. Zádori. Bounded width problems and algebras. *Algebra Universalis*, 56(3-4):439–466, 2007.

69 Benoit Larose. Taylor operations on finite reflexive structures. *Int. J. Math. Comput. Sci.*, 1(1):1–21, 2006.

70 Benoît Larose and Adrien Lemaître. List-homomorphism problems on graphs and arc consistency. *Discrete Math.*, 313(22):2525–2537, 2013. `doi:10.1016/j.disc.2013.07.018`.

71 Benoit Larose, Cynthia Loten, and Claude Tardif. A characterisation of first-order constraint satisfaction problems. *Log. Methods Comput. Sci.*, 3(4):4:6, 22, 2007. `doi:10.2168/LMCS-3(4:6)2007`.

72 Benoit Larose, Cynthia Loten, and László Zádori. A polynomial-time algorithm for near-unanimity graphs. *J. Algorithms*, 55(2):177–191, 2005. `doi:10.1016/j.jalgor.2004.04.011`.

73 Benoit Larose and Mark Siggers. Reflexive digraphs admitting nu polymorphisms. In preparation.

74 Benoit Larose and Claude Tardif. A discrete homotopy theory for binary reflexive structures. *Adv. Math.*, 189(2):268–300, 2004. `doi:10.1016/j.aim.2003.11.011`.

75 Benoît Larose and Pascal Tesson. Universal algebra and hardness results for constraint satisfaction problems. *Theoret. Comput. Sci.*, 410(18):1629–1647, 2009. `doi:10.1016/j.tcs.2008.12.048`.

76 Benoit Larose and Ross Willard. Nu series parallel posets. preprint, 2016.

77 Benoit Larose and László Zádori. Algebraic properties and dismantlability of finite posets. *Discrete Math.*, 163(1-3):89–99, 1997. `doi:10.1016/0012-365X(95)00312-K`.

78 Benoit Larose and László Zádori. The complexity of the extendibility problem for finite posets. *SIAM J. Discrete Math.*, 17(1):114–121 (electronic), 2003. `doi:10.1137/S0895480101389478`.

79 Benoit Larose and László Zádori. Finite posets and topological spaces in locally finite varieties. *Algebra Universalis*, 52(2-3):119–136, 2004. `doi:10.1007/s00012-004-1819-7`.

80 Cynthia Loten. *Retractions of chordal and related graphs.* PhD thesis, Simon Fraser University (Canada), 2004.

81 M. Maróti and L. Zádori. Reflexive digraphs with near unanimity polymorphisms. *Discrete Math.*, 312(15):2316–2328, 2012. `doi:10.1016/j.disc.2012.03.040`.

82 Miklós Maróti. Personal communication, 2016.

83 Miklós Maróti and Ralph McKenzie. Existence theorems for weakly symmetric operations. *Algebra Universalis*, 59(3-4):463–489, 2008. `doi:10.1007/s00012-008-2122-9`.

84 Ralph McKenzie. Monotone clones, residual smallness and congruence distributivity. *Bull. Austral. Math. Soc.*, 41(2):283–300, 1990. `doi:10.1017/S0004972700018104`.

85 Jaroslav Nešetřil and Claude Tardif. Duality theorems for finite structures (characterising gaps and good characterisations). *J. Combin. Theory Ser. B*, 80(1):80–97, 2000. `doi:10.1006/jctb.2000.1970`.

86 Christos H. Papadimitriou. *Computational complexity.* Addison-Wesley Publishing Company, Reading, MA, 1994.

87 Mark Siggers. Distributive lattice polymorphism on reflexive graphs. arXiv:1411.7879, 2014.

88 Mark Siggers, 2016. Personal communication.

89 Mark Siggers. Reflexive graphs with near-unanimity but no semilattice polymorphisms. preprint, 2016.

90 Narayan Vikas. Computational complexity classification of partition under compaction and retraction. In *Computing and combinatorics*, volume 3106 of *Lecture Notes in Comput. Sci.*, pages 380–391. Springer, Berlin, 2004. `doi:10.1007/978-3-540-27798-9_41`.

91 Ross Willard, 2015. Personal communication.

92 Alexander Wires. Dichotomy for finite tournaments of mixed-type. *Discrete Math.*, 338(12):2523–2538, 2015. `doi:10.1016/j.disc.2015.06.024`.

93 Alexander Wires. A quasi-Mal'cev condition with unexpected application. *Algebra Universalis*, 73(3-4):335–346, 2015. `doi:10.1007/s00012-015-0322-7`.

Approximation Algorithms for CSPs

Konstantin Makarychev[1] and Yury Makarychev[2]

1 **Microsoft Research, Redmond, WA, USA**
komakary@microsoft.com
2 **Toyota Technological Institute at Chicago, Chicago, IL, USA**
yury@ttic.edu

Abstract

In this survey, we offer an overview of approximation algorithms for constraint satisfaction problems (CSPs) – we describe main results and discuss various techniques used for solving CSPs.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Constraint satisfaction problems, Approximation algorithms, SDP, UGC

Digital Object Identifier 10.4230/DFU.Vol7.15301.287

1 Introduction

We start with recalling standard definitions and introducing the notation.

Constraint Satisfaction Problems. In a constraint satisfaction problem (CSP), we are given a set of variables $x_1, \dots, x_n$ taking values in a domain D of size d, and a set of m constraints (predicates) that depend on the specific problem at hand. Our goal is to find an assignment to the variables that maximizes the number of satisfied constraints. In a weighted CSP, every constraint has a positive weight and our goal is to maximize the total weight of satisfied constraints. All results that we discuss in this survey apply to both unweighted and weighted CSPs. However, for simplicity of exposition, we will only consider the unweighted case. We will say that a CSP is a k-CSP if all constraints have arity at most k.

An instance is $(1-\varepsilon)$-satisfiable if the optimal solution satisfied at least a $(1-\varepsilon)$ fraction of the constraints.

Approximation Algorithms. An approximation algorithm is a (randomized) polynomial-time algorithm that finds an approximate solution. The most common measure of an approximation algorithm's performance is its approximation factor. An algorithm for a maximization problem has an approximation factor $\alpha \leq 1$ if it finds a solution of value at least $\alpha\,\mathsf{OPT}$, where OPT is the value of the optimal solution; an algorithm for a minimization problem has an approximation factor $\alpha \geq 1$ if it finds a solution of value at most $\alpha\,\mathsf{OPT}$. We will say that an algorithm is an α-approximation algorithm if it has an approximation factor of α.

Objectives. We consider several objectives for constraint satisfaction problems:

1. Maximize the number of *satisfied* constraints. An α-approximation algorithm for this objective finds a solution that satisfies at least $\alpha\,\mathsf{OPT}$ constraints.
2. Find a solution that satisfies a $1-f(\varepsilon)$ fraction of the constraints given a $(1-\varepsilon)$-satisfiable instance; where f is some function that tends to 0 as $\varepsilon \to 0$ (f should not depend on n).

3. Minimize the number of *unsatisfied* constraints. An α-approximation algorithm for this objective finds a solution that satisfies at least a $(1-\alpha\varepsilon)$ fraction of the constraints given a $(1-\varepsilon)$-satisfiable instance; the approximation factor α may depend on n.

Approximation results for these objectives are often very different. In particular, it makes sense to study objectives (2) and (3) for a given CSP only if there is a polynomial-time algorithm that satisfies all the constraints when all of them are satisfiable. Consider an example – Max 2-Lin(2). This problem is a Boolean CSP of arity 2 with constraints of the form $x_i \oplus x_j = c$ (the problem is a generalization of Max Cut). We can get the following results for objectives (1)–(3): obtain a 0.87856-approximation for the maximization variant of the problem [18], satisfy a $1 - O(\sqrt{\varepsilon})$ fraction of the constraints if the optimal solution satisfies a $1-\varepsilon$ fraction of the constraints [18], and get an $O(\sqrt{\log n})$-approximation for the minimization variant [1]. The first result applies to all instances of Max 2-Lin(2); however, the guarantee it provides for almost satisfiable instances is very weak – even if the instance is completely satisfiable it only guarantees that a 0.87856 fraction of the constraints is satisfied. In contrast, the second result is most interesting for almost satisfiable instances; it guarantees that in such instances the algorithm satisfies almost all constraints. Finally, the third result is meaningful only when the instance is $(1 - c/\log n)$ satisfiable, and, is particularly interesting when $\varepsilon \ll 1/\log^2 n$ – then it gives a much better approximation guarantee than the second result.

Techniques. Most state-of-the-art approximation algorithms for constraint satisfaction problems – with the notable exception of the algorithms for Minimum Horn Deletion and Minimum Multiway Cut problems – are based on semidefinite programming (SDP). However, algorithms for different types of CSPs use very different techniques, and challenges that arise in designing them are quite different. The key parameters that determine what techniques to use are the arity k of the CSP, the domain size d, and the objective.

Boolean 2-CSPs have the simplest SDP relaxations: each variable x_i is encoded by a unit vector $\bar{u}_i$; in the intended integral solution $\bar{u}_i$ is equal to a fixed unit vector $\bar{v}_0 \in S^{n-1}$ if x_i is true, and $\bar{u}_i$ is equal to $-\bar{v}_0$ if x_i is false. The SDP objective function equals the sum of contributions of individual constraints. The contribution of a constraint $\phi(x_i, x_j)$ is

$$\sum_{\alpha,\beta\in\{0,1\}:\phi(\alpha,\beta)} \frac{\langle \bar{v}_0 - (-1)^{\alpha}\bar{u}_i,\, \bar{v}_0 - (-1)^{\beta}\bar{u}_j\rangle}{4}$$
$$= \frac{1}{4}\sum_{\alpha,\beta\in\{0,1\}:\phi(\alpha,\beta)} \big(1 + (-1)^{\alpha+\beta}\langle \bar{u}_i, \bar{u}_j\rangle - (-1)^{\alpha}\langle \bar{u}_i, \bar{v}_0\rangle - (-1)^{\beta}\langle \bar{u}_j, \bar{v}_0\rangle\big) \quad (1)$$

Here, the summation is over Boolean values α and β that satisfy the predicate $\phi(\alpha,\beta)$; 0 and 1 represent false and true, respectively. For example, the contribution of the constraint $x_i \oplus x_j = 0$ is $(1 + \langle \bar{u}_i, \bar{u}_j\rangle)/2$, the contribution of $x_i \vee x_j$ is $(3 + \langle \bar{u}_i + \bar{u}_j, \bar{v}_0\rangle - \langle \bar{u}_i, \bar{u}_j\rangle)/4$. The SDP relaxation has constraints $\|\bar{u}_i\|^2 = 1$ and, possibly, some additional constraints that depend on the CSP.

Let us consider how an SDP algorithm works at a high level. We solve the SDP relaxation and find a (nearly) optimal SDP solution $\{\bar{u}_i\}$. The SDP solution may be very different from the intended solution; in particular, vectors $\{\bar{u}_i\}$ do not have to be equal or close to vectors $\bar{v}_0$ or $-\bar{v}_0$. We use a randomized rounding procedure to transform the set of vectors $\bar{u}_i$ to a Boolean assignment for variables x_i. To ensure that the value of the obtained assignment is large, we want to transform near-by vectors to the same value with a high probability and antipodal vectors to opposite values. For some problems, we do the rounding in one step; for other problems, we use an iterative procedure to do the rounding. In the former case, it is instructive to think of the rounding procedure as consisting of two actions:

- Generate a random partition $(A, S^{n-1} \setminus A)$ of the unit sphere S^{n-1} into two pieces, which is symmetric about the origin (that is, $A = -(S^{n-1} \setminus A)$).
- Assign $x_i = 1$ if $x \in A$, and $x_i = 0$ if $x \notin A$.

Usually, the distribution of random partitions of S^{n-1} does not depend on the SDP solution, except that it may depend on the value of the SDP solution. To prove an approximation guarantee for this algorithm, we need to lower bound the probability that each constraint $\phi(x_i, x_j)$ is satisfied in terms of its SDP contribution. To this end, we only have to analyze how the random partition divides vectors $\bar{u}_i$ and $\bar{u}_j$ depending on the angles between them and between them and $\bar{v}_0$.

SDP relaxations for non-Boolean 2-CSPs are more complex. Consider a 2-CSP with domain size $d > 2$; let $D = \{1, \dots, d\}$ be its domain. Note that we can no longer encode a variable x_i with only one vector u_i. Instead, we introduce d SDP vectors $\bar{u}_{i1}, \dots, \bar{u}_{id}$ for each x_i. In the intended solution, $\bar{u}_{ij} = \bar{v}_0$ if $x_i = j$ and $\bar{u}_{ij} = 0$ otherwise. The SDP contribution of the constraint $\phi(x_i, x_j)$ equals $\sum_{\alpha,\beta \in D : \phi(\alpha,\beta)} \langle \bar{u}_{i\alpha}, \bar{u}_{j\beta} \rangle$. The SDP has constraints that require that $\sum_{j=1}^{d} \bar{u}_{ij} = \bar{v}_0$ (this constraint can be written equivalently as $\left\|\sum_{j=1}^{d} \bar{u}_{ij} - \bar{v}_0\right\|^2 = 0$), $\sum_{j=1}^{d} \|\bar{u}_{ij}\|^2 = 1$, all vectors $u_{i1}, \dots, u_{id}$ are mutually orthogonal, as well as additional constraints that depend on the CSP. Informally, we can interpret $\|u_{ij}\|^2$ as the desired probability of the event $x_i = j$ and $\langle u_{i_1 j_1}, u_{i_2 j_2} \rangle$ as the desired probability of the event $x_{i_1} = j_1$ and $x_{i_2} = j_2$. Then the SDP constraints say that the sum of the probabilities of the events $x_i = j$ over all j is equal to 1, and events $x_i = j_1$ and $x_i = j_2$ are mutually exclusive.

Rounding an SDP solution for a non-Boolean 2-CSP is considerably more challenging than rounding an SDP solution for a Boolean 2-CSP. Now for each variable x_i, we want to choose exactly one vector $\bar{u}_{ij}$ among d vectors $\bar{u}_{i1}, \dots, \bar{u}_{id}$ and assign $x_i = j$. Note that we cannot simply use the same approach as before – choose a random subset A of Euclidean space, and let $x_i = j$ if $\bar{u}_{ij} \in A$, because we cannot choose a random subset A so that exactly one of any k orthogonal vectors belong to A (in contrast, it is easy to find a subset A of S^{n-1} so that exactly one of the vectors $\bar{u}$ and $-\bar{u}$ is in A). Consequently, if we try to implement such a scheme, we will sometimes assign no value or more than one value to x_j. One approach to fix this problem is to use an iterative rounding procedure:

- Find a random subset A such that the probability that two given orthogonal vectors belong to it is sufficiently small (namely, it should it be at most $1/d^c$ for some $c > 1$).
- Assign $x_i = j$, if $\bar{u}_{ij} \in A$ and there is no $j' \neq j$ such that $\bar{u}_{ij'} \in A$. Get a partial assignment to variables x_i.
- If there are unassigned variables, repeat this procedure. Do not change the values of the already assigned variables.

We note that some algorithms do not use this approach and assign values to all variables in one step (see e.g. [12]). However, as we will see in Section 3, this approach allows us to considerably simplify the algorithms' analysis; loosely speaking, to lower bound the probability that $\phi(x_i, x_j)$ is satisfied, we only need to lower bound the probabilities $\Pr(\bar{u}_{i_2 j_2} \in A \mid \bar{u}_{i_1 j_1} \in A)$ and $\Pr(\bar{u}_{i_1 j_1} \in A \mid \bar{u}_{i_2 j_2} \in A)$ for $j_1, j_2 \in D$ satisfying the constraint $\phi(j_1, j_2)$ (both probabilities are over the random choice of A). When we do that, we can restrict our attention to vectors $\bar{u}_{i_1 j_1}$ and $\bar{u}_{i_2 j_2}$, and do not have to analyze all possible spatial configurations of vectors $\bar{u}_{i_1 1}, \dots, \bar{u}_{i_1 d}, \bar{u}_{j_1 1}, \dots, \bar{u}_{j_1 d}$. Nevertheless, the analysis is still more complicated than that for Boolean 2-CSPs. Particularly, it is very important to properly handle both directions and lengths of vectors.

Rounding SDPs for most CSPs of arity $k > 2$ – and especially non-Boolean CSPs of arity $k > 2$ – poses additional challenges. The standard SDP relaxation for CSPs of arity $k > 2$ is somewhat similar to that for non-Boolean 2-CSPs; the key difference is that for each constraint $\phi(x_{i_1}, \dots, x_{i_k})$ and each satisfying assignment $x_{i_1} = j_1, \dots, x_{i_k} = j_k$ for this constraint, we have an additional SDP vector variable $\bar{v}_{(i_1,j_1),\dots,(i_k,j_k)}$. In the intended solution, $\bar{v}_{(i_1,j_1),\dots,(i_k,j_k)} = \bar{v}_0$ if $x_{i_1} = j_1, \dots, x_{i_k} = j_k$, and $\bar{v}_{(i_1,j_1),\dots,(i_k,j_k)} = 0$, otherwise. The SDP objective function equals the sum of $\|\bar{v}_{(i_1,j_1),\dots,(i_k,j_k)}\|^2$ over all variables $\bar{v}_{\dots}$. There are additional SDP constraints of the form $\langle v_{(i_1,j_1),\dots,(i_k,j_k)}, u_{i_t,j_t}\rangle = \|v_{(i_1,j_1),\dots,(i_k,j_k)}\|^2$ and $\langle v_{(i_1,j_1),\dots,(i_k,j_k)}, u_{i_t,j'}\rangle = 0$ if $j' \neq j_t$. The main challenge is that for such problems as Max k-And, to lower bound the probability that a constraint $\phi(x_{i_1}, \dots, x_{i_k})$ is satisfied, we have to analyze the spatial configuration of all dk vectors $x_{i_1 1}, \dots, x_{i_1 d}, \dots, x_{i_k 1}, \dots, x_{i_k d}$ and vector $\bar{v}_{(i_1,j_1),\dots,(i_k,j_k)}$.

Metric Embedding Techniques. Low-distortion metric embedding techniques are among the most powerful and widely used in combinatorial optimization. Not surprisingly, they are also employed for solving certain CSPs: Min UnCut, Min 2CNF Deletion, and Unique Games (in all these problems, the objective is to minimize the number of unsatisfied constraints). However, we do not describe any algorithms that use metric embeddings in this survey; we refer the reader to papers [1, 14].

1.1 Overview of Known Results for CSPs

In this section, we give an overview of known approximation results for constraint satisfaction problems.

Boolean CSPs. First, we discuss the results for Boolean CSPs with the maximization objective (objective (1) in our list). The results are summarized in Figure 1. When we describe a CSP, we write z_i to denote a literal x_i or $\bar{x}_i$. The most basic Boolean Max 2-CSP problem is Max Cut. In this problem, each constraint is of the form $x_i \neq x_j$, or, equivalently, $x_i \oplus x_j = 1$. Goemans and Williamson designed a 0.87856 approximation algorithm for the problem [18]. Later, Khot, Kindler, Mossel, and O'Donnell showed that this algorithm is optimal assuming the Unique Games Conjecture (UGC)[30]. The best unconditional hardness result was obtained by Håstad [23], who showed that it is NP-hard to obtain a better than $16/17 \approx 0.94117$ approximation (i.e., for every constant $\delta > 0$, it is impossible to get a $(16/17 - \delta)$ approximation in polynomial time if $P \neq NP$). All these results for Max Cut also apply to a more general Max 2-Lin(2) problem, a Boolean 2-CSP with constraints of the form $x_i \oplus x_j = c$ (where $c \in \{0, 1\}$).

Lewin, Livnat, and Zwick gave a 0.94016-approximation algorithm for Max 2-SAT, a problem with disjunctive constraints of the form $z_i \vee z_j$ [36]; Austrin proved that this algorithm is optimal assuming UGC [5]. Lewin et al. also designed a 0.87401-approximation algorithm for Max 2-And, a problem with conjunctive constraints of the form $z_i \wedge z_j$. This problem is the most general maximization Boolean 2-CSP – there is an approximation-preserving reduction from any Max Boolean 2-CSP to Max 2-And (thus, if there is an α-approximation for Max 2-And, then there is an α-approximation for any Boolean 2-CSP). Therefore, the algorithm by Lewin et al. gives a 0.87401 approximation for any Boolean 2-CSP. The approximation factor of 0.87401 is not known to be optimal – the best upper bound, due to Austrin [5], is 0.87435; note that the gap between the lower and upper bounds is less than 0.0004.

Now consider CSPs of greater arities. In Max 3-SAT, each constraint is a disjunction of at most 3 literals: $z_{i_1} \wedge \dots \wedge z_{i_t}$ $(t \leq 3)$; in Max E3-SAT, each constraint is a disjunction of

problem	**constraints**	**approx. factor**	**optimal?**
	(z_i is either x_i or $\bar{x}_i$)		**upper bound**
Max Cut	$x_i \neq x_j$	0.87856 [18]	yes [30]
Max 2-Lin(2)	$x_i \oplus x_j = c_{ij}$		
Max 2-SAT	$z_i \vee z_j$	0.94016 [36]	yes [5]
Max Di-Cut	$\bar{x}_i \wedge x_j$	0.87401 [36]	0.87856 [30]
Max 2-And	$z_i \wedge z_j$		0.87435 [5]
Any Boolean 2-CSP	Boolean 2-CSP		0.87435 [5]
Max 3-SAT	$\bigvee_{j=1}^{t} z_{i_j}$ $(t \leq 3)$	7/8 [28, 51]	yes [23]
Max E3-SAT	$\bigvee_{j=1}^{3} z_{i_j}$		
Any Boolean k-CSP	Boolean k-CSP	$\frac{(0.62661 - o(1))k}{2^k}$ [40]	$\frac{(1+o(1))k}{2^k}$ [6, 11]
Max k-And	$z_{i_1} \wedge \cdots \wedge z_{i_k}$		
Max SAT	$\bigvee_{j=1}^{t} z_{i_j}$	0.7968 [7] conj. 0.8434 [7]	7/8 [23]
Max Ek-SAT ($k \geq 3$)	$\bigvee_{j=1}^{k} z_{i_j}$	$1 - 1/2^k$	yes [23]
Max k-All-Equal	$z_{i_1} = \cdots = z_{i_k}$	$\frac{0.88007\,k}{2^k}$ [13]	$\frac{(2+o(1))k}{2^k}$ [6, 11]
Max k-NAE-SAT	$\overline{z_{i_1} = \cdots = z_{i_t}}$ $(t \leq k)$	0.7499 [49] conj. 0.8279 [7]	0.87856 [30]
Max k-Lin(2) ($k \geq 3$)	$z_{i_1} \oplus \cdots \oplus z_{i_k} = 0$	1/2	yes [23]

Figure 1 List of known positive and negative results for Boolean CSPs with the maximization objective. Some hardness results assume UGC and some assume only that $P \neq NP$.

exactly 3 literals. There is a trivial 7/8-approximation algorithm for Max 3E-SAT – simply choose each x_i uniformly at random from $\{0, 1\}$ (the algorithm can be easily derandomized using the method of conditional expectations). Max 3-SAT is more difficult – Karloff and Zwick showed how to get a 7/8-approximation if a certain conjecture is true [28]; then, Zwick gave a computer-assisted proof that there is indeed a 7/8-approximation algorithm for the problem [51]. Håstad showed that Max E3-SAT is approximation resistant [23] and, thus, Max E3-SAT and Max 3-SAT do not admit a better than 7/8 approximation if $P \neq NP$.

Avidor, Berkovitch, and Zwick [7] studied the general Max SAT problem, in which each constraint is a disjunction of an arbitrary number of literals. They showed how to get a 0.7968 approximation for the problem; additionally, they gave an algorithm that gets a 0.8434 approximation if a certain conjecture is true. No hardness results have been proved specifically for Max SAT; however, Håstad's 7/8 hardness for Max E3-SAT also applies to Max SAT. Finally, we note that Max Ek-SAT, the problem in which each constraint is a disjunction of exactly k literals, is considerably simpler than Max SAT. The random assignment algorithm gives a $1 - 1/2^k$ approximation; Håstad showed that there is no better approximation algorithm for the problem when $k \geq 3$ [23].

Consider an arbitrary Boolean k-CSP with the maximization objective. There ia an approximation-preserving reduction from the problem to Max k-And (the problem in which every constraint is a conjunction of k literals[1]). The algorithm by Makarychev

[1] The variant of the problem, in which each constraint is a conjunction of *at most* k literals, is equivalent to the variant, in which each constraint is a conjunction of *exactly* k literals.

problem	objective	satisfied constraints	optimal? hardness result
Max Cut Max 2-SAT Any Boolean 2-CSP	(2)	$1 - O(\sqrt{\varepsilon})$ [18, 13]	yes [30]
Max Cut Max 2-SAT Any Boolean 2-CSP	(3)	$1 - O(\varepsilon\sqrt{\log n})$ [1]	No $O(1)$ approx. [30]
Max Horn SAT	(2)	$1 - 8\log\log\frac{1}{\varepsilon}/\log\frac{1}{\varepsilon}$ [50]	$1 - \Omega(1/\log\frac{1}{\varepsilon})$ [22]
Max Horn 2-SAT	(2) and (3)	$1 - 2\varepsilon$ [22]	yes [22]

Figure 2 List of known results for almost satisfiable instances of Boolean CSPs. The table shows what fraction of the constraints we can satisfy if the optimal solution satisfies a $(1-\varepsilon)$ fraction of the constraints. Note that the minimization versions of Max Cut, Max 2-SAT, and Max Horn SAT are known as Min Uncut, Min 2-CNF Deletion, and Min Horn Deletion, respectively.

and Makarychev [40] gives a $(0.62661 - o(1))k/2^k$ approximation for Max k-And and thus for any Max Boolean 2-CSP (the little $o(1)$ term tends to 0 as $k \to \infty$). Austrin and Mossel proved a $(1 + o(1))k/2^k$ hardness of approximation if UGC [6] is true; later, Chan proved that this hardness result holds if $P \neq NP$. Note that the lower and upper bounds differ only by a constant factor; we conjecture that the algorithm in [40] actually gets a $(1 - o(1))k/2^k$ approximation. We note that the first asymptotically optimal, up to constant factors, upper and lower bounds for the problem were obtained by Samorodnitsky and Trevisan [46] and by Charikar, Makarychev, Makarychev [13], respectively. The algorithm in [13] gives a $0.44003k/2^k$ approximation for all values of k.

In Figure 1, we also summarize known results for several other Max Boolean CSPs: Max Di-Cut, Max k-All-Equal, Max k-NAE-Equal, and Max k-Lin(2).

Now, we briefly describe results for almost satisfying instances of Boolean CSPs (with objectives (2) and (3) from our list). The results are shown in Figure 2. The algorithm by Goemans and Williamson for Max Cut satisfies a $1 - O(\sqrt{\varepsilon})$ fraction of the constraints if the optimal solution satisfies a $1 - \varepsilon$ fraction of the constraints [18]. Charikar, Makarychev, and Makarychev [13] gave an algorithm for all Boolean 2-CSPs with the same approximation guarantee of $1 - O(\sqrt{\varepsilon})$. Khot, Kindler, Mossel, and O'Donnell [30] showed that these results are asymptotically optimal if UGC is true.

Agarwal, Charikar, Makarychev, and Makarychev designed an $O(\sqrt{\log n})$ approximation algorithms for Min Uncut and Min 2-CNF Deletion, the minimization versions of Max Cut and Max 2-SAT, respectively. The algorithm for Min 2-CNF Deletion gives an $O(\sqrt{\log n})$ approximation to arbitrary minimization Boolean 2-CSPs. The $(1 - O(\sqrt{\varepsilon}))$-hardness result by Khot et al. implies that there is no constant factor approximation for Min Uncut, Min 2-CNF Deletion, and, in general, Min Boolean 2-CSPs if UGC is true. Chlebík and Chlebíková proved an unconditional $8\sqrt{5} - 15 \approx 2.88854$ NP-hardness of approximation for Min 2-CNF Deletion [15]. Håstad, Huang, Manokaran, O'Donnell, and Wright [24] proved an unconditional $11/8 = 1.375$ NP-Hardness of approximation for Min Uncut.

Finally, we describe results for Max Horn SAT(Min Horn Deletion). Recall that a Horn clause is a disjunction of literals with at most one positive (non-negated) literal.[2] There is

[2] Some authors define a Horn clause as a disjunction of literals with at most one *negated* literal (see

no approximation algorithm specifically for Horn SAT with the maximization objective; the approximation algorithm by Avidor et al. for arbitrary SAT instances also gives a 0.7968 approximation for Max Horn SAT [7]. Zwick [50] designed an algorithm for $(1-\varepsilon)$ satisfiable instances of Max Horn SAT (which is also called Min Horn Deletion); the algorithm satisfies at least a $(1 - 8\log\log\frac{1}{\varepsilon}/\log\frac{1}{\varepsilon})$ fraction of the constraints. Guruswami and Zhou proved an almost matching UGC-hardness result of $(1-\Omega(1/\log\frac{1}{\varepsilon}))$ (note that the upper and lower bounds differ by a $\log\log\frac{1}{\varepsilon}$ factor) [22]. They also presented an algorithm that satisfies a $(1-2\varepsilon)$ fraction of the constraints given a $(1-\varepsilon)$-satisfiable instance of Max Horn 2-SAT and showed that this result is optimal (if UGC is true) [22].

Non-Boolean Max k-CSP. In Max k-CSP(d), an instance is a CSP with arbitrary constraints of arity k over a domain of size d. There is an approximation preserving reduction from Max k-CSP(d) to the CSP with constraints of the form $(x_{i_1} = j_1) \wedge \cdots \wedge (x_{i_k} = j_k)$. Makarychev and Makarychev designed an $\Omega(dk/d^k)$ approximation algorithm[3] for the case when $k \geq \Omega(\log d)$. Very recently, Manurangsi, Nakkiran, and Trevisan [43] gave an $\Omega(d\log d/d^k)$ approximation algorithm for $d \leq O(\log d)$ (see also [32]). Relying on the work of Austrin and Mossel [6], Håstad proved a hardness of $\Omega(kd/d^k)$ for $k \geq d$, assuming UGC [40]. Later, Chan proved that this hardness result holds if $P \neq NP$ [11]. His results also imply an $O(d^2/d^k)$ hardness of approximation for $k < d$ and an $O(\log d/\sqrt{d})$ hardness for $k = 2$. Additionally, Manurangsi et al. [43] showed an $\frac{2^{O(k\log k)}d\,(\log d)^{k/2}}{d^k}$-hardness of approximation, assuming UGC (this result gives a better upper bound on the approximation factor when $k \ll \frac{\log d}{\log\log d}$). To summarize, the best known approximation factor for the problem is $\Omega(d\max(k,\log d)/d^k)$; it is known to be optimal up to a constant factor when $k \geq d$ (if $P \neq NP$).

Other CSPs. We describe known results for Unique Games in Section 3 and results for Minimum Multiway Cut in Section 5.

