Promise constraint satisfaction problems

Alex Brandts
St Hugh's College
University of Oxford

Thesis submitted for the degree of
Doctor of Philosophy
Trinity 2022

Abstract

The promise constraint satisfaction problem (PCSP) is a recently introduced vast generalisation of the constraint satisfaction problem (CSP) that captures approximability of satisfiable instances. A PCSP instance comes with two forms of each constraint: a strict one and a weak one. Given the promise that a solution exists under the strict constraints, the task is to find a solution under the weak constraints.

Austrin, Guruswami, and Håstad [SICOMP'17] showed that the problem of distinguishing k-CNF formulas that are g-satisfiable (some assignment satisfies at least g literals in every clause) from those that are not even 1-satisfiable is NP-hard if $\frac{g}{k} < \frac{1}{2}$ and is in P otherwise. We study a generalisation of SAT on arbitrary finite domains, with clauses that are disjunctions of unary constraints, and establish analogous behaviour. Thus we give a dichotomy for a natural fragment of PCSPs on arbitrary finite domains. The hardness side is proved using the algebraic approach via a new general NP-hardness criterion on polymorphisms, which is based on a gap version of the Layered Label Cover problem. We show that previously used criteria are insufficient, and so this problem gives an interesting benchmark of algebraic techniques for proving hardness of approximation in problems such as PCSPs.

Next we turn to non-symmetric PCSPs. While there now exist several dichotomy results for fragments of PCSPs, they all consider PCSPs that are symmetric in some way. 1-in-3-SAT and Not-All-Equal-3-SAT are classic examples of Boolean symmetric CSPs. While both problems are NP-hard, Brakensiek and Guruswami showed [SICOMP'21] that given a satisfiable instance of 1-in-3-SAT, one can efficiently find a solution to the corresponding instance of (weaker) Not-All-Equal-3-SAT. In other words, the PCSP template $(\mathbf{1\text{-}in\text{-}3}, \mathbf{NAE})$ is tractable. We study PCSP templates obtained from the Boolean template $(\mathbf{t\text{-}in\text{-}k}, \mathbf{NAE})$ by either adding tuples to **t-in-k** or removing tuples from **NAE**. For the former, we obtain an "algorithmic dichotomy" and classify all templates as either tractable or not solvable by one of the strongest known algorithms for PCSPs, the combined basic LP and affine IP relaxation of Brakensiek et al. [SICOMP'20]. For the latter, we classify all templates as either tractable or NP-hard.

Finally, we investigate problems of the form $\mathrm{PCSP}(\mathbf{1\text{-}in\text{-}3}, \mathbf{B})$ for $\mathbf{B}$ over an arbitrary finite domain. Barto, Battistelli, and Berg [STACS'21] almost completely classified such templates over domain size three, and suggested that over arbitrary domains, $(\mathbf{1\text{-}in\text{-}3}, \mathbf{NAE})$ and $(\mathbf{1\text{-}in\text{-}3}, \mathbf{C}_3)$ might be the only tractable templates, where $\mathbf{C}_3$ is a relation representing a 3-cycle. Through a connection to

number theory, we exhibit an infinite family of tractable templates, of which (**1-in-3**, **NAE**) and (**1-in-3**, $\mathbf{C}_3$) are the simplest members. We conjecture that this family contains all tractable cases of PCSP(**1-in-3**, **B**), and we prove NP-hardness or non-solvability by BLP+AIP for certain PCSPs outside the family.

Acknowledgements

I would like to thank my advisor Standa Živný for his guidance and support, and my friends and family for providing entertainment over the last four years. I would also like to thank examiners for helpful feedback on this thesis: Peter Jeavons, Andrei Krokhin, Leslie Goldberg, and Rahul Santhanam.

Declarations

I certify that I have written this thesis by myself. Parts of the thesis have appeared in [21], [22], and [23].

Contents

Chapter 1

Introduction

How hard is it to find a 6-colouring of a 3-colourable graph? We do not know but believe it to be NP-hard. Despite sustained effort, the complexity of this *approximate graph colouring problem* remains unresolved almost 50 years after it was first studied by Garey and Johnson [43]. The current state of the art for approximate graph colouring is the NP-hardness of finding a 5-colouring of a 3-colourable graph [10].

Approximate graph colouring is an example of the promise constraint satisfaction problem (PCSP), which is a vast generalisation of the constraint satisfaction problem (CSP). Many fundamental computational problems such as graph colouring and Boolean satisfiability are captured by the framework of *fixed-template* CSPs, which have been the object of intense study [31, 32, 58] since a systematic study was initiated by Feder and Vardi [40]. Roughly, a CSP consists of a set of variables and a set of constraints over these variables where the goal is to assign values to the variables so that all constraints are satisfied. More formally, for a finite relational structure $\mathbf{A}$, the computational problem $\mathrm{CSP}(\mathbf{A})$ is as follows: Given a structure $\mathbf{X}$ similar to $\mathbf{A}$, is there a homomorphism from $\mathbf{X}$ to $\mathbf{A}$? If $\mathbf{A} = \mathbf{K}_3$ is a clique on 3 vertices, then $\mathrm{CSP}(\mathbf{A})$ is the graph 3-colouring problem.

A classic result of Schaefer [65] shows that, for any $\mathbf{A}$ on a 2-element set, $\mathrm{CSP}(\mathbf{A})$ is either solvable in polynomial time or NP-hard. Hell and Nešetřil [50] showed that this dichotomy also holds if $\mathbf{A}$ is a graph. Based on these two results and a connection to logic, Feder and Vardi famously conjectured that, for any finite $\mathbf{A}$, $\mathrm{CSP}(\mathbf{A})$ is either solvable in polynomial time or NP-hard [40]. Almost 20 years later, Bulatov [24] and Zhuk [68] independently proved the conjecture in the affirmative, both relying on the *algebraic approach* to CSPs [53, 25, 13]. This approach studies CSPs via their *polymorphisms*

– operations that represent symmetries of solution spaces – and derives its power from the fact that the complexity of a CSP is completely determined by its polymorphisms [25]. The intuition behind the algebraic approach is that symmetry in solution spaces can be used to design algorithms, whereas the absence of symmetry suggests a solution space too complex for efficient algorithms.

Since CSPs exhibit a complexity dichotomy, it follows from Ladner's Theorem [60] and the assumption P≠NP that there are problems in NP not captured by the CSP framework. This raises the question of whether there exists a more general class of problems exhibiting such a dichotomy, and gaining insight into this question is one of the motivations behind the study of PCSPs.

PCSPs can be viewed as CSPs with additional information – a promise – about their instances. In the decision version of a PCSP, the promise creates a gap between the YES and NO instances, and in the search version, we seek a solution to a problem that is guaranteed to have a solution in a strong sense. More formally, let $\mathbf{A}$ and $\mathbf{B}$ be two relational structures such that there is a homomorphism from $\mathbf{A}$ to $\mathbf{B}$, denoted by $\mathbf{A} \to \mathbf{B}$. The fixed-template PCSP over $\mathbf{A}$ and $\mathbf{B}$, denoted by $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$, is the following computational problem: Given $\mathbf{X}$, return YES if $\mathbf{X} \to \mathbf{A}$ and return NO if $\mathbf{X} \not\to \mathbf{B}$. This is the decision version; the search version is as follows: Given $\mathbf{X}$ such that $\mathbf{X} \to \mathbf{A}$, find a homomorphism from $\mathbf{X}$ to $\mathbf{B}$. If we take $\mathbf{A} = \mathbf{K}_3$ to be a clique on 3 vertices and $\mathbf{B} = \mathbf{K}_6$ to be a clique on 6 vertices, then $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is an instance of the approximate graph colouring problem mentioned at the beginning of this chapter. As discussed in Section 2.2, it is known that the decision version reduces to the search version, but no reduction the other way is currently known.

Austrin, Guruswami, and Håstad [8] adapted the definition of polymorphism to the setting of PCSPs and established a complexity dichotomy for a promise version of the Boolean satisfiability problem. In a series of papers [16, 19, 17], Brakensiek and Guruswami linked PCSPs to the universal-algebraic methods developed for the study of CSPs [13]. Then, building on the result of Barto, Opršal, and Pinsker [14] that the complexity of $\mathrm{CSP}(\mathbf{A})$ is captured by certain types of identities involving polymorphisms of $\mathbf{A}$, Barto, Bulín, Krokhin, and Opršal [10] showed that the algebraic approach developed for CSPs can be generalised to PCSPs, thus introducing a general methodology for investigating the computational complexity of PCSPs.

This theory was used in [19] to give a dichotomy for symmetric Boolean PCSPs allowing negation of variables, and then by Ficak, Kozik, Olšák, and Stankiewicz [41] to obtain a stronger dichotomy for all symmetric Boolean PCSPs. Further recent results on PCSPs include the work of Krokhin and Opršal [56], Brakensiek and Guruswami [18], and Austrin, Bhangale, and Potukuchi [7], as well as

recent progress on the approximate graph colouring problem [67] and the related approximate graph homomorphism problem [56, 67]. While the complexity of approximate graph colouring remains open, hardness was proved under stronger assumptions by Dinur, Mossel, and Regev [36], and Guruswami and Sandeep [49] recently established the same result under a weaker assumption known as the d-to-1 conjecture.

We now outline the problems studied in this thesis.

SetSAT While 3-SAT is NP-hard [30, 61], 2-SAT is solvable in polynomial-time [59]. Austrin, Guruswami, and Håstad [8] considered the promise problem (a, g, k)-SAT: Given a k-CNF formula with the promise that there is an assignment satisfying at least g literals in each clause, find an assignment satisfying at least $a \leq g$ literals in each clause. The general case easily reduces to the case $a = 1$, and they showed that $(1, g, k)$-SAT is NP-hard if $\frac{g}{k} < \frac{1}{2}$ and in P otherwise. Viewing k-SAT as $(1, 1, k)$-SAT, this shows that, in a natural sense, the transition from tractability to hardness occurs just after 2 and not just before 3.

The *set-satisfiability* (SetSAT) problem, first described in [45], generalises the Boolean satisfiability problem to larger domains, and we prove that it exhibits an analogous hardness transition. As in (a, g, k)-SAT, an instance of the (a, g, k)-SetSAT problem is a conjunction of clauses, where each clause is a disjunction of k literals. However, variables $x_1, \ldots, x_n$ can take values in $[d] = \{1, \ldots, d\}$, while literals take the form "$x_i \in S$", where S is any subset of the domain $[d]$ of size $1 \leq s < d$. As in the Boolean case, an assignment $\sigma \colon \{x_1, \ldots, x_n\} \to [d]$ is called *g-satisfying* if it satisfies at least g literals in every clause. In (a, g, k)-SetSAT with set size s and domain size d, given an instance promised to be g-satisfiable, our goal is to find an a-satisfying assignment. When $s = 1$ and $d = 2$ we recover Boolean promise SAT, whereas when $a = g = 1$ we recover the non-promise version of SetSAT.

The most natural case of SetSAT allows all nontrivial unary constraints (sets) as literals, i.e., the case $s = d - 1$. More generally one could consider the problem restricted to any family of literals. Our work deals with the "folded" case: If a set S is available as a literal, then for all permutations π of the domain, $\pi(S)$ is also available as a literal. In this case only the cardinality of S matters, and in fact only the maximum available cardinality matters, so all such problems are equivalent to (a, g, k)-SetSAT, for some constants a, g, k, s, d.

Our main motivation to study SetSAT as a promise problem is the fact that it constitutes a natural

fragment of PCSPs. Variants of the Boolean satisfiability problem over larger domains have been defined using CNFs by Gil, Hermann, Salzer, and Zanuttini [45] and DNFs by Chen and Grohe [27] but, as far as we are aware, have not been studied as promise problems.

We show that $(1, g, k)$-SetSAT is tractable if $\frac{g}{k} \geq \frac{s}{s+1}$ and is NP-hard otherwise. Following [8] and the abstract algebraic framework of [10], our hardness proof relies on the study of polymorphisms. In the Boolean case, the proof of [8] shows that every polymorphism depends on only a few variables, which suffices for a reduction from the Gap Label Cover problem. In our case, this condition does not hold, and neither do the various generalisations of it used in later work on PCSPs [41, 10, 56]. In fact, we show that SetSAT has significantly richer, more robust polymorphisms, which makes the application of many such conditions impossible. Our main technical contribution is a new condition that guarantees an NP-hardness reduction from a multilayered variant of the Gap Label Cover problem.

Beyond PCSP(**1-in-3**, **NAE**) In Chapter 4 we consider PCSPs that arise from combining (positive) 1-in-3-SAT and (positive) Not-All-Equal-3-SAT. For both problems, the instance is a list of triples of variables. In 1-in-3-SAT, the task is to find a mapping from the variables to $\{0, 1\}$ so that in each specified triple exactly one variable is set to 1; we denote this problem by CSP(**1-in-3**). In Not-All-Equal-3-SAT, which we denote by CSP(**NAE**), the task is to find a mapping from the variables to $\{0, 1\}$ so that in each triple not all variables are assigned the same value.

Unlike most previous works, which focused on symmetric PCSPs, we investigate *non-symmetric* PCSPs. Our first motivation is that a classification of more concrete PCSP templates is needed to improve and extend the general algebraic theory from [10], for example by identifying new hardness and tractability criteria. At the moment, even an analogue of Schaefer's Dichotomy Theorem for Boolean PCSPs seems out of reach. Our second motivation is the pure beauty of the template (**1-in-3**, **NAE**). While CSP(**1-in-3**) and CSP(**NAE**) are both NP-hard, PCSP(**1-in-3**, **NAE**) admits a polynomial-time algorithm [19, 17], and its tractability cannot be obtained via a "gadget reduction" to tractable finite-domain CSPs [10] or via "local consistency checking" algorithms [5].

Let **t-in-k** denote the Boolean structure with a single relation of arity k that contains all tuples with exactly t 1's, and let **NAE** denote the Boolean structure with a single relation of arity k that contains all tuples except for 0^k and 1^k. Then PCSP(**t-in-k**, **NAE**) is a tractable symmetric PCSP. We study the following two questions: When can we add tuples to **t-in-k** (thus weakening the promise) and keep the PCSP tractable? When can we remove tuples from **NAE** (again weakening the promise)

and keep the PCSP tractable? These changes generally do not result in symmetric templates.

We answer the second question completely: If t is odd, k is even, and tuples of only even weight are removed from **NAE**, the resulting PCSP is tractable. In all other cases, the resulting PCSP is NP-hard. Put differently, PCSP(**t-in-k**, $\mathbf{B}$) is tractable if and only if $\mathbf{B} = \mathbf{NAE}$ or CSP($\mathbf{B}$) is tractable (assuming P≠NP).

To the first question, we give a second-best possible answer: If t is odd, k is even, and tuples of only odd weight are added to **t-in-k**, the resulting PCSP is tractable. In all other cases, the resulting PCSP is not solved by the combined basic LP and affine IP relaxation (BLP + AIP) of Brakensiek, Guruswami, Wrochna, and Živný [20], one of the currently strongest known algorithm for PCSPs. We also show that these PCSPs are not solvable by "local consistency checking" algorithms nor via "gadget reductions" to tractable finite-domain CSPs.

PCSP(**1-in-3**, $\mathbf{B}$) **on non-Boolean domains** Recent work by Barto, Battistelli, and Berg [9] extended the study of PCSP(**1-in-3**, **NAE**) in a different direction: Instead of using relations of larger arity or doing away with symmetry, they considered templates of the form (**1-in-3**, $\mathbf{B}$) with $\mathbf{B}$ over an arbitrary finite domain. To any such relation $\mathbf{B}$ they associate a directed graph $G(\mathbf{B})$: The vertex set is the domain of $\mathbf{B}$, and there is a directed edge from u to v if $(u, u, v) \in \mathbf{B}$. A symmetric relation $\mathbf{B}$ is uniquely determined by such a graph modulo the presence of *rainbow tuples* (a, b, c) with $|\{a, b, c\}| = 3$. Over domain size three, [9] gave a complexity classification of PCSP(**1-in-3**, $\mathbf{B}$) for all but a single $\mathbf{B}$. All their tractable cases followed from $G(\mathbf{B})$ being a 2- or a 3-cycle, with the former corresponding to $\mathbf{B} = \mathbf{NAE}$. (A 1-cycle corresponds to the trivial case where $\mathbf{B}$ contains a constant tuple.)

Our main result is the discovery of an infinite family of tractable templates whose tractability arises from certain closed walks in $G(\mathbf{B})$. In particular, a closed walk of length $2^\ell 3^r$ for $\ell \in \{0, 1\}$ and $r \geq 0$ that is "compatible" with $G(\mathbf{B})$ guarantees tractability. This generalises the case where $G(\mathbf{B})$ is a 2- or a 3-cycle. Our proof uses a result that for certain lengths of walks, a pseudorandom number generator has full period and therefore distributes the values of polymorphisms in a balanced way. We conjecture that all tractable cases of PCSP(**1-in-3**, $\mathbf{B}$) arise in this way, and, alongside other evidence, we show that solvability by BLP+AIP implies the existence of long cycles in $G(\mathbf{B})$, which rules out such solvability for structures $\mathbf{B}$ with $G(\mathbf{B})$ acyclic.

When $\mathbf{B}$ contains no rainbow tuples, the promise gap of (**1-in-3**, $\mathbf{B}$) is smaller, thus potentially

making hardness results more accessible. It is easy to show that if $G(\mathbf{B})$ contains a cycle of length at most three, then it is tractable by AIP; we show that otherwise PCSP(**1-in-3**, $\mathbf{B}$) is not solvable by BLP+AIP. Finally, we show that for $d \geq 4$, PCSP(**1-in-3**, $\mathbf{B}$) is NP-hard when $G(\mathbf{B})$ is a cycle of length d and $\mathbf{B}$ contains no rainbow tuples.

Thesis outline In Chapter 2 we define our notation, give examples of CSPs and PCSPs, outline the algebraic theory of PCSPs, and state existing tractability and hardness results. Chapter 3 focuses on SetSAT, Chapter 4 on PCSPs related to (**t-in-k**, **NAE**), and Chapter 5 on PCSP(**1-in-3**, $\mathbf{B}$) for $\mathbf{B}$ over an arbitrary finite domain. In Chapter 6 we summarise our results and propose directions for future work.

Chapter 2

Background

We start by defining and providing examples of CSPs and PCSPs, and then discuss polymorphisms and minions and their implications for the complexity of PCSPs.

2.1 CSPs

Let $[n] = \{1, 2, \ldots, n\}$. For a set A, we call $R \subseteq A^k$ a relation of arity $\mathrm{ar}(R) = k$. We also say that a function $f : A^k \to B$ has arity $\mathrm{ar}(f) = k$. A *relational structure* $\mathbf{A}$ is a tuple $(A; R_1, \ldots, R_p)$ where each R_i is a relation on A. Unless otherwise stated, we assume that all relational structures are over a finite set and contain only a finite number of relations, each of which has finite arity.

Two relational structures $\mathbf{A} = (A; R_1, \ldots, R_p)$ and $\mathbf{B} = (B; S_1, \ldots, S_q)$ are *similar* if $p = q$ and $\mathrm{ar}(R_i) = \mathrm{ar}(S_i)$ for every $i \in [p]$. A map $h : A \to B$ between similar relational structures is called a *homomorphism* from $\mathbf{A}$ to $\mathbf{B}$, denoted by $h : \mathbf{A} \to \mathbf{B}$, if h preserves all relations, that is, for every $i \in [p]$ and every tuple $\mathbf{x} \in R_i$, we have $h(\mathbf{x}) \in S_i$, with h applied component-wise. The existence of a homomorphism from $\mathbf{A}$ to $\mathbf{B}$ is denoted by $\mathbf{A} \to \mathbf{B}$.

For a k-ary relation R on A, we denote its complement by $R^c = A^k \setminus R$, and for a relational structure $\mathbf{A}$, we denote by $\mathbf{A}^c$ the structure with relations R^c for each relation R in $\mathbf{A}$. We say that $\mathbf{A}$ is *symmetric* if for every relation $R \in \mathbf{A}$ and $(a_1, \ldots, a_{\mathrm{ar}(R)}) \in R$, we have $(a_{\pi(1)}, \ldots, a_{\pi(\mathrm{ar}(R))}) \in R$ for every permutation π.

We denote by $\leq_p$ a polynomial-time many-one reduction and by $\equiv_p$ a polynomial-time many-one equivalence. A computational problem is called *tractable* if all of its instances can be solved in time

polynomial in the instance size.

Example 2.1. Graphs and directed graphs (digraphs) are one of the best known examples of relational structures. A digraph $\mathbf{G}$ with vertex set V and edge set E has relational structure $(V; E)$ with the single binary relation E; for graphs, we also require that the edge relation be symmetric: $(u, v) \in E$ if and only if $(v, u) \in E$.

Example 2.2. We call a hypergraph k-uniform if it has exactly k vertices in every hyperedge. A k-uniform hypergraph has $(V; E)$ as its relational structure, where E is a symmetric k-ary relation on V.

Example 2.3. Logical formulas in conjunctive normal form (k-CNF) can be expressed as relational structures: For a set of variables V, we take the structure $(V; (S_{t_1 \dots t_k})_{t_i \in \{0,1\}})$, where $S_{t_1 \dots t_k}(x_1, \dots, x_k)$ is the relation consisting of all tuples that satisfy the clause containing x_i negated if $t_i = 1$ and x_i non-negated otherwise.

Relational structures are the building blocks of CSPs, as the next definition shows.

Definition 2.4. For a relational structure $\mathbf{A}$, we define two variants of the computational problem $\mathrm{CSP}(\mathbf{A})$. Given a relational structure $\mathbf{X}$ similar to $\mathbf{A}$, the *decision problem* $\mathrm{CSP_d}(\mathbf{A})$ of $\mathrm{CSP}(\mathbf{A})$ asks whether there exists a homomorphism $\mathbf{X} \to \mathbf{A}$. The *search problem* $\mathrm{CSP_s}(\mathbf{A})$ asks to find a homomorphism from $\mathbf{X}$ to $\mathbf{A}$. We call $\mathbf{A}$ the *template* of $\mathrm{CSP}(\mathbf{A})$.

For any template $\mathbf{A}$, $\mathrm{CSP_d}(\mathbf{A})$ reduces to $\mathrm{CSP_s}(\mathbf{A})$: A search algorithm for $\mathrm{CSP_s}(\mathbf{A})$ can be modified to a decision algorithm for $\mathrm{CSP_d}(\mathbf{A})$ by returning YES if a solution is found and NO otherwise, and so we say that *decision reduces to search.* The converse, that *search reduces to decision*, is known to hold for CSPs [29, 25]. Therefore, for the sake of complexity, there is no need to distinguish between the two, and we often refer to both problems simply by $\mathrm{CSP}(\mathbf{A})$.

Example 2.5. Let $\mathbf{G}$ be a relational structure representing a graph, as in Example 2.1, and let $\mathbf{K}_c = ([c]; \neq)$ be the relational structure of the complete graph on c vertices. Then $\mathrm{CSP_d}(\mathbf{K}_c)$ is the problem of deciding whether there exists a homomorphism from $\mathbf{G} \to \mathbf{K}_c$, that is, a c-colouring of $\mathbf{G}$.

Example 2.6. A generalisation of the graph colouring problem in Example 2.5 is the $\mathbf{H}$-colouring problem: Given a graph $\mathbf{G}$, determine whether there exists a homomorphism $\mathbf{G} \to \mathbf{H}$. This problem is in P if $\mathbf{H}$ is bipartite or has a loop and is NP-hard otherwise [50].

Example 2.7. For a directed graph $\mathbf{G}$, the *directed graph homomorphism problem* CSP($\mathbf{G}$) asks to decide, for a directed graph $\mathbf{X}$, whether $\mathbf{X} \to \mathbf{G}$. Every CSP is equivalent to such a problem for some $\mathbf{G}$ [40].

Graph colouring problems can be generalised to hypergraphs in many ways, some of which we now describe. We give definitions only for k-uniform hypergraphs, but some of the generalisations can also be made to non-uniform hypergraphs by adding relations of appropriate arities to the template.

Example 2.8. For $1 \leq t < k$, the t-in-k hypergraph colouring problem asks to find a $\{0,1\}$-colouring of a k-uniform hypergraph such that in each hyperedge, exactly t variables are set to 1. Formally, the problem is CSP(**t-in-k**) where $\mathbf{t\text{-}in\text{-}k} = (\{0,1\}; \{x \in \{0,1\}^k : |\{i : x_i = 1\}| = t\})$. This problem is sometimes referred to as (positive) t-in-k-SAT, and is NP-hard for $k \geq 3$ and $1 \leq t < k$ by Schaefer's Theorem [65], which is stated precisely in Theorem 2.32.

Example 2.9. The (non-monochromatic) hypergraph colouring problem asks to find a colouring of a k-uniform hypergraph such that no hyperedge contains vertices of only one colour. Formally, the problem is CSP($\mathbf{NAE}_c$) where $\mathbf{NAE}_c = ([c]; \{x \in [c]^k : |\{x_1, \ldots, x_k\}| > 1\})$. We will often drop the subscript and write simply **NAE** when working over the Boolean domain. In this case, CSP(**NAE**) is equivalent to positive Not-All-Equal-k-SAT, the problem of deciding satisfiability of a k-CNF formula without negations, where additionally each clause must contain both a 0 and a 1. By Theorem 2.32, this problem is NP-hard for $k \geq 3$.

Example 2.10. Let ϕ be a 3-CNF formula and let $\mathbf{A} = (\{0,1\}; S_{000}, S_{001}, \ldots, S_{111})$ with notation as in Example 2.3. Then CSP($\mathbf{A}$) is the problem of deciding whether there is an assignment to the variables of ϕ that satisfies all its clauses.

The 3-SAT problem from Example 2.10 can equivalently be defined using only four relations, which represent the number of negated literals in each clause:

$$R_0(x_1, x_2, x_3) = (x_1 \vee x_2 \vee x_3) \qquad R_1(x_1, x_2, x_3) = (\overline{x}_1 \vee x_2 \vee x_3)$$
$$R_2(x_1, x_2, x_3) = (\overline{x}_1 \vee \overline{x}_2 \vee x_3) \qquad R_3(x_1, x_2, x_3) = (\overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_3).$$

The other four clauses in Example 2.10 are obtained by permuting variables in the appropriate R_i; for example, $(\overline{x_1} \vee x_2 \vee \overline{x_3}) = R_2(x_1, x_3, x_2)$. It is well known that 3-SAT is NP-hard, and in fact, the complexity of CSP($\mathbf{A}$) for all $\mathbf{A}$ over the Boolean domain was resolved by Schaefer in 1978 [65].

Example 2.11. The SetSAT problem, first described in [45], is a generalisation of Boolean k-SAT to domains of size $d \geq 2$. The template of SetSAT is the same as that of SAT, except that each literal in a SetSAT clause is given by an s-element subset of the domain containing the values satisfying the literal. Therefore SetSAT is CSP($\mathbf{A}$) for $\mathbf{A} = ([d]; (R_{S_1 \dots S_k})_{S_i \subseteq [d], |S_i| = s})$, where $(a_1, \dots, a_k) \in R_{S_1 \dots S_k}$ if and only if $a_i \in S_i$ for at least one i. Boolean k-SAT is the case $d = 2$ and $s = 1$.

Definition 2.12. We say that a Boolean template $\mathbf{A}$ is *folded* if it contains the binary relation $\{(0,1),(1,0)\}$.

A folded Boolean template allows for negation of variables by specifying that two variables in an instance always take opposite values. For example, with $N = \{(0,1),(1,0)\}$, the constraint $N(x,y)$ is equivalent to saying that $y \neq x$.

To conclude our exposition of CSPs, we mention a more general version of the CSP which allows us to parameterise by both the source and target of a homomorphism. Let $\mathcal{X}$ and $\mathcal{Y}$ be collections of relational structures and define HOM($\mathcal{X}, \mathcal{Y}$) as follows: Given $X \in \mathcal{X}$ and $Y \in \mathcal{Y}$, decide whether $X \to Y$. Writing "$-$" for the collection of all relational structures, we have that HOM($-, \mathcal{Y}$) includes problems of the form CSP($\mathbf{A}$) described so far; for example, HOM($-, \{\mathbf{K}_3\}$) is the graph 3-colouring problem. As an example from the other side, HOM($\{\mathbf{K}_5\}, -$) is the problem of deciding whether a graph has a 5-clique. While HOM($\{\mathbf{G}\}, -$) is tractable for every fixed graph $\mathbf{G}$, this is not the case for HOM($-, \{\mathbf{G}\}$).

Foldedness and symmetry of templates are examples of *language restrictions* on CSPs. However, we may also impose *structural restrictions*, which limit the ways that variables can appear in constraints. It was shown in [26, 42] that HOM($\mathcal{X}, -$) is tractable if $\mathcal{X}$ is a class of structures of *bounded treewidth*, and then an extension by Dalmau et al. [33] to bounded treewidth of the core and subsequent work by Grohe [46] showed that, unless FPT = W[1], these are the only tractable cases. It is also natural to combine structural and language restrictions: In [39, 54], a complexity classification is obtained for HOM($\mathcal{X}, \mathcal{Y}$) where $\mathcal{X}$ is planar (in a natural sense) and $\mathcal{Y}$ consists of Boolean structures which are even delta-matroids.

2.2 PCSPs

Our central object of study is a class of computational problems that vastly generalises CSPs.

Definition 2.13. Let $(\mathbf{A}, \mathbf{B})$ be a pair of similar relational structures such that there is a homomorphism $\mathbf{A} \to \mathbf{B}$. The pair $(\mathbf{A}, \mathbf{B})$ is called the *template* of the *promise constraint satisfaction problem* $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$. The *decision problem* $\mathrm{PCSP_d}(\mathbf{A}, \mathbf{B})$ of $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is as follows: Given a relational structure $\mathbf{C}$ similar to $\mathbf{A}$ and $\mathbf{B}$, output YES if $\mathbf{X} \to \mathbf{A}$ and NO if $\mathbf{X} \not\to \mathbf{B}$. The *search problem* $\mathrm{PCSP_s}(\mathbf{A}, \mathbf{B})$ asks: Given a relational structure $\mathbf{X}$ similar to $\mathbf{A}$ and $\mathbf{B}$ such that $\mathbf{X} \to \mathbf{A}$, find a homomorphism $\mathbf{X} \to \mathbf{B}$.

The *promise* in the decision problem is that it is never the case that $\mathbf{X} \to \mathbf{B}$ but $\mathbf{X} \not\to \mathbf{A}$, and in the search problem, the promise is that $\mathbf{X} \to \mathbf{A}$.

Since $\mathrm{PCSP_d}(\mathbf{A}, \mathbf{A}) = \mathrm{CSP_d}(\mathbf{A})$ and $\mathrm{PCSP_s}(\mathbf{A}, \mathbf{A}) = \mathrm{CSP_s}(\mathbf{A})$, PCSPs indeed generalise CSPs.

As for CSPs, we have $\mathrm{PCSP_d}(\mathbf{A}, \mathbf{B}) \leq \mathrm{PCSP_s}(\mathbf{A}, \mathbf{B})$: Run the search algorithm for $\mathrm{PCSP_s}(\mathbf{A}, \mathbf{B})$ for its stated time, then check that the solution produced is indeed a homomorphism $\mathbf{X} \to \mathbf{B}$. If it is, then the promise guarantees that $\mathbf{X} \to \mathbf{A}$, so we return YES. If the search algorithm does not find a solution, then return NO. Unlike for CSPs, however, it is not known whether search reduces to decision, i.e., whether $\mathrm{PCSP_s}(\mathbf{A}, \mathbf{B}) \leq \mathrm{PCSP_d}(\mathbf{A}, \mathbf{B})$. Sometimes this issue is circumvented by proving hardness for the decision problem or tractability for the search problem, implying the hardness or tractability, respectively, of the other version. Unless otherwise stated, we use $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ to refer to $\mathrm{PCSP_d}(\mathbf{A}, \mathbf{B})$ in hardness results, and to $\mathrm{PCSP_s}(\mathbf{A}, \mathbf{B})$ in tractability results.

We now provide some examples of PCSPs.

Example 2.14. The *approximate graph colouring problem* [44] asks to distinguish graphs with small chromatic number from those with large chromatic number. More formally, $\mathrm{PCSP}(\mathbf{K}_{c_1}, \mathbf{K}_{c_2})$ is the problem of distinguishing between the two cases $\mathbf{G} \to \mathbf{K}_{c_1}$ and $\mathbf{G} \not\to \mathbf{K}_{c_2}$, where $\mathbf{K}_{c_1}$ and $\mathbf{K}_{c_2}$ have domains $[c_1]$ and $[c_2]$ and the binary relations $\neq_{c_1}$ and $\neq_{c_2}$, respectively. It is conjectured that $\mathrm{PCSP}(\mathbf{K}_{c_1}, \mathbf{K}_{c_2})$ is NP-hard for all $3 \leq c_1 \leq c_2$. Note that the structures $\mathbf{K}_{c_1}$ and $\mathbf{K}_{c_2}$ have different domains.

Example 2.15. One may consider more generally $\mathrm{PCSP}(\mathbf{G}, \mathbf{H})$ for graphs $\mathbf{G}$ and $\mathbf{H}$ with $\mathbf{G} \to \mathbf{H}$. It was shown in [56] that, when $\mathbf{G}$ is 3-colourable and non-bipartite, $\mathrm{PCSP}(\mathbf{G}, \mathbf{K}_3)$ is NP-hard. Intuitively, this means that there is no 3-colourable graph $\mathbf{G}$ that helps to detect 3-colourability via the existence of a homomorphism to $\mathbf{G}$. This result was recently extended to show hardness of $\mathrm{PCSP}(\mathbf{G}, \mathbf{H})$ when $\mathbf{H}$ is from a class of graphs containing $\mathbf{K}_3$; one such class is that of graphs with no cycle of length four [67] (c.f. [57]). In [19] it was conjectured that $\mathrm{PCSP}(\mathbf{G}, \mathbf{H})$ is NP-hard for all

non-bipartite loopless graphs $\mathbf{G}$ and $\mathbf{H}$.

Example 2.16. Let $\mathbf{G}$ and $\mathbf{H}$ be directed graphs with $\mathbf{G} \to \mathbf{H}$. Then PCSP($\mathbf{G}, \mathbf{H}$) asks to determine, for a directed graph $\mathbf{X}$, whether $\mathbf{X} \to \mathbf{G}$ or $\mathbf{X} \not\to \mathbf{H}$. Just as every CSP is equivalent to a directed graph homomorphism problem (Example 2.7), every PCSP is equivalent to a directed graph homomorphism *promise* problem PCSP($\mathbf{G}, \mathbf{H}$) for some $\mathbf{G}$ and $\mathbf{H}$ [19].

Promise hypergraph colouring problems have formed an important part of PCSP research [37, 47, 6, 48, 7], owing in part to their many variants. One such variant is the following.

Example 2.17. For a k-uniform hypergraph $\mathbf{H}$, decide whether $\mathbf{H}$ is 2-colourable or not even d-colourable. This problem is PCSP($\mathbf{NAE}_2, \mathbf{NAE}_d$), where $\mathbf{NAE}_d$ is the not all equal relation on $[d]^k$, and was shown to be NP-hard for all $k \geq 3$ and $d \geq 2$ in [37].

