Electronic Colloquium on Computational Complexity, Report No. 126 (2022)

An Invitation to the Promise Constraint Satisfaction Problem

ANDREI KROKHIN, Durham University, UK
JAKUB OPRŠAL, Institute of Science and Technology Austria, Austria

The study of the complexity of the constraint satisfaction problem (CSP), centred around the Feder-Vardi Dichotomy Conjecture, has been very prominent in the last two decades. After a long concerted effort and many partial results, the Dichotomy Conjecture has been proved in 2017 independently by Bulatov and Zhuk. At about the same time, a vast generalisation of CSP, called promise CSP, has started to gain prominence. In this survey, we explain the importance of promise CSP and highlight many new very interesting features that the study of promise CSP has brought to light. The complexity classification quest for the promise CSP is wide open, and we argue that, despite the promise CSP being more general, this quest is rather more accessible to a wide range of researchers than the dichotomy-led study of the CSP has been.

1 INTRODUCTION

What kind of inherent mathematical structure makes a computational problem *tractable*, i.e., polynomial-time solvable (assuming P ≠ NP)? Finding a general answer to this question is one of the fundamental goals of theoretical computer science. The *constraint satisfaction problem (CSP)* and its variants are extensively used towards this ambitious goal for two reasons: on the one hand, the CSP framework is very general and includes a wide variety of computational problems, and on the other hand, this framework has very rich mathematical structure providing an excellent laboratory both for complexity classification methods and for algorithmic techniques.

In this section we informally discuss the CSP and the *promise CSP (PCSP)*, which is the main subject of this survey. Formal definitions and more technical aspects can be found in subsequent sections.

The (decision version of the) CSP can be defined in several equivalent ways. One way is to define it as the problem of deciding the existence of a homomorphism from one finite relational structure (for example, a digraph) to another. We focus on the so-called non-uniform CSP — when the target structure $\mathbf{A}$ is fixed (and is often called a *template* or a *constraint language*). The problem, denoted $\mathrm{CSP}(\mathbf{A})$, becomes the problem of deciding whether an input structure $\mathbf{I}$ homomorphically maps to $\mathbf{A}$ (we write $\mathbf{I} \to \mathbf{A}$). Problems of this form include k-Sat, graph k-colouring, and solving systems of linear equations over a fixed finite field.

A major line of research in CSP focuses on identifying tractable cases and understanding the mathematical structure enabling tractability [see Krokhin and Živný 2017]. Feder and Vardi [1998] conjectured that for each relational structure $\mathbf{A}$, the problem $\mathrm{CSP}(\mathbf{A})$ is either tractable (i.e., in P), or NP-complete, i.e., it cannot be NP-intermediate. Due to its very impressive scope, this CSP Dichotomy Conjecture has been widely considered one of the most important open problems in theoretical computer science. It inspired a very active research programme in the last 25 years, which culminated in two independent proofs of the conjecture in 2017, one by Bulatov [2017] and the other by Zhuk [2017, 2020]. The key part of both proofs are polynomial-time algorithms for the tractable cases, which heavily exploit structural properties of finite universal algebras associated with instances of the problem.

Approximation provides a natural way of coping with NP-hardness. One natural approximation version of the graph k-colouring problem is the *approximate graph colouring* problem, where one needs to find a c-colouring for a given k-colourable graph, for $c \geq k$. This problem is very stubborn — it is believed to be NP-hard for all constants $3 \leq k \leq c$, but, despite being studied for more than 45 years [Garey and Johnson 1976], the progress towards proving this has been very limited. Even for $k = 3$, the best NP-hardness result (without making additional assumptions) has

This survey appeared in ACM SIGLOG News, Volume 9, Issue 3, July 2022, pp 30–59, doi:10.1145/3559736.3559740.

ISSN 1433-8092

recently been improved from $c = 4$ [Khanna et al. 2000; Guruswami and Khanna 2004] to $c = 5$ [Bulín et al. 2019; Barto et al. 2021b], while the best (in terms of c) polynomial-time algorithm for colouring a 3-colourable graph with n vertices uses about $n^{0.199}$ colours [Kawarabayashi and Thorup 2017]. The huge gap between best known negative and positive results shows that we are far from understanding the mathematical nature of this problem. We note that, by additionally assuming different (perfect-completeness) variants of the Unique Games Conjecture, NP-hardness of all approximate graph colouring problems (with $k \geq 3$) was proved in [Braverman et al. 2022; Dinur et al. 2009; Guruswami and Sandeep 2020a], but further in this survey we focus on results that do not make any complexity assumptions other than P $\neq$ NP.

Given the success of the research programme aimed at the CSP Dichotomy Conjecture, it was suggested in [Austrin et al. 2017; Brakensiek and Guruswami 2016, 2021] that perhaps progress on approximate graph colouring and similar problems can be made by looking at a more general picture, using the CSP paradigm. This naturally led to the following formulation: fix not just one relational structure $\mathbf{A}$, but also another structure $\mathbf{B}$ such that $\mathbf{A} \to \mathbf{B}$. Then, given an input structure $\mathbf{I}$ with a promise that $\mathbf{I} \to \mathbf{A}$, the goal is to *find* a homomorphism from $\mathbf{I}$ to $\mathbf{B}$ (which must exist because $\mathbf{I} \to \mathbf{A} \to \mathbf{B}$). Note that a homomorphism to $\mathbf{A}$ is promised to exist, but is not given as a part of input. This promise problem is denoted $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ and is called a *promise CSP*. When $\mathbf{A} = \mathbf{K}_k$ and $\mathbf{B} = \mathbf{K}_c$ are the complete graphs on k and c (resp.) vertices, $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is precisely the approximate graph colouring problem. The assumption $\mathbf{A} \to \mathbf{B}$ is necessary for the problem to make sense. The problem defined above is a search problem. The standard *decision* problem associated with it is to distinguish input structures $\mathbf{I}$ such that $\mathbf{I} \to \mathbf{A}$ from those satisfying $\mathbf{I} \not\to \mathbf{B}$.

Let us make a few easy, but important, observations:

- When $\mathbf{A} = \mathbf{B}$, the decision version of the problem $\mathrm{PCSP}(\mathbf{A}, \mathbf{A})$ is precisely $\mathrm{CSP}(\mathbf{A})$. Thus the PCSP framework generalises the CSP framework.
- The decision version is always reducible to the search version (simply run the search algorithm and then verify its output — if the algorithm crashes or the output is wrong, then we cannot have $\mathbf{I} \to \mathbf{A}$), but it is open whether the two versions are always equivalent.
- Both decision and search $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ are in P if at least one of $\mathrm{CSP}(\mathbf{A})$ and $\mathrm{CSP}(\mathbf{B})$ is in P. This is because tractable decision CSPs have a tractable search problem, as can be shown by a simple self-reduction trick [see, e.g., Bulatov et al. 2005].

Problems $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ can be viewed as approximation versions of $\mathrm{CSP}(\mathbf{A})$ on satisfiable instances, where approximation is understood in a qualitative way (i.e., relaxing constraints, as opposed to allowing to violate some of the constraints). The quest to classify the complexity of problems of the form $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ can also be seen the following way. Fix a structure $\mathbf{A}$ such that $\mathrm{CSP}(\mathbf{A})$ is NP-complete. What are the structures $\mathbf{B}$ such that $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is in P? What are the structures $\mathbf{B}$ such that $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is in NP-hard? What mathematical properties of the structures determine the complexity?

A different way to view $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is as $\mathrm{CSP}(\mathbf{B})$ with restricted inputs (for decision problems, we can view it as either $\mathrm{CSP}(\mathbf{A})$ or $\mathrm{CSP}(\mathbf{B})$ restricted to inputs where the two answers agree). This means that not only the classification problem with respect to P or NP-hardness, but many other questions that have been studied for CSPs (e.g., classification with respect to other complexity classes or with respect to definability in various logics) make perfect sense for PCSPs, possibly with a minor adjustment.

The so-called algebraic approach was the key tool in attacking the CSP Dichotomy Conjecture and related problems. This approach has two layers. The first layer is the framework of polymorphisms. They are simple combinatorial objects — for example, a polymorphism of a graph $\mathbf{G}$ is simply a homomorphism from a direct power of $\mathbf{G}$ to $\mathbf{G}$. Polymorphisms are known to control gadget

reductions between CSPs (as we discuss in detail later). The second, much more advanced, layer is the structural theory of universal algebra. A complete translation of the CSP Dichotomy Conjecture into universal algebra allowed the whole universal algebraic community to contribute to the progress on the conjecture, which eventually lead to enormous progress both in CSP and in universal algebra itself. This theory suited the dichotomy problem so well that it enabled fantastic progress, but, on the other hand, it became so dominant in the theory of CSP that it possibly prevented some mathematical ideas from being developed and some researchers from trying to work in this direction. We will try to argue that the study of PCSP can facilitate new ideas and connections that have been overlooked so far.

The structural theory of universal algebras does not seem to be applicable for PCSPs, but polymorphisms still are. Say, for graphs, they can now be thought of as homomorphisms from a direct power of one graph $\mathbf{G}$ to another graph $\mathbf{H}$. So, one needs to get more creative in using polymorphisms. Roughly, there are three general ways to use polymorphisms for PCSPs (which we discuss in detail in Sections 3–5, respectively):

- When polymorphisms of a PCSP are rich enough, they can be used as rounding procedures in polynomial-time algorithms. One relaxes a given PCSP instance in some way, solves the relaxation, and then rounds the relaxed solution by applying appropriate (for the relaxation) polymorphisms.
- Polymorphisms provide a convenient tool for characterising the existence of specific types of reductions between PCSPs. That is, one PCSP reduces to another PCSP by a specific type of reduction (say, a gadget reduction) if and only if the polymorphisms of the involved PCSPs are related in a certain way.
- If polymorphisms of a PCSP are limited enough, this can be used directly to construct NP-hardness proofs for the PCSP.

The three general guidelines described above are far from being fully formed. There is much to be developed here: what exactly "rich enough" and "limited enough" should mean (and whether there is a gap between them), what useful types of reductions can characterised in this way, and so on.

The study of PCSPs brought to light many new aspects that were not present in the study of CSPs. Here we give a (non-exhaustive) list of such aspects.

- The dichotomy-led study of CSPs sometimes received criticism for not including specific problems of wide interest whose complexity was open. The extension to PCSP includes such problems, the most prominent one being approximate graph colouring.
- When the dichotomy-led study of CSPs started in the 1990s, the Feder-Vardi conjecture was supported by two important special cases that were fully classified by then: the Boolean case [Schaefer 1978] and the graph homomorphism case (also known as H-colouring) [Hell and Nešetřil 1990]. For PCSPs, even these two cases (i.e., when both $\mathbf{A}$ and $\mathbf{B}$ are two-element structures and when both $\mathbf{A}$ and $\mathbf{B}$ are undirected graphs, respectively) are currently wide open. In particular, as tempting as it might be, we cannot now conjecture a dichotomy for PCSPs because we simply do not have enough evidence.
- Already after a few years of research, unexpected connections with new areas appeared, such as the connection with algebraic topology [Krokhin et al. 2020], which appears to play a significant role in approximate graph colouring and probably beyond. Other examples include matrix analysis [Ciardo and Živný 2022a] and Boolean function analysis [Brakensiek et al. 2021]. There is a clear feeling that more new connections will show up because the study of PCSPs seems to be open to contributions from many different communities.

- Already on the existing evidence, new reductions and new algorithms that were overlooked in the study of CSPs can become quite important. This can possibly lead to a new simpler proof of the Feder-Vardi conjecture.
- For each CSP, the decision and the search problems are always equivalent [Bulatov et al. 2005]. For PCSPs, there is always a reduction from decision to search, but the other direction is open. In fact, the design of efficient search algorithms for PCSPs appears to bring brand new challenges [see, e.g., Brakensiek et al. 2020].
- The study of infinite-domain CSP (i.e., CSP($\mathbf{A}$) when the template $\mathbf{A}$ is infinite) has been developing over the last 20 years [Bodirsky 2021], using the algebraic approach as well as model theory and Ramsey theory. The transfer of methods and results between the infinite-domain and finite-domain cases has so far been mainly from latter to the former. The framework of (finite-domain) PCSPs appears to require transfer in the other direction too.

2 BASIC NOTIONS AND EXAMPLES

In this section, we explain the basics of the theory of promise CSPs and show a few key examples. Throughout the paper, we write $[n]$ for the set $\{1, 2, \dots, n\}$.

2.1 CSPs and PCSPs

We define CSP as a homomorphism problem for finite relational structures.

Definition 2.1. A *(relational) structure* is a tuple $\mathbf{A} = (A; R_1^{\mathbf{A}}, \dots, R_l^{\mathbf{A}})$ where each $R_i^{\mathbf{A}} \subseteq A^{\mathrm{ar}(R_i)}$ is a relation on A of arity $\mathrm{ar}(R_i) \geq 1$. We say that $\mathbf{A}$ is finite if its *domain* A is finite. We assume that structures are finite, unless specified otherwise. Two structures $\mathbf{A}$ and $\mathbf{B}$ are called *similar* if they have the same number of relations and the arities of corresponding relations coincide.

For example, a (directed) graph is relational structure with one binary relation. Any two graphs are similar structures.

We often use a single letter instead of R_i to denote a relation of a structure, e.g., $S^{\mathbf{A}}$ would denote a relation of $\mathbf{A}$, the corresponding relation in a similar structure $\mathbf{B}$, would be denoted by $S^{\mathbf{B}}$. Throughout the paper we denote the domains of structures $\mathbf{A}$, $\mathbf{B}$, $\mathbf{K}_n$ and so on by A, B, K_n, etc., respectively.

Definition 2.2. For two similar relational structures $\mathbf{A}$ and $\mathbf{B}$, a *homomorphism* from $\mathbf{A}$ to $\mathbf{B}$ is a map $h\colon A \to B$ such that, for each i, if $(a_1, \dots, a_{\mathrm{ar}(R_i)}) \in R_i^{\mathbf{A}}$, then $(h(a_1), \dots, h(a_{\mathrm{ar}(R_i)})) \in R_i^{\mathbf{B}}$. We write $h\colon \mathbf{A} \to \mathbf{B}$ to denote this, and simply $\mathbf{A} \to \mathbf{B}$ to denote that a homomorphism from $\mathbf{A}$ to $\mathbf{B}$ exists. In the latter case, we also say that $\mathbf{A}$ maps homomorphically to $\mathbf{B}$.

We defined CSP($\mathbf{B}$) in the introduction as a homomorphism problem, i.e., given a structure $\mathbf{I}$ similar to $\mathbf{B}$, decide if $\mathbf{I} \to \mathbf{B}$. There are two other standard ways to define CSP($\mathbf{B}$), all are equivalent. One of them views the domain of $\mathbf{I}$ as a set of variables, the domain of $\mathbf{B}$ as possible values for these variables, and every tuple $\mathbf{v}$ in a relation $R^{\mathbf{I}}$ in $\mathbf{I}$ gives rise to a constraint $(\mathbf{v}, R^{\mathbf{B}})$, postulating that the tuple of values for $\mathbf{v}$ must be in the relation $R^{\mathbf{B}}$. The question is then whether there is an assignment of values to the variables that satisfies all constraints. We will often use this variables-constraints view alongside the homomorphism view. The other way formalises CSP as the problem of satisfiability of primitive-positive (or pp-, for short) formulas in $\mathbf{B}$ [for more details, see Barto et al. 2017].

It is well-known and easy to see that if a CSP($\mathbf{B}$) is NP-complete for some $\mathbf{B}$ (say, graph 3-colouring), there is no polynomial time algorithm that would *find* a homomorphism $\mathbf{I} \to \mathbf{B}$ given the *promise* that such a homomorphism exists (e.g., it is NP-hard to find a 3-colouring of a graph

even when you know that it exists). The *promise CSP* asks whether finding a homomorphism becomes easier if we relax our goal (e.g., we ask to colour a 3-colourable graph with 6 colours). Formally, promise CSP is defined as follows.

