Beyond PCSP($\mathbf{1\text{-}in\text{-}3}, \mathbf{NAE}$)

Alex Brandts
Department of Computer Science, University of Oxford, UK

Stanislav Živný
Department of Computer Science, University of Oxford, UK

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 using the strict constraints, the task is to find a solution using the weak constraints. While there are by now several dichotomy results for fragments of PCSPs, they all consider (in some way) symmetric PCSPs.

1-in-3-SAT and Not-All-Equal-3-SAT are classic examples of Boolean symmetric (non-promise) CSPs. While both problems are NP-hard, Brakensiek and Guruswami showed [SODA'18] that given a satisfiable instance of 1-in-3-SAT one can 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 focus on *non-symmetric* PCSPs. In particular, we study PCSP templates obtained from the Boolean template ($\mathbf{t\text{-}in\text{-}k}, \mathbf{NAE}$) by either adding tuples to $\mathbf{t\text{-}in\text{-}k}$ or removing tuples from $\mathbf{NAE}$. For the former, we classify all templates as either tractable or not solvable by the currently strongest known algorithm for PCSPs, the combined basic LP and affine IP relaxation of Brakensiek and Guruswami [SODA'20]. For the latter, we classify all templates as either tractable or NP-hard.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness; Theory of computation → Constraint and logic programming

Keywords and phrases promise constraint satisfaction, PCSP, polymorphisms, algebraic approach

Digital Object Identifier 10.4230/LIPIcs.ICALP.2021.121

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version *Full Version*: `https://arxiv.org/abs/2104.12800` [14]

Funding This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 714532).
Alex Brandts: Royal Society Enhancement Award and NSERC PGS Doctoral Award.
Stanislav Živný: Royal Society University Research Fellowship.

1 Introduction

How hard is it to find a 6-colouring of a graph if it is promised to be 3-colourable? We do not know but believe it to be NP-hard. Despite sustained effort, this so-called *approximate graph colouring* problem has been elusive since it was considered by Garey and Johnson almost 50 years ago [22]. The current state of the art, established in 2019, is NP-hardness of finding a 5-colouring of a 3-colourable graph [17]. Approximate graph colouring is an example of the very general promise constraint satisfaction problem, which is the focus of this paper. We start with (non-promise) constraint satisfaction problems to set the stage.

Constraint satisfaction. While deciding whether a graph is 2-colourable is solvable in polynomial time, deciding 3-colourability is NP-complete [26]. The *constraint satisfaction problem* (CSP) is a general framework that captures graph colourings and many other fundamental computational problems. Feder and Vardi initiated a systematic study of so-called fixed-template decision CSPs. Let $\mathbf{A}$ be a fixed finite relational structure, called the *template* or constraint language; i.e., $\mathbf{A}$ consists of a finite universe A and finitely many relations on A, each of possibly different arity. The fixed-template CSP over $\mathbf{A}$, denoted by $\mathrm{CSP}(\mathbf{A})$, is the class of CSPs in which all constraint relations come from $\mathbf{A}$. In more detail, $\mathrm{CSP}(\mathbf{A})$ denotes the following computational problem: Given a structure $\mathbf{X}$ over the same signature as $\mathbf{A}$, is there a homomorphism from $\mathbf{X}$ to $\mathbf{A}$, denoted by $\mathbf{X} \to \mathbf{A}$? (Formal definitions can be found in Section 2.) If $\mathbf{A} = K_3$ is a clique on 3 vertices then $\mathrm{CSP}(\mathbf{A})$ is precisely the standard graph 3-colouring problem.

A classic result of Schaefer shows that, for any $\mathbf{A}$ on a 2-element set, $\mathrm{CSP}(\mathbf{A})$ is either solvable in polynomial time or NP-complete. The non-trivial tractable cases from Schaefer's classification are taught in undergraduate algorithms courses: 2-SAT, (dual) Horn-SAT, and linear equations over $\{0, 1\}$. Two concrete CSPs that are NP-hard by Schaefer's result are the (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 triple exactly one variable is set to 1. Formally, 1-in-3-SAT is $\mathrm{CSP}(\mathbf{A})$, where $\mathbf{A} = (\{0, 1\}; \{(1, 0, 0), (0, 1, 0), (0, 0, 1)\})$. In Not-All-Equal-3-SAT, 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. Formally, Not-All-Equal-3-SAT is $\mathrm{CSP}(\mathbf{A})$, where $\mathbf{A} = (\{0, 1\}; \{0, 1\}^3 \setminus \{(0, 0, 0), (1, 1, 1)\})$.

If $\mathbf{A}$ is a graph (i.e., a single symmetric binary relation) then, as shown by Hell and Nešetřil [24], $\mathrm{CSP}(\mathbf{A})$ is either solvable in polynomial time or NP-complete. In this case, essentially the only non-trivial tractable case is graph 2-colouring.

Based on these two examples and a connection to logic, Feder and Vardi famously conjectured [20] that, for any finite $\mathbf{A}$, $\mathrm{CSP}(\mathbf{A})$ is either solvable in polynomial time or NP-complete. Bulatov [16], and independently Zhuk [31], proved the conjecture in the affirmative, both relying on the algebraic approach to CSPs [25, 15, 6]. In this case, the tractable cases are complicated and hard to describe in an elementary way.

Promise constraint satisfaction. Austrin, Guruswami, and Håstad [2] and Brakensiek and Guruswami [8, 9] initiated the investigation of the *promise constraint satisfaction problem* (PCSP), which is a vast generalisation of the CSP. Let $\mathbf{A}$ and $\mathbf{B}$ be two relational structures such that $\mathbf{A} \to \mathbf{B}$. The fixed-template PCSP over $\mathbf{A}$ and $\mathbf{B}$, denoted by $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$, is the following computational problem: Given $\mathbf{X}$ such that $\mathbf{X} \to \mathbf{A}$, find a homomorphism from $\mathbf{X}$ to $\mathbf{B}$ (which exists by the composition of the promised homomorphism from $\mathbf{X}$ to $\mathbf{A}$ and the homomorphism from $\mathbf{A}$ to $\mathbf{B}$). If we take $\mathbf{A} = K_3$ to be a clique on 3 vertices and $\mathbf{B} = K_6$ 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 article. Actually, what we described is the *search* version of the PCSP. The *decision* version is as follows: Given $\mathbf{X}$, return Yes if $\mathbf{X} \to \mathbf{A}$ and return No if $\mathbf{X} \not\to \mathbf{B}$. (The promise in the decision version is that it does not happen that $\mathbf{X} \not\to \mathbf{A}$ but $\mathbf{X} \to \mathbf{B}$.) It is well known that the decision version reduces to the search version but it is not known whether there is a reduction the other way [5]. In most results (including ours), hardness is established for the decision version and tractability for the search version.

