Best-case and Worst-case Sparsifiability of Boolean CSPs

Hubie Chen
Birkbeck, University of London
hubie@dcs.bbk.ac.uk

Bart M.P. Jansen[1]
Eindhoven University of Technology
b.m.p.jansen@tue.nl
http://orcid.org/0000-0001-8204-1268

Astrid Pieterse[2]
Eindhoven University of Technology
a.pieterse@tue.nl
http://orcid.org/0000-0003-3721-6721

Abstract

We continue the investigation of polynomial-time sparsification for NP-complete Boolean Constraint Satisfaction Problems (CSPs). The goal in sparsification is to reduce the number of constraints in a problem instance without changing the answer, such that a bound on the number of resulting constraints can be given in terms of the number of variables n. We investigate how the worst-case sparsification size depends on the types of constraints allowed in the problem formulation (the constraint language). Two algorithmic results are presented. The first result essentially shows that for any arity k, the only constraint type (up to negations) for which no nontrivial sparsification is possible, is the logical OR. Our second result concerns linear sparsification, i.e., a reduction to an equivalent instance with $\mathcal{O}(n)$ constraints. Using linear algebra over rings of integers modulo prime powers, we give an elegant necessary and sufficient condition for a constraint type to be captured by a degree-1 polynomial over such a structure, which yields linear sparsifications. The combination of these algorithmic results allows us to prove two characterizations that capture the optimal sparsification sizes for a range of Boolean CSPs. For NP-complete Boolean CSPs whose constraints are *symmetric* (the satisfaction depends only on the number of 1 values in the assignment, not on their positions), we give a complete characterization of which constraint languages allow for a linear sparsification. For Boolean CSPs in which every constraint has arity at most three, we characterize the optimal size of sparsifications in terms of the largest OR that can be expressed by the constraint language.

2012 ACM Subject Classification Computing methodologies → Symbolic and algebraic algorithms, Theory of computation → Problems, reductions and completeness, Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases constraint satisfaction problems; kernelization; sparsification; lower bounds

Digital Object Identifier 10.4230/LIPIcs.IPEC.2018.23

Acknowledgements We would like to thank Emil Jeřábek for the proof of Lemma 4.2.

[1] Supported by NWO Gravitation grant "Networks".
[2] Supported by NWO Gravitation grant "Networks".

42nd Conference on Very Important Topics (CVIT 2016).
Editors: John Q. Open and Joan R. Access; Article No. 23; pp. 23:1–23:26

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Background

The framework of constraint satisfaction problems (CSPs) provides a unified way to study the computational complexity of a wide variety of combinatorial problems such as CNF-Satisfiability, Graph Coloring, and Not-All-Equal SAT. The framework uncovers algorithmic approaches that simultaneously apply to several problems, and also identifies common sources of intractability. For the purposes of this discussion, a CSP is specified using a *constraint language*, which is a set of relations; the problem is to decide the satisfiability of a set of constraints, where each constraint has a relation coming from the constraint language. The fact that many problems can be viewed as CSPs motivates the following investigation: how does the complexity of a CSP depend its constraint language? A key result in this area is Schaefer's dichotomy theorem [20], which classifies each CSP over the Boolean domain as polynomial-time solvable or NP-complete.

Continuing a recent line of investigation [12, 14, 17], we aim to understand for which NP-complete CSPs an instance can be *sparsified* in polynomial time, without changing the answer. In particular, we investigate the following questions. Can the number of constraints be reduced to a small function of the number of variables n? How does the sparsifiability of a CSP depend on its constraint language? We utilize the framework of kernelization [5, 8, 18], originating in parameterized complexity theory, to answer such questions.

