Electronic Colloquium on Computational Complexity, Report No. 82 (2025)

On the Usefulness of Promises

Per Austrin, Johan Håstad, and Björn Martinsson

KTH Royal Institute of Technology

23rd June 2025

Abstract

A Boolean predicate A is defined to be promise-useful if $\mathrm{PCSP}(A, B)$ is tractable for some non-trivial B and otherwise it is promise-useless. We initiate investigations of this notion and derive sufficient conditions for both promise-usefulness and promise-uselessness (assuming $\mathrm{P} \neq \mathrm{NP}$). While we do not obtain a complete characterization, our conditions are sufficient to classify all predicates of arity at most 4 and almost all predicates of arity 5. We also derive asymptotic results to show that for large arities a vast majority of all predicates are promise-useless.

Our results are primarily obtained by a thorough study of the "Promise-SAT" problem, in which we are given a k-SAT instance with the promise that there is a satisfying assignment for which the literal values of each clause satisfy some additional constraint.

The algorithmic results are based on the basic LP + affine IP algorithm of Brakensiek et al. (SICOMP, 2020) while we use a number of novel criteria to establish NP-hardness.

Contents

ISSN 1433-8092

1 Introduction

The class of Constraint Satisfaction Problems (CSPs) gives a very general framework that includes many well-known problems studied in mathematics and computer science. You are given a set of constraints over a set of variables, with each constraint depending only on a constant number of the variables, and your goal is to find an assignment that satisfies all constraints. By limiting the constraint language A, i.e., the types of constraints allowed, it is possible to create a wide range of different problems $\mathrm{CSP}(A)$. Central examples are given by 3-SAT and graph k-coloring.

A fundamental question for CSPs is to classify them as being tractable or being NP-hard. In the case when the variables are Boolean-valued this was done already in 1978 when Schaefer [Sch78] gave a full classification. Feder and Vardi conjectured in 1997 [FV98] that such a dichotomy between polynomial time solvability and NP-hardness holds also for general finite domains, and after a long sequence of partial results this conjecture was fully proven independently by Bulatov [Bul17] and Zhuk [Zhu20] in 2017.

A majority of CSPs are NP-hard and hence to allow for efficient algorithms some relaxation is needed. One such relaxation is in the form of approximation algorithms where we ask for an assignment that may not satisfy all constraints but satisfies a non-trivial number of constraints. For example, even if we cannot find an assignment that satisfies all the constraints, maybe we can find an assignment that satisfies 90% of the constraints using the promise that there is an assignment that satisfies all the constraints.

A different relaxation is obtained if we instead ask for an assignment that satisfies all constraints in a relaxed form. Such problems are called Promise Constraint Satisfaction Problems (PCSPs). One example of a PCSP is the classical problem of k-coloring a 3-colorable graph. The study of this type of problem is very old, but the concept of PCSPs was only recently formally introduced by Austrin, Guruswami and Håstad in their study of the "$(2+\epsilon)$-SAT" problem [AGH17]. In $\mathrm{PCSP}(A, B)$, you are given instances of $\mathrm{CSP}(B)$ with the promise that the instance has a solution if interpreted as a $\mathrm{CSP}(A)$ instance. In the case of k-coloring a 3-colorable graph, $\mathrm{CSP}(A)$ is the 3-coloring problem, and $\mathrm{CSP}(B)$ is the k-coloring problem. Note that PCSPs generalize CSPs, since $\mathrm{PCSP}(A, A)$ is the same as $\mathrm{CSP}(A)$. This implies that understanding PCSPs is at least as difficult as that of CSP, but we also expect that some tools useful for understanding CSPs will be useful for the study of PCSPs.

In the study of CSPs and the resolution of the Feder-Vardi conjecture, algebraic methods play a crucial role, and one of the key concepts here is that of polymorphisms, introduced by Jeavons [JCG97, Jea98]. A polymorphism is a function that given multiple solutions to a problem, combines them in some way to form another solution. Polymorphisms are key to understanding the complexity of CSPs, and the set of polymorphisms of a CSP determines its complexity. In other words if two CSPs have the same set of polymorphisms, they are of equal computational complexity. In general, the fewer polymorphisms a CSP has, the higher is its computational complexity.

Polymorphisms can be extended to PCSPs and the requirement now is that, given a set of solutions that satisfy the constraints of A, produce a solution to B. It turns out that, also in this case, two problems with the same set of polymorphisms are of equal complexity [BG21]. This gives a possible avenue of attack to understand the complexity of $\mathrm{PCSP}(A, B)$, but it is only a starting point. It is known that the existence of some specific polymorphisms immediately gives efficient algorithms and that establishing some specific properties of the set of polymorphisms yields NP-hardness, but the known conditions

are far from complementary. On top of this, given A and B, it is many times difficult to get a grip on the corresponding set of possible polymorphisms. Given these difficulties, the study of PCSPs is still at an early stage and for the rest of this paper we focus on the Boolean case which already is quite challenging. In other words we assume that the inputs to both A and B are Boolean. Even in this simple case our understanding is rather limited. There is dichotomy when the predicates are symmetric [BG21, FKOS19]. Brandts and Živný [BŽ22] study the question when A is a close to the t-in-k. The general understanding of even Boolean PCSPs is however very far from complete and a potential analogue of Schaefer's Theorem for this setting remains out of reach.

In the rest of this paper we restrict our attention to Boolean PCSPs where the constraint language is given by a single pair of predicates (rather than a general collection of predicates). Thus from now on, whenever we discuss $\mathrm{PCSP}(A, B)$, A and B are understood to be predicates on Boolean strings of some fixed arity. That said, our general tractability and hardness results are all polymorphism-based and thus in principle they also apply to general constraint languages.

In the setting of approximation algorithms, Austrin and Håstad [AH13] introduced a notion of useless predicates. We do not give the exact definition here, but the general idea is that a predicate is useless if, even given a promise that there is an assignment that satisfies almost all the constraints, no non-trivial solution can be found in the approximating sense even when the algorithm is allowed to replace the predicate by any other predicate of its choosing. It turns out that in this situation there is a simple and elegant characterization: assuming the Unique Games Conjecture, a predicate is useless if and only if the set of accepted strings can support a pairwise independent distribution.

We introduce and study an analogous notion of "promise-uselessness" for PCSPs. In particular, let us say that a predicate A is promise-useful if and only if there exists a (non-trivial) B such that $\mathrm{PCSP}(A, B)$ is solvable in polynomial time. Otherwise, A is said to be promise-useless. Apart from being natural in its own right, we hope this notion will be a helpful focus for further exploration and classification of the complex landscape of Boolean PCSPs.

Note that by this definition, if $\mathrm{CSP}(A)$ is tractable, then A is promise-useful. But even if $\mathrm{CSP}(A)$ is NP-hard, it is still possible that A is promise-useful. For example, 1-in-3-SAT is NP-hard by Schaefer's characterization, but 1-in-3-SAT is promise-useful since $\mathrm{PCSP}(\text{1-in-3-SAT}, \text{3-XOR})$ is tractable (since CSP(3-XOR) is solved by Gaussian elimination over $\mathbf{F}_2$). In general, for any tractable $\mathrm{CSP}(B)$, any predicate A which implies B is promise-useful. A more interesting example is (≥ 2)-in-4-SAT (where the objective is to find an assignment satisfying at least 2 out of the 4 literals in each clause): it is NP-hard and does not imply any tractable predicate. However it is still promise-useful, because $\mathrm{PCSP}((\geq 2)\text{-in-4-SAT}, \text{4-SAT})$ is tractable, as shown in the characterization of the "$(2+\epsilon)$-SAT" problem [AGH17]. A basic example of a promise-useless predicate is 3-SAT, since the only non-trivial possible B is 3-SAT, and $\mathrm{PCSP}(\text{3-SAT}, \text{3-SAT}) = \mathrm{CSP}(\text{3-SAT})$ which is NP-hard.

Even for a constraint language defined by a single fixed predicate, it is natural in the Boolean setting to consider the so-called *folded* setting, where we allow negation of variables[1]. Another natural variant is the so-called *idempotent* setting, where we allow fixing some variables to constants[2]. Because of this, the notion of promise-useful/promise-useless, even in the single-predicate Boolean case, comes in four different flavors.

In this work, our focus is the folded case. It turns out that whether or not we work

[1]This can be phrased in CSP terminology as including the "not-equal" predicate in the template.
[2]In CSP terminology this is equivalent to including all non-constant unary predicates in the template.

in the folded idempotent case or folded non-idempotent case does not make a significant difference. For the rest of the introduction, unless explicitly mentioned otherwise, promise-uselessness is understood to be the folded idempotent Boolean case, a variant we denote by fiPCSP. In this setting, it is not difficult to see that understanding which predicates are promise-useful boils down to understanding for which predicates A the fiPCSP(A, OR) problem is tractable (we establish this connection formally in Section 3). We refer to this family of problems – where we are given a k-SAT instance with an additional promise that there is a satisfying assignment which satisfies a stronger predicate A for each clause – as *Promise-SAT* problems. While the name is new, Promise-SAT was essentially the main focus of early work on general PCSPs [AGH17]. In particular, its complexity for symmetric A is well-understood from more general results [BG21, FKOS19], but for arbitrary A it has as far as we are aware not been studied in detail before.

1.1 Results

We derive general conditions to determine whether a predicate A is promise-useful or promise-useless (assuming P $\neq$ NP). We apply these conditions to all predicates of arity up to five and it turns out that we can successfully characterize the promise-usefulness of most predicates. The raw count of the classification is given in Table 1 below. For $k = 5$, there are 59 different (non-equivalent) predicates where we have been unable to determine whether the predicate is promise-useful or promise-useless. The numbers in this table apply both for the idempotent and non-idempotent settings, as both our algorithms and hardness results are agnostic to this property.

Table 1: Summary of classification of promise-usefulness and promise-uselessness of predicates A of arity up to 5.

	Total	Useful	Useless	Unknown
$k = 2$	4	4	0	0
$k = 3$	20	16	4	0
$k = 4$	400	230	170	0
$k = 5$	1 228 156	156 135	1 071 962	59

On the algorithmic side, our general condition for usefulness (yielding the results summarized in this table) is particularly simple and natural. Somewhat informally, we establish (see Theorem 4.12 for a formal statement) that A is promise-useful in the folded setting (both in idempotent and non-idempotent settings) if either

1. There is a weighted majority which is satisfied by all satisfying assignments, or
2. There is a non-trivial parity constraint which is satisfied by all satisfying assignments.

This is more or less a direct consequence of well-known tractability results for Boolean PCSPs, but what is more interesting is that despite our best efforts we have been unable to find any evidence of additional tractable cases, raising the tantalizing (if perhaps unlikely) possibility that this simple condition might exactly characterize promise-usefulness (in the folded Boolean setting).

As mentioned in the preceding section, understanding promise-usefulness boils down to understanding the complexity of the Promise-SAT problem, fiPCSP(A, OR). We consider this an interesting problem in its own right, and in fact most of the work in this

paper is devoted to it (yielding the results for promise-usefulness as a byproduct). Again we derive general tractability and hardness conditions, and apply them to all predicates of arity up to five. The results are summarized in Table 2 below (the total number of predicates is larger than for usefulness, because there are fewer direct equivalences between predicates). For arity five we classify all except 189 out of the roughly 18.6 million genuinely different predicates. Note that for $k = 2$, fiPCSP(A, OR) is always tractable since 2-SAT is tractable, so the choice of A does not matter.

Table 2: Summary of classification of complexity of fiPCSP(A, OR) for A of arity up to 5.

	Total	Tractable	NP-hard	Unknown
$k = 2$	5	5	0	0
$k = 3$	39	33	6	0
$k = 4$	1 991	956	1 035	0
$k = 5$	18 666 623	1 290 862	17 375 572	189

From our classification for small arities it is tempting to guess that as the arity increases most predicates tend to be promise-useless. This is indeed true, and we prove the following asymptotic version of this fact, showing that even exponentially sparse predicates are useless.

Theorem 1.1 (Informal version of Corollary 7.6)**.** *For any $s = \omega(k \cdot 2^{5k/6})$, a uniformly random k-ary Boolean predicate with s satisfying assignments is promise-useless with high probability, in all four settings (folded/non-folded and idempotent/non-idempotent).*

The bound on s in this result is likely far from tight and we suspect that this result is true with a much smaller bound on s, perhaps even polynomial, though we have not been able to prove this and new methods would be needed to push s down to $2^{o(k)}$. Supplementing this result, we show (see Theorem 7.16 for precise statement) that the BLP+AIP algorithm (which, as described in the next section, is the source of our tractability results) stops working once a random predicate of arity k accepts $s = \omega(k)$ strings out the 2^k possible inputs. This leaves a large set of predicates that we are unable to classify.

1.2 Overview of Techniques

A surprisingly powerful algorithm for establishing that PCSP(A, OR) is tractable is the basic LP + affine IP algorithm of Brakensiek et al. [BGWŽ20]. This algorithm is applicable assuming that the PCSP admits block-symmetric polymorphisms of arbitrarily large arity. While for larger domains there are known examples of tractable PCSPs that require different algorithms, no such examples are (currently) known for Boolean PCSPs, and BLP+AIP is the only algorithm used in the current paper to derive positive results.

The specific polymorphisms we use with BLP+AIP are partly the expected ones, but it has a slight twist. Unsurprisingly we have the three standard polymorphism families majority, odd parity and alternating threshold, but it turns out that in some cases the tractability of fiPCSP(A, OR) needs idempotent versions of minority and even parity which, as far as we are aware, have not been employed before. It is not difficult to give necessary and sufficient conditions in terms of A for each of these families to be applicable. On top of being explicit these conditions also turn out to be easy to check by computer

for a given predicate A. Thus the we do not introduce any essentially new techniques to establish tractability, we simply make good use of existing techniques.

On the other hand when it comes to establishing that A is promise-useful, the algorithmic condition becomes even simpler as described in the preceding section. While for fiPCSP(A, OR) we need all five polymorphism families described above (majority, odd parity, alternating threshold, idempotized minority, and idempotized even parity), it turns out that out of these five only majority and odd parity are needed to establish promise-usefulness: if any of the other three families can be used to establish promise-usefulness, then either majority or odd parity can as well.

Our hardness results are obtained through the study of the minion of polymorphisms of fiPCSP(A, OR). In a small extension of previous results we show that if all such polymorphisms have a small fixing assignment then the problem is NP-hard. In other words, for every polymorphism, it is possible to fix a constant number of inputs in a way such that the output is determined. This is the basic criterion we use for hardness but we develop a number of distinct, but similar, conditions to establish that a given polymorphism minion has this property.

These conditions are based on establishing properties shared by all polymorphisms and that some particular (low arity) functions are not polymorphisms. A toy example would be to show that all polymorphisms of fiPCSP(A, OR) are monotone and that there is no polymorphism f of arity $t+1$ such that if its first input is fixed to 0 then the restricted function becomes an AND of t variables. It is then not difficult to prove that all polymorphisms of fiPCSP(A, OR) have a fixing set of size $t-1$ from which it follows that fiPCSP(A, OR) is NP-hard by known theorems. In reality we use several more complicated properties of the polymorphism minion, culminating in four different conditions (Theorems 5.16, 5.17, 5.18 and 5.21). Even for a given A these conditions are most of the time too cumbersome to check by hand, but sufficiently simple that they can be checked relatively quickly by a computer.

As described in the preceding section, these diverse techniques are (somewhat surprisingly) sufficient to completely understand the complexity of fiPCSP(A, OR) for all predicates up to arity four – the small fixing assignment condition exactly complements the block-symmetry condition of the BLP+AIP algorithm for these predicates. For the unclassified predicates of arity five, we know that they are not solved by BLP+AIP, but we do not know whether or not they satisfy the small fixing assignment condition. In other words it is conceivable that the "gap" in our knowledge here is due to the concrete conditions (Theorems 5.16, 5.17, 5.18 and 5.21) only being sufficient for small fixing assignments, and not necessary.

1.3 Organization

Section 2 covers notation and background material used throughout the paper. Then in Section 3 we formally define and give some initial observations on promise-usefulness. Following this we analyze tractability in Section 4, applying the BLP+AIP algorithm and discussing the five families of block-symmetric polymorphisms that appear. We then turn to hardness in Section 5 and give the four different conditions for small fixing assignments, Theorems 5.16, 5.17, 5.18 and 5.21. In Section 6 we apply these methods to analyze the tractability and hardness of all predicates of arity up to five. The asymptotic bounds for large arities are established in Section 7, and we give some general conclusions and discuss open problems in Section 8.

2 Preliminaries

For a logical statement P, we use the Iverson notation where $[P]$ is defined to be 1 if P is true and 0 otherwise. For a positive integer n, $[n]$ denotes $\{1, \ldots, n\}$ (and in particular a generic binary string $x \in \{0,1\}^n$ is 1-indexed and written as $x_1 x_2 \ldots x_n$).

Throughout the paper, we use the following notation for binary strings. The exclusive or of two binary strings x and y (of equal length) is denoted by $x \oplus y$. For $b \in \{0,1\}$, b^ℓ denotes the all-b string of length ℓ. The number of ones in a binary string x is denoted as $w(x) = \sum x_i$. For a subset $S = \{j_1, \ldots, j_r\} \subseteq [k]$, x_S is the projection of x onto S, i.e., the binary string $(x_{j_1}, x_{j_2}, \ldots, x_{j_r})$ of length $|S| = r$. We let $\neg$ denote Boolean negation and extend it bit-strings by applying it component-wise. A Boolean function $f : \{0,1\}^\ell \to \{0,1\}$ is *folded* if $f(\neg x) = \neg f(x)$ for every x, and *idempotent* if $f(0^\ell) = 0$ and $f(1^\ell) = 1$. We often blur the distinction between sets and Boolean vectors and for a set $S \subseteq [n]$ we let $f(S)$ be f applied to the indicator vector of S (i.e., the vector $x \in \{0,1\}^n$ defined by $x_i = [i \in S]$).

We identify a *predicate* A of *arity* k with a subset of $\{0,1\}^k$ (the set of *accepting assignments* of A). In order to avoid trivial cases we do not allow A to be empty or to contain all strings of length k. On the other hand we do allow A to be independent of some of its coordinates and also that all strings in A share the same value of some coordinate.

For a k-ary predicate $A \subseteq \{0,1\}^k$ and a binary string $p \in \{0,1\}^k$, we define $A \oplus p = \{\, a \oplus p \,:\, a \in A \,\}$. In particular $A \oplus 1^k$ is the predicate formed by negating all accepting assignments of A.

We shall frequently be concerned with matrices where the columns are accepting assignments of a predicate. Given a predicate $A \subseteq \{0,1\}^k$ and an integer ℓ, let A^ℓ be the set of all $k \times \ell$ matrices M where each column is an accepting assignment of A. Its columns are denoted by $M^1, \ldots, M^\ell$, and its rows are denoted by $M_1, \ldots, M_k$. The entry at row i column j is denoted by M_i^j.

For a set $X \subseteq \mathbb{R}^k$ we denote by $K(X)$ the convex hull of X. The following standard theorem about separating convex sets by hyperplanes is useful for us.

Theorem 2.1. *Let K_1 and K_2 be two disjoint convex sets in $\mathbb{R}^k$ and suppose K_1 is closed. Then there exist real numbers such $c_1 \ldots c_k$ and b such that $\sum_{i=1}^k c_i x_i \geq b$ for any $x \in K_1$ while $\sum_{i=1}^k c_i x_i < b$ for any $x \in K_2$.*

2.1 CSPs, PCSPs, and Polymorphisms

Given a predicate A we can define the CSP(A) problem.

Definition 2.2. Let $A \subseteq \{0,1\}^k$ be a k-ary predicate. An instance $\mathcal{I} = (C, X)$ of CSP(A) consists of a set of variables $X = \{x_1, \ldots x_n\}$ and a set of constraints $C = \{c_1, \ldots, c_m\}$, where each constraint $c_i \in X^k$ is a k-tuple of variables. An assignment $\alpha : X \to \{0,1\}$ of values to the variables is a satisfying assignment to $\mathcal{I}$ if $\alpha(c_i) \in A$ for every constraint $c_1, \ldots c_m$. $\mathcal{I}$ is a Yes-instance if there exists a satisfying assignment to $\mathcal{I}$. Otherwise, $\mathcal{I}$ is a No-instance.

We, most of the time, apply predicates to literals and formally in this case we have $2n$ variables $X = \{x_1, \overline{x_1}, x_2, \overline{x_2}, \ldots, x_n, \overline{x_n}\}$, and an assignment $\alpha : X \to \{0,1\}$ where $\alpha(x_i) = \neg\alpha(\overline{x_i})$ for every i. We sometimes allow constants and in such a case we have a CSP(A) instance extended with a variable denoted x^b that always takes the value b.

For two k-ary predicates A and B such that A implies B (i.e., $A \subseteq B$), we can define PCSP(A, B).

Definition 2.3. Let $A, B \in \{0,1\}^k$ be k-ary predicates such that $A \subseteq B$. An instance $\mathcal{I} = (C, X)$ of the PCSP(A, B) problem consists of a CSP(A) instance, and the goal is to distinguish between:

Yes $\mathcal{I}$ is a satisfiable CSP(A) instance

No $\mathcal{I}$ is not even satisfiable when interpreted as a CSP(B) instance (i.e., there is no $\alpha : X \to \{0,1\}$ such that $\alpha(c) \in B$ for every constraint $c \in C$).

Note that CSPs are a special case of PCSPs are PCSP$(A, A) =$ CSP(A). Definition 2.3 describes the decision version of PCSP(A, B). There is also a search version of the problem, where we are given an instance that is promised to be a Yes instance, and the goal is to find a satisfying for the corresponding CSP(B) instance. Unlike basic CSPs, it is a major open problem whether the decision and search versions of PCSPs are equivalent.

In the definition of CSPs and PCSPs, the distinction of allowing or disallowing repetition (the same variable appearing multiple times in a clause) is unimportant since the two are polynomial time equivalent ([BG21], Section 6.6).

Let us make the useful and obvious observation that stronger promises cannot be worse than weaker promises.

Fact 2.4. Suppose $A \subset A' \subset B$, then PCSP(A', B) is not easier than PCSP(A, B). In particular if the latter is NP-hard so is the former and if the former is tractable so is the latter. Similarly if $A \subset B \subset B'$ then PCSP(A, B) is not easier than PCSP(A, B').

Let us proceed to define polymorphisms of PCSPs. These are functions f which, when given multiple solutions to an instance of CSP(A), return a solution to the corresponding CSP(B) instance. The formal definition is the following.

Definition 2.5. A Boolean function $f : \{0,1\}^\ell \to \{0,1\}$ is a *polymorphism* of PCSP(A, B) if and only if for all matrices $M \in A^\ell$, $f(M) \in B$. Here $f(M)$ denotes the column vector of length k obtained by evaluating f on each row of M, i.e., $f(M) = (f(M_1), f(M_2), \ldots, f(M_k))$. The set of polymorphisms of PCSP(A, B) is denoted by Pol(PCSP(A, B)).

The cases of allowing negations or fixed constants to the PCSP problem restrict the set of polymorphisms as follows. If we apply our predicates to literals, then all polymorphism must be folded and we denote the problem by fPCSP and if we also allow constants then all polymorphisms must be idempotent and we denote this class by fiPCSP. In other words.

- Pol(fPCSP(A, B)) $= \{ f \in$ Pol(PCSP(A, B)) $:$ f is folded $\}$
- Pol(fiPCSP(A, B)) $= \{ f \in$ Pol(PCSP(A, B)) $:$ f is folded *and* idempotent$\}$

If a Boolean function is not a polymorphisms of PCSP(A, B), then there must exist some obstruction $M \in A^\ell$ witnesses this fact.

Definition 2.6. Given a PCSP(A, B), $M \in A^\ell$ is called an *obstruction* for f if $f(M) \notin B$.

2.2 Minors and Minions

We start with a simple but useful definition.

Definition 2.7. Let $f : \{0,1\}^\ell \to \{0,1\}$ be a Boolean function. For any $\pi : [\ell] \to [\ell']$, the function $f_\pi : \{0,1\}^{\ell'} \to \{0,1\}$ defined by

$$f_\pi(x_1, \ldots, x_{\ell'}) = f(x_{\pi(1)}, \ldots x_{\pi(\ell)})$$

is called a *minor* of f.

One informal way to view this is that a minor is obtained by identifying some sets of variables. Note that it is not allowed to fix variables to constants.

Definition 2.8. A (Boolean function) *minion* $\mathcal{M}$ is a set of Boolean functions (not necessarily of the same arity) such that for every $f \in \mathcal{M}$, every minor f_π of f is also a member of $\mathcal{M}$.

It is a well-known and easy to prove fact that the set of polymorphisms of a PCSP forms a minion. A useful consequence of this is that once we have established that some small, constant size g is not a polymorphism we get a structure theorem on the set of all polymorphisms, because not containing g as a minor anywhere is indeed a severe restriction when f is of large arity. One general instantiation of this phenomenon that is of use for us is the following.

Definition 2.9. A family $\mathcal{F}$ of Boolean functions is *essentially minion-atomic* if for every minion $\mathcal{M}$ it holds that either $\mathcal{F} \subseteq \mathcal{M}$, or $\mathcal{F} \cap \mathcal{M}$ is finite.

In particular, if $\mathcal{F}$ essentially minion-atomic and some fixed $f \in \mathcal{F}$ is not a polymorphism of $\mathrm{PCSP}(A, B)$, then $\mathrm{PCSP}(A, B)$ only admits finitely many polymorphisms from $\mathcal{F}$.

2.3 Tractability Conditions for PCSPs

For PCSPs, one of the more remarkable algorithms that makes use of polymorphisms is the so-called basic LP relaxation + affine IP relaxation algorithm by Brakensiek et al. [BGWŽ20] which we from now on simply refer to as "BLP+AIP". It combines two commonly seen relaxations of the following Boolean problem.