Universal Algorithm for CSPs. In Section 6, we discuss two very important general results on approximability of CSPs: the result by Raghavendra [44] that shows that semidefinite programming gives the best possible approximation for many CSPs and the universal approximation algorithm for CSPs by Raghavendra and Steurer [45].

1.2 Organization

In Section 2, we describe the algorithm by Goemans–Williamson for Max Cut and discuss algorithms for other Boolean 2-CSPs. In Section 3, we describe known results for Unique Games and give an approximation algorithm for the problem, as well as present the framework of orthogonal separators. Then in Section 4, we discuss techniques for solving CSPs of arities $k > 2$. In Section 5, we discuss the Minimum Multiway Cut problem and present the algorithm by Călinescu, Karloff, and Rabani [10]. This is the only algorithm based on linear programming that we give in this survey; all other algorithms are based on semidefinite programming. Finally, in Section 6, we discuss the results by Raghavendra [44]

e.g. [50]). The two definitions are different; however, by negating all literals, an instance of one problem can be transformed to an instance of the other problem of the same value.

[3] We write the approximation factor as dk/d^k and not as k/d^{k-1}, because it is easier to compare it to the approximation factor $1/d^k$ of the random assignment algorithm, when it is written in this form.

and Raghavendra and Steurer [45] and describe and analyze the universal rounding algorithm for (generalized) 2-CSPs with nonnegative predicates [45]. We conclude the paper with a list of open problems.

2 Boolean CSPs or Arity 2: Max Cut and Max 2-SAT

In this section, we describe the Goemans–Williamson approximation algorithm [18] for Max Cut and discuss approximation algorithms for other Boolean 2-CSPs.

Max Cut problem may be stated as a graph partitioning or constraint satisfaction problem. In the graph partitioning formulation, we are given a graph $G = (V, E)$, and our goal is to find a cut $(S, \bar{S})$ so as to maximize the number of cut edges. In the CSP formulation, we are given a set variables $x_1, \dots, x_n$ and a set of constraints of the form $x_i \neq x_j$; our goal is to find a Boolean assignment that maximizes the number of satisfied constraints. There is a simple correspondence between the two formulations: vertices correspond to variables, edges correspond to the constraints, and a solution $(S, \bar{S})$ corresponds to a CSP solution that assigns 1 (true) to vertices in S and 0 (false) to vertices in $\bar{S}$. Below, we consider the CSP formulation of the problem. We note that all results we describe in this section also apply to a more general Max 2-Lin(2) problem.

▶ **Theorem 1** ([18]). *There exists a randomized polynomial-time approximation algorithm for Max Cut that finds a solution of value at least $\alpha_{GW}\mathsf{OPT}$, in expectation, where $\alpha_{GW} \approx 0.87856$.*

Proof. We write the following standard SDP relaxation for Max Cut.

$$\text{maximize } \frac{1}{4} \sum_{\text{constraint } x_i \neq x_j} \|\bar{u}_i - \bar{u}_j\|^2$$

$$\text{subject to}$$

$$\|\bar{u}_i\|^2 = 1 \qquad \text{for every } i \in \{1, \dots, n\}.$$

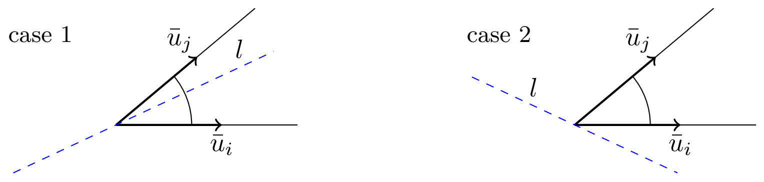
There is an SDP vector variable $\bar{u}_i$ for each CSP variable x_i. The only constraint in the SDP requires that all vectors $\bar{u}_i$ be unit vectors.

Let us verify this is indeed a relaxation; that is, the optimal value of the SDP is at most OPT. To this end, consider an optimal solution $x_i = \hat{x}_i$. Now, pick an arbitrary unit vector $\bar{v}_0$, and define a feasible SDP solution $\bar{u}_i = \bar{v}_0$ if $x_i = 1$, and $\bar{u}_i = -\bar{v}_0$ if $x_i = 0$. Observe that $\frac{1}{4}\|\bar{u}_i - \bar{u}_j\|^2 = 1$ if $x_i \neq x_j$, and $\frac{1}{4}\|\bar{u}_i - \bar{u}_j\|^2 = 0$, otherwise. Thus, the value of this SDP solution equals OPT. Therefore, the value of the optimal SDP solution is at most OPT. We denote the value of the SDP solution by SDP.

The Goemans–Williamson approximation algorithm solves the SDP relaxation and finds an optimal solution $\{\bar{u}_i\}$. Now, it chooses a random hyperplane H passing through the origin. The hyperplane partitions space into two half-spaces A and $\bar{A}$ (we arbitrarily choose which of the half-spaces is A and which is $\bar{A}$). The algorithm returns the following solution:

$$x_i = \begin{cases} 1, & \text{if } \bar{u}_i \in A, \\ 0, & \text{if } \bar{u}_i \in \bar{A}. \end{cases}$$

Let us analyze this algorithm. Consider a constraint $x_i \neq x_j$. We lower bound the probability that it is satisfied. Consider the two dimensional plane P passing through $\bar{u}_i$ and $\bar{u}_j$. Note that the intersection between P and the random hyperplane H is a random line l in P, passing through the origin. Consider line l and the angle formed by vectors $\bar{u}_i$ and $\bar{u}_j$; note that l goes through the vertex of the angle.



There are two cases: (1) P separates the sides of the angle and (2) it does not. In the former case, $\bar{u}_i$ and $\bar{u}_j$ lie in different half-spaces w. r. t. H; that is, one of them is in A and the other is in $\bar{A}$. The algorithm assigns different values to x_i and x_j, and thus satisfies the constraint $x_i \neq x_j$. In the latter case, $\bar{u}_i$ and $\bar{u}_j$ lie in the same half-space w. r. t. H; the algorithm assigns the same value to x_i and x_j, and thus violates the constraint $x_i \neq x_j$. We conclude that the probability that the constraint $x_i \neq x_j$ is satisfied equals the probability that l goes between $\bar{u}_i$ and $\bar{u}_j$. This probability equals the angle between $\bar{u}_i$ and $\bar{u}_j$ divided by π: $\arccos\langle \bar{u}_i, \bar{u}_j\rangle/\pi$. We compare this probability with the SDP contribution of the constraint $x_i \neq x_j$: $\frac{1}{4}\|\bar{u}_i - \bar{u}_j\|^2 = \frac{1-\langle \bar{u}_i,\bar{u}_j\rangle}{2}$.

$$\Pr(x_i \neq x_j) \Big/ \frac{1-\langle \bar{u}_i,\bar{u}_j\rangle}{2} = \frac{2}{\pi}\frac{\arccos\langle \bar{u}_i,\bar{u}_j\rangle}{1-\langle \bar{u}_i,\bar{u}_j\rangle} \geq \min_{x\in[-1,1]} \frac{2}{\pi}\frac{\arccos x}{1-x} \equiv \alpha_{GW} \geq 0.87856.$$

Here, α_{GW} is the minimum of the function $\frac{2}{\pi}\frac{\arccos x}{1-x}$ on $[-1,1]$. Numerically, it is greater than and approximately equal to 0.87856.

We conclude that, in expectation, the algorithm satisfies at least

$$\sum_{\text{constraint } x_i\neq x_j} \Pr(x_i \neq x_j) \geq \frac{\alpha_{GW}}{4} \sum_{\text{constraint } x_i\neq x_j} \|\bar{u}_i - \bar{u}_j\|^2 = \alpha_{GW}\,\mathsf{SDP} \geq \alpha_{GW}\,\mathsf{OPT}$$

constraints, as required. Note that if we run the algorithm sufficiently many times and output the best of the solutions we find, we get at least an $(\alpha_{GW} - \delta)$ approximation with high probability (for a polynomially small δ). ◀

▶ **Theorem 2** (Goemans and Williamson [18]). *Given a $(1-\varepsilon)$-satisfiable instance of Max Cut, the Goemans–Williamson algorithm finds a solution of value at least $1 - O(\sqrt{\varepsilon})$, in expectation.*

Proof. Let m be the number of the constraints. As we showed in the proof of Theorem 1, in expectation, the fraction of the constraints satisfied by the Goemans–Williamson algorithm is

$$\frac{1}{m}\sum_{\text{constraint } x_i\neq x_j} \Pr(x_i \neq x_j) = \frac{1}{m}\sum_{\text{constraint } x_i\neq x_j} \frac{\arccos\langle \bar{u}_i,\bar{u}_j\rangle}{\pi}.$$

Note that $\cos x \geq 1 - x^2/2$. Since $\cos x$ is decreasing on $[0,\pi]$, we have $x \leq \arccos(1-x^2/2)$; letting $y = x^2/2 - 1$, we get $\arccos y = \pi - \arccos(-y) \leq \pi - \sqrt{2(y+1)}$. Thus, $\frac{\arccos\langle \bar{u}_i,\bar{u}_j\rangle}{\pi} \geq 1 - \frac{\sqrt{2}\sqrt{1+\langle \bar{u}_i,\bar{u}_j\rangle}}{\pi}$. Applying Jensen's inequality and using that

$$\frac{1}{m}\sum_{\text{constr. } x_i\neq x_j} \frac{1-\langle \bar{u}_i,\bar{u}_j\rangle}{2} = \frac{\mathsf{SDP}}{m} \geq \frac{\mathsf{OPT}}{m} \geq 1-\varepsilon,$$

we get that the fraction of satisfied constraints is at least

$$\frac{1}{m}\sum_{\text{constraint } x_i\neq x_j}\left(1-\frac{\sqrt{2}\sqrt{1+\langle\bar{u}_i,\bar{u}_j\rangle}}{\pi}\right)\geq 1-\frac{\sqrt{2}}{\pi}\sqrt{1+\frac{1}{m}\sum_{\text{constr. } x_i\neq x_j}\langle\bar{u}_i,\bar{u}_j\rangle}\geq 1-\frac{2\sqrt{\varepsilon}}{\pi},$$

in expectation. Running this rounding procedure many times, we can find a solution satisfying a $1-\frac{2(1-\delta)\sqrt{\varepsilon}}{\pi}$ fraction of the constraints with high probability. ◀

Approximation algorithms for other Boolean 2-CSPs are more complex. Consider the Max 2-SAT problem, in which constraints are of the form $z_i \vee z_j$ (where z_i and z_j are literals). Notationally, it is convenient to introduce variables $x_{-1},\dots,x_{-n}$ for the negated literals $\bar{x}_1,\dots,\bar{x}_n$; that is, let $x_{-i}=\bar{x}_i$. Then, every constraint can be written as $x_i\vee x_j$, where $i,j\in\{\pm1,\dots,\pm n\}$. We write an SDP relaxation for the problem. We have SDP variables $\bar{u}_{\pm1},\dots,\bar{u}_{\pm n}$ for CSP variables $x_{\pm1},\dots,x_{\pm n}$. We require that $\bar{u}_i=-\bar{u}_{-i}$.

$$\text{maximize } \frac{1}{4}\sum_{\text{constraint } x_i\vee x_j}(3+\langle\bar{u}_i+\bar{u}_j,\bar{v}_0\rangle-\langle\bar{u}_i,\bar{u}_j\rangle)$$

subject to

$$\frac{3+\langle\bar{u}_i+\bar{u}_j,\bar{v}_0\rangle-\langle\bar{u}_i,\bar{u}_j\rangle}{4}\leq 1 \qquad \text{for every } i\in\{\pm1,\dots,\pm n\}$$

$$\bar{u}_i=-\bar{u}_{-i} \qquad \text{for every } i\in\{\pm1,\dots,\pm n\}$$

$$\|\bar{u}_i\|^2=\|v_0\|^2=1 \qquad \text{for every } i\in\{1,\dots,\pm n\}$$

Note that we need an extra variable $\bar{v}_0$ in the relaxation for Max 2-SAT (which was absent in the relaxation for Max Cut). Variable $\bar{v}_0$ represents true; accordingly, $-\bar{v}_0$ represents false. In the intended SDP solution corresponding to a CSP solution, $\bar{v}_0$ is an arbitrary unit vector, and $\bar{u}_i=\bar{v}_0$ if $x_i=1$, $\bar{u}_i=-\bar{v}_0$ if $x_i=0$. Note that the assignments to $\bar{u}_i$ and $\bar{u}_{-i}$ are consistent: one of them is equal to $\bar{v}_0$ and the other to $-\bar{v}_0$.

Let us informally discuss what properties a rounding procedure for Max 2-SAT should satisfy. First, note that if $\bar{u}_i$ is close to $\bar{v}_0$, then the SDP contribution of every constraint of the form $x_i\vee x_j$ (for every j) is close to 1 (in particular, if $\bar{u}_i=\bar{v}_0$, then the contribution is 1). Hence, we want to round $\bar{u}_i$ to 1 with high probability. Similarly, if $\bar{u}_i$ is close to $-\bar{v}_0$, then the SDP contribution of every constraint of the form $x_{-i}\vee x_j$ is close to 1; hence, we want to round $\bar{u}_i$ to 0 with high probability. We get the following heuristic:

Heuristic Rule 1: If $|\langle\bar{u}_i,\bar{v}_0\rangle|$ is "large", use threshold rounding. Namely, round $\bar{u}_i$ to 1 if $\langle\bar{u}_i,\bar{v}_0\rangle>0$; round $\bar{u}_i$ to 0 if $\langle\bar{u}_i,\bar{v}_0\rangle<0$.

On the other hand, if $|\langle\bar{u}_i,\bar{v}_0\rangle|$ is 0 or small (then $\bar{u}_i$ is far from both $\bar{v}_0$ and $-\bar{v}_0$), we get no or little information from $\langle\bar{u}_i,\bar{v}_0\rangle$ whether to round $\bar{u}_i$ to 1 or 0. Note that the set of vectors $S=\{\bar{u}:\langle\bar{u},\bar{v}_0\rangle=0\}$ is a sphere, which has no distinguished direction. Hence, it is natural to use the Goemans–Williamson rounding procedure for vectors in S.

Heuristic Rule 2: If $\langle\bar{u}_i,\bar{v}_0\rangle$ is "small', use the Goemans–Williamson algorithm.

Now we need to combine these two heuristics and get a rounding procedure that works for all vectors, including vectors $\bar{u}_i$ for which $|\langle\bar{u}_i,\bar{v}_0\rangle|$ is neither "small" not "large". Lewin, Livnat, and Zwick [36] use a clever combination of an "outward rotation" and "skewed" rounding to achieve that in their 0.94016-approximation algorithm for Max 2-SAT. We describe a somewhat simpler algorithm by Charikar et al. [13] that satisfies a $(1 - O(\sqrt{\varepsilon}))$ fraction of the constraint given a $(1 - \varepsilon)$ satisfiable instance. The algorithm solves the SDP relaxation and finds vectors $\bar{u}_i$. Let $\varepsilon' = 1 - \mathsf{SDP}/m$ (note that $\varepsilon' \leq \varepsilon$ since $\mathsf{SDP} \geq \mathsf{OPT}$). The algorithm chooses a random Gaussian vector g with independent components distributed as $\mathcal{N}(0, 1)$. For every i, it lets

$$x_i = \begin{cases} 1, & \text{if } \langle \bar{u}_i, \bar{v}_0 + \sqrt{\varepsilon'}\, g\rangle > 0, \\ 0, & \text{if } \langle \bar{u}_i, \bar{v}_0 + \sqrt{\varepsilon'}\, g\rangle < 0. \end{cases}$$

Note that the algorithm always assigns opposite values to x_i and x_{-i}, since $\langle \bar{u}_i, \bar{v}_0 + \sqrt{\varepsilon'}\, g\rangle = -\langle \bar{u}_{-i}, \bar{v}_0 + \sqrt{\varepsilon'}\, g\rangle$. If $\bar{u}_i$ is close to $\bar{v}_0$ or $-\bar{v}_0$, then $\langle \bar{u}_i, \bar{v}_0\rangle$ is larger in absolute value than $\langle \bar{u}_i, \sqrt{\varepsilon'}\, g\rangle$ with high probability; thus, we essentially use threshold rounding (Heuristic Rule 1); however, if $\langle \bar{u}_i, \bar{v}_0\rangle = 0$, then the algorithm rounds u_i depending on the sign of $\langle \bar{u}_i, g\rangle$; that is, it uses the Goemans–Williamson rounding (namely, all vectors in the half-space $\{u : \langle u, g\rangle > 0\}$ are rounded to 1; vectors in the half-space $\{u : \langle u, g\rangle < 0\}$ are rounded to 0).

To analyze this algorithm, Charikar et al. upper bound the probability that each constraint $x_i \vee x_j$ is not satisfied. Let the SDP contribution of the constraint $x_i \vee x_j$ be $1 - \varepsilon'_{ij}$; then, the average of all ε'_{ij} is ε'. It is proved in [13] that the probability that $x_i \vee x_j$ is violated is $O(\sqrt{\varepsilon'} + \varepsilon'_{ij}/\sqrt{\varepsilon} + \sqrt{\varepsilon'})$. Averaging over all constraints and using Jensen's inequality, we get that the expected fraction of violated constraints is $O(\sqrt{\varepsilon'}) = O(\sqrt{\varepsilon})$, as required.

Interestingly, this algorithm differs from many other algorithms in that it may violate a constraint $x_i \vee x_j$ even if the SDP contribution of the constraint is equal to 1 (then, $\varepsilon'_{ij} = 0$); loosely speaking, the algorithm violates the constraint even if the SDP "thinks" that the constraint is "certainly" satisfied. In contrast, the Goemans–Williamson algorithm always satisfies a constraint $x_i \neq x_j$ if its contribution $\frac{1}{4}\|\bar{u}_i - \bar{u}_j\|^2$ is 1 (then, vectors $\bar{u}_i$ and $\bar{u}_j$ are antipodal). It turns out that this difference is not accidental: if an algorithm never violates constraints whose SDP contribution is 1, then it can solve instances with hard constraints; however, Guruswami and Lee showed that no polynomial-time algorithm for Max 2-SAT with hard constraints can distinguish between $(1 - \varepsilon)$-satisfiable and at-most-ε-satisfiable instances (if UGC is true) [21].

3 Unique Games

In this section, we define the Unique Games problem, overview known results, and describe an algorithm for Unique Games.

▶ **Definition 3** (Unique Games)**.** Unique Games is a constraint satisfaction problem of arity 2, in which every constraint has the form $x_j = \pi_{ij}(x_i)$ for some permutation π_{ij} of the domain D.

Observe that for every fixed value of the variable x_i, there is a *unique* value for the variable x_j that satisfies the constraint between x_i and x_j. Hence, it is easy to find an exact solution for a completely satisfiable instance of Unique Games: We simply guess the value of one variable and then prorogate the values to all other variables (we do this for each connected component of the constraint graphs). However, this algorithm fails even if 1% of all the constraints are violated in the optimal solution. Khot [29] conjectured that if the optimal solution satisfies a $(1 - \varepsilon)$ fraction of the constraints, then it is NP-hard to find a solution satisfying even a δ fraction of the constraints. The conjecture is known as Khot's Unique Games Conjecture. We state it formally below.

▶ **Definition 4** (Unique Games Conjecture (UGC) [29]). For every positive ε and δ, there exists a d such that given an instance of Unique Games on a domain of size d, it is NP-hard to distinguish between the following two cases:

- There exists a solution satisfying a $(1-\varepsilon)$ fraction of all the constraints.
- Every assignment satisfies at most a δ fraction of all the constraints.

It is unknown whether the conjecture is true or false. However, UGC has proved to be very useful in obtaining hardness of approximation results. Researchers showed very strong hardness of approximation results that rely on UGC for such problems as Vertex Cover [31], Max Cut and Max 2-Lin(q) [30], ordering CSPs [20], and general MAX CSPs [44] (we discuss the last result in Section 6). Today, we do not know how to obtain similar results under weaker complexity assumptions. That is why UGC has gained a lot of popularity in approximation algorithms and hardness of approximation communities. By now, the question whether the conjecture is true of false is one of the major open questions in theoretical computer science.

There are several general purpose approximation algorithms for the problem [29, 48, 19, 12, 14]. The best approximation algorithms by Charikar, Makarychev, and Makarychev [12] and by Chlamtac, Makarychev, and Makarychev [14] find solutions satisfying a $1-O(\sqrt{\varepsilon\log k})$ fraction and $1-O(\varepsilon\sqrt{\log n\log k})$) fraction of all the constraints (respectively) given a $(1-\varepsilon)$-satisfiable instance. Khot, Kindler, Mossel, and O'Donnell [30] showed that there is no polynomial-time algorithm that satisfies more than a $1-c\sqrt{\varepsilon\log k}$ fraction of all the constraints if UGC is true. Thus, the algorithm [12] cannot be improved for general instances of Unique Games if UGC is true. However, there are known better algorithms for special families of Unique Games. Arora et al. [4] showed that UGC does not hold for instances of Unique Games whose constraint graphs are expanders (see also [39]). Kolla, Makarychev, and Makarychev [33] showed that the Unique Games Conjecture does not hold for random and semi-random instances of Unique Games. Finally, Arora, Barak, and Steurer [2] designed a sub-exponential (super-polynomial) algorithm for arbitrary instances of Unique Games. Given a $(1-\varepsilon)$-satisfiable instance, their algorithm finds a solution satisfying a $(1-\varepsilon^c)$ fraction of all the constraints (for some fixed $c>0$) in time $\exp(dn^{\varepsilon^c})$.

We now present an SDP-based approximation algorithm for Unique Games by Charikar, Makarychev, and Makarychev [12]. The exposition of the algorithm follows the paper by Chlamtac, Makarychev, and Makarychev [14] (see also [8, 37]).

▶ **Theorem 5** (Charikar, Makarychev, Makarychev [12]). *There exists an approximation algorithm that given a $(1-\varepsilon)$-satisfiable instance of Unique Games, finds a solution satisfying a $1-O(\sqrt{\varepsilon\log d})$ fraction of all the constraints.*

The approximation of Theorem 5 cannot be improved if the Unique Games conjecture is true [30]. We prove Theorem 5 using the technique of orthogonal separators [14].

In the next section, we present a standard SDP relaxation for Unique Games (without ℓ_2^2 triangle inequalities). Then, in Section 3.2, we introduce a technique of orthogonal separators. However, we postpone the proof of existence of orthogonal separators to Section 3.4. In Section 3.3, we present the approximation algorithm and prove Theorem 5. Finally, in Section 3.5, we give some useful bounds on the Gaussian distribution.

3.1 SDP Relaxation

We use a standard SDP relaxation for 2CSPs over non-Boolean domain D. We let $G=(V,E)$ be the constraint graph: The vertices of the graph correspond to the variables x_i, the edges correspond to the constraints $x_j = \pi_{ij}(x_i)$. Formally, $V = \{1, \dots, n\}$, $E = \{(i,j) :$ there is a constraint $x_j = \pi_{ij}(x_i)\}$. To simplify the notation, we assume that the graph does not have parallel edges; i.e. there is at most one constraint between every pair of variables x_i and x_j. Note that this restriction on instances can easily be removed. In the SDP relaxation, we have a vector $\bar{u}_{ia}$ for every vertex $i \in V$ and label $a \in D$. In the *intended integral* solution, the vector $\bar{u}_{ia}$ is the indicator of the event *"$x_i = a$"*. That is, if x_i^* is the optimal solution, then the corresponding integral solution is as follows:

$$\bar{u}_{ia}^* = \begin{cases} 1, & \text{if } x_i^* = a; \\ 0, & \text{otherwise.} \end{cases}$$

Observe that if a constraint (i,j) is satisfied then $\bar{u}^*_{j\pi_{ij}(a)} = \bar{u}^*_{ia}$ for all $a \in D$. If the constraint is violated, then $\bar{u}^*_{ia} = \bar{u}^*_{j\pi_{ij}(a)} = 0$ for all but exactly two a's: $\bar{u}^*_{ix_i} = 1$, but $\bar{u}^*_{j\pi_{ij}(x_i)} = 0$; and $\bar{u}^*_{jx_j} = 1$, but $\bar{u}^*_{i\pi_{ij}^{-1}(x_j)} = 0$. Thus,

$$\frac{1}{2}\sum_{a \in D} \|\bar{u}^*_{ia} - \bar{u}^*_{j\pi_{ij}(a)}\|^2 = \begin{cases} 0, & \text{if assignment } x^* \text{ satisfies constraint } (i,j); \\ 1, & \text{if assignment } x^* \text{ violates constraint } (i,j). \end{cases}$$

Therefore, the number of violated constraints equals

$$\frac{1}{2}\sum_{(i,j)\in E}\sum_{a \in D} \|\bar{u}^*_{ia} - \bar{u}^*_{j\pi_{ij}(a)}\|^2. \tag{2}$$

Our goal is to minimize this expression. Note that for a fixed variable x_i, one and only one $\bar{u}^*_{ia}$ equals 1. Hence, (1) $\langle \bar{u}^*_{ia}, \bar{u}^*_{ib}\rangle = 0$, if $a \neq b$; and (2) $\sum_{a\in D}\|\bar{u}^*_{ia}\|^2 = 1$. We now write the SDP relaxation (in this SDP, unlike many SDPs we consider in this survey, the objective measures the number of unsatisfied constraints).

$$\text{minimize} \qquad \frac{1}{2}\sum_{(u,v)\in E}\sum_{a\in D}\|\bar{u}_{ia} - \bar{u}_{j\pi_{uv}(a)}\|^2$$

subject to

$$\langle \bar{u}_{ia}, \bar{u}_{jb}\rangle = 0 \qquad \text{for all } i \in V \text{ and } a \neq b$$

$$\sum_{a\in D}\|\bar{u}_{ia}\|^2 = 1 \qquad \text{for all } i \in V$$

This is a relaxation, since for $\bar{u}_{ia} = \bar{u}^*_{ia}$, the SDP value equals the number of violated constraints (see (2)); and $\bar{u}^*_{ia}$ is a feasible solution for the SDP.

3.2 Orthogonal Separators – Overview

The main technical tool of the algorithm is a procedure for sampling a random subset of vectors, called an orthogonal separator, from the set of all vectors $\bar{u}_{ia}$ in the SDP solution. The distribution of the subset must satisfy certain properties which we describe in this section. Orthogonal separators are used not only in algorithms for Unique Games, but also in various graph partitioning algorithms [3, 8, 34, 37, 38, 41]. In fact, one can think of Unique Games as of certain graph partitioning problem.

Let X be a set of vectors in ℓ_2 of length at most 1. We say that a distribution over subsets of X is an m-orthogonal separator of X with ℓ_2 distortion $\mathcal{D}$, probability scale $\alpha > 0$ and separation threshold $\beta < 1$, if the following conditions hold for $S \subset X$ chosen randomly according to this distribution:

1. For all $\bar{u} \in X$, $\Pr(\bar{u} \in S) = \alpha\|\bar{u}\|^2$.
2. For all $\bar{u}, \bar{v} \in X$ with $\langle \bar{u}, \bar{v}\rangle \leq \beta \max(\|\bar{u}\|^2, \|\bar{v}\|^2)$,

$$\Pr(\bar{u} \in S \text{ and } \bar{v} \in S) \leq \frac{\alpha \min(\|\bar{u}\|^2, \|\bar{v}\|^2)}{m}.$$

3. For all $\bar{u}, \bar{v} \in X$,

$$\Pr(I_S(\bar{u}) \neq I_S(\bar{v})) \leq \alpha\mathcal{D}\,\|\bar{u} - \bar{v}\| \cdot \min(\|\bar{u}\|, \|\bar{v}\|) + \alpha\big|\|\bar{u}\|^2 - \|\bar{v}\|^2\big|,$$

where I_S is the indicator of the set S; i.e. $I_S(\bar{u}) = 1$, if $\bar{u} \in S$; $I_S(\bar{u}) = 0$, if $\bar{u} \notin S$.

In most cases, it is convenient to use a slightly weaker (but simpler) bound on $\Pr(I_S(\bar{u}) \neq I_S(\bar{v}))$.

3'. For all $\bar{u}, \bar{v} \in X$,

$$\Pr(I_S(\bar{u}) \neq I_S(\bar{v})) \leq \alpha\mathcal{D}'\|\bar{u} - \bar{v}\| \cdot \max(\|\bar{u}\|, \|\bar{v}\|).$$

The property ($3'$) follows from (3) with $\mathcal{D}' = \mathcal{D} + 2$, since

$$\big|\|\bar{u}\|^2 - \|\bar{v}\|^2\big| = \big|\|\bar{u}\| - \|\bar{v}\|\big| \cdot (\|\bar{u}\| + \|\bar{v}\|) \leq \|\bar{u} - \bar{v}\| \cdot 2\max(\|\bar{u}\|, \|\bar{v}\|).$$

The last inequality follows from the (regular) triangle inequality for vectors $\bar{u}$, $\bar{v}$ and $(\bar{u} - \bar{v})$.

Our algorithm for Unique Games relies on the following theorem.

▶ **Theorem 6** (see Chlamtac, Makarychev, Makarychev [14]). *There exists a polynomial-time randomized algorithm that given a set of vectors X in the unit ball and parameter m, generates an m-orthogonal separator with ℓ_2 distortion $\mathcal{D} = O\left(\sqrt{\log m}\right)$, probability scale $\alpha \geq poly(1/m)$ and separation threshold $\beta = 0$.*

▶ Remark. Chlamtac, Makarychev, Makarychev [14] proved that there exists an ℓ_2^2 orthogonal separator satisfying conditions (1), (2), and ($3''$):

3". For all $\bar{u}, \bar{v} \in X$, $\Pr(I_S(\bar{u}) \neq I_S(\bar{v})) \leq \alpha\tilde{\mathcal{D}}\|\bar{u} - \bar{v}\|^2$, where $\tilde{\mathcal{D}} = O(\sqrt{\log n \log k})$.

If we use this type of orthogonal separators in the algorithm that we present in the next section, we will get an approximation algorithm that satisfies a $1 - O(\varepsilon\sqrt{\log n \log k})$ fraction of the constraints given a $1 - \varepsilon$ satisfiable instance.

3.3 Approximation Algorithm

We now present an approximation algorithm for Unique Games that uses orthogonal separators. We prove Theorem 6 and show how to generate orthogonal separators in Section 3.4. Consider the algorithm presented in Figure 3.