If $\mathbf{A} = \mathbf{B}$ then $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is the same as $\mathrm{CSP}(\mathbf{A})$ and thus PCSPs indeed generalise CSPs. For CSPs, the decision and search versions are known to be equivalent [15].

Building on the result of Barto, Opršal, and Pinsker [7] that the complexity of $\mathrm{CSP}(\mathbf{A})$ is captured by certain types of identities of higher-order symmetries (called polymorphisms) of $\mathbf{A}$, Barto, Bulín, Krokhin, and Opršal showed that the basics of the algebraic approach developed for CSPs [7] can be generalised to PCSPs [17, 5], thus introducing a general methodology for investigating the computational complexity of PCSPs.

Related work. Motivated by the goal to understand the computational complexity of all fixed-template PCSPs, a recent line of research has focused on restricted classes of templates, with the main directions being Boolean templates (i.e., templates on a two-element set) and symmetric templates (i.e., all relations in the template satisfy that if a tuple belongs to a relation then so do all of its permutations).

Austrin, Guruswami, and Håstad [2] considered the $(1, g, k)$-SAT problem: Given an instance of k-SAT with the promise that there is an assignment satisfying at least g literals in each clause, find an assignment that satisfies at least one literal in each clause. They showed that this problem is NP-hard if $\frac{g}{k} < \frac{1}{2}$, and polynomial-time solvable otherwise. $(1, g, k)$-SAT is a Boolean PCSP with a (symmetric) template that includes the binary disequality relation and a relation containing all tuples of particular Hamming weights. The NP-hardness in [2] was proved via reduction from the label cover problem using the idea of polymorphisms lifted from CSPs to PCSPs. Building on the algebraic theory from [17, 5], Brandts, Wrochna, and Živný [13] extended the classification of $(1, g, k)$-SAT to arbitrary finite domains.

Brakensiek and Guruswami [9] managed to classify all PCSPs over symmetric Boolean templates with the disequality relation as NP-hard or solvable in polynomial time. Ficak, Kozik, Olšák, and Stankiewicz [21] extended this result to all symmetric Boolean templates.

In very recent work, Barto, Battistelli, and Berg [4] explored symmetric PCSPs on three- and four-element domains.

While the approximate graph colouring problem remains open, hardness was proved under stronger assumptions (namely Khot's 2-to-1 Conjecture [27] for k-colourings with $k \geq 4$ and its non-standard variant for 3-colourings) by Dinur, Mossel, and Regev [18]. Guruswami and Sandeep [23] recently established this result under a weaker assumption, the so-called d-to-1 conjecture for any fixed $d \geq 2$. For approximate *hypergraph* colouring, another important PCSP, NP-hardness was established by Dinur, Regev, and Smyth [19]. There has been some recent progress on approximate graph colourings [30] and related PCSPs, e.g. approximate graph homomorphism problems [28, 30], and rainbow vs. normal hypergraph colourings [1].

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 [17, 5]. At the moment, even an analogue of Schaefer's result, i.e., classifying all Boolean PCSPs, seems out of reach. Our second motivation is the pure beauty of the template $(\mathbf{1\text{-}in\text{-}3}, \mathbf{NAE})$. While $\text{PCSP}(\mathbf{1\text{-}in\text{-}3}, \mathbf{NAE})$ admits a polynomial-time algorithm [9, 10], Barto showed [3, 5] that this tractability result cannot be obtained via an algebraic reduction to tractable finite-domain CSPs.

Contributions. Consider the Boolean $\text{PCSP}(\mathbf{t\text{-}in\text{-}k}, \mathbf{NAE})$, which is a natural generalisation of $\text{PCSP}(\mathbf{1\text{-}in\text{-}3}, \mathbf{NAE})$. This is a symmetric, tractable PCSP. When can we add tuples to **t-in-k** to keep the PCSP tractable? When can we remove tuples from **NAE** to keep the PCSP tractable? Note that these changes generally do not give symmetric templates.

For the second question, we give a complete answer in Theorem 11: If t is odd, k is even, and tuples of only even Hamming weight are removed from **NAE**, the resulting PCSP is solvable in polynomial time. In all other cases, the resulting PCSP is NP-hard.

For the first question, we give a second-best possible answer in Theorem 10: If t is odd, k is even, and tuples of only odd Hamming 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 of Brakensiek and Guruswami [11], the currently strongest known algorithm for PCSPs. The power of this relaxation has recently been characterised by Brakensiek, Guruswami, Wrochna, and Živný [12].

2 Preliminaries

We denote by $[n]$ the set $\{1, 2, \ldots, n\}$. For a k-tuple $\mathbf{x}$, we write $\mathbf{x} = (x_1, \ldots, x_k)$. We denote by $\leq_p$ a polynomial-time many-one reduction.

A *relational structure* is a tuple $\mathbf{A} = (A; R_1, \ldots, R_p)$, where A is a finite set called the *domain* of $\mathbf{A}$, and each R_i is a relation of arity $\mathrm{ar}(R_i) \geq 1$, that is, R_i is a non-empty subset of $A^{\mathrm{ar}(R_i)}$. A relational structure is *symmetric* if each relation in it is invariant under any permutation of coordinates. Two relational structures $\mathbf{A} = (A; R_1, \ldots, R_p)$ and $\mathbf{B} = (B; S_1, \ldots, S_q)$ have the same *signature* if $p = q$ and $\mathrm{ar}(R_i) = \mathrm{ar}(S_i)$ for every $i \in [p]$. In this case, a mapping $\phi : A \to B$ is called a *homomorphism* from $\mathbf{A}$ to $\mathbf{B}$, denoted by $\phi : \mathbf{A} \to \mathbf{B}$, if ϕ preserves all relations; that is, for every $i \in [p]$ and every tuple $\mathbf{x} \in R_i$, we have $\phi(\mathbf{x}) \in S_i$, where ϕ is applied component-wise. The existence of a homomorphism from $\mathbf{A}$ to $\mathbf{B}$ is denoted by $\mathbf{A} \to \mathbf{B}$. A PCSP *template* is a pair $(\mathbf{A}, \mathbf{B})$ of relational structures over the same signature such that $\mathbf{A} \to \mathbf{B}$.