Definition 2.3. Given two relational structures $\mathbf{A}, \mathbf{B}$ such that $\mathbf{A} \to \mathbf{B}$, the *search* version of *promise CSP* with template $(\mathbf{A}, \mathbf{B})$ is, given an input structure $\mathbf{I}$ that is promised to map homomorphically to $\mathbf{A}$, *find* a homomorphism $h\colon \mathbf{I} \to \mathbf{B}$.

The *decision promise CSP* with template $(\mathbf{A}, \mathbf{B})$ as above is, given an input structure $\mathbf{I}$ similar to $\mathbf{A}$ and $\mathbf{B}$, output YES if $\mathbf{I} \to \mathbf{A}$, and NO if $\mathbf{I} \not\to \mathbf{B}$. The promise here is that the input falls into one of the two categories.

Both of the search and the decision versions are often denoted by the same symbol $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$, and sometimes not so much attention is given to the distinction between them. Throughout the survey, we do not specify which version (decision or search) is being discussed whenever the results apply to both versions. When the distinction between search and decision is significant, we comment on it.

Let us give some examples of the problems of the form $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ which are proper promise problems, i.e., not of the form $\mathrm{CSP}(\mathbf{A})$. Note that the interesting ones consists of templates $(\mathbf{A}, \mathbf{B})$ where both $\mathrm{CSP}(\mathbf{A})$ and $\mathrm{CSP}(\mathbf{B})$ are NP-complete.

Example 2.4 ($(2+\varepsilon)$-SAT). $(2+\varepsilon)$-SAT is a collective name for a family of promise versions of $(2k+1)$-SAT problems, where $k \geq 1$. Loosely speaking the goal is to find a satisfying assignment to a $(2k+1)$-SAT instance being promised that there is an assignment that satisfies at least k literals in each clause. Formally, the template is the following pair of relational structures:

$$\mathbf{A}_k = (\{0,1\}; \{t \in \{0,1\}^{2k+1} \mid \mathrm{Ham}(t) \geq k\}, \neq_2),$$
$$\mathbf{B}_k = (\{0,1\}; \{t \in \{0,1\}^{2k+1} \mid \mathrm{Ham}(t) \geq 1\}, \neq_2).$$

Here, Ham denotes the Hamming weight of the tuple, and $\neq_2$ the binary inequality relation on $\{0,1\}$ (which is there to capture negative literals in clauses). The problems $\mathrm{PCSP}(\mathbf{A}_k, \mathbf{B}_k)$ were proved to be NP-hard in [Austrin et al. 2017], where also the general framework of PCSPs and the name *promise constraint satisfaction problem* were introduced.

Example 2.5 (Promise 1-in-3-SAT). One notable example of a promise CSP is the promise version of monotone 1-in-3-SAT. Generally, the problem is given a satisfiable instance of monotone 1-in-3-SAT, find a solution in some other structure $\mathbf{B}$ with a ternary (symmetric) relation $R^{\mathbf{B}}$. This is the problem $\mathrm{PCSP}(\mathbf{T}, \mathbf{B})$ where

$$\mathbf{T} = (\{0,1\}; \{(1,0,0), (0,1,0), (0,0,1)\}).$$

What are the structures $\mathbf{B}$ for which $\mathrm{PCSP}(\mathbf{T}, \mathbf{B})$ is tractable (respectively, NP-hard)? This question in this generality was posed in [Barto et al. 2021b], and partially answered in [Barto et al. 2021a].

Consider the case when $\mathbf{B} = \mathbf{H}_2$ is the Boolean ternary not-all-equal structure, i.e., $\mathbf{H}_2 = (\{0,1\}; \{(x,y,z) \mid \text{not } (x=y=z)\})$. Then $\mathrm{CSP}(\mathbf{H}_2)$ is the Not-All-Equal-SAT problem. $\mathrm{PCSP}(\mathbf{T}, \mathbf{H}_2)$ provides a range of interesting new features not found in CSPs: for example, even though $\mathrm{CSP}(\mathbf{T})$ and $\mathrm{CSP}(\mathbf{H}_2)$ are (well-known) NP-hard problems, the problem $\mathrm{PCSP}(\mathbf{T}, \mathbf{H}_2)$ was shown to be in P in [Brakensiek and Guruswami 2021], along with a range of similar problems. We will discuss several other interesting properties of $\mathrm{PCSP}(\mathbf{T}, \mathbf{H}_2)$ later in this survey.

Example 2.6 (Approximate graph colouring). Probably the most famous example of promise constraint satisfaction problem is the *approximate graph colouring* that we already mentioned in the introduction. The problem is to find, for fixed $c \geq k$, a c-colouring of a given k-colourable graph. The template of this PCSP is a pair of cliques, $(\mathbf{K}_k, \mathbf{K}_c)$, where $\mathbf{K}_n = (\{0, \ldots, n-1\}; \neq_n)$.

The problem has been conjectured to be NP-hard for any fixed $3 \le k \le c$, but this is still open in many cases. The current state-of-the-art for NP-hardness is $c = 2k - 1$ for $k = 3, 4, 5$ [Barto et al. 2021b] and $c = \binom{k}{\lfloor k/2 \rfloor} - 1$ for $k \ge 5$ [Krokhin et al. 2020]. Some unconditional results about the applicability of specific algorithms for approximate graph colouring can be found in [Atserias and Dalmau 2022; Ciardo and Živný 2022b].

Example 2.7 (Approximate graph homomorphism). Approximate graph homomorphism is a natural generalisation of approximate graph colouring, it is the PCSP where the template consists of two (undirected) graphs. We note that CSP($\mathbf{H}$) for a fixed (undirected) graph $\mathbf{H}$ is often called $\mathbf{H}$-colouring. A well-known result by Hell and Nešetřil [1990] states that $\mathbf{H}$-colouring is solvable in polynomial time if $\mathbf{H}$ is bipartite or has a loop, and it is NP-complete otherwise. Brakensiek and Guruswami [2021] conjectured that the Hell-Nešetřil dichotomy extends to the promise problem, i.e., that PCSP($\mathbf{G}, \mathbf{H}$) is NP-hard for any non-bipartite loopless undirected $\mathbf{G}, \mathbf{H}$ with $\mathbf{G} \to \mathbf{H}$. It is not hard to see that it is sufficient to prove this conjecture for all cases where $\mathbf{G}$ is an odd cycle and $\mathbf{H}$ is a c-clique with $c \ge 3$. The case $\mathbf{G} = \mathbf{C}_{2k+1}$ (odd cycle), $\mathbf{H} = \mathbf{K}_3$ has been shown to be NP-hard in [Krokhin et al. 2020]. If we consider directed graphs as well, then, extending a similar result for CSPs [Feder and Vardi 1998], it was shown in [Brakensiek and Guruswami 2021] that every problem PCSP($\mathbf{A}, \mathbf{B}$) is polynomial-time equivalent to PCSP($\mathbf{G}, \mathbf{H}$) for suitable digraphs $\mathbf{G}, \mathbf{H}$.

Example 2.8 (Approximate hypergraph colouring). This problem is similar to Example 2.6, but uses the "not-all-equal" relation

$$\mathrm{NAE}_k = \{0, \dots, k-1\}^3 \setminus \{(a, a, a) \mid a \in \{0, \dots, k-1\}\}$$

instead of $\neq_k$, and similarly for c, i.e., we are talking about PCSP($\mathbf{H}_k, \mathbf{H}_c$) where $\mathbf{H}_n = (\{0, \dots, n-1\}; \mathrm{NAE}_n)$. Note that $\mathbf{H}_2$ is the template of NAE-Sat as in Example 2.5. A colouring of a hypergraph is an assignment of colours to its vertices that leaves no hyperedge monochromatic. Thus, in (the search variant of) this problem one needs to find a c-colouring for a given k-colourable 3-uniform hypergraph. This problem has been proved to be NP-hard for any fixed $2 \le k \le c$ [Dinur et al. 2005]. Some variants of this problem that involve rainbow or strong versions of hypergraph colouring, have been studied [Austrin et al. 2020; Brakensiek and Guruswami 2016; Guruswami and Lee 2017; Guruswami and Sandeep 2020b], but full classifications are still open there.

2.2 A theory of gadget reductions

We will briefly outline a general theory of promise CSPs, that is based on the influential 'algebraic approach to CSP' pioneered by Jeavons et al. [1997] and Bulatov et al. [2005] (see [Barto et al. 2017] for a gentle introduction). The core story of the algebraic approach to CSP is an abstraction from the concrete CSP template $\mathbf{A}$ to an algebraic structure, denoted by Pol($\mathbf{A}$) and called *polymorphisms of* $\mathbf{A}$, assigned to it. The main theorem then asserts that the complexity of CSP($\mathbf{A}$) (up to log-space reductions) only depends on the algebraic properties of Pol($\mathbf{A}$). The situation for promise CSP is similar: there is a natural generalisation of polymorphisms to the promise setting, but the polymorphisms do not form an *algebra* or a *clone*, but a weaker structure that is called a *minion*. The core thesis is still valid — the complexity of PCSP($\mathbf{A}, \mathbf{B}$) only depends on the properties of polymorphisms from $\mathbf{A}$ to $\mathbf{B}$, up to log-space reductions. We will define the notion of polymorphism and minion below together with a precise statement from which the above thesis immediately follows.

2.2.1 Gadgets and the scope of the theory. Often the algebraic theory investigates certain constructions γ (so-called *pp-interpretations*, *pp-constructions*, or *pp-powers*, we refer to [Barto et al. 2017, Section 3] for definitions of these notions) of the *template* $\mathbf{B}$ that allow for efficient complexity

reduction from CSP($\gamma(\mathbf{B})$) to CSP($\mathbf{B}$). Here, we would like to focus more on the transformation (of instances) used in the reduction, as opposed to transformation of the template. We discuss the connection with pp-powers near the end of this subsection.

Formally, by a *reduction* from problem A to problem B we mean an efficiently computable function ϕ from instances of problem A to instances of problem B such that a solution to an instance I can be reconstructed from a solution to the instance $\phi(I)$, e.g., if both A and B are promise decision problems, we require that ϕ maps YES instances of A to YES instances of B and NO instances of A to NO instances of B. For search problems, we moreover require an efficiently computable function from solutions of $\phi(I)$ to solutions of I.

A gadget replacement (or local replacement) is a much used technique for constructing reductions in complexity theory, where each basic unit of an instance is replaced by a different structure in a uniform way. Gadget reductions used in (promise) CSP are precisely this type of reductions, but with a more formal notion of a gadget.

We now give formal definitions of gadgets and gadget reductions that we will use. Let us remark that these definitions can be equally well expressed using pp-formulas.

For a gadget reduction from CSP($\mathbf{A}$) to CSP($\mathbf{B}$), we need the following data:

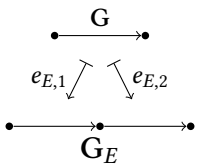
- a structure $\mathbf{G}$ similar to $\mathbf{B}$, to be used as a gadget for variables, and
- a structure $\mathbf{G}_R$ for each (k-ary) relational symbol R of $\mathbf{A}$, with fixed homomorphisms $e_{R,i} \colon \mathbf{G} \to \mathbf{G}_R$ ($1 \le i \le k$), to be used as a gadget for constraints involving R.

Definition 2.9 (Gadget replacement). Assuming the data as above, let $\mathbf{I}$ be an instance of CSP($\mathbf{A}$), i.e., a structure similar to $\mathbf{A}$. We define a gadget replacement as the following construction applied to $\mathbf{I}$, resulting in a structure $\phi(\mathbf{I})$ similar to $\mathbf{B}$.

- For each element $v \in I$, introduce into $\phi(\mathbf{I})$ a copy of $\mathbf{G}$ denoted by $\mathbf{G}^v$. We denote the elements of $\mathbf{G}^v$ by g_v where $g \in G$.
- For each constraint of $\mathbf{I}$, i.e., each relational symbol R say of arity k and all $(v_1, \dots, v_k) \in R^{\mathbf{I}}$, we introduce into $\phi(\mathbf{I})$ a copy of $\mathbf{G}_R$ denoted by $\mathbf{G}_R^{v_1,\dots,v_k}$ and its elements by $g_{R;v_1,\dots,v_k}$. And for each $i \in [k]$, we identify the image of $\mathbf{G}^{v_i}$ under $e_{R,i}$ with its image in $\mathbf{G}_R^{v_1,\dots,v_k}$, i.e., the elements g_{v_i} with $e_{R,i}(g)_{R;v_1,\dots,v_k}$ for each $g \in \mathbf{G}$.

It is well-known that such a gadget replacement can be computed in log-space.

Example 2.10. Let us show a simple gadget replacement from digraphs to digraphs. We use the following gadgets where we use the notation $\mathbf{P}_n = (\{0, \dots, n\}; \{(i, i+1) \mid i < n)$ for a path of length n: $\mathbf{G} = \mathbf{P}_1$, and $\mathbf{G}_E = \mathbf{P}_2$ with the homomorphisms $e_{E,i}$ that map $\mathbf{P}_1$ to the first and second edge of $\mathbf{P}_2$ for $i = 1, 2$ respectively. Graphically, they are represented as



The gadget replacement defined by this gadget then replaces each vertex of the input graph with an edge, and if u is connected to v by an edge, the endpoint of the new edge u is identified with the starting point of v. For example, we get that $\phi(\mathbf{P}_1) = \mathbf{P}_2$, i.e., the path with two vertices becomes a path with two edges as we would expect, but also, more generally, $\phi(\mathbf{P}_n) = \mathbf{P}_{n+1}$ for all $n \ge 1$. See Fig. 1 for more examples of applying this gadget replacement.

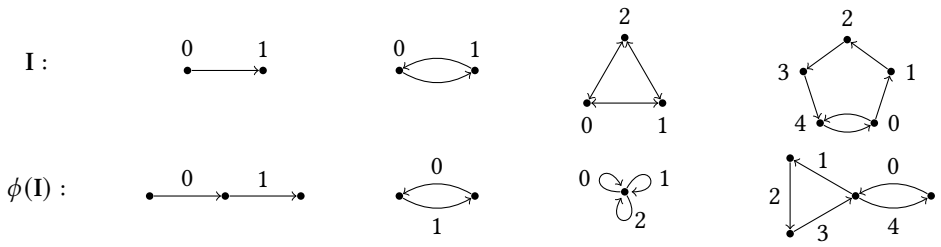


Fig. 1. An example of gadget replacement (Example 2.10)

In many previous papers, e.g., [Barto et al. 2021b, 2017], pp-powers as constructions on the templates have much more prominence than gadget replacements. We include them in this survey to show the connection between the two, and that they are in fact two different points of view on the same concept.

We define a *pp-power* using the same data as a gadget reduction, i.e., structures $\mathbf{G}$, $\mathbf{G}_R$'s, and homomorphisms $e_{R,i}$'s. The corresponding pp-power ρ of a structure $\mathbf{B}'$ similar to $\mathbf{B}$ (and hence to $\mathbf{G}$, and $\mathbf{G}_R$'s) is then defined as follows:

- the domain of $\rho(\mathbf{B}')$ is the set $\hom(\mathbf{G}, \mathbf{B}')$ of all homomorphisms $p\colon \mathbf{G} \to \mathbf{B}'$, and
- a relation $R^{\rho(\mathbf{B}')}$ of arity k is the set

$$\{(f \circ e_{R,1}, \dots, f \circ e_{R,k}) \mid f\colon \mathbf{G}_E \to \mathbf{B}'\}.$$

It is not hard to see that this definition gives (up to homomorphic equivalence[1]) the same structures as the pp-powers defined in [Barto et al. 2018, 2017] using the language of logic.

The key relation between these two constructions is that for each $\mathbf{I}$ and $\mathbf{B}'$, we have that $\mathbf{I} \to \rho(\mathbf{B}')$ if and only if $\phi(\mathbf{I}) \to \mathbf{B}'$. This relation is called *adjunction*, and it is enough to prove that ϕ is a reduction from $\mathrm{CSP}(\rho(\mathbf{B}))$ to $\mathrm{CSP}(\mathbf{B})$, and also from $\mathrm{PCSP}(\mathbf{A}, \rho(\mathbf{B}))$ to $\mathrm{PCSP}(\phi(\mathbf{A}), \mathbf{B}))$ — the proof is straightforward and can be found in [Krokhin et al. 2020, Section 4.2].