▶ **Lemma 7.** *The algorithm satisfies the constraint between variables i and j with probability $1 - O(\mathcal{D}\sqrt{\varepsilon_{ij}})$, where $\mathcal{D}$ is the distortion of the orthogonal separator sampled by the algorithm, and ε_{ij} is the SDP contribution of the term corresponding to the edge (i, j):*

$$\varepsilon_{ij} = \frac{1}{2}\sum_{a \in D} \|\bar{u}_{ia} - \bar{u}_{j\pi_{ij}(a)}\|^2.$$

```
Input: An instance of Unique Games.
Output: Assignment of labels to vertices.
1. Solve the SDP. Let $X = \{\bar{u}_{ia} : i \in V, a \in D\}$.
2. Mark all variables as active.
3. while (there are active variables)
   a. Produce an $m$-orthogonal separator $S \subset X$ with distortion $\mathcal{D}$ and probability scale $\alpha$
      as in Theorem 6, where $m = 4k$ and $\mathcal{D} = O(\sqrt{\log m})$.
   b. For all active variables $x_i$:
      - Let $S_i = \{a : \bar{u}_{ia} \in S\}$.
      - If $S_i$ contains exactly one element $a$, then let $x_i = a$; mark the variable $x_i$ as
        inactive.
4. If the algorithm performs more than $n/\alpha$ iterations, assign arbitrary values to any
   remaining variables (note that $\alpha \geq 1/poly(d)$).
```

Figure 3 Approximation algorithm for Unique Games.

Proof. If $(\mathcal{D}+2)\sqrt{\varepsilon_{ij}} \geq 1/8$, then the statement holds trivially, so we assume that $(\mathcal{D}+2)\sqrt{\varepsilon_{ij}} < 1/8$. For the sake of analysis, we also assume that π_{ij} is the identity permutation (we can simply rename the values of the variable x_j so that π_{ij} is the identity; this clearly does not affect the execution of the algorithm).

Consider the first iteration in which we assign a value to one of the variables, x_i or x_j. At the end of this iteration, we mark the constraint $x_i = \pi_{ij}(x_j)$ as satisfied or not: If we assign the same value $a \in D$ to x_i and x_j, we mark the constraint as satisfied (recall that we assume that π_{ij} is the identity permutation); otherwise, we conservatively mark the constraint as violated (a constraint marked as violated in the analysis may potentially be satisfied by the algorithm). Consider one iteration of the algorithm at which both x_i and x_j are active. There are three possible cases:

1. Both sets S_i and S_j are equal and contain exactly one element, then the constraint $x_j = \pi_{ij}(x_i)$ is satisfied after this iteration.
2. The sets S_i and S_j are equal, but contain more than one or none elements, then no values are assigned at this iteration to x_i and x_j.
3. The sets S_i and S_j are not equal, then the constraint may be violated.

Let us estimate the probabilities of each of these events. Using that for all $a \neq b$ the vectors $\bar{u}_{ia}$ and $\bar{u}_{ib}$ are orthogonal, and properties 1 and 2 of orthogonal separators we get (below α is the probability scale): for a fixed a,

$$\begin{aligned}
\Pr(|S_i| = 1;\ a \in S_i) &= \Pr(a \in S_u) - \Pr(a \in S_u \text{ and } b \in S_u \text{ for some } b \neq a)\\
&\geq \Pr(a \in S_u) - \sum_{b \in D\setminus\{a\}} \Pr(a, b \in S_u)\\
&\geq \alpha\|\bar{u}_{ia}\|^2 - \sum_{b \in D\setminus\{a\}} \frac{\alpha \min(\|\bar{u}_{ia}\|^2, \|\bar{u}_{ib}\|^2)}{4d}\\
&\geq \alpha\|\bar{u}_{ia}\|^2 - \frac{\alpha}{4d}\sum_{b \in D\setminus\{a\}} \|\bar{u}_{ia}\|^2\\
&\geq \alpha\|\bar{u}_{ia}\|^2\Big(1 - \frac{(d-1)}{4d}\Big) \geq \frac{3\alpha\|u_{ia}\|^2}{4}.
\end{aligned}$$

Then, using that $\sum_{a\in D}\|\bar{u}_{ia}\|^2 = 1$, we get

$$\Pr\left(|S_i| = 1\right) = \sum_{a\in D}\Pr\left(|S_i| = 1;\ a\in S_i\right) \geq \sum_{a\in D}\frac{3\alpha\|u_{ia}\|^2}{4} = \frac{3\alpha}{4}. \tag{3}$$

Thus, at every iteration of the algorithm when x_i is active, we assign a value to x_i with probability at least $3\alpha/4$. The probability that the constraint (i,j) is violated is at most

$$\Pr\left(S_i \neq S_j\right) \leq \sum_{a\in D}\Pr\left(I_S(\bar{u}_{ia}) \neq I_S(\bar{u}_{ja})\right).$$

We use property 3 of orthogonal separators (see property (3′)) to upper bound the right hand side

$$\Pr\left(S_i \neq S_j\right) \leq \alpha\mathcal{D}' \sum_{a\in D}\|\bar{u}_{ia} - \bar{u}_{ja}\| \cdot \max(\|\bar{u}_{ia}\|, \|\bar{u}_{ja}\|).$$

By Cauchy–Schwarz,

$$\begin{aligned}\Pr\left(S_u \neq S_v\right) &\leq \alpha\mathcal{D}'\sqrt{\sum_{a\in D}\|\bar{u}_{ia} - \bar{u}_{ja}\|^2} \cdot \sqrt{\sum_{a\in D}\max(\|\bar{u}_{ia}\|^2, \|\bar{u}_{ja}\|^2)}\\ &\leq \alpha\mathcal{D}'\sqrt{\sum_{i\in D}2\varepsilon_{ij}} \cdot \underbrace{\sqrt{\sum_{a\in D}\|\bar{u}_{ia}\|^2 + \|\bar{u}_{ja}\|^2}}_{=\sqrt{2}}\\ &= 2\alpha\mathcal{D}'\sqrt{\varepsilon_{ij}}.\end{aligned}$$

Finally, the probability of satisfying the constraint is at least

$$\Pr\left(|S_u| = 1 \text{ and } S_u = S_v\right) \geq \frac{3}{4}\alpha - 2\alpha\mathcal{D}'\sqrt{\varepsilon_{ij}} \geq \frac{1}{2}\alpha.$$

Here, we used the assumption $\mathcal{D}'\sqrt{\varepsilon_{ij}} \leq 1/8$. Since the algorithm performs n/α iterations, the probability that it does not assign any value to x_i or x_j before step 4 is exponentially small. At each iteration the probability of failure is at most $O(\mathcal{D}\sqrt{\varepsilon_{ij}})$ times the probability of success, thus the probability that the constraint is not satisfied is $O(\mathcal{D}\sqrt{\varepsilon_{ij}})$. ◀

We now show that the approximation algorithm satisfies a $1 - O(\sqrt{\varepsilon\log d})$ fraction of all the constraints.

Proof of Theorem 5. By Lemma 7, the expected number of unsatisfied constraints is equal to

$$\sum_{(u,v)\in E} O(\sqrt{\varepsilon_{ij}\log d}).$$

By Jensen's inequality for the function $t \mapsto \sqrt{t}$,

$$\frac{1}{|E|}\sum_{(u,v)\in E}\sqrt{\varepsilon_{ij}\log d} \leq \sqrt{\frac{1}{|E|}\sum_{(i,j)\in E}\varepsilon_{ij}\log d} = \sqrt{\frac{\mathsf{SDP}}{|E|}\log d}.$$

Here, $\mathsf{SDP} = \sum_{(i,j)\in E}\varepsilon_{ij}$ denotes the SDP value. If $\mathsf{OPT} \leq \varepsilon|E|$, then $\mathsf{SDP} \leq \mathsf{OPT} \leq \varepsilon|E|$. Hence, the expected cost of solution is upper bounded by $O(\sqrt{\varepsilon\log k})|E|$. ◀

3.4 Orthogonal Separators – Proofs

Proof of Theorem 6. In the proof, we denote the probability that a Gaussian $\mathcal{N}(0,1)$ random variable X is greater than a threshold t by $\bar{\Phi}(t)$. We use the following algorithm for generating m-orthogonal separators with ℓ_2 distortion: Assume w.l.o.g. that all vectors $\bar{u}$ lie in $\mathbb{R}^n$. Fix a threshold $t = \bar{\Phi}^{-1}(1/m)$ (i.e., fix t such that $\bar{\Phi}(t) = 1/m$). Sample independently a random Gaussian n dimensional vector $g \sim \mathcal{N}(0, I)$ in $\mathbb{R}^n$ and a random number r in $[0,1]$. Return the set

$$S = \{\bar{u} : \langle \bar{u}, g\rangle \geq t\|\bar{u}\| \text{ and } \|\bar{u}\|^2 \geq r\}.$$

We note that the idea of using threshold rounding was first used by Karger, Motwani, and Sudan [26] in their algorithm for approximate graph coloring. We claim that S is an m-orthogonal separator with ℓ_2 distortion $O(\sqrt{\log m})$, probability scale $\alpha = 1/m$ and $\beta = 0$. Let us verify that S satisfies the required conditions.

1. For every nonzero vector $\bar{u} \in X$, we have

$$\Pr(\bar{u} \in S) = \Pr(\langle \bar{u}, g\rangle \geq t\|\bar{u}\| \text{ and } r \leq \|\bar{u}\|^2) = \\ = \underbrace{\Pr(\langle \bar{u}/\|\bar{u}\|, g\rangle \geq t)}_{1/m} \cdot \underbrace{\Pr(r \leq \|\bar{u}\|^2)}_{\|\bar{u}\|^2} = \|\bar{u}\|^2/m \equiv \alpha\|\bar{u}\|^2.$$

Here we used that $\langle \bar{u}/\|\bar{u}\|, g\rangle$ is distributed as $\mathcal{N}(0,1)$, since $\bar{u}/\|\bar{u}\|$ is a unit vector. Then, by the choice of the threshold t, we have $\Pr(\langle \bar{u}/\|\bar{u}\|, g\rangle \geq t) = 1/m$. If $\bar{u} = 0$, then $\Pr(r \leq \|\bar{u}\|^2) = 0$, hence $\Pr(\bar{u} \in S) = 0$.

2. For every $\bar{u}, \bar{v} \in X$ with $\langle \bar{u}, \bar{v}\rangle = 0$, we have

$$\begin{aligned}\Pr(\bar{u}, \bar{v} \in S) &= \Pr(\langle \bar{u}, g\rangle \geq t;\ \langle \bar{v}, g\rangle \geq t;\ r \leq \|\bar{u}\|^2 \text{ and } r \leq \|\bar{v}\|^2) \\ &= \Pr(\langle \bar{u}, g\rangle \geq t\|\bar{u}\| \text{ and } \langle \bar{v}, g\rangle \geq t\|\bar{v}\|) \cdot \Pr(r \leq \min(\|\bar{u}\|^2, \|\bar{v}\|^2)).\end{aligned}$$

The random variables $\langle \bar{u}, g\rangle$ and $\langle \bar{v}, g\rangle$ are independent, since $\bar{u}$ and $\bar{v}$ are orthogonal vectors. Hence,

$$\Pr(\bar{u}, \bar{v} \in S) = \Pr(\langle \bar{u}, g\rangle \geq t\|\bar{u}\|) \cdot \Pr(\langle \bar{v}, g\rangle \geq t\|\bar{v}\|) \cdot \Pr(r \leq \min(\|\bar{u}\|^2, \|\bar{v}\|^2)).$$

Note that $\bar{u}/\|\bar{u}\|$ is a unit vector, and $\langle \bar{u}/\|\bar{u}\|, g\rangle \sim \mathcal{N}(0,1)$. Thus,

$$\Pr(\langle \bar{u}, g\rangle \geq t\|\bar{u}\|) = \Pr(\langle \bar{u}/\|\bar{u}\|, g\rangle \geq t) = 1/m.$$

Similarly, $\Pr(\langle \bar{v}, g\rangle \geq t\|\bar{v}\|) = 1/m$. Then, $\Pr(r \leq \min(\|\bar{u}\|^2, \|\bar{v}\|^2)) = \min(\|\bar{u}\|^2, \|\bar{v}\|^2)$, since r is uniformly distributed in $[0,1]$. We get

$$\Pr(\bar{u}, \bar{v} \in S) = \frac{\min(\|\bar{u}\|^2, \|\bar{v}\|^2)}{m^2} = \frac{\alpha \min(\|\bar{u}\|^2, \|\bar{v}\|^2)}{m}.$$

3. If $I_S(\bar{u}) \neq I_S(\bar{v})$, then either $\bar{u} \in S$ and $\bar{v} \notin S$, or $\bar{u} \notin S$ and $\bar{v} \in S$. Thus,

$$\Pr(I_S(\bar{u}) \neq I_S(\bar{v})) = \Pr(\bar{u} \in S; \bar{v} \notin S) + \Pr(\bar{u} \notin S; \bar{v} \in S).$$

We upper bound the both terms on the right hand side using the following lemma (switching $\bar{u}$ and $\bar{v}$ for the second term) and obtain the desired inequality.

▶ **Lemma 8.** *If* $\|\bar{u}\|^2 \geq \|\bar{v}\|^2$*, then*

$$\Pr(\bar{u} \in S; \bar{v} \notin S) \leq \alpha\mathcal{D}\|\bar{u} - \bar{v}\| \cdot \min(\|\bar{u}\|, \|\bar{v}\|) + \alpha\big|\|\bar{u}\| - \|\bar{v}\|\big|;$$

otherwise,

$$\Pr(\bar{u} \in S; \bar{v} \notin S) \leq \alpha\mathcal{D}\|\bar{u} - \bar{v}\| \cdot \min(\|\bar{u}\|, \|\bar{v}\|).$$

Proof of Lemma 8. We have

$$\Pr(\bar{u} \in S; \bar{v} \notin S) = \Pr(\langle \bar{u}, g\rangle \geq t\|\bar{u}\|;\ r \leq \|\bar{u}\|^2;\ v \notin S).$$

The event $\{\bar{v} \notin S\}$ is the union of two events $\{\langle \bar{v}, g\rangle \geq t\|\bar{v}\|$ and $r \leq \|\bar{v}\|^2\}$ and $\{r \geq \|\bar{v}\|^2\}$. Hence,

$$\begin{aligned}\Pr(\bar{u} \in S; \bar{v} \notin S) \leq\ & \Pr(\langle \bar{u}, g\rangle \geq t\|\bar{u}\|;\ \langle \bar{v}, g\rangle < t\|\bar{v}\|;\ r \leq \min(\|\bar{u}\|^2, \|\bar{u}\|^2)) \\ &+ \Pr(\langle \bar{u}, g\rangle \geq t\|\bar{u}\|;\ \|\bar{v}\|^2 \leq r \leq \|\bar{u}\|^2).\end{aligned} \tag{4}$$

Let $g_u = \langle \bar{u}/\|\bar{u}\|, g\rangle$ and $g_v = \langle \bar{v}/\|\bar{v}\|, g\rangle$. Both g_u and g_v are standard $\mathcal{N}(0,1)$ Gaussian random variables. Thus, $\Pr(g_u \geq t) = \Pr(g_v \geq t) = 1/m = \alpha$. We write (4) as follows:

$$\begin{aligned}\Pr(\bar{u} \in S; \bar{v} \notin S) &= \Pr(g_u \geq t;\ g_v < t)\Pr(r \leq \min(\|\bar{u}\|^2, \|\bar{v}\|^2)) && (5)\\ &\quad + \Pr(g_u \geq t)\Pr(\|\bar{v}\|^2 \leq r \leq \|\bar{u}\|^2) \\ &= \Pr(g_u \geq t;\ g_v < t) \cdot \min(\|\bar{u}\|^2, \|\bar{v}\|^2) + \alpha\Pr(\|\bar{v}\|^2 \leq r \leq \|\bar{u}\|^2). && (6)\end{aligned}$$

To finish the proof we need to estimate $\Pr(g_u \geq t;\ g_v < t)$ and $\Pr(\|\bar{v}\|^2 \leq r \leq \|\bar{u}\|^2)$. Since r is uniformly distributed in $[0,1]$, $\Pr(\|\bar{v}\|^2 \leq r \leq \|\bar{u}\|^2) = \|\bar{u}\|^2 - \|\bar{v}\|^2$, if $\|\bar{u}\|^2 - \|\bar{v}\|^2 > 0$; and $\Pr(\|\bar{v}\|^2 \leq r \leq \|\bar{u}\|^2) = 0$, otherwise.

We use Lemma 10 to upper bound $\Pr(g_u \geq t;\ g_v < t)$:

$$\Pr(g_u \geq t;\ g_v < t) \leq O\big(\sqrt{1 - \operatorname{cov}(g_u, g_v)} \cdot \sqrt{\log m}/m\big). \tag{7}$$

The covariance of g_u and g_v equals $\operatorname{cov}(g_u, g_v) = \langle \bar{u}/\|\bar{u}\|, \bar{v}/\|\bar{v}\|\rangle$ and $\|\bar{u} - \bar{v}\|^2 = \|\bar{u}\|^2 + \|\bar{v}\|^2 - 2\langle \bar{u}, \bar{v}\rangle$. Hence,

$$\begin{aligned}1 - \operatorname{cov}(g_u, g_v) = 1 - \frac{\|\bar{u}\|^2 + \|\bar{v}\|^2 - \|\bar{u} - \bar{v}\|^2}{2\|\bar{u}\|\,\|\bar{v}\|} &= \frac{\|\bar{u} - \bar{v}\|^2 - (\|\bar{u}\|^2 + \|\bar{v}\|^2 - 2\|\bar{u}\|\,\|\bar{v}\|)}{2\|\bar{u}\|\,\|\bar{v}\|} \\ &= \frac{\|\bar{u} - \bar{v}\|^2 - (\|\bar{u}\| - \|\bar{v}\|)^2}{2\|\bar{u}\|\,\|\bar{v}\|} \leq \frac{\|\bar{u} - \bar{v}\|^2}{2\|\bar{u}\|\,\|\bar{v}\|}.\end{aligned}$$

We plug this bound in (7) and get

$$\Pr(g_u \geq t;\ g_v < t) \leq \alpha \cdot \frac{\|\bar{u} - \bar{v}\|}{\sqrt{\|\bar{u}\|\,\|\bar{v}\|}} \cdot O(\sqrt{\log m}) \leq \alpha \cdot \frac{\|\bar{u} - \bar{v}\|}{\min(\|\bar{u}\|, \|\bar{v}\|)} \cdot O(\sqrt{\log m}).$$

Now, Lemma 8 follows from (6). This concludes the proof of Lemma 8 and Theorem 6. ◀ ◀

3.5 Gaussian Distribution

In this section, we prove several useful estimates on the Gaussian distribution. Let $X \sim \mathcal{N}(0,1)$ be one dimensional Gaussian random variable. Denote the probability that $X \geq t$ by $\bar{\Phi}(t)$:

$$\bar{\Phi}(t) = \Pr(X \geq t).$$

The first lemma gives a very accurate estimate on $\bar{\Phi}(t)$ for large t.

▶ **Lemma 9.** *For every $t > 0$,*

$$\frac{t}{\sqrt{2\pi}\,(t^2+1)} e^{-\frac{t^2}{2}} < \bar{\Phi}(t) < \frac{1}{\sqrt{2\pi}\,t} e^{-\frac{t^2}{2}}.$$

Proof. Write

$$\begin{aligned} \bar{\Phi}(t) &= \frac{1}{\sqrt{2\pi}} \int_t^\infty e^{-\frac{x^2}{2}} dx = \frac{1}{\sqrt{2\pi}} \left[\left. \frac{-e^{-\frac{x^2}{2}}}{x} \right|_t^\infty - \int_t^\infty \frac{e^{-\frac{x^2}{2}}}{x^2} dx \right] \\ &= \frac{1}{\sqrt{2\pi}t} e^{-\frac{t^2}{2}} - \frac{1}{\sqrt{2\pi}} \int_t^\infty \frac{e^{-\frac{x^2}{2}}}{x^2} dx. \end{aligned}$$

Thus,

$$\bar{\Phi}(t) < \frac{1}{\sqrt{2\pi}t} e^{-\frac{t^2}{2}}.$$

On the other hand,

$$\frac{1}{\sqrt{2\pi}} \int_t^\infty \frac{e^{-\frac{x^2}{2}}}{x^2} dx < \frac{1}{\sqrt{2\pi}t^2} \int_t^\infty e^{-\frac{x^2}{2}} dx = \frac{\bar{\Phi}(t)}{t^2}.$$

Hence,

$$\bar{\Phi}(t) > \frac{1}{\sqrt{2\pi}t} e^{-\frac{t^2}{2}} - \frac{\bar{\Phi}(t)}{t^2},$$

and, consequently,

$$\bar{\Phi}(t) > \frac{t}{\sqrt{2\pi}(t^2+1)} e^{-\frac{t^2}{2}}.$$ ◀

▶ **Lemma 10.** *Let X and Y be Gaussian $\mathcal{N}(0,1)$ random variables with covariance* $\mathrm{cov}(X,Y) = 1 - 2\varepsilon^2$. *Pick the threshold $t > 1$ such that $\bar{\Phi}(t) = 1/m$ for $m > 3$. Then*

$$\Pr(X \geq t \text{ and } Y \leq t) = O(\varepsilon\sqrt{\log m}/m).$$

Proof. If $\varepsilon t \geq 1$ or $\varepsilon \geq 1/2$, then we are done, since $\varepsilon\sqrt{\log m} = \Omega(\varepsilon t) = \Omega(1)$ and

$$\Pr(X \geq t \text{ and } Y \leq t) \leq \Pr(X \geq t) = \frac{1}{m}.$$

So we assume that $\varepsilon t \leq 1$ and $\varepsilon < 1/2$. Let

$$\xi = \frac{X+Y}{2\sqrt{1-\varepsilon^2}}; \qquad \eta = \frac{X-Y}{2\varepsilon}.$$

Note that ξ and η are $\mathcal{N}(0,1)$ Gaussian random variables with covariance 0. Hence, ξ and η are independent. We have

$$\Pr(X \geq t \text{ and } Y \leq t) = \Pr\big(\sqrt{1-\varepsilon^2}\,\xi + \varepsilon\eta \geq t \text{ and } \sqrt{1-\varepsilon^2}\,\xi - \varepsilon\eta \leq t\big).$$

Denote by $\mathcal{E}$ the following event:

$$\mathcal{E} = \big\{\sqrt{1-\varepsilon^2}\,\xi + \varepsilon\eta \geq t \text{ and } \sqrt{1-\varepsilon^2}\,\xi - \varepsilon\eta \leq t\big\}.$$

Then,

$$\Pr\bigl(X \geq t \text{ and } Y \leq t\bigr) = \Pr(\mathcal{E} \text{ and } \varepsilon\eta \leq t) + \Pr(\mathcal{E} \text{ and } \varepsilon\eta \geq t).$$

Observe that the second probability on the right hand side is very small. It is upper bounded by $\Pr(\varepsilon\eta \geq t)$, which, in turn, is bounded as follows:

$$\Pr(\varepsilon\eta \geq t) = \frac{1}{\sqrt{2\pi}} \int_{t/\varepsilon}^{\infty} e^{-\frac{x^2}{2}}\, dx = \bar{\Phi}(t/\varepsilon) \leq O\Bigl(\frac{\varepsilon\, e^{-\frac{t^2}{2\varepsilon^2}}}{t}\Bigr) \leq O\Bigl(\frac{\varepsilon\, e^{-\frac{t^2}{2}}}{t}\Bigr) = O(\varepsilon/m).$$

We now estimate the first probability:

$$\begin{aligned}\Pr(\mathcal{E} \text{ and } \varepsilon\eta \leq t) &= \mathbb{E}_\eta[\Pr(\mathcal{E} \text{ and } \eta \leq t/\varepsilon \mid \eta)]\\ &= \frac{1}{\sqrt{2\pi}} \int_0^{t/\varepsilon} \Pr(\mathcal{E} \mid \eta = x)\, e^{-x^2/2}\, dx\\ &= \frac{1}{\sqrt{2\pi}} \int_0^{t/\varepsilon} \Pr(\sqrt{1-\varepsilon^2}\xi \in [t-\varepsilon x, t+\varepsilon x])\, e^{-x^2/2}\, dx.\end{aligned}$$

The density of the random variable $\sqrt{1-\varepsilon^2}\,\xi$ in the interval $(t-\varepsilon x, t+\varepsilon x)$ for $x \in [0, t/\varepsilon]$ is at most

$$\frac{1}{\sqrt{2\pi(1-\varepsilon^2)}} e^{\frac{-(t-\varepsilon x)^2}{2(1-\varepsilon^2)}} \leq \frac{1}{2} e^{\frac{-(t-\varepsilon x)^2}{2}} \leq \frac{1}{2} e^{\frac{-t^2}{2}} \cdot e^{\varepsilon t x} \leq \frac{1}{2} e^{\frac{-t^2}{2}} \cdot e^{x},$$

here we used that $\varepsilon \geq 1/2$ and $\varepsilon t \geq 1$. Hence,

$$\Pr(t - \varepsilon x \leq \sqrt{1-\varepsilon^2}\,\xi \leq t + \varepsilon x) \leq \varepsilon x\, e^{\frac{-t^2}{2}} \cdot e^{x}.$$

Therefore,

$$\Pr(\mathcal{E} \text{ and } \varepsilon\eta \leq t) \leq \frac{\varepsilon\, e^{\frac{-t^2}{2}}}{\sqrt{2\pi}} \int_0^{t/\varepsilon} xe^{x} \cdot e^{\frac{-x^2}{2}} dx \leq \frac{\varepsilon\, e^{\frac{-t^2}{2}}}{\sqrt{2\pi}} \underbrace{\int_0^{\infty} xe^{x} \cdot e^{\frac{-x^2}{2}} dx}_{O(1)}.$$

The integral in the right hand side does not depend on any parameters, so it can be upper bounded by some constant (e.g. one can show that it is upper bounded by $2\sqrt{2\pi}$). We get

$$\Pr(\mathcal{E} \text{ and } \varepsilon\eta \leq t) \leq O(\varepsilon\, e^{\frac{-t^2}{2}}) = O(\varepsilon \cdot t\bar{\Phi}(t)) = O(\varepsilon\sqrt{\log m}/m).$$

This finishes the proof. ◀

4 CSPs of Higher Arities

In this section, we discuss techniques for solving Max k-CSP(d). We will not present any approximation algorithms for Max k-CSP(d), but rather we will describe an SDP relaxation for the problem and explain why rounding this SDP is challenging. To be specific, we will focus our attention on the regime when $k > \Omega(\log d)$. In this regime, the best known approximation is $\Omega(dk/d^k)$ [40].

Consider an instance of Max k-CSP(d). As we noted in the introduction, we may assume that all constraints are of the form $(x_{i_1} = j_1) \wedge \cdots \wedge (x_{i_k} = j_k)$. We write an SDP relaxation for the problem. In the SDP, we have two sets of variables. First, we have a variable $\bar{u}_{ij}$ for each x_i and $j \in D$; second, we have a variable $\bar{v}_C = \bar{v}_{(i_1,j_1),\dots,(i_k,j_k)}$ and for each constraint C of the form $(x_{i_1} = j_1) \wedge \dots \wedge (x_{i_k} = j_k)$. We denote the set of the constraints by $\mathcal{C}$.

$$\text{maximize} \sum_{C\in\mathcal{C}} \|\bar{v}_C\|^2$$

subject to

$$\sum_{j=1}^{d} \|\bar{u}_{ij}\|^2 = 1 \qquad \text{for every } i \tag{8}$$

$$\langle \bar{u}_{ij_1}, \bar{u}_{ij_2}\rangle = 0 \qquad \text{for every } i \text{ and } j_1 \neq j_2$$

$$\langle \bar{u}_{ij}, \bar{v}_C\rangle = \|\bar{v}_C\|^2 \qquad \text{for every } C \in \mathcal{C} \text{ and clause } x_i = j \text{ in } C$$

$$\langle \bar{u}_{ij}, \bar{v}_C\rangle = 0 \qquad \text{for every } C \in \mathcal{C} \text{ and clause } x_i = j \text{ not in } C \tag{9}$$

In the intended solution, for some unit vector $\bar{v}_0$, we have $\bar{u}_{ij} = \bar{v}_0$ if $x_i = j$, and $\bar{u}_{ij} = 0$, otherwise; $\bar{v}_C = \bar{v}_0$ if C is satisfied, and $\bar{v}_C = 0$, otherwise. Let us first consider a very basic rounding algorithm for the problem and discuss when it works and when it does not:

1. Choose a random Gaussian vector g with independent components distributed as $\mathcal{N}(0,1)$.
2. For every i, let $x_i = \arg\max_j |\langle \bar{u}_{ij}, g\rangle|$.

We analyze the algorithm. Let us consider a CSP constraint $C \in \mathcal{C}$ and estimate with what probability the algorithm satisfies it. Keep in mind that we want to get an $\Omega(kd/d^k)$ approximation, so the desired probability is $\Omega(kd\,|\bar{v}_C|^2/d^k)$. By renaming variables and values, we may assume that C is $(x_1 = 1) \vee \dots \vee (x_k = 1)$. Denote $\xi_{ij} = \langle \bar{u}_{ij}, g\rangle$. Note that ξ_{ij} is a Gaussian random variable with mean 0 and variance $\|\bar{u}_{ij}\|^2$. The probability that C is satisfied equals $\Pr(|\xi_{i1}| > |\xi_{ij}| \text{ for all } i \text{ and } j \neq 1)$. It is instructive to consider a very special case when the following two assumptions hold.

1. Since $\langle \bar{u}_{i1}, \bar{v}_C\rangle = \|\bar{v}_c\|^2$ for all i, we can write $\bar{u}_{i1} = \bar{v}_C + \bar{u}_{i1}^{\perp}$ where $\bar{u}_{i1}^{\perp} \perp \bar{v}_C$. Let us assume that all vectors $\bar{u}_{i1}^{\perp}$ are equal to 0, and thus $\bar{u}_{i1} = \dots = \bar{u}_{id} = \bar{v}_C$.
2. Let us assume that all vectors $\bar{u}_{ij}$ with $j \neq 1$ have the same length.