▶ **Definition 1.** *Let $(\mathbf{A}, \mathbf{B})$ be a PCSP template. The* decision version *of* PCSP$(\mathbf{A}, \mathbf{B})$ *is the following problem: Given as input a relational structure $\mathbf{X}$ over the same signature as $\mathbf{A}$ and $\mathbf{B}$, output* YES *if $\mathbf{X} \to \mathbf{A}$ and* NO *if $\mathbf{X} \not\to \mathbf{B}$. The* search version *of* PCSP$(\mathbf{A}, \mathbf{B})$ *is the following problem: Given as input a relational structure $\mathbf{X}$ over the same signature as $\mathbf{A}$ and $\mathbf{B}$ such that $\mathbf{X} \to \mathbf{A}$, find a homomorphism from $\mathbf{X}$ to $\mathbf{B}$.*

We call PCSP$(\mathbf{A}, \mathbf{B})$ *tractable* if any instance of PCSP$(\mathbf{A}, \mathbf{B})$ can be solved in polynomial time in the size of the input structure $\mathbf{X}$. It is easy to show that the decision version reduces to the search version [5]. Our hardness results will be for the decision version and our tractability results for the search version. For a relational structure $\mathbf{A}$, the constraint satisfaction problem with the template $\mathbf{A}$, denoted by CSP$(\mathbf{A})$, is PCSP$(\mathbf{A}, \mathbf{A})$.

The following notion of polymorphisms is at the heart of the algebraic approach to (P)CSPs. Intuitively, an arity m polymorphism of a PCSP template $(\mathbf{A}, \mathbf{B})$ is a homomorphism from the m-th Cartesian power of $\mathbf{A}$ to $\mathbf{B}$.

▶ **Definition 2.** *Let $(\mathbf{A}, \mathbf{B})$ be a PCSP template. A function $f : A^m \to B$ is a* polymorphism *of arity m 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 $(k \times m)$ matrix M whose columns are tuples in R_i, the application of f to rows of M gives a tuple in S_i. We denote by* Pol$(\mathbf{A}, \mathbf{B})$ *the set of all polymorphisms of $(\mathbf{A}, \mathbf{B})$.*

In a PCSP template $(\mathbf{A}, \mathbf{B})$ we view tuples from $\mathbf{A}$ and $\mathbf{B}$ as columns. When writing tuples in text we may write them as rows to simplify notation but they should still be understood as columns. For a k-ary relation R on the set $[t]$, we denote by $R^c = [t]^k \setminus R$ the complement of R. For a relational structure $\mathbf{A}$, we denote by $\mathbf{A}^c$ the structure with relations R^c for each relation R in $\mathbf{A}$. Most of our relational structures will be on the Boolean domain $\{0, 1\}$ and contain a single relation of arity k. The (Hamming) weight of a tuple $\mathbf{x} \in \{0, 1\}^k$, denoted throughout by d, is the number of 1's in $\mathbf{x}$. For $1 \leq t < k$, the Boolean relational structure **t-in-k** consists (of one relation consisting) of all k-tuples with weight t. The Boolean relational structure **NAE** contains all k-tuples except 0^k and 1^k.

We need a definition and some notation to state existing results on Boolean (P)CSPs.

▶ **Definition 3.** *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*

$$f(x_1, \ldots, x_m) = \mathbb{1}[x_1 - x_2 + x_3 - \cdots + x_m] > 0;$$

- *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 is a* $\frac{1}{2}$*-threshold.*

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 *and* XOR *the set of all* AT_m *and* XOR_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* inversion of f*, as the function* $x\mapsto 1-f(x)$*, and for a family of functions* F*, define the* inversion of F *by* $\overline{F}=\{\overline{f}|f\in F\}$*.*

Schaefer's celebrated dichotomy theorem classified all Boolean CSP templates.

▶ **Theorem 4** ([29])**.** *Let* $\mathbf{B}$ *be a Boolean CSP template. If* $\mathrm{Pol}(\mathbf{B})$ *contains a constant,* AND_2*,* OR_2*,* MAJ_3*, or* XOR_3*, then* $\mathrm{CSP}(\mathbf{B})$ *is tractable. Otherwise,* $\mathrm{CSP}(\mathbf{B})$ *is NP-hard.*

Ficak et al. classified all symmetric Boolean PCSP templates [21].

▶ **Theorem 5** ([21])**.** *Let* $(\mathbf{A},\mathbf{B})$ *be a symmetric Boolean PCSP template. If* $\mathrm{Pol}(\mathbf{A},\mathbf{B})$ *contains a constant or at least one of* OR*,* AND*,* XOR*,* AT*,* THR_q *(for some q) or their inversions, then* $\mathrm{PCSP}(\mathbf{A},\mathbf{B})$ *is tractable. Otherwise,* $\mathrm{PCSP}(\mathbf{A},\mathbf{B})$ *is NP-hard.*

The only possibly unresolved promise templates are those with NP-hard CSP templates.

▶ **Proposition 6.** *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.*

Theorem 4 established NP-hardness of two natural CSPs: $\mathrm{CSP}(\textbf{1-in-3})$ and $\mathrm{CSP}(\mathbf{NAE})$. Interestingly, $\mathrm{PCSP}(\textbf{1-in-3},\mathbf{NAE})$ is solvable in polynomial-time, as first shown by Brakensiek and Guruswami [9]. (Note that this shows that the converse of Proposition 6 is false.) A natural generalisation of **1-in-3** is **t-in-k**. Theorem 4 implies that $\mathrm{CSP}(\textbf{t-in-k})$ is NP-hard. Theorem 5 implies that the tractability of $\mathrm{PCSP}(\textbf{1-in-3},\mathbf{NAE})$ also holds for $\mathrm{PCSP}(\textbf{t-in-k},\mathbf{NAE})$; both observations are proved in the full version [14].

▶ **Proposition 7.** *For* $k\geq 3$ *and* $1\leq t<k, \mathrm{CSP}(\textbf{t-in-k})$ *is NP-hard.*

▶ **Proposition 8.** *For* $k\geq 2$ *and* $1\leq t<k, \mathrm{PCSP}(\textbf{t-in-k},\mathbf{NAE})$ *is tractable.*

We now give a characterisation of the power of a certain convex relaxation useful for tractability of (the decision version of) PCSPs. The relaxation is called the *combined basic LP and affine IP relaxation* (BLP+AIP) and was introduced in [11]; see also [10]. All known tractable PCSPs are solvable by either BLP+AIP or a reduction to a tractable CSP.

A $(2m+1)$-ary function f is called 2-block-symmetric if its $2m+1$ coordinates of f can be partitioned into two blocks of size $m+1$ and m in such a way that the value of f is invariant under any permutation of coordinates within each block. Without loss of generality, we will assume that the two blocks are the odd and even coordinates of f.