2.2.2 Polymorphisms. In this subsection, we define and explain some intuition about one of the key notions of the characterisation of gadget reductions: *polymorphisms* of the template. Loosely speaking, a polymorphism is homomorphism from a *power* (which we define below) of one structure to another (from one part of the template to the other). An intuition why polymorphisms matter comes from the observation that whenever there is a valid gadget reduction from $\mathrm{PCSP}(\mathbf{A}, \mathbf{A}')$ to $\mathrm{PCSP}(\mathbf{B}, \mathbf{B}')$ then a specific gadget reduction that uses powers of $\mathbf{B}$ (which powers are used and how they interact depends on the structure $\mathbf{A}$) will give a reduction as well.

Definition 2.11. The *n-th power of* $\mathbf{B}$, denoted by $\mathbf{B}^n$, is a structure similar to $\mathbf{B}$ with the universe is B^n, the set of all n-ary tuples of elements of $\mathbf{B}$, and relations defined as follows. For each k-ary relation $R^{\mathbf{B}}$, the relation $R^{\mathbf{B}^n}$ is the set of all tuples of the form $(\mathbf{r}_1, \dots, \mathbf{r}_k)$ where $(\mathbf{r}_1(i), \dots, \mathbf{r}_k(i)) \in R^{\mathbf{B}}$ for each $i \in [n]$ (here, $\mathbf{r}(i)$ denotes the i-th entry of $\mathbf{r}$).

Another way to describe the relations of the n-th power of $\mathbf{B}$ is: Consider all $k \times n$-matrices whose columns are tuples in $R^{\mathbf{B}}$. The relation $R^{\mathbf{B}^n}$ consists of all k-tuples of rows of such matrices (keeping the order, so each matrix corresponds to one tuple in $R^{\mathbf{B}^n}$), where each row is interpreted as an n-tuple of elements of B.

[1]Two structures are *homomorphically equivalent* if they are homomorphic to each other.

Polymorphisms are then simply functions that satisfy these power gadgets in the following sense.

Definition 2.12. Let $(\mathbf{B}, \mathbf{B}')$ be a PCSP template.[2] A *polymorphism* from $\mathbf{B}$ to $\mathbf{B}'$ of arity n is a homomorphism $f\colon \mathbf{B}^n \to \mathbf{B}'$. We also simply say that any such f is a polymorphism of the template $(\mathbf{B}, \mathbf{B}')$.

We denote the set of all such n-ary polymorphisms by $\mathrm{Pol}^{(n)}(\mathbf{B}, \mathbf{B}')$, and the set of all polymorphisms of arity $n \geq 1$ by $\mathrm{Pol}(\mathbf{B}, \mathbf{B}')$. Also, we write $\mathrm{Pol}(\mathbf{B})$ for $\mathrm{Pol}(\mathbf{B}, \mathbf{B})$.

In other words, a function $f\colon B^n \to B'$ is an n-ary polymorphism from $\mathbf{B}$ to $\mathbf{B}'$ if, for every (say, k-ary) relation symbol R the following holds: For every $k \times n$ matrix M with entries from B such that each column of M is a tuple in $R^{\mathbf{B}}$, if we apply f to each row of M (treated as a tuple) then we obtain a column representing a tuple from $R^{\mathbf{B}'}$.

These polymorphisms naturally generalise polymorphisms of a single structure that were used in the algebraic approach to CSP [see, e.g., Barto et al. 2017, Section 4].

Example 2.13. Recall Example 2.6. For any $n \geq 1$, the n-ary functions from $\mathrm{Pol}(\mathbf{K}_k, \mathbf{K}_c)$ are simply the c-colourings of $\mathbf{K}_k^n$, the n-th direct (or tensor) power of $\mathbf{K}_k$.

2.2.3 Minors, minions, and minion homomorphisms. We would like to highlight the absence of algebras from this 'algebraic approach to promise CSP', which makes this name a bit of a misnomer. The key property of algebras, or clones, that is missing in polymorphisms of PCSP templates is that the set $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ is not closed under composition — for example, if f and g are polymorphisms from $\mathbf{A}$ to $\mathbf{B}$ of arities 3 and 2 respectively, then the function h defined by $h(x, y, w, z) = f(g(x, w), y, z)$ is not necessarily a polymorphism (as it would be if $\mathbf{A} = \mathbf{B}$). In general, the composition is not even well-defined! This means that, as far as we can see now, most of fundamental results from universal algebra do not apply in PCSP, and often there is no possibility of adapting such methods from the classical CSP. In PCSP, we have to work with much weaker structure on polymorphisms. This weaker structure is called a *minion* and it is obtained from the fact that polymorphisms are always closed under taking *minors*.

Definition 2.14. An n-ary function $f\colon A^n \to B$ is called a *minor* of an m-ary function $g\colon A^m \to B$ given by a map $\pi\colon [m] \to [n]$ if

$$f(x_1, \dots, x_n) = g(x_{\pi(1)}, \dots, x_{\pi(m)})$$

for all $x_1, \dots, x_n \in A$. We will use notation g^π for $g(x_{\pi(1)}, \dots, x_{\pi(m)})$ and write $f = g^\pi$.

Alternatively, one can say that f is a minor of g if it is obtained from g by identifying variables, permuting variables, and introducing dummy variables. The classes of functions that are closed under this operations are called *(function) minions*.

Definition 2.15. A *function minion* $\mathscr{M}$ on a pair of sets (A, B) is a non-empty subset of $\{f\colon A^n \to B \mid n \geq 1\}$ that is closed under taking minors. For fixed $n \geq 1$, let $\mathscr{M}^{(n)}$ denote the set of n-ary functions from $\mathscr{M}$.[3] Unless stated otherwise, we assume that all minions are defined on finite sets.

It is very easy to check that $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ is a minion for any PCSP template $(\mathbf{A}, \mathbf{B})$. One specific minion that is used much, especially in hardness proofs, is the minion $\mathscr{P}_A$ of all projections on a set A, where an n-ary i-th *projection* (a.k.a. *dictator*) on A is the operation $p_n^i(x_1, \dots, x_n) = x_i$. Let

[2] We could also define polymorphisms between any pair of similar structures, but it is not hard to observe that there are none, unless there is a homomorphism between the two structures.

[3] If $A = \emptyset$, we still require that functions in $\mathscr{M}$ have a well-defined arity, i.e., in this case $\mathscr{M}$ is an infinite set that contains one function of each arity.

us denote the minion $\mathscr{P}_{\{0,1\}}$ simply by $\mathscr{P}$, it is easy to see that $\mathscr{P}$ is isomorphic to any minion $\mathscr{P}_A$ with $|A| \geq 2$. It is also well-known [see, e.g., Barto et al. 2017] and easy to check directly that $\mathscr{P} = \mathrm{Pol}(\mathbf{A})$ where $\mathbf{A}$ is the structure such that $\mathrm{CSP}(\mathbf{A})$ is 3-Sat (i.e., $A = \{0, 1\}$ and the relations $R_i^{\mathbf{A}}$, $0 \leq i \leq 3$, are defined by 3-clauses with i negated variables, respectively).

We remark that each polymorphism minion $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ naturally carries a topological structure (see [Krokhin et al. 2020] for details). This is a potentially powerful tool for investigating the complexity of PCSPs, but the study of this direction is still at its infancy.

The existence of gadget reductions can be characterised in terms of *minion homomorphisms* between the polymorphism minions of the involved templates, we present this characterisation in the next subsection. A minion homomorphism is the natural notion of a structure preserving map between minions: the structure of a minion is given by the minor taking operations.

Definition 2.16. Let $\mathscr{M}$ and $\mathscr{N}$ be two minions (not necessarily on the same pairs of sets). A mapping $\xi\colon \mathscr{M} \to \mathscr{N}$ is called a *minion homomorphism* if

(1) it preserves arities, i.e., $\mathrm{ar}(g) = \mathrm{ar}(\xi(g))$ for all $g \in \mathscr{M}$, and
(2) it preserves taking minors, i.e., for each $\pi\colon [m] \to [n]$ and each $g \in \mathscr{M}^{(m)}$ we have

$$\xi(g)^\pi = \xi(g^\pi).$$

We note that item (1) is required in order for (2) to make sense, otherwise $\xi(g^\pi)$ and $\xi(g)$ couldn't form a minor using π.

Example 2.17. Let us start with a trivial example. For each minion $\mathscr{M}$, there is a minion homomorphisms $\xi\colon \mathscr{P} \to \mathscr{M}$ defined as follows. Fix $h \in \mathscr{M}^{(1)}$ which exists since $\mathscr{M}$ is non-empty. And define $\xi(p_n^i)\colon x_1, \dots, x_n \mapsto h(x_i)$. Clearly, $\xi(p_n^i) \in \mathscr{M}^{(n)}$ since $\mathscr{M}$ is closed under taking minors. It is not hard to check that ξ also preserves minor relations.

Example 2.18. There is a minion homomorphism from $\mathrm{Pol}(\mathbf{K}_3, \mathbf{K}_4)$ to the minion $\mathscr{P}$. This minion homomorphism is built on the following combinatorial statement proved in [Brakensiek and Guruswami 2016, Lemma 3.4]: for each (say, n-ary) $f \in \mathrm{Pol}(\mathbf{K}_3, \mathbf{K}_4)$, there exist $t \in K_4$ (we will call any such t a *trash colour*), a coordinate $i \in [n]$, and a map $\alpha\colon K_3 \to K_4$ such that

$$f(a_1, \dots, a_n) \in \{t, \alpha(a_i)\}$$

for all $a_1, \dots, a_n \in K_3$. In other words, if we erase the value t from the table of f then the remaining partial function depends only on x_i. Moreover, it is shown in [Brakensiek and Guruswami 2016, Lemma 3.9], that while the trash colour t is not necessarily unique, the coordinate i is. We define $\xi\colon \mathrm{Pol}(\mathbf{K}_3, \mathbf{K}_4) \to \mathscr{P}$ by mapping each n-ary f to p_n^i for the i that satisfies the above. One can check that this is indeed a minion homomorphism (see [Barto et al. 2021b, Example 2.22] for more details).

Minor conditions are systems of special functional equations (height-1 identities, in the language of universal algebra) that can be satisfied in a minion, and they are useful to show that there is no minion homomorphism from one minion to another. A minor condition is a collection of finitely many *minor identities* which are formal expressions of the form:

$$f(x_1, \dots, x_n) \approx g(x_{\pi(1)}, \dots, x_{\pi(m)}) \tag{1}$$

or $f \approx g^\pi$ for short. We use the symbol $\approx$ to emphasize that this is an equation rather than an equality of two functions. In particular, the variables in this equation are the function symbols (and the x_i's are implicitly assumed to be universally quantified). We say that identity (1) is satisfied in a set of functions from A to B if this set has two specific functions f and g that make this equation an equality of functions, i.e., $f(x_1, \dots, x_n) = g(x_{\pi(1)}, \dots, x_{\pi(m)})$ for all $x_1, \dots, x_n \in A$.

A minor condition is then said to be *satisfied in a minion* $\mathscr{M}$ if for each function symbol appearing in it, there is a function in $\mathscr{M}$ of the corresponding arity, such that this collection of functions satisfy all the identities in the condition. It is easy to see that a minion homomorphism preserves satisfaction of minor conditions in the sense that if $\xi\colon \mathscr{M} \to \mathscr{N}$ is a minion homomorphism, then all minor conditions satisfied in $\mathscr{M}$ are also satisfied in $\mathscr{N}$. The converse is also true for polymorphism minions of finite PCSP templates, i.e., if every minor condition satisfied in $\mathscr{M}$ is also satisfied in $\mathscr{N}$ then there is a minion homomorphism from $\mathscr{M}$ to $\mathscr{N}$ [Barto et al. 2021b, Theorem 4.12(1)–(3)].

Example 2.19. A commonly appearing minor condition is the following collection of three minor identities which is called *(ternary) weak near unanimity*:

$$\begin{aligned} s(x,y) &\approx n(x,x,y)\\ s(x,y) &\approx n(x,y,x)\\ s(x,y) &\approx n(y,x,x) \end{aligned}$$

This minor condition is then satisfied in $\mathrm{Pol}(\mathbf{K}_3,\mathbf{K}_6)$. In order to show that we present $s \in \mathrm{Pol}^{(2)}(\mathbf{K}_3,\mathbf{K}_6)$ and $n \in \mathrm{Pol}^{(3)}(\mathbf{K}_3,\mathbf{K}_6)$: $s(x,y) = x$ for $x \in \{0,1,2\}$.

$$n(x,y,z) = \begin{cases} a & \text{if } a \text{ appears at least twice among } x,y,z, \text{ and}\\ x+3 & \text{otherwise.} \end{cases}$$

It is not hard to check that both s and n are polymorphisms of the given template, and that they indeed satisfy all the identities above.

We can also show that this minor condition is not satisfied in $\mathrm{Pol}(\mathbf{H}_2,\mathbf{H}_k)$ for any $k \geq 2$, where the structures $\mathbf{H}_k$ are from Example 2.8. Indeed, if $s, n \in \mathrm{Pol}(\mathbf{H}_2,\mathbf{H}_k)$ would satisfy such a condition, we would get that

$$n(0,0,1) = n(0,1,0) = n(1,0,0) = s(0,1).$$

which contradicts that n is a polymorphism. To see this, consider the 3×3 matrix whose columns $(0,0,1)$, $(0,1,0)$, $(1,0,0)$ are in the relation of $\mathbf{H}_2$. Applying n to the three rows of this matrix, which are also $(0,0,1)$, $(0,1,0)$, $(1,0,0)$ results in the constant tuple $(s(0,1), s(0,1), s(0,1))$ which is not in the relation of $\mathbf{H}_k$ for any k.

Consequently, this shows that there is no minion homomorphism from $\mathrm{Pol}(\mathbf{K}_3,\mathbf{K}_6)$ to $\mathrm{Pol}(\mathbf{H}_2,\mathbf{H}_k)$ for any $k \geq 2$.

Some minor conditions are useless for the above purpose since they are satisfies in all minions (or, equivalently, in the minion $\mathscr{P}$ of projections), we call such conditions *trivial*. An example of such condition would be a condition obtained from the ternary weak near unanimity by omitting any of the three identities. We remark that the problem of deciding triviality of minor condition is a different formulation of the *Label Cover* problem. Moreover, every problem $\mathrm{PCSP}(\mathbf{A},\mathbf{B})$ is equivalent to the problem of distinguishing minor conditions that are trivial from those not satisfied in $\mathrm{Pol}(\mathbf{A},\mathbf{B})$. We refer to [Barto et al. 2021b, Section 3] for a detailed exposition of the connection between PCSPs, minor conditions, and Label Cover.

Finally, let us briefly mention *abstract minions* which are getting more and more popular in the literature [e.g., Brakensiek et al. 2020; Ciardo and Živný 2022a,b]. In short, an abstract minion is to a function minion what a group is to a permutation group. In long, abstract minion $\mathscr{M}$ is a collection of arbitrary non-empty sets $\{\mathscr{M}^{(n)} \mid n \geq 1\}$ with a well-defined *minor-taking operations*, i.e., maps $(\cdot)^{\pi}\colon \mathscr{M}^{(n)} \to \mathscr{M}^{(m)}$ for each $\pi\colon [n] \to [m]$ that satisfy the (obvious) relations: $(g^{\pi})^{\sigma} = g^{\sigma\circ\pi}$ and $g^{\mathrm{id}} = g$ if id is the identity. A minion homomorphism between such minions is defined in the same way as in Definition 2.16 with the difference that the symbols g^{π} are interpreted as the minor taking

operation of the corresponding minion applied to the corresponding element g. In the language of category theory, an abstract minion is the same as a functor from the category of non-empty finite sets to the category of non-empty sets, and a minion homomorphism is the same as a natural transformation between two functors. We note that every abstract minion can be represented as a function minion on sets (A, B) where A is countable (and B is also countable if $\mathscr{M}^{(n)}$ is countable for all n).