The first assumption is not essential, and we make it to slightly simplify the computations; however, the second assumption is crucial for our analysis. We have, $\xi_{11} = \dots = \xi_{k1}$. Let $\xi = \xi_{i1}$, η be a random variable distributed as $\mathcal{N}(0, \|\bar{u}_{ij}\|^2)$, and $\rho = \mathrm{Var}[\xi]\,/\,\mathrm{Var}[\eta] = \|\bar{v}_C\|^2/\|\bar{u}_{ij}\|^2$, where $j \neq 1$. Assume that $\rho \leq 1$. It follows from (9) that all random variables ξ_{ij}, with $j \neq 1$, are independent from ξ. Thus, for every M,

$$\begin{aligned}
&\Pr(|\xi| > |\xi_{ij}| \text{ for all } i \text{ and } j \neq 1) \geq \Pr(|\xi| \geq M \text{ and } |\xi_{ij}| < M \text{ for all } i \text{ and } j \neq 1)\\
&\quad = \Pr(|\xi| \geq M)\Pr(|\xi_{ij}| < M \text{ for } i \text{ and } j \neq 1) \overset{\text{Šidák}}{\geq} \Pr(|\xi| \geq M) \prod_{i;\, j\neq 1} \Pr(|\xi_{ij}| < M)\\
&\quad = \Pr(|\xi| \geq M)\Pr(|\eta| < M)^{k(d-1)}.
\end{aligned}$$

Here, we used Šidák's theorem to get the inequality on the second line. From Lemma 9, it is easy to prove that $\Pr(|\eta| \geq M) \leq \Pr(|\xi| \geq M)^{\rho}$ for every threshold M (see [40, Lemma 2.4]). Let $p = 1/(\rho k(d-1))$ and M be such that $\Pr(|\eta| \geq M) = p$. Then the probability that the constraint is satisfied is at least

$$p^{1/\rho}(1-p)^{k(d-1)} \approx \left(\frac{1}{\rho k(d-1)}\right)^{1/\rho} e^{-pk(d-1)} \approx \left(\frac{1}{\rho k d}\right)^{1/\rho} e^{-1/\rho} = \left(\frac{1}{e\rho k d}\right)^{1/\rho}. \tag{10}$$

When $\rho > c/k$ (for sufficiently large c), the probability of satisfying C is, loosely speaking, of order $1/d^{k/c}$, which is much greater than the desired probability $dk\|\bar{v}_C\|^2/d^k$. On the other hand, when $\rho < c/k$, we can use the random assignment rounding. Indeed, from (8) and our assumptions, we get

$$1 = \|\bar{u}_{i1}\|^2 + \sum_j \|\bar{u}_{ij}\|^2 = \|\bar{v}_C\|^2 + \frac{d-1}{\rho}\|\bar{v}_C\|^2.$$

Thus, $\|\bar{v}_C\|^2 \approx \rho/d < c/(dk)$ and the desired probability of satisfying the constraint is $O(1/d^k)$, which is less than the probability $1/d^k$ with which the random assignment algorithm satisfies the constraint (when constants in the $O(\cdot)$-notation are appropriately chosen). We see that if choose uniformly at random one of the two algorithms, the basic SDP rounding and the random assignment, we will satisfy every clause with the desired probability.

This argument can be made formal. However, it crucially uses that all vectors $\bar{u}_{ij}$, for $j \neq 1$, have the same length. If some of them are considerably longer than $1/\sqrt{d}$, they will be chosen disproportionately often. For instance, assume that $\|v_C\|^2 = c/(dk)$, a δ fraction of vectors $\bar{u}_{ij}$ have length approximately $1/\sqrt{\delta d}$, and the remaining vectors $\bar{u}_{ij}$ are equal to 0. In this setting, there are $d' \approx \delta d$ non-zero vectors $\bar{u}_{ij}$ for every i, and $\rho' = \|\bar{v}_C\|^2/\|\bar{u}_{ij}\|^2 \approx c\delta/k$ (for $j > 1$ such that $\bar{u}_{ij} \neq 0$). Now, loosely speaking, we can use formula (10) with $\rho = \rho'$ and $d = d'$ to estimate the probability of satisfying the constraint[4]. We get that the probability is approximately $1/d^{k/(c\delta)}$, which is much less than the desired probability when $\delta \ll 1/c$.

Let us discuss how we can fix the algorithm. If we knew that $\|\bar{u}_{i1}\|^2 \leq O(1/d)$ for all i, we would be able to restrict our attention only to those values j such that $\|\bar{u}_{ij}\|^2 \leq O(1/d)$; that is, for each i, we would choose j that maximizes $|\xi_{ij}|$ only among those j for which $\|\bar{u}_{ij}\|^2 \leq O(1/d)$. That would eliminate values j that the rounding procedure chooses with disproportionately large probabilities and thus fix the algorithm. On the other hand, if we knew that $\|\bar{u}_{i1}\|^2 > c/d$ for all i (for some sufficiently large constant $c > 1$), then we would be able to find a good assignment using another algorithm: for every i, we would choose j uniformly at random among those j for which $\|\bar{u}_{ij}\|^2 > c/d$. Note that for every i, there are at most d/c such values of j. Therefore, the probability that we choose $j = 1$ for every i is at least $(c/d)^k \gg dk/d^k$, as desired. In fact, of course, we may have $\|\bar{u}_{i1}\|^2 < c/d$ for some values of i and $\|\bar{u}_{i1}\|^2 > c/d$ for other values of i. Nevertheless, it is shown in [40] that it is possible to combine these two approaches and get an $\Omega(dk/d^k)$ approximation (when $k = \Omega(\log d)$). We refer the reader to [40] for details.

5 Minimum Multiway Cut

In this section, we describe known approximation results for the Minimum Multiway Cut problem. From a CSP viewpoint, the problem is a CSP of arity 2 over a domain D with two types of constraints:

- Equality constraints of the form $x_i = x_j$.
- Unary constraints of the form $x_i = j$, where $j \in D$.

The objective is to minimize the number of unsatisfied constraints. The problem is usually stated as a graph partitioning problem.

▶ **Definition 11.** We are given a graph $G = (V, E)$ and a set of terminals $T = \{s_1, \dots, s_d\} \subset V$. We need to partition the graph into d pieces $P_1, \dots, P_d$ such that $s_i \in P_i$. Our goal is to minimize the number of cut edges.

[4] We showed that formula (10) is a lower bound on the probability that C is satisfied; but, in fact, it gives a reasonable estimate on the probability.

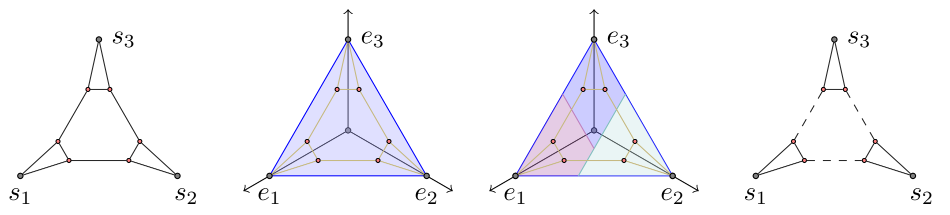


Figure 4 The figure shows (1) an input graph, (2) a feasible LP solution, (3) a random partitioning, and (4) the corresponding cut in the graph.

Observe that the two formulations are equivalent. Given a CSP instance, we construct a graph instance of the problem as follows: we introduce a vertex v_i for each variable x_i and add auxiliary vertices $s_1, \dots, s_d$; for each constraint $x_{i_1} = x_{i_2}$, we add an edge (v_{i_1}, v_{i_2}); for each constraint $x_i = j$ we add an edge (v_i, s_j). Similarly, we can transform a graph instance to a CSP instance. Note that there is a one-to-one correspondence between solutions to the problems: an assignment $A : \{x_i\} \to D$ corresponds to the partitioning $P_1, \dots, P_d$ with $P_j = \{v_i : A(x_i) = j\}$ and vice versa. Below, we will discuss the Multiway Cut problem in the standard graph partitioning formulation. Note that the problem can be solved in polynomial-time when $d = 2$ (then, it is equivalent to the Minimum Cut problem), but it is NP-hard for every $d \geq 3$ [16].

Minimum Multiway Cut was introduced by Dahlhaus, Johnson, Papadimitriou, Seymour, and Yannakakis [16] in 1994. Since then, a number of approximation algorithms have been proposed in the literature [16, 10, 27, 9, 47]. The best known algorithm is due to Sharma and Vondrák [47]. The algorithm gets a 1.309017 approximation; there is also a variant of the algorithm that gets a 1.2965 approximation, but the analysis of this algorithm is only computer-verified. Freund and Karloff [17] proved an integrality gap of $8/7 - o(1)$ for the problem. Manokaran, Naor, Raghavendra, and Schwartz [42] proved that the hardness of approximation factor for Minimum Multiway Cut is equal to the integrality gap if UGC holds. The results of [17, 42] imply a $8/7 - o(1)$ hardness of approximation (if UGC is true).

We note that the maximization version of the problem, in which the objective is to maximize the number of satisfied constraints received much less attention. Langberg, Rabani, and Swamy [35] designed a $(2 + \sqrt{2})/4 \approx 0.85355$ approximation algorithm for the problem and showed an integrality gap of $6/7 - o(1) \approx 0.85714 - o(1)$.

All known approximation algorithms for Minimum Multiway Cut – other than the 2-approximation algorithm by Dahlhaus et al. – use the linear programming (LP) relaxation by Călinescu, Karloff, and Rabani [10]. The recent algorithms by Buchbinder, Naor, and Schwartz [9] and Sharma and Vondrák [47] are significantly more involved than the algorithm by Călinescu et al. and, in particular, require computer-assisted fine tuning of the parameters to get the best approximation factors. Thus, in this survey, we describe the original LP relaxation and algorithm by Călinescu, Karloff, and Rabani [10].

Consider the following relaxation for the problem. For every vertex u, we introduce d LP variables $u_1, \dots, u_d$ and let $\bar{u} = (u_1, \dots, u_d)$. Let $\Delta = \{\bar{x} \in \mathbb{R}^d : x_1 + \dots + x_d = 1, x_1 \geq 0, \dots, x_d \geq 0\}$; Δ is a simplex with vertices $e_1 = (1, 0, \dots, 0)$, $e_2 = (0, 1, 0, \dots, 0)$, $\dots$, $e_d = (0, \dots, 0, 1)$ (see Figure 4). We write the following linear program.

$$\text{minimize } \frac{1}{2}\sum_{(u,v)\in E}\|\bar{u}-\bar{v}\|_1$$

subject to

$$\begin{aligned} &\bar{s}_j = e_j && \text{for every } j\in\{1,\dots,d\}\,,\\ &\bar{u}\in\Delta && \text{for every vertex } u. \end{aligned}$$

It is easy to see that this program is indeed a linear program – we can write the objective function as a linear function by introducing auxiliary variables. Namely, introduce additional LP variables y_{uvi}; add inequalities $y_{uvi} \geq u_i - v_i$ and $y_{uvi} \geq v_i - u_i$. Then, replace each term $\|\bar{u}-\bar{v}\|_1$ in the objective function with the expression $\sum_{i=1}^k y_{uvi}$.

We denote the value of an optimal solution by OPT, and the value of LP by LP.

▶ **Claim 12.** *The LP is a relaxation of the problem. That is,* $\mathsf{LP} \leq \mathsf{OPT}$.

Proof. Consider an optimal solution $P_1,\dots,P_d$. Let $\bar{u}=e_i$ for $u\in P_i$. Clearly, $\bar{u}$ is a feasible LP solution. We compute the value of this solution. Consider an edge $e=(u,v)$. Suppose that $u\in P_i$ and $v\in P_j$. The contribution of e to the objective function is

$$\frac{\|u-v\|_1}{2} = \frac{\|e_i-e_j\|_1}{2} = \begin{cases} 1, & \text{if } i\neq j,\\ 0, & \text{if } i=j. \end{cases}$$

That is, the contribution of e is 1 if e is cut, and 0, otherwise. Thus, the total contribution of all edges equals the number of cut edges. We get that the value of the LP solution $\{\bar{u}\}$ is OPT. Therefore, $\mathsf{LP}\leq\mathsf{OPT}$. ◀

▶ **Definition 13.** Given a feasible LP solution, we define a distance function $d(\cdot,\cdot)$ on the vertices of the graph:

$$d(u,v) = \frac{1}{2}\|\bar{u}-\bar{v}\|.$$

Let $B_r(u)$ be the ball of radius r around vertex u w.r.t. distance $d(\cdot,\cdot)$: $B_r(u) = \{v : d(u,v) < r\}$.

Now, we describe the algorithm by Călinescu, Karloff, and Rabani [10].

▶ **Theorem 14** (Călinescu, Karloff, and Rabani [10]). *There exists a 3/2-approximation algorithm for Minimum Multiway Cut.*

Proof. The algorithm is presented in Figure 5. We prove that the algorithm always returns a feasible solution.

▶ **Claim 15.** *The algorithm returns a feasible solution.*

Proof. Note that $s_i\in B_r(s_i)$ and $s_i\notin B_r(s_j)$ for every $j\neq i$. Therefore, s_i necessarily lies in P_i, as required. ◀

Now we compute the expected cost of the solution. Consider an edge $e=(u,v)$. Note that

$$d(u,s_i) = \frac{\|\bar{u}-e_i\|}{2} = \frac{(1-u_i)+\sum_{j\neq i}u_j}{2} = \frac{(1-u_i)+(1-u_i)}{2} = 1-u_i.$$

Approximation Algorithm for Multiway Cut

Input: graph $G = (V, E)$ and a set of terminals $T = \{s_1, \dots, s_d\} \subset V$.
Output: a partition $P_1, \dots, P_d$ of V such that $s_i \in P_i$.

```
solve the LP relaxation for Minimum Multiway Cut. Define $d(u,v) = \frac{1}{2}\|\bar{u} - \bar{v}\|_1$.
choose a random permutation $\pi$ of $\{1, \dots, d\}$
choose $r$ uniformly at random from $(0,1)$
let $A = \varnothing$
for $i = \pi(1), \pi(2), \dots, \pi(d-1)$ do
  let $P_i = B_r(s_i) \setminus A$
  let $A = A \cup P_i$
let $P_{\pi(d)} = V \setminus A$
return partition $P_1, \dots, P_d$
```

Figure 5 Approximation algorithm for Multiway Cut.

Let

$$A_i = \min(d(u, s_i), d(v, s_i)) \quad \text{and} \quad B_i = \max(d(u, s_i), d(v, s_i)).$$

We have,

$$B_i - A_i = |d(u, s_i) - d(v, s_i)| = |u_i - v_i|.$$

We may assume without loss of generality that

$$A_1 \le A_2 \le \dots \le A_d. \tag{11}$$

Let us write $i \prec j$ if $\pi^{-1}(i) < \pi^{-1}(j)$ (i precedes j in the order defined by π). We say that an index i settles the edge (u, v) if i is the first index w.r.t. π such that $u \in B_r(s_i)$ or $v \in B_r(s_i)$ (or both). In other words, index i settles edge e if $A_i \le r$ and $A_j > r$ for all $j \prec i$. Let $\mathcal{E}_i$ be the event that i settles (u, v). Note that at most one of the events $\mathcal{E}_i$ happens. (If no event $\mathcal{E}_i$ happens than u and v belong to $P_{\pi(k)}$, and the edge (u, v) is not cut.)

When the event $\mathcal{E}_i$ happens, we add either one or both of the vertices u and v to P_i. Note that in the former case, the edge (u, v) is cut since $u \in P_i$ and $v \notin P_i$; in the latter case, the edge (u, v) is not cut since $u, v \in P_i$. We conclude that

$$\Pr(e \text{ is cut}) = \sum_{i=1}^{d-1} \Pr(\mathcal{E}_i \text{ and } |\{u, v\} \cap B_r(s_i)| = 1) = \sum_{i=1}^{d-1} \Pr(\mathcal{E}_i \text{ and } A_i \le r < B_i).$$

We are going to show now that $\Pr(\mathcal{E}_i | r) \le 1/2$ for every $i > 1$. Consider the event $\mathcal{L}_i$ that $i \succ 1$. Since events $i \succ 1$ and $1 \succ i$ are equiprobable, event $\mathcal{L}_i$ happens with probability $1/2$. We claim that when $\mathcal{L}_i$ happens $\mathcal{E}_i$ does not happen, and, therefore,

$$\Pr(\mathcal{E}_i | r) \le 1 - \Pr(\mathcal{L}_i | r) = \frac{1}{2}.$$

Assume to the contrary that both events happen. Then

- $r \ge A_i$ and $r < A_j$ for every $j \prec i$,
- and $1 \prec i$,

therefore, $r \geq A_i$ and $r < A_1$; thus, $A_i < A_1$, which contradicts to our assumption (11). We have,

$$\begin{aligned}
\Pr(e \text{ is cut}) &= \sum_{i=1}^{d-1} \Pr(\mathcal{E}_i \text{ and } A_i \leq r < B_i) = \sum_{i=1}^{d-1} \mathbb{E}_r\left[\Pr(\mathcal{E}_i|r)\Pr(A_i \leq r < B_i|r)\right] \\
&\leq (B_1 - A_1) + \sum_{i=2}^{k} \frac{B_i - A_i}{2} = \frac{B_1 - A_1}{2} + \sum_{i=1}^{k} \frac{B_i - A_i}{2} \\
&= \frac{|u_1 - v_1|}{2} + \sum_{i=1}^{k} \frac{|u_i - v_i|}{2} = \frac{|u_1 - v_1| + \|\bar{u} - \bar{v}\|_1}{2}.
\end{aligned}$$

Observe that

$$\|u - v\|_1 \geq |u_1 - v_1| + \left|\sum_{i=2}^{k} u_i - \sum_{i=2}^{k} v_i\right| = |u_1 - v_1| + |(1 - u_1) - (1 - v_1)| = 2\,|u_1 - v_1|.$$

Thus $\Pr(e \text{ is cut}) \leq 3\|\bar{u} - \bar{v}\|_1/4$. By the linearity of expectation, the expected number of cut edges is at most

$$\sum_{(u,v)\in E} \frac{3}{4}\|\bar{u} - \bar{v}\|_1 = \frac{3\,\mathsf{LP}}{2} \leq \frac{3\,\mathsf{OPT}}{2}.$$

We proved that our algorithm gives a 3/2 approximation, in expectation. By running this algorithm many times we can get a $(3/2 + \varepsilon)$ approximation with high probability. In fact, the algorithm can be easily derandomized using the method of conditional expectations. ◀

6 Universal Rounding Algorithm

In this section, we discuss the hardness of approximation result by Raghavendra [44] and universal rounding algorithm by Raghavendra and Steurer [45]. Then, we describe in detail the rounding algorithm for a special case of CSPs of arity 2.

We consider a class of *generalized* CSPs of arity k with variables over a fixed domain $D = \{1, \ldots, d\}$ and predicates from a constant-size set Λ. Every predicate $\pi \in \Lambda$ is a function from D^k to $[-1, 1]$. We shall refer to this class of CSPs as k-CSP Λ. The value of a solution x_i^* for instance $\mathcal{I}$ of k-CSP Λ equals

$$\frac{1}{|\Pi|} \sum_{\pi(x_{i_1},\ldots,x_{i_k})\in\Pi} \pi(x_{i_1}^*, \ldots, x_{i_k}^*),$$

where Π is the set of predicates in $\mathcal{I}$. The expression above has a normalization factor $1/|\Pi|$; thus the value of any solution lies in the range $[-1, 1]$. Note that a regular (non-generalized) CSP is simply a generalized CSP in which all predicates take only values 0 and 1. We say that a predicate π is nonnegative if π is nonnegative on every assignment. Observe that finding an assignment of maximum value in an instance of k-CSP Λ is equivalent to finding an assignment of minimum value in the instance where every predicate π is replaced with $-\pi$. Thus, all results stated for maximization versions of *generalized* CSPs can be easily translated to results for minimization versions of *generalized* CSPs and vice versa. However, as we discuss later, the results for minimization and maximization CSPs with nonnegative predicates – and, in particular, results for minimization and maximization regular CSPs – are quite different.

In a breakthrough paper [44], Raghavendra showed that assuming the Unique Games Conjecture, the best possible approximation for many CSPs can be obtained by solving a standard SDP relaxation. To formally state his result, we describe the SDP relaxation and introduce some notation. First, we formulate the SDP for CSPs of arity 2; then, in the next section, we present the SDP for CSPs of higher arities. As we discussed earlier, for each variable x_i, we introduce SDP vector variables $\bar{u}_{i1}, \dots, \bar{u}_{id}$. We also introduce a special unit vector v_0.

$$\text{maximize } \frac{1}{|\Pi|} \sum_{\pi(x_i,x_j)\in\Pi} \sum_{a,b\in D} \pi(a,b)\langle \bar{u}_{ia}, \bar{u}_{jb}\rangle \tag{12}$$

$$\text{subject to} \tag{13}$$

$$\sum_{a\in D} \bar{u}_{ia} = v_0 \qquad \text{for all } i \tag{14}$$

$$\langle \bar{u}_{ia}, \bar{u}_{ib}\rangle = 0 \qquad \text{for all } i,j \in [n]; a \neq b \tag{15}$$

$$\langle \bar{u}_{ia}, \bar{u}_{jb}\rangle \geq 0 \qquad \text{for each constraint } \pi(x_i,x_j)\in\Pi \text{ and } a,b\in D \tag{16}$$

Let $\mathsf{OPT} = \mathsf{opt}(\mathcal{I})$ be the value of the optimal solution for instance $\mathcal{I}$, and $\mathsf{SDP} = \mathsf{sdp}(\mathcal{I})$ be the value of the optimal SDP solution for $\mathcal{I}$. Define gap as follows:

$$\mathsf{gap}(s) = \inf\{\mathsf{opt}(\mathcal{I}) : \mathsf{sdp}(\mathcal{I}) \geq s\}.$$

That is, $\mathsf{gap}(s)$ equals the minimum possible optimum value of an instance $\mathcal{I}$ with SDP value at least s.

▶ **Theorem 16** (Raghavendra [44]). *Assuming the Unique Games Conjecture, for every positive ε and every $s \in (-1,1)$, it is NP-hard to distinguish between instances $\mathcal{I}$ with $\mathsf{opt}(\mathcal{I}) \geq s$ and $\mathsf{opt}(\mathcal{I}) \leq \mathsf{gap}(s+\varepsilon)+\varepsilon$.*

Note that since we can find the optimal solution of the SDP in polynomial time, we can distinguish instances with (A) $\mathsf{opt}(\mathcal{I}) > s$ and (B) $\mathsf{opt}(\mathcal{I}) < \mathsf{gap}(s) - \varepsilon$: If $\mathsf{sdp}(\mathcal{I}) > \mathsf{gap}(s)$, then $\mathcal{I}$ is in the set (A), otherwise $\mathcal{I}$ is in the set (B). Furthermore, Raghavendra [44] and then, Raghavendra and Steurer [45] showed that given an instance with SDP value at least s, we can find an assignment of value at least $\mathsf{gap}(s-\varepsilon)-\varepsilon$. We present a proof of this theorem for 2-CSPs with nonnegative predicates in Section 6.2.

▶ **Theorem 17** (Raghavendra and Steurer [45]). *For every class of problems k-CSP Λ and every positive ε, there exists a randomized polynomial time algorithm that given an instance $\mathcal{I}$ of k-CSP Λ with the SDP value SDP finds a solution $\{x_i^*\}$ of value at least $\mathsf{gap}(\mathsf{SDP}-\varepsilon)-\varepsilon$.*

This result applies to both minimization and maximization problems. For a minimization problem, the algorithm by Raghavendra and Steurer [45] finds a solution of cost at most $\mathsf{gap_{min}}(\mathsf{SDP}+\varepsilon)+\varepsilon$, where $\mathsf{gap_{min}}(s) = \sup\{\mathsf{opt}(\mathcal{I}) : \mathsf{sdp}(\mathcal{I}) \leq s\}$.

Let us discuss what the results by Raghavendra [44] and Raghavendra and Steurer [45] imply for the three objectives we introduced in the introduction. Consider a generalized k-CSP Λ, and let α be the integrality gap of its SDP relaxation. Can we get an $(\alpha-\varepsilon)$ approximation using the algorithm by Raghavendra and Steurer [45]? Generally speaking, no. If the value of the optimal solution OPT is positive, but is very close to 0, then $\mathsf{gap}(\mathsf{SDP}-\varepsilon)-\varepsilon$ may be negative. Hence, we cannot even guarantee that the algorithm finds a solution of positive value.

Now, assume that all predicates $\pi \in \Lambda$ are nonnegative (in particular, this condition holds for regular CSPs), then the optimal value of any maximization k-CSP Λ is bounded away from 0 by some positive β (as we will see in a moment) and thus the algorithm always returns a solution of value at least

$$\alpha(\mathsf{SDP} - \varepsilon) - \varepsilon \geq \alpha\,\mathsf{OPT} - 2\varepsilon = \alpha\,\mathsf{OPT} - 2\beta(\varepsilon/\beta) \geq (\alpha - 2\varepsilon/\beta)\,\mathsf{OPT},$$

which is an $(\alpha - 2\varepsilon/\beta)$ approximation to the optimal value. Here we used that $\mathsf{gap}(s) \geq \alpha s$.

The constant β equals the expected value of a predicate π on a random input for the worst predicate $\pi \in \Lambda$:

$$\beta = \min_{\pi\in\Lambda} \mathbb{E}_{x_1,\dots,x_k\in D}[\pi(x_1,\dots,x_k)].$$

The value of a random assignment is at least β, in expectation, for any instance of k-CSP Λ. Thus the optimal value of any maximization k-CSP Λ is at least β.

Similarly, Theorem 16 implies that no algorithm can achieve a better than an $(\alpha + O(\varepsilon))$ approximation if UGC holds. Indeed, consider an integrality gap instance $\mathcal{I}$ with the integrality gap $\alpha' \leq \alpha + \varepsilon$. Then, by Theorem 16, it is NP-hard to find a solution of value at least $\alpha'(\mathsf{SDP}+\varepsilon)+\varepsilon$ given an instance of value SDP. Thus no algorithm has an approximation factor better than

$$\frac{\alpha'(\mathsf{SDP}+\varepsilon)+\varepsilon}{\mathsf{SDP}} \leq \alpha' + \frac{2\varepsilon}{\mathsf{SDP}} \leq \alpha' + \frac{2\beta(\varepsilon/\beta)}{\mathsf{OPT}} \leq \alpha + \varepsilon + 2\varepsilon/\beta.$$

▶ **Corollary 18** (Raghavendra and Steurer [45], Raghavendra [44]). *For every maximization k-CSP Λ with nonnegative predicates and every positive ε, there exists a polynomial time $(\alpha - \varepsilon)$-approximation algorithm, where α is the integrality gap of the SDP relaxation from Section 6.1. Further, for every positive ε, there is no $(\alpha + \varepsilon)$-approximation algorithm if UGC holds.*

Note that for many maximization CSPs the best approximation ratio α is still not known.

From Theorems 16 and 17, we cannot get an analog of Corollary 18 for minimization versions of generalized CSPs with nonnegative predicates and for regular CSPs with objective (3) (described in the introduction). The reason for that is that the cost of a minimization CSP can be arbitrarily close to 0.[5] Similarly, we cannot get an analog of Corollary 18 for objective (2). What we can get is the following. Let $f(\delta) = 1 - \mathsf{gap}(1-\delta)$. There exists an algorithm that given a $(1-\delta)$-satisfiable instance and parameter $\varepsilon > 0$, finds an assignment satisfying a $(1 - f(\delta+\varepsilon) - \varepsilon)$ fraction of the constraints. The running time of the algorithm is polynomial in n and exponential in $1/\varepsilon$. Note that typically we have to take $\delta = O(\varepsilon)$ to make this guarantee interesting. However, if we, say, let $\delta = \varepsilon$, we get an algorithm with running time exponential in $1/\delta$.

6.1 SDP Relaxation for k-CSPs with $k > 2$

The SDP relaxation for CSPs of arity $k > 2$ consists of two parts: a semidefinite program and a linear program connected by special constraints. The SDP part has the same variables $\bar{u}_{ia}$ and $\bar{v}_0$ as the SDP relaxation for $k = 2$. These variables must satisfy SDP constraints (14) and (15).

[5] In fact, most minimization CSPs studied in the literature – such as Min UnCut, Min 2CNF Deletion and Unique Games (with objective (3)) – do not admit a constant factor approximation if UGC holds. Their worst case instances have cost $o(1)$.

The LP part has a variable $p_{i_1\dots i_k}(a_1,\dots,a_k)\in[0,1]$ for every predicate $\pi(x_{i_1},\dots,x_{i_k})$ and $(a_1,\dots,a_k)\in D^k$. For every predicate $\pi(x_{i_1},\dots,x_{i_k})$, we define a local probability distribution on the assignments to the variables $x_{i_1},\dots,x_{i_k}$:

$$\widetilde{\Pr}_{i_1,\dots,i_k}\big((x_{i_1},\dots,x_{i_k})\in\mathcal{E}\big)=\sum_{(a_1,\dots,a_k)\in\mathcal{E}}p_{i_1\dots i_k}(a_1,\dots,a_k).$$

Formally, every $\widetilde{\Pr}_{i_1,\dots,i_k}$ is a linear combination of the LP variables $p_{i_1\dots i_k}(a_1,\dots,a_k)$. Now we can write the objective function as follows:

$$\frac{1}{|\Pi|}\sum_{\pi(x_{i_1},\dots,x_{i_k})\in\Pi}\ \sum_{(a_1,\dots,x_k)\in D^k}\pi(a_1,\dots,a_k)\widetilde{\Pr}_{i_1,\dots,i_k}\big(x_{i_1}=a_1,\dots,x_{i_k}=a_k\big).$$

We add an LP constraint $\widetilde{\Pr}_{i_1,\dots,i_k}\big((x_{i_1},\dots,x_{i_k})\in D^k\big)=1$. We then connect the LP and SDP by imposing the constraints

$$\widetilde{\Pr}_{i_1,\dots,i_k}(x_i=a)=\|\bar u_{ia}\|^2$$
$$\widetilde{\Pr}_{i_1,\dots,i_k}(x_i=a,\,x_j=b)=\langle\bar u_{ia},\bar u_{jb}\rangle$$

for all predicates $\pi(x_{i_1},\dots,x_{i_k})$ and $i,j\in\{i_1,\dots,i_k\}$. We refer the reader to the paper by Raghavendra [44] for more discussion on this SDP relaxation.

6.2 Rounding Algorithm for 2-CSPs with Nonnegative Predicates

The main observation behind the universal rounding algorithm is that for every SDP solution $\bar u_{ia}$ there exists another SDP solution $\bar u'_{ia}$ with a constant number (depending on ε and d) of distinct vectors $\bar u'_{ia}$ that has approximately the same SDP value as the original solution $\bar u_{ia}$. Here is the formal statement we need.

▶ **Theorem 19.** *For every positive ε, there exists a randomized polynomial time algorithm that given an instance $\mathcal{I}$ of a 2-CSP with the set of predicates Π and an SDP solution $\bar u_{ia}$ of value* SDP *returns a set of predicates Π' of size at least $(1-\varepsilon)|\Pi|$, an SDP solution $\bar u'_{ia}$ for the instance with predicates Π' of value at least $SDP-\varepsilon$, and a set $\mathcal{W}$ of size at most $f(\varepsilon,d)$ (for some function f depending only on ε and d) such that each tuple $w_i=(\bar u'_{i1},\dots,\bar u'_{id})$ belongs to $\mathcal{W}$. The algorithm fails with exponentially small probability.*

We prove Theorem 19 in Section 6.3. We now present and analyze the universal rounding algorithm.