▶ **Theorem 9** ([12, Theorem 5.1])**.** *Let* $(\mathbf{A},\mathbf{B})$ *be a PCSP template. Then (the decision version of)* $\mathrm{PCSP}(\mathbf{A},\mathbf{B})$ *is tractable via BLP+AIP if and only if* $\mathrm{Pol}(\mathbf{A},\mathbf{B})$ *contains 2-block-symmetric functions of all odd arities.*

In the proof of one of our results (Theorem 10) we will use Theorem 9 to rule out solvability by BLP+AIP. However, for our tractability results, we will only need a characterisation result (in terms of polymorphisms) of a weaker relaxation, namely the affine IP relaxation (AIP) [5], which also works for the search version of PCSPs; details can be found in [14].

3 Our results

Our results are concerned with templates that arise from $(\mathbf{t\text{-}in\text{-}k}, \mathbf{NAE})$ by either adding tuples to $\mathbf{t\text{-}in\text{-}k}$ or removing tuples from $\mathbf{NAE}$. For a set of tuples $S \subseteq \{0,1\}^k$, 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 S, and similarly for $\mathbf{NAE} \setminus \mathbf{S}$.

▶ **Theorem 10** (Main #1). *Let $k \geq 3$ and $\emptyset \neq S \subseteq (\mathbf{t\text{-}in\text{-}k})^c \cap \mathbf{NAE}$. If t is odd, k is even, and S contains tuples of only odd weight, then* $\mathrm{PCSP}(\mathbf{t\text{-}in\text{-}k} \cup \mathbf{S}, \mathbf{NAE})$ *is tractable. Otherwise,* $\mathrm{PCSP}(\mathbf{t\text{-}in\text{-}k} \cup \mathbf{S}, \mathbf{NAE})$ *is not solved by BLP+AIP.*

The tractability part of Theorem 10 follows easily from existing work, cf. [14]. Our contribution is ruling out the applicability of BLP+AIP from [11]. Using Theorem 9, we prove the second part of Theorem 10 for $t = 1$ in Section 4 and for general t in [14].

▶ **Theorem 11** (Main #2). *Let $k \geq 3$ and $\emptyset \neq S \subseteq (\mathbf{t\text{-}in\text{-}k})^c \cap \mathbf{NAE}$. If t is odd, k is even, and S contains tuples of only even weight, then* $\mathrm{PCSP}(\mathbf{t\text{-}in\text{-}k}, \mathbf{NAE} \setminus \mathbf{S})$ *is tractable. Otherwise,* $\mathrm{PCSP}(\mathbf{t\text{-}in\text{-}k}, \mathbf{NAE} \setminus \mathbf{S})$ *is NP-hard.*

Again, the tractability part of Theorem 11 follows easily from existing work. Our contribution is establishing the hardness. It essentially follows from the following result.

▶ **Theorem 12.** *Let $k \geq 3$ and let $\mathbf{T} \subseteq \{0,1\}^k$ be a relation such that* $\mathrm{CSP}(\mathbf{T})$ *is NP-hard. Then* $\mathrm{PCSP}(\mathbf{t\text{-}in\text{-}k}, \mathbf{T})$ *is tractable if and only if $\mathbf{T} = \mathbf{NAE}$.*

Theorem 12 is proved in Section 5 and relies on Theorem 5, as well as a symmetrisation trick (Proposition 14, observed independently in [4]) and the following simple observation.

▶ **Proposition 13.** *Let $t, t' \geq 1$ and let R be a symmetric relation on $[t]$. For any function $f : [t] \to [t']$, the component-wise image of R under f, denoted $f(R)$, is also symmetric.*

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 14.** *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 must 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 13, $\phi(\mathbf{A})$ 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.

The reduction $\mathrm{PCSP}(\mathbf{A}, \mathbf{B}) \leq_p \mathrm{PCSP}(\mathbf{A}, \mathbf{B}')$ is trivial since $\mathbf{B}' \subseteq \mathbf{B}$.

To see that $\mathrm{PCSP}(\mathbf{A}, \mathbf{B}') \leq_p \mathrm{PCSP}(\mathbf{A}, \mathbf{B})$, suppose that a relation $R \in \mathbf{B}$ gives rise to the symmetric relation $R' \in \mathbf{B}'$ as described above. For each constraint $C = R'(x_1, ..., x_k)$ of the symmetrised instance of $\mathrm{PCSP}(\mathbf{A}, \mathbf{B}')$, we create $k!$ constraints by taking all coordinate permutations of C. Homomorphisms to $\mathbf{A}$ are preserved since $\mathbf{A}$ is symmetric. Conversely, suppose that we have a homomorphism ψ from the created instance to $\mathbf{B}$ and suppose that the tuple $\mathbf{x}$ was removed from R to create R'. Then no constraint is assigned $\mathbf{x}$ under ψ, since by our construction this would force all permutations of $\mathbf{x}$ to appear in constraints, violating membership in the relation R. Therefore, for any removed tuple $\mathbf{x}$, applying ψ does not produce $\mathbf{x}$ in any constraint, so ψ is a homomorphism from the original input to $\mathbf{B}'$. To complete the reduction, we repeat this construction for each $R \in \mathbf{B}$. ◀

4 Adding tuples for $t = 1$

We will rule out 2-block-symmetric polymorphisms of certain odd arities for PCSP templates of the form $(\mathbf{1\text{-}in\text{-}k} \cup \{\mathbf{x}\}, \mathbf{NAE})$ such that the weight of $\mathbf{x}$ is even if k is even. This implies the second part of Theorem 10 with $t = 1$. The more general case stated in Theorem 10, for any $t \geq 1$, can be found in [14].

Any 2-block-symmetric function $f : \{0,1\}^{2m+1} \to \{0,1\}$ of odd arity $2m+1$ is determined by the values on the $(m+2)(m+1)$ possible combinations of weights of the two blocks. Thus we will represent f by $f(x,y)$, where x and y are the number of 1's on the odd and even coordinates, respectively.

▶ **Proposition 15.** *Let $k \geq 3$ and $2 \leq d \leq \frac{k+1}{2}$ be such that if k is even then so is d. Let $\mathbf{x}$ be a tuple of weight d. Then,* $\mathrm{Pol}(\mathbf{1\text{-}\mathit{in}\text{-}k} \cup \{\mathbf{x}\}, \mathbf{NAE})$ *has no 2-block-symmetric function of arity* $2(k-d+1)+1$.