2.2.4 Characterisation of gadget reductions. The following theorem is the result of a long line of gradual improvements and generalisations. Starting with [Jeavons et al. 1997] which introduced the notion of polymorphisms for CSPs, followed by [Bulatov et al. 2005; Barto et al. 2018] which further refined the statement for CSPs, [Barto et al. 2021b] which generalised the scope to PCSPs, and [Krokhin et al. 2020] which mentioned the converse implication (1)→(2) that has received very little attention before.

Theorem 2.20. *Let* $(\mathbf{A}, \mathbf{A}')$ *and* $(\mathbf{B}, \mathbf{B}')$ *be two PCSP templates. Then the following two statements are equivalent:*

(1) $\mathrm{PCSP}(\mathbf{A}, \mathbf{A}')$ *is reducible to* $\mathrm{PCSP}(\mathbf{B}, \mathbf{B}')$ *by a gadget reduction (and hence in log-space),*
(2) *there is a minion homomorphism* $\mathrm{Pol}(\mathbf{B}, \mathbf{B}') \to \mathrm{Pol}(\mathbf{A}, \mathbf{A}')$.

Since polymorphisms of PCSP templates are not closed under composition, unlike polymorphisms of CSP templates, much of the structural theory of universal algebra is not applicable in the promise setting. On the other hand, since the above theorem is a direct generalisation of the corresponding result for CSPs [Barto et al. 2018], every statement about PCSPs applies to CSPs as well, and therefore the study of PCSPs brings new insights into understanding CSPs as well.

One key concept that is useful for proving Theorem 2.20, but also in different settings, is a notion of the *free structure*. It stems from the observation that if we fix $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{B}'$, there is always a structure $\mathbf{A}'$ such that $\mathrm{PCSP}(\mathbf{A}, \mathbf{A}')$ reduces to $\mathrm{PCSP}(\mathbf{B}, \mathbf{B}')$ by a gadget reduction. Moreover, there is a universal such structure (i.e., it maps homomorphically to any suitable $\mathbf{A}'$), and this universal structure depends only on $\mathbf{A}$ and the polymorphisms from $\mathbf{B}$ to $\mathbf{B}'$. It is called the *free structure of* $\mathrm{Pol}(\mathbf{B}, \mathbf{B}')$ *generated by* $\mathbf{A}$ and denoted by $\mathbf{F}_{\mathrm{Pol}(\mathbf{B},\mathbf{B}')}(\mathbf{A})$ in [Barto et al. 2021b, Section 4.1] to which we refer to for an explicit construction, formal definitions, and other useful properties [see, in particular, Barto et al. 2021b, Theorem 4.12].

Example 2.21 (3- vs 5-colouring). Recall the approximate (hyper)graph colouring problems from Examples 2.6 and 2.8. There is a gadget reduction from hypergraph colouring $\mathrm{PCSP}(\mathbf{H}_2, \mathbf{H}_{27,480})$ to graph colouring $\mathrm{PCSP}(\mathbf{K}_3, \mathbf{K}_5)$.[4] This can be shown by arguing that every minor condition satisfied in $\mathrm{Pol}(\mathbf{K}_3, \mathbf{K}_5)$ is also satisfied in $\mathrm{Pol}(\mathbf{H}_2, \mathbf{H}_{27,480})$, and hence there is a minion homomorphism from $\mathrm{Pol}(\mathbf{K}_3, \mathbf{K}_5)$ to $\mathrm{Pol}(\mathbf{H}_2, \mathbf{H}_{27,480})$.

Alternatively, using the claim in the previous paragraph, we could argue that $\mathrm{PCSP}(\mathbf{H}_2, \mathbf{F})$ reduces to $\mathrm{PCSP}(\mathbf{K}_3, \mathbf{K}_5)$, where $\mathbf{F}$ is the free structure of $\mathrm{Pol}(\mathbf{K}_3, \mathbf{K}_5)$ generated by $\mathbf{H}_2$, and that $\mathbf{F} \to \mathbf{H}_{27,480}$ — this is the essence of the proof presented in [Barto et al. 2021b, Section 6].

One can also directly prove that the "universal gadgets" give a desired reduction from $\mathrm{PCSP}(\mathbf{H}_2, \mathbf{H}_{27,480})$ to graph colouring $\mathrm{PCSP}(\mathbf{K}_3, \mathbf{K}_5)$, see [Bulín et al. 2019, Section 4] that contains a self-contained (one-page) proof of this.

3 "RICH ENOUGH" POLYMORPHISMS, ROUNDING, AND TRACTABILITY

Currently, there is essentially one way to prove tractability of a problem $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$, which goes as follows. Assume that we have a structure $\mathbf{S}$ (for sandwich) such that $\mathbf{A} \to \mathbf{S} \to \mathbf{B}$. It is easy to see

[4] The number 27,480 is the number of binary polymorphisms from $\mathbf{K}_3$ to $\mathbf{K}_5$ which was given by a computer calculation.

that any algorithm solving CSP($\mathbf{S}$) also solves the decision version of PCSP($\mathbf{A}, \mathbf{B}$) — indeed, for any input $\mathbf{I}$, if $\mathbf{I} \nrightarrow \mathbf{S}$ then $\mathbf{I} \nrightarrow \mathbf{A}$ and if $\mathbf{I} \to \mathbf{S}$ then $\mathbf{I} \to \mathbf{B}$. We can even allow $\mathbf{S}$ to be an infinite structure (i.e., have an infinite domain) — but the instances of $\mathbf{S}$ are still finite — in fact, every instance of PCSP($\mathbf{A}, \mathbf{B}$) is also an instance of CSP($\mathbf{S}$). So, if this CSP($\mathbf{S}$) is tractable, then so is PCSP($\mathbf{A}, \mathbf{B}$). We remark, though, that many existing tractability proofs for PCSPs do not construct $\mathbf{S}$ explicitly. We will initially focus on decision PCSPs in this section, and briefly discuss search PCSPs at the end of the section.

Example 3.1. Consider PCSP($\mathbf{T}, \mathbf{H}_2$) from Example 2.5. Let $\mathbf{S}$ be the structure whose domain is $\mathbb{Z}$ and whose only relation is $\{(x, y, z) \in \mathbb{Z}^3 \mid x + y + z = 1\}$. It is easy to see that $\mathbf{T} \to \mathbf{S} \to \mathbf{H}_2$. Indeed the first homomorphism is simply a (set-theoretic) inclusion, while the second one maps all negative integers to 0 and the rest to 1. The problem CSP($\mathbf{S}$) amounts to solving linear systems over integers and thus belongs to P [Kannan and Bachem 1979]. It follows that PCSP($\mathbf{T}, \mathbf{H}_2$) is in P. There are other choices for $\mathbf{S}$ that work for this particular PCSP.

Some tractable problems PCSP($\mathbf{A}, \mathbf{B}$) can be explained by the existence of a tractable sandwich CSP with a finite template $\mathbf{S}$, but the size of the smallest possible sandwich structure $\mathbf{S}$ can be arbitrarily large [Kazda et al. 2022] even when the sizes of domains of $\mathbf{A}$ and $\mathbf{B}$ are fixed. However, there exist tractable PCSPs where tractable sandwich structures $\mathbf{S}$ exist, but are all necessarily infinite. One specific example is PCSP($\mathbf{T}, \mathbf{H}_2$) from the example above — it is shown in [Barto et al. 2021b, Section 8] that CSP($\mathbf{C}$) is NP-complete for every finite structure $\mathbf{C}$ with $\mathbf{T} \to \mathbf{C} \to \mathbf{H}_2$.

It is an open question whether every tractable problem PCSP($\mathbf{A}, \mathbf{B}$) sandwiches a tractable CSP($\mathbf{S}$), possibly with infinite $\mathbf{S}$, and whether, for every PCSP($\mathbf{A}, \mathbf{B}$) with this property, such a structure $\mathbf{S}$ can always be chosen to satisfy special nice properties. It is not even clear what such nice properties should be.

In all known cases, such an infinite sandwich structure $\mathbf{S}$ captures the possible outputs of solving a (polynomial-time solvable) relaxation of instances of CSP($\mathbf{A}$), and then appropriate polymorphisms in Pol($\mathbf{A}, \mathbf{B}$) provide a rounding procedure for this relaxation. The prime examples of this pattern are the basic linear programming (BLP) relaxation and the affine integer programming (AIP) relaxation, which we now introduce.

Let $\mathbf{I}$ be an instance of CSP($\mathbf{A}$) and consider the following (standard) 0-1 integer linear program equivalent to this instance. The variables in the program are $\mu_v(a)$ for every $v \in I$ and $a \in A$, and $\mu_{\mathbf{v},R}(\mathbf{a})$ for every relational symbol R, every $\mathbf{v} \in R^{\mathbf{I}}$ and $\mathbf{a} \in R^{\mathbf{A}}$. The intended meaning is that $\mu_v(a) = 1$ if and only if v is assigned a and that $\mu_{\mathbf{v},R}(\mathbf{a}) = 1$ if and only if $\mathbf{v}$ is assigned $\mathbf{a}$. We will write $\mathbf{v}(i)$ or $\mathbf{a}(i)$ for the i-th coordinate of the corresponding tuple. The equations in the program are

$$\sum_{a \in A} \mu_v(a) = 1 \qquad v \in I, \tag{2}$$

$$\sum_{\mathbf{a} \in R^{\mathbf{A}}} \mu_{\mathbf{v},R}(\mathbf{a}) = 1 \qquad \mathbf{v} \in R^{\mathbf{I}}, \tag{3}$$

$$\sum_{\mathbf{a} \in R^{\mathbf{A}}, \mathbf{a}(i) = a} \mu_{\mathbf{v},R}(\mathbf{a}) = \mu_{\mathbf{v}(i)}(a) \qquad a \in A, \mathbf{v} \in R^{\mathbf{I}}, i \in [\mathrm{ar}(R)]. \tag{4}$$

If we relax the above program and allow all variables to take values in the interval $[0, 1]$, we obtain an instance of the linear programming feasibility problem that we denote by $\mathrm{BLP}_{\mathbf{A}}(\mathbf{I})$. If we instead allow all variables to take arbitrary integer values, we obtain a system of linear equations over integers that we denote by $\mathrm{AIP}_{\mathbf{A}}(\mathbf{I})$. It is known that both linear programming feasibility and systems of linear equations over integers can be solved in polynomial time.

It is obvious that if $\mathbf{I} \to \mathbf{A}$ then both $\mathrm{BLP}_{\mathbf{A}}(\mathbf{I})$ and $\mathrm{AIP}_{\mathbf{A}}(\mathbf{I})$ have solutions. We say that BLP (respectively, AIP) solves $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ if we have $\mathbf{I} \to \mathbf{B}$ whenever $\mathrm{BLP}_{\mathbf{A}}(\mathbf{I})$ (respectively, $\mathrm{AIP}_{\mathbf{A}}(\mathbf{I})$) has a solution. What are the PCSPs solvable by BLP or by AIP? We explain this in some detail for BLP, the case of AIP is similar.

Given a solution to $\mathrm{BLP}_{\mathbf{A}}(\mathbf{I})$, how can we round it to a homomorphism from $\mathbf{I}$ to $\mathbf{B}$? It is not hard to see that rational-valued solutions to $\mathrm{BLP}_{\mathbf{A}}(\mathbf{I})$ precisely correspond to homomorphisms from $\mathbf{I}$ to an (infinite) structure $\mathbf{S}$, which depends on $\mathbf{A}$, constructed as follows. The domain of $\mathbf{S}$ is the set of all rational probability distributions over A, i.e., all solutions to the equation $\sum_{a\in A} x_a = 1$ (cf. equation (2)) in non-negative rational numbers. A tuple $\mathbf{t}$ of such probability distributions is in a relation $R^{\mathbf{S}}$ if there is a rational probability distribution γ over $R^{\mathbf{A}}$ such that, for all i, the i-th marginal of γ is the i-th component in $\mathbf{t}$ (cf. equations (3) and (4)). The problem $\mathrm{CSP}(\mathbf{S})$ is a subproblem of the LP feasibility problem and so is tractable. To use this structure $\mathbf{S}$ to prove tractability of $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ for some $\mathbf{B}$, we need to have $\mathbf{A} \to \mathbf{S} \to \mathbf{B}$. It is easy to see that we have $\mathbf{A} \to \mathbf{S}$, by mapping each element in A to the distribution assigning probability 1 to that element. For $n \geq 1$, let $\mathbf{S}_n$ be the (finite) substructure of $\mathbf{S}$ whose domain consists of all probability distributions where the probability of each element is ℓ/n for some $\ell \in \{0, 1, \dots, n\}$. By compactness, we have that $\mathbf{S} \to \mathbf{B}$ if and only if $\mathbf{S}_n \to \mathbf{B}$ for all n. It is not hard to see that homomorphisms from $\mathbf{S}_n$ to $\mathbf{B}$ are in 1-to-1 correspondence with n-ary polymorphisms $f \in \mathrm{Pol}(\mathbf{A}, \mathbf{B})$ such that $f(x_1, \dots, x_n) = f(x_{\pi(1)}, \dots, x_{\pi(n)})$ for all permutations π of $[n]$ (and all values of the variables). Such functions are called *symmetric*. To summarise this discussion, we have the following result (where we add condition (3) for completeness).

Theorem 3.2 ([Barto et al. 2021b]). *For any PCSP template $(\mathbf{A}, \mathbf{B})$, the following are equivalent:*

(1) *BLP solves* $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$,
(2) $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ *contains symmetric functions of all arities, and*
(3) *there is a minion homomorphism from $\mathscr{Q}_{conv}$ to* $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$, *where $\mathscr{Q}_{conv}$ is the minion $\{\sum_{i=1}^n \alpha_i x_i \mid n \geq 1, \sum_{i=1}^n \alpha_i = 1, \alpha_i \in [0, 1]$ for all $i\}$ of functions on $\mathbb{Q}$.*

So, if $\mathrm{BLP}_{\mathbf{A}}(\mathbf{I})$ has a solution then we have $\mathbf{I} \to \mathbf{S}$, and then, since $\mathbf{I}$ is finite, $\mathbf{I} \to \mathbf{S}_n$ for some n that is polynomially bounded in the size of $\mathbf{I}$. Then one can say that an n-ary symmetric polymorphism from $\mathbf{A}$ to $\mathbf{B}$ provides a rounding $\mathbf{I} \to \mathbf{S}_n \to \mathbf{B}$. Thus, if $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ is "rich enough", here in the sense that it contains symmetric functions of all arities, then the decision version of $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is in P.

A similar reasoning works for AIP in place of BLP (in fact, Example 3.1 can be seen as an application of AIP). The "rounding" polymorphisms that naturally appear in this case have the following properties (think $x_1 - x_2 + x_3 - \dots - x_{2n} + x_{2n+1}$). Say that a function $f\colon A^{2n+1} \to B$ is *alternating* if it is *parity-symmetric*, that is,

$$f(x_1, \dots, x_{2n+1}) = f(x_{\pi(1)}, \dots, x_{\pi(2n+1)}) \tag{5}$$

for all permutations π of $[2n + 1]$ that preserve parity (i.e., map odd numbers to odd and even to even), and additionally, f satisfies the following *cancellation* law:

$$f(x_1, \dots, x_{2n-1}, y, y) = f(x_1, \dots, x_{2n-1}, z, z). \tag{6}$$

Theorem 3.3 ([Barto et al. 2021b]). *For any PCSP template $(\mathbf{A}, \mathbf{B})$, the following are equivalent:*

(1) *AIP solves* $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$,
(2) $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ *contains alternating functions of all odd arities, and*
(3) *there is a minion homomorphism from $\mathscr{Z}_{aff}$ to* $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$, *where $\mathscr{Z}_{aff}$ is the minion $\{\sum_{i=1}^n \alpha_i x_i \mid n \geq 1, \sum_{i=1}^n \alpha_i = 1, \alpha_i \in \mathbb{Z}$ for all $i\}$ of functions on $\mathbb{Z}$.*

We remark that AIP was not studied in the dichotomy-led research on CSPs, so the potential of this algorithm as a subroutine in efficient algorithms for (P)CSPs and its relationship to other

algorithms for CSPs [e.g., Bulatov and Dalmau 2006; Idziak et al. 2010] are currently not understood well.