The algorithm works in three phases: First, it solves the SDP relaxation and obtains a set of vectors $\bar u_{ia}$. Then, using Theorem 19, the algorithm transforms the SDP solution into another solution $\bar u'_{ia}$ of approximately the same value such that the number of distinct tuples $w_i=(\bar u'_{i1},\dots,\bar u'_{id})$ is upper bounded by some function $f(\varepsilon,d)$ of ε and d, which does not depend on the number of variables and constraints. We denote the set of all w_i by $\mathcal{W}$. The algorithm identifies variables x_i that are mapped to the same tuple of vectors w and obtains a new instance $\widetilde{\mathcal{I}}$ of 2-CSP Λ. Formally, the instance $\widetilde{\mathcal{I}}$ has a variable x'_w for each $w\in\mathcal{W}$ and a constraint $\pi(x'_{w_i},x'_{w_j})$ for each constraint $\pi(x_i,x_j)$ in $\mathcal{I}$. Note that $\widetilde{\mathcal{I}}$ has at most $f(\varepsilon,d)$ variables. The algorithm finds the optimal solution x^*_w for $\widetilde{\mathcal{I}}$ by enumerating all $d^{|\mathcal{W}|}$ possible solutions. Finally, the algorithm outputs the solution $x_i=x^*_{w_i}$ for the original instance $\mathcal{I}$.

Input: An instance $\mathcal{I}$ of 2-CSP Λ, parameter $\varepsilon > 0$.
Output: A solution x_i of value at least $\mathsf{gap}(\mathsf{SDP} - \varepsilon) - \varepsilon$.

1. Solve the SDP and obtain vectors $\bar{u}_{ia}$.
2. Transform vectors $\bar{u}_{ia}$ to vectors $\bar{u}'_{ia}$ and construct the set $\mathcal{W}$ using the algorithm from Theorem 19. Let $w_i = (\bar{u}'_{i1}, \dots, \bar{u}'_{id})$ for each i. By Theorem 19, $w_i \in \mathcal{W}$.
3. Build a new instance $\widetilde{\mathcal{I}}$ of 2-CSP Λ: For each $w \in \mathcal{W}$, create a variable x'_w in $\widetilde{\mathcal{I}}$. For each constraint $\pi(x_i, x_j)$ between x_i and x_j in $\mathcal{I}$, add the constraint $\pi(x'_{w_i}, x'_{w_j})$ between x'_{w_i} and x'_{w_j} to $\widetilde{\mathcal{I}}$.
4. Find the optimal solution $x^*_{w_i}$ for $\widetilde{\mathcal{I}}$ by enumerating all possible solutions of $\widetilde{\mathcal{I}}$.
5. Output the solution $x_i = x^*_{w_i}$.

The running time of the algorithm is exponential in $|\mathcal{W}|$ but is polynomial in n, since the size of $\mathcal{W}$ is upper bounded by $f(\varepsilon, d)$ which depends only on ε and d. We show that the algorithm finds a solution of cost at least $\mathsf{gap}(\mathsf{SDP} - \varepsilon)$. Denote by $\mathcal{I}'$ and $\widetilde{\mathcal{I}}'$ subinstances of $\mathcal{I}$ and $\widetilde{\mathcal{I}}$ in which we removed predicates from $\Pi \setminus \Pi'$ and kept predicates from Π'. By Theorem 19, the value of the SDP solution $\bar{u}'_{ia}$ on the instance $\mathcal{I}'$ is at least $\mathsf{SDP} - \varepsilon$. Observe that the SDP solution $\bar{u}'_{ia}$ corresponds to a feasible SDP solution for the instance $\widetilde{\mathcal{I}}'$ in which the vectors for the variable x'_{w_i} are $\bar{u}'_{i1}, \dots, \bar{u}'_{id}$. This SDP solution is well defined, since $(\bar{u}'_{i1}, \dots, \bar{u}'_{id}) = w_i = w_j = (\bar{u}'_{j1}, \dots, \bar{u}'_{jd})$ if $w_i = w_j$. The value of the SDP solution $\bar{u}'_{ia}$ on the instance $\mathcal{I}'$ equals the value of the corresponding SDP solution on $\widetilde{\mathcal{I}}'$ since for every constraint $\pi(x_i, x_j)$ in $\mathcal{I}'$, we have a constraint $\pi(x'_{w_i}, x'_{w_j})$ in $\widetilde{\mathcal{I}}'$, and the SDP value of the constraint π is the same in instances $\mathcal{I}'$ and $\widetilde{\mathcal{I}}'$: It is equal to $\sum_{ab} \pi(a,b)\langle \bar{u}'_{ia}, \bar{u}'_{jb}\rangle$. Since the optimal SDP value of the instance $\mathcal{I}'$ is at least $\mathsf{SDP} - \varepsilon$, the value of the optimal solution $x^*_{w_i}$ for $\mathcal{I}'$ is at least $\mathsf{gap}(\mathsf{SDP} - \varepsilon)$ (by the definition of $\mathsf{gap}(\cdot)$). The value of the solution $x_i = x^*_{w_i}$ for the instance $\mathcal{I}'$ is the same as the value of the solution $x^*_{w_i}$ for $\widetilde{\mathcal{I}}'$. If we omit the normalization factor $1/|\Pi|$ in the definition of the value of a solution, then the value of the solution x_i for the instance $\mathcal{I}$ will be greater than or equal to the value of the solution x_i for the instance $\mathcal{I}'$, since the value of removed predicates – predicates in $\Pi \setminus \Pi'$ – is nonnegative. With the normalization, we get that the value of the solution x_i for the original instance $\mathcal{I}$ is lower bounded by $(|\Pi'|/|\Pi|) \cdot \mathsf{gap}(\mathsf{SDP} - \varepsilon) \geq (1 - \varepsilon)\mathsf{gap}(\mathsf{SDP} - \varepsilon) \geq \mathsf{gap}(\mathsf{SDP} - \varepsilon) - \varepsilon$.

6.3 Proof of Theorem 19

Proof. We describe an algorithm for constructing the set $\mathcal{W}$ and vectors $\bar{u}'_{ia}$. The algorithm works in three steps: At the first step, it embeds the original SDP solution $\bar{u}_{ia}$ into a low dimensional space using the Johnson–Lindenstrauss transform [25]. Denote the embedding of the vector $\bar{u}_{ia}$ by $\varphi(\bar{u}_{ia})$. At the second step, the algorithm slightly perturbs vectors $\varphi(\bar{u}_{ia})$, so that the perturbed vectors $\varphi'(\bar{u}_{ia})$ satisfy all SDP constraints. At this step, the algorithm also removes some predicates from the set Π. Finally, at the third step, the algorithm picks an η-net (for sufficiently small η; $\eta = \varepsilon/C_d$) in the set of tuples $(\varphi'(\bar{u}_{i1}), \dots, \varphi'(\bar{u}_{id}))$ equipped with the norm $\ell_2 \oplus_\infty \cdots \oplus_\infty \ell_2$ and for every i finds $w_i \in \mathcal{W}$ closest to $(\varphi'(\bar{u}_{i1}), \dots, \varphi'(\bar{u}_{id}))$. It lets $\bar{u}'_{ia}$ be the a-th component of w_i. We show that for most variables x_i, x_j and values $a, b \in D$, $\langle \bar{u}'_{ia}, \bar{v}'_{jb}\rangle \approx \langle \bar{u}_{ia}, \bar{v}_{jb}\rangle$, and hence the SDP value of the solution $\bar{u}'_{ia}$ approximately equals the SDP value of the optimal SDP solution $\bar{u}_{ia}$. We describe each step of the algorithm in detail below. We use the following notation in the proof: for every nonzero vector $\bar{u}$, let $\nu(\bar{u}) = \bar{u}/\|\bar{u}\|$. Let $\nu(0) = 0$.

Input: An SDP solution $\{\bar{u}_{ia}\}$ of value SDP and a parameter $\varepsilon \in (0,1)$.
Output: A set $\mathcal{W}$ of size at most $f(\varepsilon, d)$ and an SDP solution $\{\bar{u}'_{ia}\}$ of value at least $\mathsf{SDP} - \varepsilon$ such that $w_i = (\bar{u}'_{i1}, \dots, \bar{u}'_{id}) \in \mathcal{W}$ for each i.

1. Embed $\bar{u}_{ia}$ into a low dimensional space using the Johnson–Lindenstrauss transform as described in Section 6.3.1 and obtain vectors $\varphi'(\bar{u}_{ia})$.
2. Transform vectors $\varphi(\bar{u}_{ia})$ to vectors $\varphi'(\bar{u}_{ia})$ using the Gram–Schmidt process (see Lemma 24) and the procedure from Lemma 25 with $\mu = d^2(3^{d+1}\, d + 1)\eta$.
3. Find an η-net $\mathcal{W}$ in the set of all tuples $(\varphi'(\bar{u}_{i1}), \dots, \varphi'(\bar{u}_{id}))$. For each i, let w_i be the vector in $\mathcal{W}$ closest to $(\varphi'(\bar{u}_{i1}), \dots, \varphi'(\bar{u}_{id}))$ in the norm $\ell_2 \oplus_\infty \cdots \oplus_\infty \ell_2$. Let $\bar{u}'_{ia}$ be the a-th component of the tuple w_i.

6.3.1 Step I: Johnson–Lindenstrauss Transform

We use the Johnson–Lindenstrauss lemma and two simple corollaries which we state now.

▶ **Theorem 20** (Johnson–Lindenstrauss [25]). *For every $\eta \in (0,1)$ and $\delta \in (0,1)$, there exists an integer $m = O(\log(1/\delta)/\varepsilon^2)$ and a family Φ of linear operators from $\mathbb{R}^n$ to $\mathbb{R}^m$ such that for every $\bar{u} \in \mathbb{R}^n$ and a random $\varphi \in \Phi$, we have*

$$\Pr_{\varphi}(\|\bar{u}\|^2 \le \|\varphi(\bar{u})\|^2 \le (1+\eta)\|\bar{u}\|^2) \ge 1 - \delta.$$

Moreover, a random operator φ can be sampled from Φ in randomized polynomial time.

▶ **Corollary 21.** *For every $\eta \in (0,1)$, $\delta \in (0,1)$ and every unit vector $\bar{v}_0 \in \mathbb{R}^n$, there exists an integer $m = O(\log(1/\delta)/\varepsilon^2)$ and a family Φ of linear operators from $\mathbb{R}^n$ to $\mathbb{R}^m$ such that for every $\bar{u} \in \mathbb{R}^n$ and a random $\varphi \in \Phi$, we have $\|\varphi(\bar{v}_0)\| = 1$ a.s. and*

$$\Pr_{\varphi}((1-\eta)\|\bar{u}\|^2 \le \|\varphi(\bar{u})\|^2 \le (1+\eta)\|\bar{u}\|^2) \ge 1 - \delta. \tag{17}$$

Moreover, a random operator φ can be sampled from Φ in randomized polynomial time.

Proof. We simply let $\varphi(\bar{u}) = \tilde{\varphi}(\bar{u})/\|\tilde{\varphi}(\bar{v}_0)\|$, where $\tilde{\varphi}$ is the random operator from Theorem 20. ◀

▶ **Corollary 22.** *For a random $\varphi \in \Phi$ from Corollary 21 we have: for every $\bar{u}, \bar{v} \in \mathbb{R}^n$*

$$\Pr_{\varphi}(\langle \nu(\bar{u}), \nu(\bar{v})\rangle - 3\eta \le \langle \varphi(\nu(\bar{u})), \varphi(\nu(\bar{v}))\rangle \le \langle \nu(\bar{u}), \nu(\bar{v})\rangle + 3\eta) \ge 1 - 4\delta.$$

Proof. Consider φ from Corollary 21. Assume that φ preserves the lengths of vectors $\nu(\bar{u})$, $\nu(\bar{v})$, and $(\nu(\bar{u}) \pm \nu(\bar{v}))$ up to a factor $(1 \pm \eta)$ (as in Equation (17)). This happens with probability at least $1 - 4\delta$. We have

$$\begin{aligned}
2\langle \varphi(\nu(\bar{u})), \varphi(\nu(\bar{v}))\rangle &= \|\varphi(\nu(\bar{u})) + \varphi(\nu(\bar{v}))\|^2 - \|\varphi(\nu(\bar{u}))\|^2 - \|\varphi(\nu(\bar{v}))\|^2 \\
&\ge (1-\eta)\,\|\nu(\bar{u}) + \nu(\bar{v})\|^2 - (1+\eta)\,\|\nu(\bar{u})\|^2 - (1+\eta)\,\|\nu(\bar{v})\|^2 \\
&\ge \big(\|\nu(\bar{u}) + \nu(\bar{v})\|^2 - \|\nu(\bar{u})\|^2 - \|\nu(\bar{v})\|^2\big) - \\
&\quad - \eta\big(\|\nu(\bar{u}) + \nu(\bar{v})\|^2 + \|\nu(\bar{u})\|^2 + \|\nu(\bar{v})\|^2\big) \\
&\ge 2\langle \nu(\bar{u}), \nu(\bar{v})\rangle - 6\eta.
\end{aligned}$$

By applying the bound above to vectors $\bar{u}$ and $-\bar{v}$, we get $\langle \varphi(\nu(\bar{u})), \varphi(\nu(\bar{v}))\rangle \le \langle \nu(\bar{u}), \nu(\bar{v})\rangle + 3\eta$. ◀

At the first step, the algorithm embeds vectors $\bar{u}_{ia}$ into $m = O(\log(1/\delta')/\varepsilon^2)$ dimensional space for $\delta' = \eta/(8d^2)$ and $\eta' = \eta/6$ using Corollary 22. We show that vectors $\varphi(\bar{u}_{ia})$ satisfy the SDP constraint (14), almost satisfy SDP constraints (16), and almost preserve inner products between vectors.

▶ **Lemma 23.**
1. *For every* $i \in [n]$, $\sum_{a\in D} \varphi(\bar{u}_{ia}) = \varphi(v_0)$, *and* $\|\varphi(v_0)\| = 1$ *a.s.*
2. *For every* $i \in [n]$,
$$\Pr_{\varphi}\Big(|\langle \nu(\varphi(\bar{u}_{ia})), \nu(\varphi(\bar{u}_{ib}))\rangle| \leq \eta \text{ for all } a \neq b\Big) \geq 1 - \eta.$$
3. *For all* $i, j \in [n]$,
$$\Pr_{\varphi}\Big(\langle \varphi(\bar{u}_{ia}), \varphi(\bar{u}_{jb})\rangle \geq \langle \bar{u}_{ia}, \bar{u}_{jb}\rangle - \eta \text{ for all } a, b\Big) \geq 1 - \eta.$$

Proof. Item 1 holds because φ is a linear operator and $\sum_{a\in D} \bar{u}_{ia} = \bar{v}_0$. Then, for every $a \neq b$, we have $\langle \bar{u}_{ia}, \bar{u}_{ib}\rangle = 0$, and hence $\langle \nu(\bar{u}_{ia}), \nu(\bar{u}_{ib})\rangle = 0$. By Corollary 22, $|\langle \varphi(\nu(\bar{u}_{ia})), \varphi(\nu(\bar{u}_{ib}))\rangle| \leq \eta/2$ with probability at least $1 - \eta/d^2$ for each $a \neq b$, and, by Corollary 21, $\|\varphi(\nu(\bar{u}_{ia}))\| \geq 1 - \eta/6$ with probability $1 - \eta/(4d^2)$ for each a. Thus, for every $i \in V$, with probability at least $1 - \eta$, we have for all a, b:
$$|\langle \nu(\varphi(\bar{u}_{ia})), \nu(\varphi(\bar{u}_{ib}))\rangle| = \frac{|\langle \varphi(\nu(\bar{u}_{ia})), \varphi(\nu(\bar{u}_{ib}))\rangle|}{\|\varphi(\nu(\bar{u}_{ia}))\|\,\|\varphi(\nu(\bar{u}_{ia}))\|} \leq \frac{\eta/2}{(1-\eta/6)^2} < \eta.$$
Hence, item 2 holds. Similarly, for every $i, j \in V$, with probability at least $1 - \eta$, we have for all a, b,
$$\langle \varphi(\bar{u}_{ia}), \varphi(\bar{u}_{jb})\rangle \geq \langle \bar{u}_{ia}, \bar{u}_{jb}\rangle - \eta\|\bar{u}_{ia}\|\|\bar{u}_{jb}\|/2 \geq \langle \bar{u}_{ia}, \bar{u}_{jb}\rangle - \eta/2,$$
and $\|\varphi(\bar{u}_{ia})\| \leq (1 + \eta/6)\|\bar{u}_{ia}\|$, $\|\varphi(\bar{u}_{jb})\| \leq (1 + \eta/6)\|\bar{u}_{jb}\|$. Consequently,
$$\langle \varphi(\bar{u}_{ia}), \varphi(\bar{u}_{jb})\rangle \geq \langle \bar{u}_{ia}, \bar{u}_{jb}\rangle - \eta$$
for all a, b with probability $1 - \eta$. Hence, item 3 holds. ◀

6.3.2 Step II: Fixing Violated SDP Constraints

Lemma 23 shows that vectors $\varphi(\bar{u}_{ia})$ satisfy SDP constraints (14) and almost satisfy constraints (15) and (16) as $\langle \bar{u}_{ia}, \bar{u}_{jb}\rangle \approx \langle \varphi(\bar{u}_{ia}), \varphi(\bar{u}_{jb})\rangle$ for most $i, j \in [n]$ and $a, b \in D$. At the second step, the algorithm slightly perturbs vectors $\varphi(\bar{u}_{ia})$ to fix all SDP constraints. First, the algorithm applies the Gram–Schmidt orthogonalization process (described in Lemma 24 below) to vectors $\varphi(\bar{u}_{i1}), \dots, \varphi(\bar{u}_{id})$ for each $i \in [n]$ and obtains vectors $\varphi^{\perp}(\bar{u}_{i1}), \dots, \varphi^{\perp}(\bar{u}_{id})$ that satisfy constraints (15). Then, the algorithm transforms vectors $\varphi^{\perp}(\bar{u}_{ia})$ into vectors $\varphi'(\bar{u}_{ia})$ using the procedure from Lemma 25 with $\mu = d^2(3^{d+1}\, d + 1)\eta$. This fixes most constraints (16). The algorithm removes those predicates $\pi \in \Pi$ for which the constraint (16) is still violated. We denote obtained vectors by $\varphi'(\bar{u}_{ia})$ and the set of remained predicates by Π'. We now state Lemma 24 and Lemma 25.

▶ **Lemma 24** (Gram–Schmidt process)**.** *There exists a polynomial time algorithm that given vectors* $\bar{v}_1, \dots, \bar{v}_d$ *of length at most 1.5 returns vectors* $\bar{v}'_1, \dots, \bar{v}'_d$ *such that (1)* $\langle \bar{v}'_a, \bar{v}'_b\rangle = 0$ *for* $a \neq b$, *(2)* $\sum_a \bar{v}_a = \sum_a \bar{v}'_a$, *and (3)* $\|\bar{v}'_a - \bar{v}_a\| \leq 3^d d\eta$, *where* $\eta = \max_{a,b} |\langle \nu(\bar{v}_a), \nu(\bar{v}_b)\rangle|$.

▶ **Lemma 25.** *There exists a polynomial-time algorithm that transforms any set of vectors $\bar{v}_{ia}$ satisfying the SDP constraints (14) and (15) into a set of vectors $\bar{v}'_{ia}$ also satisfying SDP constraints (14) and (15) for some unit vector $\bar{v}'_0$ such that $\langle \bar{v}'_{ia}, \bar{v}'_{jb}\rangle = (1-\mu)\langle \bar{v}_{ia}, \bar{v}_{jb}\rangle + \mu/d^2$ for all $i, j \in [n]$, $i \neq j$, $a, b \in D$, where $\mu \in [0,1]$ is a parameter.*

We first show that vectors $\varphi'(\bar{u}_{ia})$ satisfy all SDP constraints for the instance with predicates Π' and estimate the value of this SDP solution. Then, we prove Lemmas 24 and 25.

▶ **Lemma 26.** *Vectors $\varphi'(\bar{u}_{ia})$ together with the vector $\bar{v}'_0 = \varphi'(v_0)$ satisfy all SDP constraints for the set of predicates Π'. Further the expected value of the SDP solution $\varphi'(\bar{u}_{ia})$ is at least $(1-2\mu)\mathsf{SDP}$, where SDP is the value of the solution $\bar{u}_{ia}$. The expectation is taken over a random embedding $\varphi \in \Phi$.*

Proof of Lemma 26. First, observe that vectors $\varphi'(\bar{u}_{ia})$ satisfy all SDP constraints (16), simply because we remove all predicates $\pi(x_i, x_j)$ for which these constraints are violated. By Lemma 25, vectors $\varphi'(\bar{u}_{ia})$ satisfy SDP constraints (14) and (15) if vectors $\varphi^{\perp}(\bar{u}_{ia})$ satisfy these constraints. By Lemma 23, item 1, we have $\sum_{a\in D}\varphi(\bar{u}_{ia}) = \varphi(v_0)$ for all i. The Gram–Schmidt process preserves all sums $\sum_{a\in D}\varphi(\bar{u}_{ia})$ (by Lemma 24, item 2). Hence, vectors $\varphi^{\perp}(\bar{u}_{ia})$ satisfy SDP constraints (14). For every i, the Gram–Schmidt process transforms vectors $\bar{u}_{i1}, \dots, \bar{u}_{id}$ into orthogonal vectors; thus, $\varphi^{\perp}(\bar{u}_{ia})$ satisfy SDP constraints (15). This shows that vectors $\varphi'(\bar{u}_{ia})$ form a feasible SDP solution.

We now estimate the SDP value of the solution $\varphi'(\bar{u}_{ia})$. Let

$$V_\eta = \{i \in [n] : |\langle \nu(\varphi(\bar{u}_{ia})), \nu(\varphi(\bar{u}_{ib}))\rangle| \leq \eta \text{ for } a \neq b; \text{ and } \|\varphi(\bar{u}_{ia})\|^2 \leq 1.5 \text{ for all } a\};$$
$$\Pi_\eta = \{\pi(x_i, x_j) \in \Pi : \langle \varphi(\bar{u}_{ia}), \varphi(\bar{u}_{jb})\rangle \geq \langle \bar{u}_{ia}, \bar{u}_{ib}\rangle - \eta \text{ for } a, b \in D; i, j \in V_\eta\}.$$

By Lemma 23, the sets V_η and Π_η contain almost all variables and predicates, respectively. Specifically, for every i, $\Pr(i \in V_\eta) \geq 1-2\eta$; for all $\pi(x_i,x_j) \in \Pi$, $\Pr(\pi(x_i,x_j) \in \Pi_\eta) \geq 1-5\eta$. We show that for all $\pi(x_i,x_j) \in \Pi_\eta$, SDP constraints (16) are satisfied, and, thus, $\Pi_\eta \subset \Pi'$. We use the following simple claim.

▶ **Claim 27.** *Consider four vectors $\bar{v}_1, \bar{v}_2$ and $\bar{v}'_1, \bar{v}'_2$. Suppose that $\|\bar{v}_1 - \bar{v}'_1\| \leq \eta$, $\|\bar{v}_2 - \bar{v}'_2\| \leq \eta$, and $\|\bar{v}'_1\|, \|\bar{v}'_2\| \leq 1$ for some positive $\eta < 1$, then $\langle \bar{v}'_1, \bar{v}'_2\rangle \geq \langle \bar{v}_1, \bar{v}_2\rangle - 3\eta$.*

Proof. We have $\langle \bar{v}_1, \bar{v}_2\rangle = \big\langle \bar{v}'_1 - (\bar{v}'_1 - \bar{v}_1), \bar{v}'_2 - (\bar{v}'_2 - \bar{v}_2)\big\rangle \geq \langle \bar{v}'_1, \bar{v}'_2\rangle - \|\bar{v}'_1 - \bar{v}_1\|\,\|\bar{v}'_2\| - \|\bar{v}'_1\|\,\|\bar{v}'_2 - \bar{v}_2\| - \|\bar{v}'_1 - \bar{v}_1\|\|\bar{v}'_2 - \bar{v}_2\| \geq \langle \bar{v}'_1, \bar{v}'_2\rangle - 3\eta$. ◀

By the definition of Π_η, we have $\langle \varphi(\bar{u}_{ia}), \varphi(\bar{u}_{jb})\rangle \geq \langle \bar{u}_{ia}, \bar{u}_{jb}\rangle - \eta$. By Lemma 24, item 3, and Claim 27, we have $\langle \varphi^{\perp}(\bar{u}_{ia}), \varphi^{\perp}(\bar{u}_{jb})\rangle \geq \langle \varphi(\bar{u}_{ia}), \varphi(\bar{u}_{jb})\rangle - 3^d d\eta$. Then, by Lemma 25, $\langle \varphi'(\bar{u}_{ia}), \varphi'(\bar{u}_{jb})\rangle = (1-\mu)\langle \varphi^{\perp}(\bar{u}_{ia}), \varphi^{\perp}(\bar{u}_{jb})\rangle + \mu/d^2$. Putting these inequalities together, we get

$$\langle \varphi'(\bar{u}_{ia}), \varphi'(\bar{u}_{jb})\rangle \geq (1-\mu)\langle \bar{u}_{ia}, \bar{u}_{ib}\rangle - (3^{d+1}\,d + 1)\eta + \mu/d^2 = (1-\mu)\langle \bar{u}_{ia}, \bar{u}_{ib}\rangle. \tag{18}$$

Here, we used that $\mu/d^2 = (3^{d+1}\,d+1)\eta$. Equation (18) implies that SDP constraints (16) are satisfied for vectors $\varphi'(\bar{u}_{ia})$.

We now estimate the value of the SDP solution. For every predicate $\pi(x_i, x_j)$ in Π_η, we have

$$\sum_{a,b\in D} \pi(a,b)\langle \varphi'(\bar{u}_{ia}), \varphi'(\bar{u}_{jb})\rangle \geq (1-\mu)\sum_{a,b\in D}\pi(a,b)\langle \bar{u}_{ia}, \bar{u}_{jb}\rangle.$$

Every predicate $\pi(x_i, x_j)$ in Π belongs to Π' with probability at least $(1-5\eta)$. Thus the expected SDP value for vectors $\varphi'(\bar{u}_{ia})$ is at least $(|\Pi|/|\Pi'|)\,(1-5\eta)(1-\mu)\mathsf{SDP} \geq (1-2\mu)\mathsf{SDP}$.

The multiplicative factor $(|\Pi|/|\Pi'|) \geq 1$ is due to the normalization factor $1/|\Pi|$ in the SDP objective. This finishes the proof of Lemma 26. ◀

Proof of Lemma 24. In the proof, we denote the projection of a vector $\bar{v}$ to a non-zero vector $\bar{u}$ by $\operatorname{proj}_{\bar{u}} \bar{v}$:

$$\operatorname{proj}_{\bar{u}} \bar{v} = \frac{\langle \bar{u}, \bar{v} \rangle}{\|\bar{u}\|^2}\, \bar{u} = \langle \bar{v}, \nu(\bar{u}) \rangle \nu(\bar{u}).$$

As before $\nu(\bar{u}) = \bar{u}/\|\bar{u}\|$ for $\bar{u} \neq 0$. Observe that $\operatorname{proj}_{\bar{u}} \bar{v}$ is collinear with $\bar{u}$; and $(\bar{v} - \operatorname{proj}_{\bar{u}} \bar{v})$ is orthogonal to $\bar{u}$.

We describe an algorithm that transforms vectors $\bar{v}_1, \dots \bar{v}_d$ to orthogonal vectors $\bar{v}'_1, \dots, \bar{v}'_d$. The algorithm works in d iterations. After iteration t, it obtains a set of vectors $\bar{v}_1(t), \dots, \bar{v}_d(t)$ satisfying the following properties: $(1')$ $\langle \bar{v}_a(t), \bar{v}_b(t) \rangle = 0$ for $a, b \leq t$, and $a \neq b$, $(2')$ $\sum_a \bar{v}_a(t) = \sum_a \bar{v}_a$, $(3')$ $\|\nu(\bar{v}_a(t)) - \nu(\bar{v}_a)\| \leq 3^a \eta$ for $a \leq t$ and $\bar{v}_a(t) = \bar{v}_a$ for $a > t$, and $(4')$ $\|\bar{v}_a(t) - \bar{v}_a\| \leq 3^t \eta$ for all a. The algorithm returns vectors $\bar{v}'_a = \bar{v}_a(d)$. Note that the desired conditions (1)–(3) follow from the conditions $(1')$–$(4')$ on vectors $\bar{v}_a(d)$. At the first iteration, we set $\bar{v}_a(1) = \bar{v}_a$ for all a. At iteration $t \geq 2$, we let

$$\bar{v}_t(t) = \bar{v}_t(t-1) - \sum_{a<t} \operatorname{proj}_{\bar{v}_a(t-1)} \bar{v}_t(t-1); \tag{19}$$

$$\bar{v}_b(t) = \bar{v}_b(t-1) + \operatorname{proj}_{\bar{v}_b(t-1)}(\bar{v}_t(t-1) - \bar{v}_t(t))) \qquad \text{for } b < t; \tag{20}$$

$$\bar{v}_b(t) = \bar{v}_t(t-1) \qquad \text{for } b > t. \tag{21}$$

We prove by induction that properties $(1')$–$(4')$ hold. It is easy to see that the properties hold for vectors $\bar{v}_a(0)$, since $\bar{v}_a(0) = \bar{v}_a$. Consider $t > 1$. Observe that vectors $\bar{v}_a(t)$ are collinear with vectors $\bar{v}_a(t-1)$ for $a < t$ and $\bar{v}_a(t) = \bar{v}_a(t-1)$ for $a > t$. The only vector that changes the direction is the vector $\bar{v}_t(t)$. The sum $\sum_{a<t} \operatorname{proj}_{\bar{v}_a(t-1)} \bar{v}_t(t-1)$ equals the projection of the vector $\bar{v}_t(t-1)$ to the span of orthogonal vectors $\bar{v}_1(t-1), \dots, \bar{v}_{t-1}(t-1)$. Hence, $\bar{v}_t(t)$ is orthogonal to $\bar{v}_1(t-1), \dots, \bar{v}_{t-1}(t-1)$, and, also, to vectors $v_1(t), \dots, \bar{v}_{t-1}(t)$, which are collinear with $\bar{v}_1(t-1), \dots, \bar{v}_{t-1}(t-1)$. Thus, property $(1')$ holds for t. The sum of vectors on the left hand side of (19–21) equals the sum of vectors on the right hand side of (19–21). Thus, property $(2')$ holds. For all $a \neq t$, we have $\nu(\bar{v}_a(t)) = \nu(\bar{v}_a(t-1))$. So we need to check $(3')$ only for $a = t$. Since $\bar{v}_t(t-1) = v_t$ and $\nu(\bar{v}_t(t-1)) = \nu(\bar{v}_t)$, we get

$$\begin{aligned}
\|\bar{v}_t(t) - \bar{v}_t\| &= \|\sum_{b<t} \operatorname{proj}_{\bar{v}_b(t)} \bar{v}_t\| = \|\sum_{b<t} \langle \nu(\bar{v}_b(t)), \bar{v}_t \rangle\, \nu(\bar{v}_b(t))\| \leq \sum_{b<t} |\langle \nu(\bar{v}_b(t)), \bar{v}_t \rangle| \\
&= \sum_{b<t} \big(|\langle \nu(\bar{v}_b), \bar{v}_t \rangle| + |\langle \nu(\bar{v}_b(t)) - \nu(\bar{v}_b), \bar{v}_t \rangle|\big) \\
&\leq \sum_{b<t} \big(|\langle \nu(\bar{v}_b), \nu(\bar{v}_t) \rangle| \cdot \|\bar{v}_t\| + \|\nu(\bar{v}_b(t)) - \nu(\bar{v}_b)\|\, \|\bar{v}_t\||\big) \\
&\leq \|\bar{v}_t\| \cdot \sum_{b<t} \big(|\langle \nu(\bar{v}_b), \nu(\bar{v}_t) \rangle| + \|\nu(\bar{v}_b(t)) - \nu(\bar{v}_b)\|\big).
\end{aligned}$$

We now upper bound $|\langle \nu(\bar{v}_b), \nu(\bar{v}_t) \rangle| \leq \eta$ and $\|\nu(\bar{v}_b(t)) - \nu(\bar{v}_b)\| \leq 3^b \eta$ and obtain the following inequality:

$$\|\bar{v}_t(t) - \bar{v}_t\| \leq \sum_{b<t} (\eta + 3^b \eta) \|\bar{v}_t\| = \big((t-1) + (3^t - 3)/2\big)\eta \|\bar{v}_t\| \leq 0.6 \cdot 3^t \eta \|\bar{v}_t\|. \tag{22}$$

The last inequality can be easily verified numerically. Let α be the angle between $\bar{v}_t(t)$ and $\bar{v}_t$. The formula above shows that $\sin(\alpha) = \|\bar{v}_t(t) - \bar{v}_t\|/\|\bar{v}_t\| \leq 0.6 \cdot 3^d \eta$. Observe that

$\|\nu(\bar{v}_t(t)) - \nu(\bar{v}_t)\| = 2\sin(\alpha/2)$ and $2\sin(\alpha/2) = \sin(\alpha)/\cos(\alpha/2) < 3^d$, since $\cos(\alpha/2) > 0.6$, if $\sin\alpha < 1/3$. This proves property (3′).