Given an instance $\mathcal{I}$ of $\mathrm{CSP}(A)$, we naturally have a Boolean variable for each variable in $\mathcal{I}$, and we add one variable for each constraint and each element of A. This variable is supposed to be true if the constraints is satisfied by the indicated element of A. We require the sum of all Boolean variables corresponding to the same constraint in $\mathcal{I}$ to be 1 and the assignment to the variables and constraints to be consistent.

The basic LP relaxation of this Boolean problem is defined as the relaxation where the Boolean variables are relaxed to rational numbers in $[0, 1]$. The affine IP relaxation takes the Boolean variables and instead relaxes the Boolean variables to integer variables.

Brakensiek et al. showed that if $\mathrm{Pol}(\mathrm{PCSP}(A, B))$ contains infinitely many symmetric (or block-symmetric with increasing block sizes) polymorphisms, then the combination of these two relaxations can be used to solve (the decision version) of $\mathrm{PCSP}(A, B)$ [BGWŽ20]. It turns out that this algorithm is able to solve all tractable Boolean CSPs (Schaefer's dichotomy theorem). There are also natural generalization of this algorithm requiring consistency of larger (but still constant) size subsets of the variables. This potentially makes is more powerful but we do not know of a Boolean CSP solved by this generalization and not by the basic combination.

Definition 2.10. A function $f : \{0, 1\}^\ell \to \{0, 1\}$ is said *symmetric* if $f(x)$ depends only on $w(x)$, and *block-symmetric* with parameters b and m which are both constants independent of ℓ, if there exists a partition $S_1, S_2, \ldots, S_b$ of $[\ell]$, of sizes $\geq m$, such that $f(x)$ depends only on $w(x_{S_1}), \ldots, w(x_{S_b})$.

Remark 2.11. Any symmetric polymorphism is also a block symmetric polymorphism, with a single block. In the block-symmetric case, the parameter m is called the width of f. The case of two blocks is also of special interest to us and we use the notation (ℓ_1, ℓ_2)-block-symmetric polymorphism for such a function with block sizes ℓ_1 and ℓ_2.

The power of the BLP+AIP algorithm is then characterized by the following theorem.

Theorem 2.12. *[BGWŽ20, Theorem 5.1] The following three statements are equivalent:*

- *The* $\mathrm{PCSP}(A, B)$ *decision problem can be solved in polynomial time by the BLP+AIP algorithm.*
- $\mathrm{Pol}(\mathrm{PCSP}(A, B))$ *contains infinitely many block-symmetric polymorphisms of arbitrary large width.*
- *For every* $\ell \geq 1$, $\mathrm{Pol}(\mathrm{PCSP}(A, B))$ *contains an* $(\ell, \ell + 1)$*-block-symmetric polymorphism.*

The last condition is very useful as it can be efficiently checked for concrete A and B and small values of ℓ. The non-existence for any specific value of ℓ rules out the possibility to apply BLP+AIP.

Note that while BLP+AIP only in general solves the decision version of a PCSP, if the block-symmetric polymorphisms are explicit enough it is possible to solve the search version. In particular this is true in all applications of BLP+AIP to explicit predicates in the current paper.

2.4 Hardness Conditions for PCSPs

There are many ways that polymorphisms can be used to show that a PCSP is NP-hard. A general approach is to make use of dictator-like properties of the polymorphisms themselves to construct a reduction from gap label cover to PCSP. There are many possible notions of being "dictator-like" and the one we use is based on the following definition.

Definition 2.13. A function $f : \{0,1\}^\ell \to \{0,1\}$ has a *t-fixing set* if there exists a set $T \subseteq [\ell]$ of size t such that $f(x) = 1$ whenever $x_T = 1^t$. More generally f has a *t-fixing assignment* (T, α) if there exists $T \subseteq [\ell]$ of size t and a partial assignment $\alpha \in \{0,1\}^t$ such that $f(x) = 1$ whenever $x_T = \alpha$.

Remark 2.14. Since our focus in this paper is on folded functions f, it makes no difference in the notion of fixing assignments whether we also allow the function to be fixed to 0 (since we can simply negate α to fix the function to the opposite value).

Brakensiek and Guruswami [BG21, Theorem 5.3] showed that if all polymorphisms of a PCSP have small fixing sets then the PCSP is NP-hard. This can be naturally extended [BBKO21, Corollary 5.13] to a general setting where we can define a "choice function" $C(f)$ which for each polymorphism f identifies a small number of "relevant" coordinates, in a way that behaves consistently across minors (meaning that $\pi(C(f)) \cap C(f_\pi) \neq \emptyset$ for every minor f_π of f). This approach, which characterizes when the standard reductions from the basic Gap Label Cover problem is applicable, has subsequently been generalized to "layered choice" [BWŽ21] (corresponding to reductions from Layered Label Cover) and "injective layered choice" [BK24] (corresponding to reductions from Smooth Layered Label Cover). The hardness condition based on small fixing sets can naturally be relaxed to only require small fixing assignments.

Theorem 2.15. *If there exists a* t *such that every* $f \in \mathrm{Pol}(\mathrm{fPCSP}(A, B))$ *has a* t*-fixing assignment, then* $\mathrm{fPCSP}(A, B)$ *is NP-hard. Likewise, if there is a* t *such that every* $f \in \mathrm{Pol}(\mathrm{fiPCSP}(A, B))$ *has a* t*-fixing assignment, then* $\mathrm{fiPCSP}(A, B)$ *is NP-hard.*

Essentially this theorem was shown by Guruswami and Sandeep [GS20] though they only state it for rainbow coloring. Unlike the fixing set condition, Theorem 2.15 does not immediately follow from the basic Gap Label Cover problem but instead seems to require Smooth Label Cover. For completeness, we give a short proof of this result in Claim C.2 based on the injective choice condition of Banakh and Kozik [BK24].

The extension from fixing sets to fixing assignments is not difficult but it is needed for some of our results. There are many cases where $\mathrm{Pol}(\mathrm{fiPCSP}(A, B))$ contains functions without small fixing sets but where all functions have small fixing assignments.

2.5 Standard Boolean Functions

Let us recall various standard (and a few not so standard) families of Boolean functions that are relevant for us. We first recall three families of (block-)symmetric functions that have previously been successfully used in conjunction with the BLP+AIP algorithm.

Definition 2.16. For odd $\ell \geq 1$, the majority function $\mathsf{Maj}_\ell : \{0,1\}^\ell \to \{0,1\}$ is defined by $\mathsf{Maj}_\ell(x) = [w(x) \geq \ell/2]$.

Claim 2.17. *The family* $\mathbf{Maj} = \{\, \mathsf{Maj}_\ell \mid \ell \text{ odd} \,\}$ *is essentially minion-atomic.*

Proof. Let $\mathcal{M}$ be a minion and suppose $\mathsf{Maj}_\ell \notin \mathcal{M}$. For any odd $\ell' = d \cdot \ell + r$ ($d \geq 1$ and $0 \leq r < \ell$), consider the function

$$f(x_1, \ldots, x_\ell) = \left[\sum_{i=1}^{r} x_i + d \sum_{i=1}^{\ell} x_i \geq \ell'/2 \right]$$

Note that f is a minor of $\mathsf{Maj}_{\ell'}$ (obtained by identifying r groups of $(d+1)$ variables, and $\ell - r$ groups of d variables). Furthermore, if $d \geq \ell$ then $f = \mathsf{Maj}_\ell$. This shows that $\mathcal{M}$ cannot contain $\mathsf{Maj}_{\ell'}$ for any $\ell' \geq \ell^2$, i.e., $\mathcal{M}$ contains only finitely many majority functions and hence **Maj** is essentially minion-atomic. □

Definition 2.18. *Parity* of arity ℓ is the function $\mathsf{Par}_\ell : \{0,1\}^\ell \to \{0,1\}$, defined by

$$\mathsf{Par}_\ell(x) = \bigoplus_{i \in [\ell]} x_i.$$

Note that Par_ℓ is folded if and only if the arity ℓ is odd. The following claim is easy to verify.

Claim 2.19. *The family* $\mathbf{Par} = \{\, \mathsf{Par}_\ell \mid \ell \text{ odd} \,\}$ *is essentially minion-atomic.*

Definition 2.20. For odd $\ell \geq 1$, the *alternating threshold* function $\mathsf{AT}_\ell : \{0,1\}^\ell \to \{0,1\}$ is defined by

$$\mathsf{AT}_\ell(x) = \left[\sum_i (-1)^{1+i+x_i} > 0 \right]. \tag{1}$$

Again it easy to see that this family is minion-atomic and we leave the verification to the reader.

Claim 2.21. *The family* $\mathbf{AT} = \{\, \mathsf{AT}_\ell \mid \ell \text{ odd} \,\}$ *is essentially minion-atomic.*

For the hardness results, we are also interested in various other, mostly standard, functions and frequently need the fact that these are minion-atomic, which we record in the following simple claim.

Claim 2.22. *Let $\{f_\ell\}$ be one of the following function families:*

$$\begin{aligned}
\mathsf{AND}_\ell(x_1,\dots,x_\ell) &= x_1 \wedge x_2 \wedge \dots \wedge x_\ell \\
\mathsf{OR}_\ell(x_1,\dots,x_\ell) &= x_1 \vee x_2 \vee \dots \vee x_\ell \\
\mathsf{NAND}_\ell(x_1,\dots,x_\ell) &= \neg\,\mathsf{AND}_\ell(x_1,\dots,x_\ell) = \mathsf{OR}_\ell(\neg x_1,\dots,\neg x_\ell) \\
\mathsf{NOR}_\ell(x_1,\dots,x_\ell) &= \neg\,\mathsf{OR}_\ell(x_1,\dots,x_\ell) = \mathsf{AND}_\ell(\neg x_1,\dots,\neg x_\ell) \\
\mathsf{ANDNOR}_\ell(x_1,\dots,x_\ell) &= x_1 \wedge \mathsf{NOR}_{\ell-1}(x_2,\dots,x_\ell)
\end{aligned}$$

For any minion $\mathcal{M}$, if $f_\ell \notin \mathcal{M}$ for some $\ell \geq 2$ then $f_{\ell+1} \notin \mathcal{M}$. In particular, each of these five families of functions is essentially minion-atomic.

That this is true is not difficult to see as identifying two appropriate variables in $f_{\ell+1}$ results in f_ℓ.

3 Promise-usefulness

The main new notion in this paper is the concept of promise-usefulness (and promise-uselessness), formally defined as follows.

Definition 3.1. A predicate $\emptyset \neq A \subsetneq \{0,1\}^k$ is *promise-useful* if there exists a non-trivial relaxation $B \supseteq A$ such that $\mathrm{PCSP}(A,B)$ is solvable in polynomial time. Otherwise A is *promise-useless*.

By "non-trivial" we mean that $B \neq \{0,1\}^k$. In the non-folded case (without negated literals) we would also demand that B contains neither 0^k nor 1^k since otherwise every instance has a trivial satisfying assignment. Throughout the paper we assume that P $\neq$ NP, and whenever we make a claim that some predicate is promise-useful, this is always under this assumption.

To make it apparent which situation we are in (folded and/or idempotent), we use the terminology fPCSP-useful for the folded case and fiPCSP-useful for the folded idempotent case.

As a first step, let us note that promise-usefulness in the folded setting boils down to understanding the Promise-SAT problem.

Lemma 3.2. *The predicate A is* fPCSP*-useful (resp.* fiPCSP*-useful), if and only if there exists $b \notin A$ such that* $\mathrm{fPCSP}(A \oplus b, \mathsf{OR})$ *(resp.* $\mathrm{fiPCSP}(A \oplus b, \mathsf{OR})$*) is in P.*

Proof. Suppose A is promise-useful, i.e., there is a non-trivial B such that $\mathrm{fPCSP}(A,B)$ is tractable, and let b be an arbitrary assignment that does not belong to B. Then $\mathrm{fPCSP}(A, \{0,1\}^k \setminus \{b\})$ is also tractable by Fact 2.4. Furthermore, since we have negations, $\mathrm{fPCSP}(A, \{0,1\}^k \setminus \{b\})$ is equivalent with $\mathrm{fPCSP}(A \oplus b, \mathsf{OR})$. Conversely, if $\mathrm{fPCSP}(A \oplus b, \mathsf{OR})$ is tractable for some $b \notin A$ then $\mathrm{fPCSP}(A, \mathsf{OR} \oplus b)$ is tractable and witnesses that A is promise-useful. The proof is the same for the idempotent case. □

So in the folded case, the concept of promise-usefulness boils down to understanding the complexity of various Promise-SAT problems and we turn to this question in Section 4 and Section 5 below.

Remark 3.3. For the non-folded case, where we do not have negations, a lemma similar to Lemma 3.2 is true, but with the OR predicate replaced by the not-all-equal (NAE) predicate: A is promise-useful if and only if $\mathrm{PCSP}(A, \mathrm{NAE})$ is tractable. However, since our main focus in this paper is the folded case, we refrain from discussing this further.

Remark 3.4. One can define promise-usefulness as either an *adaptive* or *non-adaptive* notion. In the non-adaptive version (which is what Definition 3.1 defines), there must exist a fixed B depending only on A, such that A is promise-useful for B ($\mathrm{PCSP}(A, B)$ is tractable). In the adaptive version, an algorithm would be allowed to choose B also based on the given instance of $\mathrm{CSP}(A)$ – i.e., given as input a satisfiable $\mathrm{CSP}(A)$ instance, the goal becomes to find some non-trivial $B \supseteq A$ and satisfy the instance as a $\mathrm{CSP}(B)$ instance. In the non-folded setting this is equivalent with finding an assignment to the variables such that none of the input k-tuples is constant, i.e., a 2-coloring of the underlying hypergraph, and thus adaptive vs. non-adaptive are trivially equivalent here. In the folded setting, it is equivalent to finding an assignment to the variables such that not all k-tuples appear. If $\mathrm{fPCSP}(A, B)$ is NP-hard for all non-trivial B then A is fPCSP-useless also in this adaptive sense as we can concatenate the results of all hardness reductions on disjoint sets of variables. In the opposite direction, we are not aware of a proof that an adaptive useful algorithm must yield an efficient solution to $\mathrm{PCSP}(A, B)$ for a fixed B, although this seems intuitively likely.

3.1 The Non-Idempotent Case

As stated above we restrict attention to the folded setting and we study both $\mathrm{fiPCSP}(A, B)$ as well as $\mathrm{fPCSP}(A, B)$. By the preceding discussion, our primary interest is the case when $B = \mathsf{OR}$. As our main tool is to study polymorphisms, the following simple observation is useful, implying that it is sufficient to understand the idempotent polymorphisms.

Lemma 3.5. *Let A be a predicate that does not contain 0^k. If A contains 1^k, then all polymorphism of $\mathrm{fPCSP}(A, \mathsf{OR})$ are idempotent. If A does not contain 1^k then for every $f \in \mathrm{Pol}(\mathrm{fPCSP}(A, \mathsf{OR}))$ it holds that either f is idempotent or $\neg f$ is an idempotent polymorphism of $\mathrm{fPCSP}(A \oplus 1^k, \mathsf{OR})$.*

Proof. If A contains 1^k and f is a folded polymorphism then f must be idempotent to avoid that the all-ones matrix is an obstruction. In the second case if f is not idempotent then $\neg f$ is idempotent. As $\neg f(x) = f(\neg x) = f(x \oplus 1^k)$, $\neg f$ is a polymorphism of $\mathrm{fPCSP}(A \oplus 1^k, \mathsf{OR})$. □

Note in particular that for a general A not containing 1^k this implies:

1. $\mathrm{fPCSP}(A, \mathsf{OR})$ is tractable via the BLP+AIP algorithm if and only if $\mathrm{fiPCSP}(A, \mathsf{OR})$ *or* $\mathrm{fiPCSP}(A \oplus 1^k, \mathsf{OR})$ is tractable via the BLP+AIP algorithm.
2. $\mathrm{fPCSP}(A, \mathsf{OR})$ has small fixing assignments (i.e., is NP-hard via Theorem 2.15) if and only if both $\mathrm{fiPCSP}(A, \mathsf{OR})$ *and* $\mathrm{fiPCSP}(A \oplus 1^k, \mathsf{OR})$ have small fixing assignments.

It is an interesting question whether this holds true in general regardless of algorithm or NP-hardness proof used, in other words: *Is it the case that* $\mathrm{fPCSP}(A, \mathsf{OR})$ *is tractable if and only if* $\mathrm{fiPCSP}(A, \mathsf{OR})$ *or* $\mathrm{fiPCSP}(A \oplus 1^k, \mathsf{OR})$ *is tractable?*

For promise-usefulness, combining this observation with Lemma 3.2 we obtain the following conclusion.

Corollary 3.6. *A predicate A is* fPCSP*-useful via the BLP+AIP algorithm if and only if it is* fiPCSP*-useful via the BLP+AIP algorithm, and it is* fPCSP*-useless via small fixing assignments if and only if it is* fiPCSP*-useless via small fixing assignments.*

Thus when it comes to promise-usefulness, being in the idempotent setting (i.e., allowing fixed constants) does not make a difference with the current techniques we have for proving tractability and hardness.

4 Tractability of Promise-SAT

Based on our preliminary observations in Section 3, understanding promise-usefulness boils down to understanding the Promise-SAT problem, $\mathrm{fiPCSP}(A, \mathsf{OR})$. In this section we analyze conditions for this problem to be tractable.

4.1 Identifying Families of Block-symmetric Polymorphisms

To identify tractable cases of $\mathrm{fiPCSP}(A, \mathsf{OR})$, we use the BLP+AIP algorithm, characterized in Theorem 2.12. The requirement for this algorithm is that $\mathrm{fiPCSP}(A, \mathsf{OR})$ contains infinitely many block-symmetric polymorphisms of arbitrary large arity. By Theorem 2.12 this is equivalent to the existence of block-symmetric polymorphisms with two blocks, one with size ℓ and the other with size $\ell + 1$ for each value of ℓ.

For small values of k and ℓ the condition that such polymorphisms exist can be checked by computer. There are then two outcomes. If there is no solution for some ℓ then $\mathrm{fiPCSP}(A, \mathsf{OR})$ cannot be solved using the BLP+AIP algorithm. On the other hand, if a solution is found this can help us identify block-symmetric polymorphisms of general interest. Performing such experiments, we have identified five different families of block-symmetric polymorphisms, where for each family $\mathcal{F}$, there exists a predicate A such that $\mathrm{Pol}(\mathrm{fiPCSP}(A, \mathsf{OR}))$ contains infinitely many polymorphisms from $\mathcal{F}$, but only finitely many from the other families.

Three of these families are the standard idempotent polymorphisms **Maj**, **Par** and **AT** defined in Section 2.5. The other two are non-standard, closely related to the negated functions minority $\overline{\mathsf{Maj}}_\ell(x) = \neg\,\mathsf{Maj}_\ell(x)$ and even parity $\overline{\mathsf{Par}}_\ell(x) = \neg\,\mathsf{Par}_\ell(x)$. These two functions are not idempotent and thus are not polymorphisms of $\mathrm{fiPCSP}(A, \mathsf{OR})$, but we can simply change the values on the constant strings to make them idempotent.

For a function $f : \{0,1\}^\ell \to \{0,1\}$, we denote by $\mathsf{id}f : \{0,1\}^\ell \to \{0,1\}$ the *idempotized* function

$$\mathsf{id}f(x) = \begin{cases} 0 & \text{if } x = 0^\ell \\ 1 & \text{if } x = 1^\ell \\ f(x) & \text{otherwise} \end{cases}$$

It is easy to see that a minion-atomic family of functions remains minion-atomic under negation and the idempotization operation.

Claim 4.1. *Let $\mathcal{F}$ be a family of Boolean functions. If $\mathcal{F}$ is essentially minion-atomic, then $\overline{\mathcal{F}} = \{\, \neg f \mid f \in \mathcal{F} \,\}$ is essentially minion-atomic. If additionally all functions in $\mathcal{F}$ are anti-idempotent (i.e., $\neg f$ is idempotent) then $\mathsf{id}\mathcal{F} = \{\, \mathsf{id}f \mid f \in \mathcal{F} \,\}$ is also essentially minion-atomic.*

Proof. It is easy to see that $\neg g$ is a minor of $\neg f$ if and only if g is a minor of f. This gives the first result. The second statement follows from the similar statement that $\mathsf{id}g$ is a minor of $\mathsf{id}f$ if and only if g is a minor of f. □

We then consider the families $\mathsf{id}\overline{\mathbf{Maj}} = \{\, \mathsf{id}\overline{\mathsf{Maj}}_\ell \mid \ell \text{ odd} \,\}$ and $\mathsf{id}\overline{\mathbf{Par}} = \{\, \mathsf{id}\overline{\mathsf{Par}}_\ell \mid \ell \text{ odd} \,\}$ of idempotized minority and idempotized parity of odd arities. Together with the aforementioned **Maj**, **Par**, and **AT** families, these make up our quintet of block-symmetric polymorphism families and we have the following result.

Lemma 4.2. *Consider the following five families of idempotent (block-)symmetric functions: majority (**Maj**), odd parity (**Par**), alternating threshold (**AT**), idempotized minority ($\mathsf{id}\overline{\mathbf{Maj}}$), and idempotized even parity ($\mathsf{id}\overline{\mathbf{Par}}$). For each of these families $\mathcal{F}$, there exists*

a predicate A such that $\mathrm{fiPCSP}(A, \mathsf{OR})$ *admits infinitely many polymorphisms from $\mathcal{F}$, but only finitely many from the other four families.*

1. $A = \{\mathtt{01}, \mathtt{10}, \mathtt{11}\}$ *is an example that only admits* **Maj**.
2. $A = \{\mathtt{001}, \mathtt{010}, \mathtt{100}, \mathtt{111}\}$ *is an example that only admits* **Par**.
3. $A = \{\mathtt{00011}, \mathtt{00101}, \mathtt{00110}, \mathtt{01000}, \mathtt{10000}\}$ *is an example that only admits* **AT**.
4. $A = \{\mathtt{0011}, \mathtt{0100}, \mathtt{0110}, \mathtt{1000}, \mathtt{1001}\}$ *is an example that only admits* $\mathsf{id}\overline{\mathbf{Maj}}$.
5. $A = \{\mathtt{00111}, \mathtt{01010}, \mathtt{01101}, \mathtt{10000}, \mathtt{10011}\}$ *is an example that only admits* $\mathsf{id}\overline{\mathbf{Par}}$,

Before proceeding with the proof, which is given in Section 4.4, we need characterizations of what it means for $\mathrm{fiPCSP}(A, \mathsf{OR})$ to admit infinitely many functions from each of these families. The conditions in the following sections are stated in terms of $\mathrm{fPCSP}(A, \mathsf{OR})$ admitting infinitely many polymorphisms from one of the families, but note that this is equivalent with $\mathrm{fiPCSP}(A, \mathsf{OR})$ doing so, since fiPCSP has all idempotent polymorphisms of fPCSP and no others.

4.2 Conditions for Maj, Par, and AT

We start by establishing necessary and sufficient conditions for the existence of infinitely many polymorphisms from the standard families **Maj**, **Par**, and **AT**. Recall that $K(A)$ denotes the convex hull of A. The proofs of the following three standard lemmas can be found in Appendix D.

The existence of majority as a polymorphism of arbitrarily large odd arities can be expressed as a linear program based on the following lemma.

Lemma 4.3. *The following statements are equivalent:*

1. $\mathrm{Pol}(\mathrm{fPCSP}(A, \mathsf{OR}))$ *contains infinitely many polymorphisms from* **Maj**.
2. $[0, 1/2)^k \cap K(A) = \emptyset$.
3. *There exists integers $c_1, \dots c_k \geq 0$ such that $\sum_{j=1}^{k} c_j a_j \geq \sum_{j=1}^{k} c_j/2 > 0$ for all $a \in A$.*

The test for odd parity is based on solving an affine equation modulo two.

Lemma 4.4. *The following statements are equivalent:*

1. $\mathrm{Pol}(\mathrm{fPCSP}(A, \mathsf{OR}))$ *contains infinitely many polymorphisms from* **Par**.
2. *For every odd sized subset B of A, $\bigoplus_{s \in B} s \neq 0^k$.*
3. *There exists a non-empty subset $\beta \subseteq [k]$, such that $\bigoplus_{i \in \beta} a_i = 1$ for all $a \in A$.*

Finally, for alternating threshold, the characterization corresponds to all accepting assignments of A satisfying some linear equation over the integers.

Lemma 4.5. *The following statements are equivalent:*

1. $\mathrm{Pol}(\mathrm{fPCSP}(A, \mathsf{OR}))$ *contains infinitely many polymorphisms from* **AT**.
2. $\{x - y : x, y \in K(A)\} \cap (-\infty, 0)^k = \emptyset$.
3. *There exists integers $c_1, \dots c_k \geq 0$, not all 0, such that $\sum_{j=1}^{k} c_j a_j$ takes the same value for all $a \in A$.*

4.3 Conditions for $\mathsf{id}\overline{\mathbf{Maj}}$ and $\mathsf{id}\overline{\mathbf{Par}}$

Let us now turn to necessary and sufficient conditions for when $\mathrm{Pol}(\mathrm{fPCSP}(A, \mathsf{OR}))$ contains infinitely many polymorphisms of either of the two non-standard families $\mathsf{id}\overline{\mathbf{Maj}}$ and $\mathsf{id}\overline{\mathbf{Par}}$. These conditions are formulated in terms of $\mathrm{Pol}(\mathrm{fPCSP}(A, \mathsf{OR}))$ containing infinitely many polymorphisms from $\overline{\mathbf{Maj}}$ and $\overline{\mathbf{Par}}$, respectively (and Lemmas 4.3 and 4.4 can easily be modified to give efficient tests for verifying these containments). Throughout this section we frequently use the fact that $\mathsf{id}\overline{\mathbf{Maj}}$ and $\mathsf{id}\overline{\mathbf{Par}}$ are essentially minion-atomic (Claim 4.1). In other words, having only finitely many polymorphisms from e.g. $\mathsf{id}\overline{\mathbf{Maj}}$ is equivalent to not having $\mathsf{id}\overline{\mathsf{Maj}}_\ell$ for some odd $\ell > 0$.