Alternatively, one may promise that a hypergraph has a colouring of low discrepancy: We say that a d-colouring has *discrepancy* Δ if for every hyperedge e and for all colours $i, j \in [d]$, the number of vertices of colour i in e is within Δ of the number of vertices of colour j in e. The problem is then to find a d-colouring of discrepancy $\Delta' \geq \Delta$ given that there exists a d-colouring of discrepancy Δ.

Or, the promise may be that there exists a *rainbow* colouring, in which every hyperedge contains a vertex of every colour, and the goal is to find a colouring satisfying some weaker property. Combining these colouring restrictions gives rise to a large collection of promise problems for hypergraphs.

Example 2.18. Recall the Boolean templates **1-in-3** and **NAE** defined in Examples 2.8 and 2.9, respectively. PCSP(**1-in-3**, **NAE**) corresponds to the (positive) 1-in-3-SAT vs NAE-SAT problem. Both 1-in-3-SAT and NAE-SAT are NP-hard by Theorem 2.32, but PCSP(**1-in-3**, **NAE**) is in P [19].

Example 2.19. We can generalise Example 2.18 to Boolean relations of arity $k > 3$: CSP(**t-in-k**) and CSP(**NAE**) are both NP-hard, again by Theorem 2.32, but PCSP(**t-in-k**, **NAE**) is tractable for all $1 \leq t < k$, as we show in Chapter 4. There we also explore the complexity of PCSPs obtained by adding tuples to **t-in-k** or removing tuples from **NAE**.

Example 2.20. Another generalisation of Example 2.18 involves templates of the form (**1-in-3**, $\mathbf{B}$), where $\mathbf{B}$ is a ternary template over an arbitrary finite domain. The study of these PCSPs was intiated in [9], and for $\mathbf{B}$ with domain size three, their complexity was almost completely resolved. To state these results, we define the following symmetric relations, where all permutations of each tuple are

also in the relation.

$$\mathbf{D}_1 = \{(0,0,1),(0,0,2),(0,1,2)\} \qquad \mathbf{LO}_3 = \{(0,0,1),(0,0,2),(1,1,2)\}$$
$$\mathbf{D}_2 = \{(0,0,1),(1,1,2),(0,1,2)\} \qquad \mathbf{LO}_3^+ = \{(0,0,1),(0,0,2),(1,1,2),(0,1,2)\}$$
$$\mathbf{C}_3 = \{(0,0,1),(1,1,2),(2,2,0)\}$$

In [9] the following is proved:

1. If $\mathbf{NAE} \to \mathbf{B}$ or $\mathbf{C}_3 \to \mathbf{B}$, then PCSP($\mathbf{1\text{-}in\text{-}3}, \mathbf{B}$) is in P
2. If $\mathbf{B} \to \mathbf{D}_1$, $\mathbf{D}_2$, or $\mathbf{LO}_3$, then PCSP($\mathbf{1\text{-}in\text{-}3}, \mathbf{B}$) is NP-hard.

This covers all cases except $\mathbf{B} = \mathbf{LO}_3^+$, for which it is conjectured that PCSP($\mathbf{1\text{-}in\text{-}3}, \mathbf{LO}_3^+$) is NP-hard. The template $\mathbf{LO}_3^+$ can be seen as a colouring where, if two colours in a tuple are the same, then the third colour is "higher". In Chapter 5, we investigate ($\mathbf{1\text{-}in\text{-}3}, \mathbf{B}$) for domain size four and beyond.

Example 2.21. The template $\mathbf{LO}_3^+$ from Example 2.20 was generalised to larger arities in [62]. Let $\mathbf{LO}_d^k$ be the structure with domain $[d]$ whose sole relation contains a tuple $(c_1, \ldots, c_k)$ if and only if the sequence $c_1, \ldots, c_k$ has a unique maximum. Among other results, [62] establishes the NP-hardness of PCSP($\mathbf{LO}_r^k, \mathbf{LO}_l^k$) for every $2 \leq r \leq l$ and $k \geq l - r + 4$.

Example 2.22. For $a \leq g \leq k$, the (a, g, k)-SAT problem is a promise version of the Boolean satisfiability problem that asks: Given a k-CNF formula, is there an assignment that satisfies at least g literals in each clause, or does every assignment satisfy fewer than a literals in each clause? Let $S_{t_1 \ldots t_k}(x_1, \ldots, x_k)$ be as in Example 2.3. Define the relation $R_{t_1 \ldots t_k}$ by $(a_1, \ldots, a_k) \in R_{t_1 \ldots t_k}$ if and only if $a_i \neq t_i$ for at least g values of i, and the relation $S_{t_1 \ldots t_k}$ by $(a_1, \ldots, a_k) \in S_{t_1 \ldots t_k}$ if and only if $a_i \neq t_i$ for at least a values of i. Then for $A = B = \{0, 1\}$ and for $\mathbf{A}$ and $\mathbf{B}$ containing all relations of the form $R_{t_1 \ldots t_k}$ and $S_{t_1 \ldots t_k}$, respectively, PCSP($\mathbf{A}, \mathbf{B}$) = (a, g, k)-SAT.

By Proposition 3.22, we can assume that $a = 1$ and consider only $(1, g, k)$-SAT. In [8] it was shown that (the search version of) $(1, g, k)$-SAT is in P when $\frac{g}{k} \geq 1/2$ and (that the decision version) is NP-hard otherwise, which shows that the transition from tractability to hardness in k-SAT occurs *just after* $k = 2$ and not *just before* $k = 3$.

Example 2.23. We define a promise version of SetSAT from Example 2.11 which will be studied extensively in Chapter 3. Let (a, g, k)-SetSAT be the promise problem PCSP$(\mathbf{A}, \mathbf{B})$ where $\mathbf{A}$ and $\mathbf{B}$ have domain $[d]$ and the same relations as in Example 2.11, except that in $\mathbf{A}$, we have $(a_1, \ldots, a_k) \in R_{S_1 \ldots S_k}$ if and only if $a_i \in S_i$ for at least g values of i, and $(a_1, \ldots, a_k) \in S_{S_1 \ldots S_k}$ if and only if $a_i \in S_i$ for at least a values of i. Again, it suffices to consider the cases $a = 1$ and $s = d - 1$, and we show that $(1, g, k)$-SetSAT is in P for $\frac{g}{k} \geq \frac{s}{s+1}$ and is NP-hard when $\frac{g}{k} < \frac{s}{s+1}$ [22].

Definition 2.24. For a PCSP template $(\mathbf{A}, \mathbf{B})$, if both $\mathbf{A}$ and $\mathbf{B}$ are symmetric (respectively folded), we say that $(\mathbf{A}, \mathbf{B})$ is a *symmetric* (respectively *folded*) PCSP template.

2.3 Polymorphisms

Essentially all our results on the complexity of PCSPs come from the study of polymorphisms and their algebraic properties. The connection between algebra and CSPs was made in [53, 51, 25], where the problem of classifying the complexity of CSPs was translated into the language of universal algebra. Austrin et al. [8] then adapted the notion of polymorphism to PCSPs, and a powerful algebraic theory of PCSPs was subsequently developed in [10].

Definition 2.25. Let $(\mathbf{A}, \mathbf{B})$ be a PCSP template. A function $f : A^m \to B$ is a *polymorphism* of $(\mathbf{A}, \mathbf{B})$ if for each pair of corresponding relations R_i and S_i from $\mathbf{A}$ and $\mathbf{B}$, respectively, the following holds: For any $(\mathrm{ar}(R_i) \times m)$ matrix M whose columns are tuples in R_i, the application of f to rows of M gives a tuple in S_i. In other words, an arity m polymorphism is a homomorphism from the m-th Cartesian power of $\mathbf{A}$ to $\mathbf{B}$. We denote by Pol$(\mathbf{A}, \mathbf{B})$ the set of polymorphisms of $(\mathbf{A}, \mathbf{B})$. The same definition applies in the case of a CSP template, where $\mathbf{A} = \mathbf{B}$, and we denote by Pol$(\mathbf{A})$ the set of polymorphisms of $(\mathbf{A}, \mathbf{A})$.

Every PCSP template $(\mathbf{A}, \mathbf{B})$ with $\mathbf{A} \to \mathbf{B}$ via ϕ has the operation $f(x_1, \ldots, x_m) = \phi(x_i)$ as a polymorphism: Given a matrix of inputs with columns from a relation R_j, f selects a column i and maps it via ϕ to a tuple in S_j. The membership in S_j holds since ϕ is a homomorphism.

Example 2.26. Suppose $(\mathbf{A}, \mathbf{B}) = ((A; R), (B, S))$ where R and S are ternary. The function f of arity 4 is a polymorphism of $(\mathbf{A}, \mathbf{B})$ if whenever all columns below are elements of R, the last column is an element of S.

$$\begin{array}{llllll} f(a_{11} & a_{12} & a_{13} & a_{14}) & = & b_1 \\ f(a_{21} & a_{22} & a_{23} & a_{24}) & = & b_2 \\ f(a_{31} & a_{32} & a_{33} & a_{34}) & = & b_3 \end{array}$$

We now define several classes of functions that occur frequently as polymorphisms of Boolean PCSP templates.

Definition 2.27. A function $f : \{0,1\}^m \to \{0,1\}$ is

- an OR_m (AND_m) if it returns the logical OR (respectively logical AND) of its arguments;
- an alternating threshold AT_m if $m \geq 1$ is odd and

$$f(x_1, \ldots, x_m) = 1 \text{ if and only if } x_1 - x_2 + x_3 - \cdots + x_m > 0;$$

- a parity function XOR_m if $f(x_1, \ldots, x_m) = x_1 + \cdots + x_m \mod 2$;
- a q-threshold $\mathrm{THR}_{q,m}$ (for q a rational between 0 and 1 and mq not an integer) if $f(x_1, \ldots, x_m) = 0$ if $\sum_{i=1}^{m} x_i < mq$ and 1 otherwise;
- a majority MAJ_m if $f = \mathrm{THR}_{\frac{1}{2},m}$ and m is odd.

We denote by OR and AND the set of all OR_m and AND_m functions, respectively, for $m \geq 2$. We denote by AT, XOR, and MAJ the set of all AT_m, XOR_m, and MAJ_m functions, respectively, for odd $m \geq 1$. Finally, THR_q denotes the set of all $\mathrm{THR}_{q,m}$ functions for $qm \notin \mathbb{Z}$.

Define $\overline{f}$, the *negation of* f, as the function $x \mapsto 1 - f(x)$, and for a family of functions F, define the *negation of* F by $\overline{F} = \{\overline{f} | f \in F\}$.

Example 2.28. The CSP template $(\{0,1\}; S_{00}, S_{01}, S_{10}, S_{11})$ of 2-SAT (defined in Example 2.3) has the ternary majority function MAJ_3 as a polymorphism. In fact, a Boolean template $\mathbf{A}$ has MAJ_3 as a polymorphism if and only if each relation in $\mathbf{A}$ can be expressed as a conjunction of 2-SAT clauses [52, 40].

Example 2.29. The CSP template over domain $\{0,1\}$ whose relations are the solutions in $\{0,1\}$ of equations of the form $x_1 + x_2 + \cdots + x_k = 0 \mod 2$ has the ternary parity function XOR_3 as a polymorphism. In a similar fashion to Example 2.28, a Boolean template $\mathbf{A}$ has XOR_3 as a polymorphism if and only if each relation in $\mathbf{A}$ is the solution set of a system of linear equations over $\{0,1\}$ [13].

Example 2.30. By a simple counting and averaging argument, one can show that $(1, g, 2g)$-SAT from Example 2.22 has the $\mathrm{THR}_{\frac{1}{2}}$ family of polymorphisms, whereas all polymorphisms of $(1, g, 2g+1)$-SAT depend on at most $2g-1$ inputs, and so not all functions in $\mathrm{THR}_{\frac{1}{2}}$ are polymorphisms [8].

The following type of function arises as a polymorphism of SetSAT from Example 2.11.

Definition 2.31. A function $f : A^m \to A$ is a plurality if

$$f(x_1, \dots, x_m) = \mathrm{argmax}_{a \in A}\{\#\text{ of occurrences of } a \text{ in } \{x_1, \dots, x_m\}\}.$$

In the tractable regime of SetSAT, there are plurality polymorphisms that depend on an arbitrarily large number of variables (Proposition 3.13), whereas in the hard regime, only pluralities of small arity can be polymorphisms (a consequence of Theorem 2.61).

To illustrate the power of polymorphisms in capturing complexity, we state a modern version of a classic result of Schaefer that resolved the complexity of all Boolean CSPs; see e.g. [13] for a discussion of the other forms of the theorem.

Theorem 2.32 (Schaefer's Dichotomy Theorem [65])**.** *Let* $\mathbf{A}$ *be a Boolean CSP template. Then* $\mathrm{CSP}(\mathbf{A})$ *is tractable if* $\mathrm{Pol}(\mathbf{A})$ *contains* AND_2*,* OR_2*,* MAJ_3*,* XOR_3*, or a constant function; otherwise it is NP-hard.*

We now define several polymorphism identites that play an important role in PCSP complexity characterisations. For two functions $f, g : A^m \to B$, we write $f \approx g$ to mean that $f(a_1, \dots, a_m) = g(a_1, \dots, a_m)$ for all $a_1, \dots, a_m \in A$.

Definition 2.33. A function $f : \{0,1\}^m \to \{0,1\}$ is called *folded* if $f(1 - x_1, \dots, 1 - x_m) \approx 1 - f(x_1, \dots, x_m)$.

The families XOR, AT, and THR_q (for all q) are folded, whereas AND and OR are not. Furthermore, a Boolean template $(\mathbf{A}, \mathbf{B})$ is folded (in the sense of Definition 2.24) if and only if every function in $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ is folded.

Definition 2.34. A function $f : A^m \to B$ is called *conservative* if $f(a_1, \dots, a_m) \in \{a_1, \dots, a_m\}$ for all $a_i, \dots, a_m \in A$ and *idempotent* if $f(a, \dots, a) = a$ for all $a \in A$.

Note that conservativity implies idempotence, and that over the Boolean domain, the two notions coincide. All the families in Definition 2.27 are idempotent.

Definition 2.35. A function $f : A^m \to B$ is called *symmetric* if

$$f(x_1, \dots, x_m) \approx f(x_{\pi(1)}, \dots, x_{\pi(m)})$$

for all permutations π.

All the families from Definition 2.27 are symmetric except for AT.

Definition 2.36. A function $f : A^m \to B$ is called 2-block-symmetric if its coordinates can be partitioned into two blocks of sizes L and $L+1$ such that f is invariant under coordinate permutations within each block.

The notions of symmetry and 1-block-symmetry coincide, and AT is an example of a 2-block-symmetric family that is not symmetric.

Definition 2.37. A function $f : A^m \to B$ is said to satisfy the *cancellation property* if

$$f(a_1, \dots, a_{m-2}, x, x) = f(a_1, \dots, a_{m-2}, y, y)$$

for all $a_i, x, y \in A$.

Definition 2.38. A function $f : A^{2L+1} \to B$ is called *alternating* if it is 2-block-symmetric with blocks of size L and $L+1$ and satisfies the cancellation property.

The families AT and XOR are alternating. Alternating and block-symmetric functions play an important role in characterising the power of PCSP algortihms, as we will see in Section 2.5.

The following identity characterises the tractability boundary in the CSP dichotomy theorem.

Definition 2.39. A function $f : A^m \to A$ is a *weak near-unanimity function* (WNU) if it satisfies the identity

$$f(y, x, \dots, x) \approx f(x, y, x, \dots, x) \approx \cdots \approx f(x, \dots, x, y)$$

for all $x, y \in A$.

The main challenge in proving the CSP dichotomy was to show that the WNU condition is strong enough to guarantee tractability; the hardness side was shown in [25].

Theorem 2.40 ([24, 68])**.** *For any finite template* $\mathbf{A}$*,* $\mathrm{CSP}(\mathbf{A})$ *is tractable if* $\mathrm{Pol}(\mathbf{A})$ *contains a WNU function; otherwise,* $CSP(\mathbf{A})$ *is NP-hard.*

There are other identites that characterise this boundary: Equivalent conditions include the existence of Taylor, cyclic, and Siggers polymorphisms (Definition 2.83); see [13] for more details. Unlike the WNU identity, however, the Siggers identity is of fixed arity and so testing whether a CSP has a Siggers polymorphism is decidable, and therefore so is testing for tractability.

2.4 Minions

In this section we investigate the algebraic structure of $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$. The study of the algebraic theory of PCSPs was initiated in [8] and was developed in depth in [10]. We start by defining *clones*, which are the central object of study in the algebraic theory of (non-promise) CSPs, and then we discuss *minions*, which are an analog of clones for PCSPs.

Definition 2.41. A *clone* $\mathcal{C}$ is a set of functions $f : A^{ar(f)} \to A$ with the following properties:

- $\mathcal{C}$ contains all the projection functions $\pi_i^m : A^m \to A$ where $\pi_i^m(x_1, \ldots, x_m) = x_i$ for $i \leq m$, and
- $\mathcal{C}$ is closed under composition: If $f \in \mathcal{C}$ is of arity m and $g_1, \ldots, g_m \in \mathcal{C}$ are of arity n, then

$$h(x_1, \ldots, x_n) = f(g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n))$$

  is also in $\mathcal{C}$.

The set of polymorphisms of a template $(\mathbf{A}, \mathbf{B})$ with $\mathbf{A} = \mathbf{B}$ forms a clone: A projection function simply returns one of the input tuples from the relation, and applying a polymorphism to members of the relation yields a member of the relation, which can then serve as input to a polymorphism, thus fulfilling the composition requirement. In the case of templates $(\mathbf{A}, \mathbf{B})$ with $\mathbf{A} \neq \mathbf{B}$, the set of polymorphisms no longer forms a clone, since, for example, the projection and composition conditions can fail when $\mathbf{A}$ and $\mathbf{B}$ have different domains or when some relations in $\mathbf{A}$ are strict subsets of their corresponding relations in $\mathbf{B}$. However, a weaker notion of composition remains.

For $f\colon A^m \to B$, $g\colon A^n \to B$, and $\pi\colon [m] \to [n]$, we say that g is the *minor* of f obtained from π if

$$g(x_1, \ldots, x_n) \approx f(x_{\pi(1)}, \ldots, x_{\pi(m)}), \tag{2.1}$$

We write $f \xrightarrow{\pi} g$ and $g = f_{/\pi}$ as shorthand for Equation (2.1), which is called a *minor identity*.

Definition 2.42. A *minion* $\mathcal{M}$ consists of sets $\mathcal{M}^{(m)}$ for $m \in \mathbb{N}$ equipped with operations $(\cdot)_{/\pi} : \mathcal{M}^{(m)} \to \mathcal{M}^{(n)}$ such that

- $(f_{/\pi})_{/\tau} = f_{/\tau\circ\pi}$ for $\pi : [m] \to [n]$, $\tau : [n] \to [\ell]$, and
- $f_{/\mathrm{id}} = f$.

We write $\mathcal{M}$ for the disjoint union of $\mathcal{M}^{(m)}$, $m \in \mathbb{N}$, and $\mathrm{ar}(f) = m$ for $f \in \mathcal{M}^{(m)}$. A *minion homomorphism* $\xi : \mathcal{M} \to \mathcal{N}$ is a function that preserves arity and minors: $\mathrm{ar}(\xi(f)) = \mathrm{ar}(f)$ and $\xi(f_{/\pi}) = \xi(f)_{/\pi}$.

For a pair of sets (A, B), a nonempty set of functions from A^n to B (for $n \in \mathbb{N}$) closed under taking minors is a minion. The set of polymorphisms $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ of a PCSP template $(\mathbf{A}, \mathbf{B})$ also forms a minion: The function π defining the minor acts on the arguments of a polymorphism through permutation, deletion, and duplication, all of which are compatible with the definition of a polymorphism. Therefore minor relationship between two functions $g = f_{/\pi}$ can be interpreted as computing g by plugging its inputs into a circuit for f, allowing inputs to be duplicated or ignored. Every clone is a minion since minors can be simulated via composition with projection functions.

Example 2.43. Let $\mathcal{Q}_{\mathrm{conv}}$ be the set of convex linear functions over $\mathbb{Q}$, i.e., functions $f : \mathbb{Q}^n \to \mathbb{Q}$ defined by

$$f(x_1, \ldots, x_n) = \sum_{i\in[n]} \alpha_i x_i$$

for some α_i's, $i \in [n]$, such that $\alpha_i \in [0, 1]$, $\sum_{i\in[n]} \alpha_i = 1$.

Note that $\mathcal{Q}_{\mathrm{conv}}$ contains symmetric functions of all arities: For arity n, take $f(x_1, \ldots, x_n) = \sum_{i\in[n]} x_i/n$.

Example 2.44. Let $\mathcal{Z}_{\mathrm{aff}}$ be the set of affine functions over $\mathbb{Z}$, i.e., functions $f : \mathbb{Z}^n \to \mathbb{Z}$ given by

$$f(x_1, \ldots, x_n) = \sum_{i\in[n]} \beta_i x_i$$

where $\beta_i \in \mathbb{Z}$ satisfy $\sum_{i\in[n]} \beta_i = 1$.

Example 2.45. Define the minion $\mathcal{M}_{\mathrm{BLP+Aff}}$ as follows: For $L \in \mathbb{N}$, its objects of arity L are pairs $(\alpha, \beta) \in \mathbb{Q}^L_{\geq 0} \times \mathbb{Z}^L$ satisfying $\sum_{i\in[L]} \alpha_i = 1$, $\sum_{i\in[L]} \beta_i = 1$, and such that for each $i \in [L]$, $\alpha_i = 0$

implies $\beta_i = 0$. A pair (α, β) can be seen as an L-ary function on $\mathbb{Q}^2$:

$$f((x_1, y_1), \ldots, (x_L, y_L)) = (\Sigma \alpha_i x_i, \Sigma \beta_i y_i).$$

We now describe the connection between minions and PCSPs. A *bipartite minor condition* is a finite set Σ of minor identities where the sets of function symbols used on the left- and right-hand sides are disjoint. More precisely, Σ is a pair of disjoint sets U and V of function symbols of arities m and n, respectively, and a set of minor identities of the form $f \xrightarrow{\pi} g$, where $g \in U$, $f \in V$ and $\pi : [m] \to [n]$. A bipartite minor condition Σ is *satisfied* in a minion $\mathcal{M}$ if there is an assignment $\xi : U \cup V \to \mathcal{M}$ that assigns to each function symbol a function from $\mathcal{M}$ of the corresponding arity so that for every identity $f \xrightarrow{\pi} g$ in Σ, we have $\xi(f) \xrightarrow{\pi} \xi(g)$ in $\mathcal{M}$. A bipartite minor condition is called *trivial* if it is satisfied in every minion, or equivalently, in the minion consisting of all projections on $\{0, 1\}$. Since choosing a projection of arity n is the same as choosing an element of $[n]$, deciding whether a bipartite minor condition is trivial is the same as the standard Label Cover [1]; see Section 3.4 for further discussion.

We can now define the *promise satisfaction of a minor condition* problem. For a minion $\mathcal{M}$ and an integer m, $\mathrm{PMC}_{\mathcal{M}}(m)$ is the following promise problem: Given a bipartite minor condition Σ that involves only symbols of arity at most m, return YES if Σ is trivial and NO if Σ is not satisfiable in $\mathcal{M}$. The promise is that exactly one of these cases holds. Barto et al. [10] show that $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is log-space equivalent to $\mathrm{PMC}_{\mathcal{M}}(m)$, for $\mathcal{M} = \mathrm{Pol}(\mathbf{A}, \mathbf{B})$ and m a constant depending on $\mathbf{A}$ only.

Next we define some algebraic constructions which capture the power of gadget reductions.

Definition 2.46. Let $(\mathbf{A}_1, \mathbf{B}_1)$ and $(\mathbf{A}_2, \mathbf{B}_2)$ be similar PCSP templates. We say that $(\mathbf{A}_2, \mathbf{B}_2)$ is a *homomorphic relaxation* of $(\mathbf{A}_1, \mathbf{B}_1)$ if there are homomorphisms $h_A : \mathbf{A}_2 \to \mathbf{A}_1$ and $h_B : \mathbf{B}_1 \to \mathbf{B}_2$.

Intuitively, a homomorphic relaxation of $(\mathbf{A}_1, \mathbf{B}_1)$ enlarges the gap between $\mathbf{A}_1$ and $\mathbf{B}_1$, which makes it easier to distinguish between the cases $\mathbf{X} \to \mathbf{A}_1$ and $\mathbf{X} \not\to \mathbf{B}_1$. This is formalised in Theorem 2.56, which shows that we indeed have a reduction from $\mathrm{PCSP}(\mathbf{A}_2, \mathbf{B}_2)$ to $\mathrm{PCSP}(\mathbf{A}_1, \mathbf{B}_1)$. Furthermore, the identity reduction is a valid reduction from $\mathrm{PCSP}(\mathbf{A}_2, \mathbf{B}_2)$ to $\mathrm{PCSP}(\mathbf{A}_1, \mathbf{B}_1)$ if and only if $(\mathbf{A}_2, \mathbf{B}_2)$ is a homomorphic relaxation of $(\mathbf{A}_1, \mathbf{B}_1)$.

Homomorphic relaxations also preserve solvability by algorithms, in the following sense.

Proposition 2.47. *Let* $(\mathbf{A}_2, \mathbf{B}_2)$ *be a homomorphic relaxation of* $(\mathbf{A}_1, \mathbf{B}_1)$ *and suppose that algorithm* $\mathcal{ALG}$ *solves* $\mathrm{PCSP}(\mathbf{A}_1, \mathbf{B}_1)$. *Then* $\mathcal{ALG}$ *also solves* $\mathrm{PCSP}(\mathbf{A}_2, \mathbf{B}_2)$.

Proof. Let $\mathbf{X}$ be an instance of $\mathrm{PCSP}(\mathbf{A}_2, \mathbf{B}_2)$. Since the sets of YES and NO instances of $\mathrm{PCSP}(\mathbf{A}_2, \mathbf{B}_2)$ are subsets of the sets of YES and NO instances of $\mathrm{PCSP}(\mathbf{A}_1, \mathbf{B}_1)$, respectively, $\mathbf{X}$ is a valid input to $\mathcal{ALG}$. If $\mathbf{X} \to \mathbf{A}_2$, then $\mathbf{X} \to \mathbf{A}_1$ since $\mathbf{A}_2 \to \mathbf{A}_1$, and therefore $\mathcal{ALG}(\mathbf{X}) = \text{YES}$. If $\mathbf{X} \not\to \mathbf{B}_2$, then $\mathbf{X} \not\to \mathbf{B}_1$ since otherwise we would have $\mathbf{X} \to \mathbf{B}_2$ from $\mathbf{B}_1 \to \mathbf{B}_2$, and so $\mathcal{ALG}(\mathbf{X}) = \text{NO}$.

The result also holds if $\mathcal{ALG}$ is a search algorithm that returns a homomorphism $\mathbf{X} \to \mathbf{B}_1$: Apply the homomorphism $\mathbf{B}_1 \to \mathbf{B}_2$ to obtain a homomorphism $\mathbf{X} \to \mathbf{B}_2$. □

Another component of gadget constructions is the following.

Definition 2.48. For a relational structure $\mathbf{A} = (A; R_1, \ldots, R_p)$, a *primitive positive formula (pp-formula)* over $\mathbf{A}$ is an existentially quantified conjunction of predicates of the form $(x_1, \ldots, x_{\mathrm{ar}(R_i)}) \in R_i$ or $x_j = x_k$.

Definition 2.49. Let $(\mathbf{A}_1, \mathbf{B}_1)$ be a PCSP template. A template $(\mathbf{A}_2, \mathbf{B}_2)$ over the same pair of sets (A, B) as $(\mathbf{A}_1, \mathbf{B}_1)$ is *primitive positive definable (pp-definable)* in $(\mathbf{A}_1, \mathbf{B}_1)$ if, for each pair of arity k relational symbols (R, S) of $(\mathbf{A}_2, \mathbf{B}_2)$, there are pp-formulas $\Psi^{\mathbf{A}_1}$ over $\mathbf{A}_1$ and $\Psi^{\mathbf{B}_1}$ over $\mathbf{B}_1$ such that

$$R = \{(a_1, \ldots, a_k) \in A^k | \Psi^{\mathbf{A}_1}(a_1, \ldots, a_k)\} \text{ and}$$
$$S = \{(b_1, \ldots, b_k) \in B^k | \Psi^{\mathbf{B}_1}(b_1, \ldots, b_k)\}.$$

Example 2.50. The binary equality predicate can be pp-defined from the binary disequality predicate over the domain $A = \{1, 2, 3\}$:

$$x = y \text{ if and only if } \exists a \exists b . x \neq a \wedge x \neq b \wedge y \neq a \wedge y \neq b \wedge a \neq b.$$

Example 2.51. Recall the templates **1-in-3** and **NAE** from Example 2.18. These are pp-definable from the standard 3-SAT problem: With $S_{abc}(x, y, z)$ for $a, b, c \in \{0, 1\}$ as in Example 2.10, we can define

$$\mathbf{NAE}(x, y, z) = S_{111}(x, y, z) \wedge S_{000}(x, y, z)$$
$$\textbf{1-in-3}(x, y, z) = \mathbf{NAE}(x, y, z) \wedge S_{011}(x, y, z) \wedge S_{101}(x, y, z) \wedge S_{110}(x, y, z).$$

Definition 2.52. Let $(\mathbf{A}_1, \mathbf{B}_1)$ and $(\mathbf{A}_2, \mathbf{B}_2)$ be two PCSP templates. We say that $(\mathbf{A}_2, \mathbf{B}_2)$ is an (n-th) *pp-power* of $(\mathbf{A}_1, \mathbf{B}_1)$ if $A_2 = A_1^n, B_2 = B_1^n$, and if we view k-ary relations on $\mathbf{A}_2$ and $\mathbf{B}_2$ as kn-ary relations on A_1 and B_1, respectively, then $(\mathbf{A}_2, \mathbf{B}_2)$ is pp-definable in $(\mathbf{A}_1, \mathbf{B}_1)$.

Example 2.53. The tensor product of graphs $\mathbf{G}_1, \ldots, \mathbf{G}_n$ is the graph $\mathbf{G}$ whose vertex set is the Cartesian product of the vertex sets of $\mathbf{G}_1, \ldots, \mathbf{G}_n$ and where $((u_1, \ldots, u_n), (v_1, \ldots, v_n))$ is an edge of $\mathbf{G}$ if and only if (u_i, v_i) is an edge of $\mathbf{G}_i$ for each i. Let $\mathbf{G}$ and $\mathbf{H}$ be graphs such that $\mathbf{G} \to \mathbf{H}$. Then, letting $\mathbf{G}^n$ denote the n-th tensor power of $\mathbf{G}$, the template $(\mathbf{G}^n, \mathbf{H}^n)$ is an n-th pp-power of $(\mathbf{G}, \mathbf{H})$ since the edge relation of $\mathbf{G}^n$ (and similarly that of $\mathbf{H}^n$) can be pp-defined from the edge relation of $\mathbf{G}$: $((u_1, \ldots, u_n), (v_1, \ldots, v_n)) \in E(\mathbf{G}^n)$ if and only if $(u_i, v_i) \in E(\mathbf{G})$ for all i. Note that $(\mathbf{G}^n, \mathbf{H}^n)$ is still a valid PCSP template since the existence of a homomorphism $\mathbf{G} \to \mathbf{H}$ implies the existence of a homomorphism $\mathbf{G}^n \to \mathbf{H}^n$.

Definition 2.54. We say that $(\mathbf{A}', \mathbf{B}')$ is *pp-constructible* from $(\mathbf{A}, \mathbf{B})$ if there exists a sequence

$$(\mathbf{A}, \mathbf{B}) = (\mathbf{A}_1, \mathbf{B}_1), \ldots, (\mathbf{A}_k, \mathbf{B}_k) = (\mathbf{A}', \mathbf{B}')$$

of templates where each $(\mathbf{A}_{i+1}, \mathbf{B}_{i+1})$ is a pp-power or a homomorphic relaxation of $(\mathbf{A}_i, \mathbf{B}_i)$.

The next two theorems connect the logic-based constructions from Definitions 2.49, 2.52, and 2.54 with algebraic constructions involving minions, thus characterising the power of gadgets in two different ways.

Theorem 2.55 ([10, Theorem 4.12])**.** *Let $(\mathbf{A}_i, \mathbf{B}_i)$ for $i = 1, 2$ be PCSP templates and $\mathcal{M}_i = \mathrm{Pol}(\mathbf{A}_i, \mathbf{B}_i)$. The following are equivalent:*

1. *There exists a minion homomorphism $\mathcal{M}_1 \to \mathcal{M}_2$.*
2. *$\mathcal{M}_2$ satisfies all bipartite minor conditions satisfied in $\mathcal{M}_1$.*
3. *$(\mathbf{A}_2, \mathbf{B}_2)$ is a homomorphic relaxation of a pp-power of $(\mathbf{A}_1, \mathbf{B}_1)$.*
4. *$(\mathbf{A}_2, \mathbf{B}_2)$ is pp-constructible from $(\mathbf{A}_1, \mathbf{B}_1)$.*

Each of the conditions of Theorem 2.55 implies a reduction from $\mathrm{PCSP}(\mathbf{A}_2, \mathbf{B}_2)$ to $\mathrm{PCSP}(\mathbf{A}_1, \mathbf{B}_1)$.

Theorem 2.56 ([10, Theorem 3.1])**.** *Let $(\mathbf{A}_1, \mathbf{B}_1)$ and $(\mathbf{A}_2, \mathbf{B}_2)$ be two PCSP templates, and let $\mathcal{M}_i = \mathrm{Pol}(\mathbf{A}_i, \mathbf{B}_i)$ for $i = 1, 2$. If there is a minion homomorphism $\xi : \mathcal{M}_1 \to \mathcal{M}_2$ then $\mathrm{PCSP}(\mathbf{A}_2, \mathbf{B}_2)$ is log-space reducible to $\mathrm{PCSP}(\mathbf{A}_1, \mathbf{B}_1)$.*

By taking the identity homomorphism in Theorem 2.56, we get that $\mathrm{Pol}(\mathbf{A}_1, \mathbf{B}_1) \subseteq \mathrm{Pol}(\mathbf{A}_2, \mathbf{B}_2)$ implies a log-space reduction from $\mathrm{PCSP}(\mathbf{A}_2, \mathbf{B}_2)$ to $\mathrm{PCSP}(\mathbf{A}_1, \mathbf{B}_1)$.

Using the reduction induced by homomorphic relaxation, we see that the only promise templates $(\mathbf{A}, \mathbf{B})$ with possibly unresolved complexity are those where $\mathrm{CSP}(\mathbf{A})$ and $\mathrm{CSP}(\mathbf{B})$ are both NP-hard.