The first results concerning polynomial-time sparsification in terms of the number n of variables or vertices were mainly negative. Under the assumption that $\mathsf{NP} \not\subseteq \mathsf{coNP/poly}$ (which we tacitly assume throughout this introduction), Dell and van Melkebeek [7] proved a strong lower bound: For an integer $d \geq 3$ and positive real ε, there cannot be a polynomial-time algorithm that compresses any instance φ of d-CNF-SAT on n variables, into an equivalent SAT instance φ' of bitsize $\mathcal{O}(n^{d-\varepsilon})$. In fact, there cannot even be an algorithm that transforms such φ into equivalent small instances ψ of an *arbitrary* decision problem. Since an instance of d-CNF-SAT has at most $2^d n^d \in \mathcal{O}(n^d)$ distinct clauses, it can *trivially* be sparsified to $\mathcal{O}(n^d)$ clauses by removing duplicates, and can be compressed to size $\mathcal{O}(n^d)$ by storing it as a bitstring indicating for each possible clause whether or not it is present. The cited lower bound therefore shows that the trivial sparsification for d-CNF-SAT cannot be significantly improved; we say that the problem does not admit nontrivial (polynomial-time) sparsification. Following these lower bounds for SAT, a number of other results were published [6, 11, 16] proving other problems do not admit nontrivial sparsification either.

This pessimistic state of affairs concerning nontrivial sparsification algorithms changed several years ago, when a subset of the authors [14] showed that the d-Not-All-Equal SAT problem *does* have a nontrivial sparsification. In this problem, clauses have size at most d and are satisfied if the literals do not all evaluate to the same value. While there can be $\Omega(n^d)$ different clauses in an instance, there is an efficient algorithm that finds a subset of $\mathcal{O}(n^{d-1})$ constraints that preserves the answer, resulting in a compression of bitsize $\mathcal{O}(n^{d-1} \log n)$. The first proof of this result was based on an ad-hoc application of a theorem of Lovász [19]. Later, the underlying proof technique was extracted and applied to a wider range of problems [12]. This led to the following understanding: if each relation in the constraint language can be represented by a polynomial of degree at most d, in a certain technical sense, then this allows the number of constraints in an n-variable instance of such a CSP to be reduced to $\mathcal{O}(n^d)$. The sparsification for d-Not-All-Equal SAT is then explained by noting that such constraints can be captured by polynomials of degree $d-1$. It is therefore apparent that finding a low-degree polynomial to capture the constraints of a CSP is a powerful tool

to obtain sparsification algorithms for it. Finding such polynomials of a certain degree d, or determining that they do not exist, proved a challenging and time-intensive task (cf. [13]).

The polynomial-based framework [12] also resulted in some *linear* sparsifications. Since "1-in-d" constraints (to satisfy a clause, *exactly one* out of its $\leq d$ literals should evaluate to true) can be captured by *linear* polynomials, the 1-IN-d-SAT problem has a sparsification with $\mathcal{O}(n)$ constraints for each constant d. This prompted a detailed investigation into linear sparsifications for CSPs by Lagerkvist and Wahlström [17], who used the toolkit of universal algebra in an attempt to obtain a characterization of the Boolean CSPs with a linear sparsification. Their results give a necessary and sufficient condition on the constraint language of a CSP for having a so-called Maltsev embedding over an infinite domain. They also show that when a CSP has a Maltsev embedding over a *finite* domain, then this can be used to obtain a linear sparsification. Alas, it remains unclear whether Maltsev embeddings over infinite domains can be exploited algorithmically, and a characterization of the linearly-sparsifiable CSPs is currently not known.

Our contributions

We analyze and demonstrate the power of the polynomial-based framework for sparsifying CSPs using universal algebra, linear algebra over rings, and relational analysis. We present two new algorithmic results. These allow us to characterize the sparsifiability of Boolean CSPs in two settings, wherein we show that the polynomial-based framework yields *optimal* sparsifications. In comparison to previous work [12], our results are much more fine-grained and based on a deeper understanding of the reasons why a certain CSP *cannot* be captured by low-degree polynomials.