One particular way to combine BLP and AIP was considered in [Brakensiek et al. 2020]. Roughly, this combined algorithm first solves $\mathrm{BLP}_{\mathbf{A}}(\mathbf{I})$ and, if solvable, it finds a special solution s for it (a relative interior point in the rational polytope of solutions). Then it strengthens $\mathrm{AIP}_{\mathbf{A}}(\mathbf{I})$ by adding equations $x = 0$, where x is any variable that is equal to 0 in s. Finally, it solves this modified $\mathrm{AIP}_{\mathbf{A}}(\mathbf{I})$. The algorithm rejects if either of $\mathrm{BLP}_{\mathbf{A}}(\mathbf{I})$ and the modified $\mathrm{AIP}_{\mathbf{A}}(\mathbf{I})$ has no solution and accepts otherwise. This algorithm, called BLP+AIP, can also be seen as going via an appropriate sandwich structure $\mathbf{S}$, and the power of BLP+AIP can also be characterised via polymorphisms. Several closely related characterisations were given in [Brakensiek et al. 2020], one of them is as follows.

Theorem 3.4 ([Brakensiek et al. 2020]). *BLP+AIP solves* $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ *if and only if* $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ *contains parity-symmetric functions of all odd arities.*

Theorem 3.4 also has an equivalent third condition similar to Theorems 3.2 and 3.3. It involves an abstract minion $\mathscr{M}_{\mathrm{conv+aff}}$ whose definition, though quite simple, is a bit too lengthy for this survey. See [Brakensiek et al. 2020] for details.

Using Theorem 3.4 and the description of tractable cases of Boolean CSPs [see, e.g., Barto et al. 2017], it is easy to see that BLP+AIP solves all these tractable cases. However, there are simple examples of non-Boolean CSPs that are not solvable by this algorithm [see Brakensiek et al. 2020]. One can generalise BLP, AIP, and BLP+AIP to hierarchies such as the Sherali-Adams hierarchy for linear programming (see, e.g., [Ciardo and Živný 2022a,b] for some initial studies of applying such hierarchies for PCSPs). It is open whether using (some finite level of) such hierarchies gives an algorithm that can solve all tractable decision PCSPs (or even all tractable CSPs).

Another interesting new aspect of PCSPs concerns the number of polymorphisms that are sufficient to guarantee tractability. It is known that the tractable CSPs can be characterised by the presence of a single polymorphism of arity 4 [see, e.g., Barto et al. 2017], but all theorems in this section use infinite sequences of polymorphisms. This is not a coincidence, since in PCSP one cannot compose polymorphisms, and, in particular, one cannot produce useful polymorphisms of large arities from those of smaller arities. Moreover, it follows from [Barto and Kozik 2022] that, for any finite family F of (multi-variable) functions from A to B, there is an NP-hard problem $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ with $F \subseteq \mathrm{Pol}(\mathbf{A}, \mathbf{B})$ (see also Corollary 4.4 and the discussion below it). This appears to suggest that tractable PCSPs require infinite CSP algorithms, such as BLP or AIP. The nature of polynomial-time algorithms that solve all tractable CSPs [Bulatov 2017; Zhuk 2017, 2020] is very finite. Does all this suggest that there are (potentially simpler) infinite-flavoured efficient algorithms for PCSPs, which, in particular, can solve all tractable CSPs?

3.1 Tractability of search PCSPs

Let us now discuss tractability for search PCSPs. When search is concerned, there is a difference between CSP and PCSP. As mentioned before, the search version of any tractable decision CSP is tractable, but this reduction from search to decision works only for PCSPs that sandwich a tractable $\mathrm{CSP}(\mathbf{S})$ with finite $\mathbf{S}$. Indeed, if $\mathbf{A} \to \mathbf{S} \to \mathbf{B}$ then, for any instance $\mathbf{I}$ of $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$, we can solve $\mathbf{I}$ as an instance of the search problem for $\mathrm{CSP}(\mathbf{S})$ and use a pre-computed homomorphism $\mathbf{S} \to \mathbf{B}$. However, when infinite sandwich structures $\mathbf{S}$ must be used, then we need to be able to efficiently compute both a homomorphism h from an input structure $\mathbf{I}$ to $\mathbf{S}$ and a homomorphism from the substructure $h(\mathbf{I})$ of $\mathbf{S}$ to $\mathbf{B}$ (i.e., both solve the relaxation and round the obtained solution). In this case, efficient search algorithms are currently known only for very few PCSPs, where the

underlying (specific) polymorphisms can be easily evaluated to provide efficient rounding [see, e.g., Brakensiek and Guruswami 2021].

If we want to use (say, symmetric) polymorphisms as rounding for the search version of PCSP($\mathbf{A}, \mathbf{B}$), we need to be able to evaluate them efficiently. More specifically, we need a sequence of polymorphisms $f_n \in \mathrm{Pol}(\mathbf{A}, \mathbf{B})$ such that each f_n is n-ary, and one can compute the value of f_n on a given tuple in time polynomial in n. One possible obstacle for this, though, is that the application of polymorphisms as rounding procedures for BLP+AIP in [Brakensiek et al. 2020] uses polymorphisms whose arity can be exponential in the size of the instance. It is currently not clear whether this obstacle is inherent (at least, for some cases) or can always be avoided by using polymorphisms in a suitable way. In general, however, the study of efficient polymorphisms, i.e., those that can be efficiently evaluated, is a new, wide open, and interesting direction.

3.2 Some open questions and directions

To finish this section, let us mention several open questions and directions. While many of the questions and directions below are general, partial answers and new examples would be of considerable interest because we currently see only a small part of the complexity landscape in PCSP. This comment applies also to the lists of questions and directions in Sections 4.3 and 5.3.

- The tractable problem PCSP($\mathbf{T}, \mathbf{H}_2$) from Examples 2.5 and 3.1 has the properties that both structures are Boolean (i.e., have domain $\{0, 1\}$) and all their relations are symmetric (i.e., invariant under all permutations of coordinates). All tractable PCSPs with these properties have been described in [Brakensiek and Guruswami 2021; Ficak et al. 2019], they all either admit a tractable finite sandwich or can be solved by BLP or AIP. Find other interesting examples of tractable Boolean (or indeed non-Boolean) PCSPs.
- Fix your favourite NP-hard problem of the form CSP($\mathbf{A}$) and describe all tractable problems PCSP($\mathbf{A}, \mathbf{B}$). (One can loosely call this describing tractable qualitative approximations to CSP($\mathbf{A}$)).
- Find new interesting polynomial-time algorithms that can solve (some new) PCSPs.
- Is it true that every tractable problem PCSP($\mathbf{A}, \mathbf{B}$) sandwiches a (possibly infinite) tractable CSP($\mathbf{S}$)?
- Whenever PCSP($\mathbf{A}, \mathbf{B}$) sandwiches a tractable CSP($\mathbf{S}$), where $\mathbf{S}$ is infinite, can $\mathbf{S}$ always be chosen to satisfy special nice properties?
- What are the PCSPs that admit a finite sandwich structure with tractable CSP? (Such PCSPs are called *finitely tractable* by Asimi and Barto [2021]).
- Are there other natural relaxations (and the corresponding structures and families of polymorphisms) that can be used to provide efficient algorithms for PCSPs? Can families of polymorphisms that play an important role in CSP (such as cyclic polymorphisms, [see Barto et al. 2017]) suggest new relaxations (for which they can provide rounding)?
- Can every tractable CSP (or even PCSP) be solved by combining BLP and AIP (possibly using some appropriate finite level of Sherali-Adams-like hierarchies)?
- Is it true that the search version of PCSP($\mathbf{A}, \mathbf{B}$) is tractable whenever the decision version is?
- Is the search problem tractable for the PCSPs satisfying the conditions from Theorems 3.2–3.4?
- Can a problem PCSP($\mathbf{A}, \mathbf{B}$) have an efficient search algorithm, even in the presence of symmetric polymorphisms of all arities, that is "oblivious" to the underlying polymorphisms? [Brakensiek et al. 2020]

- Is it true that when a search PCSP is tractable, this can always be explained by families of polymorphisms that can be evaluated in polynomial time?
- While we focus here mostly on the computational complexity of PCSPs, there was much research into descriptive complexity of CSPs, i.e., expressibility of problems CSP($\mathbf{A}$) in various logics [see, e.g., Bulatov et al. 2008; Atserias et al. 2009]. Many questions in this direction are wide open for PCSPs.
- The two basic polynomial-time algorithms used to solve CSPs are the bounded-width algorithm [Barto and Kozik 2014] and the few subpowers algorithm [Bulatov and Dalmau 2006; Idziak et al. 2010]. Both use composition of polymorphisms — the former in its correctness proof and the latter in the actual algorithm. Are there "composition-free" algorithms that can replace them (i.e., solve the same CSPs)?

4 REDUCTIONS THAT GO BEYOND GADGET REPLACEMENTS

In the general algebraic theory of CSP, it was sufficient to group and compare CSPs by the gadget reductions (described in Section 2.2), so that two CSPs that are reducible to each other by gadget reductions are considered essentially the same. Moreover, it follows from the CSP Dichotomy Theorem [Bulatov 2017; Zhuk 2020] that (unless P = NP) a CSP is NP-hard if and only if 3-Sat reduces to it via a gadget reduction. Of course, one can still use gadget reductions to reduce from 3-Sat (or any other NP-hard CSP) to PCSPs — however, there are NP-hard PCSPs whose NP-hardness provably cannot be shown by a gadget reduction from an NP-hard CSP. Examples of this include the $(2+\varepsilon)$-Sat problem from Example 2.4, and all the state of the art results on approximate graph colouring from Example 2.6. Hardness arguments for these examples rely on ad hoc reductions from problems that are proven NP-hard by different methods, typically by reducing from the Gap Label Cover problem, which is NP-hard by the PCP theorem and the Parallel Repetition Theorem [Arora et al. 1998; Arora and Safra 1998; Raz 1998]. This frequent use of ad hoc reductions was one reason why [Barto et al. 2021b] called for an extension of the algebraic approach with more reductions. Such reductions would provide a coarser grouping of PCSPs than the one provided by gadget replacement, where more PCSPs are considered essentially the same (than what is provided by Theorem 2.20). Obviously, this could expand the scope of tractability results. Moreover, we would like to have a general theory for PCSPs, where (as many as possible) hardness results are explained in a uniform way, ideally via reductions of some regular type (which ideally would be characterised in a similar way gadget reductions are characterised by polymorphisms) and from as few problems as possible (ideally just from 3-Sat). This section outlines a few of the recent advances in that direction.

A common feature of the new reductions described in this subsection is that they are, surprisingly, *pp-powers* (recall Section 2.2.1) — which is a construction that is efficiently computable but, when studying gadget reductions, is applied to the template rather than to an instance.

4.1 The arc-graph reduction and adjunctions

A reduction between PCSPs that provably goes beyond gadget replacements has been used in [Krokhin et al. 2020] to improve hardness results for approximate graph colouring. This reduction transforms a digraph into its *arc-graph*, which is defined as follows. For a digraph $\mathbf{G}$, its arc-graph $\delta(\mathbf{G})$ is the digraph whose vertices are the edges of $\mathbf{G}$, and whose edges are pairs of edges of $\mathbf{G}$ forming a 2-path:

$$\delta(\mathbf{G}) = (E^{\mathbf{G}}; \{((x,y),(z,w)) \mid (x,y),(z,w) \in E^{\mathbf{G}}, y = z\}).$$

It is known that, for any digraphs $\mathbf{G}, \mathbf{H}$, we have $\delta(\mathbf{G}) \to \mathbf{H}$ if and only if $\mathbf{G} \to \delta_R(\mathbf{H})$, where the transformation δ_R can be described in a few ways. For example, we can set $\delta_R(\mathbf{H})$ to be the

digraph whose vertices are all subsets of H, and there is an edge from a subset U to a subset V if there is $u \in U$ such that $(u, v) \in E^{\mathbf{H}}$ for all $v \in V$. It is not hard to check that δ is a reduction from $\mathrm{PCSP}(\mathbf{G}, \delta_R(\mathbf{H}))$ to $\mathrm{PCSP}(\delta(\mathbf{G}), \mathbf{H})$ whenever both PCSPs are defined, i.e., when $\delta(\mathbf{G}) \to \mathbf{H}$ and $\mathbf{G} \to \delta_R(\mathbf{H})$.

This reduction can be used to reduce from $\mathrm{PCSP}(\mathbf{K}_{b(k)}, \mathbf{K}_{b(c)})$ to $\mathrm{PCSP}(\mathbf{K}_k, \mathbf{K}_c)$ for any $k, c \geq 4$ where $b(k) = \binom{k}{\lfloor k/2 \rfloor}$. Starting with a result of Huang [2013], a repeated application of this reduction has led to a considerable improvement in state-of-the-art NP-hardness results for approximate graph colouring [Krokhin et al. 2020]. We will discuss this in more detail in Section 5.2.

The arc-graph reduction falls into a wider scheme of reductions that are described by *adjunctions*, which is a notion from category theory. Two transformations γ and ω are said to be *adjoint* if, for all structures $\mathbf{A}$ and $\mathbf{B}$,

$$\gamma(\mathbf{A}) \to \mathbf{B} \text{ if and only if } \mathbf{A} \to \omega(\mathbf{B}), \tag{7}$$

assuming that γ transforms $\mathbf{A}$ to a structure similar to $\mathbf{B}$ and ω transforms $\mathbf{B}$ to a structure similar to $\mathbf{A}$. In this case γ is called a *left adjoint* to ω, and ω a *right adjoint* to γ. Another example of adjunction is when γ is a gadget replacement, and ω the corresponding pp-power, as we described in Section 2.2.1. As it is known from the algebraic approach, in this case γ gives an efficiently computable reduction from $\mathrm{CSP}(\omega(\mathbf{A}))$ to $\mathrm{CSP}(\mathbf{A})$ for each relational structure $\mathbf{A}$ (to which ω can be applied). This in fact generalises to arbitrary adjunctions (naturally, from the computational complexity perspective, we are only interested in cases where γ is efficiently computable), and to PCSPs: any efficiently computable left adjoint γ to ω can be used as a reduction from $\mathrm{PCSP}(\mathbf{B}, \omega(\mathbf{A}'))$ to $\mathrm{PCSP}(\gamma(\mathbf{B}), \mathbf{A}')$ [Krokhin et al. 2020]. More information on the use of adjunctions in PCSP (and more examples) can be found in [Krokhin et al. 2020].

We remark that it is currently not known how (or whether) reductions given by adjunctions, even in the special case of the arc-graph reduction, can be characterised in terms of polymorphisms of the involved templates.

4.2 Generalising minion homomorphisms

We briefly outline the core results of a recent paper by Barto and Kozik [2022] that describes a reduction between PCSPs that is more general than gadget reductions and gives a sufficient condition for the existence of this reduction from $\mathrm{PCSP}(\mathbf{A}, \mathbf{A}')$ to $\mathrm{PCSP}(\mathbf{B}, \mathbf{B}')$ that is weaker than the existence of minion homomorphism from $\mathrm{Pol}(\mathbf{B}, \mathbf{B}')$ to $\mathrm{Pol}(\mathbf{A}, \mathbf{A}')$. Moreover, this sufficient condition is still expressed in terms of the two polymorphism minions. This result goes towards the general goal described in the beginning of this section. In particular, it allows one to to show NP-hardness of some PCSPs by a regular type of reduction from 3-Sat through the use of polymorphisms. This covers several NP-hardness results, e.g., those from Examples 2.4, 2.7, and 2.8, where original proofs relied on Gap Label Cover and the PCP theorem.