Proposition 2.57. *Let* $(\mathbf{A}, \mathbf{B})$ *be a promise template such that at least one of* $\mathrm{CSP}(\mathbf{A})$, $\mathrm{CSP}(\mathbf{B})$ *is tractable. Then* $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ *is tractable.*

Proof. We have $\mathrm{PCSP}(\mathbf{A}, \mathbf{B}) \leq_p \mathrm{PCSP}(\mathbf{A}, \mathbf{A}) = \mathrm{CSP}(\mathbf{A})$, since $(\mathbf{A}, \mathbf{B})$ is a homomorphic relaxation of $(\mathbf{A}, \mathbf{A})$ as $\mathbf{A} \to \mathbf{B}$ by assumption. Similarly, $\mathrm{PCSP}(\mathbf{A}, \mathbf{B}) \leq_p \mathrm{PCSP}(\mathbf{B}, \mathbf{B}) = \mathrm{CSP}(\mathbf{B})$, since $(\mathbf{A}, \mathbf{B})$ is a homomorphic relaxation of $(\mathbf{B}, \mathbf{B})$ as $\mathbf{A} \to \mathbf{B}$ by assumption. □

Barto and Kozik [12] recently introduced a more general notion of minion homomorphism, called a *minion* (d, r)*-homomorphism*, which allows for additional reductions between PCSPs. See [12] for further discussion.

2.5 Tractability

Following the intuition that symmetries of computational problems lead to tractability, we present several algorithms for PCSPs and discuss polymorphism families that guarantee their applicability. Many of these results are in fact dichotomies, which prove that $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is tractable if $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ contains certain families of functions, and is NP-hard otherwise. We focus on sufficient conditions for tractability but will also mention such dichotomies when they are known to exist.

In the rest of this section, let $(\mathbf{A}, \mathbf{B})$ be a PCSP template, where $\mathbf{A} = (A; R_1, \ldots, R_p)$ and $\mathbf{B} = (B; S_1, \ldots, S_p)$. Let $\mathbf{X} = (X; T_1, \ldots, T_p)$ be an instance of $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$. We assume without loss of generality that all three structures contain a unary relation equal to X in $\mathbf{X}$, A in $\mathbf{A}$, and B in $\mathbf{B}$; the relation is called R_u in $\mathbf{A}$. If this is not the case, the template and the instance can be extended without changing the set of solutions.

The *basic linear programming relaxation* (BLP) of $\mathbf{X}$, denoted by $\mathrm{BLP}(\mathbf{X}, \mathbf{A})$, is as follows. The variables are $\lambda_{\mathbf{x},i}(\mathbf{a})$ for every $i \in [p]$, $\mathbf{x} \in T_i$, and $\mathbf{a} \in R_i$, and the constraints are given in Figure 2.1. Note that $\mathrm{BLP}(\mathbf{X}, \mathbf{A})$ does not depend on $\mathbf{B}$. Intuitively, the BLP relaxes a problem by allowing the variables $\lambda_{\mathbf{x},i}(\mathbf{a})$ to take values in $[0, 1]$ rather than just $\{0, 1\}$. In the $\{0, 1\}$ case, the first and second constraints of Figure 2.1 ensure that each tuple of variables is assigned a single legal relational tuple,

and the third constraint ensures that these assignments are consistent with the assignments of domain values to individual variables.

$$\begin{aligned}
0 \;\leq\; \lambda_{\mathbf{x},i}(\mathbf{a}) \;&\leq\; 1 && \forall i \in [p], \mathbf{x} \in T_i, \mathbf{a} \in R_i \\
\sum_{\mathbf{a} \in R_i} \lambda_{\mathbf{x},i}(\mathbf{a}) \;&=\; 1 && \forall i \in [p], \mathbf{x} \in T_i \\
\sum_{\substack{\mathbf{a} \in R_i \\ \mathbf{a}_j = a}} \lambda_{\mathbf{x},i}(\mathbf{a}) \;&=\; \lambda_{x_j,R_u}(a) && \forall i \in [p], \mathbf{x} \in T_i, a \in A, j \in [\mathrm{ar}(R_i)]
\end{aligned}$$

Figure 2.1: Definition of $\mathrm{BLP}(\mathbf{X}, \mathbf{A})$.

By construction, if $\mathbf{X} \to \mathbf{A}$, then $\mathrm{BLP}(\mathbf{X}, \mathbf{A})$ is feasible. We say that BLP *solves* $\mathrm{PCSP}_\mathrm{d}(\mathbf{A}, \mathbf{B})$ if for every instance $\mathbf{X}$ such that $\mathrm{BLP}(\mathbf{X}, \mathbf{A})$ is feasible, we have $\mathbf{X} \to \mathbf{B}$.

Theorem 2.58 ([10, Theorem 7.9])**.** *Let* $(\mathbf{A}, \mathbf{B})$ *be a PCSP template. The following are equivalent:*

1. *BLP solves* $\mathrm{PCSP}_\mathrm{d}(\mathbf{A}, \mathbf{B})$.
2. $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ *contains symmetric functions of all arities.*
3. $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ *admits a minion homomorphism from* $\mathcal{Q}_{conv}$.

The proof of Theorem 2.58 assumes only the *existence* of symmetric polymorphisms, which is sufficient to show that $\mathrm{PCSP}_\mathrm{d}(\mathbf{A}, \mathbf{B})$ is solved by BLP, but solving $\mathrm{PCSP}_\mathrm{s}(\mathbf{A}, \mathbf{B})$ would require knowledge of the concrete polymorphisms and so BLP in this form does not solve the search problem in general.

The *basic affine integer programming relaxation* (AIP) of $\mathbf{X}$, denoted by $\mathrm{AIP}(\mathbf{X}, \mathbf{A})$, is as follows. The variables are $\tau_{\mathbf{x},i}(\mathbf{a})$ for every $i \in [p]$, $\mathbf{x} \in T_i$, and $\mathbf{a} \in R_i$, and the constraints are given in Figure 2.2. Intuitively, the AIP relaxes a problem by allowing the variables $\tau_{\mathbf{x},i}(\mathbf{a})$ to take values in $\mathbb{Z}$ rather than just $\{0, 1\}$.

$$\begin{aligned}
\tau_{\mathbf{x},i}(\mathbf{a}) \;&\in\; \mathbb{Z} && \forall i \in [p], \mathbf{x} \in T_i, \mathbf{a} \in R_i \\
\sum_{\mathbf{a} \in R_i} \tau_{\mathbf{x},i}(\mathbf{a}) \;&=\; 1 && \forall i \in [p], \mathbf{x} \in T_i \\
\sum_{\substack{\mathbf{a} \in R_i \\ \mathbf{a}_j = a}} \tau_{\mathbf{x},i}(\mathbf{a}) \;&=\; \tau_{x_j,R_u}(a) && \forall i \in [p], \mathbf{x} \in T_i, a \in A, j \in [\mathrm{ar}(R_i)]
\end{aligned}$$

Figure 2.2: Definition of $\mathrm{AIP}(\mathbf{X}, \mathbf{A})$.

By construction, if $\mathbf{X} \to \mathbf{A}$, then $\mathrm{AIP}(\mathbf{X}, \mathbf{A})$ is feasible. We say that AIP *solves* $\mathrm{PCSP}_{\mathrm{d}}(\mathbf{A}, \mathbf{B})$ if for every instance $\mathbf{X}$ such that $\mathrm{AIP}(\mathbf{X}, \mathbf{A})$ is feasible, we have $\mathbf{X} \to \mathbf{B}$.

Theorem 2.59 ([10, Theorem 7.19])**.** *Let* $(\mathbf{A}, \mathbf{B})$ *be a PCSP template. The following are equivalent:*

1. *AIP solves* $\mathrm{PCSP}_{\mathrm{d}}(\mathbf{A}, \mathbf{B})$.
2. $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ *contains alternating functions of all odd arities.*
3. $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ *admits a minion homomorphism from* $\mathcal{Z}_{aff}$.

As was the case with BLP, AIP in this form does not solve the search problem $\mathrm{PCSP}_{\mathrm{s}}(\mathbf{A}, \mathbf{B})$.

In Algorithm 1 we present the *combined basic LP and affine IP algorithm* (BLP + AIP) [20], which makes use of both $\mathrm{BLP}(\mathbf{X}, \mathbf{A})$ and $\mathrm{AIP}(\mathbf{X}, \mathbf{A})$. Intuitively, the BLP and AIP provide information about solutions that are not possible, and then the BLP+AIP algorithm combines these two sources of information.

Algorithm 1 The BLP+AIP algorithm

Input: an instance $\mathbf{X}$ of $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$
Output: YES if $\mathbf{X} \to \mathbf{A}$ and NO if $\mathbf{X} \not\to \mathbf{B}$

find a relative interior point $(\lambda_{\mathbf{x},i}(\mathbf{a}))_{i\in[p],\mathbf{x}\in T_i,\mathbf{a}\in R_i}$ of $\mathrm{BLP}(\mathbf{X}, \mathbf{A})$
if no relative interior point exists **then Reject**
refine $\mathrm{AIP}(\mathbf{X}, \mathbf{A})$ by setting $\tau_{\mathbf{x},i}(\mathbf{a}) = 0$ if $\lambda_{\mathbf{x},i}(\mathbf{a}) = 0$
if the refined $\mathrm{AIP}(\mathbf{X}, \mathbf{A})$ is feasible **then Accept**
else Reject

As shown in [20], if $\mathbf{X} \to \mathbf{A}$, then BLP + AIP accepts. We say that BLP + AIP *solves* $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ if for every instance $\mathbf{X}$ accepted by BLP + AIP we have $\mathbf{X} \to \mathbf{B}$. We now characterise the power of BLP+AIP.

Theorem 2.60 ([20, Theorem 5.1])**.** *Let* $(\mathbf{A}, \mathbf{B})$ *be a PCSP template. The following are equivalent:*

1. *BLP+AIP solves* $\mathrm{PCSP}_{\mathrm{d}}(\mathbf{A}, \mathbf{B})$.
2. *For each* $L \in \mathbb{N}$, $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ *has a 2-block-symmetric function of arity* $2L + 1$ *with blocks of size* L *and* $L + 1$.
3. $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ *admits a minion homomorphism from* $\mathcal{M}_{BLP+Aff}$.

We also have the following sufficient condition for solvability by BLP+AIP.

Theorem 2.61 ([20, Theorem 3.2]). *If* $(\mathbf{A}, \mathbf{B})$ *is a PCSP template such that* $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ *contains symmetric functions of arbitrarily large arity, then BLP+AIP solves* $\mathrm{PCSP_d}(\mathbf{A}, \mathbf{B})$.

Another powerful feature of BLP+AIP is that it solves all cases of Schaefer's Theorem in a uniform way. Therefore, as mentioned in [20], it is a tantalising possibility that BLP with a fixed number of rounds of Sherali Adams + AIP could provide a uniform algorithm for all tractable CSPs.

The existence of block-symmetric polymorphisms is used only in the proofs of Theorems 2.60 and 2.61 and not by the BLP+AIP algorithm itself, and therefore it is unknown whether there exists an "oblivious" polynomial-time algorithm that solves the search problem $\mathrm{PCSP_s}(\mathbf{A}, \mathbf{B})$ in general, without any knowledge of the concrete polymorphisms.

We now present in Algorithm 2 a modified BLP algorithm that solves the tractable cases in both the symmetric folded Boolean dichotomy of [19] and the symmetric Boolean dichotomy of [41].

Algorithm 2 Singleton BLP algorithm for Boolean $\mathrm{PCSP_d}(\mathbf{A}, \mathbf{B})$ from [19]

Input: an instance $\mathbf{X}$ of $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$
Output: YES if $\mathbf{X} \to \mathbf{A}$ and NO if $\mathbf{X} \not\to \mathbf{B}$

1: **for** each variable x of $\mathbf{X}$ **do**
2: fix $\lambda_{x,u}(0) = 1$ and re-solve the LP from Figure 2.1
3: **if** the LP is infeasible **then** set $\lambda_{x,u}(1) = 1$ and re-solve the LP from Figure 2.1
4: **if** the LP is still infeasible **then Reject**
5: **Accept**

The symmetric Boolean dichotomy from [41] is as follows.

Theorem 2.62 ([41]). *Let* $(\mathbf{A}, \mathbf{B})$ *be a Boolean PCSP template. If* $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ *contains a constant or at least one of* AND, OR, XOR, AT, THR_q *(for some* q*), or their negations, then* $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ *is polynomial-time tractable. Otherwise, if* $(\mathbf{A}, \mathbf{B})$ *is also symmetric, then* $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ *is NP-hard.*

Algorithm 2 solves all Boolean PCSPs with AT or THR_q polymorphisms, and in the other cases, $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ reduces to a CSP which falls under the scope of Theorem 2.32. The AT and THR_q polymorphisms are used to round the LP solutions, and Algorithm 2 can be modified to solve the search problem $\mathrm{PCSP_s}(\mathbf{A}, \mathbf{B})$ as well, as shown in [19]. The symmetry condition on the templates is needed only for hardness and not tractability, so Algorithm 2 solves any Boolean PCSP with the stated polymorphism families.

By fixing assignments to entire constraints rather than only to single variables (as in Algorithm 2), one obtains the *constraint BLP*. Ciardo and Živný [28] recently introduced the *CLAP algorithm*, which

runs the constraint BLP followed by AIP. They showed that CLAP is more powerful than BLP+AIP and that it solves all PCSPs with *H-symmetric* polymorphisms of arbitrarily large arity; see [28] for definitions. The power of CLAP was characterised in terms of minion homomorphisms, but no polymorphism characterisation along the lines of Theorem 2.60 is known, and it is suspected that H-symmetry is not the correct notion for such a characterisation.

In [10] it was observed that all currently known tractable PCSPs are homomorphic relaxations of tractable CSPs, possibly over an infinite domain, so that $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ reduces to a tractable $\mathrm{CSP}(\mathbf{C})$ with $\mathbf{A} \to \mathbf{C} \to \mathbf{B}$. Moreover, infinite domains can be necessary, as was shown for $\mathrm{PCSP}(\textbf{1-in-3}, \textbf{NAE})$.

Theorem 2.63 ([10, Theorem 8.1])**.** *Let $\mathbf{A}$ be a finite relational structure such that $(\textbf{1-in-3}, \textbf{NAE})$ is pp-constructible from $\mathbf{A}$. Then $\mathrm{CSP}(\mathbf{A})$ is NP-hard.*

2.6 Hardness

If symmetry leads to tractability, then the absence of symmetry should lead to hardness. In this section we present various properties of $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ that indicate a lack of symmetry and thus lead to sufficient conditions for hardness. The sufficiency of these conditions follows from the hardness of a promise problem called *Gap Label Cover*, whose hardness is a direct consequence of the PCP theorem [2, 3, 34]. We begin with some simple conditions, and then explore more general ones related to the algebraic theory of PCSPs.

Definition 2.64. Let $f : A^m \to B$. A coordinate $i \in [m]$ is called *essential* if there exists $(a_1, \dots, a_m) \in A^m$ and $b \in A$ such that

$$f(a_1, \dots, a_{i-1}, a_i, a_{i+1}, \dots, a_m) \neq f(a_1, \dots, a_{i-1}, b, a_{i+1}, \dots, a_m).$$

A set of functions is said to have *bounded essential arity* if there exists a constant c such that every function in the set has at most c essential coordinates.

Theorem 2.65 ([10, Proposition 5.14])**.** *Let $(\mathbf{A}, \mathbf{B})$ be a PCSP template. Assume that there exists a minion homomorphism $\xi : \mathrm{Pol}(\mathbf{A}, \mathbf{B}) \to \mathcal{N}$ for some minion $\mathcal{N}$, possibly on infinite sets, which has bounded essential arity and does not contain a constant function. Then $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is NP-hard.*

Theorem 2.65 is stronger than the statement that PCSP($\mathbf{A}, \mathbf{B}$) is hard if Pol($\mathbf{A}, \mathbf{B}$) has bounded essential arity: The homomorphism requirement means that Pol($\mathbf{A}, \mathbf{B}$) need not itself have bounded essential arity, though this is certainly sufficient via the identity homomorphism. Bounded essential arity was used in [8] to show hardness of promise SAT (Example 2.22), and then in a much more involved way in [57] to show hardness of promise graph homomorphism problems (Example 2.15).

Theorem 2.65 also gives a necessary condition for tractability: If Pol($\mathbf{A}, \mathbf{B}$) contains only finitely many operations (and no constants), then PCSP($\mathbf{A}, \mathbf{B}$) is NP-hard. This contrasts with the situation for CSPs and clones, where low-arity operations (for instance, AND_2) can be composed to get operations of arbitrarily large essential arity; merely taking minors does not increase essential arity.

Other than for constant functions, no single bipartite minor condition implies tractability, since such a condition involves only functions of bounded arity and would thus be satisfied in a minion with bounded essential arity, making PCSP($\mathbf{A}, \mathbf{B}$) NP-hard as above. Therefore, an infinite set of bipartite minor conditions is necessary for tractability.

We now define another property of polymorphisms which is sufficient for hardness and is weaker than bounded essential arity. For a function $f(x_1, \dots, x_m) : \{0,1\}^m \to B$ and a set $U \subseteq [m]$, we denote by $f(U)$ the value obtained by setting $x_i = 1$ if $i \in U$ and $x_i = 0$ otherwise.

Definition 2.66. A function $f : \{0,1\}^m \to B$ is called *c-fixing* if there exists a set $U \subseteq [m]$, $|U| \leq c$, such that

- $f(x_1, \dots, x_m) = 0$ whenever $x_i = 0$ for all $i \in U$, and
- $f(x_1, \dots, x_m) = 1$ whenever $x_i = 1$ for all $i \in U$.

A set of functions $\{0,1\}^m \to B$ is called *c-fixing* or said to have *small fixing sets* if there exists a constant c such that all functions in the set are c-fixing.

Theorem 2.67 ([10, Proposition 5.16])**.** *Let* $(\mathbf{A}, \mathbf{B})$ *be a Boolean PCSP template. Assume that there exists a minion homomorphism* $\xi : \mathrm{Pol}(\mathbf{A}, \mathbf{B}) \to \mathcal{N}$ *for some minion* $\mathcal{N}$ *of Boolean functions which is c-fixing for some c. Then* PCSP($\mathbf{A}, \mathbf{B}$) *is NP-hard.*

In [19], c-fixing sets are used to prove the hardness part of their symmetric folded Boolean dichotomy. A slightly weaker notion of fixing sets is used in [41] to prove the symmetric Boolean dichotomy (Theorem 2.62), requiring only one of the conditions in Definition 2.66 to hold, rather than both simultaneously. However, this latter result also depends on another property of polymorphisms, which we now define.

Definition 2.68. For a function $f : \{0,1\}^m \to B$, we say that $U \subseteq [m]$ is an i-set if $f(U) = i$.

The hardness in the symmetric Boolean dichotomy relies on the relevant polymorphism minions having not only small fixing sets, but also *bounded antichains*: For some constant c, no function has c pairwise disjoint 1-sets, and no function has c pairwise disjoint 0-sets. Together, these two combinatorial properties suffice for hardness, which is achieved through the following two theorems.

Theorem 2.69 ([41, Theorem 2])**.** *Let* $(\mathbf{A}, \mathbf{B})$ *be a symmetric Boolean PCSP template such that* $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ *is idempotent. If* $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ *does not include any of* AND*,* OR*,* AT*,* XOR*, and* THR_q *(for any* q*), then* $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ *has small fixing sets and bounded antichains.*

Theorem 2.70 ([41, Proposition 5.2])**.** *Let* $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ *be an idempotent minion with small fixing sets and bounded antichains. Then* $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ *is NP-hard.*

The notion of i-sets has been applied to non-Boolean PCSPs as well, including in [9] to PCSPs of the form $(\mathbf{1\text{-}in\text{-}3}, \mathbf{B})$ (Example 2.20).

Having exhibited some examples of concrete hardness conditions, we now describe the Label Cover machinery behind them.

Definition 2.71. Fix a positive integer N. We define the *Label Cover problem* $\mathrm{LC}(N)$ as the following decision problem: The input is a tuple (U, V, E, ℓ, r, Π) where

- $G = (U, V; E)$ is a bipartite graph,
- $\ell, r \leq N$ are positive integers, and
- Π is a family of maps $\pi_e : [r] \to [\ell]$, one for each $e \in E$.

The goal is to decide whether there is a labelling of vertices from U and V with labels from $[r]$ and $[\ell]$, respectively, such that if $(u, v) \in E$, then the label of v is mapped by $\pi_{(u,v)}$ to the label of u. $\mathrm{LC}(N)$ is a CSP where vertices are variables and edges correspond to constraints: A constraint corresponding to an edge $e \in E$ is given by π_e.

Definition 2.72. The *Gap Label Cover problem* with parameters δ (completeness), ϵ (soundness), and N, denoted by $\mathrm{GLC}_{\delta,\epsilon}(N)$, is a promise problem, which, given an instance of $\mathrm{LC}(N)$,

- accepts if there is an assignment that satisfies at least a δ-fraction of the constraints
- rejects if no assignment satisfies more than an ϵ-fraction of the constraints.

The hardness of Gap Label Cover with perfect completeness $\delta = 1$ and soundness $\epsilon < 1$ is a direct result of the PCP theorem. The soundness parameter can be made arbitrarily small by the parallel repetition theorem of Raz [64].

Theorem 2.73 ([2, 3, 34, 64])**.** *There exist constants $K_1, K_2 > 0$ such that for every $\epsilon > 0$ and every $N \geq K_1 \epsilon^{-K_2}$, $GLC_{1,\epsilon}(N)$ is NP-hard.*

The following definition and theorem connect Gap Label Cover with the algebraic theory of PCSPs.

Definition 2.74. Let $\epsilon > 0$. We say that a bipartite minor condition Σ is *ϵ-robust* if no ϵ-fraction of identities from Σ is trivial.

Theorem 2.75 ([10, Corollary 5.10])**.** *If there exists an $\epsilon > 0$ such that $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ does not satisfy any ϵ-robust bipartite minor condition, then $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is NP-hard.*

Theorems 2.65, 2.67, and 2.70 all follow from Theorem 2.75; for completeness, we give a proof of Theorem 2.65 from Theorem 2.75.

Theorem (Theorem 2.65 restated)**.** *Let $\mathcal{M}$ be a minion where every function has essential arity at most c. Then $\mathcal{M}$ does not satisfy any $\frac{1}{c}$-robust bipartite minor condition.*

Proof. Let Σ be a $\frac{1}{c}$-robust bipartite minor condition. Suppose Σ is satisfied by $\mathcal{M}$, that is, there is an assignment ξ from symbols in Σ to functions in $\mathcal{M}$ of the same arity such that for every condition $f \xrightarrow{\pi} g$ in Σ we have $\xi(f) \xrightarrow{\pi} \xi(g)$. Let $I(\xi(f))$ be the set of essential coordinates in $\xi(f)$. It is easy to check that essential coordinates of a minor $\xi(g)$ of a function $\xi(f)$ correspond to essential coordinates of $\xi(f)$, that is: $\xi(f) \xrightarrow{\pi} \xi(g)$ implies $I(\xi(g)) \subseteq \pi(I(\xi(f)))$. Hence if we fix $\iota(g) \in I(\xi(g))$ arbitrarily, for each symbol g on one side of Σ, and choose $\iota(f) \in I(\xi(f))$ uniformly at random, for each symbol f on the other side of Σ, then for each condition $f \xrightarrow{\pi} g$ the corresponding condition $\pi(\iota(f)) = \iota(g)$ is satisfied with probability at least $\frac{1}{c}$. Equivalently, replacing $\xi(g)$ with the projection to $\iota(g)$ and $\xi(f)$ with the projection $\iota(f)$, the condition $p_{\iota(f)} \xrightarrow{\pi} p_{\iota(g)}$ is satisfied with probability at least $\frac{1}{c}$. Therefore, there exists an assignment with projections that satisfies at least $\frac{1}{c}$ of the conditions, which means Σ is not $\frac{1}{c}$-robust. □

The following is a strengthening of Theorem 2.75 that allows a minion to be decomposed into finitely many subsets (that do not need to be minions), so that the ϵ-robustness condition can be tested separately on each of the subsets.

Theorem 2.76 ([10, Theorem 5.21])**.** *Let* $(\mathbf{A}, \mathbf{B})$ *be a PCSP template and let* $\epsilon : \mathbb{N} \to \mathbb{R}$ *be a function such that* $\epsilon(N) \in \Omega(N^{-K})$ *for each* $K > 0$*. Assume that* $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ *is a union of finitely many sets* $\mathcal{M}_1, \dots, \mathcal{M}_k$*, none of which satisfies (for any* N*) any* $\epsilon(N)$*-robust bipartite minor condition involving symbols of arity at most* N*. Then* $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ *is NP-hard.*

Theorem 2.76 is especially powerful when combined with the following lemma as it yields Corollary 2.78, which gives a very general sufficient condition for hardness.

Lemma 2.77 ([10, Lemma 5.11])**.** *Let* A *and* B *be sets, let* $\mathcal{M}$ *be a set of functions* $A \to B$*, and let* $c : \mathbb{N} \to \mathbb{N}$*. Assume that there exists a mapping* I *assigning to each element of* $\mathcal{M}^{(m)}$ *a subset of* $[m]$ *of size at most* $c(m)$ *such that for each* $\pi : [m] \to [n]$ *and each* $f \in \mathcal{M}^{(m)}$ *with* $f_{/\pi} \in \mathcal{M}$*, we have*

$$\pi(I(f(x_1, \dots, x_m)) \cap I(f(x_{\pi(1)}, \dots, x_{\pi(m)})) \neq \emptyset.$$

Then $\mathcal{M}$ *satisfies no* $(1/c(N)^2)$*-robust bipartite minor condition involving symbols of arity at most* N*.*

Corollary 2.78. *Let* $(\mathbf{A}, \mathbf{B})$ *be a PCSP template and let* $\mathcal{M} = \mathrm{Pol}(\mathbf{A}, \mathbf{B})$*. Assume that* $\mathcal{M}$ *is a union of finitely many sets* $\mathcal{M}_1, \dots, M_k$ *and that there exists* c *such that for each* $1 \leq i \leq k$ *there is a function* $I : \mathcal{M}_i \to P(\mathbb{N})$ *satisfying the following three conditions for each* $f \in \mathcal{M}_i^{(m)}$*:*

1. $I(f) \subseteq [m]$
2. $|I(f)| \leq c$
3. *for each* $\pi : [m] \to [n]$ *such that* $f_{/\pi} \in \mathcal{M}_i$*, we have* $\pi(I(f)) \cap I(f_{/\pi}) \neq \emptyset$*.*

Then $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ *is NP-hard.*

Corollary 2.78 was used in [10] to give a concise hardness proof for the promise hypergraph colouring problem $\mathrm{PCSP}(\mathbf{NAE}_2, \mathbf{NAE}_d)$ from Example 2.17, a result first established in [37].

One of the strongest known NP-hardness critera for PCSPs was devised in [22] to prove hardness for SetSAT. It follows from a new layered version of the Gap Label Cover problem that is compatible with many combinatorial properties of polymorphisms such as i-sets, c-fixing sets, and sets of essential coordinates. A full description of this hardness source appears in Chapter 3; here we state only its main consequence.

Theorem 2.79. *Let $\mathcal{M} = \mathrm{Pol}(\mathbf{A}, \mathbf{B})$ and suppose we have a notion of a "special set of coordinates" for functions in $\mathcal{M}$. Suppose there are constants $k, \ell \in \mathbb{N}$ such that the following holds, for every $f \in \mathcal{M}$.*

1. *f has a special set of coordinates of size at most k*
2. *f has no family of more than ℓ (pairwise) disjoint special sets of coordinates,*
3. *if g is a minor of f via π and S is a special set of coordinates of g, then $\pi^{-1}(S)$ is a special set of coordinates of f.*

Then $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ *is NP-hard.*

This hardness result has been abstracted to an algebraic hardness condition in [12].

We conclude this section by describing how to lift hardness results for concrete PCSPs to more general hardness conditions. By Theorem 2.56, a minion homomorphism $\mathrm{Pol}(\mathbf{A}_1, \mathbf{B}_1) \to \mathrm{Pol}(\mathbf{A}_2, \mathbf{B}_2)$ gives a reduction from $\mathrm{PCSP}(\mathbf{A}_2, \mathbf{B}_2)$ to $\mathrm{PCSP}(\mathbf{A}_1, \mathbf{B}_1)$, so that hardness of $\mathrm{PCSP}(\mathbf{A}_2, \mathbf{B}_2)$ implies that of $\mathrm{PCSP}(\mathbf{A}_1, \mathbf{B}_1)$. If we had simple conditions characterising the existence of minion homomorphisms to $\mathrm{Pol}(\mathbf{A}_2, \mathbf{B}_2)$, with $(\mathbf{A}_2, \mathbf{B}_2)$ NP-hard, then these conditions could serve as a hardness test. Several such conditions are obtained in [10]; we first present one for $\mathrm{PCSP}(\mathbf{NAE}_2, \mathbf{NAE}_d)$.

Definition 2.80. An Olšák function is a 6-ary function o that satisfies

$$o(x, x, y, y, y, x) \approx$$
$$o(x, y, x, y, x, y) \approx$$
$$o(y, x, x, x, y, y).$$

The six columns in this condition correspond to 2-colourings of a 3-uniform hypergraph.

Theorem 2.81 ([10, Theorem 6.2])**.** *Let $\mathcal{M}$ be a minion. The following are equivalent:*

1. *There exists $d \geq 2$ and a minion homomorphism $\mathcal{M} \to \mathrm{Pol}(\mathbf{NAE}_2, \mathbf{NAE}_d)$.*
2. *$\mathcal{M}$ does not contain an Olšák function.*

Theorem 2.56 then implies the following.

Theorem 2.82 ([10, Corollary 6.3])**.** *For every PCSP template* $(\mathbf{A}, \mathbf{B})$ *that does not have an Olšák polymorphism,* $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ *is NP-hard.*

This result was used in [10] to achieve new hardness results for approximate graph colouring. In particular, it was shown that 5-colouring a 3-colourable graph is NP-hard.

Next, we give a characterisation of reducibility from the approximate graph colouring problem.

Definition 2.83. A Siggers function [66] is a 6-ary function s satisfying

$$s(x, y, x, z, y, z) \approx s(y, x, z, x, z, y).$$

Theorem 2.84 ([10, Theorem 6.9])**.** *The following are equivalent:*

1. *For every PCSP template* $(\mathbf{A}, \mathbf{B})$ *that does not have a Siggers polymorphism,* $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ *is NP-hard.*
2. $\mathrm{PCSP}(\mathbf{K}_k, \mathbf{K}_c)$ *is NP-hard for all* $c \geq k \geq 3$.
3. $\mathrm{PCSP}(\mathbf{K}_3, \mathbf{K}_c)$ *is NP-hard for all* $c \geq 3$.

Finally, we present a generalisation of the Siggers condition, which characterises reducibility from the promise graph homomorphism problem.

Definition 2.85. Fix a non-bipartite loopless graph $\mathbf{G}$ with vertices $v_1, \ldots, v_n$ and edges $(a_1, b_1), \ldots, (a_m, b_m)$, where each edge (u, v) appears as both (u, v) and (v, u). The $\mathbf{G}$*-loop condition* is the bipartite minor condition

$$f(x_{v_1}, \ldots, x_{v_n}) \approx e(x_{a_1}, \ldots, x_{a_m})$$
$$f(x_{v_1}, \ldots, x_{v_n}) \approx e(x_{b_1}, \ldots, x_{b_m}).$$

Note that the Siggers condition corresponds to $\mathbf{G} = \mathbf{K}_3$.

Theorem 2.86 ([10, Theorem 6.12])**.** *The following are equivalent:*

1. *For every PCSP template* $(\mathbf{A}, \mathbf{B})$ *that does not satisfy the* $\mathbf{G}$*-loop condition for some non-bipartite loopless graph* $\mathbf{G}$*,* $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ *is NP-hard.*

2. $\mathrm{PCSP}(\mathbf{C}_{2k+1}, \mathbf{K}_c)$ *is NP-hard for all* $c \geq 3$, $k \geq 1$, *where* $\mathbf{C}_n$ *is the* n*-cycle.*

3. $\mathrm{PCSP}(\mathbf{G}, \mathbf{H})$ *is NP-hard for any two non-bipartite loopless graphs* $\mathbf{G}$ *and* $\mathbf{H}$ *with* $\mathbf{G} \to \mathbf{H}$.

Chapter 3

SetSAT

We investigate and completely resolve the complexity of (a, g, k)-SetSAT from Example 2.23. This work appears in [21] and [22].

3.1 Preliminaries and results

The domain of the variables in SetSAT is $[d]$ and for a fixed $0 < s < d$ we identify each literal with the indicator function of some $S \subseteq [d]$, $|S| = s$, so that $S(x) = \mathbb{1}[x \in S]$. Every SetSAT instance is unsatisfiable when $s = 0$ and satisfiable when $s = d$, so we exclude these cases in our analysis. For a SetSAT instance (or *formula*) Ψ with n variables $x_1, \ldots, x_n$, an assignment to the variables is a function $\sigma : \{x_1, \ldots, x_n\} \to [d]$. An assignment σ is called g-satisfying if σ satisfies at least g literals in every clause of Ψ. A 1-satisfying assignment is usually simply called a satisfying assignment. A formula is called g-satisfiable if there exists a g-satisfying assignment to its variables, and satisfiable if there exists a 1-satisfying assignment.

Definition 3.1. Let $0 < s < d$ and $1 \leq a \leq g \leq k$. The (a, g, k)-SetSAT problem is the following promise problem. In the decision version, given a SetSAT instance where each clause has k (not necessarily distinct) literals, accept the instance if it is g-satisfiable and reject it if it is not a-satisfiable. In the search version, given a g-satisfiable SetSAT instance, find an a-satisfying assignment.

Theorem 3.2. *Let $1 \leq s < d$ and $1 \leq g \leq k$. The problem (a, g, k)-SetSAT with set size s and domain size d is solvable in polynomial time if $\frac{g-a+1}{k-a+1} \geq \frac{s}{s+1}$ and is NP-hard otherwise.*

Theorem 3.2 easily follows, as described in Section 3.6, from the next result, where we show that the complexity of $(1, g, k)$-SetSAT depends only on the ratio $\frac{g}{k}$, thus generalising the case of $(1, g, k)$-SAT studied in [8] with $s = 1$.

Theorem 3.3. *$(1, g, k)$-SetSAT with set size s and domain size $s + 1$ is solvable in polynomial time if $\frac{g}{k} \geq \frac{s}{s+1}$ and is NP-hard otherwise.*

We prove hardness only for the decision version of (a, g, k)-SetSAT and tractability only for the search version; by the discussion after Definition 2.13, this covers all cases of both the search and decision problems.

The structure of the rest of this chapter is as follows. In Section 3.2, we show that for certain choices of parameters, hardness can be derived by simple gadget constructions and via a result of Guruswami and Lee on hypergraph colourings [47]; the main difficulty is in proving NP-hardness when the ratio $\frac{g}{k}$ is close to $\frac{s}{s+1}$. The tractability part of Theorem 3.3 is proved in Section 3.3 by a simple randomised algorithm based on classical work of Papadimitriou [63], just as in the Boolean case [8]. We also establish the (non-)applicability of certain convex relaxations. In Section 3.4 we define a new variant of the Label Cover problem and connect it with a general property of polymorphisms, and in Sections 3.5 and 3.6, we use this property to show hardness for SetSAT. We show in Section 3.7 that SetSAT has a rich collection of polymorphisms, which prevents us from using certain general NP-hardness criteria to prove hardness, and finally, in Section 3.8, we exhibit a connection between SetSAT and hypergraph colouring.