Proof. Since $\mathbf{NAE}$ is symmetric, permuting the rows of a matrix of inputs to a polymorphism permutes the values of the output tuple and does not affect membership in $\mathbf{NAE}$. Hence we will assume that $\mathbf{x} = 1^d 0^{k-d}$. Let f be a 2-block-symmetric function of arity $2(k-d+1)+1$. Thus the odd block is of size $k-d+2$ and the even block is of size $k-d+1$. We exhibit a set of tableaux such that for any f, one of these tableaux prevents f from being a polymorphism of $(\mathbf{1\text{-}in\text{-}k} \cup \{\mathbf{x}\}, \mathbf{NAE})$. Each tableau in this set contains the same construction on its even coordinates: the tuple $\mathbf{x}$ as its first column, followed by the $(k-d) \times (k-d)$ identity matrix below and to the right of $\mathbf{x}$, so that every row has exactly one 1. An illustration is given in Figure (1a).

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

(a) Even block.

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

(b) Odd block if $f(0,1) = f(1,1)$.

Figure 1 Example with $k = 9$ and $d = 4$.

It remains to give only the description of each tableau's odd coordinates.

If $f(0,1) = f(1,1)$, then the block of odd coordinates consists of the $(k-d+2) \times (k-d+2)$ identity matrix on top of $d-2$ rows of zeros. Since $2 \leq d < k$, the dimensions of the identity matrix are between 3 and k. An illustration is given in Figure (1b).

If $f(1,1) = f(2,1)$, then we obtain the block of odd coordinates by adding any tuple of weight 1 to the block of even coordinates.

Otherwise we have $f(0,1) \neq f(1,1) \neq f(2,1)$, so $f(0,1) = f(2,1)$.

If the number of odd coordinates $k-d+2$ is even, then we place two copies of the $\frac{k-d+2}{2} \times \frac{k-d+2}{2}$ identity matrix in the first $\frac{k-d+2}{2}$ rows, followed by zeros in the remaining rows. There are always enough rows to accommodate these matrices since $\frac{k-d+2}{2} \leq k \Leftrightarrow k+d \geq 2$. An illustration is given in Figure (2a).

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

(a) Odd block with $d = 5$ and $k - d + 2 = 6$ even.

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

(b) Odd block with $d = 4$ and $k - d + 2 = 7$ odd.

Figure 2 Example with $k = 9$ and $f(0, 1) = f(2, 1)$.

If $k - d + 2$ is odd then so is $k - d$. By the assumption "even k implies even d", we must have k odd and d even. The construction consists of the tuple $\mathbf{x}$ with the $d \times d$ identity matrix to its right. To fill the remaining $k - 2d + 1$ columns (assuming $d \leq \frac{k+1}{2}$), we place two copies of the $\frac{k-2d+1}{2} \times \frac{k-2d+1}{2}$ identity matrix starting from row $d + 1$ and column $d + 2$, and fill any remaining rows with zeros. There are always enough rows to accommodate the identity matrices since $\frac{k-2d+1}{2} \leq k - d$. An illustration is given in Figure (2b). ◀

▶ **Proposition 16.** *Let $k \geq 3$ and $\frac{k+1}{2} < d < k$ be such that if k is even then so is d. Let $\mathbf{x}$ be a tuple of weight d. Then,* $\text{Pol}(\mathbf{1\text{-}in\text{-}k} \cup \{\mathbf{x}\}, \mathbf{NAE})$ *has no 2-block-symmetric function of arity $2k + 1$.*

Proof. Without loss of generality let $\mathbf{x} = 1^d 0^{k-d}$. The proof is similar to the proof of Proposition 15: We will again exhibit a set of tableaux such that for any f, one of these tableaux prevents f from being a polymorphism of $(\mathbf{1\text{-}in\text{-}k} \cup \{\mathbf{x}\}, \mathbf{NAE})$. Each tableau in this set has the $k \times k$ identity matrix as its even coordinates, so it remains only to give a description of each tableau's odd coordinates. Consider the values $f(1, 1)$, $f(2, 1)$, and $f(3, 1)$: At least two of these must be equal since f takes only the values 0 and 1.

If $f(1, 1) = f(2, 1)$, then taking the odd coordinates to be the $k \times k$ identity matrix along with any other tuple of weight 1 prevents f from being a polymorphism.

If $f(1, 1) = f(3, 1)$, the odd coordinates are as follows: the first column is the tuple $\mathbf{x}$, followed immediately below and to the right by the $(k - d) \times (k - d)$ identity matrix, and then by two copies of the $\frac{d}{2} \times \frac{d}{2}$ identity matrix in the upper-right corner. The tuple $\mathbf{x}$ and the $(k - d) \times (k - d)$ matrix occupy $k - d + 1$ columns, leaving d columns to be filled by the two $\frac{d}{2} \times \frac{d}{2}$ identity matrices, and since d is even, $\frac{d}{2}$ is always an integer. An illustration is given in Figure (3a).

Finally, if $f(2, 1) = f(3, 1)$, the odd coordinates are as follows: two columns of $\mathbf{x}$, followed immediately below and to the right by two copies of the $(k - d) \times (k - d)$ identity matrix, and then by the $(2d - k - 1) \times (2d - k - 1)$ identity matrix in the upper-right corner. This fills all the columns. To place the two $(k - d) \times (k - d)$ identity matrices requires $2(k - d)$ columns after the $\mathbf{x}$'s. This is always possible since $2(k - d) \leq k - 1 \Leftrightarrow d \geq \frac{k+1}{2}$. After placing these, there remain $2d - k - 1$ columns for the final identity matrix, and since $(2d - k - 1) \leq k \Leftrightarrow d \leq k + \frac{1}{2}$, the number of available rows is never exceeded. An illustration is given in Figure (3b). ◀

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

(a) Odd block with $f(1,1) = f(3,1)$.

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

(b) Odd block with $f(2,1) = f(3,1)$.

Figure 3 Example with $k = 9$ and $d = 6$.

5 Removing tuples

In this section, we will prove Theorem 12 and show how it implies Theorem 11.

▶ **Theorem** (Theorem 12 restated). *Let $k \geq 3$ and let $\mathbf{T} \subseteq \{0,1\}^k$ be a relation such that $\mathrm{CSP}(\mathbf{T})$ is NP-hard. Then $\mathrm{PCSP}(\mathbf{t\text{-}in\text{-}k}, \mathbf{T})$ is tractable if and only if $\mathbf{T} = \mathbf{NAE}$.*

Proof. By Proposition 14, we can assume that $\mathbf{T}$ is symmetric. If $\mathbf{T} = \mathbf{NAE}$ then $\mathrm{PCSP}(\mathbf{t\text{-}in\text{-}k}, \mathbf{T})$ is tractable by Proposition 8. Otherwise, we show that $\mathrm{Pol}(\mathbf{t\text{-}in\text{-}k}, \mathbf{T})$ does not contain any of the tractable polymorphism families identified in the symmetric Boolean PCSP dichotomy (Theorem 5), and therefore $\mathrm{PCSP}(\mathbf{t\text{-}in\text{-}k}, \mathbf{T})$ is NP-hard.