For a predicate $A \subseteq \{0,1\}^k$ and $b \in \{0,1\}$, we say that A has a *forced* 1*-bit* if there is an i such that $a_i = 1$ for all $a \in A$. Note that whenever A has a forced 1-bit, $\mathrm{Pol}(\mathrm{fPCSP}(A, \mathsf{OR}))$ contains *all* idempotent odd functions. We first show that, for $\mathrm{fPCSP}(A\,\mathsf{OR})$, admitting $\mathsf{id}\overline{\mathbf{Maj}}$ or $\mathsf{id}\overline{\mathbf{Par}}$ implies admitting their non-idempotized counterparts, except in the trivial case when A has a forced 1-bit.

Lemma 4.6. *Suppose $A \subseteq \{0,1\}^k$ does not have a forced 1-bit. If $\mathrm{fPCSP}(A, \mathsf{OR})$ admits an infinite number of polymorphisms from $\mathsf{id}\overline{\mathbf{Maj}}$ then it also admits an infinite number of polymorphisms from $\overline{\mathbf{Maj}}$.*

Proof. Since A does not have a forced 1-bit, there are k (not necessarily distinct) strings $x_1 \ldots, x_k \in A$ such that x_i has a 0 in position i.

Suppose there is an obstruction matrix $M \in A^\ell$ showing that $\overline{\mathsf{Maj}}_\ell$ is not a polymorphism of $\mathrm{fPCSP}(A, B)$ for some (odd) ℓ. Thus in every row of M there are at least $(\ell+1)/2$ ones. Create a new matrix M' with $\ell' = (k+1)\ell + k$ columns by taking $(k+1)$ copies of each column, and adding the k columns $x_1, \ldots, x_k$. Now M' has at least $(k+1)(\ell+1)/2 = (\ell'+1)/2$ ones in every row, and no row is identically 1. Thus this is an obstruction that shows that $\mathsf{id}\overline{\mathsf{Maj}}_{\ell'}$ is not a polymorphism and the lemma now follows by the fact that both families are essentially minion-atomic. □

By a similar reasoning we establish the same fact for $\mathsf{id}\overline{\mathbf{Par}}$ and $\overline{\mathbf{Par}}$.

Lemma 4.7. *Suppose $A \subseteq \{0,1\}^k$ does not have a forced 1-bit. If $\mathrm{fPCSP}(A, \mathsf{OR})$ admits an infinite number of polymorphisms from $\mathsf{id}\overline{\mathbf{Par}}$ then it also admits an infinite number of polymorphisms from $\overline{\mathbf{Par}}$.*

Proof sketch. We proceed as in the proof of Lemma 4.6, but when creating M' we do not make any copies of the existing rows and instead only add two copies each of $x_1, \ldots, x_k$. This maintains parity of every row while making sure no row is all-ones (and no row of M was all-zeros since M was an obstruction for $\overline{\mathsf{Par}}_\ell$ being a polymorphism of $\mathrm{fPCSP}(A, \mathsf{OR})$), ensuring $\mathsf{id}\overline{\mathsf{Par}}$ and $\overline{\mathsf{Par}}$ behave the same on M'. □

In what follows, we use the notation A^0_S to denote the set $\{\, a_S \mid a \in A \text{ such that } a_{\overline{S}} = 0^{|\overline{S}|} \,\}$, where $A \subseteq \{0,1\}^k$ is a k-ary predicate and $S \subseteq [k]$. This makes A^0_S a kind of projection which only keeps accepting assignments $a \in A$ such that $a_i = 0$ for all $i \notin S$.

In the other direction, we have the following general property.

Lemma 4.8. *Let $\mathcal{F}$ be an essentially minion-atomic infinite family of idempotent Boolean functions. If $\mathrm{fPCSP}(A, \mathsf{OR})$ admits only finitely many polymorphisms from $\mathsf{id}\overline{\mathcal{F}}$ then there exists a subset $S \subseteq [k]$ such that (i) A^0_S is non-empty and does not have a forced 1-bit, and (ii) $\mathrm{fPCSP}(A^0_S, \mathsf{OR})$ admits only finitely many polymorphisms from $\overline{\mathcal{F}}$.*

Proof. Suppose that Pol(fPCSP(A, OR)) contains only finitely many polymorphisms from $\mathsf{id}\overline{\mathcal{F}}$. Let M be an obstruction for $\mathsf{id}\overline{f}$ for some $f \in \mathcal{F}$. Note that since M is an obstruction for an idempotent polymorphism, no row in M is all-ones. Let S be the set of rows that are not equal to 0^ℓ, $k' = |S|$ and let M' denote the $k' \times \ell$ sub-matrix of M given by the rows of S. The predicate A^0_S is clearly not empty since the columns of M' come from A^0_S. And since M' has no all-ones row, A^0_S cannot have a forced 1-bit, establishing property (i).

By construction M' has no constant rows and hence $\overline{f}$ and $\mathsf{id}\overline{f}$ behave the same on M'. Thus M' is also an obstruction of $\overline{f}$ for Pol(fiPCSP(A^0_S, OR)), which together with $\overline{\mathcal{F}}$ being minion-atomic establishes property (ii). □

With these pieces in place let us formulate the characterization for $\mathsf{id}\overline{\mathbf{Maj}}$.

Lemma 4.9. fPCSP(A, OR) *admits only finitely many polymorphisms from* $\mathsf{id}\overline{\mathbf{Maj}}$ *if and only if there exists a subset* $S \subseteq [k]$ *such that* A^0_S *is non-empty and does not have a forced* 1*-bit, and* fPCSP(A^0_S, OR) *admits only finitely many polymorphisms from* $\overline{\mathbf{Maj}}$.

Proof. The forward direction is Lemma 4.8 with $\mathcal{F} = \mathbf{Maj}$. For the other direction, suppose that for some $S \subseteq [k]$, A^0_S is non-empty with no forced 1-bit and fPCSP(A^0_S, OR) admits only finitely many polymorphisms from $\overline{\mathbf{Maj}}$. By Lemma 4.6, fPCSP(A^0_S, OR) admits also only finitely many polymorphisms from $\mathsf{id}\overline{\mathbf{Maj}}$, but since Pol(fPCSP($A^0_S$, OR)) $\supseteq$ Pol(fPCSP(A, OR)), the latter then also contains only finitely many functions from $\mathsf{id}\overline{\mathbf{Maj}}$. □

The characterization for $\mathsf{id}\overline{\mathbf{Par}}$ is analogous.

Lemma 4.10. fPCSP(A, OR) *admits only finitely many polymorphisms from* $\mathsf{id}\overline{\mathbf{Par}}$ *if and only if there exists a subset* $S \subseteq [k]$ *such that* A^0_S *is non-empty and does not have a forced* 1*-bit, and* fPCSP(A^0_S, OR) *admits only finitely many polymorphisms from* $\overline{\mathbf{Par}}$.

Proof sketch. The proof is identical to the proof of Lemma 4.9, with Maj replaced by Par and the invocation of Lemma 4.6 replaced by Lemma 4.7. □

4.4 Proof of Lemma 4.2

In this section we establish our main tractability lemma for fiPCSPs, restated here for convenience.

Lemma 4.2. *Consider the following five families of idempotent (block-)symmetric functions: majority (*$\mathbf{Maj}$*), odd parity (*$\mathbf{Par}$*), alternating threshold (*$\mathbf{AT}$*), idempotized minority (*$\mathsf{id}\overline{\mathbf{Maj}}$*), and idempotized even parity (*$\mathsf{id}\overline{\mathbf{Par}}$*). For each of these families* $\mathcal{F}$*, there exists a predicate* A *such that* fiPCSP(A, OR) *admits infinitely many polymorphisms from* $\mathcal{F}$*, but only finitely many from the other four families.*

1. $A = \{01, 10, 11\}$ *is an example that only admits* $\mathbf{Maj}$.
2. $A = \{001, 010, 100, 111\}$ *is an example that only admits* $\mathbf{Par}$.
3. $A = \{00011, 00101, 00110, 01000, 10000\}$ *is an example that only admits* $\mathbf{AT}$.
4. $A = \{0011, 0100, 0110, 1000, 1001\}$ *is an example that only admits* $\mathsf{id}\overline{\mathbf{Maj}}$.
5. $A = \{00111, 01010, 01101, 10000, 10011\}$ *is an example that only admits* $\mathsf{id}\overline{\mathbf{Par}}$,

Proof of Lemma 4.2. We verify the five cases one by one. Note that by Claims 2.17, 2.19, 2.21 and 4.1, the five families of functions are all essentially minion-atomic and hence to prove that we do not have infinitely many from one of the families, it suffices to exhibit a single obstruction for that family.

1. $A = \{\mathtt{01}, \mathtt{10}, \mathtt{11}\}$. In this case we have $a_1 + a_2 \geq 1$ for all $a \in A$, so by Lemma 4.3 we get infinitely many polymorphisms from **Maj**. Obstructions for the other families are:
$$\mathsf{Par}_3 : \begin{pmatrix}011\\101\end{pmatrix}, \mathsf{AT}_3 : \begin{pmatrix}011\\110\end{pmatrix}, \mathsf{id}\overline{\mathsf{Maj}}_3 : \begin{pmatrix}011\\101\end{pmatrix}, \mathsf{id}\overline{\mathsf{Par}}_5 : \begin{pmatrix}00111\\11001\end{pmatrix}.$$

2. $A = \{\mathtt{001}, \mathtt{010}, \mathtt{100}, \mathtt{111}\}$. In this case we have $a_1 \oplus a_2 \oplus a_3 = 1$ for all $a \in A$, so by Lemma 4.4 we get infinitely many polymorphisms from **Par**. Obstructions for the other families are:
$$\mathsf{Maj}_3 : \begin{pmatrix}001\\010\\100\end{pmatrix}, \mathsf{AT}_3 : \begin{pmatrix}011\\010\\110\end{pmatrix}, \mathsf{id}\overline{\mathsf{Maj}}_5 : \begin{pmatrix}00111\\01111\\10111\end{pmatrix}, \mathsf{id}\overline{\mathsf{Par}}_3 : \begin{pmatrix}001\\010\\100\end{pmatrix}.$$

3. $A = \{\mathtt{00011}, \mathtt{00101}, \mathtt{00110}, \mathtt{01000}, \mathtt{10000}\}$. In this case we have $2a_1 + 2a_2 + a_3 + a_4 + a_5 = 2$ for all $a \in A$, so by Lemma 4.5 we get infinitely many polymorphisms from **AT**. Obstructions for the other families are:
$$\mathsf{Maj}_3 : \begin{pmatrix}001\\010\\000\\100\\100\end{pmatrix}, \mathsf{Par}_3 : \begin{pmatrix}000\\000\\011\\101\\110\end{pmatrix}, \mathsf{id}\overline{\mathsf{Maj}}_3 : \begin{pmatrix}000\\000\\011\\101\\110\end{pmatrix}, \mathsf{id}\overline{\mathsf{Par}}_3 : \begin{pmatrix}001\\010\\000\\100\\100\end{pmatrix}.$$

4. $A = \{\mathtt{0011}, \mathtt{0100}, \mathtt{0110}, \mathtt{1000}, \mathtt{1001}\}$. Checking the conditions of Lemma 4.9 requires an exhaustive search over sets $S \subseteq [4]$ that we omit in this version. Obstructions for the other families are:
$$\mathsf{Maj}_3 : \begin{pmatrix}001\\010\\100\\100\end{pmatrix}, \mathsf{Par}_5 : \begin{pmatrix}00011\\01100\\10100\\10001\end{pmatrix}, \mathsf{AT}_5 : \begin{pmatrix}00011\\01100\\11000\\10010\end{pmatrix}, \mathsf{id}\overline{\mathsf{Par}}_3 : \begin{pmatrix}001\\010\\100\\100\end{pmatrix}.$$

5. $A = \{\mathtt{00111}, \mathtt{01010}, \mathtt{01101}, \mathtt{10000}, \mathtt{10011}\}$. Checking the conditions of Lemma 4.10 requires an exhaustive search over sets $S \subseteq [5]$ that we omit in this version. Obstructions for the other families are:
$$\mathsf{Maj}_5 : \begin{pmatrix}00011\\01100\\10100\\11000\\10100\end{pmatrix}, \mathsf{Par}_3 : \begin{pmatrix}000\\011\\101\\110\\101\end{pmatrix}, \mathsf{AT}_7 : \begin{pmatrix}0001111\\0110000\\1100000\\1011010\\1101010\end{pmatrix}, \mathsf{id}\overline{\mathsf{Maj}}_3 : \begin{pmatrix}000\\011\\101\\110\\101\end{pmatrix}.$$

□

4.5 Tractability in the Non-idempotent Case

To find the (block)-symmetric polymorphisms needed to apply the BLP+AIP algorithm in the non-idempotent case Lemma 4.6 and Lemma 4.7 tells us that $\mathsf{id}\overline{\mathbf{Par}}$ and $\mathsf{id}\overline{\mathbf{Maj}}$ are not needed since they can be replaced by their non-idempotent counter-parts. It turns out that also **AT** is not essential.

Lemma 4.11. *If* $\mathrm{Pol}(\mathrm{fPCSP}(A, \mathsf{OR}))$ *contains infinitely many polymorphisms from* **AT**, *then* $\mathrm{Pol}(\mathrm{fPCSP}(A, \mathsf{OR}))$ *also contains infinitely many polymorphisms from at least one of* **Maj** *or* $\overline{\mathbf{Maj}}$.

Proof. According to Lemma 4.5, if $\mathrm{Pol}(\mathrm{fPCSP}(A, \mathsf{OR}))$ contains infinitely many polymorphisms from **AT** then there exists integers $c_1, \ldots, c_k \geq 0$, not all 0, such that $\sum_{i=1}^{k} c_i a_i$ has the same value b for all $a \in A$.

If $b \geq \sum_{i=1}^{k} c_i/2$, then according to Lemma 4.3, $\mathrm{Pol}(\mathrm{fPCSP}(A, \mathsf{OR}))$ contains infinitely many polymorphisms from **Maj**. It easy to see, by an argument similar to the proof of Lemma 4.3 that if $b \leq \sum_i c_i/2$, then $\mathrm{Pol}(\mathrm{fPCSP}(A, \mathsf{OR}))$ contains infinitely many polymorphisms from $\overline{\mathbf{Maj}}$. □

We conclude from Lemmas 4.6, 4.7 and 4.11 that if $\mathrm{Pol}(\mathrm{fPCSP}(A, \mathsf{OR}))$ contains infinitely many polymorphisms from **AT**, $\mathsf{id}\overline{\mathbf{Par}}$, or $\mathsf{id}\overline{\mathbf{Maj}}$, then $\mathrm{Pol}(\mathrm{fPCSP}(A, \mathsf{OR}))$ also contains infinitely many polymorphisms from at least one of **Maj**, **Par**, $\overline{\mathbf{Maj}}$, or $\overline{\mathbf{Par}}$. Thus, to establish tractability of $\mathrm{fPCSP}(A, \mathsf{OR})$, it is not necessary to consider any of **AT**, $\mathsf{id}\overline{\mathbf{Par}}$, or $\mathsf{id}\overline{\mathbf{Maj}}$.

4.6 Conditions for Promise-usefulness

In the previous sections, we identified some specific families of block-symmetric polymorphisms, **Maj**, **Par**, **AT**, $\mathsf{id}\overline{\mathbf{Maj}}$, or $\mathsf{id}\overline{\mathbf{Par}}$, that can be used to show that A is $\mathrm{fiPCSP}(A, \mathsf{OR})$ is tractable. It turns out that only the first two out of the five are relevant to establish promise usefulness and we have the following tractability theorem.

Theorem 4.12. *A predicate* $A \subseteq \{0,1\}^k$ *is* fiPCSP*-useful and* fPCSP*-useful if either*

1. *There exists integers* $\alpha_1, \ldots, \alpha_n$ *that are not all* 0, *such that for all* $a \in A$, $\sum_i \alpha_i(a_i - 1/2) \geq 0$.
2. *There exists a non-empty subset* $\beta \subseteq [k]$ *such that for all* $a \in A$, $\oplus_{i \in \beta} a_i$ *is constant (either* 0 *or* 1*).*

Furthermore, if A *does not satisfy any of the conditions above, then for all* $b \notin A$, $\mathrm{Pol}(\mathrm{fPCSP}(A \oplus b, \mathsf{OR}))$ *contains at most finitely many polymorphisms from* **Maj**, **Par**, **AT**, $\mathsf{id}\overline{\mathbf{Maj}}$, *and* $\mathsf{id}\overline{\mathbf{Par}}$.

Proof. That the conditions are sufficient to establish tractability is more or less already established in Lemma 4.3 and Lemma 4.4, respectively. For majority (the first case) we define $b \in \{0,1\}^k$ as

$$b_i = \begin{cases} 0 \text{ if } \alpha_i \geq 0 \\ 1 \text{ otherwise.} \end{cases}$$

and it is easy to see that $\mathrm{fiPCSP}(A \oplus b, \mathsf{OR})$ admits **Maj**, and thus it is tractable. In the second case, if the parity is odd (the constant is 1), then $\mathrm{fiPCSP}(A, \mathsf{OR})$ admits **Par** and thus $\mathrm{fiPCSP}(A, \mathsf{OR})$ is tractable. Otherwise, if the parity is even (the constant is 0),

then by toggling one bit in β we can effectively achieve odd parity. Let $b \in \{0,1\}^k$ be 0 everywhere expect for a single index $i \in \beta$ where $b_i = 1$. We get that fiPCSP$(A \oplus b, \mathsf{OR})$ admits **Par**, and thus fiPCSP$(A \oplus b, \mathsf{OR})$ is tractable.

We now prove the final part of the theorem: if neither condition holds, then A is not fPCSP-useful via any of **Maj**, **Par**, **AT**, id$\overline{\textbf{Maj}}$, or id$\overline{\textbf{Par}}$. The cases of **Maj** or **Par** follows directly from the corresponding conditions stated in Lemma 4.3 and Lemma 4.4.

By Lemmas 4.6, 4.7 and 4.11, if fPCSP$(A \oplus b, \mathsf{OR})$ admits one of **AT**, id$\overline{\textbf{Par}}$, or id$\overline{\textbf{Maj}}$, then fPCSP$(A \oplus b, \mathsf{OR})$ admits at least one of **Maj**, **Par**, $\overline{\textbf{Maj}}$, or $\overline{\textbf{Par}}$. Since we have already established that **Maj** and **Par** cannot be polymorphisms, the only remaining cases are $\overline{\textbf{Maj}}$, and $\overline{\textbf{Par}}$. Note that if fPCSP$(A \oplus b, \mathsf{OR})$ admits $\overline{\textbf{Maj}}$ or $\overline{\textbf{Par}}$, then fPCSP$(A \oplus b \oplus 1^k, \mathsf{OR})$ admits **Maj** or **Par**, which we have already ruled out as a possibility. The conclusion is that A cannot be fPCSP-useful via any of **Maj**, **Par**, **AT**, id$\overline{\textbf{Maj}}$, or id$\overline{\textbf{Par}}$.

□

5 Hardness conditions for Promise-SAT

Our general tool to prove hardness is Theorem 2.15 – whenever all polymorphisms of fiPCSP(A, OR) have small fixing assignments, it is NP-hard. However it is not clear for a general A how to check this condition, so in this section we develop a number of general conditions that are sufficient (but not necessary) to guarantee that all functions in a minion $\mathcal{M}$ have small fixing assignments, while still being relatively easy to test for.

Throughout this section, we often apply terminology for functions to minions, meaning that all functions in the minion have a property. For example a folded minion is a minion in which all functions are folded. Similarly we say that $\mathcal{M}$ has small fixing sets/assignments if all $f \in \mathcal{M}$ have a fixing set/assignment of size at most t for some constant t.

Let us also introduce some notation used in this section. For a minion $\mathcal{M}$, let $\mathcal{M}^0 \supseteq \mathcal{M}$ be the minion consisting of all functions f obtained by taking a function $g \in \mathcal{M}$ and fixing some, possibly empty, set of variables to 0. Analogously we denote by $\mathcal{M}^1$ and $\mathcal{M}^{0,1}$ the minions obtained by allowing fixing variables to 1, and to both 0 and 1, respectively. It is good to keep in mind that even if $\mathcal{M}$ only contains folded and idempotent functions this is no longer true for these derived minions. For a function $f : \{0,1\}^\ell \to \{0,1\}$ which is not identically 0, let $\mathrm{minw}(f) = \min\{\, w(x) \mid f(x) = 1 \,\}$.

5.1 Overview

An obvious *necessary* condition for a folded minion $\mathcal{M}$ to have small fixing assignments is that all (non-constant) $f \in \mathcal{M}$ have bounded $\mathrm{minw}(f)$ (i.e., a low-weight assignment x such that $f(x) = 1$). Note that in the case of monotone (non-decreasing) functions, this condition is also sufficient, but in general it is not. It is not clear whether characterizing this low-weight property is any easier than characterizing small fixing assignments, but our first step is to observe that this property is easily characterized in $\mathcal{M}^0$ (which gives a simple sufficient condition for $\mathcal{M}$ to have the property):

Claim 5.1. *Let $\mathcal{M}$ be a minion. Then every $f \in \mathcal{M}^0$ which is not identically 0 has $\mathrm{minw}(f) \le t - 1$ if and only if $\mathsf{AND}_t \notin \mathcal{M}^0$.*

Proof. Clearly if $\mathsf{AND}_t \in \mathcal{M}^0$ then this is a function with $\mathrm{minw}(f) \ge t$. In the other direction, suppose $\mathsf{AND}_t \notin \mathcal{M}^0$, take any not identically 0 function $f \in \mathcal{M}^0$, and let S be a minimum-cardinality set of coordinates such that $f(S) = 1$. Consider the function $g \in M^0$ of f of arity $|S|$ obtained from f by fixing all variables outside S to 0. By the

minimality of S, we see that $g = \mathsf{AND}_{|S|}$. Hence we conclude that $\mathrm{minw}(f) = |S| < t$ (using the simple fact that $\mathcal{M}^0$ also does not contain any $\mathsf{AND}_{t'}$ for $t' > t$ as shown in Claim 2.22). □

In the following subsections we proceed to give several incomparable conditions, which, together with $\mathsf{AND}_t \notin \mathcal{M}^0$, are sufficient to guarantee small fixing assignments. Some needed definitions are given below but in summary the conditions are as follows.

1. $\mathcal{M}$ only contains functions with small *matching number* (Lemma 5.4 in Section 5.2). In this case we even obtain small fixing sets.

2. $\mathcal{M}$ only contains functions with small *inverted matching number* (Lemma 5.8 in Section 5.3).

3. $\mathcal{M}$ only contains *unate* functions, and additionally $\mathcal{M}^0$ not containing arbitrarily large ANDNOR functions (Lemma 5.10 in Section 5.4), where

$$\mathsf{ANDNOR}_t(x) = x_1 \wedge \mathsf{NOR}(x_2, \ldots, x_t) = x_1 \wedge \neg x_2 \wedge \ldots \wedge \neg x_t.$$

   Forbidding large ANDNOR is a natural variant of the AND condition which guarantees that a small number of additional variables can be set to 0 to extend a low-weight 1-assignment to a fixing assignment.

Many predicates A yield a minion that satisfies one of the three conditions above but it turns out that a "bottleneck" (in the sense that a large fraction of hard promises A are not covered by it) is the shared condition $\mathsf{AND}_t \notin \mathcal{M}^0$ for guaranteeing that $\mathrm{minw}(f)$ is small. To address this, we identify a second, more complicated, explicit condition for bounding $\mathrm{minw}(f)$, which we refer to as $\mathcal{M}$ being t-ADA-free (Definition 5.11) This condition can replace the AND condition in all three of the fixing assignment results mentioned above, resulting in stronger versions (Theorems 5.16, 5.17 and 5.18) of these results which significantly reduces the number of predicates A that we are unable to classify.

As one last step to further sharpen these results, we give a different condition for the unate case (item 3) above, where instead of forbidding the ANDNOR function (which seems to be the main bottleneck after introducing ADA-freeness), we forbid certain functions that we refer to as UnCADAs and UnDADAs (Theorem 5.21 in Section 5.7). Thus we have in total four different hardness conditions and Figure 1 gives a graphical overview of these.

Crucially, given a concrete predicate $A \subseteq \{0,1\}^k$, checking all the various properties shown in Figure 1 for $\mathcal{M} = \mathrm{Pol}(\mathrm{fiPCSP}(A, \mathsf{OR}))$ can be done relatively efficiently because they all boil down to (non-)existence of certain polymorphisms of small constant arity. We discuss these computational aspects in more detail in Appendix A.

5.2 Bounded Matchings

An immediate consequence of Claim 5.1 is, as mentioned in the overview, that if all functions in $\mathcal{M}$ are monotone and $\mathsf{AND}_t \notin \mathcal{M}^0$, then all $f \in \mathcal{M}$ have fixing sets of size at most $t - 1$. However, the monotonicity property can be relaxed, leading us to the following notion which quantifies to what extent a function can take the value 1 on a large number of disjoint sets.

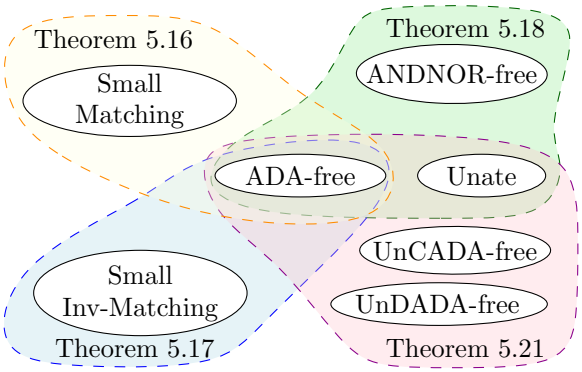

Figure 1: Overview of the conditions required for Theorems 5.16, 5.17, 5.18 and 5.21 that ensure the existence of small fixing assignments.