We formulate a simplified version of the reduction used in [Barto and Kozik 2022] — while formally, this formulation is different, it can be shown that if there is a reduction according to [Barto and Kozik 2022] between two PCSPs, then the reduction described here also works. We use the notation $\binom{X}{\leq k}$ for the set of all at most k-element non-empty subsets of X. And we refine a definition of a power of a structure to allow sets for exponents; the structure $\mathbf{B}^X$ is isomorphic to $\mathbf{B}^{|X|}$.

Definition 4.1. Let X be a set and $\mathbf{B}$ a relational structure, we define the *X-th power of* $\mathbf{B}$ denoted by $\mathbf{B}^X$ as the structure $(B^X; R^{\mathbf{B}^X}, \dots)$ where B^X is the set of all functions from X to B, and

$$R^{\mathbf{B}^X} = \{(b_1, \dots, b_k) \mid (b_1(x), \dots, b_k(x)) \in R^{\mathbf{B}} \text{ for all } x \in X\}$$

for each k-ary relational symbol R of $\mathbf{B}$.

We now give a concise description of the reduction which we will call here, for the lack of better name, *k-reduction*. Assume that we are reducing from PCSP($\mathbf{A}, \mathbf{A}'$) to PCSP($\mathbf{B}, \mathbf{B}'$), and fix a (large enough) positive integer k. Intuitively, the reduction is very similar to a gadget replacement with the key difference that we consider each set of at most k variables at a time as one unit replaced by one gadget — each of these gadgets is a power of $\mathbf{B}$ as defined above.

Let $\mathbf{I}$ be an instance of PCSP($\mathbf{A}, \mathbf{A}'$), we construct an instance $\phi(\mathbf{I})$ as follows. For a subset $K \subseteq I$, let $\mathbf{I}[K]$ denote the (induced) substructure of $\mathbf{I}$ with domain K obtained by removing all tuples containing elements not from K from all relations in $\mathbf{I}$.

- For each $K \in \binom{I}{\leq k}$, let $\mathcal{F}_K$ be the set of all homomorphisms from $\mathbf{I}[K]$ to $\mathbf{A}$. For each such K, introduce into $\phi(\mathbf{I})$ a copy of $\mathbf{B}^{\mathcal{F}_K}$.
- For each $L \subset K \in \binom{I}{\leq k}$, there is a natural map from $\mathcal{F}_K$ to $\mathcal{F}_L$ that maps h to its restriction $h|_L$. We identify each element b of $\mathbf{B}^{\mathcal{F}_L}$ with the element b_K of $\mathbf{B}^{\mathcal{F}_K}$ defined as $b_K(h) = b(h|_L)$.

It is not hard to see that this transformation ϕ can be computed in log-space. When is this transformation a reduction from PCSP($\mathbf{A}, \mathbf{A}'$) to PCSP($\mathbf{B}, \mathbf{B}'$)? Barto and Kozik [2022] provide a partial answer to this question. We also note that they only claim that their reduction is polynomial time since in order to provide a log-space reduction between search problems, we need to provide an efficient way to reconstruct a solution for $\mathbf{I}$ from a solution $\phi(\mathbf{I})$. Theorem 4.3 below only asserts that if there is a solution to $\phi(\mathbf{I})$ then there is also a solution to $\mathbf{I}$, but it is not obvious whether it can be computed in log-space.

We remark that k-reduction is also closely connected with Sherali-Adams (SA) hierarchy [see Sherali and Adams 1990; Ciardo and Živný 2022b] in the following sense. First, SA can be essentially viewed as a reduction from a CSP to linear programming. Naturally, this reduction is different from the basic linear programming relaxation. Namely, the difference lies in the fact that the k-th level of SA considers k variables at the same time in very much the same way as in the definition above. A difference from the above construction is in the step where the sets $\mathcal{F}_K$ are converted into a gadget ($\mathbf{B}^{\mathcal{F}_K}$ above): SA uses a more efficient gadget consisting of all probability distributions on $\mathcal{F}_K$. This consequently also alters the identification step (second item above).

Barto and Kozik [2022] introduce a generalisation of minion homomorphisms that is a sufficient condition for the existence of a k-reduction for some k. The simpler version of the new homomorphism is the so-called *d-homomorphism* which is a mapping that assigns to each function from Pol($\mathbf{B}, \mathbf{B}'$) a list of at most d *candidates* from Pol($\mathbf{A}, \mathbf{A}'$) of the same arity, such that, for each minor relation $f = g^\pi$, there is a choice of candidates, f' from image of f and g' from the image of g, that satisfy the same minor relation $f' = (g')^\pi$. (The standard minion homomorphism is thus a 1-homomorphism.) This notion further generalises to (d, r)-homomorphisms defined below; a d-homomorphism is a $(d, 1)$-homomorphism. Loosely speaking, a d-homomorphism forces us to weakly satisfy each minor relation, while a $(1, r)$-homomorphism does not require that each minor relation is satisfied, but at least one that from each chain of r minor relations. A precise definition that combines both these behaviours is given below.

Definition 4.2. Fix a minion $\mathscr{M}$, a *chain of minors* in $\mathscr{M}$ of length r is a sequence $t_0, \dots, t_r \in \mathscr{M}$ together with functions $\pi_{i,j}$ for $0 \leq i < j \leq r$, such that $t_j = t_i^{\pi_{i,j}}$ for all $i < j$ and $\pi_{i,k} = \pi_{i,j} \circ \pi_{j,k}$, for all $i < j < k$.

Let $\mathscr{M}$ and $\mathscr{N}$ be two minions. A *(d, r)-homomorphism* from $\mathscr{M}$ to $\mathscr{N}$ is a mapping ξ that assigns to each element of $\mathscr{M}$ a non-empty subset of $\mathscr{N}$ of size at most d such that

(1) for all $f \in \mathscr{M}$, every $g \in \xi(f)$ has the same arity as f.
(2) for each chain of minors in $\mathscr{M}$ as above, there exist $i < j$ and $g_i \in \xi(t_i)$, $g_j \in \xi(t_j)$ such that $g_j = g_i^{\pi_{i,j}}$.

Note that, in a chain of minors, the maps $\pi_{i,j}$ are determined by $\pi_{i,i+1}$ since all others are obtained by composition. So, one way to create such a chain is to start with t_0 and to iteratively take a minor $t_{i+1} = t_i^{\pi_{i,i+1}}$.

Often a (d,r)-homomorphism can be given by providing r partial d-homomorphisms that cover the whole minion $\mathscr{M}$. More precisely, assume that $\mathscr{M} = \mathscr{M}_1 \cup \cdots \cup \mathscr{M}_r$ where each $\mathscr{M}_i$ is not necessarily a minion, and that we have maps $\xi_i \colon \mathscr{M}_i \to \mathscr{N}$ that are *partial d-homomorphisms*, i.e., they satisfy the conditions (1) and (2) in Definition 4.2 for every chain of minors of length 1 that is contained in the same $\mathscr{M}_i$. Then there is a (d,r)-homomorphism $\xi \colon \mathscr{M} \to \mathscr{N}$ which is defined by taking the union of ξ_i's and breaking ties arbitrarily whenever some of ξ_i's disagree. This is a (d,r)-homomorphism, since, for every chain of minors $t_0, \dots, t_r$, there will be some s such that some t_i and t_j belong to the same $\mathscr{M}_s$ and $\xi(t_i) = \xi_s(t_i)$ and $\xi(t_j) = \xi_s(t_j)$.

Theorem 4.3 ([Barto and Kozik 2022, Theorem 5.1]). *Let $(\mathbf{A}, \mathbf{A}')$ and $(\mathbf{B}, \mathbf{B}')$ be two PCSP templates. If there is a (d,r)-homomorphism from $\mathrm{Pol}(\mathbf{B}, \mathbf{B}')$ to $\mathrm{Pol}(\mathbf{A}, \mathbf{A}')$, then there exists k such that $\mathrm{PCSP}(\mathbf{A}, \mathbf{A}')$ reduces to $\mathrm{PCSP}(\mathbf{B}, \mathbf{B}')$ by the k-reduction.*

One important consequence of the above theorem (Corollary 4.4 below) is that polymorphisms of a bounded arity (or, more generally, of bounded essential arity) have no influence on the complexity of a PCSP. In particular, one cannot make any conclusions about the complexity of $\mathrm{PCSP}(\mathbf{A}, \mathbf{A}')$ solely on the basis of the presence of any single finite set of functions in the polymorphism minion $\mathrm{Pol}(\mathbf{A}, \mathbf{A}')$. A coordinate i of a function $f \colon A^n \to A'$ is called *inessential* if, for all $i \in [n]$ and $a_1, \dots, a_n, a_i' \in A$, we have $f(a_1, \dots, a_i, \dots, a_n) = f(a_1, \dots, a_i', \dots, a_n)$. Otherwise, the coordinate i is called *essential*. The number of essential coordinates of a function is called its *essential arity*.

Corollary 4.4. *Fix any number $m \geq 1$. If there is a partial minion homomorphism*

$$\xi \colon \mathscr{B} \to \mathrm{Pol}(\mathbf{A}, \mathbf{A}')$$

where $\mathscr{B} \subseteq \mathrm{Pol}(\mathbf{B}, \mathbf{B}')$ is the set of all polymorphisms of essential arity at least m then $\mathrm{PCSP}(\mathbf{A}, \mathbf{A}')$ is reducible to $\mathrm{PCSP}(\mathbf{B}, \mathbf{B}')$ by k-reduction for some k.

By partial minion homomorphism in the above theorem, we mean that minors $f = g^\pi$ are preserved for all functions $f, g \in \mathscr{B}$.

As a special case of Corollary 4.4, consider the situation when $A = B$, $A' = B'$ and $\mathrm{Pol}(\mathbf{A}, \mathbf{A}')$ and $\mathrm{Pol}(\mathbf{B}, \mathbf{B}')$ have the same functions of essential arity at least m, for some m, (but there is assumption on functions of essential arity less than m in the two minions). Then Corollary 4.4 implies that $\mathrm{PCSP}(\mathbf{A}, \mathbf{A}')$ and $\mathrm{PCSP}(\mathbf{B}, \mathbf{B}')$ are reducible to each other.

The proof of Corollary 4.4 is straightforward, we can extend the partial homomorphism ξ to a full d-homomorphism, where d is the number of m-ary functions in $\mathrm{Pol}(\mathbf{A}, \mathbf{A}')$. To define the extension, map any n-ary function $f \in \mathrm{Pol}(\mathbf{B}, \mathbf{B}')$ of essential arity less then m to the set of all n-ary functions in $\mathrm{Pol}(\mathbf{A}, \mathbf{A}')$ whose set of essential coordinates is contained in the set of essential coordinates of f. It is not hard to check that this indeed gives a d-homomorphism from $\mathrm{Pol}(\mathbf{B}, \mathbf{B}')$ to $\mathrm{Pol}(\mathbf{A}, \mathbf{A}')$, and one can apply Theorem 4.3.

An argument similar to the one before Theorem 4.3 gives the following.

Corollary 4.5. *Let $\mathscr{A} = \mathrm{Pol}(\mathbf{A}, \mathbf{A}')$ and $\mathscr{B} = \mathrm{Pol}(\mathbf{B}, \mathbf{B}')$ If there is r and a collection of r partial minion d-homomorphism $\xi_i \colon \mathscr{B}_i \to \mathscr{A}$ where $\mathscr{B}_i \subseteq \mathscr{B}$ for each $i = 0, \dots, r-1$, and $\mathscr{B} = \bigcup_{i=0}^{r-1} \mathscr{B}_i$ then $\mathrm{PCSP}(\mathbf{A}, \mathbf{A}')$ is reducible to $\mathrm{PCSP}(\mathbf{B}, \mathbf{B}')$ by k-reduction for some k.*

One open question relating (d,r)-homomorphisms with the arc-graph reduction is whether there is a (d,r)-homomorphism for some d and r from $\mathrm{Pol}(\delta\mathbf{G}, \mathbf{H})$ to $\mathrm{Pol}(\mathbf{G}, \delta_R\mathbf{H})$ for any digraphs $\mathbf{G}$ and $\mathbf{H}$. We know that the arc-graph reduction is a reduction between the corresponding PCSPs. In fact,

it is possible to show using the construction δ_R that 3-reduction also provides such a reduction in this case. Nevertheless, it is not clear that the sufficient condition from Theorem 4.3 is satisfied in this case. If it is not, this would mean that (d, r)-homomorphisms are not necessary for k-reduction to work.

We are far from understanding reductions (beyond gadget reductions) in many ways: Is k-reduction sufficient to build a strong enough general theory for PCSPs? Is there a characterisation when a k-reduction is an actual reduction from one PCSP to another in terms of the polymorphism minions of the two templates? Would a more general reduction be more open to such a characterisation?

4.3 Some open questions and directions

- Find a class of reductions between PCSPs that is strong enough (say, to cover more than gadget reductions) together with a characterisation when such a reduction is applicable.
- Can reductions given by adjunctions be described in terms of polymorphisms? The special case of the arc-graph construction is a good place to start.
- Is (d, r)-homomorphism also a necessary condition for the existence of a k-reduction? If not, find a more general (polymorphism-based) condition that would be both sufficient and necessary for such reductions.
- Is there a (d, r)-homomorphism for some d and r from $\mathrm{Pol}(\delta\mathbf{G}, \mathbf{H})$ to $\mathrm{Pol}(\mathbf{G}, \delta_R\mathbf{H})$ for all (or some interesting) digraphs $\mathbf{G}$ and $\mathbf{H}$?
- It is known [Krokhin et al. 2020] that each minion of polymorphisms carries a topological structure. Investigate how this topological structure can be used to provide more reductions between PCSPs.

5 "LIMITED ENOUGH" POLYMORPHISMS AND NP-HARDNESS

For CSPs, we know exactly which problems $\mathrm{CSP}(\mathbf{B})$ are NP-hard (assuming P ≠ NP), by [Bulatov 2017; Zhuk 2017, 2020]. In fact, the hardness part of the CSP Dichotomy Theorem is easy and has long been known [Bulatov et al. 2005]. One way to characterise the NP-hard CSPs is as follows [Barto et al. 2018]. Recall the minion $\mathscr{P}$ of all projections on $\{0, 1\}$ from Section 2.2.3. Then the NP-complete CSPs are the problems $\mathrm{CSP}(\mathbf{B})$ such that there exists a minion homomorphism from $\mathrm{Pol}(\mathbf{B})$ to $\mathscr{P}$. Since, as we mentioned in Section 2.2.3, $\mathscr{P}$ is the polymorphism minion of the structure corresponding to 3-Sat, this means that every NP-hard $\mathrm{CSP}(\mathbf{B})$ admits a gadget reduction from 3-Sat (unless P = NP). Since every projection of a given arity n is uniquely determined by one of its coordinates, the existence of such a minion homomorphism means exactly that, for each function f in $\mathrm{Pol}(\mathbf{B})$, one can select one "important" coordinate i_f in such a way that if $f = g^\pi$ then $i_f = \pi(i_g)$.

For PCSPs, the situation is more complicated, since, as mentioned before, there are many NP-hard PCSPs whose polymorphism minion does not admits a minion homomorphism to $\mathscr{P}$ (and hence there is no gadget reduction from 3-Sat). In fact, there is some belief [Brakensiek and Guruswami 2021] that, unlike for CSPs, understanding hardness in PCSPs will be at least as difficult as understanding tractability (regardless of the question about the existence of dichotomy for PCSPs).

5.1 Proving NP-hardness results via polymorphisms

A prevalent (so far) use of Theorem 4.3 to prove NP-hardness of PCSPs is to show a reduction from 3-Sat, i.e., with $\mathbf{A} = \mathbf{A}'$ being the template of 3-Sat, and hence $\mathrm{Pol}(\mathbf{A}, \mathbf{A}') = \mathscr{P}$.

Corollary 5.1. *Let* $(\mathbf{B}, \mathbf{B}')$ *be a PCSP template. If, for some fixed* $d, r \geq 1$*, there is a* (d, r)*-homomorphism from* $\mathrm{Pol}(\mathbf{B}, \mathbf{B}')$ *to* $\mathscr{P}$ *then* $\mathrm{PCSP}(\mathbf{B}, \mathbf{B}')$ *is NP-hard.*

Corollary 5.1 for the case $r = 1$ was proved in [Barto et al. 2021b] (the core of the argument appeared in [Austrin et al. 2017]) and for the case $d = 1$ and arbitrary r in [Brandts et al. 2021], but both of these results predate [Barto and Kozik 2022] and their original proofs relied on the PCP theorem.