3.2 Simple reductions

Proposition 3.4. *For any $1 \leq s < d$, there is a polynomial-time reduction from (a, g, k)-SetSAT to $(a, g - 1, k)$-SetSAT.*

Proof. A g-satsifying assignment is also a $(g - 1)$-satisfying assignment. □

Proposition 3.5. *For any $1 \leq s < d$, there is a polynomial-time reduction from (a, g, k)-SetSAT to $(a, g, k + 1)$-SetSAT.*

Proof. For $i \in [s + 1]$, let $N_i(x) = \mathbb{1}[x \in [s + 1] \setminus \{i\}]$ be a literal not satisfied by i. For each original clause, create $s + 1$ new clauses by adding in turn each of the literals $N_i(y)$ to the original clause, where y is a variable not appearing in the original instance. Note that a g-satisfying assignment

to the original instance is also a g-satisfying assignment to the new instance. Conversely, if the original instance is not a-satisfiable, then neither is the new instance, as the literals $N_i(y)$ cannot simultaneously be satisfied in all the new clauses for any value of y. □

NP-hardness for the case $\frac{g}{k} \leq \frac{1}{2}$ is much easier to obtain when $d \geq 3$ compared to $d = 2$ [8], as shown below in Corollary 3.8. First we prove more directly that the generalisation of (1,1,3)-SAT to larger domains remains hard.

Proposition 3.6. *Let $s \geq 2$ and $d = s + 1$. Then $(1, 1, 3)$-SetSAT is NP-hard.*

Proof. We give a reduction from 3-SAT. We interpret the value 1 as false and 2 as true. To illustrate the reduction, consider the clause $(x_1 \vee \overline{x}_2 \vee \overline{x}_3)$. From this clause we create a new clause $(N_1(x_1) \vee N_2(x_2) \vee N_2(x_3))$ where $N_i(x)$ is defined as in the proof of Proposition 3.5. We add such a clause for each clause in the original 3-SAT instance. Then to enforce the binary nature of the original variables, we add the clauses $(N_i(x_j) \vee N_i(x_j) \vee N_i(x_j))$ for $3 \leq i \leq s+1$ and $1 \leq j \leq n$, which restrict the new variables to take values in $\{1, 2\}$.

If the 3-SAT instance is satisfiable, then the (1,1,3)-SetSAT instance is also satisfiable. Conversely, if the SetSAT instance is satisfiable, then its variables take only the values 1 and 2 and we can translate back to a satisfying assignment for the 3-SAT instance. □

The first deviation from the results of the SAT world is that the SetSAT analogue of 2-SAT is hard except in the case $s = 1$, which corresponds to 2-SAT.

Proposition 3.7. *Let $s \geq 2$ and $d = s + 1$. Then $(1, 1, 2)$-SetSAT is NP-hard.*

Proof. Each clause can be represented as $S_{a,b}(x, y)$, which forbids the assignment $(x, y) = (a, b)$. Therefore $\bigwedge_{i=1}^{s+1} S_{i,i}(x, y)$ is the disequality relation $x \neq y$, so we can simulate graph colouring, which is NP-hard for $s + 1 \geq 3$ colours. □

We can extend this result to larger clauses as follows.

Corollary 3.8. *Let $s \geq 2$ and $d = s + 1$. Then $(1, g, 2g)$-SetSAT is NP-hard for all $g \geq 1$.*

Proof. A reduction from $(1, 1, 2)$-SetSAT analogous to the reduction from $(1, 1, 3)$-SAT to $(1, g, 3g)$-SAT in [8] gives the result. The clauses of the new $(1, g, 2g)$-SetSAT instance are obtained by choosing, in all possible ways, g clauses from the $(1, 1, 2)$-SetSAT instance and taking the union of their literals.

If the $(1,1,2)$-SetSAT instance is satisfiable, then the obtained $(1,g,2g)$-SetSAT instance is g-satisfiable for the same assignment. Conversely, since the $(1,g,2g)$-SetSAT instance contains clauses made from copying an old clause g times, satisfiability of the new formula implies satisfiability of the old one, again for the same assignment. □

Finally we show that certain results on hypergraph colouring hardness obtained by Guruswami and Lee [47] already imply NP-hardness fairly close to the real boundary.

Proposition 3.9. *For $d = s+1$ and all $g \geq 1$, $(1, s(g-1), (s+1)g)$-SetSAT is NP-hard.*

Proof. We give a reduction from the following hypergraph colouring problem, whose hardness was proved in [47]. For $g, r, c \geq 2$, given as input a gr-uniform hypergraph that is promised to have an r-colouring where each colour appears at least $g-1$ times in every hyperedge, find a c-colouring that does not create a monochromatic hyperedge. The hardness reduction is as follows.

Let $r = c = s+1$. For each hyperedge $\{x_1, \ldots, x_{(s+1)g}\}$ we create, for $1 \leq i \leq s+1$, the SetSAT clauses $C_i = \big(N_i(x_1) \vee \ldots \vee N_i(x_{(s+1)g})\big)$, where N_i are as in the proof of Proposition 3.5. If the hypergraph instance has an $(s+1)$-colouring where every colour appears at least $g-1$ times in each hyperedge, then the obtained formula will be $s(g-1)$ satisfiable: Under the promised assignment, the clause C_i contains the group of satisfied literals whose variables are not equal to i, and there are at least $g-1$ literals in each of the s such groups.

Conversely, if the SetSAT formula is satisfied, the variables $x_1, \ldots, x_{(s+1)g}$ cannot all take the same value i for any i, as otherwise the clause C_i would be false. Therefore no hyperedge in the hypergraph is left monochromatic by a satisfying assignment. □

3.3 Tractability

How big must one make the fraction of satisfied literals in order for the SetSAT problem to become tractable? The following proposition shows that $\frac{s}{s+1}$ is sufficient.

Proposition 3.10. *For $1 \leq s < d$ and $\frac{g}{k} \geq \frac{s}{s+1}$, $(1,g,k)$-SetSAT is solvable in expected polynomial time.*

Proof. Algorithm 3 finds a satisfying assignment to a g-satisfiable formula in expected polynomial time. The algorithm and its analysis are based on [8, Proposition 6.1], which in turn is based on Papadimitriou's randomised algorithm for 2-SAT [63].

Algorithm 3 Randomised algorithm for $(1, g, k)$-SetSAT with $\frac{g}{k} \geq \frac{s}{s+1}$

Input: an instance Ψ of $(1, g, k)$-SetSAT
Output: a satisfying assignment to Ψ

1: $x \leftarrow$ arbitrary assignment
2: **while** x does not satisfy Ψ **do**
3: arbitrarily pick a falsified clause C of Ψ
4: randomly choose from C a literal $S(x_i)$
5: randomly choose a value for x_i so that $S(x_i)$ is satisfied
return x

Suppose that Ψ has a g-satisfying assignment x^*. Let x^t be the assignment obtained in iteration t of the algorithm, and let $D_t = \text{dist}(x^t, x^*)$, where $\text{dist}(x, y)$ is the Hamming distance between x and y. Since $D_t - D_{t-1} \in \{-1, 0, 1\}$ for every t,[1] we have

$$\begin{aligned}\mathbb{E}(D_t - D_{t-1}) &= \mathbb{P}(D_t - D_{t-1} = 1) - \mathbb{P}(D_t - D_{t-1} = -1) \\ &\leq \frac{k-g}{k} - \frac{g}{k}\frac{1}{s} \leq 0 \quad \text{if and only if} \quad \frac{g}{k} \geq \frac{s}{s+1}.\end{aligned}$$

The sequence $D_0, D_1, \ldots$ is a random walk starting between 0 and n with each step either unbiased or biased towards 0. This is a "gambler's ruin" chain with reflecting barrier (because the distance cannot increase beyond n). With constant probability, such a walk hits 0 ("the gambler is broke") within n^2 steps and the probability that the algorithm fails to find a satisfying assignment within crn^2 steps is at most 2^{-r} for some constant c, so the expected running time is $O(n^2)$. □

The proof of Proposition 3.10 can be modified to show that Algorithm 3 also finds a satisfying assignment when each literal corresponds to a set of size *at most* s. This makes sense intuitively, as smaller literals give the algorithm a better chance of setting x_i equal to x_i^* in Step 5.

We show that if $\frac{g}{k} \geq \frac{s}{s+1}$, then $(1, g, k)$-SetSAT has a specific family of polymorphisms that leads to a *deterministic* algorithm based on linear programming.

Definition 3.11. A symmetric function $f : [d]^m \to [d]$ is a *plurality* if

$$f(x_1, \ldots, x_m) = \text{argmax}_{a \in [d]}\{\#\text{ of occurrences of } a \text{ in } (x_1, \ldots, x_m)\},$$

with ties broken in such a way that f is symmetric.

[1] In the special case $d = 2$, we have $D_t - D_{t-1} \in \{-1, 1\}$.

When $d = s + 1$, all polymorphisms of SetSAT are *conservative*; i.e., they always return one of their input values, as the following proposition shows. This will be important when showing hardness, and also has implications for solvability by linear programming algorithms.

Proposition 3.12. *If $d = s + 1$, then all polymorphisms of $(1, g, k)$-SetSAT are conservative.*

Proof. Let $f : [d]^m \to [d]$ be such that $f(a_1, \ldots, a_m) = b$ and $b \notin \{a_1, \ldots, a_m\}$. If S is a literal not containing b, then the clause $(S(x_1) \vee \ldots \vee S(x_k))$ is g-satisfied (even k-satisfied) by setting all x_i equal to any one of the a_j. Thus taking the m assignments $(x_1 = \cdots = x_k = a_j)_{1 \le j \le m}$ and applying f to each component, we get the assignment $x_1 = \cdots = x_k = b$ which clearly does not 1-satisfy the clause, and so f cannot be a polymorphism. □

Proposition 3.13. *Let $1 \le s < d$. If $\frac{g}{k} > \frac{s}{s+1}$ then every plurality function is a polymorphism of $(1, g, k)$-SetSAT. If $\frac{g}{k} = \frac{s}{s+1}$ then every plurality function of arity $m \not\equiv 0 \mod s+1$ is a polymorphism of $(1, g, k)$-SetSAT, and no symmetric function of arity $m \equiv 0 \mod s + 1$ is a polymorphism of $(1, g, k)$-SetSAT if we also have $d = s + 1$.*

Proof. Let f be a plurality function of arity m. Given m g-satisfying assignments to a clause of width k, we are guaranteed to have at least mg satisfying values among the mk total values. Therefore there is a coordinate i, $1 \le i \le k$, containing at least $\lceil \frac{mg}{k} \rceil$ satisfying values, that is, at least $\lceil \frac{mg}{k} \rceil$ values not equal to any of the values $b_1, \ldots, b_{d-s}$ forbidden by the i-th literal of the clause. For f to be a polymorphism, it suffices that the most frequent satisfying value in coordinate i appears more often than any of the $b_1, \ldots, b_{d-s}$, for which it suffices that $\lceil \frac{mg}{k} \rceil / s > m - \lceil \frac{mg}{k} \rceil$. This is equivalent to $\lceil \frac{mg}{k} \rceil > \frac{s}{s+1} m$, which follows from $\frac{g}{k} > \frac{s}{s+1}$, so f is a polymorphism.

In the case $\frac{g}{k} = \frac{s}{s+1}$, the same argument works so long as $\frac{mg}{k}$ is not an integer, since by taking the ceiling we obtain a value strictly greater than $\frac{s}{s+1} m$. Since $\frac{g}{k} = \frac{s}{s+1}$, we have $\frac{g}{k} m = \frac{s}{s+1} m$ and this is an integer only if m is a multiple of $s + 1$.

To show that there are no symmetric polymorphisms when $\frac{g}{k} = \frac{s}{s+1}$, m is a multiple of $s+1$, and $d = s+1$, note that $\frac{g}{k} = \frac{s}{s+1}$ implies that k is divisible by $s+1$. Let M be the $(s+1) \times (s+1)$ matrix whose first row is $12 \cdots s+1$ and whose i-th row for $2 \le i \le s+1$ is obtained from the $(i-1)$-st row by shifting it cyclically to the left by one coordinate. We stack $\frac{k}{s+1}$ copies of M on top of each other and take $\frac{m}{s+1}$ copies of this stack side-by-side to form the $k \times m$ matrix M'. If f is symmetric, it returns the same value $b \in \{1, \ldots, s+1\}$ when applied to each row of M'. Every column of M' satisfies exactly an $\frac{s}{s+1}$-fraction of the literals in the clause whose k literals are all $S_{\{1,\ldots,s+1\} \setminus \{b\}}$. On

the other hand, the assignment produced by applying f to each row of M' does not even 1-satisfy this clause, so f is not a polymorphism. □

Proposition 3.13 has interesting consequences for solvability of $(1, g, k)$-SetSAT via convex relaxations. By Theorem 2.58, $(1, g, k)$-SetSAT is solvable by BLP when $\frac{g}{k} > \frac{s}{s+1}$ (since there exist symmetric polymorphisms of all arities) but not solvable by BLP when $\frac{g}{k} = \frac{s}{s+1}$ and $d = s + 1$ (since there do not exist symmetric polymorphisms of all arities). By Theorem 2.61, $(1, g, k)$-SetSAT is solvable by BLP+AIP if $\frac{g}{k} \geq \frac{s}{s+1}$ (since there exist symmetric polymorphisms of infinitely many arities). We note that iterative rounding of the BLP relaxation could also be used to get a deterministic algorithm as in [8].

3.4 Layered Label Cover and smug sets

In this section we define a variant of the Label Cover problem, which reduces the task of showing hardness to showing that all polymorphisms satisfy a certain combinatorial property, and then in Section 3.5, we show that the polymorphisms of SetSAT satisfy this property. From now on we assume that $d = s + 1$, as in Theorem 3.3.

An *ℓ-Layered Label Cover* instance is a sequence of $\ell+1$ sets $X_0, \ldots, X_\ell$ (called *layers*) of variables with range $[m]$, for some *domain size* $m \in \mathbb{N}$, and a set of constraints Φ. Each constraint is a function (often called a projection constraint) from a variable $x \in X_i$ to a variable in a further layer $y \in X_j$, $i < j$, that is, a function denoted $\phi_{x \to y}$ which is satisfied by an assignment $\sigma \colon X_0 \cup \cdots \cup X_\ell \to [m]$ if $\sigma(y) = \phi_{x \to y}(\sigma(x))$. A *chain* is a sequence of variables $x_i \in X_i$ for $i = 0, \ldots, \ell$ such that there are constraints $\phi_{x_i \to x_j}$ between them, for $i < j$. A chain is *weakly satisfied* if at least one of these constraints is satisfied.

The basis for our hardness result is the hardness of distinguishing fully satisfiable instances from those where no constant fraction of chains can be weakly satisfied. This follows by a simple adaptation of a reduction from the work of Dinur, Guruswami, Khot, and Regev [35].

Theorem 3.14. *For every $\ell \in \mathbb{N}$ and $\varepsilon > 0$, there is an $m \in \mathbb{N}$ such that it is NP-hard to distinguish ℓ-Layered Label Cover instances with domain size m that are fully satisfiable from those where not even an ε-fraction of all chains is weakly satisfied.*

Proof. For $\ell = 1$ a chain consists of just one constraint, so weakly satisfying the chain is the same as satisfying its constraint. The claim is then equivalent to the hardness of the standard Bipartite Gap

Label Cover problem, which holds even for *bi-regular* instances, that is, instances (Y, Z) such that every variable in Y occurs in constraints with exactly d_+ variables in Z and every variable in Z occurs in constraints with exactly d_- variables in Y, for some $d_+, d_- \in \mathbb{N}$. (This hardness follows from the PCP theorem [2, 3] and Raz's parallel repetition theorem [64], cf. [35] and [1].)

For $\ell > 1$ we reduce from a bi-regular instance of Bipartite Gap Label Cover with variable sets Y and Z, domain size m, constraints Γ and gap $\varepsilon' := \varepsilon/\binom{\ell+1}{2}$. Let the domain size of the constructed instance Φ be m^ℓ. Let the variable sets be $X_i := Z^i \times Y^{\ell-i}$ for $i = 0, \dots, \ell$ (that is, ℓ-tuples of i variables from Z followed by $\ell - i$ variables from Y; this makes indices notationally more convenient than the other way around). Let the constraints between X_i and X_j (for $0 \le i < j \le \ell$) be defined for pairs of tuples $\bar{x}$ and $\bar{x}'$ of the form:

$$\bar{x} = (z_1, \dots, z_i,\ \ y_{i+1}, \dots, y_j,\ \ y_{j+1}, \dots, y_\ell) \in X_i \quad \text{and}$$
$$\bar{x}' = (z_1, \dots, z_i,\ \ z_{i+1}, \dots, z_j,\ \ y_{j+1}, \dots, y_\ell) \in X_j$$

such that the original instance has a constraint $\phi_{y_k \to z_k} \in \Gamma$ for $k = i+1, \dots, j$. Let the new projection constraint $\phi_{\bar{x} \to \bar{x}'}$ map $(a_1, \dots, a_\ell)$ to $(b_1, \dots, b_\ell)$ where $b_k := \phi_{y_k \to z_k}(a_k)$ for $k = i+1, \dots, j$ and $b_k := a_k$ otherwise. This concludes the construction.

Note that chains in this instance are in bijection with ℓ-tuples of original constraints in Γ. Indeed, a chain $\bar{x}_i \in X_i$ $(i = 0, \dots, \ell)$ is determined by $\bar{x}_0 = (y_1, \dots, y_\ell)$ and $\bar{x}_\ell = (z_1, \dots, z_\ell)$ such that Γ has constraints $\phi_{y_k \to z_k}$ for $k = 1, \dots, \ell$. Moreover, for each $i < j$, every constraint $\phi_{\bar{x} \to \bar{x}'}$ between $\bar{x} \in X_i$ and $\bar{x}' \in X_j$ appears in the same number of chains (namely ${d_-}^i \cdot {d_+}^{\ell-j}$).

If the original instance Γ was fully satisfiable then so is the new one Φ: indeed, if σ is a satisfying assignment for Γ, then $\bar{x} \mapsto (\sigma(x_1), \dots, \sigma(x_\ell))$ is a satisfying assignment for Φ.

Suppose now that in Φ, an assignment $\sigma \colon X_0 \cup \dots \cup X_\ell \to [m]^\ell$ weakly satisfies at least ε of all chains. Then there exists $0 \le i < j \le \ell$ such that at least $\varepsilon/\binom{\ell+1}{2} = \varepsilon'$ of all chains are weakly satisfied at a constraint between X_i and X_j. Every constraint between X_i and X_j is contained in the same number of chains, say C, hence at least ε' of the constraints between X_i and X_j are satisfied (indeed, the number of thus satisfied chains is exactly C times the number of satisfied constraints; similarly, the number of all chains is exactly C times the number of all constraints between X_i and X_j).

Choose an arbitrary coordinate k in $i+1, \dots, j$. Partition X_i into equivalence classes such that $\bar{x}, \bar{x}'$

are in the same class if they are identical on all coordinates except possibly coordinate k. Partition X_j in the same way. There exists a pair of classes between which constraints exist and at least ε' of them are satisfied. That is, there are

$$x_1, \dots, x_{k-1},\ x_{k+1}, \dots, x_\ell \in Y \cup Z \quad \text{and}$$
$$x'_1, \dots, x'_{k-1},\ x'_{k+1}, \dots, x'_\ell \in Y \cup Z$$

such that σ satisfies at least ε' of the constraints between pairs of the form

$$(x_1, \dots, x_{k-1},\ y,\ x_{k+1}, \dots, x_\ell) \in X_i$$
$$(x'_1, \dots, x'_{k-1},\ z,\ x'_{k+1}, \dots, x'_\ell) \in X_j$$

where a constraint $\phi_{y\to z}$ exists in Γ. Therefore, one can define an assignment $\sigma' \colon Y \cup Z \to [m]$ by letting $\sigma'(y)$ and $\sigma'(z)$ be the k-th element of the value in $[m]^\ell$ resulting from applying σ to the above tuples, respectively for $y \in Y$ and $z \in Z$. This assignment then satisfies at least ε' of all the constraints $\phi_{y\to z}$ of the original instance Γ. □

A piece of notation before we prove a corollary of Theorem 3.14. A *chain of minors* is a sequence of the form $f_0 \xrightarrow{\pi_{0,1}} f_1 \xrightarrow{\pi_{1,2}} \dots \xrightarrow{\pi_{\ell-1,\ell}} f_\ell$. We shall then write $\pi_{i,j} \colon [\mathrm{ar}(f_i)] \to [\mathrm{ar}(f_j)]$ for the composition of $\pi_{i,i+1}, \dots, \pi_{j-1,j}$, for any $0 \le i < j \le \ell$. Note that $f_i \xrightarrow{\pi_{i,j}} f_j$.

Corollary 3.15 (of Theorem 3.14)**.** *Let $\mathcal{M}$ be a minion. Suppose there are constants $k, \ell \in \mathbb{N}$ and an assignment of a set of at most k coordinates $\mathrm{sel}(f) \subseteq [\mathrm{ar}(f)]$ to every $f \in \mathcal{M}$ such that for every chain of minors $f_0 \xrightarrow{\pi_{0,1}} f_1 \xrightarrow{\pi_{1,2}} \dots \xrightarrow{\pi_{\ell-1,\ell}} f_\ell$, there are $0 \le i < j \le \ell$ such that $\pi_{i,j}(\mathrm{sel}(f_i)) \cap \mathrm{sel}(f_j) \neq \emptyset$. Then $\mathrm{PMC}_{\mathcal{M}}(m)$ is NP-hard, for m large enough. In particular, if $\mathcal{M} = \mathrm{Pol}(\mathbf{A}, \mathbf{B})$, then $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is NP-hard.*

Proof. For ℓ, k as in the assumption, let $\varepsilon := \frac{1}{k^2}$ and let m be as given by Theorem 3.14. We reduce an ℓ-Layered Label Cover instance by replacing each variable x with a symbol f_x of arity m and each constraint $\phi_{x\to y} \colon [m] \to [m]$ by the minor condition $f_x \xrightarrow{\phi_{x\to y}} f_y$. If the original instance was fully satisfiable, the new instance is trivial (i.e., fully satisfiable by projections).

If the constructed instance is satisfied by functions in the minion $\mathcal{M}$, we define an assignment to the original instance by selecting, for each variable x, a random coordinate from $\mathrm{sel}(f_x) \subseteq [m]$ (uniformly, independently). The assumption guarantees a set of constraints $\phi_{x\to y}$ such that (1) each

chain contains at least one and (2) for each such constraint $\phi_{x\to y}$, we have $\phi_{x\to y}(\mathrm{sel}(f_x))\cap\mathrm{sel}(f_y)\neq\emptyset$. The random choice then satisfies each of these constraints, and hence weakly satisfies each chain, with probability at least $\frac{1}{k^2}=\varepsilon$. The expected fraction of weakly satisfied chains is thus at least ε and a standard maximisation-of-expectation procedure deterministically finds an assignment which certifies this. □

We now introduce the combinatorial property of polymorphisms which is crucial for our results.

Definition 3.16. For a function $f\colon A^{\mathrm{ar}(f)}\to B$ we say that a set of coordinates $S\subseteq[\mathrm{ar}(f)]$ is a *smug* set if there is an input vector $v\in A^{\mathrm{ar}(f)}$ such that $S=\{i\mid v_i=f(v)\}$.

Since $d=s+1$, Proposition 3.12 applies and so for every polymorphism f of SetSAT and every $\bar{v}$, the set $\{i\mid v_i=f(\bar{v})\}$ is nonempty. The following result connects Layered Label Cover to smug sets and will be used to prove hardness of $(1,g,k)$-SetSAT.

Corollary 3.17. *Let $\mathcal{M}$ be a minion. Suppose there are constants $k,\ell\in\mathbb{N}$ such that the following holds, for every $f\in\mathcal{M}$:*

1. *f has a smug set of at most k coordinates,*
2. *f has no family of more than ℓ (pairwise) disjoint smug sets,*
3. *if $f\xrightarrow{\pi}g$ and S is a smug set of g, then $\pi^{-1}(S)$ is a smug set of f.*

Then $\mathrm{PMC}_{\mathcal{M}}(m)$ is NP-hard, for m large enough. In particular, if $\mathcal{M}=\mathrm{Pol}(\mathbf{A},\mathbf{B})$, then $\mathrm{PCSP}(\mathbf{A},\mathbf{B})$ is NP-hard.

Proof. For each $f\in\mathcal{M}$, we define $\mathrm{sel}(f)$ as a smug set of at most k coordinates, arbitrarily chosen (some such set exists by the first condition). Consider a chain $f_0\xrightarrow{\pi_{0,1}}f_1\xrightarrow{\pi_{1,2}}\dots\xrightarrow{\pi_{\ell-1,\ell}}f_\ell$. Suppose to the contrary that for each $0\le i<j\le\ell$, $\pi_{i,j}(\mathrm{sel}(f_i))$ is disjoint from $\mathrm{sel}(f_j)$, or equivalently, that $\mathrm{sel}(f_i)$ is disjoint from $\pi_{i,j}^{-1}(\mathrm{sel}(f_j))$. This implies that $\pi_{0,i}^{-1}(\mathrm{sel}(f_i))$ is disjoint from $\pi_{0,i}^{-1}(\pi_{i,j}^{-1}(\mathrm{sel}(f_j)))=\pi_{0,j}^{-1}(\mathrm{sel}(f_j))$. That is, the sets $\pi_{0,i}^{-1}(\mathrm{sel}(f_i))$ for $i=0\dots\ell$ are pairwise disjoint. By the third condition they are smug sets of f_0. But by the second condition this is impossible. □

We note that in the proof of Corollary 3.17, the exact definition of "smug" is irrelevant, as long as it satisfies the above three conditions.

It is easy to check that the definition of "smug" satisfies the third condition for any functions $f\xrightarrow{\pi}g$, not necessarily polymorphisms. Indeed, if an input $\bar{v}\in A^{\mathrm{ar}(g)}$ to g gives a smug set $S=\{j\mid v_j=g(\bar{v})\}$,

then the corresponding input $\bar{u} \in A^{\mathrm{ar}(f)}$ to f defined as $u_i := v_{\pi(i)}$ satisfies $f(\bar{u}) = g(\bar{v})$ and hence gives a smug set $\{i \mid u_i = f(\bar{u})\} = \{i \mid v_{\pi(i)} = g(\bar{v})\} = \{i \mid \pi(i) \in S\} = \pi^{-1}(S)$.

The definition of "smug" is particularly well-suited to our problem, because whether f is a polymorphisms or not depends only on its family of smug sets.

Lemma 3.18. *Let $1 \leq s$ and $1 \leq g < k$. A function $f\colon [s+1]^m \to [s+1]$ is a polymorphism of $(1, g, k)$-SetSAT if and only if there is no multiset $S_1, \ldots, S_k$ of smug sets of f, such that each coordinate $\ell \in [m]$ is contained in at most $k - g$ of them.*

Before setting out to prove Lemma 3.18, we give an example in Figure 3.1 of a function that is not a polymorphism of $(1, 3, 5)$-SetSAT with set size 2 and domain size 3.

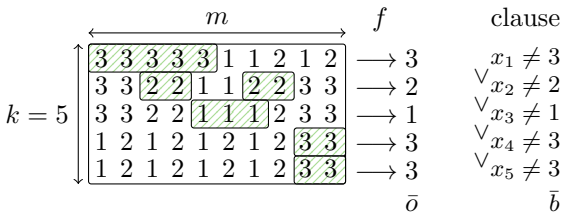

Figure 3.1: Illustration of Lemma 3.18. Smug sets $S \subseteq [m]$ are highlighted in each row.

Proof of Lemma 3.18. A function $f\colon [s+1]^m \to [s+1]$ is not a polymorphism if and only if there is a clause of the form $x_1 \neq b_1 \vee \cdots \vee x_k \neq b_k$ (for some column vector $\bar{b} \in [s+1]^k$) and a sequence of m column vectors $\bar{v}^1, \ldots, \bar{v}^m \in [s+1]^k$ each of which g-satisfies the clause, but for which the vector $\bar{o} = f(\bar{v}^1, \ldots, \bar{v}^m)$ (with f applied coordinatewise) does not even 1-satisfy the clause. The latter is equivalent to saying that $o_i = b_i$ for $i \in [k]$, that is, applying f to the i-th row gives $f(v_i^1, \ldots, v_i^m) = b_i$. The former is equivalent to saying that for each column $\bar{v}$ in $\bar{v}^1, \ldots, \bar{v}^m$, the condition $v_i \neq b_i$ holds for at least g indices $i \in [k]$ of that column. The two are hence equivalent to saying that for each column $\bar{v}^\ell$, $\ell \in [m]$, the condition $v_i^\ell = f(v_i^1, \ldots, v_i^m)$ holds for at most $k - g$ indices $i \in [k]$ in that column. In other words, the k row vectors $(v_i^1, \ldots, v_i^m)$ for $i \in [k]$ have smug sets such that ℓ is contained in at most $k - g$ of these sets, for each coordinate $\ell \in [m]$. □

Checking the second condition for polymorphisms of our SetSAT problem is easy.

Lemma 3.19. *For every polymorphism f of $(1, g, k)$-SetSAT, if $S_1, \ldots, S_t$ are disjoint smug sets of f, then $t < \frac{k}{k-g}$.*

Proof. Suppose to the contrary that $t \geq \frac{k}{k-g}$. Then we can build a multiset containing each S_i up to $k-g$ times until we have exactly k in total. We thus obtain a multiset of k smug sets such that every coordinate is contained in at most $k-g$ of them. □

We have now shown that the second and third conditions of Corollary 3.17 hold, so it remains only to show that every polymorphism of SetSAT has small smug sets.

3.5 Finding small smug sets

It is easy to show NP-hardness when $\frac{g}{k} \leq \frac{1}{2}$ (cf. Proposition 3.8). We now show a general reduction by finding a small smug set for $(1, g, k)$-SetSAT whenever $\frac{g}{k} < \frac{s}{s+1}$. Again we assume $d = s + 1$.

Lemma 3.20. *Let $f\colon [s+1]^m \to [s+1]$ be a polymorphism of $(1, g, k)$-SetSAT with set size s and domain size $s+1$. There exists a smug set of f of size at most $s-1$, or a family of s disjoint minimal smug sets $S_1, \ldots, S_s$.*

Proof. Suppose that every smug set has size at least s. We show by induction on t that there is a family of t disjoint minimal smug sets $S_1, \ldots, S_t$. Suppose we found $S_1, \ldots, S_t$ for some $0 \leq t < s$ and we want to find S_{t+1}. Let T be a set containing one arbitrary coordinate from each S_i, $i = 1 \ldots t$. Let $\bar{v} \in [s+1]^m$ be the input vector with values $t+2$ on T, i on $S_i \setminus T$ (for $i = 1 \ldots t$) and $t+1$ on the remaining coordinates $R := [m] \setminus (S_1 \cup \cdots \cup S_t)$. Since $|T| \leq t < s$, T is not smug, so $f(\bar{v}) \neq t+2$. By minimality, $S_i \setminus T$ are not smug for $i = 1 \ldots t$, so $f(\bar{v}) \neq i$. Therefore, by conservativity of f (Proposition 3.12), the only remaining option is $f(\bar{v}) = t+1$. Thus R is smug and disjoint from S_i. Taking S_{t+1} to be a minimal smug set contained in R proves the induction step. □

Together with Lemma 3.19, Lemma 3.20 already establishes (via Corollary 3.17) NP-hardness when $s \geq \frac{k}{k-g} = \frac{g}{k-g} + 1$ (equivalently, $\frac{g}{k} \leq \frac{s-1}{s}$): since there cannot be s disjoint smug sets, every polymorphism has a smug set of size at most $s-1$. The proof in the general case, when $\frac{g}{k} < \frac{s}{s+1}$, extends this approach by first finding (assuming there are no small smug sets) disjoint minimal smug sets $S_1, \ldots, S_s$, then exploiting the fact that each has a special coordinate whose removal makes it not smug, and using these coordinates to find further variants of each S_i with new special coordinates.

Lemma 3.21. *Let $\frac{g}{k} < \frac{s}{s+1}$ and let $f\colon [s+1]^m \to [s+1]$ be a polymorphism of $(1, g, k)$-SetSAT with set size s and domain size $s+1$. Then f has a smug set of size at most g.*

Proof. Consider a polymorphism $f\colon [s+1]^m \to [s+1]$ of $(1,g,k)$-SetSAT. We prove by induction on t that there is a smug set of size at most $t-1$, or there is a sequence of smug sets $S_1,\dots,S_t$ and a set T such that (see Figure 3.2):

(i) $|T| = t$ and $|T \cap S_i| = 1$ for $i = 1 \dots t$ (and by the construction below, $S_i \cap T \neq S_{i'} \cap T$ for $i \neq i'$);

(ii) $S_i \setminus T$ is not smug for $i = 1 \dots t$;

(iii) $S_i \cap S_{i'} = \emptyset$ if $i \not\equiv i' \mod s$;

(iv) $S_i \supseteq S_{i-s} \setminus T$ for $i > s$.

By Lemma 3.20 we can start with $t = s$ (by taking any T containing one coordinate from each S_i). Suppose the above is true for $t \geq s$ and let us prove the same for $t+1$. If there is a smug set of size at most t then we are done, so assume that T is not smug. Let $\bar{v} \in [s+1]^m$ be the input vector with value $s+1$ on T and different values from $\{1,\dots,s\}$ on $S_{t-i} \setminus T$ for $i = 0 \dots s-2$ and on the set of remaining coordinates $R := [m] \setminus (S_t \cup \dots \cup S_{t-s+2} \cup T)$. Then by (ii), R is smug.

Observe that R contains $S_{t-s+1} \setminus T$, because $S_t,\dots,S_{t-s+2},T$ are disjoint from that set by (iii). We define S_{t+1} to be a minimal subset of R among smug sets containing $S_{t-s+1} \setminus T$. By (ii), $S_{t-s+1} \setminus T$ itself is not smug, so there exists some coordinate ℓ in $S_{t+1} \setminus S_{t-s+1}$. We choose it arbitrarily and set $T' := T \cup \{\ell\}$.

We claim that the sequence of smug sets $S_1,\dots,S_{t+1}$ and the set T' satisfy the above conditions. By minimality $S_{t+1} \setminus T'$ is not smug, so it satisfies (ii) and by definition it satisfies (iv). The set S_{t+1} is disjoint from $S_t,\dots,S_{t-s+2},T$, because R was. It is also disjoint from S_i for $i \not\equiv t+1 \mod s$, because for every such i, $S_i \setminus T$ is contained in one of $S_t,\dots,S_{t-s+2}$; this proves (iii). In particular ℓ is not contained in any of these sets, and since it is not contained in S_{t-s+1}, it is in fact not contained in any S_i with $i < t+1$. Hence $|T'| = t+1$ and $|T' \cap S_i| = |T \cap S_i| = 1$ for $i < t+1$. Clearly also $|T' \cap S_{t+1}| = |\{\ell\}| = 1$. Therefore, (i) is satisfied, concluding the inductive proof.