The families are constants, OR, AND, XOR, AT, and THR_q for $q \in \mathbb{Q}$, as well as their inversions. We deal first with the non-inverted families. Since $0^k \notin \mathbf{T}$ and $1^k \notin \mathbf{T}$, $\mathrm{Pol}(\mathbf{t\text{-}in\text{-}k}, \mathbf{T})$ does not contain constants. Let C_k^t be the $k \times k$ matrix containing the k cyclic shifts of the column $1^t 0^{k-t}$. Then C_k^t prevents the polymorphism families OR, AND, XOR (if k is odd), and THR_q for all $q \neq \frac{t}{k}$. For all k, it remains to show that $\mathrm{Pol}(\mathbf{t\text{-}in\text{-}k}, \mathbf{T})$ contains neither $\mathrm{THR}_{\frac{t}{k}}$ nor AT. For even k it remains to show that $\mathrm{Pol}(\mathbf{t\text{-}in\text{-}k}, \mathbf{T})$ excludes XOR when t is even, and likewise when t is odd and $\mathbf{T}$ is missing a tuple of odd weight.

$\mathbf{THR}_{\frac{t}{k}}$: Suppose that $\mathbf{T}$ does not contain the tuple $\mathbf{x} = 1^d 0^{k-d}$ of weight d where $1 \leq d < k$. Note that we cannot have $d = t$ as this would violate $\mathbf{t\text{-}in\text{-}k} \to \mathbf{T}$, so either $t < d$ or $t > d$.

If $t < d$, then the $k \times d$ matrix M formed by placing C_d^t on top of the $(k-d) \times d$ zero matrix returns the forbidden tuple $\mathbf{x}$ when $\mathrm{THR}_{\frac{t}{k},d}$ is applied. An illustration is given in Figure (4a).

We must check that $\mathrm{THR}_{\frac{t}{k},d}$ is indeed in $\mathrm{THR}_{\frac{t}{k}}$, which requires that $d \times \frac{t}{k} \notin \mathbb{Z}$. This is equivalent to $dt \not\equiv 0 \pmod{k}$. However this is not always true, for example when $t = 2$, $k = 6$ and $d = 3$. When $dt \equiv 0 \pmod{k}$, we take instead $\mathrm{THR}_{\frac{t}{k},2d+1}$ and the $k \times (2d+1)$ input matrix M' consisting of two side-by-side copies of M and any extra column from M. In the first d rows of M, t of the d entries are 1's, so $\mathrm{THR}_{\frac{t}{k},d}$ returns 1 since $\frac{t}{d} > \frac{t}{k}$. Alternatively, in the first d rows of M', at least $2t$ of the $2d+1$ entries are 1's, so $\mathrm{THR}_{\frac{t}{k},2d+1}$ returns 1 since $\frac{2t}{2d+1} > \frac{t}{k}$. Both $\mathrm{THR}_{\frac{t}{k},d}$ and $\mathrm{THR}_{\frac{t}{k},2d+1}$ return 0 when applied to any of the last $k-d$ rows of M and M', respectively, since these rows contain only 0's. This completes the case $t < d$.

If $t > d$, let M be the $d \times (k-d)$ 1's matrix on top of C_{k-d}^{t-d}. Then applying $\mathrm{THR}_{\frac{t}{k},k-d}$ $\mathrm{THR}_{\frac{t}{k}}$ to M returns the forbidden tuple $\mathbf{x}$. An illustration is given in Figure (4b).

$$\mathrm{THR}_{\frac{3}{7},5}\left(\begin{array}{ccccc} 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ \hline 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right) = \begin{array}{c} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{array}$$

(a) $t = 3$ and $d = 5$.

$$\mathrm{THR}_{\frac{4}{7},5}\left(\begin{array}{ccccc} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{array}\right) = \begin{array}{c} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{array}$$

(b) $t = 4$ and $d = 2$.

Figure 4 Example with $k = 7$.

Again we have to check that $(k - d) \times \frac{t}{k} \notin \mathbb{Z}$, or equivalently, that $(k - d)t \not\equiv 0 \pmod{k}$. If $(k - d)t \equiv 0 \pmod{k}$, then $d \neq 1$, and we use instead $\mathrm{THR}_{\frac{t}{k},k-d+1}$ and the input matrix M' which is equal to M but with any one column repeated. Both $\mathrm{THR}_{\frac{t}{k},k-d}$ and $\mathrm{THR}_{\frac{t}{k},k-d+1}$ return 1 when applied to the first d rows of M and M', respectively. In the last $k - d$ rows of M, $t - d$ of the $k - d$ entries are 1's, so $\mathrm{THR}_{\frac{t}{k},k-d}$ returns 0 as $\frac{t-d}{k-d} < \frac{t}{k}$. In the last $k - d + 1$ rows of M', at most $t - d + 1$ of the $k - d + 1$ entries are 1's, so $\mathrm{THR}_{\frac{t}{k},k-d+1}$ returns 0 since $\frac{t-d+1}{k-d+1} < \frac{t}{k}$ (note that $d > 1$ in this case). This completes the proof for $t > d$, and thus we conclude that $\mathrm{THR}_{\frac{t}{k}} \not\subseteq \mathrm{Pol}(\mathbf{t\text{-in-}k}, \mathbf{T})$.

AT: Again we split into the cases $t < d$ and $t > d$.

Let $t < d$ and suppose that T does not contain the tuple $\mathbf{x} = 1^d 0^{k-d}$ of weight d where $1 \leq d < k$. We construct an input that returns $\mathbf{x}$ when a specific AT function is applied. This is done in two stages. First, we construct a matrix M such that for some AT function f, $f(M)$ agrees with $\mathbf{x}$ on all coordinates but one. Next we pad the input M with a matrix P such that for another AT function f', we have $f'(M|P|\cdots|P) = \mathbf{x}$.

To ease readability, we will separate the odd and even columns into two contiguous blocks, with the odd columns appearing first in the matrices. We take $f = \mathrm{AT}_{2d-1}$ and define M to be the following $k \times (2d - 1)$ matrix. In the d odd columns, we fill the first d rows with the matrix C_d^t. We fill the remaining $k - d$ rows with 0's. In the $d - 1$ even columns, we place on top a row of 1's, followed by $d - 1$ rows of 0's, then $t - 1$ rows of 1's, and finally $k - d - t + 1$ rows of 0's. An illustration is given in Figure (5a).

$$\mathrm{AT}_9\left(\begin{array}{ccccc|cccc} 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ \cline{6-9} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right) = \begin{array}{c} 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{array}$$

(a) AT_9.

$$P = \left(\begin{array}{ccc|ccc} 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \cline{4-6} 0 & 0 & 0 & 1 & 1 & 1 \end{array}\right)$$

(b) P.