Finally, property (4′) holds, since for $a < t$, $\|\bar{v}_a(t) - \bar{v}_a(t-1)\| = |\langle \bar{v}_t(t) - \bar{v}_t(t-1), \nu(\bar{v}_a(t-1))\rangle| \leq 3^t\eta$ by (22); $\|\bar{v}_t(t) - \bar{v}_t(t-1)\| \leq 3^t\eta$ also by (22); and for $a > t$, $\bar{v}_a(t) = \bar{v}_a(t-1)$. ◀

Proof of Lemma 25. Observe that it is sufficient to prove Lemma 25 for $\mu^* = 1$: If vectors $\bar{z}_{ia}$ satisfy the conditions of this lemma for $\mu^* = 1$, then vectors $\bar{v}'_{ia} = \sqrt{1-\mu}\,\bar{v}_{ia} \oplus \sqrt{\mu}\,\bar{z}_{ia}$ satisfy the conditions of the lemma for any μ (with $\bar{v}'_0 = \sum_a \bar{v}'_{ia}$, which does not depend on i). Here $\oplus$ denotes the direct sum. This follows from the following identity: $\langle \bar{v}'_{ia}, \bar{v}'_{jb}\rangle = (1-\mu)\langle \bar{v}_{ia}, \bar{v}_{jb}\rangle + \mu\langle \bar{z}_{ia}, \bar{z}_{jb}\rangle$.

To construct vectors $\bar{z}_{ia}$, consider the random assignment algorithm. The algorithm independently assigns a random value from D to every variable x_i. Let x_{ia}^{rnd} be the indicator random variable of the event that the algorithm sets $x_i = a$. The random variables x_{ia}^{rnd} lie in the L_2 space equipped with the standard inner product $\langle x_{ia}^{rnd}, x_{jb}^{rnd}\rangle = \mathbb{E}[x_{ia}^{rnd}\, x_{jb}^{rnd}]$. It easy to see that $\|x_{ia}^{rnd}\|^2 = 1/d$, $\langle x_{ia}^{rnd}, x_{jb}^{rnd}\rangle = 1/d^2$ for $i \neq j$, $\langle x_{ia}^{rnd}, x_{ib}^{rnd}\rangle = 0$ for $a \neq b$. Furthermore, $\sum_a x_{ia}^{rnd} = 1$ for every i. We isometrically embed x_{ia}^{rnd} from L_2 into ℓ_2 and obtain vectors $\bar{z}_{ia}$. ◀

6.3.3 Step III: Rounding to a η-Net

At the last step, the algorithm picks an η-net in the set of all tuples $(\varphi'(\bar{u}_{i1}), \dots, \varphi'(\bar{u}_{id}))$ equipped with the norm $\ell_2 \oplus_\infty \cdots \oplus_\infty \ell_2$. In this norm, the distance between two tuples $(\varphi'(\bar{u}_{i1}), \dots, \varphi'(\bar{u}_{id}))$ and $(\varphi'(\bar{u}_{j1}), \dots, \varphi'(\bar{u}_{jd}))$ equals $\max_{a\in D}\|\varphi'(\bar{u}_{ia}) - \varphi'(\bar{u}_{ja})\|_2$. We denote the η-net by $\mathcal{W}$. The size of an η-net in the m-dimensional space, where all vectors $\varphi'(\bar{u}_{ia})$ lie, is upper bounded by $(1+2/\eta)^m$. Thus the size of $\mathcal{W}$ is upper bounded by $(1+2/\eta)^{md}$. This number, $(1+2/\eta)^{md}$, depends on m, d and η, which in turn depend only on ε and d. For each $i \in [n]$, the algorithm picks $w_i \in \mathcal{W}$ closest to $(\varphi'(\bar{u}_{i1}), \dots, \varphi'(\bar{u}_{id}))$, sets $\bar{u}'_{ia}$ to be the a-th component of w_i, and outputs all vectors $\bar{u}'_{ia}$.

Observe that for each i there is a j such that $\bar{u}'_{ia} = \varphi'(\bar{u}_{ja})$ for all a. Thus, vectors $\bar{u}'_{ia}$ satisfy all SDP constraints. We need to lower bound the SDP value of the solution $\bar{u}'_{ia}$. Note that for each i and a, $\|\bar{u}_{ia} - \varphi'(\bar{u}_{ia})\| \leq \eta$, since $\mathcal{W}$ is an η-net. By Claim 27, $\langle \bar{u}'_{ia}, \bar{u}'_{jb}\rangle \geq \langle \varphi'(\bar{u}_{ia}), \varphi'(\bar{u}_{jb})\rangle - 3\eta$. Thus, the value of the SDP solution u'_{ia} is at least the value of the SDP solution for $\varphi'(\bar{u}_{ia})$ minus $3d^2\eta$. Thus, it is lower bounded by $(1-2\mu)\mathsf{SDP} - 3d^2\eta \geq \mathsf{SDP} - d^2(3^{d+1}\,d + 4)\eta$, where SDP is the SDP value of the solution $\bar{u}_{ia}$. This finishes the proof of Theorem 20. ◀

7 Open Problems

The most important open problem in the field is to prove or disprove the Unique Games Conjecture. Here, we list some other interesting open problems.

▶ Open Problem 1. Close the gap between the known approximation factors and hardness results for the following problems: Max 2-And, Max SAT, Multiway Cut.

▶ Open Problem 2. We know that the best possible approximation factor for Max k-And and all Boolean k-CSPs is $c_k k/2^k$, where $c_k = \Theta(1)$. Find the limit of c_k as $k \to \infty$.

Austrin and Mossel [6] and Chan [11] showed that $c_k \leq 1 + o(1)$. We conjecture that this upper bound is tight, $c_k = 1 \pm o(1)$, and, moreover, the algorithm from [40] has an approximation factor of $(1 - o(1))k/2^k$.

▶ Open Problem 3. Prove or disprove that the currently best known approximation factor of $\Omega(d \max(k, \log d)/d^k)$ for k-CSP(d) is asymptotically optimal. It is known that this approximation factor is optimal when $d = \Omega(k)$ [11].

▶ Open Problem 4. Suppose that the integrality gap of a minimization k-CSP Λ is α_n (α_n may depend on the number of variables n). Does there exist a polynomial-time algorithm with an approximation factor $(1+\varepsilon)\alpha_n$ for every positive ε?

References

1 Amit Agarwal, Moses Charikar, Konstantin Makarychev, and Yury Makarychev. $O(\sqrt{\log n})$ approximation algorithms for Min UnCut, Min 2CNF Deletion, and directed cut problems. In *Proceedings of the Symposium on Theory of Computing*, pages 573–581, 2005. doi:10.1145/1060590.1060675.

2 Sanjeev Arora, Boaz Barak, and David Steurer. Subexponential algorithms for unique games and related problems. In *In Proceedings of the Symposium on Foundations of Computer Science*, pages 563–572, 2010. doi:10.1109/FOCS.2010.59.

3 Sanjeev Arora, Rong Ge, and Ali Kemal Sinop. Towards a better approximation for sparsest cut? In *Proceedings of the Foundations of Computer Science*, pages 270–279, 2013.

4 Sanjeev Arora, Subhash A. Khot, Alexandra Kolla, David Steurer, Madhur Tulsiani, and Nisheeth K. Vishnoi. Unique games on expanding constraint graphs are easy. In *Proceedings of the Symposium on Theory of Computing*, pages 21–28, 2008. doi:10.1145/1374376.1374380.

5 Per Austrin. Towards sharp inapproximability for any 2-CSP. *SIAM Journal on Computing*, 39(6):2430–2463, 2010.

6 Per Austrin and Elchanan Mossel. Approximation resistant predicates from pairwise independence. *Computational Complexity*, 18(2):249–271, 2009.

7 Adi Avidor, Ido Berkovitch, and Uri Zwick. Improved approximation algorithms for Max NAE-SAT and Max SAT. In *Approximation and Online Algorithms*, pages 27–40. Springer, 2005.

8 Nikhil Bansal, Uriel Feige, Robert Krauthgamer, Konstantin Makarychev, Viswanath Nagarajan, Joseph Naor, and Roy Schwartz. Min-max graph partitioning and small set expansion. In *FOCS*, pages 17–26, 2011.

9 Niv Buchbinder, Joseph Seffi Naor, and Roy Schwartz. Simplex partitioning via exponential clocks and the multiway cut problem. In *Proceedings of the Symposium on Theory of Computing*, pages 535–544, 2013.

10 Gruia Călinescu, Howard Karloff, and Yuval Rabani. An improved approximation algorithm for multiway cut. In *Proceedings of the Symposium on Theory of Computing*, pages 48–52, 1998.

11 Siu On Chan. Approximation resistance from pairwise independent subgroups. In *Proceedings of the Symposium on Theory of Computing*, pages 447–456. ACM, 2013.

12 Moses Charikar, Konstantin Makarychev, and Yury Makarychev. Near-optimal algorithms for unique games. In *Proceedings of the Symposium on Theory of Computing*, pages 205–214, 2006. doi:10.1145/1132516.1132547.

13 Moses Charikar, Konstantin Makarychev, and Yury Makarychev. Near-optimal algorithms for maximum constraint satisfaction problems. *ACM Transactions on Algorithms (TALG)*, 5(3):32, 2009.

14 Eden Chlamtac, Konstantin Makarychev, and Yury Makarychev. How to play unique games using embeddings. In *Proceedings of the Symposium on Foundations of Computer Science*, pages 687–696, 2006.

15 Miroslav Chlebík and Janka Chlebíková. On approximation hardness of the Minimum 2SAT-Deletion problem. In *Proceedings of the International Symposium on Mathematical Foundations of Computer Science*, pages 263–273, 2004.

16 Elias Dahlhaus, David S. Johnson, Christos H. Papadimitriou, Paul D. Seymour, and Mihalis Yannakakis. The Complexity of Multiterminal Cuts. *SIAM Journal on Computing*, 23:864–894, 1994. `doi:10.1137/S0097539792225297`.

17 Ari Freund and Howard Karloff. A lower bound of 8/(7+ 1k- 1) on the integrality ratio of the călinescu–karloff–rabani relaxation for multiway cut. *Information Processing Letters*, 75(1):43–50, 2000.

18 Michel X. Goemans and David P. Williamson. Improved approximation algorithms for Maximum Cut and Satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995.

19 Anupam Gupta and Kunal Talwar. Approximating unique games. In *Proceedings of the Symposium on Discrete Algorithm*, pages 99–106, 2006.

20 Venkatesan Guruswami, Johan Håstad, Rajsekar Manokaran, Prasad Raghavendra, and Moses Charikar. Beating the random ordering is hard: Every ordering CSP is approximation resistant. *SIAM Journal on Computing*, 40(3):878–914, 2011.

21 Venkatesan Guruswami and Euiwoong Lee. Complexity of approximating csp with balance/hard constraints. In *Proceedings of the Conference on Innovations in Theoretical Computer Science*, pages 439–448, 2014.

22 Venkatesan Guruswami and Yuan Zhou. Tight bounds on the approximability of almost-satisfiable horn sat and exact hitting set. In *Proceedings of the Symposium on Discrete Algorithms*, pages 1574–1589, 2011.

23 Johan Håstad. Some optimal inapproximability results. *Journal of the ACM (JACM)*, 48(4):798–859, 2001.

24 Johan Håstad, Sangxia Huang, Rajsekar Manokaran, Ryan O'Donnell, and John Wright. Improved NP-Inapproximability for 2-Variable Linear Equations. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2015, August 24-26, 2015, Princeton, NJ, USA*, volume 40 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 341–360. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2015. `doi:10.4230/LIPIcs.APPROX-RANDOM.2015.341`.

25 William B Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

26 David Karger, Rajeev Motwani, and Madhu Sudan. Approximate graph coloring by semidefinite programming. *Journal of the ACM (JACM)*, 45(2):246–265, 1998.

27 David R Karger, Philip Klein, Cliff Stein, Mikkel Thorup, and Neal E Young. Rounding algorithms for a geometric embedding of minimum multiway cut. *Mathematics of Operations Research*, 29(3):436–461, 2004.

28 Howard Karloff and Uri Zwick. A 7/8-approximation algorithm for MAX 3SAT? In *Proceedings of the Foundations of Computer Science*, pages 406–415, 1997.

29 Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the Symposium on Theory of Computing*, pages 767–775, 2002.

30 Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for MAX-CUT and other 2-variable CSPs? *SIAM J. Comput.*, 37(1):319–357, 2007.

31 Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within $2-\varepsilon$. In *IEEE Conference on Computational Complexity*, pages 379–386, 2003.

32 Guy Kindler, Alexandra Kolla, and Luca Trevisan. Approximation of non-boolean 2CSP. In *Proceedings of the Symposium on Discrete Algorithms*, pages 1705–1714, 2016.

33 Alexandra Kolla, Konstantin Makarychev, and Yury Makarychev. How to play unique games against a semi-random adversary: Study of semi-random models of unique games. In *Proceedings of the Symposium on Foundations of Computer Science*, pages 443–452, 2011.

34 Robert Krauthgamer, Joseph Seffi Naor, and Roy Schwartz. Partitioning graphs into balanced components. In *Proceedings of the Symposium on Discrete Algorithms*, pages 942–949, 2009.

35 Michael Langberg, Yuval Rabani, and Chaitanya Swamy. Approximation algorithms for graph homomorphism problems. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 176–187. Springer, 2006.

36 Michael Lewin, Dror Livnat, and Uri Zwick. Improved rounding techniques for the MAX 2-SAT and MAX DI-CUT problems. In *Integer Programming and Combinatorial Optimization*, pages 67–82. Springer, 2002.

37 Anand Louis and Konstantin Makarychev. Approximation algorithm for sparsest k-partitioning. In *Proceedings of the Symposium on Discrete Algorithms*, pages 1244–1255, 2014.

38 Anand Louis and Yury Makarychev. Approximation Algorithms for Hypergraph Small Set Expansion and Small Set Vertex Expansion. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2014, September 4-6, 2014, Barcelona, Spain*, volume 28 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 339–355. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2014. doi:10.4230/LIPIcs.APPROX-RANDOM.2014.339.

39 Konstantin Makarychev and Yury Makarychev. How to play unique games on expanders. In *Approximation and Online Algorithms*, volume 6534 of *Lecture Notes in Computer Science*, pages 190–200. Springer Berlin / Heidelberg, 2011.

40 Konstantin Makarychev and Yury Makarychev. Approximation algorithm for non-Boolean Max k-CSP. *Theory of Computing*, 10(13):341–358, 2014.

41 Konstantin Makarychev and Yury Makarychev. Nonuniform graph partitioning with unrelated weights. In *Proceedings of the International Colloquium on Automata, Languages, and Programming*, pages 812–822, 2014.

42 Rajsekar Manokaran, Joseph Seffi Naor, Prasad Raghavendra, and Roy Schwartz. Sdp gaps and ugc hardness for multiway cut, 0-extension, and metric labeling. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 11–20, 2008.

43 Pasin Manurangsi, Preetum Nakkiran, and Luca Trevisan. Near-optimal UGC-hardness of approximating Max k-CSP$_r$. In *Proceedings of the Workshop on Approximation Algorithms for Combinatorial Optimization Problems (to appear)*, 2016.

44 Prasad Raghavendra. Optimal algorithms and inapproximability results for every CSP? In *Proceedings of the Symposium on Theory of Computing*, pages 245–254, 2008. doi:10.1145/1374376.1374414.

45 Prasad Raghavendra and David Steurer. Graph expansion and the unique games conjecture. In *Proceedings of the Symposium on Theory of Computing*, pages 755–764, 2010. doi:10.1145/1806689.1806792.

46 Alex Samorodnitsky and Luca Trevisan. A PCP characterization of NP with optimal amortized query complexity. In *Proceedings of the Symposium on Theory of Computing*, pages 191–199, 2000.

47 Ankit Sharma and Jan Vondrák. Multiway cut, pairwise realizable distributions, and descending thresholds. In *Proceedings of the Symposium on Theory of Computing*, pages 724–733, 2014.

48 Luca Trevisan. Approximation algorithms for unique games. In *Proceedings of the Symposium on Foundations of Computer Science*, pages 197–205, 2005.

49 Jiawei Zhang, Yinyu Ye, and Qiaoming Han. Improved approximations for Max set splitting and Max NAE SAT. *Discrete Applied Mathematics*, 142(1):133–149, 2004.

50 Uri Zwick. Finding almost-satisfying assignments. In *Proceedings of the Symposium on Theory of Computing*, pages 551–560, 1998.

51 Uri Zwick. Computer assisted proof of optimal approximability results. In *Proceedings of the Symposium on Discrete Algorithms*, pages 496–505, 2002.

Quantified Constraints in Twenty Seventeen

Barnaby Martin

School of Engineering and Computing Sciences, University of Durham, UK
barnaby.d.martin@durham.ac.uk

Abstract

I present a survey of recent advances in the algorithmic and computational complexity theory of non-Boolean Quantified Constraint Satisfaction Problems, incorporating some more modern research directions.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Quantified constraints, Constraint satisfaction problems, Computational complexity, Parameterized complexity, Universal algebra

Digital Object Identifier 10.4230/DFU.Vol7.15301.327

1 Introduction

The *Quantified Constraint Satisfaction Problem* (QCSP) might be thought of as the dissolute younger brother of its better-studied restriction, the *Constraint Satisfaction Problem* (CSP). The CSP has been called *Königsproblem*[1] as it sits at the interface between Combinatorics, Logic and Universal Algebra. The QCSP is a logical generalisation of the CSP whose combinatorial definition is ugly. Similarly, although the algebraic theory of the QCSP is useful, its algebraic objects are a bit unwieldy since surjective operations are not closed under composition. CSPs are ubiquitous in Computer Science, especially when one considers them in their infinite-domain generality, while QCSPs can not nearly claim to be so important in applications. This is no doubt due to modelling difficulties induced by the universal quantifier in the absence of disjunction, together with the lack of ability to relativise (guard) the universal quantifiers. The variant of QCSP in which both quantifiers may be relativised (for the CSP this is called the *list* or *conservative* case) has in fact been fully classified [7]. The only remaining QCSPs for which such relativisation is not desirable are the Boolean QCSPs. These are long since classified and Quantified Satisfiability itself, better-known as QBF (Quantified Boolean Formulas) – which is indeed useful – may be considered its own research area, and is therefore left out of the scope of this survey (see [75]).

In my papers on QCSP, I have often cited its use in planning [93] and modelling non-monotonic reasoning [56], but inspection of both these papers reveals this is just the Boolean case of QBF. There are various claims for the usefulness of non-Boolean QCSPs but specific explanations are sparse and even the examples given for QCSPs often involve more than just conjunction in the quantifier-free part (Example 1 in [89] is of this form). Thus, what is left of the true non-Boolean QCSP is a problem I believe to be mostly of interest to theorists, especially those who are not afraid of getting their hands dirty. For this is the lot of the researcher into complexity of QCSPs! However, in this quagmire there is still beauty to be found and interesting structured classifications are known often mixing curiously techniques combinatorial and algebraic. Indeed, the complexity researcher can even draw succour from

[1] King of problems, rather than problem of kings.

The Constraint Satisfaction Problem: Complexity and Approximability. *Dagstuhl Follow-Ups*, Volume 7, ISBN 978-3-95977-003-3.
Editors: Andrei Krokhin and Stanislav Živný; pp. 327–346
DAGSTUHL FOLLOW-UPS Dagstuhl Publishing
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Germany

the spurious claim that QCSPs are more important than CSPs since a classification for the former embeds that of the latter.

1.1 Previous Surveys

I know of two previous articles on general QCSPs that might be considered as surveys, both written by Hubie Chen [32, 36]. Neither is marketed as as a survey, [32] serves as an introduction to the topic and [36] is a more reflective piece from the viewpoint of the author. In this survey, I will try to be more comprehensive, within the scope, and at least in regard of recent work. However, this article is not intended to be introductory, and will not even contain definitions for all the concepts introduced.

After a section of background, this survey will have three principal sections, corresponding to what I see as the three largest research themes into non-Boolean QCSPs in the past decade. In Section 4, I will survey the state-of-the-art in classical complexity classifications for QCSP while in Section 5, I will discuss the recent work done in the area of their parameterized complexities. In Section 6, I will look at new algorithms for QCSPs and proof theories for their evaluation.

2 Preliminaries

In this article we tend toward the logical definitions for constraint satisfaction. A *constraint language* may therefore be seen as a relational first-order structure (possibly with constants). Relational structures are denoted $\mathcal{B}$ with domain B of cardinality $|B|$. Infinity might appear in two guises, either in the domain size or number of relations in the signature. When there is an infinite number of relations there is an issue as to how they are encoded, so we prefer to avoid this within computational problems. When we describe a constraint language as finite, we mean both in the domain size and the number of relations. We sometimes talk of a constraint language or structure (or CSP) *with constants* where we assume that all elements of the domain are named by their own constant.

The logic associated with CSP, the fragment of first-order containing just $\exists$, $\wedge$ and $=$, is usually known as *primitive positive* (pp) logic. The generalisation to QCSP involves the restoration of $\forall$ to primitive positive logic and appears in the literature, like the devil, under a myriad of names. In older mathematical logic texts it is known as *positive Horn* [74] while more modern works term it *quantified constraint formulas* [33], *quantified conjunctive-positive* [41], *quantified conjunctive* [20, 37], *conjunctive positive* [28] and even *few* [32] (this last is from *forall-exists-wedge*). The foundational [18] even leaves the logic unnamed. Since *positive conjunctive* is not among these names, it is the designation this article will use.

For a constraint language $\mathcal{B}$, the problem QCSP($\mathcal{B}$) takes as input a sentence ϕ of positive conjunctive logic and returns a yes-instance precisely when ϕ is true on $\mathcal{B}$, denoted $\mathcal{B} \models \phi$. The problem CSP($\mathcal{B}$) is defined similarly but for primitive positive logic. We sometimes also refer to the *(Q)CSP of* $\mathcal{B}$ as a disingenuous shorthand for (Q)CSP($\mathcal{B}$).

Note that, in fullest generality, the (so-called *uniform*) QCSP takes as input a pair $(\phi, \mathcal{B})$, where we may imagine finite $\mathcal{B}$ to be given by listing domain and tuples in relations, and asks whether $\mathcal{B} \models \phi$. The (non-uniform) problems QCSP($\mathcal{B}$) are examples of *right-hand* restrictions of QCSP, where *left-hand* restrictions involve limiting the form of the positive conjunctive sentence that may be input.

A quantifier block (sequence of quantified variables) is in Π_{2k} when it begins with universally quantified variables and alternates between the two quantifiers no more than

$2k-1$ times thereafter. Π_{2k}-CSP($\mathcal{B}$) is the restriction of QCSP($\mathcal{B}$) in which the positive conjunctive input is restricted to be prenex with quantifier block Π_{2k}.

A *homomorphism* from a structure $\mathcal{A}$ to a structure $\mathcal{B}$ in the same signature is a function f from A to B such that for each k-ary relation R, if $R(x_1,\ldots,x_k)$ holds in $\mathcal{A}$, then $R(f(x_1),\ldots,f(x_k))$ holds in $\mathcal{B}$. An *endomorphism* of $\mathcal{B}$ is a homomorphism from $\mathcal{B}$ to itself. A structure is a *core* when all of its endomorphisms are automorphisms. The CSP is well-known to be equivalent to the homomorphism problem, that given finite input $\mathcal{A}$ and $\mathcal{B}$, asks whether there is a homomorphism from $\mathcal{A}$ to $\mathcal{B}$? In this guise, it is apparent whence the names left- and right-hand restrictions, seen in the previous paragraph, arose. Let us dwell a little on what makes the CSP equivalent to the homomorphism problem, i.e. how we translate between primitive positive sentences and relational structures. This is through the juxtaposition of *canonical query* and *canonical database* [77], where one turns a prenex primitive positive sentence to a relational structure by mapping variables v_i to elements v_i and positive atoms $R(v_1,\ldots,v_k)$ in the conjunction to tuples $(v_1,\ldots,v_k)$ in a relation R.

An *algebra* is a first-order functional structure. Algebras are denoted $\mathbb{A}$ with domain A. If R is an m-ary relation over B and f is a k-ary operation on B, then we say f *preserves* R if, for any $(x_1^1,\ldots,x_1^m),\ldots,(x_k^1,\ldots,x_k^m)\in R$ we have also $(f(x_1^1,\ldots,x_k^1),\ldots,f(x_1^m,\ldots,x_k^m))\in R$. When f preserves R we also say that R is *invariant* under f and f is a *polymorphism* of R. A constraint language $\mathcal{B}$ is preserved by f if all of its relations are. Note that the unary polymorphisms of $\mathcal{B}$ are precisely its endomorphisms (whence the term polymorphism). Let Pol($\mathcal{B}$) be the set of polymorphisms of $\mathcal{B}$ and let Inv($\mathbb{A}$) be the set of relations on A which are invariant under (each of) the operations of $\mathbb{A}$. Pol($\mathcal{B}$) is an object known in Universal Algebra as a *clone*, which is a set of operations containing all projections and closed under composition (superposition). I will conflate sets of operations over the same domain and algebras just as I do sets of relations over the same domain and constraint languages (relational structures). Indeed, the only technical difference between such objects is the movement away from an ordered signature, which is not something we will ever need. Let $\langle\mathbb{A}\rangle$ be the clone generated by (the set of operations) of $\mathbb{A}$ and let $\mathrm{s}(\langle\mathbb{A}\rangle)$ be this clone restricted to its surjective operations. Finally, let sPol($\mathcal{B}$) be the set of surjective polymorphisms of $\mathcal{B}$ and let $\langle\mathcal{B}\rangle_{\mathrm{pp}}$ and $\langle\mathcal{B}\rangle_{\mathrm{pc}}$ be the class of relations definable on $\mathcal{B}$ in primitive positive and positive conjunctive logic, respectively.

The algebraic approach to CSP and QCSP is based on the following observations, each producing a *Galois Correspondence*. Let $\mathcal{B}$ be a constraint language and $\mathbb{A}$ an algebra, over the same finite domain. Then,

$$\begin{array}{llcll} \mathrm{Inv}(\mathrm{Pol}(\mathcal{B})) = \langle\mathcal{B}\rangle_{\mathrm{pp}} & [63,\ 14] & \& & \mathrm{Inv}(\mathrm{sPol}(\mathcal{B})) = \langle\mathcal{B}\rangle_{\mathrm{pc}} & [18] \\ \mathrm{Pol}(\mathrm{Inv}(\mathbb{A})) = \langle\mathbb{A}\rangle & [92] & \& & \mathrm{sPol}(\mathrm{Inv}(\mathbb{A})) = \mathrm{s}(\langle\mathbb{A}\rangle). & \end{array}$$

We consider a Galois Correspondence to be an order-inverting isomorphism between two lattices. This is exactly in line with the original correspondence of Evariste Galois, except that we will allow our lattices to be infinite. The order-inverting isomorphisms are given by Inv and Pol, for the CSP above left, between clones and sets of relations closed under primitive positive definability. For the QCSP above right, they are given by Inv and sPol, between the surjective reducts of clones and sets of relations closed under positive conjunctive definability (the second part of this correspondence, appearing on the bottom row above, is seldom noted in the literature). It is stated in [33] that a careful reading of the proof from [18] shows that conjunctive positive definability in $\langle\mathcal{B}\rangle_{\mathrm{pc}}$ may even be replaced with its Π_2 fragment. Thus, on finite structures, positive conjunctive logic collapses to its Π_2 fragment.

The consequence of the Galois Correspondence is that whenever $\mathrm{Pol}(\mathcal{B})\subseteq\mathrm{Pol}(\mathcal{B}')$ there is a logspace reduction from CSP($\mathcal{B}'$) to CSP($\mathcal{B}$). Note that $\mathcal{B}$ and $\mathcal{B}'$ must share the same domain. Similarly, when $\mathrm{sPol}(\mathcal{B}) \subseteq \mathrm{sPol}(\mathcal{B}')$ there is a logspace reduction from $\mathrm{QCSP}(\mathcal{B}')$ to $\mathrm{QCSP}(\mathcal{B})$. Thus, the *(surjective) polymorphisms control the complexity* of these problems.

I assume a basic familiarity with the modern theory of Computational Complexity. For more details on complexity classes, I refer the reader to [91] for (classical) Complexity Theory and [54] for Parameterized Complexity. Further, I will carelessly use *tractable* as a synonym for polynomially solvable. For finite $\mathcal{B}$, we note that the problems $\mathrm{CSP}(\mathcal{B})$, Π_{2k}-$\mathrm{CSP}(\mathcal{B})$ and $\mathrm{QCSP}(\mathcal{B})$ are in the complexity classes NP, Π^{P}_{2k} and Pspace, respectively.