Let us now consider sets as guaranteed above for $t = g+1$ (assuming there is no smug set of size at most g). Let $\bar{v} \in [s+1]^m$ be the input vector with value $i+1$ on $S_{t-i} \setminus T$ for $i = 0 \dots s-1$, and value $s+1$ on the remaining coordinates $R := ([m] \setminus (S_t \cup \dots \cup S_{t-s+1})) \cup T$. By (ii) the sets $S_{t-i} \setminus T$ are not smug, so R is smug. We claim that the multiset obtained from $\{S_1,\dots,S_t\}$ by adding $(k-g-1)$ copies of the set R contradicts Lemma 3.18: that is, each coordinate in $[m]$ is covered at most $k-g$ times by this multiset.

Consider first the coordinates contained in R. By definition of R, they are disjoint from $S_{t-i} \setminus T$ for $i = 0 \dots s-1$. By (iv), they are also disjoint from all sets $S_i \setminus T$ for $i = 0 \dots t$, because every such set is contained in one of the former. Hence if a coordinate in R is also contained in one of $S_1, \dots, S_t$, then it is contained in T and therefore in at most one of $S_1, \dots, S_t$, by (i). In total, it is thus covered at most $(k-g-1)+1 = k-g$ times.

Consider now coordinates outside of R. By (iii), they can be covered only by sets S_i with congruent indices $i \mod s$. Since $s > \frac{g}{k-g}$, we have $s(k-g) > g$, so there are $t = g+1 \leq s(k-g)$ distinct indices in total in $\{1, \dots, t\}$. Hence at most $k-g$ of them can be pairwise congruent to each other mod s. Thus coordinates outside of R are also covered at most $k-g$ times. □

This concludes the proof that smug sets satisfy the first condition of Corollary 3.17 for polymorphisms of $(1, g, k)$-SetSAT with set size s and domain size $s+1$, assuming $\frac{g}{k} < \frac{s}{s+1}$. Therefore, the problem is NP-hard and we have established the hardness part of Theorem 3.3.

3.6 The general case

In this section we show how Theorem 3.3 implies Theorem 3.2. This amounts to arguing that a classification of (a, g, k)-SetSAT can be obtained from the special case with $a = 1$ and $d = s+1$. We start with two easy reductions.

Proposition 3.22. *For any $1 \leq s < d$, the problems (a, g, k)-SetSAT and $(a+1, g+1, k+1)$-SetSAT are polynomial-time reducible to each other.*

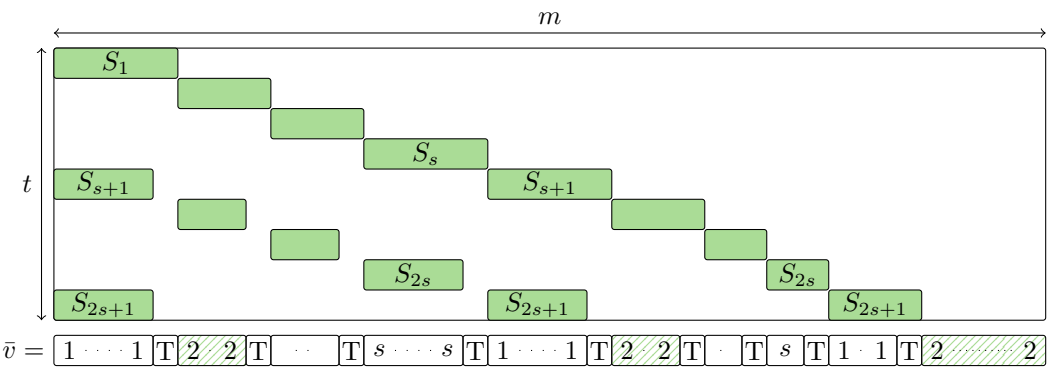

Figure 3.2: Illustration of smug sets obtained in the proof of Lemma 3.21. Each row represents one of the sets in the sequence $S_1, \dots, S_t$. The set T is formed by coordinates with a T and get values $s+1$. The vector $\bar{v}$ is used to find the next row S_{t+1}.

Proof. To reduce (a, g, k)-SetSAT to $(a+1, g+1, k+1)$-SetSAT, introduce a new variable y and add $S(y)$ to each existing clause, where S is any literal. If the original instance has a g-satisfying assignment, then the same assignment, extended by assigning y to a value satisfying S, is a $(g+1)$-satisfying assignment to the new instance. Conversely, if the old instance is not a-satisfiable, then the new instance cannot be $(a+1)$-satisfiable, as each new clause contains at most one additional satisfied literal.

In the other direction, from $(a+1, g+1, k+1)$-SetSAT to (a, g, k)-SetSAT, let Ψ be the orignal instance. For each clause C of Ψ we make $k+1$ new clauses by taking all subsets of k literals of C. If Ψ has a $(g+1)$-satisfying assignment, then the same assignment is g-satisfying for the new instance since we have removed only one literal from each clause of Ψ. Conversely, if every assignment to Ψ is not $(a+1)$-satisfying, then every assignment is at most a-satisfying. Removing one of the satisfied literals from a clause C of Ψ creates a new clause that is at most $(a-1)$-satisfiable. Therefore, in the new instance, every assignment is at most $(a-1)$-satisfying. □

Proposition 3.23. *There is a polynomial-time reduction from $(1, g, k)$-SetSAT with set size s and domain size d to $(1, g, k)$-SetSAT with set size s and domain size $d+1$.*

Proof. The new instance produced by the reduction is the same as the old instance Ψ. If Ψ is g-satisfiable, then it is again g-satisfiable by the same assignment and we ignore the new domain value. Conversely, a satisfying assignment over $[d+1]$ to Ψ restricts to a satisfying assignment over $[d]$ by replacing $d+1$ with any value from $[d]$. This does not falsify any literals since all the literals of Ψ range over $[d]$ only. □

Proof of Theorem 3.2. By Proposition 3.22, we can assume $a = 1$. The algorithm in Proposition 3.10 solves the problem in polynomial time as long as $\frac{g}{k} \geq \frac{s}{s+1}$ (independent of d). Theorem 3.3 states that $(1, g, k)$-SetSAT is NP-hard when $\frac{g}{k} < \frac{s}{s+1}$ and $d = s+1$. Proposition 3.23 then extends this to larger d. □

We finish this section by proving that literals described by sets of size less than s can be emulated by literals of size exactly s.

Proposition 3.24. *If $s \leq d-2$, there is a polynomial-time reduction from $(1, g, k)$-SetSAT with set size s and domain size d to $(1, g, k)$-SetSAT with set size $s+1$ and domain size d.*

Proof. We replace each clause $S_1(x_1) \vee \cdots \vee S_k(x_k)$ with a set of $(d-s)^k$ clauses $S'_1(x_1) \vee \cdots \vee S'_k(x_k)$, where S'_i ranges over all supersets of S_i of size $s+1$. Any g-satisfying assignment to the former clearly satisfies the latter. For a 1-satisfying assignment σ to the latter, we claim that for every new clause $S'_1(x_1) \vee \cdots \vee S'_k(x_k)$, at least one of the literals in $S_1(x_1) \vee \cdots \vee S_k(x_k)$ must be satisfied by σ. Suppose to the contrary that $\sigma(x_i) = a_i$ where $a_i \notin S_i$ for all $1 \leq i \leq k$. Then the clause formed by the literals $S'_i = S_i \cup \{b_i\}$, where $b_i \in [d] \setminus (S_i \cup \{a_i\})$, would not be satisfied by σ, a contradiction. Note that as $d - s \geq 2$, the set $[d] \setminus (S_i \cup \{a_i\})$ is non-empty. □

3.7 Impossibility results

Here we show that $(1, g, k)$-SetSAT has rich polymorphisms (satisfying many non-trivial minor conditions), even in the NP-hard range of parameters. We thus demonstrate that certain sufficient condition for NP-hardness from [10] and earlier work cannot be used to establish hardness of $(1, g, k)$-SetSAT for non-Boolean domains.

We show that polymorphisms of SetSAT satisfy robust conditions and therefore the assumptions of Theorem 2.65 and Theorem 2.75 are not met. The same construction will also give polymorphisms excluding other approaches (e.g. polymorphisms without small fixing sets). We first describe how to reconstruct a polymorphism from a family of smug sets.

Definition 3.25. Consider $(1, g, k)$-SetSAT with domain size $s + 1$. Let U be a finite set and let $\mathscr{S} = (S_1, \ldots, S_{|\mathscr{S}|})$ be a sequence of non-empty subsets of U with the following properties:

- for every partition $U = U_1 \cup \cdots \cup U_{s+1}$ into $s+1$ possibly empty sets, at least one of $U_1, \ldots, U_{s+1}$ is in $\mathscr{S}$.
- for every k-tuple $(S_{i_1}, \ldots, S_{i_k}) \in \mathscr{S}^k$, some $u \in U$ is contained in at least $k - g + 1$ of the k sets.

Let $q_{\mathscr{S}} \colon [s+1]^{|U|} \to [s+1]$ be defined as follows. For an input $\bar{x} \in [s+1]^{|U|}$, partition the coordinates according to their value: that is, for $i \in [s+1]$ let $U_i := \{u \in U \colon x_u = i\}$. Let $q_{\mathscr{S}}(\bar{x})$ be the value $i \in [s+1]$ such that $U_i \in \mathscr{S}$; if there are many such i, choose U_i to be first in the sequence $\mathscr{S}$.

By construction, all the smug sets of $q_{\mathscr{S}}$ are contained in $\mathscr{S}$. By Lemma 3.18, $q_{\mathscr{S}}$ is a polymorphism. Note that because of the preference for earlier sets in $\mathscr{S}$, not all sets in $\mathscr{S}$ have to be smug, and there may exist different functions with the same family of smug sets. On the other hand, the ordering in $\mathscr{S}$ matters only when comparing disjoint sets.

The following polymorphisms satisfy many non-trivial minor conditions. For notational convenience we consider only the case $k - g + 1 = 3$.

Definition 3.26. For $m \in \mathbb{N}$, let $U := \binom{[m]}{3} \cup \{\bot\}$. That is, we will index coordinates with triples $\{i_1, i_2, i_3\}$ in $[m]$, with one additional special coordinate $\bot$. For $i \in [m]$, let $S_i \subseteq U$ be the set of triples containing i. Let $\mathscr{S}_m$ be the family of all supersets of sets in $\{S_1, \dots, S_m, \{\bot\}\}$, ordered so that sets not containing $\bot$ are all earlier than sets containing $\bot$. Let $q_m := q_{\mathscr{S}_m}$.

Observe that the minimal smug sets of q_m are exactly $S_1, \dots, S_m, \{\bot\}$. Note also that every two sets in $S_1, \dots, S_m$ (and hence any two of their supersets) intersect, so the ordering between them is irrelevant, and similarly for every two sets containing $\bot$; hence q_m is defined unambiguously.

Proposition 3.27. *Let $m \geq 4$, $k - g + 1 = 3$, and $\frac{g}{k} > \frac{1}{2}$. Then*

(i) q_m is a polymorphism of $(1, g, k)$-SetSAT of arity $\binom{m}{3} + 1$;

(ii) q_m and projections of arity $m + 1$ satisfy a $\frac{4}{m}$-robust minor condition;

(iii) for every partial assignment to less than $\frac{m}{3}$ coordinates of q_m and every value $a \in [s + 1]$, there is an assignment to the remaining coordinates that makes q_m take value a. (In particular this means q_m does not have small "weakly fixing" or "avoiding" sets).

Proof. To check (i), we have to check that $\mathscr{S}_m$ satisfies the two conditions of Definition 3.25. The first condition is trivial because all sets containing $\bot$ are in $\mathscr{S}_m$. To check the second condition, suppose for contradiction that some k-tuple of sets in $\mathscr{S}_m$ covers every coordinate at most $k - g = 2$ times. In particular $\bot$ would be covered at most 2 times, leaving at least $g \geq k - g + 1 = 3$ sets not containing $\bot$. By definition of $\mathscr{S}_m$ these three sets would be supersets of $S_{i_1}, S_{i_2}, S_{i_3}$ respectively for some $i_1, i_2, i_3 \in [m]$ (not necessarily distinct), hence taking $I \in \binom{[m]}{3} \subseteq U$ to be any triple containing $\{i_1, i_2, i_3\}$, we see that the coordinate I is covered by all three sets, and hence by some $3 = k - g + 1$ sets.

For (ii), we start with an informal description; the formal argument is below. Let us first consider a minor of q_m defined by identifying all coordinates that are triples containing some $i \in [m]$. Observe that this minor is a projection to the resulting coordinate, for all $i \in [m]$. This gives m identities between q_m and a projection p. However, the same identities could be satisfied by replacing q_m with a projection to $\bot$; to avoid this, we map $\bot$ to a different coordinate of p for each $i \in [m]$.

Formally, let $p\colon [s+1]^{m+1} \to [s+1]$ be the projection of arity $m+1$ to the last coordinate, $p(x_1, \dots, x_{m+1}) = x_{m+1}$. For $i \in [m]$, let $\pi_i\colon U \to [m+1]$ be defined as $\pi_i(I) = m+1$ if $I \in \binom{[m]}{3}$ and $I \ni i$, otherwise set $\pi_i(I) = i$ (in particular $\pi_i(\bot) = i$). Then $q_m \xrightarrow{\pi_i} p$ for each $i \in [m]$.

Consider the bipartite minor condition Σ with two symbols f, g of arity $|U|$ and $m+1$, respectively, and m identities $f \xrightarrow{\pi_i} g$. Clearly this condition is satisfied by q_m, p. We claim the condition is $\frac{4}{m}$-robust, that is, no four identities out of the m identities of Σ can be simultaneously satisfied by projections. Suppose the opposite, that is, assigning $f = p_I$ for some $I \in U$ and $g = p_i$ for some $i \in [m+1]$ satisfies four identities. Without loss of generality these identities are $p_I \xrightarrow{\pi_1} p_i$, $p_I \xrightarrow{\pi_2} p_i$, $p_I \xrightarrow{\pi_3} p_i$, and $p_I \xrightarrow{\pi_4} p_i$. Equivalently, $\pi_1(I) = i$, $\pi_2(I) = i$, $\pi_3(I) = i$, and $\pi_4(I) = i$. The first condition implies that i is either 1 or $m+1$; similarly the second implies that i is either 2 or $m+1$; hence $i = m+1$. The condition $\pi_1(I) = m+1$ then implies that I is a triple in $\binom{[m]}{3}$ containing 1. Similarly I must contain 2, 3, and 4. This is a contradiction, so Σ is indeed $\frac{4}{m}$-robust.

For (iii), consider a partial assignment to some $k < \frac{m}{3}$ coordinates $I_1, \dots, I_k$ of q_m. Let I be the set of values $i \in [m]$ that are contained in some triple among $I_1, \dots, I_k$. Then $|I| \leq 3k < m$, so there is a value $i^* \in [m] \setminus I$. This means no coordinate in S_{i^*} has been assigned yet. Therefore, for any $a \in [s+1]$, assigning the value a to all coordinates in S_{i^*} (and remaining coordinates arbitrarily) makes q_m take the value a. □

As a side note, another way to obtain a projection as a minor of q_m is as follows. Let $T \subseteq U$ be any set intersecting each of $S_1, \dots, S_m, \{\bot\}$. Then identifying all coordinates in T yields a projection to the resulting coordinate; indeed, for any input $\bar{x} \in [s+1]^U$, the smug set of $\bar{x}$ in q_m contains one of $S_1, \dots, S_m, \{\bot\}$ and hence contains a coordinate in T.

Corollary 3.28. *Suppose $k - g + 1 = 3$ and $g \geq 3$. Then the polymorphisms of $(1, g, k)$-SetSAT do not admit a minion homomorphism to a minion of bounded essential arity (or in fact to any minion with functions of arity m having essential arity at most $\frac{m^{1/3}}{4}$).*

Therefore, by Corollary 3.28, the bounded essentially arity assumption of Theorem 2.65 does not apply and thus NP-hardness cannot be derived from Theorem 2.65. By Proposition 3.27 ii, Theorem 2.75 does not apply either and neither does the weaker assumption where constants (bounding essential arity or $1/\epsilon$ in assumptions involving ϵ-robustness) can be replaced by functions subpolynomial in the arity of the polymorphisms (as in [10, Theorem 5.9]).

Even in the very general setting of Theorem 2.76, which is also proved using a layered version of

the PCP theorem, we were unable to prove hardness (specifically in our Corollary 3.17) despite several attempts. On the other hand, we were unable to construct polymorphisms to show that this general assumption fails for SetSAT.

We show that the polymorphisms of SetSAT include an Olšák function, so that Theorem 2.82 cannot be used to show NP-hardness of SetSAT.

Proposition 3.29. *Suppose $\frac{g}{k} > \frac{1}{2}$. There is a polymorphism of $(1, g, k)$-SetSAT with domain size $s+1$ that is an Olšák function.*

Proof. Let us define three sets corresponding to positions of x in the three rows defining an Olšák function: $S_1 = \{1, 2, 6\}, S_2 = \{1, 3, 5\}, S_3 = \{2, 3, 4\}$. Let S_4 be an arbitrary singleton, say $S_4 = \{1\}$. Let $\mathscr{S}$ be the set of supersets of S_1, S_2, S_3, S_4, ordered so that supersets of S_1, S_2, S_3 come earlier. We claim the sequence of sets $\mathscr{S} = S_1, S_2, S_3, S_4$ satisfies the conditions of Definition 3.25. The first condition is trivially satisfied because all sets containing 1 are in $\mathscr{S}$. To check the second condition suppose for contradiction there is a k-tuple of sets in $\mathscr{S}$ that covers every coordinate at most $k - g$ times. For each of these k sets, choose one of the sets S_1, S_2, S_3, S_4 it contains. Let n_1, n_2, n_3, n_4 be the number of times we chose S_1, S_2, S_3, S_4, respectively. Then $n_1 + n_2 + n_3 + n_4 = k$. Since the first coordinate (contained in S_1, S_2, S_4) is covered at most $k - g$ times, we have $n_1 + n_2 + n_4 \leq k - g$. Similarly, other coordinates give us inequalities $n_1 + n_3 \leq k - g$, $n_2 + n_3 \leq k - g$. This implies $k \leq n_1 + 2n_2 + n_3 + n_4 \leq 2(k - g)$ and hence $2g \leq k$. This contradicts $\frac{g}{k} > 2$, so $\mathscr{S}$ satisfies the second condition of Definition 3.25.

We claim that $q_{\mathscr{S}}$ is an Olšák function. Indeed, by definition, $q_{\mathscr{S}}(x, x, y, y, y, x) = x$, for all $x, y \in [s+1]$. Similarly $q_{\mathscr{S}}(x, y, x, y, x, y) = q_{\mathscr{S}}(y, x, x, x, y, y) = x$. □

Theorems 2.84 and 2.86 show that ruling out Siggers or **G**-loop polymorphisms would imply that SetSAT is NP-hard conditional on the NP-hardness of the approximate graph colouring or promise graph homomorphism problems, respectively. However, such polymorphisms of $(1, g, k)$-SetSAT for $\frac{g}{k} > \frac{1}{2}$ are easily constructed as in Proposition 3.29: In detail, for $\mathbf{G} = \mathbf{K}_3$, it suffices to define $f(x_1, x_2, x_3, x_4, x_5, x_6)$ as x_1 if $x_1 = x_3 \ \wedge \ x_2 = x_5 \ \wedge \ x_3 = x_6$ and x_2 otherwise. Thus, even conditional NP-hardness of SetSAT would not follow this way. To the best of our knowledge, SetSAT is the first known NP-hard PCSP that admits **G**-loop polymorphisms.

Table 3.1: Discrepancy d-colouring promise problems; $r \geq 1$, $d \geq 2$, and $0 < a < d$.

Problem	Uniformity	Discrepancy	Weak constraint	Complexity
1	dr	0	non-monochromatic	P
2	$dr + a$	1	non-monochromatic	unknown
3	dr	0	three distinct colours (for $d \geq 3$)	unknown
4	dr	0	two distinct colours at least twice	unknown
5	dr	2	non-monochromatic	unknown

3.8 Discrepancy colourings

SetSAT is closely related to hypergraph colourings, and in this section we show that the hardness part of Theorem 3.3 follows from the conjectured NP-hardness of a natural family of hypergraph colouring problems based on the following notion.

Definition 3.30. A d-colouring of a hypergraph has *discrepancy* Δ if for every hyperedge e we have

$$\max_{c \in [d]}(\#\text{ of occurrences of colour } c \text{ in } e) - \min_{c \in [d]}(\#\text{ of occurrences of colour } c \text{ in } e) \leq \Delta.$$

Table 3.1 gives a list of discrepany colouring promise problems. The promise in each case is that there exists a d-colouring with small discrepancy, and the goal is to find a d-colouring satisfying a weaker constraint.

We first prove some simple results about the complexity of these problems and the connections between them.

Proposition 3.31. *Problem 1 is tractable.*

Proof. We set $s = d - 1$ and give a reduction to $(1, sr, (s+1)r)$-SetSAT with domain size $d = s + 1$, which is tractable by Theorem 3.3.

For each hyperedge $(v_1, \ldots, v_{dr})$ we add the clauses $(N_i(x_1) \vee \ldots \vee N_i(x_{dr}))$ for $1 \leq i \leq d$, where $N_i(x)$ is the literal forbidding $x = i$. Now suppose that the hypergraph admits a d-colouring of discrepancy 0. Then each of the d colours appears exactly r times in each hyperedge, which gives an assignment satisfying exactly $(d-1)r = sr$ literals in each clause of the SetSAT formula, and so the SetSAT formula is sr-satisfiable. Conversely, any satisfying assignment to the SetSAT formula must contain two distinct values, thereby preventing a monochromatic hyperedge. □

Proposition 3.32. *Problem 2 with $a = d - 1$ reduces to Problems 3 and 4.*

Proof. The reduction is the same for both problems. We reduce from an instance H of Problem 2 with uniformity $dr + a = dr + d - 1 = d(r + 1) - 1$. To each hyperedge of H we add a unique new vertex, which brings the uniformity of the new hypergraph H' to $d(r + 1)$. Since H has a d-colouring with discrepancy 1, we can achieve discrepancy 0 in H' by colouring each new variable the appropriate missing colour. Conversely, a d-colouring that leaves at least three distinct colours (or two distinct colours at least twice) in each hyperedge of H' yields only non-monochromatic hyperedges in H. □

Proposition 3.33. *Problem 2 with $a = 1$ reduces to Problem 5.*

Proof. For each hyperedge of the input hypergraph, we take all its subsets of size dr as hyperedges of the new instance. This increases the discrepancy of any colouring by at most 1 and preserves the property of having a monochromatic hyperedge. □

We conjecture that Problem 2 is hard for all choices of parameters.

Conjecture 3.34. *For all $r \geq 1$, $d \geq 2$, and $0 < a < d$, it is NP-hard to distinguish between $(dr + a)$-uniform hypergraphs that have a d-colouring with discrepancy 1 from those where every d-colouring creates a monochromatic hyperedge.*

As evidence for the conjecture, we note that the case $d = 2$ was shown to be NP-hard in [8]. If true, the conjecture and Propositions 3.32 and 3.33 would imply that the tractability of Problem 1 is optimal, in the sense that weakening the promise on either end gives hardness. We now show that our hardness results for SetSAT would also follow from the conjecture.

Theorem (Hardness part of Theorem 3.3 restated)**.** *For set size s, domain size $s + 1$, and $\frac{g}{k} < \frac{s}{s+1}$, $(1, g, k)$-SetSAT is NP-hard.*

Proof. **Case 1:** $k \not\equiv 0 \pmod{s+1}$. Let $d = s+1$ and let r and $0 < a < d$ be such that $k = (s+1)r+a$. We reduce Problem 2 with these parameters to $(1, g' = sr+a-1, k)$-SetSAT, which by Proposition 3.4 reduces to $(1, g, k)$-SetSAT if $g \leq g'$. Suppose that $g > g'$. Then

$$\frac{g}{k} \geq \frac{g' + 1}{k} = \frac{sr + a}{(s + 1)r + a} > \frac{s}{s + 1},$$

contradicting $\frac{g}{k} < \frac{s}{s+1}$, so we conclude that $g \leq g'$.

The reduction itself is straightforward: For each hyperedge $(v_1, \ldots, v_k)$ we create the clauses $(N_i(x_1) \vee \ldots \vee N_i(x_k))$ for $1 \leq i \leq s + 1$. An $(s + 1)$-colouring of discrepancy 1 then satisfies s of

$s+1$ literals in each of the r groups of $s+1$ literals, and at least $a-1$ literals among the remaining a literals, and therefore is a g'-satisfying assignment. Conversely, a 1-satisfying assignment has at least two distinct values in each clause, thus preventing a monochromatic hyperedge.

Case 2: $k \equiv 0 \pmod{s+1}$. We give a reduction from Problem 5, whose NP-hardness would follow from Conjecture 3.34 via Proposition 3.33. Let $d = s+1$, and let r be such that $k = (s+1)r$. We reduce Problem 5 with these parameters to $(1, g' = sr-1, k)$-SetSAT, which by Proposition 3.4 reduces to $(1, g, k)$-SetSAT if $g \leq g'$. Suppose that $g > g'$. Then

$$\frac{g}{k} \geq \frac{g'+1}{k} = \frac{sr}{(s+1)r} = \frac{s}{s+1},$$

contradicting $\frac{g}{k} < \frac{s}{s+1}$, so we conclude that $g \leq g'$.

The reduction is the same as in Case 1. An $(s+1)$-colouring of discrepancy 2 satisfies at least $sr-1$ literals in each clause, and conversely, 1-satisfying assignments prevent monochromatic hyperedges as before. □

Chapter 4

Beyond PCSP(**1-in-3**, **NAE**)

Recall the relations **t-in-k** and **NAE** from Examples 2.8 and 2.9, respectively. In this chapter we investigate PCSPs that arise from the template (**t-in-k**, **NAE**) either by adding tuples to **t-in-k** or removing tuples from **NAE**. This work appears in [23] and is currently being reviewed for publication in *Information and Computation*, having gone through a revision.

4.1 Preliminaries and results

For a set of tuples $\mathbf{S} \subseteq \{0,1\}^k$, we write $\mathbf{t\text{-}in\text{-}k} \cup \mathbf{S}$ for the relational structure whose (only) relation contains all k-tuples of weight t and the tuples from $\mathbf{S}$, and similarly for $\mathbf{NAE} \setminus \mathbf{S}$. By Theorem 2.32, CSP(**NAE**) and CSP(**t-in-k**) are both NP-hard, but combining the two yields a tractable PCSP, as first shown in [19].

Proposition 4.1. *For $k \geq 2$ and $1 \leq t < k$,* PCSP(**t-in-k**, **NAE**) *is tractable.*

Proof. By [19, Claim 4.6], the AT family maps collections of **t-in-k** tuples into **NAE**, so by Theorem 2.59, PCSP(**t-in-k**, **NAE**) is tractable . □

Our first result is an algorithmic dichotomy for templates constructed by adding tuples to **t-in-k**.

Theorem 4.2. *Let $k \geq 3$ and $\emptyset \neq \mathbf{S} \subseteq (\mathbf{t\text{-}in\text{-}k})^c \cap \mathbf{NAE}$. If t is odd, k is even, and $\mathbf{S}$ contains tuples of only odd weight, then* PCSP(**t-in-k** $\cup$ **S**, **NAE**) *is tractable via* AIP. *Otherwise,* PCSP(**t-in-k** $\cup$ **S**, **NAE**) *is not solved by* BLP + AIP.

Barto et al. [10] showed that PCSP(**1-in-3**, **NAE**) is not "finitely tractable", meaning that there is no finite **C** such that **1-in-3** $\to$ **C** $\to$ **NAE** and CSP(**C**) is tractable. In other words, the tractability of PCSP(**1-in-3**, **NAE**) cannot be achieved via a gadget reduction to tractable finite-domain CSPs. This result was then extended by Asimi and Barto [4] to PCSP(**t-in-k**, **NAE**) for $k \geq 3$, $t < k$ when t is even or k is odd. Since the templates proved finitely intractable in [4] are homomorphic relaxations of the BLP + AIP-hard cases in Theorem 4.2, the latter are also finitely intractable.

A recent result of Atserias and Dalmau that gives a necessary condition for PCSPs to be solvable by a local consistency checking algorithm [5] implies that all templates from Theorem 4.2 (and in particular those not solved by BLP + AIP) are *not* solved by a local consistency checking algorithm. By [5, Corollary 4.2], such an algorithm does not solve PCSP(**t-in-k**, **NAE**) for any $k \geq 3$ and $t < k$, and since (**t-in-k**, **NAE**) is a homomorphic relaxation of the templates from Theorem 4.2, our claim follows from [10, Lemma 7.5].

Our second result is a complexity dichotomy for templates obtained by removing tuples from **NAE**.

Theorem 4.3. *Let $k \geq 3$ and $\emptyset \neq \mathbf{S} \subseteq (\mathbf{t\text{-}in\text{-}k})^c \cap \mathbf{NAE}$. If t is odd, k is even, and $\mathbf{S}$ contains tuples of only even weight, then* PCSP(**t-in-k**, **NAE** $\setminus$ **S**) *is tractable. Otherwise,* PCSP(**t-in-k**, **NAE** $\setminus$ **S**) *is NP-hard.*

Theorem 4.3 is an easy consequence of the following.

Theorem 4.4. *Let $k \geq 3$ and let $\mathbf{B} \subseteq \{0,1\}^k$ be a relation such that* **t-in-k** $\to$ **B** *and* CSP(**B**) *is NP-hard. Then* PCSP(**t-in-k**, **B**) *is tractable if and only if* **B** = **NAE**, *unless P=NP.*

In other words, PCSP(**t-in-k**, **B**) is tractable if CSP(**B**) is tractable or **B** = **NAE**, and is NP-hard otherwise.

One ingredient of the proof of Theorem 4.4 is a symmetrisation trick (Proposition 4.6, observed independently in [9]) and the following observation.

Proposition 4.5. *Let R be a symmetric relation on a set A. For any function $f : A \to B$, the component-wise image of R under f, denoted $f(R)$, is a symmetric relation on B.*

Proof. Suppose that $\mathbf{y} \in f(R)$, so $\mathbf{y} = f(\mathbf{x})$ for some $\mathbf{x} \in R$. We must show that $\pi(\mathbf{y}) \in f(R)$ for an arbitrary permutation π. But since R is symmetric, we have $\pi(\mathbf{x}) \in R$, and so $f(\pi(\mathbf{x})) = \pi(\mathbf{y})$ since f is applied component-wise. □

Proposition 4.6. *Let* $(\mathbf{A}, \mathbf{B})$ *be a PCSP template with* $\mathbf{A}$ *symmetric. For each relation* $R \in \mathbf{B}$*, let* R' *be the largest symmetric relation contained in* R*. Let* $\mathbf{B}'$ *be the relational structure with the same domain as* $\mathbf{B}$ *but with relations* R' *instead of* R*. Then* $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ *is polynomial-time equivalent to* $\mathrm{PCSP}(\mathbf{A}, \mathbf{B}')$.

Proof. We first check that $(\mathbf{A}, \mathbf{B}')$ is a valid PCSP template, i.e., that there is a homomorphism $\mathbf{A} \to \mathbf{B}'$. Let ϕ be a homomorphism from $\mathbf{A}$ to $\mathbf{B}$. By Proposition 4.5, $\phi(\mathbf{A})$ is symmetric, and since $\mathbf{B}'$ is the largest symmetric relational structure contained in $\mathbf{B}$, we have $\phi(\mathbf{A}) \subseteq \mathbf{B}'$. Therefore $(\mathbf{A}, \mathbf{B}')$ is a valid PCSP template.

For a function $f : A^m \to B$ and a relation $R \in \mathbf{A}$, let $f(R)$ be the image of R under f, that is, we apply f componentwise to every m-tuple of elements of R, as when applying a polymorphism. Let $f(\mathbf{A})$ be the structure with relations $f(R)$ for $R \in \mathbf{A}$. Since $\mathbf{A}$ is symmetric, so is $f(\mathbf{A})$, and if $f \in \mathrm{Pol}(\mathbf{A}, \mathbf{B})$, then $f(A) \subseteq \mathbf{B}$. Therefore $f(\mathbf{A}) \subseteq \mathbf{B}'$ and $\mathrm{Pol}(\mathbf{A}, \mathbf{B}) \subseteq \mathrm{Pol}(\mathbf{A}, \mathbf{B}')$. The reverse inclusion follows from $\mathbf{B}' \subseteq \mathbf{B}$ and gives $\mathrm{Pol}(\mathbf{A}, \mathbf{B}) = \mathrm{Pol}(\mathbf{A}, \mathbf{B}')$, which implies by Theorem 2.56 that $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ and $\mathrm{PCSP}(\mathbf{A}, \mathbf{B}')$ are log-space equivalent and therefore also polynomial-time equivalent. □

The tractability parts of Theorems 4.2 and 4.3 follow easily from existing work, as we now show.

Proposition 4.7. *Let* $\textbf{odd-}\boldsymbol{in}\textbf{-k} = \{x \in \{0,1\}^k : |\{i : x_i = 1\}| \equiv 1 \pmod 2\}$*. Then for* k *even, (the search version of)* $\mathrm{CSP}(\textbf{odd-}\boldsymbol{in}\textbf{-k})$ *is tractable.*

Proof. We claim that $\mathrm{XOR}_3 \in \mathrm{Pol}(\textbf{odd-in-k})$ so that tractability will follow from Theorem 2.32. Suppose that XOR_3 returns a tuple of even weight d. Then in the $k \times 3$ matrix of inputs with three odd weight tuples as columns, there are d rows with an odd number of 1's and $k - d$ rows with an even number of 1's. Together these give an even total number of 1's in the matrix. But since the three input columns have odd weight, the total number of 1's in the matrix is odd. Contradiction. □

Proof of the tractability part of Theorems 4.2 and 4.3. Under the tractability criterion of Theorem 4.2, $\textbf{t-in-k} \cup \mathbf{S} \subseteq \textbf{odd-in-k} \subseteq \mathbf{NAE}$, so $(\textbf{t-in-k} \cup \mathbf{S}, \mathbf{NAE})$ is a homomorphic relaxation of the template $(\textbf{odd-in-k}, \textbf{odd-in-k})$. By Theorem 2.56, this implies that

$$\mathrm{PCSP}(\textbf{t-in-k} \cup \mathbf{S}, \mathbf{NAE}) \leq_p \mathrm{PCSP}(\textbf{odd-in-k}, \textbf{odd-in-k}) = \mathrm{CSP}(\textbf{odd-in-k}),$$

where $\mathrm{CSP}(\textbf{odd-in-k})$ is tractable by Proposition 4.7 (for even k). Similarly, under the tractability criterion of Theorem 4.3, we have $\textbf{t-in-k} \subseteq \textbf{odd-in-k} \subseteq \mathbf{NAE} \backslash \mathbf{S}$ and thus $\mathrm{PCSP}(\textbf{t-in-k}, \mathbf{NAE} \backslash \mathbf{S}) \leq_p$

CSP(**odd-in-k**). By composing XOR_3 functions from the proof of Proposition 4.7, or by observing that **odd-in-k** is an affine subspace, we have XOR $\subseteq$ Pol(**odd-in-k**), which implies via the inclusion that our PCSP templates have the XOR family of polymorphisms and thus are solvable by AIP. □

In the next section we rule out the applicability of BLP + AIP as stated in Theorem 4.2, then in Section 4.3 we discuss some obstacles to achieving NP-hardness, and finally in Section 4.4 we prove the hardness part of Theorem 4.3.