Figure 5 $t = 3$, $k = 8$, and $d = 5$.

Notice that $\mathrm{AT}_{2d-1}(M)$ agrees with $\mathbf{x}$ everywhere except the first coordinate. This happens because the first row of M has too many 1's in the even columns; we would require at least $(d-1)-t+1=d-t$ more 1's in the odd columns for f to output 1 on the first row. We can achieve this by padding M with a matrix P that increases the proportion of 1's in the odd coordinates of the first row but does not affect the output of AT functions on the other rows.

Take $f' = \mathrm{AT}_{2d-1+2t(d-t)}$ and define the $k \times 2t$ matrix P as follows. In the t odd columns of P, we place t rows of 1's followed by $k-t$ rows of 0's. In the even columns of P, we place C_t^{t-1} in the first t rows, followed by $k-t-1$ rows of 0's, and then by a final row of 1's. An illustration is given in Figure (5b).

By padding M with copies of P, we increase the proportion of 1's in odd columns in the first $t<d$ rows, and in particular in the first row. The 0's in rows $t+1,\dots,k-1$ do not affect the output, and the 1's in the last row do not affect the output since they are in even columns and the last coordinate of x is always zero since $d<k$. Therefore $f'(M|P|\cdots|P)=\mathbf{x}$, where P appears $d-t$ times. This completes the case $t<d$.

Now let $t>d$ and again suppose that T does not contain the tuple $\mathbf{x}=1^d0^{k-d}$ of weight d. Define the $k\times(2d-1)$ input matrix M to AT_{2d-1} as follows. In the d odd columns of M, we first place d rows of 1's. Then in next $t-d+1$ rows, we place side-by-side as many copies of C_{t-d+1}^{t-d} as we can, removing columns of the last copy if necessary. We fill the remaining rows with 0's. In the $d-1$ even columns, we place a row of 0's, then t rows of 1's, and then fill the remaining rows with 0's. An illustration is given in Figure (6).

$$\mathrm{AT}_7\left(\begin{array}{cccc|ccc} 1&1&1&1&0&0&0\\ \cline{5-7} 1&1&1&1&1&1&1\\ 1&1&1&1&1&1&1\\ 1&1&1&1&1&1&1\\ \cline{1-4} 1&1&0&1&1&1&1\\ 1&0&1&1&1&1&1\\ 0&1&1&0&1&1&1\\ \hline 0&0&0&0&0&0&0 \end{array}\right) = \begin{array}{c}1\\1\\1\\1\\0\\0\\0\\0\end{array}$$

Figure 6 $t=6$, $k=8$, and $d=4$.

We have $\mathrm{AT}_{2d-1}(M)=x$, which concludes the case $t>d$. Therefore $\mathrm{AT}\not\subseteq \mathrm{Pol}(\mathbf{t\text{-}in\text{-}k},\mathbf{T})$.

XOR: Let t and k 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 $k-1, k-3, \dots, t+1$, we swap the entries in the pairs of rows $(k-1,k-2),(k-3,k-4),\dots,(t+1,t)$, 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 (7a).

Now let t be odd, k be even, and suppose that T does not contain the tuple $\mathbf{x}=1^d0^{k-d}$ of odd weight d. If $t<d$, then XOR_d applied to the input matrix C_d^t padded with $k-d$ rows of 0's returns $\mathbf{x}$. If $t>d$, let M be the $k\times(t-d+1)$ matrix with d rows of 1's followed by the matrix C_{t-d+1}^{t-d}. Fill any extra rows with 0's. Then $\mathrm{XOR}_{t-d+1}(M)=\mathbf{x}$. An illustration is given in Figure (7b). Therefore $\mathrm{XOR}\not\subseteq \mathrm{Pol}(\mathbf{t\text{-}in\text{-}k},\mathbf{T})$.

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

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

$$\mathrm{XOR}_3\left(\begin{array}{ccc} 1&1&1\\ \hline 1&1&0\\ 1&0&1\\ 0&1&1\\ \hline 0&0&0\\ 0&0&0 \end{array}\right) = \begin{matrix}1\\0\\0\\0\\0\\0\end{matrix}$$

(b) $t = 3$, $k = 6$, and $d = 1$.

Figure 7 XOR.

Inversions: Let F be a family of functions. We reduce the task of showing $\overline{F} \not\subseteq \mathrm{Pol}(\mathbf{t\text{-}in\text{-}k}, \mathbf{T})$ to the already completed task of showing $F \not\subseteq \mathrm{Pol}(\mathbf{t\text{-}in\text{-}k}, \mathbf{T})$. Let $\mathbf{x} \in \{0,1\}^k \setminus \mathbf{T}$, 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 $\mathbf{t\text{-}in\text{-}k}$ tuples. We established $F \not\subseteq \mathrm{Pol}(\mathbf{t\text{-}in\text{-}k}, \mathbf{T})$ 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 t is odd and k is even) are excluded from $\mathrm{Pol}(\mathbf{t\text{-}in\text{-}k}, \mathbf{T})$ 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 $\mathbf{x} \notin \mathbf{T}$ be a tuple of weight $d, d \neq t, 1 \leq d < k$. 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 t is odd and k is even, and $\mathbf{T}$ 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 $\overline{\mathbf{x}}$. ◀

Schaefer's dichotomy theorem (Theorem 4) allows us to obtain a simple description of all $\mathbf{T}$ with $\mathrm{CSP}(\mathbf{T})$ tractable and $\mathbf{t\text{-}in\text{-}k} \to \mathbf{T}$; a proof can be found in [14].

▶ **Proposition 17.** *Let $k \geq 3$, $1 \leq t < k$, and suppose that $\mathbf{t\text{-}in\text{-}k} \to \mathbf{T}$. Then $\mathrm{CSP}(\mathbf{T})$ is tractable if and only if*

1. *$0^k \in \mathbf{T}$ or $1^k \in \mathbf{T}$, or*
2. *t is odd, k is even, and $\mathbf{T}$ contains all tuples of odd weight.*

Observe that Proposition 17 in particular implies Proposition 7, NP-hardness of $\mathrm{CSP}(\mathbf{t\text{-}in\text{-}k})$.