Definition 5.2. Let $f : \{0,1\}^\ell \to \{0,1\}$. A *matching* of f of size t is a collection of t disjoint subsets $S_1, S_2, \ldots, S_t$ of $[\ell]$ such that $f(S_i) = 1$ for all $i \in [t]$. The *matching number* of f is the maximum size of any matching of f.

A folded idempotent Boolean function is monotone if and only if it has matching number 1. This is because two sets $S \subset T$ such that $f(S) = 1$ and $f(T) = 0$ implies that f is 1 on the two disjoint sets S and $\overline{T}$. Hence the notion of having bounded matching number can be viewed as a generalization of monotonicity. If the functions of a minion $\mathcal{M}$ have bounded matching number, then this rules out families such as Par and AT from $\mathcal{M}$, but it does not rule out Maj.

Remark 5.3. If a folded function f has a t-fixing set then it has a matching number at most t. This follows since if $f(S) = 1$ then S must intersect the fixing set, otherwise the fixing set would force $f(\overline{S}) = 1$ contradicting that f is folded. In other words, if we are looking for small fixing sets, then bounded matching number is a necessary condition to achieve this.

Having bounded matching number and forbidding AND in $\mathcal{M}^0$ is enough to conclude the existence of small fixing sets.

Lemma 5.4. *Let $\mathcal{M}$ be a folded idempotent minion, and suppose that there exists constants t_1, t_2, such that*

1. $\mathsf{AND}_{t_1} \notin \mathcal{M}^0$.
2. *Every $f \in \mathcal{M}$ has matching number $\leq t_2$.*

Then all $f \in \mathcal{M}$ have a fixing set of size at most $(t_1 - 1) \cdot t_2$.

Proof. Let $f \in \mathcal{M}$ and let S_1 be the smallest cardinality subset such that $f(S_1) = 1$ (note that $\mathcal{M}$ being idempotent guarantees S_1 is non-empty). Let S_2 be the smallest cardinality subset disjoint from S_1 such that $f(S_2) = 1$. Continue this procedure until there is no set S disjoint from all selected sets such that $f(S) = 1$. Suppose the sequence we end up with is $S_1, S_2, \ldots S_m$. Since f has matching number $\leq t_2$, we know that $m \leq t_2$ and since f is folded, $f(S) = 1$ for any set that contains the union of the selected sets and hence $\bigcup_{i\in[m]} S_i$ is a fixing set.

Finally by Claim 5.1 it follows that we can ensure $|S_i| \leq t_1 - 1$ for each i and hence the fixing set $\bigcup_{i\in[m]} S_i$ is of size at most $(t_1 - 1)t_2$, as desired. □

5.3 Bounded Inverted Matchings

For the condition in Section 5.2 using bounded matching numbers, it is natural to consider a variant where a function cannot have many disjoint sets of variables that can flip the function from 1 to 0. This leads us to the following definition.

Definition 5.5. For a function $f : \{0,1\}^\ell \to \{0,1\}$ and a, possibly empty, set $S \subseteq [\ell]$ such that $f(S) = 1$, an *inverted matching* of size t of f with respect to S is a collection of t disjoint subsets $T_1, \dots, T_t$ of $[\ell]$ such that $f(S \cup T_i) = 0$ for all $i \in [t]$. The *inverted matching number* of f is the maximum size of any inverted matching of f with respect to any $S \in f^{-1}(1)$.

To obtain small fixing assignments for functions with low inverted matching number, we also need a condition that guarantees that we can always make the sets T_i involved in an inverted matching small. With these definitions, we then have the following direct generalization of Lemma 5.4.

Lemma 5.6. *Let $\mathcal{M}$ be a minion such that:*

1. $\mathsf{NAND}_{t_1} \notin \mathcal{M}^{0,1}$, *and*
2. *Every $f \in \mathcal{M}$ has inverted matching number $\leq t_2$.*

Then for any $f \in \mathcal{M}$ and S such that $f(S) = 1$, there is a set T, disjoint from S, of size at most $(t_1 - 1)t_2$ such that setting $x_S = 1$ and $x_T = 0$ is a fixing assignment of f.

Proof. The proof is analogous to that of Lemma 5.4. Create a sequence $T_1, \dots, T_m$ of subsets where T_i is a minimum-cardinality set of coordinates disjoint from S and $T_1, \dots, T_{i-1}$ such that $f(S \cup T_i) = 0$, stopping when no more such sets exist. Since f has bounded inverted matching number, we have $m \leq t_2$. Furthermore, it is clear that fixing S to 1 and the coordinates of $T := \bigcup_{i=1}^m T_i$ to 0 is a fixing assignment for f, so it remains to bound the size $|T|$.

Consider the function $g \in M^{0,1}$ of arity $|T_i|$ obtained from f by fixing all variables of S to 1 and all variables outside $S \cup T_i$ to 0. By the minimality of T_i, the function g is the $\mathsf{NAND}_{|T_i|}$ function and hence $|T_i| \leq t_1 - 1$. Thus $|T| = \sum |T_i| \leq (t_1 - 1) \cdot t_2$, as desired. □

For folded minions, bounded inverted matching number implies not having NAND.

Claim 5.7. *Let $\mathcal{M}$ be a folded minion. If all $f \in \mathcal{M}$ have inverted matching number $< t$, then $\mathsf{NAND}_t \notin \mathcal{M}^{0,1}$.*

Proof. Suppose for contradiction that $\mathsf{NAND}_t \in \mathcal{M}^{0,1}$. Since $\mathcal{M}$ is folded this happens if and only if $\mathsf{NOR}_t \in \mathcal{M}^{0,1}$ (because in the folded setting, switching which variables are fixed to 0 and which variables are fixed to 1 takes a function f to its dual). In other words, if $\mathsf{NAND}_t \in \mathcal{M}^{0,1}$ there is a function $f \in \mathcal{M}$ of arity $t+2$ such that $f(0,1,x) = \mathsf{NOR}_t(x)$. In particular $f(0,1,0^t) = 1$, and $f(0,1,e_i) = 0$ for $1 \leq i \leq t$, so f has inverted matching number at least t. □

Combining Claims 5.1 and 5.7 and Lemma 5.6 yields the following immediate corollary.

Lemma 5.8. *Let $\mathcal{M}$ be a folded minion such that $\mathsf{AND}_{t_1} \notin \mathcal{M}^0$ and all $f \in \mathcal{M}$ have inverted matching number $\leq t_2$. Then all $f \in \mathcal{M}$ have a fixing assignment of size at most $t_1 - 1 + t_2^2$.*

Proof. By Claim 5.7 we know that $\mathsf{NAND}_{t_2+1} \notin \mathcal{M}^{0,1}$ and by Claim 5.1 we have $\mathrm{minw}(f) \leq t_1 - 1$ for any $f \in \mathcal{M}$. The lemma now follows by Lemma 5.6. □

Note that unlike Lemma 5.4, this result does not need $\mathcal{M}$ to be idempotent. It is also possible to refrain from using Claim 5.7 and instead get the same result with the requirement that $\mathcal{M}$ is folded replaced by $\mathsf{NAND}_t \notin \mathcal{M}^{0,1}$.

5.4 Unate Functions

Let us turn to a different generalization of being monotone. A function is *unate* if for each variable x_i, f is either positive in x_i or negative in x_i (or both, if f does not depend on x_i). In other words there cannot exist an i and two sets S, T such that both $f(S) < f(S \cup \{i\})$ and $f(T) > f(T \cup \{i\})$. Note that in a fixing assignment for a unate function, we can without loss of generality set all positive variables included to 1, and all negative variables included to 0. This motivates the following strategy: first pick a small set S of positive variables such that $f(S) = 1$ (as guaranteed by e.g. Claim 5.1). Then pick a small number T of negative variables, such that even if all other negative variables outside T are set to 1, f still equals 1. Fixing S to 1 and T to 0 then yields a fixing assignment of size $|S| + |T|$.

In order to bound the size of T in this plan, we can employ a similar idea as in Claim 5.1. The function to forbid is now the less natural function $\mathsf{ANDNOR}_t = x_1 \wedge \mathsf{NOR}(x_2, \ldots, x_t)$. We have the following lemma.

Lemma 5.9. *Let $\mathcal{M}$ be an idempotent and unate minion such that $\mathsf{ANDNOR}_t \notin \mathcal{M}^0$ for some $t \geq 2$. Then for every $f \in \mathcal{M}$ and every S such that $f(S) = 1$, there is a set $T \subseteq [\ell] \setminus S$ of size $|T| \leq t - 2$ such that setting $x_S = 1$ and $x_T = 0$ is a fixing assignment of f (of size $|S| + |T|$).*

Proof. Let Y be the set of negative variables of f (except those included in S, if any) and let $T \subseteq Y$ be minimal such that $f(S \cup (Y \setminus T)) = 1$ (such a T exists since $T = Y$ satisfies the requirement). Construct a function $g \in \mathcal{M}^0$ of arity $|T| + 1|$ by fixing all variables of $[\ell] \setminus (S \cup Y)$ to 0, identifying all variables of $S \cup (Y \setminus T)$ into a new variable x_0, and keeping the variables of T as $x_1, \ldots, x_{|T|}$. By the minimality of T, the function g satisfies

$$g(1, x_1, \ldots, x_{|T|}) = \mathsf{NOR}(x_1, \ldots, x_t)$$

Furthermore, since $\mathcal{M}$ is idempotent we have $g(0, 0, \ldots, 0) = 0$, which together with g being non-positive in $x_1, \ldots, x_{|T|}$ implies that $g(0, x_1, \ldots, x_{|T|}) = 0$. We thus see that $g = \mathsf{ANDNOR}_{|T|+1}$ and hence since $\mathsf{ANDNOR}_t \notin \mathcal{M}^0$ (and using Claim 2.22) we have $|T| + 1 < t$.

To see that setting $x_S = 1$ and $x_T = 0$ yields a fixing assignment, note that since f is unate we have for any such x that $f(x) \geq f(S \cup (Y \setminus T)) = 1$. □

Claim 5.1 and Lemma 5.9 yield the following immediate corollary.

Lemma 5.10. *Let $\mathcal{M}$ be an idempotent and unate minion such that $\mathsf{AND}_{t_1} \notin \mathcal{M}^0$ and $\mathsf{ANDNOR}_{t_2} \notin \mathcal{M}^0$ for some t_1, t_2. Then every $f \in \mathcal{M}$ has a $(t_1 + t_2 - 3)$-fixing assignment.*

Proof. Let $f \in \mathcal{M}$. If f is identically 0 it has an empty fixing assignment, otherwise Claim 5.1 yields an S of size at most $t_1 - 1$ such that $f(S) = 1$, and then by Lemma 5.9 there is a T of size at most $t_2 - 2$ which together with S forms a fixing assignment of size $|S| + |T| = t_1 + t_2 - 3$. □

5.5 Beyond AND: Approximate Double-Ands

While not á priori clear, it turns out from experiments that the main bottleneck in the hitherto described hardness conditions is Claim 5.1 which guarantees that all $f \in \mathcal{M}$ have low-weight 1-assignments whenever $\mathsf{AND}_t \notin \mathcal{M}^0$. To alleviate this we now give a weaker sufficient condition to bound $\mathrm{minw}(f)$. The idealized type of function we now want to forbid are functions of the form $\mathsf{AND}(x_S) \vee \mathsf{AND}(x_T)$ for two overlapping but incomparable sets S and T of size $\geq t$. This is still somewhat restrictive, and the actual functions we forbid can be viewed as approximate versions of this. Let us give the formal definition.

Definition 5.11. For positive integers c, d, a (c,d)-ADA (Approximate Double-AND) is a $(c+2d)$-ary function $f : \{0,1\}^d \times \{0,1\}^c \times \{0,1\}^d \to \{0,1\}$ such that the following holds:

(a) $f(1^d, 1^c, 0^d) = f(0^d, 1^c, 1^d) = 1$,

(b) $f(x,y,z) = 0$ if $w(x) + w(y) + w(z) < c + d$.

(c) $f(x,y,z) = 0$ unless at least two of x, y, z are all-1s

We say that a minion $\mathcal{M}$ is *t-ADA-free*, $t \geq 2$, if it does not contain a $(t-d, d)$-ADA for any $1 \leq v \leq t-1$.

It is easy to see that being t-ADA-free is a weaker property than not having AND_t.

Claim 5.12. *Let $\mathcal{M}$ be a minion. If $\mathsf{AND}_t \notin \mathcal{M}^0$ then $\mathcal{M}^0$ is t-ADA-free.*

Proof. To prove the contrapositive, note that if $\mathcal{M}^0$ contains a (c,d)-ADA $f(x,y,z)$ for some $c + d = t$ then fixing $z = 0^d$ yields the AND_t function. □

We have the following technical lemma, which effectively generalizes Claim 5.1.

Lemma 5.13. *Let $\mathcal{M}$ be a minion such that $\mathcal{M}^0$ is t-ADA-free for some $t \geq 2$. Then for every f in $\mathcal{M}^0$ which is not identically 0, at least one of the following two properties holds:*

1. $\mathrm{minw}(f) \leq t-1$, *or*
2. *$S \cap T = \emptyset$ for every pair of inclusion-wise minimal $S, T \in f^{-1}(1)$.*

In the next section, we show how this enables us to strengthen the hardness conditions of the previous sections that were based on Claim 5.1. But first, let us prove the technical lemma.

Proof. We proceed by induction on the arity ℓ of f. The base cases $\ell \leq t$ clearly holds – either $\mathrm{minw}(f) \leq t-1$, or the only set in $f^{-1}(1)$ is $[\ell]$ itself.

For the inductive step consider a function $f : \{0,1\}^\ell \to \{0,1\}$ and suppose the lemma is true for all $\ell' < \ell$. Suppose for contradiction that the lemma fails for f, i.e.,

1. $\mathrm{minw}(f) \geq t$, *and*
2. There exists inclusion-wise minimal $S, T \in f^{-1}(1)$ such that $S \cap T \neq \emptyset$.

Note that we must have $S \cup T = [\ell]$, since otherwise fixing a variable of f outside $S \cup T$ to 0 yields a contradiction to the inductive hypothesis. Among all possible ways of choosing the pair (S, T), choose one which minimizes $(|S|, |T|)$ (i.e., first minimize $|S|$, then subject to this minimize $|T|$). We then have the following claim.

Claim 5.14. $|S| = |T| = t$.

Proof. Let us start with $|S|$ and suppose for contradiction that $|S| > t \geq 2$. Let $i, j \in S$ be such that either both are in T or neither are in T (since $|S| \geq 3$ such i, j must exist). Consider the minor g of f obtained by identifying x_i and x_j, and the sets of coordinates S' and T' obtained from S and T by identifying these two coordinates; note that $|S'| = |S| - 1$, while $|T'|$ is either $|T| - 1$ or $|T|$ depending on whether the identified coordinates i, j are in T or not.

Since S' and T' are inclusion-wise minimal sets in $g^{-1}(1)$ that intersect, the inductive hypothesis implies that $\mathrm{minw}(g) = t - 1$. It follows that there exists a set $X \subseteq [\ell]$ of size $|X| = t < |S|$, containing i, j, such that $f(X) = 1$. Since S is inclusion-wise minimal we cannot have $X \subseteq S$. But then (X, S) would have been a valid choice of the sets (S, T) as they intersect at i. Since S was chosen with minimum possible cardinality, this contradicts the assumption that $|S| > t$.

Having established $|S| = t$, suppose for contradiction that $|T| > t$. This means that $|T \setminus S| \geq 2$ (since S cannot be contained in T). Repeating the argument above with two elements in $T \setminus S$ we establish the existence of an $X \subseteq [\ell]$ of size $|X| = t$ and intersecting $T \setminus S$ with $f(X) = 1$. By inclusion-wise minimality of T, X must intersect $[\ell] \setminus T = S \setminus T$ and hence (S, X) would have been a valid choice of the sets (S, T). □

Let $c = |S \cap T|$ and $d = |S \setminus T| = |T \setminus S| = t - c$. By reordering the $\ell = |S \cup T| = c + 2d$ variables of f we can write f as a function $f(x, y, z)$ on $\{0,1\}^d \times \{0,1\}^c \times \{0,1\}^d$, with x being the variables from $S \setminus T$, z the variables from $T \setminus S$, and y the variables from $S \cap T$. We claim that with this ordering of the variables, f is a (c, d)-ADA. To wit, let us verify the properties:

(a) $f(1^d, 1^c, 0^d) = f(0^d, 1^c, 1^d) = 1$. This is clear since the first value is $f(S)$ and the second is $f(T)$.

(b) $f(x, y, z) = 0$ if $w(x) + w(y) + w(z) < c + d$. This follows since $\mathrm{minw}(f) \geq t = c + d$.

(c) $f(x, y, z) = 0$ unless at least two of x, y, z are all-1s. Suppose for contradiction that this is not true and take a counterexample (x, y, z) of minimum weight $w(x) + w(y) + w(z)$. Assume without loss of generality that $x \neq 1^d$ (otherwise, we can swap the roles of S and T which swaps x and z). We then have $(y, z) \neq (1^c, 1^d)$, and in particular the subset $\tilde{S} \subseteq [\ell]$ of 1-coordinates of (x, y, z) satisfies the following two properties:

- $\tilde{S} \not\supseteq S$ (since $x \neq 1^d$) and $\tilde{S} \not\supseteq T$ (since $(y, z) \neq (1^c, 1^d)$).
- $\tilde{S}$ is inclusion-wise minimal in $f^{-1}(1)$ (since (x, y, z) were chosen with minimum possible weight)

Now consider the function $\tilde{f} \in \mathcal{M}^0$ of arity $\ell - 1$ obtained by fixing some coordinate in $S \setminus \tilde{S}$ to 0. Then $(\tilde{S}, T)$ is a pair of inclusion-wise minimal but intersecting sets in $\tilde{f}^{-1}(1)$ and hence $\mathrm{minw}(\tilde{f}) \leq t - 1$ by the induction hypothesis, but then $\mathrm{minw}(f) \leq \mathrm{minw}(\tilde{f}) \leq t - 1$.

Thus we have now established that $f \in \mathcal{M}^0$ is a $(t-d,d)$-ADA, which contradicts the fact that $\mathcal{M}^0$ is t-ADA-free, and the inductive step follows. □

A useful corollary of the technical lemma is the following.

Corollary 5.15. *Let $\mathcal{M}$ be a folded minion such that $\mathcal{M}^0$ is t-ADA-free. Then for every $f \in \mathcal{M}$ it holds that* $\operatorname{minw}(f) \leq t-1$.

Proof. Proof by induction on the arity ℓ of f. As base case we have $\ell < 2t-1$ for which the result is trivially true since every folded function f of arity ℓ has $\operatorname{minw}(f) \leq \lceil \ell/2 \rceil$.

For the inductive case assume the claim is true for $\ell - 1$ and assume for contradiction that it is false for some f of arity $\ell \geq 2t-1 > t$. For two arbitrary coordinates $i, j \in [\ell]$, consider the minor g of f formed by identifying i and j. By the induction hypothesis, there is a set S' of size $\leq t-1$ for g such that $g(S') = 1$. Since f does not have such a set, S' must include the new coordinate. Replacing the new coordinate by i and j we get a set $S \subseteq [\ell]$ of size t such that $f(S) = 1$ and $i, j \in S$.

Let $i' \in S$ and $j' \notin S$, and repeat this process to obtain a set $T \subseteq [\ell]$ of size t such that $f(T) = 1$ and $i', j' \in S$. This gives two intersecting sets contradicting Lemma 5.13. □

5.6 Strengthened Hardness Conditions

Using the notion of t-ADA-free minions we can strengthen the previous conditions for small fixing assignments.

Theorem 5.16 (Strengthening of Lemma 5.4)**.** *Let $\mathcal{M}$ be a folded idempotent minion such that $\mathcal{M}^0$ is t_1-ADA-free and every $f \in \mathcal{M}$ has matching number $f \leq t_2$. Then all $f \in \mathcal{M}$ have a fixing set of size at most $(t_1 - 1) \cdot t_2$.*

Theorem 5.17 (Strengthening of Lemma 5.8)**.** *Let $\mathcal{M}$ be a folded minion such that $\mathcal{M}^0$ is t_1-ADA-free and all $f \in \mathcal{M}$ have inverted matching number $\leq t_2$. Then all $f \in \mathcal{M}$ have a fixing assignment of size at most $t_1 - 1 + t_2^2$.*

Theorem 5.18 (Strengthening of Lemma 5.10)**.** *Let $\mathcal{M}$ be a folded, idempotent, and unate minion such that $\mathcal{M}^0$ is t_1-ADA-free and $\mathsf{ANDNOR}_{t_2} \notin \mathcal{M}^0$. Then every f in $\mathcal{M}$ has a $(t_1 + t_2 - 3)$-fixing assignment.*

Theorems 5.17 and 5.18 follow immediately from Corollary 5.15 combined with the previous results. For Theorem 5.17 we use Lemma 5.6 and Claim 5.7 while for Theorem 5.18 we use Lemma 5.9.

Theorem 5.16 is not quite as immediate, because the proof of the original condition Lemma 5.4 uses the conclusion of Claim 5.1 that all $f \in \mathcal{M}^0$ have small $\operatorname{minw}(f)$ (as opposed to the other two results which only use the property that all $f \in \mathcal{M}$ have small $\operatorname{minw}(f)$), and this is not guaranteed by Lemma 5.13 or Corollary 5.15. Nevertheless, we can extend the proof of Lemma 5.4 to obtain the desired result.

Proof of Theorem 5.16. We begin as in the proof Lemma 5.4, picking a sequence of disjoint $S_1, \ldots, S_m$ such that each $f(S_i) = 1$ and S_i is of minimum possible cardinality subject to being disjoint from $S_1 \cup \ldots \cup S_{i-1}$ (the assumption that f is idempotent guarantees $S_i \neq \emptyset$). The union of these forms a fixing set and $m \leq t_2$. If all $|S_i| \leq t_1 - 1$ then we are done.

Otherwise, let i be the first index such that $|S_i| \geq t_1$, and consider the function $g \in \mathcal{M}^0$ obtained by fixing the coordinates of $S_1, \ldots, S_{i-1}$ to 0. Since $\operatorname{minw}(g) = |S_i| \geq t_1$, we have by Lemma 5.13 that all inclusion-wise minimal sets in $g^{-1}(1)$ are pairwise disjoint.

This implies that $S_i, \ldots, S_m$ are the only inclusion-wise minimal sets in $g^{-1}(1)$ and in particular any $T \in g^{-1}(1)$ contains S_j for some $j \in \{i, \ldots, m\}$. Thus picking one coordinate each from the sets $S_i, \ldots, S_m$ yields a hitting set for $g^{-1}(1)$ and setting these together with $S_1, \ldots, S_{i-1}$ to 0 yields a fixing assignment forcing $f(x) = 0$ of size at most $(i-1)(t_1-1) + m - i + 1 \leq (t_1 - 1) \cdot m \leq (t_1 - 1) \cdot t_2$. As f is folded, the variables set to 0 is fixing set. □

5.7 Beyond ANDNOR: Unate Controlled ADAs

In this section we focus on unate minions and describe another sufficient condition for such minions to have small fixing assignments. We say that a function f is (p, q)-unate if it can be written as $f : \{0,1\}^p \times \{0,1\}^q \to \{0,1\}$ where $f(x, y)$ is non-negative in x, and non-positive in y (if f does not depend on some variable it can be placed in either group, so it is possible for a function to be both (p, q)-unate and $(p+1, q-1)$-unate).

For fixing assignments of unate functions, we say that $S \subseteq [p], T \subseteq [q]$ of sizes $|S| = s$ and $|T| = t$ is an (s, t)-fixing assignment for a (p, q)-unate function $f : \{0,1\}^p \times \{0,1\}^q \to \{0,1\}$ if $f(S, \overline{T}) = 1$ (i.e., setting all non-negative variables of S to 1, and all non-positive variables of T to 0, fixes f to 1). Our general goal in this section is to identify finite conditions which are sufficient to guarantee that all functions in a unate minion have $(t-1, 1)$-fixing assignments for some t, i.e., fixing assignments of size t which sets one variable to 0 and $t-1$ variables to 1.

Similar to the approach in Section 5.5 and the notion of t-ADA-free minions, the condition is based on forbidding certain functions that look similar to a disjunction of two overlapping ANDs, but in this case each of the ANDs is "controlled" by some negative inputs and we generally refer to them as Controlled ADAs. There are (unfortunately) two very similar but subtly different types of functions. Let us define the first (main) type.

Definition 5.19. A (c, d)-*UnCADA* (Unate Controlled Approximate Double-AND) is a $(c + 2d + 1, 3)$-unate folded and idempotent function $f : \{0,1\}^{c+2d+1} \times \{0,1\}^3 \to \{0,1\}$ such that

(a) $f(1^d 1^c 0^d 0, 011) = f(0^d 1^c 1^d 0, 101) = 1$

(b) for every $x \in \{0,1\}^{c+2d}$ and $y \in \{0,1\}^2$ such that $w(x) \leq c + d - 1$ and $w(y) \geq 1$ it holds that $f(x0, y1) = 0$.

A minion $\mathcal{M}$ is *t-UnCADA-free*, $t \geq 2$, if it does not contain a $(t-d, d)$-UnCADA for any $1 \leq d \leq t-1$.

It may be instructive to compare Definition 5.19 with Definition 5.11 of (c, d)-ADAs, since the definitions are very similar. The conditions (a) in both definitions are essentially the same, saying that f has two 1-assignments with a certain overlap pattern determined by c and d. Likewise the conditions (b) are analogous, with the main difference stemming from Definition 5.19 requiring that f is positive in the x-variables and negative in the y-variables (unlike an ADA, which has no such requirements). Another minor difference is the presence of the last positive variable in Definition 5.19, which is 0 in all prescribed function values. This takes a similar role as M^0 does in previous arguments – rather than dropping this variable and using $\mathcal{M}^0$ being UnCADA-free, we explicitly include it in the definition and operate directly on the original $\mathcal{M}$. The reason for this is to be able to require f being positive in this variable.