2.1 More Algebra

Let $[n] := \{1, \ldots, n\}$. An operation f is called *idempotent* if, for each x, $f(x, \ldots, x) = x$. It is a *majority* if it is ternary and satisfies $f(x, x, y) = f(x, y, x) = f(y, x, x) = x$. It is *Mal'tsev* if it is ternary and satisfies $f(x, x, y) = f(y, x, x) = y$. On a totally ordered domain, the binary operations *min* and *max* return the minimum and maximum of their two arguments, respectively. A *semilattice* operation is a binary operation that is associative, commutative and idempotent. Both *min* and *max* are semilattice operations. For 2-*semilattice*, one relaxes associativity to the weaker condition $f(f(x, x), y) = f(x, f(x, y))$. A *unit* for a semilattice operation f is an element i so that $f(x, i) = x$ for all x. A k-ary operation f is called a *set operation* if $f(x_1, \ldots, x_k) = f(y_1, \ldots, y_k)$ whenever $\{x_1, \ldots, x_k\} = \{y_1, \ldots, y_k\}$. An algebra is called *idempotent trivial* if all of its idempotent operations are projections. A constraint language is called idempotent trivial if its polymorphism clone is.

Majority operations can be generalised to k-ary *near-unanimity* operations, which satisfy $f(x, \ldots, x, y) = f(x, \ldots, y, x) = \ldots = f(y, \ldots, x, x) = x$. This definition can now be relaxed for *weak near-unanimity* operations which are idempotent operations that only satisfy $f(x, \ldots, x, y) = f(x, \ldots, y, x) = \ldots = f(y, \ldots, x, x)$. Finally, a k-ary operation t is *Taylor* if it satisfies a system of identities $t(x^1_i, \ldots, t^k_i) = t(y^1_i, \ldots, y^k_i)$, for $i \in [k]$, in the variables x and y, where $x^i_i = x$ and $y^i_i = y$. One can see that weak near-unanimities are examples of Taylor operations.

For a finite-domain algebra $\mathbb{A}$ we associate a function $f_{\mathbb{A}} : \mathbb{N} \to \mathbb{N}$, giving the cardinality of the minimal generating sets of the sequence $\mathbb{A}, \mathbb{A}^2, \mathbb{A}^3, \ldots$ as $f(1), f(2), f(3), \ldots$, respectively. We may say $\mathbb{A}$ has the g-GP if $f(m) \leq g(m)$ for all m. The question then arises as to the growth rate of f and specifically regarding the behaviours constant, logarithmic, linear, polynomial and exponential. Wiegold proved in [98] that if $\mathbb{A}$ is a finite semigroup then $f_{\mathbb{A}}$ is either linear or exponential, with the former prevailing precisely when $\mathbb{A}$ is a monoid. This dichotomy classification may be seen as a gap theorem because no growth rates intermediate between linear and exponential may occur. We say $\mathbb{A}$ enjoys the *polynomially generated powers* property (PGP) if there exists a polynomial p so that $f_{\mathbb{A}} = O(p)$ and the *exponentially generated powers* property (EGP) if there exists a constant b so that $f_{\mathbb{A}} = \Omega(g)$ where $g(i) = b^i$.

For a finite-domain, idempotent algebra $\mathbb{A}$, *k-collapsibility* may be seen as a special form of the PGP in which the generating set for $\mathbb{A}^m$ is constituted of all tuples $(x_1, \ldots, x_m)$ in which at least $m - k$ of these elements are equal. *k-switchability* may be seen as another special form of the PGP in which the generating set for $\mathbb{A}^m$ is constituted of all tuples $(x_1, \ldots, x_m)$ in which there exists $a_i < \ldots < a_{k'}$, for $k' \leq k$, so that

$$(x_1, \ldots, x_m) = (x_1, \ldots, x_{a_1}, x_{a_1+1}, \ldots, x_{a_2}, x_{a_2+1}, \ldots, \ldots, x_{a'_k}, x_{a'_k+1}, \ldots, x_m),$$

where $x_1 = \ldots = x_{a_1-1}$, $x_{a_1} = \ldots = x_{a_2-1}$, $\ldots$, $x_{a_{k'}} = \ldots = x_{a_m}$. Thus, $a_1, a_2, \ldots, a_{k'}$ are the indices where the tuple switches value. Note that these are not the original definitions [33, 35] but they are provably equivalent [27]. We say that $\mathbb{A}$ is collapsible (switchable) if there exists k such that it is k-collapsible (k-switchable). For any finite algebra, k-collapsibility implies k-switchability and for any 2-element algebra, k-switchability implies k-collapsibility. Chen originally introduced switchability because he found a 3-element algebra that enjoyed the PGP but was not collapsible [35].

An alternative algebraic formulation of the CSP and QCSP has the form, for the latter, of QCSP($\mathbb{A}$) for some algebra $\mathbb{A}$. One might imagine a restriction here to surjective operations but it seems tractability may be found beyond this. The input to this problem is of the form $(\phi, \mathcal{B})$ where $\mathcal{B}$ is a finite constraint language which is invariant under $\mathbb{A}$. Note that $\mathcal{B}$ and $\mathbb{A}$ share the same domain and for fixed, finite $\mathbb{A}$, the problem to determine if input $\mathcal{B}$ is invariant under $\mathbb{A}$ is in polynomial time.

2.2 Various Digraphs

A *digraph* is a structure with a single binary relation E; if this is symmetric then it is further a *graph*. A *(self-)loop* on a vertex x is an instance of $(x, x) \in E$. A graph without self-loops is termed *irreflexive* and a graph with loops on every vertex is termed *reflexive*. Sometimes we term a graph *partially reflexive* to emphasise we are somewhere inbetween.

A *clique* is an irreflexive graph where all distinct vertices are adjacent. A *k-partite* graph is one whose vertices may be partitioned into k classes where there are no edges between vertices in the same class. If all other edges are present one refers to a *complete* k-partite (or *multipartite*) graph. A graph is a *tree* if it is connected and contains no cycles, and a *forest* is the disjoint union of trees. A *pseudotree* is a connected graph with at most one cycle, and a *pseudoforest* is the disjoint union of pseudotrees.

A *semicomplete* digraph is an irreflexive graph so that for distinct x, y at least one of $(x, y), (y, x) \in E$. A *tournament* further satisfies that precisely one of $(x, y), (y, x) \in E$. A *local tournament* satisfies, for every x and distinct y and z, such that either both $(x, y), (x, z) \in E$ or both $(y, x), (z, x) \in E$, that there be precisely one of the edges (y, z) or (z, y) in E. A *source* (respectively, *sink*) in a digraph is a vertex with in-degree (respectively, out-degree) zero. A digraph is *smooth* if it has neither source nor sink.

2.3 The Modern Study of CSPs

The foundational paper for studying the complexity of CSPs came in 1978 from Thomas Schaefer [95] in which he proved a P versus NP-complete dichotomy for Boolean CSPs. The classification has six tractable classes which we will give below alongside their associated polymorphism. The characterisation with polymorphisms appeared first in [70] which is the foundational paper for the algebraic approach to CSPs.

0-valid (constant 0)	Horn (min)	bijunctive (majority)
1-valid (constant 1)	dual Horn (max)	affine (Mal'tsev)

Note that there are unique operations on the Boolean domain that are majority and Mal'tsev, respectively. Obviously, this applies also to min and max once the order $0 < 1$ is assumed. The outstanding conjecture in the area of finite-domain CSPs was later formulated by Feder and Vardi in [58] and is that these are all either in P or are NP-complete, which is surprising given these CSPs appear to form a large microcosm of NP, and NP itself is thought unlikely to have this dichotomy property since the work of [79]. It seems Feder and Vardi tried very hard to reproduce an argument à la Ladner, in a logic that can express all finite-domain CSPs, yet failed. The original Feder-Vardi conjecture did not specify where the boundary between P and NP-complete should be, but this has now been concretely conjectured in the algebraic language [21]. This conjecture remains unsettled, although dichotomy is now known on substantial classes, e.g. structures with domains of size ≤ 3 [95, 22] and smooth digraphs [68, 5].[2]

The conjectured complexity delineation in [21] was that a finite-domain constraint language $\mathcal{B}$ that is expanded with all constants should be so that its CSP is NP-complete precisely if Pol($\mathcal{B}$) has a G-set as a factor. Remember that all polymorphisms of a constraint language expanded with its constants are idempotent. We will not define what it is for a G-set to be a factor since there are various more modern specifications that are more user-friendly for our purposes. For example, the condition for tractability is equivalent to possessing a Taylor operation [81] or a weak near-unanimity operation [84]. We note that the backward direction to this conjecture is known to be true [21].

3 Background

Schaefer announced a dichotomy theorem for Boolean QCSP in the same paper as he proved the dichotomy for Boolean CSP [95]. The proof was omitted and the result in any case could ostensibly only apply to the situation with constants (0 and 1) allowed in instances. The resolution of the Pspace-hard cases in the case where constants do not appear was finally given much later, independently in [47] and [48]. Schaefer was also quite vague about how to extend the polynomial algorithms for subclasses of Boolean CSP to the same subclasses of Boolean QCSP, and articles fleshing out these algorithms continued for a number of years: [72] (Horn and dual-Horn cases), [1] (bijunctive case).

The situation for non-Boolean QCSPs seems to have been taken up largely only in the new millennium with two communities working on the problem, one more applied, with a background in Constraint Programming [16, 15, 64, 60] and one more theoretical with a background in Algorithms and Complexity (and often Universal Algebra) [18, 33]. Both communities are united in their attempts to take established algorithmic methods for CSPs, such as *local consistency* and *linear equations* and adapt them for QCSPs. I belong to the theoretical community, and the main thrust of this survey will be in this direction.

Following Schaefer, and in line with the like program for CSPs, the bulk of the research into complexity of QCSPs in the theoretical community has been in the non-uniform, right-hand framework in which one parameterises the problem by the constraint language. We already noted that such a complexity taxonomy for QCSPs embeds the similar one for CSPs. This is because CSP($\mathcal{B}$) and QCSP($\mathcal{B} \uplus 1$) are polynomially equivalent for all $\mathcal{B}$, where $\mathcal{B} \uplus 1$ denotes the disjoint union of $\mathcal{B}$ with an isolated element. The algebraic approach to QCSP was pioneered in [17] whose expanded journal version (also with another author) appeared much later as [18]. This early work gave the Inv-sPol Galois Correspondence already mentioned and provided uniform explanations of QCSP tractability for classes of problems based on the presence of certain surjective (even idempotent) polymorphisms. This fruitful approach was continued in [33] where the key idea of *collapsibility* was introduced. Collapsibility was originally introduced as a (relational) property of constraint languages but was later explored as a property of idempotent algebras [33]. When a constraint language $\mathcal{B}$, expanded with all constants, is collapsible then the evaluations of instances ϕ of QCSP($\mathcal{B}$) may be reduced to a polynomial system of instances of CSP($\mathcal{B}$), and so the maximal complexity of the problem is

[2] Petar Markovic announced a proof of dichotomy for 4-element domains in 2011. The argument has since passed several years in confinement for refinement. It is in preparation as this goes to press. Very recently, Dmitriy Zhuk has announced a proof for the 5-element case (at AAA 91, February 2016, Brno).

reduced to NP [33]. If, further CSP($\mathcal{B}$) is in P, then this yields polynomial solvability for the QCSP.

Another manner of extension of ideas from the CSP to the QCSP arises in [39] where the notion of *establishing strong k-consistency*[3] is generalised for the QCSP. Establishing strong k-consistency is a well-known procedure for solving various CSPs, dating back to at least [53, 97] and is now known to be an algorithm for an even larger class of problems since its relationship with a certain pebble game was uncovered in [78]. It will be convenient to assume that our constraint language is closed under projections of its relations and further that our input ϕ is closed under projections of its aliquot atoms (thus, for example, if it contains an atom $R(v_1, \ldots, v_k)$ then it also contains atoms corresponding to each $\exists v_i R(v_1, \ldots, v_k)$). We say that ϕ is *k-consistent* (with respect to $\mathcal{B}$) if, for every assignment α of $k-1$ variables $x_1, \ldots, x_{k-1}$ from ϕ to elements from B that satisfy the conjuncts of ϕ that involve no other variables than $x_1, \ldots, x_{k-1}$, and for every variable x_k from ϕ, the assignment α can be extended to x_k such that all conjuncts of ϕ that involve no other variables than $x_1, \ldots, x_k$ are satisfied on $\mathcal{B}$ by the extension of α. We say that ϕ is *strongly k-consistent* if ϕ is j-consistent for all j with $2 \leq j \leq k$. We would say that ϕ is *globally consistent* if ϕ is k-consistent for all $k > 0$.

When strong k-consistency implies global consistency, then we have that establishing strong k-consistency will be an algorithm for the CSP. Further, establishing strong-k-consistency is an algorithm for the CSP for a large class of constraint languages including those preserved by near-unanimity [71] and set [50] polymorphisms.

In [39], a suitable pebble game is found for the variant of establishing k-consistency associated with the QCSP and this gives positive algorithms for left-hand restrictions where a generalisation of treewidth comes into play (we will return to this line of enquiry later). Returning to the land of dexterity, establishing k-consistency is shown to be a polynomial algorithm for QCSP($\mathcal{B}$) when $\mathcal{B}$ is preserved by a near-unanimity operation. Something stronger than establishing [strong] k-consistency is termed establishing *default* k-consistency [39] and this is shown to give a polynomial algorithm for QCSP($\mathcal{B}$) when $\mathcal{B}$ is preserved by an idempotent set operation f that has an additional property termed *unique minimal coherent set* with respect to f [39, 31]. As it happens, both of these classes of polymorphism bestow collapsibility and so the polynomial algorithm is implied from [33], together with the corresponding works on the CSP [50]. However, the direct algorithm explained through the pebble game gives a simple and unifying description of these tractabilities.

3.1 Constants and Idempotency

The complexity classification problem for finite-domain CSPs is greatly facilitated by the fact that one may assume that all constants are primitive positive definable, up to isomorphism. This is due to the key notion of cores (recall these are structures for which all endomorphisms are automorphisms). Every finite-domain constraint language is homomorphically equivalent to a constraint language that is a core and in this core all constants are primitive positive definable up to isomorphism. In the algebraic language this corresponds to the assumption, without loss of generality, that all polymorphisms are idempotent. A robust notion of core for positive conjunctive logic and QCSP is not exactly known. The putative notion Q-core of [42] has been successfully deployed in complexity classifications [85, 83], as we shall see later; however, many of its properties (even "uniqueness") are not yet clear. Thus, it is not known

[3] The authors of [39] call this establishing k-consistency but I align with, inter alia, [78].

if the QCSP algebraic classification may be reduced, from sets of surjective operations, to idempotent clones. Certainly, a constraint language, agreeing on all positive conjunctive sentences, with constants up to isomorphism positive conjunctive definable, is not always possible [42]. As a result of this, a certain amount of the literature on QCSP classifications is based on the idempotent assumption and consequently only deals with constraint languages already expanded with constants. This is true for many of the papers in which algebra plays a central role, for example, [33, 27]. Even in the foundational [18], all of the polymorphisms giving tractability are idempotent (if the idempotent reduct of sPol($\mathcal{B}$) is tractable for QCSP then clearly so is sPol($\mathcal{B}$) itself). Thus, although the results of [18] apply where sPol($\mathcal{B}$) need not contain only idempotent operations, the work itself may be said to focus on the idempotent. Several works involving combinatorial results operate in the wilderness outside of idempotency [86, 85, 83] but the only totally algebraic paper in this *terra incognita* (sic!) is [44]. In this last work, the object of study is QCSP($\mathbb{A}$) where $\mathbb{A}$ is a monoid. It follows from collapsibility that this problem is always in NP and the authors give a P versus NP-complete dichotomy with the condition for tractability being that $\mathbb{A}$ is a *block group* that is generated by its *regular* elements. The condition for tractability of CSP($\mathbb{A}$), even in the more general situation in which $\mathbb{A}$ is a semigroup, is just $\mathbb{A}$ being a block group [23]. The really interesting thing about the QCSP classification here is an explanation of tractability through a binary monoid operation f that may not be idempotent. Curiously, the tractability comes from a term operation generated by f that may not even be surjective (though f itself is). However, when this term operation is restricted to the so-called *idempotents* of $\mathbb{A}$ it is surjective (indeed, idempotent).

4 Classical Complexity

The modern sport for complexity classifications for QCSP was apparent already in [18]. In this paper the new algebraic methods were leveraged to obtain the first trichotomy for QCSP. A binary relation is a *graph of a permutation* if it is the set of pairs $\{(i, \pi(i)) : i \in [n]\}$ for some permutation π. Let $\mathcal{B}$ be a constraint language on $[n]$ that contains all $(n!)$ graphs of permutations. Then QCSP($\mathcal{B}$) is either in P, is NP-complete or is Pspace-complete (Theorem 7.4 in [18] deals with $n \geq 3$, the remainder come from Schaefer [95]). The polynomial cases possess either a majority or Mal'tsev polymorphism. For NP membership of the NP-complete cases, an early form of "collapsibility" appears as Proposition 7.1. Then, for the remaining cases of NP-hardness, the classification for CSP from [49] is cited.

The next QCSP classification seems to have come in [30]. 2-semilattice polymorphisms r are sufficient to make the CSP tractable [24] but for QCSP(Inv(r)), Chen observed two types of behaviours, namely being in P and being co-NP-hard [30]. The separating criterion has to do with the number of strongly connected minimal components in the graph given by $E(x, y)$ iff $r(a, b) = a$. When this number of minimal components is one (it is unique) the problem is in P, otherwise it is co-NP-hard. The situation for semilattice operations s is more fully understood. QCSP(Inv(s)) is in P, if f has a unit, else QCSP(Inv(s)) is Pspace-complete (the Pspace-hardness in this case seems to be due to Bulatov in [18] and did not appear in any previous version of that paper).

A largely combinatorial approach to QCSP complexity was followed in the works [86, 85, 83], deriving various classifications, although the algebraic method was used for tractability arguments in the latter two. These papers were an amusing diversion for their authors but do not necessarily shed much light on how one might argue for complexity classifications in general. The graphs considered in these papers are all instances of partially reflexive pseudoforests, for which a complexity classification for the Retraction problem is known. The Retraction problem may be seen as the CSP in which all constants are available (the CSP classification for partially reflexive pseudoforests without constants is nearly trivial).

In [85] a dichotomy is given for QCSP($\mathcal{H}$) where $\mathcal{H}$ is a partially reflexive forest. In the case of partially reflexive forests, this is between NL and NP-hard, whereas for partially reflexive paths it is between NL and Pspace-complete. A key idea here is *loop-connectedness*, which asks whether the subgraph induced by the self-loops is connected. This plays an important role in the classification of the Retraction problem for (partially reflexive) pseudoforests in [59]. In [85] it is noted that loop-connectivity of a partially reflexive tree $\mathcal{H}$ is a sufficient criterion for NL membership of QCSP($\mathcal{H}$), and this is witnessed by a majority polymorphism (thus even when $\mathcal{H}$ is expanded with constants the problem remains in NL). However, it is not a necessary condition and a simple example is furnished by the undirected path on five vertices, $\mathcal{P}_{10100}$, with the middle vertex as well as one end a loop. This is clearly not loop-connected, yet QCSP($\mathcal{P}_{10100}$) is in NL. Indeed, QCSP($\mathcal{P}_{10100}$) coincides with QCSP($\mathcal{P}_{100}$), where $\mathcal{P}_{100}$ is the undirected path on three vertices with a loop at one end. Clearly, $\mathcal{P}_{100}$ is loop-connected, and so the result follows. It is here where the notion of Q-core explored in [42] is useful, because the Q-core of $\mathcal{P}_{10100}$ is $\mathcal{P}_{100}$. Indeed, when $\mathcal{H}$ is a partially reflexive tree that is a Q-core all of the known classifications for the QCSP (from [86, 85, 83]) are consistent, in the sense of P versus NP-hard, with the classification for the Retraction problem in [59]. Note that the jump from pseudotrees to pseudoforests creates a disconnected graph which causes a collapse in QCSP complexity to NP; and even further to NL, in the case that the graph has a loop or is bipartite.

The QCSP complexity classification for irreflexive pseudotrees is given in [86] (one can easily infer the result for irreflexive pseudoforests). Finally, the classification for partially reflexive cycles is given in [83]. An interesting observation appearing in this classification is the identification of a QCSP tractable graph $\mathcal{C}_{1110}$, the undirected 4-cycle with three self-loops, without a majority polymorphism. Indeed, $\mathcal{C}_{1110}$ is even a Q-core. This is surprising because all tractable cases from the classification for partially reflexive forests possess these majorities in their Q-core.

In [27], a QCSP classification is given for the case of partially reflexive paths expanded with constants. Here, the distinction between NL and NP-hard perfectly follows the classification of [59], but now the case NP-complete becomes possible (some NL cases become NP-complete, for example, this is the case for $\mathcal{P}_{10100}$ expanded with constants). Thus, we reach here a classification which is a trichotomy between NL, NP-complete and Pspace-complete, where we had a dichotomy without constants between NL and Pspace-complete. Note that once we expand with constants we have a constraint language that is both a core and a Q-core.

In [51], a more advanced marriage of combinatorics and algebra was attempted, and for the first-time this mixed approach gave quite sophisticated algebraically proven lower bounds. Semicomplete digraphs are cores so one may assume without affecting QCSP complexity that they are expanded with constants naming their vertices and that their polymorphisms are all idempotent. The complexity classification for QCSP($\mathcal{H}$), where $\mathcal{H}$ is a semicomplete digraph, given in [51] is rooted in the long proof that smooth semicompletes with more than one cycle are idempotent trivial. It follows that QCSP($\mathcal{H}$) is Pspace-complete in this case [18]. Note that the only smooth semicompletes with no more than one cycle are the directed 2- and 3-cycles. It turns out QCSP($\mathcal{H}$) is also Pspace-complete when $\mathcal{H}$ is a smooth semicomplete with more than one cycle with a sequence of sinks added (respectively, sources added). The remainder of the classification is simple, if $\mathcal{H}$ has both a source and a sink, then QCSP($\mathcal{H}$) is in NP; and if $\mathcal{H}$ is the directed 2- or 3-cycle with a sequence of sinks (respectively, sources) added, then QCSP($\mathcal{H}$) is in NL. The NP-complete cases (source and sink plus more than one cycle) can be inferred from the classification for the CSP [4].

4.1 Bounded Alternation

An interesting restriction of the QCSP addresses the case where one allows only bounded alternation of the quantifiers in a (prenex) input instance. An interesting application for the Π_2-CSP is given in Section 3 of [6] (but note the universal quantifiers are relativised). The situation for Boolean constraint languages parallels the QCSP, with the problems Π_{2k}-CSP($\mathcal{B}$) being Π_{2k}^{P}-complete precisely when QCSP($\mathcal{B}$) is Pspace-complete [69].

Bounded alternation reappeared in the theoretical study of the QCSP in the remarkable paper [34] where it was noted that for certain constraint languages $\mathcal{B}$, Π_{2k}-CSP($\mathcal{B}$) is in co-NP, for each k. Indeed, an example of co-NP-completeness may be given by some constraint languages invariant under a semilattice operation without unit [34, 30]. In fact, if f is a semilattice operation, then either Π_{2k}-CSP(Inv(f)) is in P, if f has a unit, or otherwise Π_{2k}-CSP(Inv(f)) is co-NP-complete. Let us recall the dichotomy for the QCSP in this situation, already mentioned, that QCSP(Inv(f)) is in P, if f has a unit, else QCSP(Inv(f)) is Pspace-complete [18]. So, it seems for bounded alternation QCSP, there are three complexity regimes above P: NP-complete, co-NP-complete and maximal complexity (Π_{2k}-complete), where for the QCSP there are only NP-complete and Pspace-complete.

Chen proved in [35] that a computationally effective form of PGP is sufficient to place Π_{2k}-CSP($\mathbb{A}$) in NP. Subsequently, Zhuk has proved that switchability is the only type of PGP for finite algebras [101], thus all forms of PGP associated with finite-domain Π_{2k}-CSP and QCSP are computationally effective.

4.2 Counting Quantifiers

Quite recently has appeared, in the context of the CSP, the study of counting quantifiers of the form $\exists^{\geq j}$ [87]. These quantifiers allow one to assert the existence of at least j elements such that the ensuing property holds, so on a structure $\mathcal{B}$ with domain of size $|B|$, the quantifiers $\exists^{\geq 1}$ and $\exists^{\geq |B|}$ are precisely $\exists$ and $\forall$, respectively. Thus one can study variants of CSP($\mathcal{B}$) in which the input sentence to be evaluated on $\mathcal{B}$ remains positive conjunctive in its quantifier-free part, but is quantified by various counting quantifiers from some non-empty set. For $X \subseteq \{1, \ldots, |B|\}$, $X \neq \emptyset$, the X-CSP($\mathcal{B}$) takes as input a sentence given by a conjunction of atoms quantified by quantifiers of the form $\exists^{\geq j}$ for $j \in X$. It then asks whether this sentence is true on $\mathcal{B}$. In this fashion, the X-CSP may be seen as a natural generalisation of the CSP and QCSP.

In [87] a panoply of classifications is given for X-CSP($\mathcal{B}$), mostly for the situation where $\mathcal{B}$ is some kind of graph. In general, as with the QCSP, complexities of the form P, NP-complete and Pspace-complete are readily available and classifications are either trichotomies or dichotomies between P and NP-hard. Interestingly, when one has access to all the quantifiers $\exists^{\geq 1}, \ldots, \exists^{\geq |B|}$ the intermediate complexity NP-complete seems to disappear (at least I do not know a case of it). The problems X-CSP($\mathcal{H}$) are completely characterised, for any $X \subseteq [|H|]$, when $\mathcal{H}$ is an undirected clique or cycle, into the classes P, NP-complete and Pspace-complete. Then the problem $\{1, 2\}$-CSP($\mathcal{H}$) is considered where $\mathcal{H}$ is an undirected graph, something of a companion to the theorem of Hell and Nešetřil that these CSPs are in P if $\mathcal{H}$ is bipartite and are NP-complete otherwise. The authors show that $\{1, 2\}$-CSP($\mathcal{H}$) is in P if $\mathcal{H}$ is a forest or a bipartite graph with a 4-cycle, and is NP-hard otherwise. For bipartite graphs $\mathcal{H}$ that are neither forest nor contain a 4-cycle it is even shown that $\{1, 2\}$-CSP($\mathcal{H}$) is Pspace-complete.

Finally, a trichotomy theorem is shown for $\{1,2\}$-CSP($\mathcal{H}$) when $\mathcal{H}$ is a complete multipartite graph, with such problems being either in L, NP-complete or Pspace-complete.

The algebraic method, so potent in understanding the complexity of CSPs and QCSPs has recently been tailored to counting quantifiers [25]. The algebraic objects may be seen as a kind of *expanding* polymorphism. Call an operation $f : B^k \to B$ j-*expanding* if, for all $X_1, \ldots, X_k \subseteq B$ such that $|X_1| = \ldots = |X_k| = j$, we have $|f(X_1, \ldots, X_k)| \geq j$. This condition at $j = 1$ is trivial (it says that f is a function) and at $j = |B|$ asserts surjectivity. For $X \subseteq \{1, \ldots, |B|\}$, we say that f is X-*expanding* if it is j-expanding for all $j \in X$. Now, the relations that are X-pp-definable over $\mathcal{B}$ are exactly those that are preserved by the X-expanding polymorphisms of $\mathcal{B}$. In the case of $\{1\}$-pp and $\{1,|B|\}$-pp, this includes the Galois Correspondences we have already met.

Applying this algebraic theory, as was done with the QCSP, à la [18], would be splendid but a number of the arguments seem to fail for expanding polymorphisms (though Mal'tsev seems to go through).

4.3 Infinite Domains

The study of QCSP($\mathcal{B}$), for infinite-domain $\mathcal{B}$, is not as advanced as the like program for CSPs, and was pioneered in [8]. The authors gave an L, NP-complete, co-NP-hard trichotomy for *equality languages*, which are those constraint languages that admit a first-order definition in $(\mathbb{Q}; =)$.

In the modern systematic study of infinite-domain CSPs, the first classification following equality languages was that for *temporal languages* [10], that admit a first-order definition in $(\mathbb{Q}; <)$, and it seems the major thrust in infinite-domain QCSPs today is in this class. Note that $(\mathbb{Q}; =)$ and $(\mathbb{Q}; <)$ both admit quantifier elimination and all first-order definable relations are already quantifier-free definable, say in conjunctive normal form (CNF). For the classification of [8] a key role is played by *negative* and *positive* languages. The latter do not permit negation of any form while the former is broadly the opposite, allowing only disequalities in CNFs except for singleton clauses (conjuncts) of equality. When $\mathcal{B}$ is an equality language, QCSP($\mathcal{B}$) is in L if $\mathcal{B}$ is negative, QCSP($\mathcal{B}$) is NP-complete if $\mathcal{B}$ positive but not negative[4], and co-NP-hard otherwise.

In the papers [29, 28], a classification for QCSPs is given for the positive temporal languages, that is with a positive definition in $(\mathbb{Q}; \leq)$, where the authors show that these QCSPs are either in L, NL-complete, P-complete, NP-complete or Pspace-complete. Each of these cases is both algebraically and syntactically characterised. In the history of temporal CSPs, the fragment *Ord-Horn* played a key role [90]. In [46] a subclass known as *Guarded Ord-Horn* is established as tractable for QCSP. The significance of this class is noted in [100] where it is shown that Guarded Ord-Horn languages are the only tractable case within the *dually-closed* (if language $\mathcal{B}$ over numeric domain pp-defines k-ary R then it also pp-defines $\{(-x_1, \ldots, -x_k) : (x_1, \ldots, x_k) \in R\}$) Ord-Horn languages. That is, when $\mathcal{B}$ is a dually-closed temporal language that is not Guarded Ord-Horn, then QCSP($\mathcal{B}$) is co-NP-hard [100].

In [9] it is shown that temporal constraint languages with polymorphism *min* (also *max*) and *mx* (also its dual) have a tractable QCSP. I will leave the polymorphism *mx* undefined but suffice it to say that it plays an important role in the temporal CSP classification [11].

Counting quantifiers have also been taken to the domain of the equality languages in [88] with various classifications given. Here it is appropriate to also consider quantifiers of the

[4] A relation is both negative and positive when it is just a conjunction of equalities.

form $\forall^{>j}$, meaning that the associated binding holds on all but at most j elements of the domain.

5 Parameterized Complexity

There has been a plethora of papers in recent years devoted to the parameterized complexity of model-checking classes of structures in various fragments of first-order logic. QCSP($\mathcal{B}$) is nothing other than the model-checking problem for positive conjunctive logic on the singleton class $\{\mathcal{B}\}$ and so it is unsurprising that this new line of research impinges upon its study.

The paper [20] considers the positive conjunctive model-checking problem for *posets* where the sentence and poset are both input but where the poset is restricted to come from a certain class. A preliminary result shows a concrete, four-element poset for which the QCSP is NP-hard (most likely one could prove this is Pspace-complete). The main result then is that model-checking positive conjunctive logic on bounded width posets is FPT, where the parameter is the sentence size. In fact, they prove a stronger result where the class of posets is unrestricted, that is model-checking positive conjunctive logic on posets is FPT, where the parameter is the sentence size plus the poset width. The paper is a companion to one proving a result similar to the first but for existential logic [19]. This line of enquiry was pursued by other authors culminating in the demonstration that model-checking first-order logic on posets is FPT when the parameter is the sentence size plus the poset width [62]. Parameterized intractability also features in [20], where the model-checking problem for positive conjunctive logic on posets of bounded depth and bounded cover-degree is shown to be co-W[2]-hard.