4.2 Adding tuples

The following result implies, by Theorem 2.60, the non-tractability part of Theorem 4.2.

Theorem 4.8. *Let $k \geq 3$, $1 \leq t < k$, and $\mathbf{x}$ be a k-tuple of weight $1 \leq d < k$ with $d \neq t$. Then, (**t-in-k** $\cup \{\mathbf{x}\}$, **NAE**) does not have 2-block-symmetric polymorphisms of all odd arities, unless t is odd, k is even, and d is odd.*

The implication is as follows: In the non-tractability case of Theorem 4.2, $\mathbf{S}$ contains a tuple $\mathbf{x}$ of weight d such that if d is odd, then t is even, k is odd, or both. Therefore Pol(**t-in-k** $\cup\, \mathbf{S}$, **NAE**) $\subseteq$ Pol(**t-in-k** $\cup \{\mathbf{x}\}$, **NAE**), so it suffices to rule out 2-block-symmetric polymorphisms for templates of the form (**t-in-k** $\cup \{\mathbf{x}\}$, **NAE**).

We start with two simple observations which reduce the number of cases to deal with. Since permuting the rows of a matrix of inputs to a polymorphism permutes the values of the output tuple and does not affect membership in the symmetric **NAE** relation, we have the following.

Observation 4.9. *Let $\mathbf{x}$ and $\mathbf{y}$ be two k-tuples of weight d. Then,* Pol(**t-in-k** $\cup \{\mathbf{x}\}$, **NAE**) = Pol(**t-in-k** $\cup \{\mathbf{y}\}$, **NAE**).

By Observation 4.9, it suffices to prove Theorem 4.8 for $\mathbf{x}$ of the form $\mathbf{x} = 1^d 0^{k-d}$.

Observation 4.10. *There is a bijection between* Pol(**t-in-k** $\cup \{\mathbf{x}\}$, **NAE**) *and* Pol((**k** − **t**)**-in-k** $\cup$ $\{\overline{\mathbf{x}}\}$, **NAE**) *given by $f(x_1, \ldots, x_m) \mapsto f(1 - x_1, \ldots, 1 - x_m)$, where $\overline{\mathbf{x}}$ is the negation of $\mathbf{x}$.*

Observation 4.10 implies that 2-block-symmetry of polymorphisms is preserved when swapping 0's and 1's.

There are eight combinations of the parities of k, t, and d. The case $(k, t, d) \equiv (0, 1, 1)$ is out of the scope of Theorem 4.8 and is covered by the tractable case of Theorem 4.2. The case $(k, t, d) \equiv (1, 0, 0)$

is covered for $d > t$ in Proposition 4.13 and $d < t$ in Proposition 4.14, and all other cases are covered for $d > t$ in Proposition 4.11 and $d < t$ in Proposition 4.12. By applying Observation 4.10, we may assume that $d + t \leq k$, which allows a single construction to work in each of these propositions. We start with a brief account of the idea behind the proofs.

Let C_k^t be the $k \times k$ matrix containing the k cyclic shifts of the column $1^t 0^{k-t}$. The matrix C_k^t can be used to fill one of the coordinate blocks of a 2-block-symmetric function f of arity $2k \pm 1$. For example, suppose that C_k^t is used to fill the "first" coordinate block. It does not matter whether the first block contains the odd or even coordinates. Then f depends only on the weights in each row of the "second" block, since the first block has the same weight in every row. This allows f to be analysed as a symmetric (1-block-symmetric) function.

For each k, t, and d, and for any f such that one of its blocks can be filled by C_k^t, we exhibit a set of tableaux for the other block that prevents f from being a polymorphism. Suppose we have filled one block with C_k^t, so that f can now be represented as a unary function of the weight on its other block. For any weights w_1, w_2, w_3, we have at least one of $f(w_1) = f(w_2)$, $f(w_1) = f(w_3)$, and $f(w_2) = f(w_3)$. Therefore by constructing three tableaux corresponding to these three pairs of weights, we are guaranteed that f will return an all-equal tuple and hence not be a polymorphism.

Proposition 4.11. *Let $k \geq 3$, $1 \leq t < k$, and $t < d < k$ be such that $d + t \leq k$ and $t \equiv k$ or $d \not\equiv t \pmod 2$. Let $\mathbf{x}$ be a tuple of weight d. Then* $\mathrm{Pol}(\mathbf{t\text{-}in\text{-}k} \cup \{\mathbf{x}\}, \mathbf{NAE})$ *does not have 2-block-symmetric functions of arity $2k - 1$.*

Proof. Let f be a 2-block-symmetric function of arity $2k - 1$. We will show that f is not a polymorphism of $(\mathbf{t\text{-}in\text{-}k} \cup \{\mathbf{x}\}, \mathbf{NAE})$. Let the odd block contain the tableau C_k^t and denote by C_k^{t-} the tableau obtained from C_k^t by removing its last column. We describe how to construct the even block with $k - 1$ columns in the cases below. We use $t - 1$, t, and $t + 1$ as our three weights.

Case 1: weights $t - 1$ and t.

The even tableau is C_k^{t-}. Thus each row in the even block has weight either $t-1$ or t. If $f(t-1) = f(t)$ then f returns an all-equal tuple 0^k or 1^k, so $f \notin \mathrm{Pol}(\mathbf{t\text{-}in\text{-}k} \cup \{\mathbf{x}\}, \mathbf{NAE})$.

Case 2: weights $t - 1$ and $t + 1$.

We take C_k^{t-} and replace some of the **t-in-k** tuples with $\mathbf{x}$ as necessary.

Case 2a: $t \equiv k \pmod 2$.

The tableau C_k^{t-} has an even number $k - t$ of rows with weight t. These can be paired up and

1's exchanged so that each row has weight either $t-1$ or $t+1$. In particular, in the columns $t+1, t+3, \ldots, k-1$, we swap the values in the pairs of rows $(t, t+1), (t+2, t+3), \ldots, (k-2, k-1)$, respectively. An illustration is given in Figure (4.1a).

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & \mathbf{0} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & \mathbf{0} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & \mathbf{1} \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & \mathbf{0} \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

(a) Case 2a with $t = 3$.

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & \mathbf{0} & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & \mathbf{1} \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & \mathbf{0} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

(b) Case 2b with $t = 2$ and $d = 5$.

Figure 4.1: Example with $k = 9$ for $f(t-1) = f(t+1)$. Swapped values in bold.

Case 2b: $(k, t, d) \equiv (0, 1, 0)$ or $(1, 0, 1) \pmod 2$.

The tableau C_k^{t-} has an odd number $k-t$ of rows with weight t. We replace the first column with $\mathbf{x}$. This adds $d-t$ 1's to the first column, and the $d-t$ rows with the added 1's now have weight $t+1$. There remains an even number $k-d$ of rows of weight t, which can be paired up to exchange 1's and achieve weight $t-1$ or $t+1$ in each row. In particular, we swap the values at positions $(t, 2)$ and $(d+1, 2)$, and then in the columns $d+3, d+5, \ldots, k-1$, we swap the values in the pairs of rows $(d+2, d+3), (d+4, d+5), \ldots, (k-2, k-1)$, respectively. An illustration is given in Figure (4.1b).

Cases 2a and 2b cover all possible parities of k, t, and d under the proposition's assumptions. In both cases, each row in the even block has weight either $t-1$ or $t+1$. If $f(t-1) = f(t+1)$ then f returns an all-equal tuple, so $f \notin \mathrm{Pol}(\textbf{t-in-k} \cup \{\mathbf{x}\}, \textbf{NAE})$.

Case 3: weights t and $t+1$.

We first give a general description of the tableau, and then derive values for its parameters. We place the tuple $\mathbf{x}$ in the first r columns and fill the remaining $k-1-r$ columns with **t-in-k** tuples that have specific behaviours in the upper d rows and lower $k-d$ rows. Our goal is to distribute the weight of the **t-in-k** tuples between these two blocks of rows so that every row in the full tableau has weight t or $t+1$.

Denoting by a the average column weight within the upper group of rows, we place either b or $b+1$ 1's from each **t-in-k** tuple in the upper block, where b is an integer close to a. More precisely, to fill the upper d rows, we use $0 \le s \le k-1-r$ columns of weight b and $k-1-r-s$ columns

of weight $b+1$, and in the lower block of rows, we use s columns of weight $t-b$ and $k-1-r-s$ columns of weight $t-(b+1)$, respectively, so that the full columns are **t-in-k** tuples.

We now describe the position of the 1's in the upper group of rows; the construction for the lower group is analogous. We fill columns with 1's from top to bottom, starting at the left-most column, and moving to the right after placing a column's quota of 1's (either b or $b+1$). The order of the weight b and $b+1$ tuples does not matter. When moving right to the next column, we continue placing 1's in the row immediately below the lowest row containing a 1 in the previous column. Once we reach the bottom of the group of rows, we wrap around to the top and continue in this way.

A $k \times (k-1)$ tableau containing only **t-in-k** tuples needs at least t more 1's to achieve weight $\geq t$ in each row. Each time we replace a **t-in-k** tuple with $\mathbf{x}$, the tableau gains $d-t$ 1's, and therefore at least $r = \left\lceil \frac{t}{d-t} \right\rceil$ occurrences of $\mathbf{x}$ are necessary. This turns out to be sufficient. The remainder of the proof is devoted to showing that there exist a, b, and s which allow our construction to work.

A crucial observation connecting the column and row weights in our tableau is that within each block of rows, the weight between rows varies by at most one. It therefore suffices for the *average* row weight in each block to be between t and $t+1$.

To achieve average row weight at least t, a must be such that the total weights in the upper and lower blocks are at least $d(t-r)$ and $t(k-d)$, respectively. This is guaranteed when both $a(k-1-r) \geq d(t-r)$ and $(t-a)(k-1-r) \geq t(k-d)$, or equivalently,

$$\frac{d(t-r)}{k-1-r} \;\leq\; a \;\leq\; \frac{t(d-r-1)}{k-1-r}. \tag{4.1}$$

To achieve average row weight at most $t+1$, a must be such that the total weights in the upper and lower blocks are at most $d(t+1-r)$ and $(k-d)(t+1)$, respectively. This is guaranteed when both $a(k-1-r) \leq d(t+1-r)$ and $(t-a)(k-1-r) \leq (k-d)(t+1)$, or equivalently,

$$\frac{t(d-r-1)-k+d}{k-1-r} \;\leq\; a \;\leq\; \frac{d(t-r+1)}{k-1-r}. \tag{4.2}$$

Finally, to ensure that each block of rows is tall enough to accommodate the 1's specified in the construction, it suffices that $0 \leq a \leq d$ and $0 \leq t-a \leq k-d$, which together are equivalent to $\max(0, d+t-k) \leq a \leq \min(d,t)$. By our assumptions, this reduces to $0 \leq a \leq t$.

We now show that these inequalities can all be simultaneously satisfied. In 4.1, the upper bound is at least the lower bound if and only if $r \geq \frac{t}{d-t}$, which holds for our choice of r, and in 4.2, the upper

bound is at least the lower bound if and only if $r \leq \frac{t}{d-t} + \frac{k}{d-t}$, which holds since $\frac{k}{d-t} \geq 1$. Exchanging the upper/lower bound pairs in 4.1 and 4.2 results in two pairs of inequalities on a that are always satisfied. Finally, for $0 \leq a \leq t$, it suffices that at least one of the lower bounds is nonnegative, and at least one of the upper bounds is at most t. The lower bound in 4.1 is nonnegative if and only if $t \geq r$, which always holds, and the upper bound is at most t if and only if $d \leq k$, which also always holds.

Therefore there exists $0 \leq a \leq t$ satisfying 4.1 and 4.2, and since these inequalities are not strict, we can take a to be rational with denominator $k-1-r$. Let $b = \lfloor a \rfloor$. Recalling that s is the number of columns of weight b in the upper block, computing the total weight in the upper block gives $sb + (k-1-r-s)(b+1) = a(k-1-r)$, so that $s = (k-1-r)(b+1-a)$. This is an integer since a is a fraction with denominator $k-1-r$. As a sanity check, note that if $a = b$ or $a = b+1$, then $s = k-1-r$ or $s = 0$, respectively.

We have shown that there exist a, b, and s which permit us to construct the tableau with weight t or $t+1$ in each row. Thus if $f(t) = f(t+1)$, then f returns the all-equal tuple 0^k or 1^k, so $f \notin \mathrm{Pol}(\textbf{t-in-k} \cup \{\mathbf{x}\}, \textbf{NAE})$. This ends the proof of Case 3.

Since we must have at least one of $f(t-1) = f(t)$, $f(t-1) = f(t+1)$, and $f(t) = f(t+1)$, the three cases complete the proof.

An example with $t = 7$, $k = 15$, and $d = 10$ is illustrated in Figure 4.2. In this case, we have $r = \left\lceil \frac{t}{d-t} \right\rceil = 3$ and we get the inequalities $\frac{40}{11} \leq a \leq \frac{42}{11}$ and $\frac{37}{11} \leq a \leq \frac{50}{11}$. We take $a = \frac{41}{11}$; the values $\frac{40}{11}$ and $\frac{42}{11}$ would also work. Then $b = 3$ and $s = 3$, so in the upper group we have 3 columns of weight $b = 3$ and 8 columns of weight $b+1 = 4$. The columns containing $\mathbf{x}$ are shown in addition to the construction on $k-1-r$ columns.

□

Proposition 4.12. *Let $k \geq 3$, $1 < t < k$, and $1 \leq d < t$ be such that $d + t \leq k$ and $t \equiv k$ or $d \not\equiv t \pmod 2$. Let $\mathbf{x}$ be a tuple of weight d. Then* $\mathrm{Pol}(\textbf{t-in-k} \cup \{\mathbf{x}\}, \textbf{NAE})$ *does not have 2-block-symmetric functions of arity $2k+1$.*

Proof. The proof is similar to the case $d > t$ established in Proposition 4.11, except that now we can reduce the number of 1's in the tableaux by replacing **t-in-k** tuples with $\mathbf{x}$. We place the tableau C_k^t in the even coordinates, so that there are $k+1$ columns to be filled in the odd coordinates. As before, we give tableaux for the three pairs of weights from $t-1$, t, and $t+1$.

Let C_k^{t+} be the $k \times (k+1)$ matrix C_k^t with an extra column $1^t 0^{k-t}$.

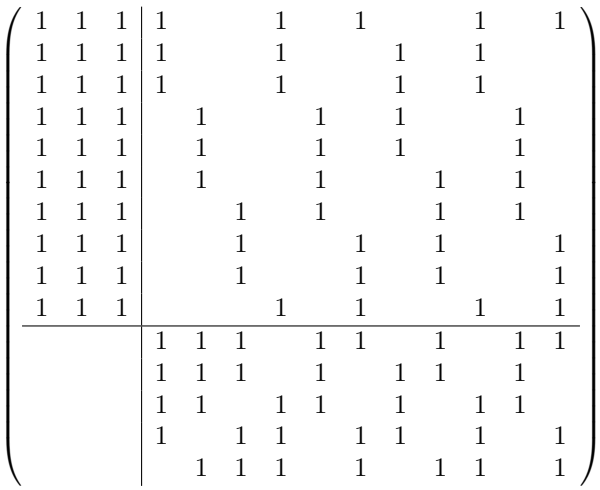

Figure 4.2: Example with $t = 7$, $k = 15$, and $d = 10$ from Case 3, for $f(t) = f(t+1)$.

Case 1: weights t and $t+1$.

The odd tableau is C_k^{t+}, so each row in the odd block has weight either t or $t+1$.

Case 2: weights $t-1$ and $t+1$.

The tableaux are similar to Case 2 in the proof of Proposition 4.11. When $t \equiv k$, we modify C_k^{t+} in the columns $t+2, t+4, \ldots, k$ by swapping the values in the pairs of rows $(t+1, t+2), (t+3, t+4), \ldots, (k-1, k)$, respectively. When $(k, t, d) \equiv (0, 1, 0)$ or $(1, 0, 1)$, we replace the first column of C_k^{t+} with $\mathbf{x}$, which leaves an even number $k-d$ of rows of weight t. Therefore in columns $d+2, d+4, \ldots, k$ we swap the values in the pairs of rows $(d+1, d+2), (d+3, d+4), \ldots, (k-1, k)$, respectively, to get weight $t-1$ or $t+1$ in each row.

Case 3: weights $t-1$ and t.

This case is similar to Case 3 in the proof of Proposition 4.11. The tableau C_k^{t+} has t rows with weight $t+1$ and $k-t$ rows with weight t, so we must reduce the total weight by at least t. Replacing a **t-in-k** tuple with $\mathbf{x}$ reduces the weight by $t-d$, which suggests $r = \left\lceil \frac{t}{t-d} \right\rceil$ such replacements.

Let a be the average column weight in the upper d rows. To achieve average row weight at most t, a must be such that the total weights in the upper and lower blocks are at most $d(t-r)$ and $t(k-d)$, respectively. This is guaranteed when both $a(k+1-r) \leq d(t-r)$ and $(t-a)(k+1-r) \leq t(k-d)$, or equivalently,

$$\frac{t(d-r+1)}{k+1-r} \leq a \leq \frac{d(t-r)}{k+1-r}. \tag{4.3}$$

To achieve average row weight at least $t-1$, a must be such that the total weights in the upper and lower blocks are at least $d(t-1-r)$ and $(k-d)(t-1)$, respectively. This is guaranteed when both $a(k+1-r) \geq d(t-1-r)$ and $(t-a)(k+1-r) \geq (k-d)(t-1)$, or equivalently,

$$\frac{d(t-1-r)}{k+1-r} \;\leq\; a \;\leq\; \frac{t(d-r+1)+k-d}{k+1-r}. \tag{4.4}$$

Finally, to ensure that each block of rows is tall enough to accommodate the 1's specified by the construction, it suffices that $0 \leq a \leq d$ and $0 \leq t-a \leq k-d$, which together are equivalent to $\max(0, d+t-k) \leq a \leq \min(d,t)$. By our assumptions, this reduces to $0 \leq a \leq d$.

We now show that these inequalities can all be simultaneously satisfied. In 4.3, the upper bound is at least the lower bound if and only if $r \geq \frac{t}{t-d}$, which holds for our choice of r. In 4.4, the upper bound is at least the lower bound if and only if $r \leq \frac{t}{t-d} + \frac{k}{t-d}$, which holds since $\frac{k}{t-d} \geq 1$. Exchanging the upper/lower bound pairs in 4.3 and 4.4 results in two pairs of inequalities on a that are always satisfied. Finally, for $0 \leq a \leq d$, it suffices that at least one of the lower bounds is nonnegative, and at least one of the upper bounds is at most d. The lower bound in 4.3 is nonnegative if and only if $t \geq d+1$, and the upper bound is at most d if and only if $t \leq k+1$, both of which always hold. The rest of the proof follows the same reasoning as in Proposition 4.11. □

Proposition 4.13. *Let $k \geq 3$, $1 \leq t < k$, and $t < d < k$ be such that $d+t \leq k$ and $(k,t,d) \equiv (1,0,0) \pmod 2$. Let $\mathbf{x}$ be a tuple of weight d. Then* Pol(**t-in-k**$\cup\{\mathbf{x}\}$, **NAE**) *does not have 2-block-symmetric functions of all odd arities.*

Proof. The parities of k, t, and d prevent us from using the weights $t-1$ and $t+1$ in Case 2, which necessitates a different choice of weights and a slightly more complicated construction for Case 3. We take $L \geq 1$ copies of C_k^t in the even block of the tableau, and leave $Lk+1$ columns to be filled in the odd block, for a total arity of $2Lk+1$. The three weights we use are Lt, $Lt+1$, and $Lt+2$, with L determined later.

Case 1: weights Lt and $Lt+1$.

We take $L-1$ copies of C_k^t and one copy of C_k^{t+}, so that each row has weight Lt or $Lt+1$.

Case 2: weights Lt and $Lt+2$.

We take $L-1$ copies of C_k^t and one copy of C_k^{t+}, leaving t rows of weight $Lt+1$, and since t is even, these rows can be paired and values swapped so that each row has weight Lt or $Lt+2$. In particular, in columns $2, 4, \ldots, t$, we swap the values in the pairs of rows $(1,2), (3,4), \ldots, (t-1, t)$, respectively.

Case 3: weights $Lt+1$ and $Lt+2$.

Let $r = \left\lceil \frac{k-t}{d-t} \right\rceil$ and let a be the average column weight in the upper d rows. To achieve average row weight at least $Lt+1$, a must be such that the total weights in the upper and lower blocks are at least $d(Lt+1-r)$ and $(Lt+1)(k-d)$, respectively. This is guaranteed when both $a(Lk+1-r) \geq d(Lt+1-r)$ and $(t-a)(Lk+1-r) \geq (Lt+1)(k-d)$, or equivalently,

$$\frac{d(Lt+1-r)}{Lk+1-r} \;\leq\; a \;\leq\; \frac{t(Ld+1-r)-k+d}{Lk+1-r}. \tag{4.5}$$

To achieve average row weight at most $Lt+2$, a must be such that the total weights in the upper and lower blocks are at most $d(Lt+2-r)$ and $(k-d)(Lt+2)$, respectively. This is guaranteed when both $a(Lk+1-r) \leq d(Lt+2-r)$ and $(t-a)(Lk+1-r) \leq (k-d)(Lt+2)$, or equivalently,

$$\frac{t(Ld+1-r)-2(k+d)}{Lk+1-r} \;\leq\; a \;\leq\; \frac{d(Lt+2-r)}{Lk+1-r}. \tag{4.6}$$

Finally, to ensure that each block of rows is tall enough to accommodate the 1's specified in the construction, it suffices that $0 \leq a \leq d$ and $0 \leq t-a \leq k-d$, which together are equivalent to $\max(0, d+t-k) \leq a \leq \min(d,t)$. By our assumptions, this reduces to $0 \leq a \leq t$.

We now show that these inequalities can all be simultaneously satisfied. In 4.5, the upper bound is at least the lower bound if and only if $r \geq \frac{k-t}{d-t}$, which holds for our choice of r, and in 4.6, the upper bound is at least the lower bound if and only if $r \leq \frac{k-t}{d-t} + \frac{k}{d-t}$, which holds since $\frac{k}{d-t} \geq 1$. Exchanging the upper/lower bound pairs in 4.5 and 4.6 results in two pairs of inequalities on a that are always satisfied. Finally, for $0 \leq a \leq t$, it suffices that at least one of the lower bounds is nonnegative, and at least one of the upper bounds is at most t. The lower bound in 4.5 is nonnegative if and only if $L \geq \frac{r-1}{t}$, and the upper bound is at most t if and only if $L \geq -\frac{1}{t}$, so it suffices to take $L \geq \frac{r-1}{t}$. The rest of the proof follows the same reasoning as in Proposition 4.11. □

Proposition 4.14. *Let $k \geq 3$, $1 < t < k$, and $1 \leq d < t$ be such that $t+d \leq k$ and $(k,t,d) \equiv (1,0,0) \pmod 2$. Let $\mathbf{x}$ be a tuple of weight d. Then* $\mathrm{Pol}(\mathbf{t\text{-}in\text{-}k} \cup \{\mathbf{x}\}, \mathbf{NAE})$ *does not have 2-block-symmetric functions of all odd arities.*

Proof. We place $L \geq 1$ copies of C_k^t in the odd block of our tableau, leaving $Lk-1$ columns to be filled in the even block for a total arity of $2Lk-1$. The three weights used are Lt, $Lt-1$, and $Lt-2$, with L determined later.

Case 1: weights Lt and $Lt-1$.

We take $L-1$ copies of C_k^t and one copy of C_k^{t-}, so that each row has weight Lt or $Lt-1$.

Case 2: weights Lt and $Lt-2$.

We take $L-1$ copies of C_k^t and one copy of C_k^{t-}, leaving t rows of weight $Lt-1$, and since t is even, these rows can be paired and values swapped so that each row has weight Lt or $Lt-2$. In particular, in columns $2, 4, \ldots, t-2$, we swap the values in the pairs of rows $(1,2), (3,4), \ldots, (t-3, t-2)$, respectively. Finally, in column $k-1$, we swap the values in rows k and $t-1$.

Case 3: weights $Lt-1$ and $Lt-2$.

Let $r = \left\lceil \frac{k-t}{t-d} \right\rceil$ and let a be the average column weight in the upper d rows. To achieve average row weight at most $Lt-1$, a must be such that the total weights in the upper and lower blocks are at most $d(Lt-1-r)$ and $(Lt-1)(k-d)$, respectively. This is guaranteed when both $a(Lk-1-r) \leq d(Lt-1-r)$ and $(t-a)(Lk-1-r) \leq (Lt-1)(k-d)$, or equivalently,

$$\frac{t(Ld-1-r)+k-d}{Lk-1-r} \;\leq\; a \;\leq\; \frac{d(Lt-1-r)}{Lk-1-r}. \tag{4.7}$$

To achieve average row weight at least $Lt-2$, a must be such that the total weights in the upper and lower blocks are at least $d(Lt-2-r)$ and $(k-d)(Lt-2)$, respectively. This is guaranteed when both $a(Lk-1-r) \geq d(Lt-2-r)$ and $(t-a)(Lk-1-r) \geq (k-d)(Lt-2)$, or equivalently,

$$\frac{d(Lt-2-r)}{Lk-1-r} \;\leq\; a \;\leq\; \frac{t(Ld-1-r)+2(k-d)}{Lk-1-r}. \tag{4.8}$$

Finally, to ensure that each block of rows is tall enough to accommodate the 1's specified in the construction, it suffices that $0 \leq a \leq d$ and $0 \leq t-a \leq k-d$, which together are equivalent to $\max(0, d+t-k) \leq a \leq \min(d, t)$. By our assumptions, this reduces to $0 \leq a \leq d$.

We now show that these inequalities can all be simultaneously satisfied. In 4.7, the upper bound is at least the lower bound if and only if $r \geq \frac{k-t}{t-d}$, which holds for our choice of r, and in 4.8, the upper bound is at least the lower bound if and only if $r \leq \frac{k-t}{t-d} + \frac{k}{t-d}$, which holds since $\frac{k}{t-d} \geq 1$. Exchanging the upper/lower bound pairs in 4.7 and 4.8 results in two pairs of inequalities on a that are always satisfied. Finally, for $0 \leq a \leq d$, it suffices that at least one of the lower bounds is nonnegative, and at least one of the upper bounds is at most d. The lower bound in 4.8 is nonnegative if and only if $L \geq \frac{r+2}{t}$, and the upper bound in 4.7 is at most d if and only if $t \leq k$, so it suffices to take $L \geq \frac{r+2}{t}$. The rest of the proof follows the same reasoning as in Proposition 4.11. □

4.3 Obstacles to proving NP-hardness

The ultimate goal for PCSP($\mathbf{t\text{-}in\text{-}k} \cup \mathbf{S}, \mathbf{NAE}$) is to prove NP-hardness in Theorem 4.2 instead of non-solvability by BLP+AIP. Here we exhibit choices of t, k, and $\mathbf{S}$ such that Pol($\mathbf{t\text{-}in\text{-}k} \cup \mathbf{S}, \mathbf{NAE}$) has unbounded essential arity, Olšák functions, or Siggers functions, thus showing that Theorems 2.65, 2.82, and 2.84 are insufficient for hardness in general.

Proposition 4.15. *Let $k \geq 4$ and let $\mathbf{x}$ be a k-tuple of weight $2 \leq d \leq k-2$. Then* Pol($\mathbf{1\text{-}in\text{-}k} \cup \{\mathbf{x}\}, \mathbf{NAE}$) *does not have bounded essential arity.*

Proof. We exhibit a family of functions in Pol($\mathbf{1\text{-}in\text{-}k} \cup \{\mathbf{x}\}, \mathbf{NAE}$) with the desired properties. Let $q \in (\frac{1}{k-d}, 1)$ and let m be such that $mq \notin \mathbb{Z}$. Define

$$f(x_1, \ldots, x_m) = \mathrm{OR}(x_1, \mathrm{THR}_q(x_1, \ldots, x_m)).$$

To see that $f \in$ Pol($\mathbf{1\text{-}in\text{-}k} \cup \{\mathbf{x}\}, \mathbf{NAE}$), note that the output tuple will always contain at least one 1 since the first column of the input does, so it suffices to check that we never return the tuple 1^k. There exist at least $k-d$ rows that do not have a 1 in their first coordinate and that share at most $m-1$ 1's. Therefore, one of these rows must have at most $\lfloor \frac{m-1}{k-d} \rfloor$ 1's, and since $\lfloor \frac{m-1}{k-d} \rfloor \leq \lfloor \frac{m}{k-d} \rfloor \leq \frac{m}{k-d} < mq$, we conclude that f returns 0 on this row.

It is easy to see that f does not have bounded essential arity: If we set the first $\lceil (1-q)m \rceil$ coordinates to 0 and the remaining $\lfloor qm \rfloor$ cooridnates to 1, then flipping any of the 0's to a 1 will flip the output of f from 0 to 1, and the number of coordinates that can be flipped in this way is linear in m. □

Proposition 4.16. *Let* $\mathbf{x} = 11000$. *Then* Pol($\mathbf{1\text{-}in\text{-}5} \cup \{\mathbf{x}\}, \mathbf{NAE}$) *contains both Olšák and Siggers functions.*

Proof. It suffices to show that Pol($\mathbf{1\text{-}in\text{-}5} \cup \{\mathbf{x}\}, \mathbf{NAE}$) contains a symmetric function of arity 6, since any symmetric function satisfies both the Olšák and Siggers identities. We claim that the $\frac{1}{4}$-threshold function f of arity 6 is in Pol($\mathbf{1\text{-}in\text{-}5} \cup \{\mathbf{x}\}, \mathbf{NAE}$).

Let M be a 5×6 matrix whose columns are $\mathbf{1\text{-}in\text{-}5} \cup \{\mathbf{x}\}$ tuples. If $f(M) = 1^5$, then each row of M must contain at least two 1's, for a total of at least 10 1's in M. This forces at least four columns of M to contain $\mathbf{x}$, which in turn forces $f(M) = \mathbf{x}$, a contradiction. If $f(M) = 0^5$, then there is at

most one 1 in each row, for a total of at most five 1's in M. But each column contains at least one 1, so M has in total at least six 1's, a contradiction. □

4.4 Removing tuples

In this section we prove Theorem 4.4 and show how it implies Theorem 4.3.

Schaefer's dichotomy theorem (Theorem 2.32) allows us to obtain a simple description of all $\mathbf{B}$ with CSP($\mathbf{B}$) tractable and **t-in-k** $\to \mathbf{B}$.

Proposition 4.17. *Let $k \geq 3$, $1 \leq t < k$, and suppose that* **t-in-k** $\to \mathbf{B}$. *Then* CSP($\mathbf{B}$) *is tractable if and only if*

1. $0^k \in \mathbf{B}$ *or* $1^k \in \mathbf{B}$, *or*
2. *t is odd, k is even, and* $\mathbf{B} =$ **odd-*in*-k**.

Observe that Proposition 4.17 directly implies the NP-hardness of CSP(**t-in-k**).

Proof. In Case 1, Pol($\mathbf{B}$) contains a constant function so CSP($\mathbf{B}$) is tractable by Theorem 2.32, and in Case 2, tractability is given by Proposition 4.7.

We now turn to hardness. For the rest of the proof, assume that neither (1) nor (2) of the proposition statement applies.

Suppose that **t-in-k** $\to \mathbf{B}$ by the function $\phi : \{0,1\} \to \{0,1\}$. If ϕ is constant, then either $\mathbf{B} = \{0^k\}$ or $\mathbf{B} = \{1^k\}$ and we are in case (1), a contradiction. If $\phi(x) = 1 - x$, note that $\phi($**t-in-k**$)$ satisfies conditions (1) and (2) precisely when **t-in-k** does, so it suffices to consider only the case where ϕ is the identity and **t-in-k** $\subseteq \mathbf{B}$.

We show that Pol($\mathbf{B}$) contains none of the functions AND_2, OR_2, MAJ_3, and XOR_3 from Theorem 2.32. To accomplish this, we assume that one of these functions f is present in Pol($\mathbf{B}$), and then show that repeated application of f to a certain set of tuples leads to (1) or (2) of the proposition, a contradiction. Recall that we denote by $f(R)$ the image of the relation R under f. We make seven claims below, which together exclude the four functions as polymorphisms: Case (i) covers AND_2, case (ii) covers OR_2, (overlapping) cases (iii) and (iv) cover MAJ_3, and cases (v), (vi), and (vii) cover XOR_3. Case (vii) contradicts (2) of the proposition; all others contradict (1).[1]

[1] Another way of establishing this result for XOR_3 is via linear algebra (being closed under XOR_3 is the same as being an affine subspace) and for MAJ_3 from the fact that relations closed under MAJ_3 are determined by their binary projections.

We give more details for case (i) for illustration. Taking AND_2 of the two tuples of weight t shown in the equation below, we get

$$\mathrm{AND}_2(1^t0^{k-t}, 01^t0^{k-t-1}) = 01^{t-1}0^{k-t},$$

a tuple of weight $t-1$. By symmetry, we obtain all tuples of weight $t-1$, which is the statement of case (i). Continuing this way, we obtain all tuples of weight $t-2$, $t-3$, etc. until we eventually obtain 0^k, which gives a contradiction as we assume that (1) of the proposition does not apply, so $0^k \notin \mathbf{B}$.

i $(\mathbf{t-1})$**-in-k** $\subseteq \mathrm{AND}_2($**t-in-k**$)$

tuples: 1^t0^{k-t} and 01^t0^{k-t-1}

eventual output: 0^k

ii $(\mathbf{t+1})$**-in-k** $\subseteq \mathrm{OR}_2($**t-in-k**$)$

tuples: 1^t0^{k-t} and 01^t0^{k-t-1}

eventual output: 1^k

iii if $t \geq 2$ then $(\mathbf{t+1})$**-in-k** $\subseteq \mathrm{MAJ}_3($**t-in-k**$)$

tuples: $1^{t-2}0^{k-t-1}110, 1^{t-2}0^{k-t-1}101$, and $1^{t-2}0^{k-t-1}011$

eventual output: 1^k

iv if $t \leq k-2$ then $(\mathbf{t-1})$**-in-k** $\subseteq \mathrm{MAJ}_3($**t-in-k**$)$

tuples: $1^{t-1}0^{k-t-2}100, 1^{t-1}0^{k-t-2}010$, and $1^{t-1}0^{k-t-2}001$

eventual output: 0^k

v if t is even, then $(\mathbf{t-2})$**-in-k** $\subseteq \mathrm{XOR}_3($**t-in-k**$)$

tuples: $1^{t-2}0^{k-t-1}110, 1^{t-2}0^{k-t-1}101$, and $1^{t-2}0^{k-t-1}011$

eventual output: 0^k

vi if t and k are odd, then $(\mathbf{t+2})$**-in-k** $\subseteq \mathrm{XOR}_3($**t-in-k**$)$

tuples: $1^{t-1}0^{k-t-2}100, 1^{t-1}0^{k-t-2}010$, and $1^{t-1}0^{k-t-2}001$

eventual output: 1^k

vii if t is odd and k is even,

then $(\mathbf{t+2})$**-in-k** $\subseteq \mathrm{XOR}_3($**t-in-k**$)$ if $t < k-2$,

and $(\mathbf{t-2})$**-in-k** $\subseteq \mathrm{XOR}_3($**t-in-k**$)$ if $t > 2$.

tuples:

$1^{t-1}0^{k-t-2}100, 1^{t-1}0^{k-t-2}010$, and $1^{t-1}0^{k-t-2}001$

$1^{t-2}0^{k-t-1}110, 1^{t-2}0^{k-t-1}101$, and $1^{t-2}0^{k-t-1}011$

eventual output: all odd weight tuples

□

With Proposition 4.17 in hand, we can prove Theorem 4.3.