The second type of function has an easier definition.

Definition 5.20. For $t \geq 3$, a *t-UnDADA* (Unate Double-controlled Approximate Double-AND) is a $(t,4)$-unate folded and idempotent function $f : \{0,1\}^t \times \{0,1\}^4 \to \{0,1\}$ such that

(a) $f(1^{t-1}0, 0011) = f(01^{t-1}, 1001) = 1$

(b) for every $x \in \{0,1\}^t$ and $y \in \{0,1\}^3$ such that $w(x) \leq t-1$ and $w(y) \geq 2$, it holds that $f(x, y1) = 0$

Thus in a UnDADA, two negative variables ("control bits") need to be set to 0 to activate each of the "ANDs", as opposed to a single negative variable in an UnCADA. One important but subtle difference in the definition is that an UnCADA has an extra positive variable which is 0 in all prescribed function values, but the UnDADA does not have this. While this may seem like a minor detail, it turns out to be very important – there are many minions of interest which do not have t-UnDADAs, but that would have them if we were to add the additional 0-variable in the UnDADA definition.

We are finally ready to state our fixing assignment condition.

Theorem 5.21. *Let $\mathcal{M}$ be a folded, idempotent, and unate minion which is t-ADA-free, t-UnCADA-free, and does not have a t-UnDADA. Then every $f \in \mathcal{M}$ has a $(t-1,1)$-fixing assignment.*

Let us first prove the following preparatory general claim, which does not require any specific properties of $\mathcal{M}$ but captures how the inductive hypothesis is used in the proof.

Claim 5.22. *Let $\mathcal{M}$ be a unate minion such that all $(p-1,q)$-unate $f \in \mathcal{M}$ and all $(p,q-1)$-unate $f \in \mathcal{M}$ have a $(t-1,1)$-fixing assignment. Then for any (p,q)-unate $f \in \mathcal{M}$, either f has a $(t-1,1)$-fixing assignment, or the following conditions all hold:*

(a) For any $i \in [p]$ and $j \in [q]$, f has a $(t,1)$-fixing assignment (S,T) such that $i \in S$ and $j \notin T$.

(b) For any $i \in [p]$ and $j \in [q]$, f has a $(t-1,2)$-fixing assignment (S,T) such that $i \notin S$ and $j \in T$

(c) For any $j, j' \in [q]$, f has a $(t-1,2)$-fixing assignment $(S, \{j,j'\})$.

Proof. For (a) and (b), identify the positive variable $i \in [p]$ and the non-positive $j \in [q]$ to obtain a minor g. The function g is either a $(p-1,q)$-unate or $(p,q-1)$-unate function, so by the assumption of the lemma, g has a $(t-1,1)$-fixing assignment (S',T'). Note that the newly created variable cannot be included in the fixing assignment – if it was, it would either correspond to a $(t-1,2)$-fixing assignment which sets the negative variable j to 1, or a $(t,1)$-fixing assignment which sets the positive variable i to 0, and in either case one variable can be dropped to obtain a $(t-1,1)$-fixing assignment for f.

This implies that $(S' \cup \{i\}, T')$ and $(S', T' \cup \{j\})$ are fixing assignments of f, establishing items (a) and (b).

The last item (c) follows in a similar manner by identifying the two negative variables j and j', obtaining a $(p,q-1)$-unate function. The resulting minor g must now have a $(t-1,1)$-fixing where the non-positive variable used is the newly identified variable. □

Proof of Theorem 5.21. Consider a general (p,q)-unate function $f \in \mathcal{M}$. We do double induction on p and q. The base cases $p \leq t-1$ are trivial as $\mathcal{M}$ is idenpotent. For the inductive step, we have to treat the first step $p = t$ differently, so we divide into two cases $p = t$ and $p > t$.

Case 1: $p = t$ The case $p = t$ is special and requires a bit of extra care (in fact it may be more instructive to read the general case $p > t$ below first). In this case the base cases $q \leq 2$ follow by f being folded – any folded (p, q)-unate function has a $(\lceil p/2 \rceil, \lceil q/2 \rceil)$-fixing assignment. So we may assume $q \geq 3$.

For $i \in [p]$ and $j, j' \in [q]$, let us say that the triple (i, j, j') is fixing if $([p] - i, \{j, j'\})$ is a $(t-1, 2)$-fixing assignment of f. By Claim 5.22 we have the following two properties:

- For every $i \in [p]$ and $j \in [q]$, there is a $j' \in [q]$ such that (i, j, j') is fixing.
- For every $j, j' \in [q]$, there is an $i \in [p]$ such that (i, j, j') is fixing.

These two properties imply that there are two fixing triples (i_1, j_1, j') and (i_2, j_2, j') with $i_2 \neq i_1$ and $j_2 \neq j_1$. (To see this, note that if this was not the case then the second property would imply that there is a unique i such that for every j, j' only (i, j, j') is fixing, but this contradicts the first property.)

Let $S_1 = [p] - i_1$, $T_1 = \{j_1, j'\}$, $S_2 = [p] - i_2$, and $T_2 = \{j_2, j'\}$ – these are now two $(t-1, 2)$-fixing assignments such that $S_1 \neq S_2$ but not disjoint, and $T_1 \neq T_2$ but not disjoint. Create a minor $g(x, y)$ of f on $\{0,1\}^t \times \{0,1\}^4$ as follows:

- x are the t coordinates of $[p]$, with i_1 as x_1 and i_2 as x_t.
- y_1, y_2, y_3 are the 3 coordinates j_1, j', j_2 of $T_1 \cup T_2$.
- y_4 is the remaining $q - 3$ coordinates of $[q]$, identified to a single variable.

We can now check that g is in fact a t-UnDADA. It is folded and idempotent since f is. Let us verify the other properties.

(a) $g(1^{t-1}0, 0011) = f(S_1, \overline{T_1}) = 1$
$g(01^{t-1}, 1001) = f(S_t, \overline{T_2}) = 1$

(b) If $w(x) \leq t-1$ and $w(y) \geq 2$ then $g(x, y1) = 0$ since f does not have a $(t-1, 1)$-fixing assignment.

This contradicts the assumption that $\mathcal{M}$ does not have a t-UnDADA, so f must have a $(t-1, 1)$-fixing assignment.

Case 2: $p > t$. When $p > t$, the base cases $q \in \{0, 1\}$ follow by $\mathcal{M}$ being t-ADA-free and Corollary 5.15, so we may assume $q \geq 2$. By Claim 5.22 and the inductive hypothesis, f has a $(t, 1)$-fixing assignment $(S_1, T_1 = \{j_1\})$. Applying Claim 5.22 again to some $i \notin S_1$ (such i exists since $p \geq t + 1$) and j_1, we see that f also has a $(t, 1)$-fixing assignment $(S_2, T_2 = \{j_2\})$ such that $i \in S_2$ and $T_2 = \{j_2\} \neq \{j_1\} = T_1$. In particular $S_1 \neq S_2$ and $T_1 \neq T_2$.

Let $c = |S_1 \cap S_2|$ and $d = |S_1 \setminus S_2| = |S_2 \setminus S_1|$. Create a minor $g(x, y)$ of f on $\{0,1\}^{c+2d+1} \times \{0,1\}^3$ as follows:

- $x_1 \dots x_d$ are the coordinates of $S_1 \setminus S_2$.
- $x_{1+d} \dots x_{c+d}$ are the coordinates of $S_1 \cap S_2$.
- $x_{c+d+1} \dots x_{c+2d}$ are the coordinates of $S_2 \setminus S_1$.
- x_{c+2d+1} are the remaining positive coordinates, $[p] \setminus (S_1 \cup S_2)$, identified to a single variable.

- y_1 is the coordinate T_1.
- y_2 is the coordinate T_2.
- y_3 are the remaining negative coordinates $[q] \setminus (T_1 \cup T_2)$, identified to a single variable.

Some of these (in particular the variables identified into x_{c+2d+1} and y_3) might be empty, in which case g simply does not depend on that variable.

We can now check that g is in fact a (c, d)-UnCADA. It is idempotent and folded since f is. Let us verify the other properties.

(a) $g(1^d1^c0^d0, 011) = f(S_1, \overline{T_1}) = 1$
$g(0^d1^c1^d0, 101) = f(S_2, \overline{T_2}) = 1$

(b) For any $x \in \{0,1\}^{c+2d}$ and $y \in \{0,1\}^2$ such that $w(x) \leq c + d - 1 = t - 1$ and $w(y) \geq 1$, it holds that $g(x0, y1) = 0$ since f does not have a $(t-1, 1)$-fixing assignment.

This contradicts the assumption that $\mathcal{M}$ is t-UnCADA-free, so f must have a $(t-1, 1)$-fixing assignment. This concludes the $p > t$ case of the proof, and finishes the overall proof. □

Example 5.23. The main reason for the introduction of the hardness condition given by Theorem 5.21 that uses UnCADA and UnDADA is to capture the hardness of fiPCSP($\{\texttt{0011}, \texttt{0101}, \texttt{0110}, \texttt{1000}, \texttt{1001}\}$, OR). This predicate is the lone example for $k = 4$ of a PCSP whose hardness is not captured by any of the other hardness conditions (Theorems 5.16, 5.17 and 5.18). The inspiration for Theorem 5.21 came directly from studying the polymorphisms of this PCSP.

6 Results for Small Arities

In this section we use the tools developed to classify predicates of small arities. It turns out that we are able to completely characterize all predicates of arity $k \leq 4$. For arity $k = 5$, we are able to classify the vast majority of predicates but there is a small fraction ($\approx 10^{-5}$) of predicates left unclassified by the conditions in Section 5. We do not study the case of $k \geq 6$ due to the large number of different predicates which makes them difficult to study individually even by computer.

There are several algorithmic aspects to consider when applying our algorithm and hardness conditions to a given Promise-SAT problem. These are discussed in some detail in Appendix A.

6.1 Results for Promise-SAT (with idempotence)

Let us start by discussing the complexity of fiPCSP(A, OR) for all predicates of arity $k \leq 5$.

6.1.1 Equivalence Classes

Since the OR predicate is symmetric, taking a predicate A and permuting the k input bits has no effect on the complexity of fiPCSP(A, OR). Thus any two predicates which differ only by such a permutation are considered equivalent, and we only consider one

predicate from each equivalence class. Thus while the total number of predicates of arity k equals $2^{2^k-1}-1$ (one -1 comes from A not accepting the all-0 string, the other from A being non-empty), the total number of equivalence classes is approximately $\frac{2^{2^k-1}}{k!}$ (but a bit larger because some equivalence classes are smaller than $k!$).

The representative of each equivalence class is chosen as follows. Each A is naturally identified by the integer $r(A) = \sum_{x \in A} 2^{b(x)}$, where $b(x)$ means that we interpret $x \in \{0,1\}^k$ as an integer in base 2. E.g., the predicate $A = \{\mathtt{001}, \mathtt{011}\}$ on three bits would be identified by $r(A) = 2^1 + 2^3 = 10$. For each equivalence class of predicates we choose the member for which this identifier is minimized as the representative of the class. For example, in the case of $k = 2$, we have 5 different equivalence classes. The representatives of these are $\{\mathtt{01}\}, \{\mathtt{01}, \mathtt{10}\}, \{\mathtt{11}\}, \{\mathtt{01}, \mathtt{11}\}, \{\mathtt{01}, \mathtt{10}, \mathtt{11}\}$.

6.1.2 Summary

It turns out that for $k \leq 4$, we are able to determine the complexity of fiPCSP(A, OR) for all A while for arity $k = 5$, we are able to classify all except 189 out of the over 18.6 million predicates. A summary is given in Table 3.

Table 3: Summary of classification of complexity of fiPCSP(A, OR) for A of arity up to 5.

	Total	Tractable	NP-hard	Unknown
$k = 2$	5	5	0	0
$k = 3$	39	33	6	0
$k = 4$	1 991	956	1 035	0
$k = 5$	18 666 623	1 290 862	17 375 572	189

For the tractable predicates, it is interesting to analyze which families of block-symmetric polymorphisms are admitted by fiPCSP(A, OR). As discussed in Section 4 we have only seen five families of polymorphisms that enable us to apply the BLP+AIP algorithm. The relative frequencies of the five families are given in Table 4. It is interesting to note that majority is, by a huge margin, the most useful polymorphism.

Table 4: Overview of sources of tractability for fiPCSP(A, OR). The number after the slash is the count of predicates that admit the corresponding family of polymorphisms. The number before the slash is the count of predicates that *only* admit this family and none of the other four families.

	Maj	**Par**	**AT**	id$\overline{\textbf{Maj}}$	id$\overline{\textbf{Par}}$
$k = 2$	1/5	0/4	0/4	0/4	0/4
$k = 3$	13/31	1/19	0/18	0/17	0/17
$k = 4$	720/915	31/219	0/172	1/163	0/162
$k = 5$	1 267 621/1 282 927	7 259/21 557	15/11 485	260/11 970	11/11 182

For the hardness results, we have several different conditions which guarantee small fixing assignments. The frequencies at which they apply are shown in Table 5. It may seem surprising that all four conditions give essentially the same set of hard predicates, but this is because most predicates are hard for many reasons. A perhaps more fair

picture is given by only looking at the minimal hard A's. Note also that Theorem 5.18 (for the unate case) does not yield any hardness results that are not also given by one of the three other results. This is not surprising as the more complicated Theorem 5.21 was designed to provide stronger hardness in the unate case. However we are not aware of a proof that Theorem 5.21 covers all results implied by Theorem 5.18 so we include also the latter in the overview.

Table 5: An overview of all known NP-hard predicates for each k. The table count of how many different predicates that satisfies each category. The second number of each pair is the number of predicates that satisfy this condition while the first only counts the number of predicates that satisfy no other condition.

	Theorem 5.16	Theorem 5.17	Theorem 5.18	Theorem 5.21
$k = 2$	0/0	0/0	0/0	0/0
$k = 3$	0/6	0/6	0/6	0/6
$k = 4$	2/1 029	0/1 031	0/1 030	1/1 032
$k = 5$	409/17 368 311	93/17 373 401	0/17 371 388	687/17 355 043

6.1.3 Detailed results

Let us now provide a more detailed inspection of the tractable and NP-hard predicates. For each arity up to 4, we describe the minimal NP-hard, and maximal tractable predicates. Minimal/maximal here means that if we remove/add one satisfying assignment from/to that predicate, then the complexity for the predicate changes. By the monotonicity of fiPCSP(A, OR) (Fact 2.4), the complexity of any other predicate can be derived from these two lists of minimal NP-hard and maximal tractable predicates.

For the tractable predicates, we indicate which families of polymorphisms enable applying the BLP+AIP algorithm for this predicate, and for the NP-hard predicates we indicate which of the different hardness conditions yields the hardness result.

Arity 2. In the case of $k = 2$, the situation is very simple as all 5 predicates are tractable – they can be solved by 2-SAT. In polymorphism terminology they all allow the **Maj** family.

Arity 3. For $k = 3$ there are a total of 39 non-equivalent predicates of which 33 are tractable and 6 are NP-hard. There are two maximal easy predicates and these are given in Table 6. A $*$ indicates that this coordinate can take any value. It is easy to see that the first predicate is parity and the second 2-SAT in the first two variables. There is a single minimal NP-hard predicate, given in Table 7. It is "1-in-3-SAT" augmented with a single assignment of weight two.

Table 6: A list of the two maximal tractable predicates for $k = 3$. The second number in the last column indicates the number of tractable predicates that are subsets of this predicate. The first number counts the number of such predicates not subset of any other maximal tractible predicate.

	Predicate	Maj	Par	AT	id$\overline{\text{Maj}}$	id$\overline{\text{Par}}$	Dep.
1.	$\{001, 010, 100, 111\}$		✓				2/7
2.	$\{*01, *10, *11\}$	✓					26/31

Table 7: The only minimal NP-hard predicate for $k = 3$.

Predicate	5.16	5.17	5.18	5.21
$\{001, 010, 011, 100\}$	✓	✓	✓	✓

Arity 4. For $k = 4$ there are a total of 1991 predicates of which 956 are tractable and 1035 are NP-hard. There are six maximal tractable predicates, given in Table 8. Easy to recognize predicates are parity of all variables (predicate number 3) and parity of the first three variables (number 4). Our old friend 2-SAT is present as number 5 while the sixth predicate is (non-strict) majority of all four variables. The first two predicates are less standard. There are 14 minimal NP-hard predicates of arity 4, given in Table 9. These are more difficult to name and we indicate what properties we can establish of the polymorphisms.

Table 8: A list of the 6 maximal tractable predicates for $k = 4$. The last column is as in Table 6 and the $*$s indicated that this position is free to take any value.

	Predicate	Maj	Par	AT	id$\overline{\text{Maj}}$	id$\overline{\text{Par}}$	Dep.
1.	$\{0011, 0101, 0110, 1000\}$			✓		✓	1/7
2.	$\{0011, 0100, 0110, 1000, 1001\}$				✓		1/17
3.	$\{0001, 0010, 0100, 0111, 1000, 1011, 1101, 1110\}$		✓				11/34
4.	$\{*001, *010, *100, *111\}$		✓				24/79
5.	$\{**01, **10, **11\}$	✓					684/905
6.	$\{0011, 0101, 0110, 0111, 1001, 1010, 1011, 1100, 1101, 1110, 1111\}$	✓					10/179

Table 9: A list of the 14 minimal NP-hard predicates for $k = 4$. The last column called dependency lists how many predicates are NP-hard by direct implication and how many of these which are not implied by any other predicate.

	Predicate	5.16	5.17	5.18	5.21	Dep
1.	$\{0001, 0010, 0011, 0100\}$	✓	✓	✓	✓	45/570
2.	$\{0001, 0011, 0101, 0110, 1000\}$		✓	✓	✓	1/360

Continued on next page

Continued from previous page

	Predicate	5.16	5.17	5.18	5.21	Dep
3.	{0011, 0100, 0111, 1000}	✓	✓	✓	✓	36/694
4.	{0011, 0101, 0110, 0111, 1000}	✓				1/240
5.	{0010, 0100, 0110, 1001}	✓	✓	✓	✓	13/520
6.	{0010, 0011, 0100, 0111, 1001}	✓	✓	✓	✓	5/560
7.	{0011, 0101, 0110, 1000, 1001}				✓	3/360
8.	{0010, 0100, 0111, 1000, 1001}	✓	✓			1/398
9.	{0011, 0100, 0111, 1001, 1010}	✓	✓	✓	✓	3/372
10.	{0001, 0010, 0111, 1011, 1100}	✓	✓	✓	✓	2/304
11.	{0001, 0010, 0100, 1000, 1111}	✓				1/90
12.	{0011, 0101, 0110, 1000, 1111}	✓	✓	✓	✓	1/220
13.	{0001, 0010, 0111, 1000, 1111}	✓	✓	✓	✓	6/279
14.	{0011, 0100, 0110, 1000, 1001, 1111}	✓	✓	✓	✓	1/200

Arity 5. For $k = 5$ there are a total of 18 666 623 predicates of which 1 290 862 are tractable, 17 375 572 are NP-hard and 189 are unknown. There are 32 maximal tractable predicates, described in detail in Appendix B.1. There is a rather large number of 241 minimal hard predicates and we refrain from listing these. Of more interest are the minimal and maximal of the 189 unknown predicates. There are 25 and 19 such predicates, respectively, and they can be found in Appendix B.2. It is also interesting to look at the distribution of tractable vs NP-hard predicates as a function of the number of satisfying assignments, and this is shown in Figure 2.

6.2 Results for Promise-SAT (without idempotence)

Let us now turn to the non-idempotent setting and the complexity of $\mathrm{fPCSP}(A, \mathsf{OR})$. Here we use Lemma 3.5, which as discussed in Section 3.1 effectively says that the complexity of $\mathrm{fPCSP}(A, \mathsf{OR})$ is captured by either $\mathrm{fiPCSP}(A, \mathsf{OR})$ or $\mathrm{fiPCSP}(A \oplus 1^k, \mathsf{OR})$, whichever is easier (with the second option only being possible when $1^k \notin A$).

In this setting, there is an additional symmetry causing the equivalence classes of predicates to be larger: if $1^k \notin A$, then the complexity of $\mathrm{fPCSP}(A, \mathsf{OR})$ is the same as the complexity of $\mathrm{fPCSP}(A \oplus 1^k, \mathsf{OR})$ and thus we consider the two predicates A and $A \oplus 1^k$ equivalent. As a result the total number of non-equivalent predicates is approximately 25% smaller in this setting than in the fiPCSP setting (approximately half the predicates A satisfy $1^k \notin A$ and these are paired up in equivalent pairs by this additional symmetry).

Using Lemma 3.5 together with our classification of $\mathrm{fiPCSP}(A, \mathsf{OR})$, it is straightforward to obtain a similar classification for $\mathrm{fPCSP}(A, \mathsf{OR})$. For brevity we only give a summary in Table 10 and refrain from performing a deeper dive into the data, as it looks largely similar to the idempotent case in Section 6.1. However one interesting detail to note is that the relative number of unknown predicates for arity 5 is noticeably smaller here than in Section 6.1. In other words many of the unknown predicates A for $\mathrm{fiPCSP}(A, \mathsf{OR})$ are such that $\mathrm{fPCSP}(A, \mathsf{OR})$ is easy by virtue of having non-idempotent block-symmetric polymorphisms (in particular either $\overline{\mathsf{Maj}}$ or $\overline{\mathsf{Par}}$).

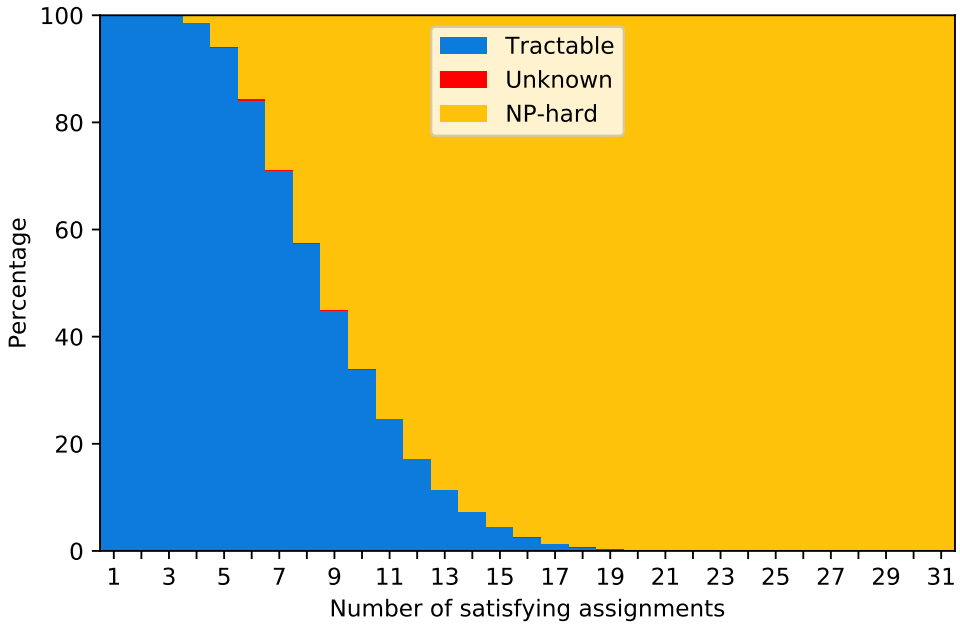

Figure 2: Distribution of predicates A of arity 5 such that fiPCSP(A, OR) is tractable or NP-hard, by weight. The number of unknown predicates is very small but exists for weights between 6 and 12 (both inclusive), and can be glimpsed as thin red lines at weights 6, 7, and 8.

Table 10: Summary of classification of complexity of fPCSP(A, OR) for A of arity up to 5.

	Total	Tractable	NP-hard	Unknown
$k = 2$	5	5	0	0
$k = 3$	32	28	4	0
$k = 4$	1 549	848	701	0
$k = 5$	14 003 603	1 253 003	12 750 541	59

6.3 Results for Promise-Usefulness

Finally let us consider the classification of which predicates A are promise-useful. As shown in Corollary 3.6, the two á priori possibly different notions of fiPCSP-usefulness and fPCSP-usefulness cannot be distinguished by BLP+AIP or small fixing assignments, so the data we provide here applies for both fiPCSP-usefulness and fPCSP-usefulness.

After having classified the complexity of fiPCSP(A, OR) for all (or most) A, Lemma 3.2 gives a simple procedure to tell if a predicate A is fiPCSP-useful or fiPCSP-useless. It states that A is fiPCSP-useful if and only if there exists $b \notin A$ such that fiPCSP$(A \oplus b, \mathsf{OR})$ is tractable. If for a single b this is the case we can conclude that A is promise useful. On the other hand we need to establish that it is NP-hard for all possible b to conclude that A is promise-useless. This enables us to fully classify which predicates are promise-useful for $k \leq 4$ while for $k = 5$ the complexity of several predicates remains unknown.

In addition to the already applied symmetry of the input bits (Section 6.1.1), there is now additional symmetry – clearly from the characterization above it follows that A and

$A \oplus b$ are equivalent (w.r.t. promise-usefulness) for any $b \notin A$. Thus the total number of (equivalence classes of) predicates is smaller here than in Section 6.1 by a factor of roughly 2^{k-1} (since this is the typical number of $b \notin A$).

Table 11 gives an overview of the number of promise-useful and promise-useless (and unknown) predicates of arities up to 5.

Table 11: Summary of classification of promise-usefulness of predicates A of arity up to 5.

	Total	Useful	Useless	Unknown
$k = 2$	4	4	0	0
$k = 3$	20	16	4	0
$k = 4$	400	230	170	0
$k = 5$	1 228 156	156 135	1 071 962	59

Arity 2. Again for $k = 2$ all predicates are useful as they are indeed useful for OR.