Let us spell out what the condition in the above corollary says. Fix a (d, r)-homomorphism $\xi\colon \mathrm{Pol}(\mathbf{B}, \mathbf{B}') \to \mathscr{P}$. If f is an n-ary function in $\mathrm{Pol}(\mathbf{B}, \mathbf{B}')$ then $\xi(f)$ is a non-empty collection of (at most) d projections $p_n^{s_1}, \ldots, p_n^{s_d}$ of arity n, where $1 \leq s_j \leq n$ for each j. In other words, ξ can be viewed as selecting a set $\mathrm{sel}(f) = \{s_1, \ldots, s_d\} \subseteq [n]$ of (at most) d "special" coordinates in f. Now assume that we have a chain of minors $t_0, \ldots, t_r \in \mathrm{Pol}(\mathbf{B}, \mathbf{B}')$ such that $t_j = t_i^{\pi_{i,j}}$, for each $i < j$, where $\pi_{i,k} = \pi_{i,j} \circ \pi_{j,k}$, for all $i < j < k$. Then there must exist $i < j$ such that $\mathrm{sel}(t_j) \cap \pi_{i,j}(\mathrm{sel}(t_i)) \neq \emptyset$. In other words, it cannot be that, for all i, j, the image under $\pi_{i,j}$ of the "special" coordinates in t_i does not contain any "special" coordinates of t_j.

Informally, the fact that one can identify a bounded number of special coordinates in every polymorphism (in a coordinated way) shows that these polymorphisms are in some way "lopsided", and so $\mathrm{Pol}(\mathbf{B}, \mathbf{B}')$ is "limited enough". To contrast this intuition with "rich enough" polymorphisms from Section 3, let us explain why there cannot be a (d, r)-homomorphism from $\mathrm{Pol}(\mathbf{B}, \mathbf{B}')$ to $\mathscr{P}$ if $\mathrm{Pol}(\mathbf{B}, \mathbf{B}')$ contains symmetric functions of all arities. Indeed, assume such a (d, r)-homomorphism ξ exists for some d, r. Choose any symmetric function t_0 of arity $d \cdot (r + 1)$ in $\mathrm{Pol}(\mathbf{B}, \mathbf{B}')$ and let $S = sel(t_0)$ be the set of (at most) d special coordinates of t_0 selected by ξ. Choose a permutation π on $[d \cdot (r + 1)]$ so that S intersects none of the sets $\pi(S), \pi^2(S), \ldots, \pi^{d(r+1)-1}(S)$, clearly this can be done. Consider the chain of minors $t_0, \ldots, t_r$ where $t_i = t_0^{\pi^i}$ (and so $\pi_{i,j} = \pi^{j-i}$ for all $i < j$). Note that we have $t_0 = t_1 = \ldots = t_r$, since t_0 is symmetric, but $S = \mathrm{sel}(t_j)$ is disjoint from $\pi_{i,j}(\mathrm{sel}(t_i)) = \pi^{j-i}(S)$ for all $i < j$, by the choice of π.

Let us now discuss how Corollary 5.1 is applied. Naturally, it is all about selecting special sets of coordinates in polymorphisms. How exactly to select the special variables for a given application is the creative part.

Let's start with the case $r = 1$. In this case we want to select, for each polymorphism f, a set $\mathrm{sel}(f)$ of (at most) d "special" coordinates in such a way that, for every minor relation $t_1 = t_0^{\pi}$ in $\mathrm{Pol}(\mathbf{B}, \mathbf{B}')$, the set of special coordinates in t_0, after applying π, i.e., $\pi(\mathrm{sel}(t_0))$, shares at least one element with the set $\mathrm{sel}(t_1)$.

Example 5.2. Assume that we can prove that each $f \in \mathrm{Pol}(\mathbf{B}, \mathbf{B}')$ has at least one and at most d essential variables (which means that the values of the remaining, non-essential, variables have no effect on the output of f). We can set $\mathrm{sel}(f)$ to be the set of essential variables of f. In this case it is easy to check that if $t_1 = t_0^{\pi}$ then $\mathrm{sel}(t_1)$ and $\pi(\mathrm{sel}(t_0))$ can never be disjoint. This reasoning was used, for example, in [Austrin et al. 2017] to prove NP-hardness of $(2 + \varepsilon)$-Sat problems (see Example 2.4) — though via the PCP theorem.

The case $r = 1$ can be applicable even when polymorphisms have an unbounded number of essential variables — the special variables can be selected on the basis of some other property, as in Example 2.18 or in the following example.

Example 5.3. Let $\mathbf{C}_{2k+1}$ be an undirected cycle with $2k + 1$ nodes, $k \geq 1$, and let $\mathbf{K}_3$ be a 3-clique. The approximate graph homomorphism problem $\mathrm{PCSP}(\mathbf{C}_{2k+1}, \mathbf{K}_3)$ (recall Example 2.7) has been proved NP-hard in [Krokhin et al. 2020] by applying Corollary 5.1 as follows. One can associate a topological space to every graph. The space associated with an odd cycle is a circle, and the space associated with the n-th power of an odd cycle is an n-torus. Furthermore, any graph homomorphism

f, and hence also a polymorphism, induces a continuous map $\hat{f}$ between the corresponding spaces, hence we assign to a polymorphism $f\colon \mathbf{C}^n_{2k+1} \to \mathbf{K}_3$ a continuous map $\hat{f}$ from n-torus to the circle. Observing that the main topological invariant of a continuous map from a circle to itself is its degree (or winding number), one can then prove, roughly, that $\hat{f}$ will have a bounded number of coordinates with non-zero degree. These coordinates are then chosen to form the set $\mathrm{sel}(f)$, and one can prove that $\mathrm{sel}(t_1) \cap \pi(\mathrm{sel}(t_0)) \neq \emptyset$ holds whenever $t_1 = t_0^\pi$.

The previous example is a rare (so far) case when the analysis of polymorphisms in a hardness proof for PCSP is performed not by direct combinatorics, but instead by first mapping them by a minion homomorphism to a different abstract minion, where the behaviour of their images is easier to understand. It would be of significant interest to find more examples of hardness of proofs of this sort, where the target abstract minion can be of topological or of some other nature.

Now let us discuss applications of Corollary 5.1 with $r > 1$. One useful pattern is provided in the following example, taken from [Brandts et al. 2021], where it is applied to prove NP-hardness of certain generalisations of $(2+\varepsilon)$-Sat to non-Boolean domains.

Example 5.4. Assume that we can define the notion of a set of coordinates, called *smug sets* in [Brandts et al. 2021], for functions in $\mathrm{Pol}(\mathbf{B}, \mathbf{B}')$ so that the following conditions are satisfied:

- every $f \in \mathrm{Pol}(\mathbf{B}, \mathbf{B}')$ has a smug set consisting of at most d coordinates;
- no function in $\mathrm{Pol}(\mathbf{B}, \mathbf{B}')$ can have more than r pairwise disjoint smug sets;
- if $f = g^\pi$ for some $g, f \in \mathrm{Pol}(\mathbf{B}, \mathbf{B}')$ then the π-preimage of any smug set in f is a smug set in g.

Then, for any $f \in \mathrm{Pol}(\mathbf{B}, \mathbf{B}')$, we can set $\mathrm{sel}(f)$ to be an arbitrary smug set of variables of size at most d in it. This does not in general guarantee that $\mathrm{sel}(t_1) \cap \pi(\mathrm{sel}(t_0)) \neq \emptyset$ whenever $t_1 = t_0^\pi$. However, the last two conditions in the definition of a smug set imply that, for any chain of minors $t_0, \dots, t_r \in \mathrm{Pol}(\mathbf{B}, \mathbf{B}')$, it is impossible that $\mathrm{sel}(t_j) \cap \pi_{i,j}(\mathrm{sel}(t_i)) = \emptyset$ for all $i < j$.

Another useful pattern of applying Corollary 5.1 with $r > 1$ is when functions in $\mathrm{Pol}(\mathbf{B}, \mathbf{B}')$ can be shown to exhibit r different types of behaviour, so that $\mathrm{Pol}(\mathbf{B}, \mathbf{B}')$ can be divided into r (not necessarily disjoint) types, where the definition of $\mathrm{sel}(f)$ depends on the type of a polymorphism, and one can show that $\mathrm{sel}(g) \cap \pi(\mathrm{sel}(f)) \neq \emptyset$ holds whenever $g = f^\pi$ and g and f are of the same type. This can also be described as representing $\mathrm{Pol}(\mathbf{B}, \mathbf{B}')$ as $\mathrm{Pol}(\mathbf{B}, \mathbf{B}') = \mathscr{M}_1 \cup \dots \cup \mathscr{M}_r$, and providing partial d-homomorphisms $\xi_i\colon \mathscr{M}_i \to \mathscr{P}$ (see Definition 4.2 and the discussion after it).

Example 5.5. A few simple applications of the above pattern can be found in [Barto et al. 2021a], where it is applied to problems $\mathrm{PCSP}(\mathbf{T}, \mathbf{B})$ (as in Example 2.5) with $\mathbf{B}$ being a structure with domain $\{0,1,2\}$. For example, consider the case when the relation of $\mathbf{B}$ consists of triples $(0,0,1)$, $(0,0,2)$, $(1,1,2)$ and all triples obtained from them by permuting coordinates. For a function $f : \{0,1\}^n \to \{0,1,2\}$, call a set X of coordinates of f a 2-set if f evaluates to 2 whenever all variables in X are equal to 1. Also, define $E(f)$ to be the set of all coordinates i such that $f(\mathbf{x}_i) \neq 0$, where $\mathbf{x}_i$ is the tuple where 1 appears only in position i. It is shown in [Barto et al. 2021a] that each $f \in \mathrm{Pol}(\mathbf{T}, \mathbf{B})$ has at least one of the following properties: f has a 2-set X of size at most 2 or the set $E(f)$ has size at most 5. Whichever case applies for each f, select the corresponding set, X or $E(f)$, to be $\mathrm{sel}(f)$, breaking ties arbitrarily when there is any choice. Thus, we have $\mathrm{Pol}(\mathbf{T}, \mathbf{B}) = \mathscr{M}_1 \cup \mathscr{M}_2$, where $\mathscr{M}_i$ consists all functions $f \in \mathrm{Pol}(\mathbf{T}, \mathbf{B})$ such that $\mathrm{sel}(f)$ is defined by the first or by the second property above. It is then shown that this provides partial 5-homomorphisms from each $\mathscr{M}_i$ to $\mathscr{P}$, and hence a $(5,2)$-homomorphism from $\mathrm{Pol}(\mathbf{T}, \mathbf{B})$ to $\mathscr{P}$.

NP-hardness results for various Boolean PCSPs that can be obtained by using Corollary 5.1 (even though the original proofs are not always presented that way) can be found in [Brakensiek and Guruswami 2021; Brakensiek et al. 2021; Brandts and Živný 2021; Ficak et al. 2019].

5.2 Hardness of PCSPs from Gap Label Cover

NP-hardness of the Gap Label Cover problem is the starting point of many hardness proofs in inapproximability. Many hardness results for graph and hypergraph colouring were obtained via ad hoc reductions from Gap Label Cover, and will discuss some of them here (see [Barto et al. 2021b, Section 5] for more details). We remark that one advantage of hardness proofs via Corollary 5.1 (or similar), as compared with reductions from Gap Label Cover is that the reduction itself is not ad hoc, and thus is more suitable for building a general theory.

We start with two examples where the original hardness proofs used Gap Label Cover, but subsequently new proofs via Corollary 5.1 were found.

Example 5.6. Recall the hypergraph colouring problem $\mathrm{PCSP}(\mathbf{H}_k, \mathbf{H}_c)$ from Example 2.8. It was proved in [Dinur et al. 2005] that $\mathrm{PCSP}(\mathbf{H}_k, \mathbf{H}_c)$ is NP-hard for all $2 \le k \le c$, by using a reduction from a multi-layered version of Gap Label Cover. It was then shown in [Barto et al. 2021b, Section 5.3] how the proof from [Dinur et al. 2005] can be directly translated into the language of polymorphisms. Wrochna [2022] refined this to provide a short combinatorial proof that there is a way to select special coordinates in functions from $\mathrm{Pol}(\mathbf{H}_2, \mathbf{H}_c)$ that gives a (d, r)-homomorphism from $\mathrm{Pol}(\mathbf{H}_2, \mathbf{H}_c)$ to $\mathscr{P}$ for appropriately chosen d and r, so this hardness result can now also be explained by using Corollary 5.1.

Example 5.7. Recall Example 2.21. For any $k \ge 3$, the problem $\mathrm{PCSP}(\mathbf{K}_k, \mathbf{K}_{2k-1})$ has been proved NP-hard in [Barto et al. 2021b] by providing a minion homomorphism from $\mathrm{Pol}(\mathbf{K}_k, \mathbf{K}_{2k-1})$ to $\mathrm{Pol}(\mathbf{H}_2, \mathbf{H}_c)$, where c is the number of binary functions in $\mathrm{Pol}(\mathbf{K}_k, \mathbf{K}_{2k-1})$. From the previous example, there is a (d, r)-homomorphism from $\mathrm{Pol}(\mathbf{H}_2, \mathbf{H}_c)$ to $\mathscr{P}$, for some d, r. It is easy to check that a composition of a minion homomorphism with a (d, r)-homomorphism is again a (d, r)-homomorphism, so we conclude that there is a (d, r)-homomorphism from $\mathrm{Pol}(\mathbf{K}_k, \mathbf{K}_{2k-1})$ to $\mathscr{P}$.

Let us now discuss hardness results for approximate graph colouring that have not been explained via polymorphisms so far. Notably, all state-of-the-art NP-hardness results for $\mathrm{Pol}(\mathbf{K}_k, \mathbf{K}_c)$ with $6 \le k \le c$ are in this category.

Example 5.8. The strongest known NP-hardness results for $\mathrm{PCSP}(\mathbf{K}_k, \mathbf{K}_c)$ with $k = 3, 4, 5$ are those from the previous example, with $c = 2k - 1$. In [Huang 2013], it was proved, by a reduction from Gap Label Cover, that $\mathrm{PCSP}(\mathbf{K}_k, \mathbf{K}_c)$ is NP-hard for $c = 2^{\Omega(k^{1/3})}$ and k large enough. By starting from Huang's result and repeatedly applying the arc-graph reduction (see Section 4.1), it was shown in [Krokhin et al. 2020] that $\mathrm{PCSP}(\mathbf{K}_k, \mathbf{K}_c)$ is NP-hard for all $k \ge 4$ and $c = \binom{k}{\lfloor k/2 \rfloor} - 1$ (which coincides with $2k - 1$ for $k = 5$ and is the strongest known result for $k \ge 6$). We note that neither Huang's result nor the arc-graph reduction currently have an explanation in terms of polymorphisms.

Another result from [Krokhin et al. 2020] obtained by using the arc-digraph construction is that it is enough to prove NP-hardness of $\mathrm{PCSP}(\mathbf{K}_k, \mathbf{K}_c)$ for some fixed $k \ge 3$ and all $c \ge k$, the same will then follow for all $3 \le k \le c$.

Other examples of NP-hardness results that were obtained by ad hoc reductions from Gap Label Cover (and cannot currently be explained in terms of polymorphisms) include results on rainbow vs. normal and strong vs. normal hypergraph colouring, see references in Example 2.8.