With Proposition 17 in hand, we can prove Theorem 11.

▶ **Theorem** (Theorem 11 restated)**.** *Let $k \geq 3$ and $\emptyset \neq S \subseteq (\mathbf{t\text{-}in\text{-}k})^c \cap \mathbf{NAE}$. If t is odd, k is even, and S contains tuples of only even weight, then $\mathrm{PCSP}(\mathbf{t\text{-}in\text{-}k}, \mathbf{NAE} \setminus \mathbf{S})$ is tractable. Otherwise, $\mathrm{PCSP}(\mathbf{t\text{-}in\text{-}k}, \mathbf{NAE} \setminus \mathbf{S})$ is NP-hard.*

Proof. The tractability in the first statement of the theorem is proved in [14]. Otherwise, t is even, or k is odd, or S contains a tuple of odd weight. Take $\mathbf{T} = \mathbf{NAE} \setminus \mathbf{S}$. Observe that case (1) of Proposition 17 does not apply as neither 0^k nor 1^k is part of the template. Moreover, case (2) of Proposition 17 does not apply either: If t is odd and k is even then S contains a tuple of odd weight and hence $\mathbf{NAE} \setminus \mathbf{S}$ cannot have all odd weight tuples. Thus, by Proposition 17, CSP($\mathbf{T}$) is NP-hard. Then, by Theorem 12, PCSP($\mathbf{t\text{-}in\text{-}k}, \mathbf{T}$) = PCSP($\mathbf{t\text{-}in\text{-}k}, \mathbf{NAE} \setminus \mathbf{S}$) is NP-hard. ◀

References

1 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, 2020. doi:10.1137/1.9781611975994.90.

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

3 Libor Barto. Promises Make Finite (Constraint Satisfaction) Problems Infinitary. In *Proceedings of the 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS'19)*, pages 1–8. IEEE, 2019. doi:10.1109/LICS.2019.8785671.

4 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. doi:10.4230/LIPIcs.STACS.2021.10.

5 Libor Barto, Jakub Bulín, Andrei A. Krokhin, and Jakub Opršal. Algebraic approach to promise constraint satisfaction. *J. ACM*, 2018. arXiv:1811.00970.

6 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 für Informatik, Dagstuhl, Germany, 2017. doi:10.4230/DFU.Vol7.15301.1.

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

8 Joshua Brakensiek and Venkatesan Guruswami. New hardness results for graph and hypergraph colorings. In Ran Raz, editor, *31st Conference on Computational Complexity (CCC 2016)*, volume 50 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 14:1–14:27, Dagstuhl, Germany, 2016. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.CCC.2016.14.

9 Joshua Brakensiek and Venkatesan Guruswami. Promise Constraint Satisfaction: Structure Theory and a Symmetric Boolean Dichotomy. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'18)*, pages 1782–1801. SIAM, 2018. doi:10.1137/1.9781611975031.117.

10 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. doi:10.1137/1.9781611975482.28.

11 Joshua Brakensiek and Venkatesan Guruswami. Symmetric polymorphisms and efficient decidability of promise CSPs. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 297–304, 2020. doi:10.1137/1.9781611975994.18.

12 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. doi:10.1137/20M1312745.

13 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. doi:10.4230/LIPIcs.ICALP.2020.17.

14 Alex Brandts and Stanislav Živný. Beyond PCSP(1-in-3,NAE), 2021. `arXiv:2104.12800`.

15 Andrei Bulatov, Peter Jeavons, and Andrei Krokhin. Classifying the complexity of constraints using finite algebras. *SIAM Journal on Computing*, 34(3):720–742, 2005. `doi:10.1137/S0097539700376676`.

16 Andrei A. Bulatov. A dichotomy theorem for nonuniform CSPs. In *Proceedings of the 58th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 319–330, 2017. `doi:10.1109/FOCS.2017.37`.

17 Jakub Bulín, Andrei Krokhin, and Jakub Opršal. Algebraic approach to promise constraint satisfaction. In *Proceedings of the 51st Annual ACM SIGACT Symposium on the Theory of Computing (STOC '19)*, New York, NY, USA, 2019. ACM. `doi:10.1145/3313276.3316300`.

18 Irit Dinur, Elchanan Mossel, and Oded Regev. Conditional hardness for approximate coloring. *SIAM J. Comput.*, 39(3):843–873, 2009. `doi:10.1137/07068062X`.

19 Irit Dinur, Oded Regev, and Clifford Smyth. The hardness of 3-uniform hypergraph coloring. *Combinatorica*, 25(5):519–535, 2005. `doi:10.1007/s00493-005-0032-4`.

20 Tomás Feder and Moshe Y. Vardi. The Computational Structure of Monotone Monadic SNP and Constraint Satisfaction: A Study through Datalog and Group Theory. *SIAM Journal on Computing*, 28(1):57–104, 1998. `doi:10.1137/S0097539794266766`.

21 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, pages 57:1–57:12. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019. `doi:10.4230/LIPIcs.ICALP.2019.57`.

22 M. R. Garey and David S. Johnson. The complexity of near-optimal graph coloring. *J. ACM*, 23(1):43–49, 1976. `doi:10.1145/321921.321926`.

23 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. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. `doi:10.4230/LIPIcs.ICALP.2020.62`.

24 Pavol Hell and Jaroslav Nešetřil. On the Complexity of H-coloring. *Journal of Combinatorial Theory, Series B*, 48(1):92–110, 1990. `doi:10.1016/0095-8956(90)90132-J`.

25 Peter Jeavons, David A. Cohen, and Marc Gyssens. Closure properties of constraints. *J. ACM*, 44(4):527–548, 1997. `doi:10.1145/263867.263489`.

26 Richard M. Karp. Reducibility Among Combinatorial Problems. In *Proceedings of a Symposium on the Complexity of Computer Computations*, pages 85–103, 1972. URL: `http://www.cs.berkeley.edu/%7Eluca/cs172/karp.pdf`.

27 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. `doi:10.1145/509907.510017`.

28 Andrei Krokhin and Jakub Opršal. The complexity of 3-colouring H-colourable graphs. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS'19)*, pages 1227–1239, 2019. `doi:10.1109/FOCS.2019.00076`.

29 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. `doi:10.1145/800133.804350`.

30 Marcin Wrochna and Stanislav Živný. Improved hardness for H-colourings of G-colourable graphs. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'20)*, pages 1426–1435, 2020. `doi:10.1137/1.9781611975994.86`.

31 Dmitriy Zhuk. A proof of the CSP dichotomy conjecture. *J. ACM*, 67(5):30:1–30:78, 2020. `doi:10.1145/3402029`.