Arity 3. For $k = 3$ it is again parity and 2-SAT that give the maximal useful predicates as indicated in Table 12. We also have two minimally useless predicates as indicated by Table 13.

Table 12: A list of the two maximal promise-useful predicates for $k = 3$.

	Predicate	Maj	Par	Dep.
1.	{00*, 01*, 10*}	✓		12/15
2.	{000, 011, 101, 110}		✓	1/4

Table 13: A list of the two minimal promise-useless predicates for $k = 3$.

	Predicate	Dep.
1.	{000, 001, 011, 101, 110}	2/3
2.	{001, 010, 011, 100, 101, 110}	1/2

Arity 4. The maximal useful predicates for $k = 4$ are shown in Table 14. The two first maximal predicates correspond to 2-SAT and (non-strict) majority of 4 variables (recall that we can negate variables – in our ordering the representatives chosen are the negated versions of these two predicates). The other two predicates are parity constraints on three or four variables. It turns out that there are 5 minimal useless predicates of arity 4 and these are given in Table 15.

Table 14: A list of the 4 maximal promise-useful predicates for $k = 4$.

	Predicate	Maj	Par	Dep.
1.	{00**, 01**, 10**}	✓		91/216

Continued on next page

Continued from previous page

	Predicate	Maj	Par	Dep.
2.	{0000, 0001, 0010, 0011, 0100, 0101, 0110, 1000, 1001, 1010, 1100}	✓		8/132
3.	{000*, 011*, 101*, 110*}		✓	3/21
4.	{0001, 0010, 0100, 0111, 1000, 1011, 1101, 1110}		✓	1/15

Table 15: A list of the 5 minimal promise-useless predicates for $k = 4$.

	Predicate	Dep.
1.	{0000, 0111, 1001, 1010, 1100}	19/132
2.	{0001, 0010, 0011, 0111, 1001, 1010, 1100}	3/105
3.	{0011, 0100, 0111, 1001, 1010, 1100}	10/138
4.	{0011, 0101, 0110, 0111, 1000, 1001, 1010, 1100}	1/39
5.	{0011, 0100, 0110, 0111, 1000, 1001, 1011, 1100}	1/29

Arity 5. For $k = 5$ we have 6 maximal useful predicates shown in Table 16 – the 4 maximal predicates of arity 4, and two obvious generalizations of them to arity 5. Note that the second predicate does not quite correspond to a simply majority predicate but is a bit more complicated, whereas the last predicate is simply even parity on all five variables. There is a large number of 73 minimal promise-useless predicates and we refrain from listing these. Of more interest are the minimal and maximal predicates whose promise-usefulness we are unable to classify. There are 9 and 7 of these, respectively, and they can be found in Appendix B.3.

Table 16: A list of the 6 maximal promise-useful predicates for $k = 5$.

	Predicate	Maj	Par	Dep.
1.	{00***, 01***, 10***}	✓		91 485/141 063
2.	{00000, 00001, 00010, 00011, 00100, 00101, 00110, 00111, 01000, 01001, 01010, 01011, 01100, 01101, 01110, 10000, 10001, 10010, 10100, 11000}	✓		355/30 027
3.	{0000*, 0001*, 0010*, 0011*, 0100*, 0101*, 0110*, 1000*, 1001*, 1010*, 1100*}	✓		12 549/60 975
4.	{000**, 011**, 101**, 110**}		✓	205/645
5.	{0001*, 0010*, 0100*, 0111*, 1000*, 1011*, 1101*, 1110*}		✓	58/458
6.	{00000, 00011, 00101, 00110, 01001, 01010, 01100, 01111, 10001, 10010, 10100, 10111, 11000, 11011, 11101, 11110}		✓	50/150

7 Asymptotic Bounds for Large Arities

In this section, we study promise-usefulness and fiPCSP(A, OR) for predicates A of large arity k, and show that almost all such A, even with a relatively small number of satisfying assignments, are hard.

Let $\mathcal{A}_{k,p}$ denote the distribution over a random predicate A on $\{0,1\}^k$, where each $x \in \{0,1\}^k$ is included in A with probability p, independently. We condition our results on A not containing 0^k. As the p we consider is very small this is a very mild conditioning but allowing this to happen makes some of our arguments more uniform, avoiding special cases.

Theorem 7.1. *We have*

$$\Pr_{A\sim\mathcal{A}_{k,p}}\left[\text{fPCSP}(A,\mathsf{OR}) \text{ admits a non-dictator} \,|\, 0^k \notin A\right] \leq O\left(2^k(e^{-p2^{k/6}} + e^{-p^2 2^k})\right).$$

In particular, for p is slightly larger than $2^{-k/6}$, a random predicate A from $\mathcal{A}_{k,p}$ is NP-hard with high probability as $k \to \infty$. It is not unlikely that the value for p can be improved but currently we see no way of establishing hardness in the very sparse regime with a p of the form $2^{o(k)-k}$, i.e., for predicates accepting only $2^{o(k)}$ strings.

The problem is that very sparse predicates allow for all polymorphisms of small arity. For example, suppose A contains K strings, and we are interested in polymorphisms of arity 6. In this case there are K^6 possible obstruction matrices, each being, except for repetitions, totally random. The probability that an individual random matrix lacks any fixed row, say 000001 is at most $(63/64)^k$. Thus if $K = 2^{o(k)}$ all rows are likely to be present in all matrices and thus all arity 6 functions are polymorphisms. This implies that to get results for this small range of ps we need to analyze polymorphisms of large arity and this seems to require new methods.

Let us turn to the proof of Theorem 7.1 and first prove that it is enough to study polymorphisms of arity 6.

Claim 7.2. *Let $\mathcal{M}$ be a minion containing a non-dictator function $f : \{0,1\}^\ell \to \{0,1\}$. Then $\mathcal{M}$ contains a non-dictator function $g : \{0,1\}^6 \to \{0,1\}$ of arity 6.*

Proof. A function is a non-dictator iff it depends on at least two variables and hence there are inputs x and y and $i \neq j$ such that $f(x) \neq f(x \oplus e_i)$ and $f(y) \neq f(y \oplus e_j)$. We can divide the variables outside indices i and j into four groups depending on the value of the pair (x_k, y_k). Identify the variables inside a group by a single variable. These four variables jointly with i and j gives a six variable minor of f which is not a dictator. □

Next we have the following observation.

Lemma 7.3. *Let $S \subseteq [k]$ be of cardinality L. The probability that there is some assignment α on S such that A contains no string that is equal to α on S is bounded by $2^L e^{-p2^{k-L}}$.*

Proof. We do a union bound over different values of α (resulting in the factor 2^L). Once we fix α, there are 2^{k-L} candidate strings that that are equal to α on S. The probability that none of them belongs to A is $(1-p)^{2^{k-L}} \leq e^{-p2^{k-L}}$. □

We need one more simple fact.

Lemma 7.4. *Fix an $\alpha \in \{0,1\}^k$, $\alpha \neq 0^k$. The probability that A does not contain two strings a and b such that $a \oplus b = \alpha$ is at most $e^{-p^2 2^{k-1}}$.*

Proof. There are 2^{k-1} possible pairs (a, b) with the given property. Each pair is in A with probability p^2 and as they are disjoint, the events are independent, and the lemma follows by a simple calculation. □

Proof of Theorem 7.1. By Claim 7.2, it suffices to prove that with high probability, $A \sim \mathcal{A}_{k,p}$ has no non-dictator polymorphisms of arity 6. Let f be a non-dictator function of arity 6 and let us construct an obstruction matrix for it. Let us first observe that f, for each j either can written as $x_j \oplus g(x')$ where x' gives the other five variables or there is some input $x^{(j)}$ such that $f(x^{(j)}) = f(x^{(j)} \oplus e_j) = 0$. Using this observation and permuting the variables we can assume that $f(x) = \oplus_{i=1}^{t} x_i \oplus g(x')$ where x' corresponds to the variables $x_{t+1} \dots x_6$ and we have assignments $x^{(j)}$ such that $f(x^{(j)}) = f(x^{(j)} \oplus e_j) = 0$ for $t + 1 \leq j \leq 6$.

If g is constant then f is a parity function. In this case t is odd size since f is folded. If $t = 1$ then f is a dictator and there is nothing to prove. If $t = 5$ we can make columns equal reducing to the case $t = 3$. In this case, fix the first column to any element of A. Then, since we condition on $0^k \notin A$, Lemma 7.4 implies that we can, with high probability, pick the second and third columns to complete the obstruction matrix. Even using a union bound over all values of the first column, the failure probability is bounded by $2^k e^{-p^2 2^{k-1}}$.

Now suppose that g is non-constant. We construct an obstruction matrix and suppose for notational convenience that $k/6$ is an integer. For each value $1 \leq j \leq 6$ we assign $k/6$ rows. Let y and z be such that $g(y) = 0$ and $g(z) = 1$. Now for $1 \leq j \leq t$ we assign $k/6$ rows that are either e_j followed by z or 0^t followed by y, For $t + 1 \leq j \leq 6$ we have $k/6$ rows that are either $x^{(j)}$ or $x^{(j)} \oplus e_j$ where $x^{(j)}$ is as discussed above. Note that, by construction, f evaluates to 0 on each row and we claim that, with high probability, we can make the choices such that each column belongs to A.

Let us first make the choices for the first t columns. Note for these columns, all values outside the $k/6$ rows that belongs the column itself are independent of the choices for other rows as the two alternatives give equal values in this coordinate. Hence using Lemma 7.3, except with probability $2^{5k/6} e^{-p 2^{k/6}}$ we can make choices to make this column belong to A.

The argument can now be repeated for the remaining $6 - j$ columns. The values outside its own rows are now fixed and we can apply Lemma 7.3. □

Remark 7.5. If we do not allow negations (and hence are interested in f which are not folded), then the equivalent of Theorem 7.1 still holds. In this situation we need to study PCSP(A, NAE). In the above proof we used twice that a potential polymorphism is folded. Firstly to conclude that if f is a parity then this is of odd size. It is, however, straightforward to deal with parities of even size. We also used the fact that there are inputs $x^{(j)}$ such that $f(x^{(j)}) = f(x^{(j)} \oplus e_j) = 0$. This is no longer true if f is not folded. The corresponding fact is that f takes the value b on at least half the inputs then for each j it is possible to find a $x^{(j)}$ such that $f(x^{(j)}) = f(x^{(j)} \oplus e_j) = b$. It is then possible to find an obstruction matrix where the output is b^k. The rest of the proof is unchanged. We omit the details.

As observed above the probability that A is useful for OR is bounded by $2^{-\omega(k)}$ whenever $p = \omega(k 2^{-k/6})$. The analysis that A is useful for any $\mathsf{OR} \oplus b$ is identical and as there are only 2^k possible values for b we conclude that any such A is useless except with probability $2^{-\omega(k)}$. We record this as a corollary.

Corollary 7.6. *Suppose A is selected according to $\mathcal{A}_{k,p}$. If $p = \omega(k2^{-k/6})$ then, except with probability $2^{-\omega(k)}$, we have that A is promise-useless. This is true with or without negations, as well as with or without allowed constants.*

We next proceed to prove BLP+AIP is only applicable to very sparse random predicates.

7.1 Threshold for the BLP+AIP Algorithm

In this section it is easier to work in ± 1-notation with -1 taking the role of 1. We start with a definition.

Definition 7.7. Let $\epsilon > 0$. A set, $A \subseteq \{-1,1\}^k$ is ϵ-somewhat spread if for any unit vector u there is an $a \in A$ such that $\langle a, u\rangle \leq -\epsilon$.

We get an immediate consequence.

Lemma 7.8. *Suppose $\|v\| \geq k/\epsilon \geq 1$ and that $A \subseteq \{-1,1\}^k$ is ϵ-somewhat spread. Then there is an $a \in A$ such that $\|v + a\|^2 \leq \|v\|^2 - k$.*

Proof. Take a vector in a with $\langle a, v\rangle \leq -\epsilon\|v\|$. Then

$$\langle v + a, v + a\rangle = \langle v, v\rangle + 2\langle a, v\rangle + \langle a, a\rangle \leq \|v\|^2 - 2\epsilon\|v\| + k \leq \|v\|^2 - k.$$

□

We are interested in a second property of vectors and in order to formulate this we need to study integer lattices. We start by recalling some basic definitions. We do not state definitions in full generality and for a more careful treatment we refer to a text-book such as [MG02]. In our case we are interested in lattices that are subsets of $\mathbb{Z}^n$ and let L be a generic lattice. It is usually given by a basis, $(b_i)_{i=1}^n$ with integer coordinates and L is given by the points $\sum_{i=1}^n a_i b_i$ where $a_i \in \mathbb{Z}$. In a basis the vectors are linearly independent over $\mathbb{R}$ but it is possible to consider a larger set of possibly linearly dependent vectors and use them as a generating set. The set of all integer combinations of such a set is also a lattice and it is easy to compute a basis for this lattice given the generators.

The determinant, $\det(L)$, of L is the absolute value of the determinant of the matrix which has the basis vectors as columns. As we are considering only lattices which are subsets of $\mathbb{Z}^n$ a lattice can be described by a set of modular equations. The system can be written as $\sum_{j=1}^n a_{ij}x_j \equiv 0$ modulo $p_i^{m_i}$ where p_i are (not necessarily distinct) prime numbers and $m_i \geq 1$. The equations can be chosen such that $\prod_i p_i^{m_i} = \det(L)$.

For any $a \in A$ we create a vector a' of dimension $k+1$ by adding an additional coordinate that always takes the value 1. Let L be the lattice in $k+1$ dimension that is generated by these vectors. We say that A is *modular free* if L consists of all integer vectors where all coordinates have the same parity. As all a' in the generating set have this property property this is as large as L can be. The two defined properties just defined turn out to imply that A does not have any (block)-symmetric polymorphisms by the following lemma.

Lemma 7.9. *Suppose $A \subseteq \{-1,1\}^k$ is modular free and ϵ-somewhat spread for some $\epsilon > 0$. Then for all $x \in \{-1,1\}^k$, and all sufficiently large odd integers ℓ, the vector $(x_1, \ldots, x_k, \ell)$ can we written as a non-negative integer combination of a' where $a \in A$.*

Before proving this lemma we give the important corollary.

Corollary 7.10. *Suppose A is modular-free and ϵ-somewhat spread then for all sufficiently large odd integers ℓ,* $\mathrm{fPCSP}(A, \mathsf{OR})$ *does not admit a block-symmetric polymorphism with block sizes ℓ and $\ell+1$.*

Proof. Let x be an arbitrary element of A. By Lemma 7.9, the vectors $(1^k, \ell)$, $((-1)^k, \ell)$ and $(-x, \ell)$ can all be written as non-negative combinations of a' for $a \in A$. Clearly then also $(0^k, \ell+1)$ can be written as such a combination.

A block-symmetric function $f : \{-1,1\}^{2\ell+1} \to \{-1,1\}$ with block sizes ℓ and $\ell+1$ can be written as $f(x) = g(\sum_{i=1}^{\ell} x_i, \sum_{i=\ell+1}^{2\ell+1} x_i)$. Since $(1^k, \ell)$, $((-1)^k, \ell)$ and $(0^k, \ell+1)$ can all be written as non-negative integer combinations as described above, we can create two $k \times (2\ell+1)$ matrices $M^{(+)}$ and $M^{(-)}$ such that $f(x,y) = g(1,0)$ for all rows of $M^{(+)}$ and $f(x,y) = g(-1,0)$ for all rows of $M^{(-)}$. However if f is folded, one of $g(1,0)$ and $g(-1,0)$ is false, yielding an obstruction to f being a polymorphism of $\mathrm{fPCSP}(A, \mathsf{OR})$. □

Proof of Lemma 7.9. By the assumption of being modular-free we know that any integer vector of the form $(v, 0)$ where all components of v are even can be written in form $\sum_{a \in A} b_a a'$ for some integers b_a. Let M be the maximal absolute value needed as a coefficient to represent any vector v with norm at most $2k/\epsilon$. Define $w = M \sum_{a \in A} a'$ and notice that $\|w\| \leq k 2^k M$. Using Lemma 7.8 we can add additional vectors a' with $a \in A$ to w such that for all sufficiently large odd ℓ we can get a vector of the form (v, ℓ) where the length of v is bounded by k/ϵ. Using that $(x - v, 0)$ can be written as an integer combination of a' with coefficients bounded by M we conclude that (x, ℓ) can be written as a positive integer combination of a' with $a \in A$. This follows as any coefficient of w already is at least M preventing it from turning negative. □

We need to prove that a random A has the two properties. On road to this we have the following definition.

Definition 7.11. A set $B \subseteq \mathbb{R}$ of unit vectors is an ϵ-net if for each unit vector $v \in \mathbb{R}^k$ there is $b \in B$ such that $\|b - v\| \leq \epsilon$.

It is well known that it is possible to construct an ϵ-net of cardinality $(1/\epsilon)^{O(k)}$ [Koc94]. We want to prove that if A has Ck random elements for a sufficiently large value of C, then it is very likely to be c-somewhat spread for an absolute constant c. Before embarking on the proof let us point out that the result is straightforward if A has $Ck \log k$ elements.

Let us consider an $1/4\sqrt{k}$-net B. It has $2^{O(k \log k)}$ elements. Take any $b \in B$. The probability that $\langle a, b \rangle \leq -1/2$ for a random a is easily seen to be $\Omega(1)$. By the union bound, for a sufficiently large constant C, if we pick $Ck \log k$ random a, then, with high probability, for each $b \in B$ there exists and $a \in A$ such that $\langle a, b \rangle \leq -1/2$. This implies that for any unit vector u, if b is the closest element of B, then $\langle a, u \rangle \leq -1/4$ for any a such that $\langle a, b \rangle \leq -1/2$. To make do with $O(k)$ vectors in A we need a more complicated argument and we use the techniques of [AH11]. Let us recall the highlights.

We choose B to be an ϵ-not for a small, but absolute constant ϵ specified below. Let K be the convex body $\sum_a c_a a'$ where $c_a \in [0,1]$ and the sum is over $a \in A$. For any direction $b \in B$ we look at the number $m_b = \min_{x \in K} \langle b, x \rangle$ and $M_b = \max_{x \in K} \langle b, x \rangle$. By inspection we have that $m_b = \sum_a \min(0, \langle b, a' \rangle)$. When picking a random a we have that $\langle a', b \rangle$ is a symmetric random variable with mean 0 and variance 1. It is not difficult to see that the expectation of $|\langle a, b \rangle|$ is e_b, where there are absolute constants, m and M such that $0 < m \leq e_b \leq M$ for any unit vector b. By symmetry it follows that $E[\min(0, \langle b, x \rangle] = -e_b/2$.

Set $\epsilon = m/16M$ and note that $\epsilon < 1/16$. Using standard Chernoff estimates it follows that, for sufficiently large constant C, with high probability when A has Ck random vectors, then $m_b \leq -Cke_b/4$ and $M_b \leq Cke_b$ for any $b \in B$. Suppose this is the case. We have some simple claims.

Claim 7.12. *For any $x \in K$ we have $\|x\| \leq 2CkM$.*

Proof. Take any vector $x \in K$ and let $b \in B$ be as close as possible to $x/\|x\|$. It is not difficult to see that $\langle b, x\rangle \geq \|x\|/2$ and the claim follows by the bound on M_b. □

Claim 7.13. *For any unit vector u we have $m_u \leq -mCk/8$.*

Proof. As B is an ϵ-net we can write $u = b + w$ where $b \in B$ and $\|w\| \leq \epsilon$. Take $x \in K$ that minimizes $\langle b, x\rangle$, then

$$\langle u, x\rangle = \langle b, x\rangle + \langle w, x\rangle \leq -mCk/4 + \|w\|\|x\| \leq -mCk/4 + 2CkM\epsilon \leq -mCk/8.$$

□

As $m_u = \sum_{a\in A} \min(0, \langle u, a'\rangle)$ and the sum has Ck terms it follows from Claim 7.13 that A is $m/8$ well spread and we turn to the question of A being modular free.

We start by establishing that we get a full dimensional lattice.

Lemma 7.14. *Suppose A contains $3k$ random elements, then there is a constant $c < 0$ such that with probability $1 - 2^{-ck}$ the vectors a' for $a \in A$ span $\mathbb{R}^{k+1}$.*

Proof. We select the elements of A one by one. We claim that if the corresponding vectors a' chosen so far do not span $\mathbb{R}^{k+1}$ then with probability at least $1/2$ the next vector is linearly independent of the vectors so far. This follows as any $k-1$-dimensional space in $\mathbb{R}^k$ only contains at most half the points of the hypercube. This implies that the probability of not spanning the entire space is at most the probability of getting $k-1$ zeroes in a random binary string of length $3k$. It is well known that the probability of this event is as stated in the lemma. □

The first time the vectors a' span $\mathbb{R}^{k+1}$ we can consider the full dimensional lattice, L^0 spanned by these vectors. Hadamard's inequality, saying that the determinant of a matrix is at most the product of the lengths of the row vectors, implies that the determinant of L^0 is at most $(k+1)^{(k+1)/2}$. Let L^t be the lattice where we have added an additional t elements to A. Suppose L^0 is defined by the equations $\sum_j a_{ij}x_j = 0$ modulo $p_i^{m_i}$ for $i = 1, \ldots r$. By the bound on the determinant of L^0 we know that $r \leq k \log k$. We first treat powers of prime different from 2.

Lemma 7.15. *Fix any prime $p > 2$ that divides $det(L^0)$. The probability that p divides $det(L^{3k})$ is bounded by 2^{-ck} for some constant $c > 0$.*

Proof. Each time we add a vector to A, the probability that a' satisfies any existing non-trivial relation modulo p is bounded by $1/2$. To see this, assume without loss of generality the relations depends on $x - 1$. Then, fixing all coordinate of a except the first, only one of the two possible values for a_1 makes the added vector satisfy the relation. We have at most $k-1$ linearly independent equations modulo p satisfied by L^0. Thus for p to divide $det(L^{3k})$ an event that happens with probability $1/2$ must happen at least $2k$ times out of $3k$ possibilities. This is the same event as we considered in the proof of the previous lemma. □

The situation for $p = 2$ is slightly different as each vector added has only odd coordinates. Instead of ruling out relations modulo 2 we need to rule out relations modulo 4 but nothing essential differs. We omit the details.

We sum up the above reasoning in a theorem.

Theorem 7.16. *Suppose A is selected according to $\mathcal{A}_{k,p}$. If $p \geq Ck2^{-k}$, then, for a sufficiently large value of C, the probability that BLP+AIP can be used to solve* fiPCSP(A, OR) *is* $2^{-\Omega(k)}$.

8 Concluding Remarks

We have introduced the notion of promise-usefulness, and studied this in the folded Boolean setting together with the closely related Promise-SAT problem. While many questions remain on the road to a complete characterization, our results are strong enough to characterize almost all predicates of arity up to five, as well as the vast majority of all predicates of large arity.

The most obvious open problem is of course to obtain a complete classification of promise-usefulness and Promise-SAT (a classification of the latter yields a classification of the former, but not necessarily the other way around), and beyond that Boolean PCSPs in general.

While we only studied the folded setting in this paper, the non-folded setting also warrants investigation. When it comes to the distinction between the idempotent and non-idempotent setting, the current techniques do not distinguish between them, and it would be interesting to understand if this is true in general or not (see discussion in Section 3.1).

The algorithmic results in this paper are applications of the known characterization of the BLP+AIP algorithm. A key question here is of course whether there are tractable cases of Promise-SAT (or again, more generally Boolean PCSP) which are not solvable by BLP+AIP. Less ambitiously, it is not clear if there are examples of Promise-SAT problems solvable by BLP+AIP but requiring other block-symmetric polymorphisms than one of the five families used in this paper. If such examples turn out not to exist, then the simple sufficient conditions of Theorem 4.12 are an exact characterization of what can be shown promise-useful using BLP+AIP. The evidence for such a nice state of affairs is not strong at this point but it is a possibility.

The hardness results are more involved, and we are quite certain that these can be both improved and simplified. Among the four conditions for a minion having small fixing assignments, Theorems 5.16 to 5.18 and 5.21, the first three are in our opinion relatively natural. However, the fourth result based on UnCADA and UnDADA, Theorem 5.21, is different. It was specifically tailored to show the hardness of fiPCSP$(\{\mathtt{0011}, \mathtt{0101}, \mathtt{0110}, \mathtt{1000}, \mathtt{1001}\}, \mathsf{OR})$ as mentioned in Example 5.23. The issue with Theorem 5.21 is that the two families of polymorphisms involved, UnCADA and UnDADA, are somewhat ad-hoc, and from a practical viewpoint they have relatively high arity making them computationally expensive to test for. It is highly doubtful that the route taken here is the right approach and there may exist a much simpler underlying hardness condition that can replace Theorem 5.21.

In the asymptotic regime, it would be interesting to determine the threshold on the number of satisfying assignments (as a function of the arity k) where most predicates become useless. There is a rather large gap between the upper bound $\approx 2^{5k/6}$ given by Corollary 7.6, and the lower bound of $\approx k$.

8.1 Concrete Predicates

It would be interesting to classify also the small number of unknown predicates for $k = 5$. Recall that, as stated in Appendix A.1, we have established that BLP+AIP is not sufficient to prove tractability for these. However, while our concrete hardness conditions are not sufficient to establish NP-hardness, it is conceivable that all remaining unknown predicates have small fixing assignments and are thus NP-hard by Theorem 2.15 – we have not found any families of polymorphisms with arbitrarily large minimal fixing assignments for any of the unknown predicates. Being less ambitious, one can try to classify some of the concrete remaining individual predicates of arity five. Let us highlight a few of them that we consider of particular interest.