Algorithmic results Our first result (Section 3) shows that, contrary to the pessimistic picture that arose during the initial investigation of sparsifiability, the phenomenon of nontrivial sparsification is widespread and occurs for almost all Boolean CSPs! We prove that if Γ is a constraint language whose largest constraint has arity k, then the *only* reason that CSP(Γ) does not have a nontrivial sparsification, is that it contains an arity-k relation that is essentially the k-ary OR (up to negating variables). When $R \subseteq \{0,1\}^k$ is a relation with $|\{0,1\}^k \setminus R| \neq 1$ (the number of assignments that fail to satisfy the constraints is not equal to 1), then it can be captured by a polynomial of degree $k-1$. This yields a nontrivial sparsification compared to the $\Omega(n^k)$ distinct applications of this constraint that can be in such an instance.

Our second algorithmic result (Section 4) concerns the power of the polynomial-based framework for obtaining linear sparsifications. We give a necessary and sufficient condition for a relation to be captured by a degree-1 polynomial. Say that a Boolean relation $R \subseteq \{0,1\}^k$ is *balanced* if there is *no* sequence of vectors $s_1, \ldots, s_{2n}, s_{2n+1} \in R$ for $n \geq 1$ such that $s_1 - s_2 + s_3 \ldots - s_{2n} + s_{2n+1} = u \in \{0,1\}^k \setminus R$. (The same vector may appear multiple times in this sum.) In other words: R is balanced if one cannot find an odd-length sequence of vectors in R for which alternating between adding and subtracting these vectors component-wise results in a 0/1-bitvector u that is outside R. For example, the binary OR relation 2-OR $= \{0,1\}^2 \setminus \{(0,0)\}$ is *not* balanced, since $(0,1) - (1,1) + (1,0) = (0,0) \notin$ 2-OR, but the 1-in-3 relation $R_{=1} = \{(1,0,0),(0,1,0),(0,0,1)\}$ is. We prove that if a Boolean relation R is balanced, then it can efficiently be captured by a degree-1 polynomial and the number of constraints that are applications of this relation can be reduced to $\mathcal{O}(n)$. Hence when *all* relations in a constraint language Γ are balanced—we call such a constraint language *balanced*—then CSP(Γ) has a sparsification with $\mathcal{O}(n)$ constraints. We also show that, on

the other hand, if a Boolean relation R is not balanced, then there does not exist a degree-1 polynomial (over any ring $\mathbb{Z}/q\mathbb{Z}$) that captures R in the sense required for application of the polynomial framework. The property of being balanced is (as defined) a universal-algebraic property; these results thus tightly bridge universal algebra and the polynomial framework.

Characterizations The property of being balanced gives an easy way to prove that certain Boolean CSPs admit linear sparsifications. But perhaps more importantly, this characterization constructively exhibits a certain *witness* when a relation can *not* be captured by a degree-1 polynomial, in the form of the alternating sum of satisfying assignments that yield an unsatisfying assignment. In several scenarios, we can turn this witness structure against degree-1 polynomials into a lower bound proving that the problem does not have a linear sparsification. As a consequence, we can prove two fine-grained characterizations of sparsification complexity.

Characterization of symmetric CSPs with a linear sparsification (Section 5). We say that a Boolean relation is *symmetric* if the satisfaction of a constraint only depends on the *number* of 1-values taken by the variables (the *weight* of the assignment), but does not depend on the *positions* where these values appear. For example, "1-in-k"-constraints are symmetric, just as "not-all-equal"-constraints, but the relation $R_{a\to b} = \{(0,1),(1,1)\}$ corresponding to the truth value of $a \to b$ is not. We prove that if a symmetric Boolean relation R is not balanced, then it can implement a binary OR using constants and negations but without having to introduce fresh variables. Building on this, we prove that if such an unbalanced symmetric relation R occurs in a constraint language Γ for which CSP(Γ) is NP-complete, then CSP(Γ) does *not* admit a sparsification of size $\mathcal{O}(n^{2-\varepsilon})$ for any $\varepsilon > 0$. Consequently, we obtain a characterization of the sparsification complexity of NP-complete Boolean CSPs whose constraint language consists of symmetric relations: there is a linear sparsification if and only if the constraint language is balanced. This yields linear sparsifications in several new scenarios that were not known before.