5.3 Some open questions and directions

- At the moment, Corollary 5.1 is the most general known formalisation of what it means for a PCSP to have "limited enough" polymorphisms (to guarantee NP-hardness). Find other (incomparable or more general) formalisations.
- We mentioned symmetric Boolean PCSPs in Section 3.2. All such PCSPs that are not tractable are NP-hard [Brakensiek and Guruswami 2021; Ficak et al. 2019]. Find other interesting examples of NP-hard Boolean (or non-Boolean) PCSPs.
- Which of the problems PCSP($\mathbf{T}, \mathbf{B}$) from Example 2.5 are NP-hard? Specifically, solve this for the case $\mathbf{B} = \mathbf{LO}_k$ whose domain is $\{0, 1, \dots, k-1\}, k \geq 3$, and the relation consists of all tuples (x, y, z) that have a unique maximal element (or, in other words, if two of x, y, z are equal then the third element is larger than the two equal ones). See [Barto et al. 2021a; Nakajima and Živný 2022] for related results.
- Pick your favourite structure $\mathbf{A}$ with NP-hard CSP($\mathbf{A}$) and describe all NP-hard problems PCSP($\mathbf{A}, \mathbf{B}$). More generally, classify the complexity of PCSP($\mathbf{A}, \mathbf{B}$) depending on $\mathbf{B}$.
- Recall that a smooth digraph is one that has neither sources nor sinks (in other words, each vertex has positive in- and out-degrees). See [Barto et al. 2017, 2009] for an explicit description of smooth digraphs $\mathbf{G}$ with tractable (NP-hard, resp.) CSP($\mathbf{G}$). Generalising the Brakensiek-Guruswami conjecture from Example 2.7, are all problems PCSP($\mathbf{G}, \mathbf{H}$) NP-hard when $\mathbf{G}$ and $\mathbf{H}$ are smooth digraphs with NP-hard CSPs?
- Further develop the topology-based approach mentioned in Example 5.3 to attack the approximate graph colouring problem (and possibly other PCSPs such as variants of hypergraph colouring mentioned in Example 2.8).
- Topology has been applied in hardness proofs for PCSPs in different ways: combinatorial analysis via Borsuk-Ulam-style theorems [Austrin et al. 2020; Dinur et al. 2005] and topological analysis of polymorphisms [Krokhin et al. 2020]. Investigate whether there is a common pattern behind these applications.
- In the introduction, we mentioned that the truth of some NP-hardness conjectures (related to variants of unique games) is known to imply NP-hardness of all approximate graph colouring problems. Are the reverse implications true for approximate graph colouring (or some other PCSPs)?
- Investigate logical inexpressibility for PCSPs (see [Atserias and Dalmau 2022] for the first example of work in this direction).

REFERENCES

Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. 1998. Proof verification and the hardness of approximation problems. *J. ACM* 45, 3 (1998), 501–555.

Sanjeev Arora and Shmuel Safra. 1998. Probabilistic Checking of Proofs: A New Characterization of NP. *J. ACM* 45, 1 (Jan. 1998), 70–122. https://doi.org/10.1145/273865.273901

Kristina Asimi and Libor Barto. 2021. Finitely Tractable Promise Constraint Satisfaction Problems. In *46th International Symposium on Mathematical Foundations of Computer Science, MFCS 2021 (LIPIcs)*. Dagstuhl, Germany, 11:1–11:16. https://doi.org/10.4230/LIPIcs.MFCS.2021.11

Albert Atserias, Andrei A. Bulatov, and Anuj Dawar. 2009. Affine systems of equations and counting infinitary logic. *Theor. Comput. Sci.* 410, 18 (2009), 1666–1683. https://doi.org/10.1016/j.tcs.2008.12.049

Albert Atserias and Víctor Dalmau. 2022. Promise Constraint Satisfaction and Width. In *Proc. ACM-SIAM Symposium on Discrete Algorithms, SODA 2022*. 1129–1153. https://doi.org/10.1137/1.9781611977073.48

Per Austrin, Amey Bhangale, and Aditya Potukuchi. 2020. Improved Inapproximability of Rainbow Coloring. In *Proc. ACM-SIAM Symposium on Discrete Algorithms, SODA 2020*. 1479–1495. https://doi.org/10.1137/1.9781611975994.90 arXiv:1810.02784

Per Austrin, Venkatesan Guruswami, and Johan Håstad. 2017. $(2+\varepsilon)$-Sat Is NP-hard. *SIAM J. Comput.* 46, 5 (2017), 1554–1573. https://doi.org/10.1137/15M1006507

Libor Barto, Diego Battistelli, and Kevin M. Berg. 2021a. Symmetric Promise Constraint Satisfaction Problems: Beyond the Boolean Case. In *Symp. on Theoretical Aspects of Computer Science, STACS 2021 (LIPIcs)*. Schloss Dagstuhl–LZI, Dagstuhl, Germany, 10:1–10:16. https://doi.org/10.4230/LIPIcs.STACS.2021.10

Libor Barto, Jakub Bulín, Andrei Krokhin, and Jakub Opršal. 2021b. Algebraic Approach to Promise Constraint Satisfaction. *J. ACM* 68, 4, Article 28 (Aug. 2021), 66 pages. https://doi.org/10.1145/3457606

Libor Barto and Marcin Kozik. 2014. Constraint Satisfaction Problems Solvable by Local Consistency Methods. *J. ACM* 61, 1 (Jan. 2014), 3:1–3:19. https://doi.org/10.1145/2556646

Libor Barto and Marcin Kozik. 2022. Combinatorial Gap Theorem and Reductions between Promise CSPs. In *Proc. ACM-SIAM Symposium on Discrete Algorithms, SODA 2022*. SIAM, Virtual Conference, 1204–1220. https://doi.org/10.1137/1.9781611977073.50

Libor Barto, Marcin Kozik, and Todd Niven. 2009. 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 (2009), 1782–1802. https://doi.org/10.1137/070708093

Libor Barto, Andrei Krokhin, and Ross Willard. 2017. *Polymorphisms, and How to Use Them*, 1–44. In Krokhin and Živný [Krokhin and Živný 2017]. https://doi.org/10.4230/DFU.Vol7.15301.1

Libor Barto, Jakub Opršal, and Michael Pinsker. 2018. The wonderland of reflections. *Israel Journal of Mathematics* 223, 1 (Feb 2018), 363–398. https://doi.org/10.1007/s11856-017-1621-9

Manuel Bodirsky. 2021. *Complexity of Infinite-Domain Constraint Satisfaction*. Cambridge University Press. https://doi.org/10.1017/9781107337534

Joshua Brakensiek and Venkatesan Guruswami. 2016. New Hardness Results for Graph and Hypergraph Colorings. In *Conference on Computational Complexity (CCC 2016) (LIPIcs, Vol. 50)*, Ran Raz (Ed.). Schloss Dagstuhl–LZI, Dagstuhl, Germany, 14:1–14:27. https://doi.org/10.4230/LIPIcs.CCC.2016.14

Joshua Brakensiek and Venkatesan Guruswami. 2021. Promise Constraint Satisfaction: Algebraic Structure and a Symmetric Boolean Dichotomy. *SIAM J. Comput.* 50, 6 (2021), 1663–1700. https://doi.org/10.1137/19M128212X

Joshua Brakensiek, Venkatesan Guruswami, and Sai Sandeep. 2021. Conditional Dichotomy of Boolean Ordered Promise CSPs. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021*. 37:1–37:15. https://doi.org/10.4230/LIPIcs.ICALP.2021.37

Joshua Brakensiek, Venkatesan Guruswami, Marcin Wrochna, and Stanislav Živný. 2020. The Power of the Combined Basic LP and Affine Relaxation for Promise CSPs. *SIAM J. Comput.* 49, 6 (2020), 1232–1248. https://doi.org/10.1137/20M1312745 arXiv:1907.04383v3

Alex Brandts, Marcin Wrochna, and Stanislav Živný. 2021. The Complexity of Promise SAT on Non-Boolean Domains. *ACM Trans. Comput. Theory* 13, 4 (2021), 26:1–26:20. https://doi.org/10.1145/3470867

Alex Brandts and Stanislav Živný. 2021. Beyond PCSP(1-in-3, NAE). In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021*. 121:1–121:14. https://doi.org/10.4230/LIPIcs.ICALP.2021.121

Mark Braverman, Subhash Khot, Noam Lifshitz, and Dor Minzer. 2022. An Invariance Principle for the Multi-slice, with Applications. In *IEEE 62nd Annual Symposium on Foundations of Computer Science, FOCS 2021*. 228–236. https://doi.org/10.1109/FOCS52979.2021.00030

Andrei Bulatov, Peter Jeavons, and Andrei Krokhin. 2005. Classifying the Complexity of Constraints Using Finite Algebras. *SIAM J. Comput.* 34, 3 (March 2005), 720–742. https://doi.org/10.1137/S0097539700376676

Andrei A. Bulatov. 2017. A Dichotomy Theorem for Nonuniform CSPs. In *IEEE 58th Annual Symposium on Foundations of Computer Science, FOCS 2017*. 319–330. https://doi.org/10.1109/FOCS.2017.37

Andrei A. Bulatov and Víctor Dalmau. 2006. A Simple Algorithm for Mal'tsev Constraints. *SIAM J. Comput.* 36, 1 (2006), 16–27. https://doi.org/10.1137/050628957

Andrei A. Bulatov, Andrei Krokhin, and Benoit Larose. 2008. Dualities for Constraint Satisfaction Problems. In *Complexity of Constraints: An Overview of Current Research Themes*. Springer, Berlin, Heidelberg, 93–124. https://doi.org/10.1007/978-3-540-92800-3_5

Jakub Bulín, Andrei Krokhin, and Jakub Opršal. 2019. Algebraic Approach to Promise Constraint Satisfaction. In *Proc. of the 51st Annual ACM-SIGACT Symposium on the Theory of Computing, STOC 2019*. ACM, New York, USA, 602–613. https://doi.org/10.1145/3313276.3316300

Lorenzo Ciardo and Stanislav Živný. 2022a. CLAP: A New Algorithm for Promise CSPs. In *Proc. ACM-SIAM Symposium on Discrete Algorithms, SODA 2022*. 1057–1068. https://doi.org/10.1137/1.9781611977073.46

Lorenzo Ciardo and Stanislav Živný. 2022b. The Sherali-Adams Hierarchy for Promise CSPs through Tensors. (2022). https://doi.org/10.48550/arXiv.2203.02478 arXiv:2203.02478

Irit Dinur, Elchanan Mossel, and Oded Regev. 2009. Conditional Hardness for Approximate Coloring. *SIAM J. Comput.* 39, 3 (2009), 843–873. https://doi.org/10.1137/07068062X

Irit Dinur, Oded Regev, and Clifford Smyth. 2005. The Hardness of 3-Uniform Hypergraph Coloring. *Combinatorica* 25, 5 (Sept. 2005), 519–535. https://doi.org/10.1007/s00493-005-0032-4

Tomás Feder and Moshe Y. Vardi. 1998. The Computational Structure of Monotone Monadic SNP and Constraint Satisfaction: A Study Through Datalog and Group Theory. *SIAM J. Comput.* 28, 1 (Feb. 1998), 57–104. https://doi.org/10.1137/S0097539794266766

Miron Ficak, Marcin Kozik, Miroslav Olšák, and Szymon Stankiewicz. 2019. Dichotomy for Symmetric Boolean PCSPs. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019 (LIPIcs, Vol. 132)*. Schloss Dagstuhl–LZI, Dagstuhl, Germany, 57:1–57:12. https://doi.org/10.4230/LIPIcs.ICALP.2019.57 arXiv:1904.12424

M. R. Garey and David S. Johnson. 1976. The Complexity of Near-Optimal Graph Coloring. *J. ACM* 23, 1 (1976), 43–49. https://doi.org/10.1145/321921.321926

Venkatesan Guruswami and Sanjeev Khanna. 2004. On the Hardness of 4-Coloring a 3-Colorable Graph. *SIAM Journal on Discrete Mathematics* 18, 1 (2004), 30–40. https://doi.org/10.1137/S0895480100376794

Venkatesan Guruswami and Euiwoong Lee. 2017. Strong inapproximability results on balanced rainbow-colorable hypergraphs. *Combinatorica* (14 Dec. 2017). https://doi.org/10.1007/s00493-016-3383-0

Venkatesan Guruswami and Sai Sandeep. 2020a. d-To-1 Hardness of Coloring 3-Colorable Graphs with $O(1)$ Colors. In *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020 (LIPIcs)*. Schloss Dagstuhl–LZI, Dagstuhl, Germany, 62:1–62:12. https://doi.org/10.4230/LIPIcs.ICALP.2020.62

Venkatesan Guruswami and Sai Sandeep. 2020b. Rainbow Coloring Hardness via Low Sensitivity Polymorphisms. *SIAM J. Discrete Math.* 34, 1 (2020), 520–537. https://doi.org/10.1137/19M127731X

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

Sangxia Huang. 2013. Improved Hardness of Approximating Chromatic Number. In *APPROX 2013 & RANDOM 2013*. Springer, Berlin, Heidelberg, 233–243. https://doi.org/10.1007/978-3-642-40328-_17

Paweł M. Idziak, Petar Marković, Ralph McKenzie, Matthew Valeriote, and Ross Willard. 2010. Tractability and Learnability Arising from Algebras with Few Subpowers. *SIAM J. Comput.* 39, 7 (2010), 3023–3037. https://doi.org/10.1137/090775646

Peter Jeavons, David Cohen, and Marc Gyssens. 1997. Closure Properties of Constraints. *J. ACM* 44, 4 (July 1997), 527–548. https://doi.org/10.1145/263867.263489

Ravindran Kannan and Achim Bachem. 1979. Polynomial Algorithms for Computing the Smith and Hermite Normal Forms of an Integer Matrix. *SIAM J. Comput.* 8, 4 (1979), 499–507. https://doi.org/10.1137/0208040

Ken-ichi Kawarabayashi and Mikkel Thorup. 2017. Coloring 3-Colorable Graphs with Less than $n^{1/5}$ Colors. *J. ACM* 64, 1 (2017), 4:1–4:23. https://doi.org/10.1145/3001582

Alexandr Kazda, Peter Mayr, and Dmitriy Zhuk. 2022. Small Promise CSPs that reduce to large CSPs. *Logical Methods in Computer Science* 18 (Aug. 2022). Issue 3. https://doi.org/10.46298/lmcs-18(3:25)2022 arXiv:2109.07924

Sanjeev Khanna, Nathan Linial, and Shmuel Safra. 2000. On the Hardness of Approximating the Chromatic Number. *Combinatorica* 20, 3 (01 Mar 2000), 393–415. https://doi.org/10.1007/s004930070013

Andrei Krokhin, Jakub Opršal, Marcin Wrochna, and Stanislav Živný. 2020. Topology and adjunction in promise constraint satisfaction. (2020). https://doi.org/10.48550/arXiv.2003.11351 arXiv:2003.11351

Andrei Krokhin and Stanislav Živný (Eds.). 2017. *The Constraint Satisfaction Problem: Complexity and Approximability*. Schloss Dagstuhl – LZI.

Tamio-Vesa Nakajima and Stanislav Živný. 2022. Linearly ordered colourings of hypergraphs. (2022). https://doi.org/10.48550/arXiv.2204.05628 arXiv:2204.05628

Ran Raz. 1998. A Parallel Repetition Theorem. *SIAM J. Comput.* 27, 3 (1998), 763–803. https://doi.org/10.1137/S0097539795280895

Thomas J. Schaefer. 1978. The Complexity of Satisfiability Problems. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing (STOC '78)*. ACM, New York, USA, 216–226. https://doi.org/10.1145/800133.804350

Hanif D. Sherali and Warren P. Adams. 1990. A Hierarchy of Relaxations Between the Continuous and Convex Hull Representations for Zero-One Programming Problems. *SIAM J. Discret. Math.* 3, 3 (1990), 411–430. https://doi.org/10.1137/0403036

Marcin Wrochna. 2022. A note on hardness of promise hypergraph colouring. (2022). https://doi.org/10.48550/arXiv.2205.14719 arXiv:2205.14719

Dmitriy Zhuk. 2017. A Proof of CSP Dichotomy Conjecture. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*. 331–342. https://doi.org/10.1109/FOCS.2017.38

Dmitriy Zhuk. 2020. A Proof of the CSP Dichotomy Conjecture. *J. ACM* 67, 5 (Aug. 2020), 30:1–30:78. https://doi.org/10.1145/3402029

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il