1. Consider the predicate A that accepts the 10 cyclic strings of weight 2 or 3 where all ones (and hence all zeroes) are consecutive. In other words,

$$A = \{\mathtt{00011}, \mathtt{00110}, \mathtt{01100}, \mathtt{11000}, \mathtt{10001}, \\ \mathtt{11100}, \mathtt{11001}, \mathtt{10011}, \mathtt{00111}, \mathtt{01110}\}.$$

   This is a nice symmetric predicate, closed under both cyclic shifts and negations. In our tables in Appendix B.2 listing minimal and maximal unknown predicates for $k = 5$, the representative of A's equivalence class is the 25th predicate in Table 19 and the 16th predicate in Table 20. Notably, it is both a minimal and a maximal unknown predicate, meaning that removing any assignment from A makes $\mathrm{fiPCSP}(A, \mathsf{OR})$ tractable, and adding any assignment to A makes it NP-hard.

   This predicate is perhaps even more interesting and natural in the non-folded non-idempotent setting. In particular $\mathrm{PCSP}(A, \mathrm{NAE})$ is a natural discrepancy-type hypergraph coloring problem: given an ordered hypergraph which has a 2-coloring with discrepancy 1 *where all red vertices are consecutive within an edge*, the objective is to find a normal 2-coloring. This can be compared with the early results on PCSPs [AGH17], establishing that this problem without the guarantee on red vertices being consecutive is NP-hard (corresponding to the setting when A is all strings of weight 2 and 3).

2. Consider the predicate $A = \{\mathtt{00111}, \mathtt{01011}, \mathtt{01100}, \mathtt{10001}, \mathtt{10010}, \mathtt{10100}\}$ (predicate number 23 in Table 19. Unlike the previous example this has no clear structure or symmetries, but it is still interesting for a few reasons: (i) $\mathrm{fiPCSP}(A, \mathsf{OR})$ has a quite rich set of polymorphisms. For example, it admits AT_{13} (but not AT_{15}), which does not have fixing assignments of size 6. This is in contrast to the NP-hard predicates of arity five, where we are typically able to bound the size of fixing assignments by 4 or less. This suggests that some new ideas might be needed to handle it. (ii) Almost half (93 out of 189) of the remaining unknown predicates of arity 5 are supersets of A, so if $\mathrm{fiPCSP}(A, \mathsf{OR})$ is NP-hard this alone would reduce the number of unknown predicates significantly.

   This predicate and several other remaining ones in fact even have $(7, 8)$-block-symmetric polymorphisms. The predicates of Table 19 that have this property are predicates number 8, 13, 16, 21, 23, and 24.

References

[AGH17] Per Austrin, Venkatesan Guruswami, and Johan Håstad. $(2 + \varepsilon)$-Sat Is NP-hard. *SIAM Journal on Computing*, 46(5):1554–1573, 2017.

[AH11] Per Austrin and Johan Håstad. Randomly supported independence and resistance. *SIAM Journal on Computing*, 40(1):1–27, 2011.

[AH13] Per Austrin and Johan Håstad. On the usefulness of predicates. *ACM Trans. Comput. Theory*, 5(1), May 2013.

[BBKO21] Libor Barto, Jakub Bulín, Andrei Krokhin, and Jakub Opršal. Algebraic approach to promise constraint satisfaction. *J. ACM*, 68(4), July 2021.

[BG21] Joshua Brakensiek and Venkatesan Guruswami. Promise constraint satisfaction: Structure theory and a symmetric boolean dichotomy. *SIAM Journal on Computing*, 50(6):1663–1700, 2021.

[BGWŽ20] Joshua Brakensiek, Venkatesan Guruswami, Marcin Wrochna, and Stanislav Živný. The power of the combined basic linear programming and affine relaxation for promise constraint satisfaction problems. *SIAM Journal on Computing*, 49(6):1232–1248, 2020.

[BK24] Demian Banakh and Marcin Kozik. Injective hardness condition for PCSPs. In Pawel Sobocinski, Ugo Dal Lago, and Javier Esparza, editors, *Proceedings of the 39th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2024, Tallinn, Estonia, July 8-11, 2024*, pages 8:1–8:10. ACM, 2024.

[Bul17] Andrei A. Bulatov. A dichotomy theorem for nonuniform csps. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 319–330, 2017.

[BWŽ21] Alex Brandts, Marcin Wrochna, and Stanislav Živný. The complexity of promise SAT on non-Boolean domains. *ACM Trans. Comput. Theory*, 13(4), September 2021.

[BŽ22] Alex Brandts and Stanislav Živný. Beyond PCSP(1-in-3,nae). *Information and Computation*, 289:104954, 2022.

[FKOS19] Miron Ficak, Marcin Kozik, Miroslav Olšák, and Szymon Stankiewicz. Dichotomy for Symmetric Boolean PCSPs. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 57:1–57:12, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[FV98] Tomás Feder and Moshe Y. Vardi. The computational structure of monotone monadic snp and constraint satisfaction: A study through datalog and group theory. *SIAM Journal on Computing*, 28(1):57–104, 1998.

[GS20] Venkatesan Guruswami and Sai Sandeep. Rainbow coloring hardness via low sensitivity polymorphisms. *SIAM Journal on Discrete Mathematics*, 34(1):520–537, 2020.

[JCG97] Peter Jeavons, David Cohen, and Marc Gyssens. Closure properties of constraints. *Journal of the ACM*, 44(4):527–548, July 1997.

[Jea98] Peter Jeavons. On the algebraic structure of combinatorial problems. *Theoretical Computer Science*, 200(1):185–204, 1998.

[Koc94] M. Kochol. Constructive approximation of a ball by polytopes. *Math. Slovaca*, 44(1):99–105, 1994.

[MG02] Daniele Micciancio and S. Goldwasser. *Complexity of Lattice Problems*. Kluwer Academic Publishers, USA, 2002.

[Sch78] Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, STOC '78, page 216–226, New York, NY, USA, 1978. Association for Computing Machinery.

[Zhu20] Dmitriy Zhuk. A proof of the CSP dichotomy conjecture. *Journal of the ACM*, 67(5), August 2020.

A Computational Aspects of The Conditions

In this section we discuss some practical details of how to check for the various conditions described in the previous sections for concrete predicates. We first briefly discuss testing applicability of the BLP+AIP algorithm and then discuss the various hardness conditions in more detail.

A.1 Testing Applicability of BLP+AIP

Our tractability condition is to check whether fiPCSP(A, OR) can be solved using the BLP + AIP algorithm. Here we use Theorem 2.12 and hence we need to study the possible existence of (block-)symmetric polymorphisms of arbitrary large arity. We do this in two steps.

We first test if Pol(fiPCSP(A, OR)) contains infinitely many functions from any of the five families of (block-)symmetric functions from Lemma 4.2 – $\mathbf{Maj}$, $\mathbf{Par}$, $\mathbf{AT}$, $\mathsf{id}\overline{\mathbf{Maj}}$, $\mathsf{id}\overline{\mathbf{Par}}$. In the case of $\mathbf{Maj}$, $\mathbf{AT}$, $\mathsf{id}\overline{\mathbf{Maj}}$, this test can be done using an LP-solver by finding the separating hyperplanes described in Lemmas 4.3, 4.5 and 4.9. In the case of $\mathbf{Par}$ and $\mathsf{id}\overline{\mathbf{Par}}$, the affine conditions given by Lemmas 4.4 and 4.10 can be checked efficiently using basic bit operations on integers.

If this test fails, then we test whether Pol(fiPCSP(A, OR)) does not contain large block-symmetric polymorphisms of other kinds. According to Theorem 2.12 this is equivalent to Pol(fiPCSP(A, OR)) not containing an $(\ell, \ell+1)$-block-symmetric polymorphism for some ℓ. To use this condition we keep incrementing ℓ until we can establish that no $(\ell, \ell+1)$-block-symmetric polymorphisms exist. In principle this process might not even terminate, but it turns out that for $k \leq 5$ this search terminates relatively quickly for all remaining predicates. Even so this second step is relatively time-consuming, but in our case when analyzing all predicates of a given arity, we can be speed it up significantly by only performing it for (inclusion-wise) minimal predicates that are not resolved already in the first step.

So following these steps, we can completely determine for which predicates A of arity at most five fiPCSP(A, OR) can be solved using the BLP+AIP algorithm (and as a by-product we also establish that all such predicates can be handled using the five families of block-symmetric functions described above).

A.2 Ruling out Restricted Polymorphisms

In general for a k-ary predicate A, the set of ℓ-ary polymorphisms $\mathrm{Pol}(\mathrm{fiPCSP}(A, \mathsf{OR}))$ is the solution set of a large k-SAT formula , where each matrix $M \in A^\ell$ gives rise to a clause $f(M_1) \vee f(M_2) \vee \ldots \vee f(M_k)$, and we additionally require $f(x) = \neg f(x)$ (folded) and $f(0^k) = 0$ (idempotent). All of our hardness conditions boil down to proving that there are no ℓ-ary polymorphisms that take certain predefined values at a given list of inputs. In principle this can then simply be checked by generating the aforementioned SAT formula, fixing the predefined values, and then asking a SAT solver whether the formula is satisfiable. In practice the formulas we are interested in are quite large (sometimes on the order billions of clauses) and it is essential to use the predefined values already when generating the SAT formula to avoid generating already satisfied clauses.

While these SAT formulas in general have a very large number of variables and clauses and could be computationally infeasible to analyze, it turns out that they are in practice often very easy. For the specific forbidden functions in our hardness conditions, it surprisingly turns out that it is often possible to rule them out using simple unit propagation of the restricted SAT formulas.

Several hardness conditions also involve the existence of certain functions $f \in \mathrm{Pol}(\mathrm{fiPCSP}(A, \mathsf{OR}))^0$ (the set of functions obtained by taking a polymorphism and fixing some number of variables to 0). Any such function f of arity ℓ can be obtained from an actual polymorphism $\tilde{f}$ of arity $\ell+1$ by fixing a single bit to 0. I.e., $\mathrm{Pol}(\mathrm{fiPCSP}(A, \mathsf{OR}))^0$ contains a function f satisfying $f(x) = y$ for all predetermined values $(x, y) \in P$ if and only if $\mathrm{Pol}(\mathrm{fiPCSP}(A, \mathsf{OR}))$ contains a function $\tilde{f}$ satisfying $f(0x) = y$ for all $(x, y) \in P$.

A.3 Sufficient parameters for small arities

In practice it can be difficult to find obstructions for polymorphisms of large arity because the SAT formulas become very large. However, it is not always needed to consider high arity polymorphisms. To illustrate this, we provide a table, Table 17, listing sufficient values of the parameter t used in hardness conditions given by Theorems 5.16, 5.17, 5.18 and 5.21. The table contains the parameters that are sufficient to show that at least one of the hardness conditions, Theorems 5.16, 5.17, 5.18 and 5.21, can be applied.

Table 17: Given any predicate A that we have identified is NP-hard using Theorems 5.16, 5.17, 5.18 and 5.21, this table contains the smallest values of t that we have identified to be sufficient to apply at least one of theorems.

	Matching	Inv. Matching	t-ADA-free	t-UnCADA-free	t-UnDADA-free
$k = 3$	1	2	2	2	3
$k = 4$	3	2	3	3	4
$k = 5$	3	3	5	4	4

Out of the different conditions given in the table, the most difficult one to handle computationally is UnCADA. When $k = 5$ and $t = 4$, the polymorphisms in question have arity 11. In our experience, identifying all predicates that are 4-UnCADA-free using a Python script on a laptop is instantaneous for $k = 3$, requires several minutes for $k = 4$, and takes several days for $k = 5$.

Another parameter of interest is the largest value of ℓ such that $\mathrm{Pol}(\mathrm{fiPCSP}(A, \mathsf{OR}))$ contains a $(\ell, \ell+1)$-block-symmetric polymorphism but no $(\ell+1, \ell+2)$-block-symmetric

polymorphism. Or in other words, what is the maximum value of ℓ for which fiPCSP(A, OR) contains at least one $(\ell, \ell+1)$-block-symmetric polymorphism but where the PCSP is not solvable by BLP+AIP. We verified numerically, by an exhaustive search, that if $k = 3$ then $\ell = 1$, if $k = 4$ then $\ell = 3$, and if $k = 5$ then $\ell = 7$.

A.4 Unate Minions

The condition of a minion being unate is easy to check because of the following lemma.

Lemma A.1. *A minion $\mathcal{M}$ contains only unate functions if and only if $\mathcal{M}$ does not contain a function f of arity 5 satisfying*

$$f(00011) = 0 \qquad f(00101) = 1$$
$$f(10011) = 1 \qquad f(10101) = 0$$

Proof. Suppose $g \in \mathcal{M}$ is not unate and let a and b witness this for some variable i. Partition all variables $j \neq i$ into four groups depending on the values of a_j and b_j. Create a minor f of g by identifying all variables in the same group with new variables and keeping x_i as its own variable. It is easy to check that with an appropriate ordering of the five variables (x_i becomes the first variable) this function f satisfies the constraints above. □

A.5 Bounded Matching and Inverted Matching

For bounded matchings, we base our test on the observation that having a matching family of size t is in fact determined by having a polymorphism satisfying a condition that only depends on t.

Lemma A.2. *All functions in a minion $\mathcal{M}$ have matching number $\leq t$ if and only if $\mathcal{M}$ does not contain a $(t+2)$-ary function f such that $f(\{i\}) = 1$ for all $1 \leq i \leq t+1$.*

Proof. If all $f \in \mathcal{M}$ have matching number $\leq t$ then clearly $\mathcal{M}$ does not contain such a function since it would have matching number $t+1$. n Conversely, assume $\mathcal{M}$ does contains an ℓ-arity function g with matching number $t+1$, where the matching consists of the sets $S_1, \ldots, S_{t+1}$. We can without loss of generality assume that $\bigcup_i S_i \subsetneq [\ell]$ since otherwise we could extend f to arity $\ell+1$ by adding an extra unused variable to f.

Now the $(t+2)$-ary minor f of g obtained by identifying all variables inside each S_i, as well as the variables in $[\ell] - (S_1 \cup \ldots \cup S_{t+1})$ is a function f of the prescribed type. □

Remark A.3. Lemma A.2 gives a method to test whether all $f \in \mathcal{M}$ have matching number t, but it does not say anything about how large t can be for our polymorphism minions of interest coming from Promise-SAT problems. Clearly if the block-symmetric functions Par and AT exist of arbitrarily large arity then $\mathcal{M}$ does not have bounded matching number, but this is not particularly interesting since the existence of these already guarantee tractability of the underlying Promise-SAT problem anyway. A more useful property, which is applicable also in non-tractable cases, is whether $\mathcal{M}^0$ contains arbitrarily large OR functions – it is easy to see that if this is the case then $\mathcal{M}$ does not have bounded matching number (and also does not have bounded fixing sets). Somewhat surprisingly, it turns out that for arity up to five, all fiPCSP(A, OR) problems that are not tractable by BLP+AIP either have matching number at most 5, or have arbitrarily large OR functions in $\mathcal{M}^0$ and therefore unbounded matching number.

Analogously we have a similar characterization of inverted matching number.

Lemma A.4. *All functions in a minion $\mathcal{M}$ have inverted matching number $\leq t$ if and only if $\mathcal{M}$ does not contain a $(t+3)$-ary function f such that $f(\{t+3\}) = 1$ and $f(\{i, t+3\}) = 0$ for all $1 \leq i \leq t+1$.*

The proof is identical to the preceding lemma and we omit it here.

A.6 The obstructions of $\mathsf{AND}_t \in \mathcal{M}^0$

In Section 5 there are two types of hardness conditions, those based on $\mathsf{AND}_t \notin \mathcal{M}^0$ for some t, Lemmas 5.4, 5.8 and 5.10, and the strictly stronger results based on $\mathcal{M}^0$ being t-ADA-free for some t, Theorems 5.16, 5.17, 5.18 and 5.21. One of the reasons why we choose to introduce hardness conditions based on $\mathsf{AND}_t \notin \mathcal{M}^0$, despite these results being strictly weaker than those derived from being t-ADA-free, is that it is straightforward to determine if $\mathsf{AND}_t \notin \mathrm{Pol}(\mathrm{fiPCSP}(A, \mathsf{OR}))^0$ for some t.

Let us see how to construct an obstruction matrix for $\mathsf{AND}_t \in \mathrm{Pol}(\mathrm{fiPCSP}(A, \mathsf{OR}))^0$. This is a matrix $M \in A^{t+1}$ where the first column corresponds to the variables fixed to 0 as discussed at the end of Appendix A.2. Using the fact that the polymorphisms of $\mathrm{fiPCSP}(A, \mathsf{OR})$ are folded, the matrix is an obstruction of $\mathsf{AND}_t \in \mathrm{Pol}(\mathrm{fiPCSP}(A, \mathsf{OR}))^0$ if and only if

- for every row i that starts with a 0, $\mathsf{AND}_t(M_i^2, \ldots, M_i^{t+1}) = 0$.
- for every row i that starts with a 1, $\overline{\mathsf{AND}}_t(\neg M_i^2, \ldots, \neg M_i^{t+1}) = 0$. Or in other words, $M_i^2 = \ldots = M_i^{t+1} = 0$.

Start by fixing the first column of M. The second condition restricts the set of columns to choose from to the subset of A with only zeroes in the positions where the first column is one. For each such element, a, let S_a denote the set of coordinates where a is zero. If the union of these sets does not cover the set of rows where the first column is zero, then it is not possible to construct an obstruction matrix with this first column. If the union of the S_a cover all rows where the first columns has a zero, then this coverage can be used to construct an obstruction matrix if $t \geq k-1$. Note that we need at most $k-1$ columns to cover all rows that start with a zero [3]. This yields an obstruction matrix for AND_{k-1}. Appending additional columns to this matrix gives an obstruction matrix for larger t. We summarize this in a lemma.

Lemma A.5. *We have $\mathsf{AND}_{k-1} \in \mathrm{Pol}(\mathrm{fiPCSP}(A, B))^0$ if and only if $\mathsf{AND}_t \in \mathrm{Pol}(\mathrm{fiPCSP}(A, B))^0$ for every $t \geq k-1$.*

A.7 The obstructions of $\mathsf{ANDNOR}_t \in \mathcal{M}^0$

In order to be able to apply hardness conditions given by Lemma 5.10 and Theorem 5.18 for unate polymorphisms, we need to be able to tell if there exists a t such that $\mathsf{ANDNOR}_t \notin \mathrm{Pol}(\mathrm{fiPCSP}(A, \mathsf{OR}))^0$. The obstruction matrices of ANDNOR_t can be constructed in a similar manner to those of AND_t. Using the fact that the polymorphisms of $\mathrm{fiPCSP}(A, \mathsf{OR})$ are folded, a matrix $M \in A^{t+1}$ is an obstruction matrix of $\mathsf{ANDNOR}_t \in \mathrm{Pol}(\mathrm{fiPCSP}(A, \mathsf{OR}))^0$ if and only if

- for every row i that starts with a 0, $\mathsf{ANDNOR}_t(M_i^2, \ldots, M_i^{t+1}) = 0$.

[3] As a is not 0^k, only at most $k-1$ rows start with a zero.

- for every row i that starts with a 1, $\overline{\mathsf{ANDNOR}}_t(\neg M_i^2, \ldots, \neg M_i^{t+1}) = 0$.

This equates to the following conditions.

- No row starts with 11.
- For every row i that starts with 01, $\mathsf{OR}_{t-1}(M_i^3, \ldots, M_i^{t+1}) = 1$.
- for every row i that starts with 10, $M_i^3 = \ldots = M_i^{t+1} = 1$.

Analogously as with AND, we start by fixing the first two columns such that no row starts with 11. The third condition then restricts the set of columns to choose from to the subset of A with only ones in the positions where the first two columns are 10. For each such element a, let S_a be the set of coordinates where a is 1. If the union the S_a does not cover the set of all rows that start with 01, then it is impossible to satisfy the second condition, implying that this choice of the first two columns cannot yield an obstruction matrix. But if the union does cover all rows, then we can construct an obstruction matrix. Note that we need at most $k-1$ columns to cover to cover all the rows that start with 01 since there are at most $k-1$ rows to be covered. This yields an obstruction matrix for ANDNOR_k. Appending additional columns results in an obstruction matrix for larger t. This is summarized in the following lemma, which is an analogue of Lemma A.5.

Lemma A.6. *We have* $\mathsf{ANDNOR}_k \in \mathrm{Pol}(\mathrm{fiPCSP}(A,B))^0$ *if and only if* $\mathsf{ANDNOR}_t \in \mathrm{Pol}(\mathrm{fiPCSP}(A,B))^0$ *for every* $t \geq k$.

A.8 Monotonicity of ADA-free minions

When it comes to ADAs, these are more complicated to rule out. But we can at least establish some basic monotonicity properties which reduce the amount of different (c,d)-ADAs we have to forbid, and establishing that a t-ADA-free minion must also be $(t+1)$-ADA-free.

Claim A.7. *Let $\mathcal{M}$ be a minion. If $\mathcal{M}$ does not contain a (c,d)-ADA for some $c,d \geq 1$, then it also does not contain a $(c+1,d)$-ADA.*

Proof. The contrapositive is easily proved: if f is a $(c+1,d)$-ADA then the minor obtained by identifying two y-variables is a (c,d)-ADA. □

Claim A.8. *Let $\mathcal{M}$ be a minion. If $\mathcal{M}$ does not contain a (c,d)-ADA for some $d \geq c \geq 1$, then it also does not contain a $(c,d+1)$-ADA.*

Proof. To prove the contrapositive, assume $\mathcal{M}$ contains a $(c,d+1)$-ADA $f : \{0,1\}^{d+1} \times \{0,1\}^c \times \{0,1\}^{d+1}$. Consider the minor $g : \{0,1\}^d \times \{0,1\}^c \times \{0,1\}^d$ obtained by identifying x_d with x_{d+1} and z_d with z_{d+1}, i.e.,

$$g(x,y,z) = f(x',y',z')$$

where $x' = (x_1, \ldots, x_d, x_d)$, $y' = y$, and $z' = (z_1, \ldots, z_d, z_d)$. We claim that g is a (c,d)-ADA. Let us verify the properties.

(a) $g(1^d, 1^c, 0^d) = g(0^d, 1^c, 1^d) = 1$ by construction.

(b) $g(x,y,z) = f(x',y',z') = 0$ if $w(x)+w(y)+w(z) < c+d$. Note that if either $x_d = 0$ or $z_d = 0$, then $w(x')+w(y')+w(z') \leq w(x)+w(y)+w(z)+1 < c+d+1$ so $f(x',y',z') = 0$ due to f being a $(c,d+1)$-ADA. If $x_d = z_d = 1$ then $w(x')+w(y')+w(z') = w(x)+w(y)+w(z)+2$. If $w(x)+w(y)+w(z) < c+d-1$ we are again done, so the remaining case is $x_d = z_d = 1$ and $w(x')+w(y')+w(z') = c+d+1$. Note that since $d \geq c$, not both of x' and z' can equal 1^{d+1}. But since x' and z' are both non-zero, this implies that at most one of x', y', z' is all-ones, so $f(x',y',z') = 0$ as desired since f is an ADA.

(c) $g(x,y,z) = f(x',y',z') = 0$ unless at least two of x,y,z are all-1s is easily verified by construction.

□

A.9 Monotonicity of UnCADA-free and UnDADA-free minions

The two families UnCADA (Unate Controlled Approximate Double-AND) and UnDADA (Unate Double-controlled Approximate Double-AND) are part of the hardness condition of Theorem 5.21. While their definitions arise naturally from the inductive proof of Theorem 5.21, they are nonetheless quite intricate. Some monotonic properties of being UnCADA-free and UnDADA-free follows almost directly from their definitions. Such as $\mathcal{M}$ not containing (c,d)-UnCADA implying that $\mathcal{M}$ does not contain a $(c+1,d)$-UnCADA and $\mathcal{M}$ does not contain a t-UnDADA implying that $\mathcal{M}$ does not contain a $(t+1)$-UnDADA.

Claim A.9. *Let $\mathcal{M}$ be a minion. If $\mathcal{M}$ does not contain a (c,d)-UnCADA for some $c,d \geq 1$, then it also does not contain a $(c+1,d)$-UnCADA.*

Proof. The contrapositive is easily proved: if $f : \{0,1\}^{(c+1)+2d+1} \times \{0,1\}^3 \to \{0,1\}$ is a $(c+1,d)$-UnCADA then the minor obtained from $f(x,y)$ by identifying x_{d+1} and x_{d+2} is a (c,d)-UnCADA. □

Claim A.10. *Let $\mathcal{M}$ be a minion. If $\mathcal{M}$ does not contain a t-UnDADA for some $t \geq 3$, then it also does not contain a $(t+1)$-UnDADA.*

Proof. According to the definition of UnDADA, Definition 5.20, a function $f : \{0,1\}^t \times \{0,1\}^4 \to \{0,1\}$ is a t-UnDADA if and only if

(a) $f(1^{t-1}0, 0011) = f(01^{t-1}, 1001) = 1$

(b) for every $x \in \{0,1\}^t$ and $y \in \{0,1\}^3$ such that $w(x) \leq t-1$ and $w(y) \geq 2$, it holds that $f(x, y1) = 0$

From these conditions it follows that identifying the second and third variable of f results in minor that is a $(t-1)$-UnDADA, assuming $t \geq 4$. This shows that if $\mathcal{M}$ does not contain a t-UnDADA for some $t \geq 3$, then it also does not contain a $(t+1)$-UnDADA. □

The following lemma establishes that if $\mathcal{M}$ is t-UnCADA-free, then it is also $(t+1)$-UnCADA-free, a fact that is not immediately evident from its definition.

Lemma A.11. *Let $\mathcal{M}$ be a minion. If $\mathcal{M}$ is t-UnCADA-free for some $t \geq 2$, then it is also $(t+1)$-UnCADA-free.*

Proof. By Claim A.9, the assumption that $\mathcal{M}$ is t-UnCADA-free implies that it does not contain a $(c+1, t-c)$-UnCADA for any $1 \leq c \leq t-1$, so it remains to prove that $\mathcal{M}$ does not contain a $(1,t)$-UnCADA.