Another research direction has to do with left-hand restrictions, where the class from which the input sentence comes is restricted. In this area results usually may appear in both classical and parameterized flavours since they appear as gaps between P and not FPT (assuming W[1] $=$ FPT). The first outstanding result in this area is due to Grohe [66], in which it is noted, when relational arity is bounded, that if the restricted class of primitive positive sentences does not possess *bounded tree-width*, the model-checking problem for this class is not FPT, unless W[1] $=$ FPT. In this line of research the parameter is always the size of the sentence. The converse, that bounded-treewidth yields tractability for this model-checking problem was already known [61], so this result gives a typical type of classification theorem, based on a complexity-theoretic assumption. It is not immediately apparent what the graph-theoretic property of bounded-treewidth means for a class of sentences, yet the translation between primitive positive sentences and relational structures through canonical query and canonical database has been discussed. It is similarly possible to consider such measures for other classes of arity-bounded first-order formulas, and this line of thought was pursued by other authors. Bounded treewidth alone does not guarantee tractability for the model-checking problem for positive conjunctive logic unless both the relational arity and constraint language size are bound [65].

A precursor of Grohe's result was furnished in [67] where only classes of sentences satisfying a certain closure property (broadly speaking that of isomorphism, when the sentence is viewed as a graph) were considered. This situation was generalised for positive conjunctive logic in [40] and then to full first-order logic in [38]. The key notion in these works, and the closure condition alluded to, is *graphical closure*. A class of sentences is graphically closed if it satisfies two types of closure, the first syntactic and the second graphical. The syntactic closure deals with types of logical equivalence including de Morgans laws, associativity, commutativity and distributivity, together with some rules when a bound variable does not a appear free in a literal in a conjunct or disjunct. The graphical closure deals with substitutions of relation symbols in atoms, such that if, e.g., E and F are ternary relation symbols in the signature, then any sentence containing the atoms $E(x, y, x)$ may have this substituted by $F(x, y, x)$ (di-graphical might be more appropriate here, since the order and multiplicity of x and y matter). The main result of [38] uses a relative of tree-width to discern for which graphically closed sets of sentences Φ of bounded arity model-checking first-order logic is FPT. Specifically, if Φ satisfies a condition known as *bounded thickness* then the model-checking problem is in P and therefore FPT; otherwise it is not (assuming FPT $\neq W[1]$). The restriction of bounded thickness to positive conjunctive sentences is *elimination width*, for which the gap between P and not FPT was proved in [40]. The polynomial result of that paper subsumes the earlier polynomial result of [39], where the notion of treewidth was differently generalised for positive conjunctive logic.

Finally, a true analog of Grohe's Theorem, for positive conjunctive logic, not making the assumption of graphical closure, has been given in [43].

A further investigation into parameterized complexity comes in the very recent [57] where a new parameter *prefix pathwidth* is introduced for QBF. Atserias and Oliva [2] had previously shown that, in contrast to SAT, many of the well-known decompositional parameters (such as treewidth and pathwidth) do not reduce the complexity of QBF. The main reason for this appears to be a blindness of these parameters towards the quantifier dependencies between variables of a QBF formula. Prefix pathwidth mitigates some of these difficulties and it is proved that QBF is FPT with respect to this parameter (and the width of the dependency poset). The result directly applies also to QCSP with any bounded domain size and hence has been eligble for inclusion in this survey.

6 Proof Theory and Evaluation

The canonical proof system for propositional formulas in CNF is *Resolution* [52, 94] in which one tries to prove a system of clauses is contradictory. This, therefore, gives the proof theory for SAT. Resolution has been extended for QBF to the popular system of *Q-Resolution* [76]. Q-Resolution, being a system for QBF, is outside the scope of this survey, but in [37] Chen identifies two of its weaknesses as being the restriction to the Boolean domain as well as requiring the input sentence to be in prenex form. In his *QCSP proof system* (glorious in its anonymity) he overcomes these shortcomings. Egly [55] had previously proposed a proof system for non-prenex QBF, but Chen's system appears to be the first to allow for the possibility of a non-Boolean domain. The width notion of this proof system is associated with the notion of *k-judge-consistency* which implies an earlier notion of consistency which Chen and Dalmau used to demonstrate algorithmic tractability in [39].

Recall the positive conjunctive sentence width notion arising from [38] (cf. Section 5) was elimination width. We now designate the *Q-width* of a positive conjunctive sentence to be the maximum of its elimination width and any arity of a relation appearing within. Chen's work [37] has an algorithmic side-effect, giving a simple generic polynomial method for deciding the (uniform) instances $(\phi, \mathcal{B})$ of QCSP where we assume ϕ has Q-width bounded by some constant k. Of course the polynomiality of this has long since been known but relies on such non-trivial-but-tractable devices as the computing of tree decompositions. The positive algorithmic result from [37] extends that from the earlier [39] in two important ways: firstly, there is no prenex assumption; and, secondly, the notion of Q-width is more general than the generalised treewidth. We also already mentioned that k-judge consistency implies the earlier notion of consistency, which can be seen as a further generality of [37] over [39], though this allows the condition in the former to activate the algorithm in the latter. It might also be said that in [37] a bird's eye view of the landscape is obtained through a marriage between the proof system and its associated algorithm.

7 Future Prospects

In [36], Chen made a number of natural conjectures regarding QCSPs, typically concerning idempotent algebras $\mathbb{A}$. My favourite speculated that QCSP($\mathbb{A}$) should be in NP if $\mathbb{A}$ has the PGP and is otherwise Pspace-complete (see Conjectures 5 and 6 in [36]). Dmitriy Zhuk has settled the backward direction by proving that the only form of PGP for finite-domain algebras is switchability [101]. I suspect there is some tight relationship between NP-membership for the QCSP, and PGP in the associated algebra, which will soon find expression.

A structure is *ω-categorical* if it is up to isomorphism the only countably infinite model of its first-order theory. The Galois Correspondence Inv-Pol is known to hold for ω-categorical constraint languages [12] and has been instrumental in a number of recent CSP classifications (e.g. [11, 13]). For the operational side one needs to insist the clones satisfy a certain topological (local) closure [96]. It is not known whether $\mathrm{Inv}(\mathrm{sPol}(\mathcal{B})) = \langle\mathcal{B}\rangle_{\mathrm{pc}}$ for ω-categorical $\mathcal{B}$. The best general result in this direction involves the *periomorphisms* of [45] which work on periodic elements in $\mathcal{B}^\omega$, which have the form $(b_1, \ldots, b_k)^\omega$. The periodic elements induce a countable substructure $\mathcal{B}^{\mathrm{per}}$ in $\mathcal{B}^\omega$ and a periomorphism is a homomorphism from this structure to $\mathcal{B}$. The fact that $\mathcal{B}^{\mathrm{per}}$ is countable permits the use of a standard back-and-forth argument whence it is shown that if a relation is invariant under the periomorphisms of ω-categorical $\mathcal{B}$, then indeed it is positive conjunctive definable. The correspondence $\mathrm{Inv}(\mathrm{sPol}(\mathcal{B})) = \langle\mathcal{B}\rangle_{\mathrm{pc}}$ is known, a posteriori, when $\mathcal{B}$ is an equality language [8]. For languages first-order definable in $(\mathbb{Q}; <)$ it is still in general open.

Meanwhile, let us leave the unfinished classification for temporal QCSPs to ponder that for equality languages. In the conference version of [8] the trichotomy was announced to be between L, NP-complete and Pspace-complete but the greater lower bound was reduced to co-NP-hard in the journal version. The culprit is the erroneous supporting Theorem 4.1, for which one can construct a counterexample, and from which the missing Galois Correspondence of the previous paragraph would have followed (Theorem 4.1 holds for infinite direct products). The major open question from the journal version is whether QCSP($\mathbb{Q}; x = y \rightarrow y = z$), known to be co-NP-hard, is in fact Pspace-complete. Were this to be Pspace-complete, it would complete the promotion of all the outstanding co-NP-hard cases to Pspace-complete. However, were it to be in co-NP, for example, there would remain additional work to be done, not to mention that the trichotomy would become a tetrachotomy, since many cases, including QCSP($\mathbb{Q}; x = y \rightarrow u = v$), are known to be Pspace-complete.

It would be interesting to unite the results of [86, 85, 83] into a QCSP classification for partially reflexive pseudoforests with the classes likely to be NL, NP-complete and Pspace-complete. Even a partial classification into NL and NP-hard might require a patience and diligence that could remain unrewarded by the result. For partially reflexive pseudoforests that are Q-cores, most likely the NL/ NP-hard boundary follows that for Retraction [59].

A number of open questions arise regarding CSP with counting quantifiers and the most interesting relate to potential applications of the new algebraic theory. A more combinatorial question is as to the precise complexity of $\{2\}$-CSP($\mathcal{K}_4$), where $\mathcal{K}_4$ is the 4-clique, which is known to be in P but not in L or NL. This question is interesting as it is the only case in the classification of X-CSP($\mathcal{H}$) in [87], where $\mathcal{H}$ is an undirected clique or cycle, that is known to be in P but not L.

The question of the idempotent remains a thorn in our side. For every finite $\mathcal{B}$, is there a finite $\mathcal{C}$ so that, say, QCSP($\mathcal{B}$) and QCSP($\mathcal{C}$) are polynomially equivalent, and all constants are positive conjunctive definable in $\mathcal{C}$ up to isomorphism? We know this is not true if we strengthen polynomial equivalence to positive conjunctive equivalent [42].

The QCSP program initiated in [51] continues a fascinating combinatorial-algebraic program itself well-established, e.g. in [80, 26, 3]. Bandelt has classified both which reflexive and which bipartite graphs admit a majority polymorphism (see [3]). Indeed, the distinction between majority and not is established for partially reflexive trees in [85]. In a similar vein, in [26] it is established precisely which digraphs have a Mal'tsev polymorphism. Curiously, Malt'sev digraphs also have a majority [73]. The business of [51] is more in line with several investigations of Benoit Larose into idempotent triviality (see [80]). Other recent work has focused on whether certain constraint languages for which we know the CSP classification follow also the conjectured algebraic classification (which indeed they do). MacGillivray and Swarts [82] prove that the (irreflexive) locally semicomplete digraphs whose CSPs are tractable are exactly those that admit a weak near-unanimity polymorphism, and Wires has proved [99] that the partially reflexive tournaments whose CSP with constants is tractable are exactly those that admit a Taylor polymorphism. Recall that a finite idempotent algebra generates a weak near-unanimity operation iff it generates a Taylor operation. It would be fun to establish which irreflexive locally semicomplete and partially reflexive tournaments are idempotent trivial, with a view to leveraging this knowledge towards a QCSP classification, of the kind in [51].

The outstanding question left open in parameterized complexity in the region of QCSP, though somewhat superseding it, is to unify the works [38] add [43]. That is, to give a theorem à la Grohe, as the latter, for full first-order logic.

Acknowledgements. I am grateful for helpful discussions with a number of people and especially corrections to an early draft from Hubie Chen and Michał Wrona. I am also grateful for corrections from an anonymous referee appointed by the editors.

References

1 Bengt Aspvall, Michael F. Plass, and Robert Endre Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Inf. Process. Lett.*, 8(3):121–123, 1979.

2 Albert Atserias and Sergi Oliva. Bounded-width QBF is PSPACE-complete. *J. Comput. Syst. Sci.*, 80(7):1415–1429, 2014.

3 Hans-Jürgen Bandelt. Graphs with edge-preserving majority functions. *Discrete Mathematics*, 103(1):1–5, 1992.

4 Jørgen Bang-Jensen, Pavol Hell, and Gary MacGillivray. The complexity of colouring by semicomplete digraphs. *SIAM J. Discrete Math.*, 1(3):281–298, 1988.

5 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 Journal on Computing*, 38(5):1782–1802, 2009.

6 Frédéric Benhamou and Frédéric Goualard. *Principles and Practice of Constraint Programming – CP 2000: 6th International Conference, CP 2000 Singapore, September 18–21, 2000 Proceedings*, chapter Universally Quantified Interval Constraints, pages 67–82. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

7 Manuel Bodirsky and Hubie Chen. Relatively quantified constraint satisfaction. *Constraints*, 14(1):3–15, 2009.

8 Manuel Bodirsky and Hubie Chen. Quantified equality constraints. *SIAM J. Comput.*, 39(8):3682–3699, 2010. Conference version appeared at LICS 2007.

9 Manuel Bodirsky, Hubie Chen, and Michal Wrona. Tractability of quantified temporal constraints to the max. *IJAC*, 24(8):1141–1156, 2014.

10 Manuel Bodirsky and Jan Kára. The complexity of temporal constraint satisfaction problems. In *Proceedings of STOC'08*, pages 29–38, 2008. Accepted for publication in J. ACM.

11 Manuel Bodirsky and Jan Kára. The complexity of temporal constraint satisfaction problems. *J. ACM*, 57(2), 2010.

12 Manuel Bodirsky and Jaroslav Nešetřil. Constraint satisfaction with countable homogeneous templates. *Journal of Logic and Computation*, 16(3):359–373, 2006.

13 Manuel Bodirsky and Michael Pinsker. Schaefer's theorem for graphs. *Journal of the ACM*, 62(3):Article no. 19, 1–52, 2015. A conference version appeared in the Proceedings of STOC 2011, pages 655–664.

14 V. G. Bodnarčuk, L. A. Kalužnin, V. N. Kotov, and B. A. Romov. Galois theory for Post algebras, part I and II. *Cybernetics*, 5:243–539, 1969.

15 Lucas Bordeaux, Marco Cadoli, and Toni Mancini. Generalizing consistency and other constraint properties to quantified constraints. *ACM Trans. Comput. Log.*, 10(3), 2009. Extended abstract appeared at AAAI 2005 titled CSP properties for for quantified constraints: Definitions and complexity.

16 Lucas Bordeaux and Eric Monfroy. Beyond NP: arc-consistency for quantified constraints. In *Principles and Practice of Constraint Programming – CP 2002, 8th International Conference, CP 2002, Ithaca, NY, USA, September 9-13, 2002, Proceedings*, pages 371–386, 2002.

17 F. Börner, A. Krokhin, A. Bulatov, and P. Jeavons. Quantified constraints and surjective polymorphisms. Technical Report PRG-RR-02-11, Oxford University, 2002. Conference version appeared at CSL 2003 titled: Quantified Constraints – Algorithms and Complexity.

18 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.

19 Simone Bova, Robert Ganian, and Stefan Szeider. Model checking existential logic on partially ordered sets. In *Joint Meeting of the 23rd EACSL Annual Conf. on Computer Science Logic (CSL) and the 29th Annual ACM/IEEE Symp. on Logic in Computer Science (LICS), CSL-LICS'14, Vienna, Austria, July 14 – 18, 2014*, pages 21:1–21:10, 2014.

20 Simone Bova, Robert Ganian, and Stefan Szeider. Quantified conjunctive queries on partially ordered sets. *Theor. Comput. Sci.*, 618:72–84, 2016. Extended abstract appeared at IPEC 2004.

21 A. Bulatov, A. Krokhin, and P. G. Jeavons. Classifying the complexity of constraints using finite algebras. *SIAM Journal on Computing*, 34:720–742, 2005.

22 Andrei Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-element set. *J. ACM*, 53(1):66–120, 2006.

23 Andrei Bulatov, Peter Jeavons, and Mikhail Volkov. Finite semigroups imposing tractable constraints. In *Proceedings of the School on Algorithmic Aspects of the Theory of Semigroups and its Applications, Coimbra, Portugal, 2001*, pages 313–329. World Scientific, Singapore, 2002.

24 Andrei A. Bulatov. Combinatorial problems raised from 2-semilattices. *Journal of Algebra*, 298(2):321–339, 2006.

25 Andrei A. Bulatov and Amir Hedayaty. Galois correspondence for counting quantifiers. *Multiple-Valued Logic and Soft Computing*, 24(5-6):405–424, 2015. First version appeared on arxiv in 2012.

26 Catarina Carvalho, László Egri, Marcel Jackson, and Todd Niven. On maltsev digraphs. *Electr. J. Comb.*, 22(1):P1.47, 2015. Extended abstract appeared at CSR 2011.

27 Catarina Carvalho, Florent R. Madelaine, and Barnaby Martin. From complexity to algebra and back: Digraph classes, collapsibility, and the PGP. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015*, pages 462–474, 2015.

28 Witold Charatonik and Michal Wrona. Quantified positive temporal constraints. In *Computer Science Logic, 22nd International Workshop, CSL 2008, 17th Annual Conference of the EACSL, Bertinoro, Italy, September 16-19, 2008. Proceedings*, pages 94–108, 2008.

29 Witold Charatonik and Michal Wrona. Tractable quantified constraint satisfaction problems over positive temporal templates. In *Logic for Programming, Artificial Intelligence, and Reasoning, 15th International Conference, LPAR 2008, Doha, Qatar, November 22-27, 2008. Proceedings*, pages 543–557, 2008.

30 Hubie Chen. Quantified constraint satisfaction and 2-semilattice polymorphisms. In *Principles and Practice of Constraint Programming – CP 2004, 10th International Conference, CP 2004, Toronto, Canada, September 27 – October 1, 2004, Proceedings*, pages 168–181, 2004.

31 Hubie Chen. Quantified constraint satisfaction, maximal constraint languages, and symmetric polymorphisms. In *STACS 2005, 22nd Annual Symposium on Theoretical Aspects of Computer Science, Stuttgart, Germany, February 24-26, 2005, Proceedings*, pages 315–326, 2005.

32 Hubie Chen. A rendezvous of logic, complexity, and algebra. *SIGACT News*, 2006.

33 Hubie Chen. The complexity of quantified constraint satisfaction: Collapsibility, sink algebras, and the three-element case. *SIAM J. Comput.*, 37(5):1674–1701, 2008. Incorporates extended abstract titled *Collapsibility and Consistency in Quantified Constraint Satisfaction* presented at AAAI 2004.

34 Hubie Chen. Existentially restricted quantified constraint satisfaction. *Inf. Comput.*, 207(3):369–388, 2009.

35 Hubie Chen. Quantified constraint satisfaction and the polynomially generated powers property. *Algebra universalis*, 65(3):213–241, 2011. An extended abstract appeared in ICALP B 2008.

36 Hubie Chen. Meditations on quantified constraint satisfaction. In *Logic and Program Semantics – Essays Dedicated to Dexter Kozen on the Occasion of His 60th Birthday*, pages 35–49, 2012.

37 Hubie Chen. Beyond q-resolution and prenex form: A proof system for quantified constraint satisfaction. *Logical Methods in Computer Science*, 10(4), 2014.

38 Hubie Chen. The tractability frontier of graph-like first-order query sets. In *Joint Meeting of the 23rd EACSL Annual Conf. on Computer Science Logic (CSL) and the 29th Annual ACM/IEEE Symp. on Logic in Computer Science (LICS), CSL-LICS'14, Vienna, Austria, July 14-18, 2014*, pages 31:1–31:9, 2014.

39 Hubie Chen and Víctor Dalmau. From pebble games to tractability: An ambidextrous consistency algorithm for quantified constraint satisfaction. In *Computer Science Logic, 19th International Workshop, CSL 2005, 14th Annual Conference of the EACSL, Oxford, UK, August 22-25, 2005, Proceedings*, pages 232–247, 2005.

40 Hubie Chen and Víctor Dalmau. Decomposing quantified conjunctive (or disjunctive) formulas. In *Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25-28, 2012*, pages 205–214, 2012.

41 Hubie Chen, Florent Madelaine, and Barnaby Martin. Quantified constraints and containment problems. In *23rd Annual IEEE Symposium on Logic in Computer Science*, pages 317–328, 2008.

42 Hubie Chen, Florent R. Madelaine, and Barnaby Martin. Quantified constraints and containment problems. *Logical Methods in Computer Science*, 11(3), 2015. Extended abstract appeared at LICS 2008. This journal version incorporates principal part of CP 2012 *Containment, Equivalence and Coreness from CSP to QCSP and Beyond.*

43 Hubie Chen and Dániel Marx. Block-sorted quantified conjunctive queries. In *Automata, Languages, and Programming – 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part II*, pages 125–136, 2013.

44 Hubie Chen and Peter Mayr. Quantified constraint satisfaction on monoids, 2016.

45 Hubie Chen and Moritz Müller. An algebraic preservation theorem for aleph-zero categorical quantified constraint satisfaction. *Logical Methods in Computer Science*, 9(1), 2012. An extended abstract appeared at LICS 2012.

46 Hubie Chen and Michal Wrona. Guarded ord-horn: A tractable fragment of quantified constraint satisfaction. In *19th International Symp. on Temporal Representation and Reasoning, TIME 2012, Leicester, United Kingdom, September 12-14, 2012*, pages 99–106, 2012.

47 Nadia Creignou, Sanjeev Khanna, and Madhu Sudan. *Complexity Classifications of Boolean Constraint Satisfaction Problems.* SIAM Monographs on Discrete Mathematics and Applications 7, 2001.

48 Victor Dalmau. Some dichotomy theorems on constant-free quantified boolean formulas. Technical Report LSI-97-43-R. Departament LSI (UPC), 1997.

49 Víctor Dalmau. A new tractable class of constraint satisfaction problems. *Ann. Math. Artif. Intell.*, 44(1-2):61–85, 2005.

50 Víctor Dalmau and Justin Pearson. Closure functions and width 1 problems. In *Principles and Practice of Constraint Programming – CP'99, 5th International Conference, Alexandria, Virginia, USA, October 11-14, 1999, Proceedings*, pages 159–173, 1999.

51 Petar Dapic, Petar Markovic, and Barnaby Martin. QCSP on semicomplete digraphs. In *Automata, Languages, and Programming – 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, pages 847–858, 2014.

52 Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *J. ACM*, 7(3):201–215, July 1960.

53 Rina Dechter. From local to global consistency. *Artif. Intell.*, 55(1):87–108, 1992.

54 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity.* Texts in Computer Science. Springer, 2013.

55 Uwe Egly. On sequent systems and resolution for QBFs. In *Theory and Applications of Satisfiability Testing – SAT 2012 – 15th International Conference, Trento, Italy, June 17-20, 2012. Proceedings*, pages 100–113, 2012.

56 Uwe Egly, Thomas Eiter, Hans Tompits, and Stefan Woltran. Solving advanced reasoning tasks using quantified boolean formulas. In *Proc. 17th Nat. Conf. on Artificial Intelligence and 12th Conf. on Innovative Applications of Artificial Intelligence*, pages 417–422. AAAI Press/ The MIT Press, 2000.

57 Eduard Eiben, Robert Ganian, and Sebastian Ordyniak. Using decomposition-parameters for qbf: Mind the prefix! In *AAAI 2016*, 2016.

58 T. Feder and M. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory. *SIAM Journal on Computing*, 28:57–104, 1999.

59 Tomás Feder, Pavol Hell, Peter Jonsson, Andrei A. Krokhin, and Gustav Nordh. Retractions to pseudoforests. *SIAM J. Discrete Math.*, 24(1):101–112, 2010.

60 Alex Ferguson and Barry O'Sullivan. Relaxations and explanations for quantified constraint satisfaction problems. In *Principles and Practice of Constraint Programming – CP 2006, 12th International Conf., CP 2006, Nantes, France, September 25-29, 2006, Proceedings*, pages 690–694, 2006.

61 E. C. Freuder. Complexity of k-tree structured constraint satisfaction problems. In *Proceedings of the 8th National Conference on Artificial Intelligence*, pages 4–9, 1990.

62 Jakub Gajarský, Petr Hlinený, Daniel Lokshtanov, Jan Obdrzálek, Sebastian Ordyniak, M. S. Ramanujan, and Saket Saurabh. FO model checking on posets of bounded width. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 963–974, 2015.

63 D. Geiger. Closed systems of functions and predicates. *Pacific Journal of Mathematics*, 27:95–100, 1968.

64 Ian P. Gent, Peter Nightingale, Andrew G. D. Rowley, and Kostas Stergiou. Solving quantified constraint satisfaction problems. *Artif. Intell.*, 172(6-7):738–771, 2008. This is an extended version of conference papers Encoding Quantified CSPs as Quantified Boolean Formulae (ECAI 2004) and QCSP-Solve: A Solver for Quantified Constraint Satisfaction Problems (IJCAI 2005).

65 Georg Gottlob, Gianluigi Greco, and Francesco Scarcello. The complexity of quantified constraint satisfaction problems under structural restrictions. In *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30-August 5, 2005*, pages 150–155, 2005.

66 Martin Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *J. ACM*, 54(1), 2007. Extended abstract appeared at FOCS 2003.

67 Martin Grohe, Thomas Schwentick, and Luc Segoufin. When is the evaluation of conjunctive queries tractable? In *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 657–666, 2001.

68 P. Hell and J. Nešetřil. On the complexity of H-coloring. *Journal of Combinatorial Theory, Series B*, 48:92–110, 1990.

69 Edith Hemaspaandra. Dichotomy theorems for alternation-bounded quantified boolean formulas. *CoRR*, cs.CC/0406006, 2004.

70 P. G. Jeavons. On the algebraic structure of combinatorial problems. *Theoretical Computer Science*, 200:185–204, 1998.

71 Peter Jeavons, David Cohen, and Martin Cooper. Constraints, consistency and closure. *AI*, 101(1-2):251–265, 1998.

72 Marek Karpinski, Hans Kleine Büning, and Peter H. Schmitt. On the computational complexity of quantified horn clauses. In *CSL'87, 1st Workshop on Computer Science Logic, Karlsruhe, Germany, October 12-16, 1987, Proceedings*, pages 129–137, 1987.

73 Alexandr Kazda. Maltsev digraphs have a majority polymorphism. *Eur. J. Comb.*, 32(3):390–397, 2011.

74 Jerome Keisler. Reduced products and Horn classes. *Trans. Amer. Math. Soc.*, 117:307–328, 1965.

75 Hans Kleine Büning and Uwe Bubeck. Theory of quantified boolean formulas. In *Handbook of Satisfiability*, pages 735–760. 2009.

76 Hans Kleine Büning, Marek Karpinski, and Andreas Flögel. Resolution for quantified boolean formulas. *Inf. Comput.*, 117(1):12–18, 1995.

77 Ph. G. Kolaitis and M. Y. Vardi. *Finite Model Theory and Its Applications (Texts in Theoretical Computer Science. An EATCS Series)*, chapter A logical Approach to Constraint Satisfaction. Springer-Verlag New York, Inc., 2005.

78 Ph. G. Kolaitis and M. Y. Vardi. A game-theoretic approach to constraint satisfaction. In *Proceedings of the 17th National Conference on AI*, pages 175–181, 2000.

79 Richard E. Ladner. On the structure of polynomial time reducibility. *J.ACM*, 22(1):155–171, 1975.

80 Benoit Larose. Taylor operations on finite reflexive structures. *International Journal of Mathematics and Computer Science*, 1:1–21, 2006.

81 Benoit Larose and László Zádori. Taylor terms, constraint satisfaction and the complexity of polynomial equations over finite algebras. *IJAC*, 16(3):563–582, 2006.

82 Gary MacGillivray and Jacobus Swarts. Weak near-unanimity functions and digraph homomorphism problems. *Theor. Comput. Sci.*, 477:32–47, 2013.

83 Florent R. Madelaine and Barnaby Martin. QCSP on partially reflexive cycles – the wavy line of tractability. In *Computer Science – Theory and Applications – 8th International Computer Science Symposium in Russia, CSR 2013, Ekaterinburg, Russia, June 25-29, 2013. Proceedings*, pages 322–333, 2013.

84 Miklós Maróti and Ralph McKenzie. Existence theorems for weakly symmetric operations. *Algebra universalis*, 59(3):463–489, 2008.

85 Barnaby Martin. QCSP on partially reflexive forests. In *Principles and Practice of Constraint Programming – 17th International Conference, CP 2011*, 2011.

86 Barnaby Martin and Florent Madelaine. Towards a trichotomy for quantified H-coloring. In *2nd Conf. on Computatibility in Europe, LNCS 3988*, pages 342–352, 2006.

87 Barnaby Martin, Florent R. Madelaine, and Juraj Stacho. Constraint satisfaction with counting quantifiers. *SIAM J. Discrete Math.*, 29(2):1065–1113, 2015. This unifies and expands conference papers from CSR 2012 and 2014.

88 Barnaby Martin, András Pongrácz, and Michal Wrona. The complexity of counting quantifiers on equality languages. In *Pursuit of the Universal – 12th Conf. on Computability in Europe, CiE 2016, Paris, France, 2016, Proceedings*, pages 333–342, 2016.

89 Deepak Mehta, Barry O'Sullivan, and Luis Quesada. Extending the notion of preferred explanations for quantified constraint satisfaction problems. In *Theoretical Aspects of Computing – ICTAC 2015 – 12th International Colloquium Cali, Colombia, October 29-31, 2015, Proceedings*, pages 309–327, 2015.

90 Bernhard Nebel and Hans-Jürgen Bürckert. Reasoning about temporal relations: A maximal tractable subclass of Allen's interval algebra. *J. ACM*, 42(1):43–66, 1995.

91 Christos H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

92 R. Pöschel and L.,A. Kalužnin. *Funktionen- und Relationenalgebren.* Deutscher Verlag der Wissenschaften, 1979.

93 Jussi Rintanen. Constructing conditional plans by a theorem-prover. *Journal of Artificial Intelligence Research*, 10:323–352, 1999.

94 J. A. Robinson. A machine-oriented logic based on the resolution principle. *J. ACM*, 12(1):23–41, January 1965.

95 T. J. Schaefer. The complexity of satisfiability problems. In *Proceedings of STOC'78*, pages 216–226, 1978.

96 Á. Szendrei. *Clones in universal algebra.* Séminaire de mathématiques supérieures. Presses de l'Université de Montréal, 1986.

97 Peter van Beek. On the minimality and decomposability of constraint networks. In *Proceedings of the 10th National Conference on Artificial Intelligence. San Jose, CA, July 12-16, 1992.*, pages 447–452, 1992.

98 James Wiegold. Growth sequences of finite semigroups. *Journal of the Australian Mathematical Society (Series A)*, 43:16–20, 8 1987. Communicated by H. Lausch.

99 Alexander Wires. Dichotomy for finite tournaments of mixed-type. *Discrete Mathematics*, 338(12):2523–2538, 2015.

100 Michal Wrona. Tractability frontier for dually-closed ord-horn quantified constraint satisfaction problems. In *Mathematical Foundations of Computer Science 2014 – 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part I*, pages 535–546, 2014.

101 D. Zhuk. The Size of Generating Sets of Powers. *ArXiv e-prints*, April 2015.