Theorem (Theorem 4.4 restated)**.** *Let $k \geq 3$ and let $\mathbf{B} \subseteq \{0,1\}^k$ be a relation such that* **t-in-k** $\to \mathbf{B}$ *and* CSP($\mathbf{B}$) *is NP-hard. Then* PCSP(**t-in-k**, $\mathbf{B}$) *is tractable if and only if* $\mathbf{B} =$ **NAE**.

Proof. By Proposition 4.6, we can assume that $\mathbf{B}$ is symmetric. If $\mathbf{B} =$ **NAE** then PCSP(**t-in-k**, $\mathbf{B}$) is tractable by Proposition 4.1. Otherwise, we show that Pol(**t-in-k**, $\mathbf{B}$) does not contain any of the families from Theorem 2.62, and therefore PCSP(**t-in-k**, $\mathbf{B}$) is NP-hard.

The families we need to rule out are constants, OR, AND, XOR, AT, and THR_q for $q \in \mathbb{Q}$, as well as their negations. We deal first with the non-negated families. Since CSP($\mathbf{B}$) is NP-hard, by Proposition 4.17, we have $0^k \notin \mathbf{B}$ and $1^k \notin \mathbf{B}$. Hence, Pol(**t-in-k**, $\mathbf{B}$) does not contain constants.

Let C_k^t be the $k \times k$ matrix containing the k cyclic shifts of the column 1^t0^{k-t}. Then C_k^t prevents the polymorphism families OR, AND, XOR (if k is odd), and THR_q for all $q \neq \frac{t}{k}$. The case $q = \frac{t}{k}$ is ruled out by [41, Fact B.3], and AT is ruled out by [19, Claim 4.6]. Since CSP($\mathbf{B}$) is NP-hard, by Proposition 4.17, it remains to show that for even k, Pol(**t-in-k**, $\mathbf{B}$) excludes XOR when t is even, and likewise when t is odd and $\mathbf{B}$ is missing a tuple of odd weight.

Let k and t be even. Applying XOR_k to the matrix C_k^t returns the tuple 0^k, so applying XOR_{k-1} to the first $k-1$ columns of C_k^t returns the last column $1^{t-1}0^{k-t}1$. We can "fill in" the 0's in the output by swapping 0/1 pairs of values in the input matrix. In particular, in the columns $k-1, k-3, \ldots, t+1$, we swap the entries in the pairs of rows $(k-1, k-2), (k-3, k-4), \ldots, (t+1, t)$, respectively. The resulting $k \times (k-1)$ matrix M then satisfies $\mathrm{XOR}_{k-1}(M) = 1^k$ and the arity $k-1$ is odd as required. An example with swapped values in bold is illustrated in Figure (4.3a).

$$\mathrm{XOR}_7\begin{pmatrix} 1&0&0&0&0&1&1\\ 1&1&0&0&0&0&1\\ 1&1&1&0&0&0&0\\ 1&1&1&1&\mathbf{1}&0&0\\ 0&1&1&1&\mathbf{0}&0&0\\ 0&0&1&1&1&1&\mathbf{1}\\ 0&0&0&1&1&1&\mathbf{0}\\ 0&0&0&0&1&1&1 \end{pmatrix} = \begin{matrix}1\\1\\1\\1\\1\\1\\1\\1\end{matrix}$$

(a) $t = 4$ and $k = 8$.

$$\mathrm{XOR}_5\left(\begin{array}{ccccc} 1&0&0&1&1\\ 1&1&0&0&1\\ 1&1&1&0&0\\ 0&1&1&1&0\\ 0&0&1&1&1\\ \hline 0&0&0&0&0 \end{array}\right) = \begin{matrix}1\\1\\1\\1\\1\\0\end{matrix}$$

(b) $t = 3$, $k = 6$, and $d = 5$.

Figure 4.3: XOR.

Now let k be even, t be odd, and suppose that $\mathbf{B}$ does not contain the tuple $\mathbf{x} = 1^d0^{k-d}$ of odd weight d. By Observation 4.10, we can assume without loss of generality that $t < d$. Then XOR_d applied to the matrix C_d^t padded with $k - d$ rows of 0's returns $\mathbf{x}$. An illustration is given in Figure (4.3b). Therefore $\mathrm{XOR} \not\subseteq \mathrm{Pol}(\textbf{t-in-k}, \mathbf{B})$.

Negations: Let F be a family of functions. We reduce the task of showing $\overline{F} \not\subseteq \mathrm{Pol}(\textbf{t-in-k}, \mathbf{B})$ to the already completed task of showing $F \not\subseteq \mathrm{Pol}(\textbf{t-in-k}, \mathbf{B})$. Let $\mathbf{x} \in \{0,1\}^k \setminus \mathbf{B}$, let $f \in F$ be a function of arity m, and let M be a $k \times m$ matrix of inputs to f whose columns are **t-in-k** tuples. We established $F \not\subseteq \mathrm{Pol}(\textbf{t-in-k}, \mathbf{B})$ by finding f and M with $f(M) = \mathbf{x}$, and in the remaining cases we must find $\overline{f} \in \overline{F}$ and M such that $\overline{f}(M) = \mathbf{x}$. But since $\overline{f}(M) = \mathbf{x} \Leftrightarrow f(M) = \overline{\mathbf{x}}$, it suffices to find $f \in F$ such that $f(M) = \overline{\mathbf{x}}$, where $\overline{\mathbf{x}} = (1 - x_1, \ldots, 1 - x_k)$ if $\mathbf{x} = (x_1, \ldots, x_k)$.

The families $\overline{\mathrm{AND}}$, $\overline{\mathrm{OR}}$, and $\overline{\mathrm{XOR}}$ (except when k is even and t is odd) are excluded from $\mathrm{Pol}(\textbf{t-in-k}, \mathbf{B})$ in the same way as AND, OR, and XOR, with the same matrices serving as counterexamples. To see that $\overline{\mathrm{AT}}$ and $\overline{\mathrm{THR}_{\frac{t}{k}}}$ are also excluded, let $\mathbf{x} \notin \mathbf{B}$ be a tuple of weight $d \neq t$. Then the tuple $\overline{\mathbf{x}}$ of weight $k - d$ can be returned by an AT function and a $\mathrm{THR}_{\frac{t}{k}}$ function by the arguments above. If $k - d = t$, then the AT and $\mathrm{THR}_{\frac{t}{k}}$ functions of arity 1 output $\overline{\mathbf{x}}$ on input $\overline{\mathbf{x}}$.

Finally, when k is even, t is odd, and $\mathbf{B}$ does not contain the tuple $\mathbf{x}$ of odd weight d, the XOR argument above applies since $\overline{\mathbf{x}}$ also has odd weight $k - d$. Again, if $k - d = t$, then the XOR function of arity 1 outputs $\overline{\mathbf{x}}$ on input $\overline{\mathbf{x}}$. □

Theorem (Theorem 4.3 restated)**.** *Let $k \geq 3$ and $\emptyset \neq \mathbf{S} \subseteq (\textbf{t-in-k})^c \cap \mathbf{NAE}$. If t is odd, k is even, and $\mathbf{S}$ contains tuples of only even weight, then* $\mathrm{PCSP}(\textbf{t-in-k}, \mathbf{NAE} \setminus \mathbf{S})$ *is tractable. Otherwise,* $\mathrm{PCSP}(\textbf{t-in-k}, \mathbf{NAE} \setminus \mathbf{S})$ *is NP-hard.*

Proof. The tractability in the first statement of the theorem is proved in Section 4.1. Otherwise, t is

even, or k is odd, or $\mathbf{S}$ contains a tuple of odd weight. Take $\mathbf{B} = \mathbf{NAE} \setminus \mathbf{S}$. Observe that case (1) of Proposition 4.17 does not apply as neither 0^k nor 1^k is part of the template. Moreover, case (2) of Proposition 4.17 does not apply either: If t is odd and k is even then $\mathbf{S}$ contains a tuple of odd weight and hence $\mathbf{NAE} \setminus \mathbf{S}$ cannot have all odd weight tuples. Thus, by Proposition 4.17, CSP($\mathbf{B}$) is NP-hard. Then, by Theorem 4.4, PCSP(**t-in-k**, $\mathbf{B}$) = PCSP(**t-in-k**, $\mathbf{NAE} \setminus \mathbf{S}$) is NP-hard. □

Chapter 5

PCSP(**1-in-3**, **B**) on non-Boolean domains

We establish new results on the complexity of PCSP(**1-in-3**, **B**).

5.1 Preliminaries and results

Let **B** contain a single ternary relation over the domain $[d] = \{0, 1, \ldots, d-1\}$ (for this chapter we redefine $[d]$ to include 0 and exclude d, as this makes it easier to work with modular arithmetic). Since the template **1-in-3** is symmetric, by Proposition 4.6 we can assume without loss of generality that **B** is symmetric. To any such **B** we associate a directed graph $G(\mathbf{B}) = ([d], \{(b_1, b_2) : (b_1, b_1, b_2) \in \mathbf{B}\})$. The complexity of PCSP(**1-in-3**, **B**) may depend on the presence in **B** of *rainbow tuples* (b_1, b_2, b_3) with $|\{b_1, b_2, b_3\}| = 3$, but the graph representation does not, so different choices of **B** can produce the same graph. For two graphs G and H over domains of possibly different sizes, we write $G \subseteq H$ when G is isomorphic to a subgraph of H.

The following two families of graphs play an important role in our results.

Definition 5.1. Let $b_1, \ldots, b_r$ be distinct elements of $[d]$. We define the *cyclic* and *linear order*

graphs, respectively, as follows:

$$C_r = (\{b_1, \ldots, b_r\}, \{(b_1, b_2), (b_2, b_3), \ldots, (b_r, b_1)\})$$
$$LO_r = (\{b_1, \ldots, b_r\}, \{(b_i, b_j) : b_i < b_j\}).$$

From a given graph we can recover a unique structure **B** by specifying which rainbow tuples to include; here we are primarily interested in the cases where all or no rainbow tuples are present.

Definition 5.2. Let G be a graph with vertex set $[d]$. We denote by $\mathbf{G}^+$ the structure with graph G that contains all rainbow tuples over $[d]$, and by $\mathbf{G}$ the same structure without rainbow tuples.

Example 5.3. Following Definition 5.1, let $C_3 = (\{0,1,2\}, \{(0,1),(1,2),(2,0)\})$ be a cyclic graph on three vertices. Then Definition 5.2 gives us the following relations.

$$\begin{aligned}\mathbf{C}_3 = \{&(0,0,1),(0,1,0),(1,0,0),\\ &(1,1,2),(1,2,1),(2,1,1),\\ &(2,2,0),(2,0,2),(0,2,2)\}\end{aligned}$$

$$\mathbf{C}_3^+ = \mathbf{C}_3 \cup \{(0,1,2),(0,2,1),(1,0,2),(1,2,0),(2,0,1),(2,1,0)\}$$

Observation 5.4.

- *For any G we have $\mathbf{G} \subseteq \mathbf{G}^+$, so that* PCSP(**1-in-3**, $\mathbf{G}^+$) $\leq_p$ PCSP(**1-in-3**, **G**).
- *If $G \subseteq H$, then* PCSP(**1-in-3**, $\mathbf{H} \cup \mathbf{S}$) $\leq_p$ PCSP(**1-in-3**, $\mathbf{G} \cup \mathbf{S}$) *for any set of rainbow tuples* **S**.

The following proposition presents instances of PCSP(**1-in-3**, **B**) that are known to be tractable.

Proposition 5.5. *If $\mathbf{C}_i \to \mathbf{B}$ for some $i \in \{1,2,3\}$, then* PCSP(**1-in-3**, **B**) *is tractable via AIP.*

Proof.

- If $\mathbf{C}_1 \to \mathbf{B}$, then **B** contains a constant tuple.
- The template $\mathbf{C}_2$ is the same as (Boolean) **NAE**, so (**1-in-3**, **B**) is a homomorphic relaxation of (**1-in-3**, **NAE**). Therefore, by Proposition 2.47, PCSP(**1-in-3**, **B**) is solved by AIP since PCSP(**1-in-3**, **NAE**) is [17, 19].

- If $\mathbf{C}_3 \to \mathbf{B}$, then $(\mathbf{1\text{-}in\text{-}3}, \mathbf{B})$ is a homomorphic relaxation of $\mathrm{CSP}(\mathbf{C}_3)$. We can write $\mathbf{C}_3 = ([3]; \{(x,y,z) : x+y+z \equiv 1 \pmod 3\})$, which, being a system of linear equations over a field, is solved by AIP. Therefore, by Proposition 2.47, $\mathrm{PCSP}(\mathbf{1\text{-}in\text{-}3}, \mathbf{B})$ is also solved by AIP.

□

The tractable cases in Proposition 5.5 are in fact just the beginning of a more general pattern.

Theorem 5.6. *For $\ell \in \{0,1\}$ and $r \geq 0$, $\mathrm{PCSP}(\mathbf{1\text{-}in\text{-}3}, \mathbf{C}^+_{2^\ell 3^r})$ is tractable via AIP.*

Next we show non-solvability by BLP+AIP for a certain family of PCSPs, namely those where $\mathbf{B}$ represents a linear order. Let $\mathbf{LO}_d$ be the symmetric relation containing all tuples (b_1, b_1, b_2) where $b_1 < b_2$ in the natural order on $[d]$. Note that $\mathbf{1\text{-}in\text{-}3} = \mathbf{LO}_2$.

Theorem 5.7. *For $k, d \geq 3$, $\mathrm{PCSP}(\mathbf{1\text{-}in\text{-}3}, \mathbf{LO}^+_d)$ is not solved by BLP+AIP.*

Theorem 5.7 shows that when $G(\mathbf{B})$ is the maximal acyclic graph over $[d]$, BLP+AIP does not solve $\mathrm{PCSP}(\mathbf{1\text{-}in\text{-}3}, \mathbf{B})$, so it is likely that cycles are necessary for tractability. Taking this result a step further to NP-hardness seems difficult: Already for $d = 3$, the authors of [9] were unable to show NP-hardness of $\mathrm{PCSP}(\mathbf{1\text{-}in\text{-}3}, \mathbf{LO}^+_3)$, which is the only remaining case of $\mathrm{PCSP}(\mathbf{1\text{-}in\text{-}3}, \mathbf{B})$ with unknown complexity for $d = 3$.

We turn now to templates $(\mathbf{1\text{-}in\text{-}3}, \mathbf{B})$ where $\mathbf{B}$ contains no rainbow tuples. Our first result is an algorithmic dichotomy.

Theorem 5.8. *Let $\mathbf{B}$ be a symmetric relation without rainbow tuples. Then $\mathrm{PCSP}(\mathbf{1\text{-}in\text{-}3}, \mathbf{B})$ is solvable by AIP if $C_i \subseteq G(\mathbf{B})$ for some $i \in \{1,2,3\}$, and is not solvable by BLP+AIP otherwise.*

By generalising [9, Theorem 32], we obtain NP-hardness for a subset of these rainbow-free templates.

Theorem 5.9. *For $d \geq 4$, $\mathrm{PCSP}(\mathbf{1\text{-}in\text{-}3}, \mathbf{C}_d)$ is NP-hard.*

The remainder of this chapter is dedicated to proving the four preceeding theorems.

5.2 All rainbows: tractability

We use the following number-theoretic result to construct polymorphisms of $(\mathbf{1\text{-}in\text{-}3}, \mathbf{C}^+_{2^\ell 3^r})$ which give the tractability in Theorem 5.6.

Theorem 5.10 ([38, Theorem 1])**.** *Let $0 < a, c < n$ and define the recurrence $s_t = as_{t-1} + c \mod n$.*

For all seed values s_0, we have $\{s_0, s_1, \ldots, s_{n-1}\} = \{0, 1, \ldots, n-1\}$ if and only if all of the following hold.

1. $\gcd(n, c) = 1$
2. *for all primes p, if $p|n$, then $p|(a-1)$*
3. *if $4|n$, then $4|(a-1)$.*

Theorem 5.6 is proved for $\ell = 0$ in Lemma 5.11 and $\ell = 1$ in Lemma 5.13.

Lemma 5.11. *For $r \geq 0$, PCSP(**1-in-3**, $\mathbf{C}^+_{3^r}$) is solvable by AIP.*

Proof. We show that Pol(**1-in-3**, $\mathbf{C}^+_{3^r}$) has alternating functions (Definition 2.38) of all odd arities. An alternating function $f : \{0,1\}^{2L+1} \to [d]$ can be represented by a unary function $f(s) : \{-L, \ldots, L+1\} \to [d]$ whose argument s encodes the alternating sum of the inputs to f:

$$s = \sum_{\substack{i=1 \\ i \text{ odd}}}^{2L+1} x_i - \sum_{\substack{i=2 \\ i \text{ even}}}^{2L} x_i.$$

Any function whose output is determined solely by s is alternating. We will use the values of s mod 3^r to define a family of alternating functions in Pol(**1-in-3**, $\mathbf{C}^+_{3^r}$), where each member f of the family satisfies $f(s) = f(s')$ if $s \equiv s' \pmod{3^r}$. All congruences in this proof are mod 3^r, so for readability we will no longer specify this. However, we will still write "mod 3^r" when it is necessary to obtain a value in $\{0, \ldots, 3^r - 1\}$.

To obtain a sequence of output values for our functions, we apply Theorem 5.10 with $n = 3^r$, $a = n - 2$, and $c = 1$ to the recurrence $s_t = 1 - 2s_{t-1} \pmod{3^r}$; the reason for this recurrence will become clear later. The first condition of the theorem is satisfied since $c = 1$. Next, the only prime dividing n is 3, which also divides $a - 1 = n - 3$, so the second condition is satisfied. Finally, for $n = 3^r$ we never have $4|n$, so the third condition is vacuously true. Therefore there exists a sequence $\mathbf{x} = \mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{3^r-1}$ of 3^r distinct values that satisfies the recurrence. Without loss of generality (since the domain values can be cyclically rotated) we set $\mathbf{x}_0 = 0$ and define $f : \{0, \ldots, 3^r - 1\} \to \{0, \ldots, 3^r - 1\}$ by $f(s) = t$, where t is the unique index satisfying $\mathbf{x}_t \equiv s$.

We now check that f is a polymorphism. Let s_1, s_2, and s_3 be the coordinate-wise alternating sums over a set of $2L+1$ **1-in-3** tuples. Since there are $L+1$ odd coordinates and L even coordinates

in each sum, we have $s_1+s_2+s_3 = 1$. It suffices to check that if $s_1+s_2+s_3 = 1$ and $f(s_1) = f(s_2) = t$, then $f(s_3) = t+1 \bmod 3^r$, so that the resulting tuple is in $\mathbf{C}^+_{3^r}$.

By definition, t satisfies $\mathbf{x}_t \equiv s_1 \equiv s_2$, so we have

$$\begin{aligned} s_1 + s_2 + s_3 &= 1 \\ s_1 + s_2 + s_3 &\equiv 1 \\ \mathbf{x}_t + \mathbf{x}_t + s_3 &\equiv 1 \\ s_3 &\equiv 1 - 2\mathbf{x}_t \\ s_3 &\equiv \mathbf{x}_{t+1 \bmod 3^r}, \end{aligned}$$

where the final line follows from the definition of the recurrence, to which $\mathbf{x}$ is a solution. Therefore $f(s_3) = t+1 \bmod 3^r$, as desired.

Below is an example of $f \in \mathrm{Pol}(\mathbf{1\text{-}in\text{-}3}, \mathbf{C}^+_9)$ of arity 43.

sequence $\mathbf{x}$	0	1	8	3	4	2	6	7	5
$f(s)$	0	1	2	3	4	5	6	7	8
	0	1	8	3	4	2	6	7	5
signed	9	10	17	12	13	11	15	16	14
weights	18	19	-1	21	22	20	-3	-2	-4
s	-9	-8	-10	-6	-5	-7	-12	-11	-13
	-18	-17	-19	-15	-14	-16	-21	-20	

□

Corollary 5.12. *The polymorphisms defined in the proof of Lemma 5.11 also give tractability for templates* $(\mathbf{1\text{-}in\text{-}3}, \mathbf{B})$ *obtained by removing some rainbow tuples from* $(\mathbf{1\text{-}in\text{-}3}, \mathbf{C}^+_{3^r})$. *In particular, with* $\mathbf{x}$ *as in the proof of Lemma 5.11,* $\mathrm{PCSP}(\mathbf{1\text{-}in\text{-}3}, \mathbf{B})$ *is tractable when* $\mathbf{B}_0 \to \mathbf{B}$, *where*

$$\mathbf{B}_0 = \mathbf{C}_{3^r} \cup \{(a,b,c) : |\{a,b,c\}| = 3,\ \mathbf{x}_a + \mathbf{x}_b + \mathbf{x}_c \equiv 1 \ (mod\ 3^r)\}.$$

However, some rainbow tuples are necessary for solvability by BLP+AIP, as shown in Proposition 5.8.

Lemma 5.13. *For* $r \geq 0$, $\mathrm{PCSP}(\mathbf{1\text{-}in\text{-}3}, \mathbf{C}^+_{2\cdot 3^r})$ *is solvable by AIP.*

Proof. We extend the construction from the proof of Lemma 5.11 by doubling the number of columns in the function table and then separating the weights by sign. Let $\mathbf{x} = \mathbf{x}_0, \ldots, \mathbf{x}_{3^r-1}$ be as in the proof of Lemma 5.11. Let $\mathbf{y} = \mathbf{x}, \mathbf{x}$ be the sequence of length $2 \cdot 3^r$ obtained by appending $\mathbf{x}$ to itself. We set $f(s) = t$, where for $s > 0$, t is the odd index satisfying $\mathbf{y}_t \equiv s \pmod{3^r}$, and for $s \leq 0$, t is the even index satisfying $\mathbf{y}_t \equiv s \pmod{3^r}$. Since $\mathbf{x}$ has odd length, each congruence class will appear exactly once at an odd index of $\mathbf{y}$ and exactly once at an even index of $\mathbf{y}$.

To see that f is a polymorphism, we must check that if $s_1 + s_2 + s_3 = 1$ and $f(s_1) = f(s_2) = t$, then $f(s_3) = t + 1 \mod 2 \cdot 3^r$. By definition, t satisfies $\mathbf{y}_t \equiv s_1 \equiv s_2 \pmod{3^r}$, so we have

$$\begin{aligned}
s_1 + s_2 + s_3 &= 1 \\
s_1 + s_2 + s_3 &\equiv 1 && \pmod{3^r} \\
\mathbf{y}_t + \mathbf{y}_t + s_3 &\equiv 1 && \pmod{3^r} \\
s_3 &\equiv 1 - 2\mathbf{y}_t && \pmod{3^r} \\
s_3 &\equiv 1 - 2\mathbf{x}_{t \bmod 3^r} && \pmod{3^r} \\
s_3 &\equiv \mathbf{x}_{t+1 \bmod 3^r} && \pmod{3^r}.
\end{aligned}$$

The penultimate congruence follows from $\mathbf{x}_{t \bmod 3^r} = \mathbf{y}_t$, since $\mathbf{y} = \mathbf{x}, \mathbf{x}$ implies that indices at a (cyclic) distance of 3^r contain the same value. The last congruence follows from the definition of the recurrence, to which $\mathbf{x}$ is a solution.

There are exactly two indices in $\mathbf{y}$ (one in each block $\mathbf{x}$) whose entries are equal to $\mathbf{x}_{t+1 \bmod 3^r}$: index $t + 1 \mod 2 \cdot 3^r$, and index $t + 1 + 3^r \mod 2 \cdot 3^r$. Therefore, we get

$$\begin{aligned}
s_3 &\equiv \mathbf{y}_{t+1 \bmod 2 \cdot 3^r} \pmod{3^r} \quad \text{and} \\
s_3 &\equiv \mathbf{y}_{t+1+3^r \bmod 2 \cdot 3^r} \pmod{3^r},
\end{aligned}$$

so that either

$$\begin{aligned}
f(s_3) &= t + 1 \bmod 2 \cdot 3^r \quad \text{or} \\
f(s_3) &= t + 1 + 3^r \bmod 2 \cdot 3^r.
\end{aligned}$$

Since s_1 and s_2 have the same sign ($\text{sign}(0) = -1$), s_3 must have the opposite sign, so $f(s_3)$ has the opposite parity of $f(s_1) = t$. But t and $t + 1 + 3^r \mod 2 \cdot 3^r$ have the same parity, so we conclude that $f(s_3) = t + 1 \mod 2 \cdot 3^r$.

Example of $f \in \text{Pol}(\mathbf{1\text{-}in\text{-}3}, \mathbf{C}_6^+)$ of arity 21.

sequence $\mathbf{y}$	0	1	2	0	1	2
$f(s)$	0	1	2	3	4	5
signed	0	1	-1	3	-2	2
weights	-3	4	-4	6	-5	5
s	-6	7	-7	9	-8	8
	-9	10	-10			11

□

Corollary 5.14. *The polymorphisms defined in the proof of Lemma 5.13 also give tractability for templates* $(\mathbf{1\text{-}in\text{-}3}, \mathbf{B})$ *obtained by removing some rainbow tuples from* $(\mathbf{1\text{-}in\text{-}3}, \mathbf{C}_{2\cdot 3^r}^+)$. *In particular, with* $\mathbf{y}$ *as in the proof of Lemma 5.13,* $\text{PCSP}(\mathbf{1\text{-}in\text{-}3}, \mathbf{B})$ *is tractable when* $\mathbf{B}_1 \to \mathbf{B}$, *where*

$$\begin{aligned}\mathbf{B}_1 = \mathbf{C}_{3^r} \cup \{(a,b,c) :\ & |\{a,b,c\}| = 3,\\ & \mathbf{y}_a + \mathbf{y}_b + \mathbf{y}_c \equiv 1 \ (\mathit{mod}\ 3^r),\\ & \{a,b,c\} \textit{ contains both even and odd values}\}.\end{aligned}$$

Conjecture 5.15. *Let* $\mathbf{B}_0$ *and* $\mathbf{B}_1$ *be as in Corollaries 5.12 and 5.14, respectively. Then* $\text{PCSP}(\mathbf{1\text{-}in\text{-}3}, \mathbf{B})$ *is tractable if and only if* $\mathbf{B}_0 \to \mathbf{B}$ *or* $\mathbf{B}_1 \to \mathbf{B}$.

Tractability arises from promise Tractability of $\text{PCSP}(\mathbf{1\text{-}in\text{-}3}, \mathbf{B})$ is interesting only when both $\text{CSP}(\mathbf{1\text{-}in\text{-}3})$ and $\text{CSP}(\mathbf{B})$ are NP-hard. By Theorem 2.32, $\text{CSP}(\mathbf{1\text{-}in\text{-}3})$ is NP-hard, so it remains to show that $\text{CSP}(\mathbf{B})$ is hard in the relevant cases. The case of domain size at most three is completely resolved in [9], and here we give a result for domain size at least four.

Definition 5.16. A function $f : A^m \to B$ is called *cyclic* if for all $a_1, \ldots, a_m \in A$, $f(a_1, \ldots, a_m) = f(a_2, a_3, \ldots, a_m, a_1)$.

Theorem 5.17 ([11, Theorem 4.1])**.** *Let* $\mathbf{B}$ *be a finite relational structure, and let* p *be a prime larger than the size of the domain of* $\mathbf{B}$. *If* $\text{Pol}(\mathbf{B})$ *has no cyclic polymorphism of arity* p, *then* $\text{CSP}(\mathbf{B})$ *is*

NP-hard.

Theorem 5.17 as stated in [11] applies only to *idempotent* CSPs; see [15, Theorem 6.9.2] for a discussion of the non-idempotent case.

Proposition 5.18. *Let* $\mathbf{B}$ *be a symmetric ternary relation over domain* $[d]$, $d \geq 4$, *that contains no constant tuples. If there exists* $X \subseteq [d]$, $|X| \geq 4$, *such that for all* $a, b, c \in X$ *with* $|\{a, b, c\}| = 3$ *we have* $(a, b, c) \in \mathbf{B}$, *then* CSP($\mathbf{B}$) *is NP-hard.*

Proof. Let $p > d$ be a prime with $p \equiv 3 \pmod 4$. Write $p = 4r + 3$ and assume without loss of generality that $X = \{0, 1, 2, 3\}$. Let $\mathbf{x}$ be the vector of length p consisting of r blocks of $(0, 1, 2, 3)$ followed by $(0, 1, 2)$. Let $\mathbf{y}$ and $\mathbf{z}$ be the cyclic shifts of $\mathbf{x}$ to the left by one and two indices, respectively:

$$\mathbf{x} = 0123\ 0123\ \ldots\ 0123\ 012$$
$$\mathbf{y} = 1230\ 1230\ \ldots\ 1230\ 120$$
$$\mathbf{z} = 2301\ 2301\ \ldots\ 2301\ 201.$$

For each i, $(\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i)$ is a rainbow tuple and hence is in $\mathbf{B}$. If f is a cyclic function of arity p, then $f(\mathbf{x}) = f(\mathbf{y}) = f(\mathbf{z})$ and so $(f(\mathbf{x}), f(\mathbf{y}), f(\mathbf{z}))$ is a constant tuple. Therefore $\mathbf{B}$ has no cyclic polymorphism of arity p, and we conclude by Theorem 5.17 that CSP($\mathbf{B}$) is NP-hard. □

Proposition 5.18 shows in particular that CSP($\mathbf{C}_d^+$) is NP-hard for $d \geq 4$, so the tractability established in Lemmas 5.11 and 5.13 arises from the promise structure.

Tractability from infinite CSPs All currently known PCSP tractability results can be obtained as homomorphic relaxations of tractable CSPs, possibly over infinite domains. In fact, infinite domains may be necessary [10]. We show that the preceeding tractability results can all be obtained from a homomorphic relaxation of a single tractable infinite domain CSP.

Proposition 5.19. *Let* $\mathbf{C} = (\mathbb{Z}, \{(x_1, x_2, x_3) : x_1 + x_2 + x_3 = 1\})$. *Then* PCSP(**1-in-3**, $\mathbf{B}$) *is solvable by AIP if and only if there exists a homomorphism* $\mathbf{C} \to \mathbf{B}$.

Proof. If there is a homomorphism $\mathbf{C} \to \mathbf{B}$, then we have homomorphisms **1-in-3** $\to \mathbf{C} \to \mathbf{B}$ so that (**1-in-3**, $\mathbf{B}$) is a homomorphic relaxation of $\mathbf{C}$. Therefore by [10, Lemma 4.8] there exists a

homomorphism $\mathrm{Pol}(\mathbf{C}) \to \mathrm{Pol}(\mathbf{1\text{-}in\text{-}3}, \mathbf{B})$, and since $\mathrm{CSP}(\mathbf{C})$ is solvable by AIP, [10, Theorem 7.19] implies that $\mathrm{PCSP}(\mathbf{1\text{-}in\text{-}3}, \mathbf{B})$ is as well. The converse follows from [10, Theorem 4.12]. □

The tractability results in Lemmas 5.11 and 5.13 can be seen as constructing a homomorphism $\mathbf{C} \to \mathbf{B}$, and then applying Proposition 5.19. We suspect that the tractable cases of $\mathrm{PCSP}(\mathbf{1\text{-}in\text{-}3}, \mathbf{B})$ are finitely intractable in the sense of [10], but the characterisation of finite intractability from [4] seems difficult to generalise to the non-Boolean setting.

5.3 All rainbows: hardness

In this section we prove Theorem 5.7 as well as non-solvability by BLP+AIP of some templates of the form $(\mathbf{1\text{-}in\text{-}3}, \mathbf{C}_d^+)$.

A next step toward the conjecture could be to prove non-solvability by BLP+AIP for templates $(\mathbf{1\text{-}in\text{-}3}, \mathbf{C}_d^+)$ where d is not of the form $2^\ell 3^r$ as in Theorem 5.6. We prove the cases $d = 4, 5, 7$ in Proposition 5.23, and have verified up to $d = 15$ with the help of a computer search, but a general method for all d remains elusive.

Lemma 5.20 (Generalisation of [9, Lemma 33])**.** *If* $\mathrm{Pol}(\mathbf{1\text{-}in\text{-}3}, \mathbf{B})$ *contains 2-block-symmetric functions of all odd arities, then* $\mathrm{Pol}(\mathbf{1\text{-}in\text{-}3}, \mathbf{B})$ *contains symmetric functions of all arities not divisible by 3.*

Proof. Let m be a positive integer not divisble by 3. We show that there exists a symmetric $f \in \mathrm{Pol}(\mathbf{1\text{-}in\text{-}3}, \mathbf{B})$ of arity m.