Suppose for contradiction that $\mathcal{M}$ contains a $(1,t)$-UnCADA $f : \{0,1\}^{2t+2} \times \{0,1\}^3$. Consider the minor $g(x,y)$ of $f(x',y)$ obtained by identifying x'_t, x'_{t+1}, and x'_{t+2}. The function g satisfies the following properties:

(a) $g(1^{t-1}10^{t-1}0, 011) = f(1^t110^{t-1}0, 011) \geq f(1^t10^t0, 011) = 1$ (since f is positive in x), and analogously
$g(0^{t-1}11^{t-1}0, 101) = 1$.

(b) If $w(x) < t-1$, or if $w(x) < t+1$ and $x_t = 0$, then $g(x0, y1) = 0$ for all y of weight $w(y) \geq 1$. This follows since such an input corresponds to an input x', y for f such that $w(x') < t+1$ and $w(y) \geq 1$ and hence $f(x'0, y1) = 0$ since f is a $(1,t)$-UnCADA.

Thus, since g cannot be a $(t-1,1)$-UnCADA (by the assumption that $\mathcal{M}$ is t-UnCADA-free), there must x' of $w(x') = t+1$ such that $x'_t = x'_{t+1} = x'_{t+2} = 1$ and y with $w(y) \geq 1$, such that $f(x'0, y1) = 1$. Without loss of generality we may assume $w(y) = 1$ since f is negative in y. Suppose $y = 01$ (the other case $y = 10$ is symmetric). Then f satisfies $f(x'0, 011) = 1$ and $f(1^t10^t, 101) = 1$ since f is a $(1,t)$-UnCADA. Since $w(x') = t+1$ and $x_t = x_{t+1} = 1$, there must be some $t+2 \leq i \leq 2t+1$ such that $x'_i = 0$. Let I be the set of all such i, and consider the minor h of f obtained by identifying all coordinates of I with the last x-variable. We claim that, after applying an appropriate reordering of variables, h is a $(|I|+1, t-|I|)$-UnCADA:

(a) There are two assignments $x^{(1)}$ and $x^{(2)}$ of weight $t+1$ and overlap $|I|+1$ such that $h(x^{(1)}0, 011) = h(x^{(2)}0, 101) = 1$ – these are the two assignments x' and 1^t10^t with the coordinates of I removed.

(a) $h(x0, y1) = 0$ for all x of $w(x) \leq t$ and $w(y) \geq 1$ follows since our identification of the variables of I with the last x-variable is effectively just fixing those variables to 0.

But this is now a contradiction to the property, noted above, that $\mathcal{M}$ does not contain a $(c+1, t-c)$-UnCADA for any $1 \leq c \leq t-1$. □

B Lists of Predicates of Arity 5

In this section we list various interesting categories of predicates of arity 5.

B.1 Maximal Tractable Predicates for fiPCSP(A, OR)

The maximal tractable (as far as we know) predicates for fiPCSP(A, OR) of arity 5 are given in Table 18. Easy to recognize predicates are parity of three, four or five variables (number 29, 27, and 28). Of course 2-SAT is present as number 30 as well as the (non-strict) majority of four variables (number 31). Predicate 32 is the closely related function which is a threshold function where the first coordinate has weight 2 and the other coordinates have weight 1.

Table 18: A list of the 32 maximal tractable predicates for $k = 5$. See Table 6 (Section 6.1.3) for explanation of the last column.

	Predicate	Maj	Par	AT	id$\overline{\text{Maj}}$	id$\overline{\text{Par}}$	Dep.
1.	{00011, 00101, 00110, 01000, 10000}			✓			1/11
2.	{00011, 00101, 00110, 01001, 01010, 01100, 10000}			✓			3/21
3.	{00011, 00100, 00110, 01000, 01001, 10000, 10001}				✓		2/48
4.	{00011, 00101, 00111, 01000, 01001, 10000, 10001}				✓		1/51
5.	{0011∗, 0101∗, 0110∗, 1000∗}			✓		✓	12/77
6.	{00011, 00101, 00111, 01000, 01010, 01100, 01110, 10000, 10001}				✓		8/253
7.	{00010, 00110, 00111, 01000, 01001, 01100, 01101, 10001, 10010}				✓		4/320
8.	{00111, 01001, 01010, 01101, 01110, 10000, 10011}					✓	2/67
9.	{00011, 00110, 00111, 01000, 01001, 01100, 01101, 10001, 10011}				✓		2/342
10.	{00100, 00101, 00110, 01010, 01101, 10000, 10001, 10010, 10011}				✓		4/330
11.	{0011∗, 0100∗, 0110∗, 1000∗, 1001∗}				✓		36/449
12.	{00101, 00110, 01001, 01010, 01101, 01110, 10000, 10001, 10010, 10011}				✓		11/320
13.	{00110, 01001, 01100, 01101, 01110, 10000, 10001, 10010, 10011}				✓		1/227
14.	{00010, 00100, 01010, 01011, 01100, 01101, 10001, 10010, 10100}				✓		2/233
15.	{00011, 01001, 01010, 01101, 01110, 10011, 10100}					✓	2/52
16.	{00111, 01001, 01010, 01101, 01110, 10011, 10100}					✓	1/57
17.	{00110, 01000, 01001, 01010, 01101, 10000, 10001, 10011, 10100}				✓		3/215
18.	{00110, 01001, 01011, 01100, 01110, 10000, 10001, 10011, 10100}				✓		4/329
19.	{00110, 01001, 01010, 01101, 01110, 10000, 10001, 10011, 10100}				✓		2/325
20.	{00110, 01001, 01100, 01101, 01110, 10000, 10001, 10011, 10100}				✓		2/358

Continued on next page

Continued from previous page

	Predicate	Maj	Par	AT	id$\overline{\text{Maj}}$	id$\overline{\text{Par}}$	Dep.
21.	{00110, 01011, 01101, 01110, 10000, 10011, 10101}					✓	2/63
22.	{00101, 01010, 01011, 01100, 01101, 10001, 10010, 10011, 10101}				✓		2/184
23.	{00101, 01010, 01100, 01101, 01110, 10001, 10010, 10011, 10101}				✓		1/127
24.	{00011, 00100, 01001, 01011, 01100, 01110, 10010, 10011, 10100, 10101}				✓		16/446
25.	{00011, 00101, 00110, 01001, 01010, 01100, 10001, 10010, 10100, 11000}			✓			11/33
26.	{00101, 00110, 01011, 01110, 10011, 10101, 11000}					✓	1/53
27.	{0001*, 0010*, 0100*, 0111*, 1000*, 1011*, 1101*, 1110*}		✓				2 388/3 875
28.	{00001, 00010, 00100, 00111, 01000, 01011, 01101, 01110, 10000, 10011, 10101, 10110, 11001, 11010, 11100, 11111}		✓				674/1 087
29.	{**001, **010, **100, **111}		✓				4 313/7 099
30.	{***01, ***10, ***11}	✓					1 118 234/1 249 651
31.	{*0011, *0101, *0110, *0111, *1001, *1010, *1011, *1100, *1101, *1110, *1111}	✓					31 133/157 103
32.	{00011, 00101, 00111, 01001, 01011, 01101, 01110, 01111, 10001, 10011, 10101, 10110, 10111, 11001, 11010, 11011, 11100, 11101, 11110, 11111}	✓					355/43 951

B.2 Minimal and Maximal Unknown Predicates for fiPCSP(A, OR)

Table 19 lists the minimal predicates A of arity 5 where we have been unable to determine the complexity of fiPCSP(A, OR). Table 20 lists the maximal such predicates.

Table 19: A list of the 25 minimal unknown predicates for Promise-SAT of arity $k = 5$. A checkmark in the "unate" column indicates that all polymorphisms are unate. A cross in the ADA (resp. UnCADA or UnDADA) columns indicates that the polymorphism minion is t-ADA-free (resp. t-UnCADA-free or t-UnDADA-free) for some t. The second value in the "Dep." column gives the number of unknown predicates implied by this predicate (in particular this number of unknown Promise-SAT problems would be NP-hard if this predicate is shown NP-hard), while the first value in the "Dep." column gives the number of such predicates that are not implied by any other predicate in the table.

	Predicate	Unate	ADA	UnCADA	UnDADA	Dep.
1.	{00011, 00111, 01001, 01010, 01100, 10000}	✓	✗		✗	1/2
2.	{00101, 00110, 00111, 01011, 01100, 10000}		✗	✗	✗	1/3
3.	{00110, 00111, 01001, 01011, 01100, 10000}	✓	✗		✗	1/3
4.	{00011, 00110, 00111, 01011, 01101, 10000}		✗	✗	✗	1/8
5.	{00011, 00111, 01011, 01101, 01110, 10000}		✗			1/6
6.	{00110, 00111, 01001, 01010, 01100, 10001}	✓	✗		✗	1/4
7.	{00011, 00110, 01010, 01100, 10000, 10001}	✓	✗			1/2
8.	{00111, 01001, 01010, 01100, 10000, 10001}		✗			3/5
9.	{00101, 00111, 01011, 01100, 10000, 10001}	✓	✗			1/4
10.	{00111, 01011, 01100, 01110, 10000, 10001}		✗			1/8
11.	{00111, 01011, 01101, 01110, 10000, 10001}		✗			4/7
12.	{00011, 00111, 01001, 01100, 10001, 10010}		✗			2/14
13.	{00101, 00111, 01011, 01100, 10001, 10010}	✓	✗			3/28
14.	{00110, 01010, 01100, 01101, 10001, 10010}		✗			1/10
15.	{00111, 01010, 01100, 01101, 10001, 10010}	✓	✗			2/25
16.	{00111, 01011, 01100, 01101, 10001, 10010}		✗			9/81
17.	{00111, 01011, 01100, 10000, 10001, 10010}		✗			1/5
18.	{00111, 01010, 01100, 01101, 10000, 10011}		✗	✗	✗	1/4
19.	{00111, 01010, 01101, 10000, 10001, 10011}		✗	✗	✗	1/4
20.	{00111, 01001, 01010, 01100, 10001, 10010, 10011}		✗			2/3
21.	{00111, 01011, 01100, 10001, 10010, 10011}		✗			4/57
22.	{00101, 01011, 01100, 10001, 10010, 10100}	✓	✗			3/21
23.	{00111, 01011, 01100, 10001, 10010, 10100}	✓	✗			10/93
24.	{00011, 01101, 01110, 10011, 10100, 11000}	✓	✗		✗	4/18
25.	{00110, 01001, 01100, 01101, 01110, 10001, 10010, 10011, 10110, 11001}			✗	✗	1/1

Table 20: A list of the 19 maximal unknown predicates for Promise-SAT of arity $k = 5$. Most columns are as in Table 19 but the "Dep." column differs. The second value in this column now gives the number of unknown predicates that imply this predicate (in particular this number of unknown Promise-SAT problems would be tractable if this predicate is shown tractable), while the first value gives the number of such predicates that do not imply any other predicate in the table.

	Predicate	Unate	ADA	UnCADA	UnDADA	Dep.
1.	{00011, 00101, 00111, 01001, 01011, 01101, 01110, 10000}		✗	✗	✗	3/6
2.	{00110, 00111, 01001, 01010, 01100, 10000, 10001}	✓	✗		✗	3/5
3.	{00110, 00111, 01010, 01011, 01101, 01110, 10000, 10001}		✗	✗	✗	1/8
4.	{00110, 00111, 01001, 01011, 01100, 10000, 10001, 10010}	✓	✗		✗	6/8
5.	{00111, 01011, 01100, 01101, 01110, 10000, 10001, 10010}		✗			2/10
6.	{00111, 01010, 01100, 01101, 01110, 10000, 10001, 10011}		✗	✗	✗	6/9
7.	{00101, 00111, 01011, 01100, 10001, 10010, 10011}	✓	✗		✗	1/3
8.	{00111, 01001, 01010, 01100, 10000, 10001, 10010, 10011}		✗	✗		3/5
9.	{00111, 01011, 01101, 01110, 10000, 10001, 10010, 10011}		✗	✗		2/4
10.	{00011, 00111, 01001, 01010, 01100, 10001, 10010, 10100}	✓	✗		✗	2/5
11.	{00111, 01001, 01010, 01100, 10001, 10010, 10100, 11000}	✓	✗			1/3
12.	{00110, 00111, 01010, 01011, 01100, 01101, 10001, 10010, 10100, 11000}	✓	✗		✗	15/30
13.	{00110, 01010, 01101, 01110, 10001, 10011, 10100, 11000}		✗	✗	✗	1/2
14.	{00111, 01001, 01011, 01100, 01110, 10010, 10011, 10100, 10101, 11000}	✓	✗		✗	12/30
15.	{00111, 01001, 01010, 01011, 01100, 01101, 01110, 10001, 10011, 10101, 10110, 11000}	✓	✗		✗	18/71
16.	{00110, 01001, 01100, 01101, 01110, 10001, 10010, 10011, 10110, 11001}			✗	✗	1/1

Continued on next page

Continued from previous page

	Predicate	Unate	ADA	UnCADA	UnDADA	Dep.
17.	{00011, 00111, 01010, 01011, 01101, 01110, 10010, 10100, 10110, 11001}	✓	✗		✗	4/21
18.	{00011, 00110, 01010, 01011, 01101, 01110, 10011, 10100, 10110, 11001}	✓	✗		✗	6/25
19.	{00111, 01011, 01100, 01101, 01110, 10001, 10010, 10011, 10100, 10110, 11000, 11001}	✓	✗		✗	16/67

B.3 Minimal and Maximal Unknown Predicates for Promise-Usefulness

Table 21 lists the minimal predicates of arity 5 where we have been unable to determine whether they are promise-useful or not. Table 22 lists the maximal such predicates.

Table 21: A list of the 9 minimal predicates with unknown promise-usefulness status for $k = 5$.

	Predicate	Dep.
1.	{00011, 01101, 01110, 10011, 10100, 11000}	13/18
2.	{00110, 00111, 01001, 01101, 01110, 10011, 10100, 11000}	2/16
3.	{00111, 01001, 01010, 01101, 01110, 10011, 10100, 11000}	3/14
4.	{00110, 01001, 01100, 01101, 01110, 10011, 10100, 11000}	1/12
5.	{00110, 01010, 01101, 01111, 10000, 10011, 10100, 11000}	1/1
6.	{00110, 01001, 01101, 01110, 10001, 10011, 10100, 11000}	2/14
7.	{00111, 01010, 01101, 01110, 10001, 10011, 10100, 11000}	6/24
8.	{00111, 01011, 01100, 01101, 01110, 10001, 10010, 10011, 10100, 11000}	1/4
9.	{00111, 01000, 01100, 01110, 01111, 10000, 10001, 10011, 10111, 11000}	1/1

Table 22: A list of the 7 maximal predicates with unknown promise-usefulness status for $k = 5$.

	Predicate	Dep.
1.	{00011, 00110, 00111, 01001, 01100, 01101, 01110, 10011, 10100, 11000}	6/17
2.	{00011, 00111, 01001, 01010, 01100, 01101, 01110, 10011, 10100, 11000}	4/14
3.	{00110, 01010, 01101, 01111, 10000, 10011, 10100, 11000}	1/1
4.	{00101, 00110, 00111, 01011, 01100, 01101, 01110, 10001, 10010, 10011, 10100, 11000}	8/19
5.	{00110, 00111, 01001, 01011, 01100, 01101, 01110, 10001, 10010, 10011, 10100, 11000}	16/27

Continued on next page

Continued from previous page

	Predicate	Dep.
6.	$\{00111, 01001, 01011, 01100, 01110, 10010, 10011, 10100, 10101, 11000\}$	1/4
7.	$\{00111, 01000, 01100, 01110, 01111, 10000, 10001, 10011, 10111, 11000\}$	1/1

C Proof of Theorem 2.15

In what follows, a *choice function* C is a function defined on a minion which, given an ℓ-ary function f, identifies a subset $C(f) \subseteq [\ell]$ of the coordinates of f. Banakh and Kozik [BK24] proved the following result (this is the special case of Theorem 3.1 in their paper with two layers).

Theorem C.1. *Suppose there exists a constant t and choice function C on the polymorphism minion* $\mathrm{Pol}(\mathrm{PCSP}(A, B))$ *with the following properties:*

1. $|C(f)| \leq t$ *for all polymorphisms f of* $\mathrm{PCSP}(A, B)$.
2. *For every polymorphism f and minor f_π of f such that π is injective on $C(f)$, it holds that $\pi(C(f)) \cap C(f_\pi) \neq \emptyset$.*

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

As our choices come from fixing assignments the below simple claim is useful. The proof is standard as if two fixing assignments give values to disjoint sets of variables then one can simultaneously force the function to take the value 0 and the value 1.

Claim C.2. *Let $f : \{0,1\}^\ell \to \{0,1\}$ be a folded Boolean function. Then any two fixing assignments (S_1, α_1), (S_2, α_2) must satisfy $S_1 \cap S_2 \neq \emptyset$.*

Let us now prove Theorem 2.15, restated here for convenience.

Theorem 2.15. *If there exists a t such that every $f \in \mathrm{Pol}(\mathrm{fPCSP}(A, B))$ has a t-fixing assignment, then* $\mathrm{fPCSP}(A, B)$ *is NP-hard. Likewise, if there is a t such that every $f \in \mathrm{Pol}(\mathrm{fiPCSP}(A, B))$ has a t-fixing assignment, then* $\mathrm{fiPCSP}(A, B)$ *is NP-hard.*

Proof. We define a choice function C on the polymorphisms of $\mathrm{PCSP}(A, B)$ such that $C(f)$ returns the coordinates of an arbitrarily chosen (say, the lexicographically smallest) fixing assignment T of size t. Clearly this satisfies the first condition of Theorem C.1 and we now verify the second.

Take an arbitrary polymorphism $f : \{0,1\}^\ell$ and minor $f_\pi : \{0,1\}^{\ell'}$ for which $\pi : [\ell] \to [\ell']$ is injective on $C(f)$. Since π is injective, every partial assignment $x_{C(f)}$ for f is the preimage (under π) of a partial assignment $y_{\pi(C(f))}$ for f_π. In particular for f's fixing assignment on $C(f)$ there is a corresponding fixing assignment for f_π on $\pi(C(f))$. In other words, both $(C(f_\pi), \alpha_1)$ and $(\pi(C(f), \alpha_2))$ are fixing assignments for f for some α_1, α_2, so by Claim C.2 $C(f_\pi)$ and $\pi(C(f))$ cannot be disjoint. □

D Proofs for Maj, Par, and AT

In this section we prove the (standard) lemmas from Section 4.2, characterizing **Maj**, **Par**, and **AT** as polymorphisms of Promise-SAT.

Lemma 4.3. *The following statements are equivalent:*

1. Pol(fPCSP(A, OR)) *contains infinitely many polymorphisms from* **Maj**.
2. $[0, 1/2)^k \cap K(A) = \emptyset$.
3. *There exists integers* $c_1, \dots c_k \geq 0$ *such that* $\sum_{j=1}^k c_j a_j \geq \sum_{j=1}^k c_j/2 > 0$ *for all* $a \in A$.

Proof. Suppose that Maj_ℓ is not a polymorphism for some ℓ, then there exists a $k \times \ell$ obstruction matrix M such that $\sum_j M^j/\ell \in [0, 1/2)^k$. As $\sum_j M^j/\ell$ is a point in $K(A)$ this shows that $[0, 1/2)^k \cap K(A) \neq \emptyset$. We conclude that 2 implies 1.

On the other hand, suppose that $[0, 1/2)^k \cap K(A) \neq \emptyset$ and contains the point x. As all vectors in A have rational entries, x can be chosen to be rational and can hence be written as $\sum_i \frac{\alpha_i}{\beta} a^i$, where $\sum_i \alpha_i = \beta$ and $\alpha_1, \alpha_2, \dots$ and β are non-negative integers and $a^i \in A$. As the components of x are rational numbers with denominator β, each such component is bounded from above by $\frac{1}{2} - \frac{1}{2\beta}$.

We claim that for $\ell = 2d\beta + \gamma$ where $d \geq \beta$ and $\gamma < 2\beta$, Maj_ℓ is not a polymorphism. Indeed, its obstruction matrix can be constructed by taking $2d\alpha_i$ columns of each a^i, joint with γ arbitrary element of A. In this matrix, the sum of each row is bounded by

$$2d\beta\left(\frac{1}{2} - \frac{1}{2\beta}\right) + \gamma \leq d\beta - d + \gamma < \ell/2.$$

As any odd number greater than $2\beta^2$ can be written on the given form we conclude that 1 implies 2.

Finally, 3 immediately implies 2 and let us establish the reverse implication. Firstly, note that $[0, 1/2)^k \cap K(A) = (-\infty, 1/2)^k \cap K(A)$ as all vectors in $K(A)$ only have non-negative components. We apply Theorem 2.1 to obtain $c_1, \dots, c_k$ and b such that $\sum_{j=1}^k x_j c_j < b$ for all $x \in (-\infty, 1/2)^k$ and $\sum_{j=1}^k y_j c_j \geq b$ for all $y \in K(A)$. It is easy to see that the first condition implies that $c_1, \dots, c_k$ are non-negative and in view of this, b can be modified to take the value $\sum_{j=1}^k c_j/2$. Finally the set of vectors c that satisfy the inequalities in 3 is a rational polytope and hence if it is nonempty it contains a rational vector. Since it is closed under multiplication by scalars it also contains an integral vector. This completes the proof. □

Lemma 4.4. *The following statements are equivalent:*

1. Pol(fPCSP(A, OR)) *contains infinitely many polymorphisms from* **Par**.
2. *For every odd sized subset* B *of* A, $\bigoplus_{s \in B} s \neq 0^k$.
3. *There exists a non-empty subset* $\beta \subseteq [k]$, *such that* $\bigoplus_{i \in \beta} a_i = 1$ *for all* $a \in A$.

Proof. Let M be an obstruction matrix for Par_ℓ, ℓ odd. Note that the columns of M are elements of A, and that $\oplus_j M^j = 0^k$. This implies the existence of an odd sized subset B such that $\oplus_{s \in B} s = 0^k$. This shows that 2 implies 1.

On the other hand, suppose that there exists an odd sized subset B of A such that $\oplus_{s \in B} s = 0^k$. We use this subset can be to create obstructions for Par_ℓ for any odd $\ell \geq |B|$. The first $|B|$ columns are given by B while the remaining columns all equal some fixed element of A. As this column appears an even number of times, it will not change the parity of the matrix. This shows that 1 implies 2.

To show that 2 and 3 are equivalent, note that $\{\oplus_{s \in B} s : B \subseteq A, |B| \text{ odd}\}$ is an affine subspace in $\mathbf{F}_2^k$. Any affine subspace can always be represented as the set of solutions x

of a linear equation system $Mx = b$ for some matrix $M \in \mathbf{F}_2^{\ell \times k}$. The statement that the affine set does not contain 0^k is equivalent to $b \neq 0^\ell$ and we can take any $i \in [\ell]$ where $b_i \neq 0$ and let M_i define the set β in 3. We conclude that 2 implies 3 and the reverse implication is easy to see. □

Lemma 4.5. *The following statements are equivalent:*

1. $\mathrm{Pol}(\mathrm{fPCSP}(A, \mathsf{OR}))$ *contains infinitely many polymorphisms from* **AT**.
2. $\{x - y : x, y \in K(A)\} \cap (-\infty, 0)^k = \emptyset$.
3. *There exists integers* $c_1, \dots c_k \geq 0$, *not all* 0, *such that* $\sum_{j=1}^k c_j a_j$ *takes the same value for all* $a \in A$.

Proof. The proof is very similar to the proof of Lemma 4.3 but for completeness let us give most details. Suppose $\mathsf{AT}_\ell \notin \mathrm{Pol}(\mathrm{fPCSP}(A, \mathsf{OR}))$ for some odd ℓ and let M be an obstruction matrix M. Let $z = \frac{2}{\ell-1} \sum_{i=1}^{\ell-1} (-1)^{i+1} M_i$ and note that z can written as $x - y$ where both x and y are in $K(A)$ by letting x be the sum of the odd terms, and y be the sum of the even terms. Since M is an obstruction for AT_ℓ, $\sum_{i=1}^{\ell} (-1)^{i+1} M_i^j < 0$ for all j and as the last term of this sum is positive $\sum_{i=1}^{\ell-1} (-1)^{i+1} M_i^j < 0$. This shows that $z \in (-\inf, 0)^k$. We conclude that 2 implies 1 and let us establish the reverse inclusion.

Assume that $\{x - y : x, y \in K(A)\} \cap (-\infty, 0)^k \neq \emptyset$ and let $x, y \in K(A)$ such that $x - y \in (-\infty, 0)^k$. We can assume that x and y have rational coefficients and hence can be expressed as $x = \sum_i \alpha_i^x / \beta a^i$ and $y = \sum_i \alpha_i^y / \beta a^i$, where $\sum_i \alpha_i^x = \sum_i \alpha_i^y = \beta$ and $\alpha_1^x, \alpha_2, \dots, \alpha_1^y, \alpha_2^y, \dots$ and β are non-negative integers and $a^i \in A$. Note also than any coordinate of $x - y$ is bounded from above by $-1/\beta$.

We construct an obstruction matrix M for AT_ℓ for any $\ell > 4\beta^2$. Suppose $\ell = 2\beta d + \gamma$ where $\gamma < 2\beta$. At odd indices, put $d\alpha_i^x$ columns a^i and at even columns $d\alpha_i^y$ columns a^i and complete this by any γ columns from A. It is easy to see that the alternating sum defining AT is bounded by $-d + \gamma$ in any coordinate and hence M is an obstruction matrix.

It is easy to see that 3 implies 2 as the linear form given by the c-vector is 0 on $K(A) - K(A)$ and negative in the negative orthant. The reverse implication follows, as previously, by the separation theorem, Theorem 2.1. We omit the details.

□

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il