Since $3 \nmid m$, we have either $3|m+1$ or $3|m-1$. Assume $3|m+1$; the proof of the other case is analogous. Let $g = g(\mathbf{x}; \mathbf{y}) \in \mathrm{Pol}(\mathbf{1\text{-}in\text{-}3}, \mathbf{B})$ be a 2-block-symmetric function of arity $2m+1$ with symmetric blocks $\mathbf{x}$ and $\mathbf{y}$ of sizes m and $m+1$ respectively. We define $f(x_1, \ldots, x_m) = g(x_1, \ldots, x_m; \mathbf{y})$, where $\mathbf{y}$ has Hamming weight $\frac{m+1}{3}$. Then g depends only on its first block, and the weights in the second block allow a valid placement of $m+1$ **1-in-3** tuples. Since $g \in \mathrm{Pol}(\mathbf{1\text{-}in\text{-}3}, \mathbf{B})$, it follows that $f \in \mathrm{Pol}(\mathbf{1\text{-}in\text{-}3}, \mathbf{B})$. □

Corollary 5.21. *If* $\mathrm{Pol}(\mathbf{1\text{-}in\text{-}3}, \mathbf{B})$ *has no symmetric function of arity* m *for some* m *not divisible by 3, then* $\mathrm{PCSP}(\mathbf{1\text{-}in\text{-}3}, \mathbf{B})$ *is not solved by BLP+AIP.*

Lemma 5.22. *For every* $n \geq 1$*, there exists a sequence* $s_0, s_1, \ldots, s_{n+1}$ *of distinct elements of* $\{0, \ldots, 2^n\}$ *such that for each* t*, we have* $s_t + 2s_{t+1} = 2^n$*.*

Proof. Set

$$s_t = \frac{(-1)^t 2^{n+1-t} + 2^n}{3}.$$

To see that $s_t \in \mathbb{Z}$ for $0 \leq t \leq n+1$, we must show that $(-1)^t 2^{n+1-t} + 2^n \in 3\mathbb{Z}$. For all $i \in \mathbb{Z}$, $2^i \equiv 1 \pmod 3$ if i is even and $2^i \equiv 2 \pmod 3$ if i is odd. If t is even, then

$$(-1)^t 2^{n+1-t} + 2^n = 2^{n+1-t} + 2^n \equiv 2^{n+1} + 2^n \equiv 0 \pmod 3,$$

and if t is odd, then

$$(-1)^t 2^{n+1-t} + 2^n = -2^{n+1-t} + 2^n \equiv -2^n + 2^n \equiv 0 \pmod 3.$$

Finally, we check that $s_t + 2s_{t+1} = 2^n$.

$$\begin{aligned} s_t + 2s_{t+1} &= \frac{(-1)^t 2^{n+1-t} + 2^n}{3} + 2\frac{(-1)^{t+1} 2^{n+1-(t+1)} + 2^n}{3} \\ &= \frac{(-1)^t 2^{n+1-t} + 2(-1)^{t+1} 2^{n-t} + 2^n + 2^{n+1}}{3} \\ &= \frac{2^{n+1-t}((-1)^t + (-1)^{t+1}) + 3 \cdot 2^n}{3} \\ &= 2^n. \end{aligned}$$

□

Theorem (Theorem 5.7 restated)**.** *For $d \geq 2$,* PCSP(**1-in-3**, $\mathbf{LO}_d^+$) *is not solved by BLP+AIP.*

Proof. We show that Pol(**1-in-3**, $\mathbf{LO}_d^+$) contains no symmetric function f of arity 2^{d-1}. Let (s_t) be the sequence from Lemma 5.22 with $n = d-1$, so (s_t) has $d+1$ terms and satisfies $s_t + 2s_{t+1} = 2^{d-1}$. Therefore for each t, (s_t, s_{t+1}, s_{t+1}) is a valid tuple of Hamming weights arising from 2^{d-1} **1-in-3** tuples used as inputs to f.

We prove by induction on t that $f(s_t) < d - t$. First, note that $f(s_1) < d - 1$ as there is no tuple of the form $(f(s_0), f(s_1), f(s_1))$ with $f(s_1) = d - 1$ in $\mathbf{LO}_d^+$. Next, if $f(s_t) < d - t$, then $(f(s_t), f(s_{t+1}), f(s_{t+1})) \in \mathbf{LO}_d^+$ implies $f(s_{t+1}) < f(s_t) < d - t$, so $f(s_{t+1}) < d - t - 1 = d - (t+1)$ and the induction argument is complete.

Taking $t = d$ we get $f(s_d) < d - d = 0$, so there is no possible value for $f(s_d)$. Therefore $f \notin$ Pol(**1-in-3**, **B**), and by Corollary 5.21 PCSP(**1-in-3**, **B**) is not solved by BLP+AIP. □

Proposition 5.23. *For $d \in \{4, 5, 7\}$,* PCSP(**1-in-3**, $\mathbf{C}_d^+$) *is not solved by BLP+AIP.*

Proof. In each case we show that Pol(**1-in-3**, $\mathbf{C}_d^+$) does not contain a symmetric function of arity 2^d, so that the result follows by Corollary 5.21.

Solvability of PCSP(**1-in-3**, $\mathbf{C}_4^+$) by BLP+AIP was ruled out in [9, Theorem 34] by showing that Pol(**1-in-3**, $\mathbf{C}_4^+$) contains no symmetric function of arity 23, so our non-solvability result for $d = 4$ is not new. However, by ruling out symmetric polymorphisms of arity $2^4 = 16$, we show that $d = 4$ is just the first case of a more general pattern. Except for d =4, 5, and 7, our results are established by an exhaustive computer search.

As before, for a symmetric function $f : \{0, 1\}^m \to [d]$, we write $f(w)$ for the value of f on an input of Hamming weight $0 \leq w \leq m$.

Case $d = 4$: Suppose without loss of generality that $f(5) = 0$. Then f must map the tuple of input weights $(5, 5, 6)$ to $(0, 0, 1)$, so $f(6) = 1$. Similarly, we deduce that $f(4) = 2$, $f(8) = 3$, $f(0) = 0$, $f(16) = 1$, and $f(11) = 1$, giving the following table of values for f.

output	0	1	2	3
input	5	6	4	8
weight	0	16		
		11		

We now show that all possible values for $f(7)$ lead to a contradiction with the values of f already defined. Consider the tuple of input weights $(5, 4, 7)$: If $f(7) = 0$, then we get the tuple $(0, 2, 0)$, and if $f(7) = 2$, we get the tuple $(0, 2, 2)$, neither of which is in **B**. If $f(7) = 1$, then $f(2) = f(3) = 2$, which together imply that $f(11) = 3$, in contradiction with $f(11) = 1$. Finally, if $f(7) = 3$, then $f(1) = 0$, which implies that $f(10) = 1$ and $f(0) = 2$, contradicting $f(0) = 0$.

Case $d = 5$: Suppose without loss of generality that $f(11) = 0$. By applying the same arguments as in the case $d = 4$, we get the following table of values for f.

output	0	1	2	3	4
input	0	10	1	8	5
weight	11	21	12	19	16
	22	32		30	

We now show that all possible values for $f(9)$ lead to a contradiction with the values of f already defined. Consider the tuple of input weights $(11, 12, 9)$: If $f(9) = 0$, then we get the tuple $(0, 2, 0)$, and

if $f(9) = 2$, then we get the tuple $(0, 2, 2)$, neither of which is in $\mathbf{B}$. If $f(9) = 1$, then $f(14) = f(13) = 2$, which gives $f(5) = 3$, contradicting $f(5) = 4$. If $f(9) = 3$, then $f(4) = 4$ and $f(12) = 0$, contradicting $f(12) = 2$. Finally, if $f(9) = 4$, then $f(7) = f(18) = 0$ and so f maps $(7, 7, 18)$ to $(0, 0, 0)$ which is not in $\mathbf{B}$.

Case $d = 7$: Suppose without loss of generality that $f(43) = 0$. By applying the same arguments as above, we get the following table of values for f.

output	0	1	2	3	4	5	6
input	0	42	1	40	5	32	21
weight	43	85	44	83	48	75	64
	86	128		126		118	

We now show that all possible values for $f(41)$ lead to a contradiction with the values of f already defined. Consider the tuple of input weights $(43, 44, 41)$: If $f(41) = 0$, then we get the tuple $(0, 2, 0)$, and if $f(41) = 2$, then we get the tuple $(0, 2, 2)$, neither of which is in $\mathbf{B}$. If $f(41) = 1$, then $f(45) = f(46) = 2$, $f(37) = f(38) = 3$, $f(53) = f(54) = 4)$, and $f(21) = 5$, contradicting $f(21) = 6$. If $f(41) = 3$, then $f(47) = 4$, $f(33) = 5$, $f(63) = 6$, and $f(1) = 0$, contradicting $f(1) = 2$. If $f(41) = 4$, then $f(46) = f(82) = 5$ and $f(0) = 6$, contradicting $f(0) = 0$. If $f(41) = 5$, then $f(46) = f(55) = 6$, $f(18) = f(36) = f(61) = 0$, $f(6) = f(56) = 1$, $f(66) = 2$, and $f(18) = 3$, contradicting $f(18) = 0$. Finally, if $f(41) = 6$, $f(23) = f(46) = 0$, $f(36) = f(59) = f(62) = 1$, $f(10) = f(56) = 2$, and $f(62) = 3$, contradicting $f(62) = 1$. □

While the proof of Proposition 5.23 is quite ad hoc, it does offer some insight into the polymorphisms of $(\mathbf{1\text{-}in\text{-}3}, \mathbf{C}_d^+)$ for general d: Certain sequences of input weights force these functions to follow the directed edges of the cycle. We now give one way to construct such sequences, which we will use in our BLP+AIP unsolvability results.

5.4 No rainbows: algorithmic dichotomy

Theorem (Theorem 5.8 restated)**.** PCSP(**1-in-3**, **B**) *is solvable by AIP if $C_i \subseteq G(\mathbf{B})$ for some $i \in \{1, 2, 3\}$, and is not solvable by BLP+AIP otherwise.*

Proof. The case where $G := G(\mathbf{B})$ contains a cycle of length at most three is covered by Proposition 5.5, so it remains to show the non-tractability part of the theorem. We assume that G contains a

cycle, as otherwise the result is an immediate consequence of Theorem 5.7. We show that there exists some m not divisble by 3 such that PCSP($\mathbf{1\text{-}in\text{-}3}, \mathbf{B}$) has no symmetric polymorphism of arity m; the result then follows by Corollary 5.21.

The outline of the proof is as follows. A symmetric function $f \in \mathrm{Pol}(\mathbf{1\text{-}in\text{-}3}, \mathbf{B})$ is significantly constrained by its behaviour on a particular sequence of input weights. By taking f of sufficiently large arity, this sequence of inputs is long enough to ensure that the corresponding sequence of outputs contains a closed walk on the vertices of G. The absence of rainbow tuples then forces the existence of many "shortcuts" within this walk, which together imply the existence of a directed 3-cycle, thus contradicting the assumption on G.

Let f be a symmetric function of arity $m = 2^{d+1} \not\equiv 0 \pmod 3$ and apply Lemma 5.22 with $n = d+1$ to obtain a sequence of weights s_t of length $d+3$. Each tuple $(f(s_i), f(s_{i+1}), f(s_{i+1}))$ with s_i, s_{i+1} consecutive elements of s_t represents a directed edge $f(s_{i+1}) \to f(s_i)$ in G. In this way, we get a directed walk $f(s_{d+3}) \to f(s_{d+2}) \to \cdots \to f(s_4) \to f(s_3)$. We ignore $f(s_2) \to f(s_1) \to f(s_0)$; the reason for this will become apparent later.

The walk $f(s_{d+3}) \to \cdots \to f(s_3)$ passes through $d+1$ vertices, and since G contains d distinct vertices, there exist $b > a$ such that $f(s_a) = f(s_b)$, that is, we have a closed walk W of the form $f(s_b) \to f(s_{b-1}) \to \cdots \to f(s_a) = f(s_b)$. Denote the weights giving rise to W by $w_i = s_{b-i}$ for $i = 0, 1, \ldots, b-(a+1)$. Since G contains none of C_1, C_2, C_3, the length of W is at least four.

Consider the tuples of weights $(w_0, w_i, m - w_0 - w_i)$ for $i = 2, 3, \ldots, b-(a+2)$. For these to be valid input weights for f, we require $0 \le m - w_0 - w_i \le m$. This follows from restricing the original walk s_t to $t \ge 3$: In this range, $0 < s_t < m/2$, so $0 < m - w_0 - w_i < m$ is a valid input weight.

First we show the existence of a shortcut in W from $f(w_0)$ to $f(w_2)$. Consider the possible values for $f(m-w_0-w_2)$. Since $\mathbf{B}$ contains no rainbow tuples, we must have either $f(m-w_0-w_2) = f(w_0)$ or $f(m-w_0-w_2) = f(w_2)$. In the latter case, we would obtain a tuple $(f(w_0), f(w_2), f(w_2))$, which implies the existence of an edge $f(w_2) \to f(w_0)$ in G and a directed 3-cycle $f(w_0) \to f(w_1) \to f(w_2) \to f(w_0)$, a contradiction. Therefore $f(m - w_0 - w_2) = f(w_0)$, and by the same reasoning there exists an edge $f(w_0) \to f(w_2)$. This is not a contradiction since the presence of the edge $f(w_0) \to f(w_2)$ does not create a *directed* 3-cycle.

We repeat this argument to show that there exists a shortcut in W from $f(w_0)$ to $f(w_3)$. Again we have either $f(m-w_0-w_3) = f(w_0)$ or $f(m-w_0-w_3) = f(w_3)$. The latter implies that $\mathbf{B}$ contains the tuple $(f(w_0), f(w_3), f(w_3))$ and therefore G contains the edge $f(w_3) \to f(w_0)$, resulting in a directed

3-cycle $f(w_0) \to f(w_2) \to f(w_3) \to f(w_0)$, again a contradiction. Therefore $f(m - w_0 - w_3) = f(w_0)$, so there exists an edge $f(w_0) \to f(w_3)$ in G.

Continuing this argument, we deduce that G has edges $f(w_0) \to f(w_i)$ for $i = 2, 3, \ldots, b-(a+2)$. But then $f(w_0) \to f(w_{b-(a+2)}) \to f(w_{b-(a+1)}) \to f(w_0)$ is a directed 3-cycle. Contradiction. □

5.5 NP-hardness of PCSP(**1-in-3**, $\mathbf{C}_d$)

We prove several properties about polymorphisms of (**1-in-3**, $\mathbf{C}_d$). Let $f : \{0,1\} \to \{0,1,\ldots,d-1\}$ be such a polymorphism of arity m. For a set $X \subseteq [m]$, define $f(X) = f(x_1, \ldots, x_m)$ where $x_i = 1$ if $i \in X$ and $x_i = 0$ otherwise. We say that X is an i-set for f if $f(X) = i$. We assume without loss of generality that $f(\emptyset) = 0$, since $f \in \mathrm{Pol}(\mathbf{1\text{-}in\text{-}3}, \mathbf{C}_d)$ if and only if $(f + i \mod d) \in \mathrm{Pol}(\mathbf{1\text{-}in\text{-}3}, \mathbf{C}_d)$ for all $i \in [d]$.

Lemma 5.24 (Generalisation of [9, Lemma 29])**.** *Suppose $f(\emptyset) = 0$. Then*

1. *f has no i sets for $i \in \{2, 3, \ldots, d-2\}$*
2. *f has no two disjoint 1-sets*

Proof. Suppose that X is an i-set and $i \in \{2, 3, \ldots, d-2\}$. Then f would map the tuple $(\emptyset, X, X^C)$ to $(0, i, z)$ which is not in $\mathbf{C}_d$ no matter the value of z.

If X and Y are disjoint 1-sets of f, then f maps $(X, Y, (X \cup Y)^C)$ to $(1, 1, z)$, which is in $\mathbf{C}_d$ only if $z = 2$. But then f has a 2-set, a contradiction. □

Lemma 5.25 (Generalisation of [9, Lemma 30])**.** *Let X and Y be disjoint subsets of $[m]$.*

1. *if $f(\emptyset) = f(X) = f(Y) = 0$, then $f(X \cup Y) = 0$*
2. *if $f(\emptyset) = 0$ and $f(X) = f(Y) = d-1$, then $f(X \cup Y) = 1$*

Proof. By compatibility with $\mathbf{C}_d$, $(X \cup Y)^C$ must be a 1-set so that f maps $(X, Y, (X \cup Y)^C)$ to $(0, 0, 1)$. Then f must map $(\emptyset, (X \cup Y)^C, X \cup Y)$ to $(0, 1, 0)$, giving $f(X \cup Y) = 0$.

For the second item, f maps $(X, Y, (X \cup Y)^C)$ to $(d-1, d-1, 0)$, and then $((X \cup Y), (X \cup Y)^C, \emptyset)$ maps to $(1, 0, 0)$, and we conclude that $f(X \cup Y) = 1$. □

Lemma 5.26 (Generalisation of [9, Lemma 31])**.** *If $f(\emptyset) = 0$ and f has no $(d-1)$-set of size at most two, then f has a singleton 1-set.*

Proof. Since $\emptyset$ is a 0-set, f applied to $(\emptyset, \emptyset, [m])$ must give $(0, 0, 1)$, so $[m]$ is a 1-set. Suppose that there does not exist a singleton 1-set y. By Lemma 5.24, singletons are not i-sets for $i \in \{2, 3, \dots, d-2\}$, nor are they $(d-1)$-sets or 1-sets by the assumption. Therefore every singleton is a 0-set.

By adding singletons to $\emptyset$ one at a time and applying the first part of Lemma 5.25, we conclude that $[m]$ is a 0-set, a contradiction. □

Theorem (Theorem 5.9 restated)**.** *For $d \geq 4$,* PCSP(**1-in-3**, $\mathbf{C}_d$) *is NP-hard.*

Proof. We apply Corollary 3.15 with $k = 2$ and $\ell = 5$. We assign to a polymorphism its *type* and define $\mathrm{sel}(f)$ as follows.

1. Type 1: f has a $(d-1)$-set X of size at most two. In this case we set $\mathrm{sel}(f) = X$.
2. Type 2: f does not have a $(d-1)$-set of size at most two but has a singleton 1-set $\{x\}$. We set $\mathrm{sel}(f) = \{x\}$.

Lemma 5.26 guarantees that every f is of one of these two types.

Let $(f_0, \alpha_{0,1}, \dots, f_\ell)$ be a chain of minors consisting of polymorphisms and note that $f_i(\emptyset) = 0$ for all i. If for some $i < j$, both f_i and f_j have type 2, then $\mathrm{sel}(f_i)$ and $\alpha_{i,j}^{-1}(\mathrm{sel}(f_j))$ are both 1-sets, so they have a nonempty intersection by the second part of Lemma 5.24.

Otherwise, since $\ell = 5$, the chain contains four polymorphisms $f_{i_1}, f_{i_2}, f_{i_3}, f_{i_4}$ of type 1 (with $i_1 < i_2 < i_3 < i_4$). Let $X_1 = \mathrm{sel}(f_{i_1})$ and $X_j = \alpha_{i_1,i_j}^{-1}$ for $j = 2, 3, 4$.

These four sets are $(d-1)$-sets (as preimages of $(d-1)$-sets). If they are pairwise disjoint, then $X_1 \cup X_2$ and $X_3 \cup X_4$ are disjoint sets and are 1-sets by the second part of Lemma 5.25, contradicting the second part of Lemma 5.24.

Therefore two of these sets, say X_j and $X_{j'}$, have a nonempty intersection. But then $Y := \mathrm{sel}(f_{i_j})$ and $Z := \alpha_{i_j,i_{j'}}^{-1}(\mathrm{sel}(f_{i_{j'}}))$ also have nonempty intersection as $X_j = \alpha_{i_1,i_j}^{-1}(Y)$ and $X_{j'} = \alpha_{i_1,i_j}^{-1}(Z)$. □

Chapter 6

Conclusions

In Chapter 3, we proved a dichotomy for a natural fragment of non-Boolean PCSPs. We showed that many existing NP-hardness criteria cannot achieve such a result, which suggests that the analysis of non-Boolean PCSPs is significantly more involved than that of their Boolean counterparts. Future work could focus on non-Boolean PCSPs satisfying weaker notions of symmetry: SetSAT is symmetric in the sense that it has a CNF representation, where unary literals are applied to the variables, but it is not symmetric in the relational sense. Bridging the gap between these two notions of symmetry is a promising departure point for further exploration of the non-Boolean world.

Another variant of SetSAT is obtained by drawing literals from an arbitrary collection of sets, for which we conjecture that the tractability threshold is $\frac{s}{s+1}$, where s is the size of the largest set in the collection. As evidence for this conjecture, we note that the random walk algorithm from the proof of Proposition 3.10 works down to the stated threshold. The challenge is to show hardness in this new setting, where polymorphisms are no longer conservative and so our smug set techniques cannot immediately be imported.

Rather than changing the structure of SetSAT, one could investigate approximability in its NP-hard regime, as was done for promise SAT in [8]. Not only is it NP-hard to distinguish between g-satisfiable and unsatisfiable instances of $(1, g, 2g+1)$-SAT, but assuming the Unique Games Conjecture (UGC) [55], it is also NP-hard to distinguish between instances where at least a $(1-\epsilon)$-fraction of the clauses are g-satisfiable and instances where no assignment satisfies even a $(1-2^{-2g-1}+\epsilon)$-fraction of the clauses [8]. In other words, almost-satisfiable instances of $(1, g, 2g+1)$-SAT exhibit *approximation resistance,* meaning that it is unlikely that any polynomial-time algorithm performs better than the

trivial randomised algorithm which achieves approximation ratio $1 - 2^{-2g-1}$. We conjecture that SetSAT also exhibits this phenomenon. More precisely, we expect the following problem to be NP-hard under the UGC: Given a $(1, g, k)$-SetSAT instance with $\frac{g}{k} < \frac{s}{s+1}$, decide whether it is possible to g-satisfy at least a $(1 - \epsilon)$-fraction of the clauses, or whether no assignment satisfies even a $(1 - (s+1)^{-k} + \epsilon)$-fraction of the clauses, for some $\epsilon > 0$.

Turning now to Chapter 4, we believe one of its main takeaways to be that the tractability of $(\mathbf{t\text{-}in\text{-}k}, \mathbf{NAE})$ is very fragile, which highlights its beauty and importance. Via Proposition 4.6, we showed that the classification of symmetric Boolean PCSPs from [41] holds more generally and requires only that the first structure be symmetric. Future work could therefore also consider PCSP templates $(\mathbf{A}, \mathbf{B})$ where only one of $\mathbf{A}$, $\mathbf{B}$ is subject to a domain or relational restriction, with the aim of lifting existing results to this more general setting. It remains a challenge to prove NP-hardness for PCSP$(\mathbf{t\text{-}in\text{-}k} \cup \{\mathbf{x}\}, \mathbf{NAE})$, and these problems could serve as a benchmark to evaluate future hardness criteria for PCSPs.

The hypergraph colouring problem PCSP$(\mathbf{1\text{-}in\text{-}3}, \mathbf{B})$ in Chapter 5 combines features of the PCSPs in Chapters 3 and 4: The richness imparted by non-Boolean domains, and the AIP-solvability that seems to be associated with **1-in-3**. We uncovered a surprising connection to number theory that gives rise to a non-trivial family of alternating polymorphisms, and provided evidence leading us to conjecture that all tractability arises from these polymorphisms. A next step toward resolving this conjecture could be to prove NP-hardness of PCSP$(\mathbf{1\text{-}in\text{-}3}, \mathbf{LO}_d)$ (without rainbow tuples) for all $d \geq 3$. The case $d = 3$ was solved in [9, Theorem 26], and since cycles play an important role in tractability, such a result may be within reach. The complexity of PCSP$(\mathbf{1\text{-}in\text{-}3}, \mathbf{LO}_3^+)$, the only unresolved case for domain size three in [9], also remains open.

Templates of the form $(\mathbf{1\text{-}in\text{-}3}, \mathbf{B})$ may be a natural starting point for further exploration of finite intractability, as all our tractable cases are relaxations of tractable infinite-domain CSPs. Finite intractability was established in [4] for a class of Boolean templates that includes $(\mathbf{1\text{-}in\text{-}3}, \mathbf{NAE})$, which is a special case of $(\mathbf{1\text{-}in\text{-}3}, \mathbf{B})$, but has yet to be studied in the non-Boolean setting.

One could also attempt to generalise our results to PCSP$(\mathbf{t\text{-}in\text{-}k}, \mathbf{B})$, where $\mathbf{B}$ is a symmetric k-ary relation: For certain t, k, and $\mathbf{B}$, Theorem 5.10 gives tractability along the lines of Theorem 5.6. However, the set of rainbow tuples is much more complicated when $k > 3$, and dependencies between sequences of such tuples can pose obstacles to tractability, as was the case with cycles when $k = 3$. Therefore we refrain from making a conjecture on the complexity of PCSP$(\mathbf{t\text{-}in\text{-}k}, \mathbf{B})$ in general.

Bibliography

[1] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

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

[3] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998.

[4] Kristina Asimi and Libor Barto. Finitely tractable promise constraint satisfaction problems. In *Proceedings of the 46th International Symposium on Mathematical Foundations of Computer Science (MFCS'21)*, volume 202 of *LIPIcs*, pages 11:1–11:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[5] Albert Atserias and Víctor Dalmau. Promise Constraint Satisfaction and Width. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms (SODA'22)*, pages 1129–1153, 2022.

[6] Per Austrin, Amey Bhangale, and Aditya Potukuchi. Simplified inapproximability of hypergraph coloring via t-agreeing families. *Electronic Colloquium on Computational Complexity*, 2019.

[7] Per Austrin, Amey Bhangale, and Aditya Potukuchi. Improved inapproximability of rainbow coloring. In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'20)*, pages 1479–1495. SIAM, 2020.

[8] Per Austrin, Venkatesan Guruswami, and Johan Håstad. $(2+\epsilon)$-SAT Is NP-hard. *SIAM Journal on Computing*, 46(5):1554–1573, 2017.

[9] Libor Barto, Diego Battistelli, and Kevin M. Berg. Symmetric Promise Constraint Satisfaction Problems: Beyond the Boolean Case. In *Proceedings of the 38th International Symposium on*

Theoretical Aspects of Computer Science (STACS'21), volume 187 of *LIPIcs*, pages 10:1–10:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[10] Libor Barto, Jakub Bulín, Andrei A. Krokhin, and Jakub Opršal. Algebraic approach to promise constraint satisfaction. *Journal of the ACM*, 68(4):28:1–28:66, 2021.

[11] Libor Barto and Marcin Kozik. Absorbing subalgebras, cyclic terms, and the constraint satisfaction problem. *Logical Methods in Computer Science*, 8(1):1–26, 2012.

[12] Libor Barto and Marcin Kozik. Combinatorial gap theorem and reductions between promise CSPs. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms (SODA'22)*, pages 1204–1220, 2022.

[13] Libor Barto, Andrei Krokhin, and Ross Willard. Polymorphisms, and How to Use Them. In Andrei Krokhin and Stanislav Živný, editors, *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*, pages 1–44. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2017.

[14] Libor Barto, Jakub Opršal, and Michael Pinsker. The wonderland of reflections. *Israel Journal of Mathematics*, 223(1):363–398, 2018.

[15] Manuel Bodirsky. *Complexity of Infinite-Domain Constraint Satisfaction.* Cambridge University Press, 2021.

[16] Joshua Brakensiek and Venkatesan Guruswami. New Hardness Results for Graph and Hypergraph Colorings. In *Proceedings of the 31st Conference on Computational Complexity (CCC'16)*, volume 50 of *LIPIcs*, pages 14:1–14:27. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

[17] Joshua Brakensiek and Venkatesan Guruswami. An Algorithmic Blend of LPs and Ring Equations for Promise CSPs. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'19)*, pages 436–455. SIAM, 2019.

[18] Joshua Brakensiek and Venkatesan Guruswami. Symmetric polymorphisms and efficient decidability of PCSPs. In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'20)*, pages 297–304. SIAM, 2020.

[19] Joshua Brakensiek and Venkatesan Guruswami. Promise Constraint Satisfaction: Algebraic Structure and a Symmetric Boolean Dichotomy. *SIAM Journal on Computing*, 50(6):1663–1700, 2021.

[20] Joshua Brakensiek, Venkatesan Guruswami, Marcin Wrochna, and Stanislav Živný. The power of the combined basic LP and affine relaxation for promise CSPs. *SIAM Journal on Computing*, 49:1232–1248, 2020.

[21] Alex Brandts, Marcin Wrochna, and Stanislav Živný. The Complexity of Promise SAT on Non-Boolean Domains. In *Proceedings of the 47th International Colloquium on Automata, Languages, and Programming (ICALP'20)*, volume 168, pages 17:1–17:13. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020.

[22] Alex Brandts, Marcin Wrochna, and Stanislav Živný. The complexity of promise SAT on non-Boolean domains. *ACM Transactions on Computation Theory*, 13(4):26:1–26:20, 2021.

[23] Alex Brandts and Stanislav Živný. Beyond PCSP(1-in-3,NAE). In *Proceedings of the 48th International Colloquium on Automata, Languages, and Programming (ICALP'21)*, volume 198 of *LIPIcs*, pages 121:1–121:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[24] Andrei Bulatov. A dichotomy theorem for nonuniform CSPs. In *Proceedings of the 58th Annual Symposium on Foundations of Computer Science (FOCS'17)*, pages 319–330. IEEE, 2017.

[25] Andrei Bulatov, Peter Jeavons, and Andrei Krokhin. Classifying the complexity of constraints using finite algebras. *SIAM journal of computing*, 34(3):720–742, 2005.

[26] Anand Ramannujan Chandra Chekuri. Conjunctive query containment revisited. In *Proceedings of the 5th International Conference on Database Theory*, 1997.

[27] Hubie Chen and Martin Grohe. Constraint satisfaction with succinctly specified relations. *Journal of Computer and System Sciences*, 76(8):847–860, 2010.

[28] Lorenzo Ciardo and Stanislav Živný. CLAP: A New Algorithm for Promise CSPs. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms (SODA'22)*, pages 1057–1068, 2022.

[29] David A. Cohen. Tractable decision for a constraint language implies tractable search. *Constraints*, 9(3):219–229, 2004.

[30] Stephen Cook. The complexity of theorem proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing (STOC'71)*, pages 151–158, 1971.

[31] Nadia Creignou, Sanjeev Khanna, and Madhu Sudan. *Complexity classifications of Boolean constraint satisfaction problems*, volume 7 of *SIAM monographs on discrete mathematics and applications*. SIAM, 2001.

[32] Nadia Creignou, Phokion G. Kolaitis, and Heribert Vollmer, editors. *Complexity of Constraints - An Overview of Current Research Themes [Result of a Dagstuhl Seminar]*, volume 5250 of *Lecture Notes in Computer Science*. Springer, 2008.

[33] Víctor Dalmau, Phokion G. Kolaitis, and Moshe Y. Vardi. Constraint satisfaction, bounded treewidth, and finite-variable logics. In *Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming*, volume 2470, pages 310–326, 2002.

[34] Irit Dinur. The PCP theorem by gap amplification. *Journal of the ACM*, 54(3), 2007.

[35] Irit Dinur, Venkatesan Guruswami, Subhash Khot, and Oded Regev. A new multilayered PCP and the hardness of hypergraph vertex cover. *SIAM Journal on Computing*, 34(5):1129–1146, 2005.

[36] Irit Dinur, Elchanan Mossel, and Oded Regev. Conditional hardness for approximate coloring. *SIAM Journal on Computing*, 39(3):843–873, 2009.

[37] Irit Dinur, Oded Regev, and Clifford Smyth. The hardness of 3-uniform hypergraph coloring. *Combinatorica*, 25(5):519–535, 2005.

[38] A.R. Dobell and T.E. Hull. Random number generators. *SIAM Review*, 4(3), 1962.

[39] Zdeněk Dvořák and Martin Kupec. On planar Boolean CSP. In *Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming (ICALP'15)*, 2015.

[40] Tomás Feder and Moshe Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: a study through datalog and group theory. *SIAM journal on computing*, 28(1):57–104, 1998.

[41] Miron Ficak, Marcin Kozik, Miroslav Olšák, and Szymon Stankiewicz. Dichotomy for Symmetric Boolean PCSPs. In *Proceedings of the 46th International Colloquium on Automata,*

Languages, and Programming (ICALP'19), volume 132 of *LIPIcs*, pages 57:1–57:12. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019.

[42] Eugene Freuder. Complexity of k-tree structured constraint satisfaction problems. In *Proceedings of the 8th National Conference on Artificial Intelligence (AAAI'90)*, volume 1, pages 4–9, 1990.

[43] Michael Garey and David Johnson. The complexity of near-optimal graph coloring. *Journal of the ACM*, 23(1):43–49, 1976.

[44] Michael Garey and David Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[45] Àngel J. Gil, Miki Hermann, Gernot Salzer, and Bruno Zanuttini. Efficient algorithms for description problems over finite totally ordered domains. *SIAM Journal on Computing*, 38(3):922–945, 2008.

[46] Martin Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *Journal of the ACM*, 54:1–24, 2007.

[47] Venkatesan Guruswami and Euiwoong Lee. Strong inapproximability results on balanced rainbow-colorable hypergraphs. *Combinatorica*, 38:547–599, 2018.

[48] Venkatesan Guruswami and Sai Sandeep. Rainbow coloring hardness via low-sensitivity polymorphisms. In *International Conference on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'19)*, 2019.

[49] Venkatesan Guruswami and Sai Sandeep. d-To-1 Hardness of Coloring 3-Colorable Graphs with O(1) Colors. In *Proceedings of the 47th International Colloquium on Automata, Languages, and Programming (ICALP'20)*, volume 168 of *LIPIcs*, pages 62:1–62:12, 2020.

[50] Pavol Hell and Jaroslav Nešetřil. On the complexity of H-coloring. *Journal of combinatorial theory, series B*, 48(1):92–110, 1990.

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

[52] Peter Jeavons, David Cohen, and Martin C. Cooper. Constraints, consistency and closure. *Artificial Intelligence*, 101(1):251–265, 1998.

[53] Peter Jeavons, David A. Cohen, and Marc Gyssens. Closure properties of constraints. *Journal of the ACM*, 44(4):527–548, 1997.

[54] Alexandr Kazda, Vladimir Kolmogorov, and Michal Rolínek. Even delta-matroids and the complexity of planar Boolean CSPs. *ACM Transactions on Algorithms*, 15(22):1–33, 2019.

[55] Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC'02)*, pages 767–775. ACM, 2002.

[56] Andrei Krokhin and Jakub Opršal. The complexity of 3-colouring H-colourable graphs. In *Proceedings of the 60th Annual IEEE Symposium on Foundations of Computer Science (FOCS'19)*, pages 1227–1239. IEEE, 2019.

[57] Andrei A. Krokhin, Jakub Opršal, Marcin Wrochna, and Stanislav Živný. Topology and adjunction in promise constraint satisfaction. *CoRR*, abs/2003.11351, 2020.

[58] Andrei A. Krokhin and Stanislav Živný, editors. *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

[59] Melven Krom. The decision problem for a class of first-order formulas in which all disjunctions are binary. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 13:15–20, 1967.

[60] Richard Ladner. On the structure of polynomial time reducibility. *Journal of the ACM*, 22(1):155–171, 1975.

[61] Leonid Levin. Universal search problems (in Russian). *Problems of Information Transmission (in Russian)*, 9(3):115–116, 1973.

[62] Tamio-Vesa Nakajima and Stanislav Živný. Linearly ordered colourings of hypergraphs. In *Proceedings of the 49th International Colloquium on Automata, Languages, and Programming (ICALP'22)*, 2022.

[63] Christos H. Papadimitriou. On selecting a satisfying truth assignment. In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science (FOCS'91)*, pages 163–169. IEEE Computer Society, 1991.

[64] Ran Raz. A parallel repetition theorem. *SIAM Journal on Computing*, 27(3):763–803, 1998.

[65] Thomas Schaefer. The complexity of satisfiability problems. In *Proceedings of the tenth Annual ACM Symposium on the Theory of Computing (STOC '78)*, pages 216–226, 1978.

[66] Mark Siggers. A strong Mal'cev condition for locally finite varieties omitting the unary type. *Algebra Universalis*, 64(1):15–20, 2010.

[67] Marcin Wrochna and Stanislav Živný. Improved hardness for H-colourings of G-colourable graphs. In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'20)*, pages 1426–1435, 2020.

[68] Dmitriy Zhuk. A proof of the CSP dichotomy conjecture. *Journal of the ACM*, 67(5):30:1–30:78, 2020.