Characterization of sparsification complexity for CSPs of low arity (Section 6). By combining the linear sparsifications guaranteed by balanced constraint languages with the nontrivial sparsification when the largest-arity relations do not have exactly one falsifying assignment, we obtain an exact characterization of the optimal sparsification size for all Boolean CSPs where each relation has arity at most three. For a Boolean constraint language Γ consisting of relations of arity at most three, we characterize the sparsification complexity of Γ as an integer $k \in \{1,2,3\}$ that represents the largest OR that Γ can implement using constants and negations, but without introducing fresh variables. Then we prove that CSP(Γ) has a sparsification of size $\mathcal{O}(n^k)$, but no sparsification of size $\mathcal{O}(n^{k-\varepsilon})$ for any $\varepsilon > 0$, giving matching upper and lower bounds. Hence for all Boolean CSPs with constraints of arity at most three, the polynomial-based framework gives provably *optimal* sparsifications in all cases.

2 Preliminaries

For a positive integer n, define $[n] := \{1,2,\ldots,n\}$. For an integer q, we let $\mathbb{Z}/q\mathbb{Z}$ denote the integers modulo q. These form a field if q is prime, and a ring otherwise. We will use $x \equiv_q y$ to denote that x and y are congruent modulo q, and $x \not\equiv_q y$ to denote that they are incongruent modulo q. For statements marked with a star (★), the (full) proof can be found in Appendix A.

Parameterized complexity A parameterized problem $\mathcal{Q}$ is a subset of $\Sigma^* \times \mathbb{N}$, where Σ is a finite alphabet. Let $\mathcal{Q}, \mathcal{Q}' \subseteq \Sigma^* \times \mathbb{N}$ be parameterized problems and let $h\colon \mathbb{N} \to \mathbb{N}$ be a computable function. A *generalized kernel for $\mathcal{Q}$ into $\mathcal{Q}'$ of size $h(k)$* is an algorithm that, on input $(x,k) \in \Sigma^* \times \mathbb{N}$, takes time polynomial in $|x| + k$ and outputs an instance (x',k') such that: (i) $|x'|$ and k' are bounded by $h(k)$, and (ii) $(x',k') \in \mathcal{Q}'$ if and only if $(x,k) \in \mathcal{Q}$. The algorithm is a *kernel* for $\mathcal{Q}$ if $\mathcal{Q}' = \mathcal{Q}$.

Since a polynomial-time reduction to an equivalent sparse instance yields a generalized kernel, lower bounds against generalized kernels can be used to prove the non-existence of such sparsification algorithms. To relate the sparsifiability of different problems to each other, the following notion is useful.

▶ **Definition 2.1.** Let $\mathcal{P}, \mathcal{Q} \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized problems. A *linear-parameter transformation* from $\mathcal{P}$ to $\mathcal{Q}$ is a polynomial-time algorithm that, given an instance $(x,k) \in \Sigma^* \times \mathbb{N}$ of $\mathcal{P}$, outputs an instance $(x',k') \in \Sigma^* \times \mathbb{N}$ of $\mathcal{Q}$ such that the following holds:

1. $(x,k) \in \mathcal{Q}$ if and only if $(x',k') \in \mathcal{P}$, and
2. $k' \in \mathcal{O}(k)$.

It is well-known (cf. [1, 2]) that the existence of a polynomial-time linear-parameter transformation from problem $\mathcal{P}$ to $\mathcal{Q}$ implies that any generalized kernelization lower bound for $\mathcal{P}$, also holds for $\mathcal{Q}$.

Operations, relations, and preservation A *Boolean operation* is a mapping from $\{0,1\}^k$ to $\{0,1\}$, where k, a natural number, is said to be the *arity* of the operation; we assume throughout that operations have positive arity. From here, we define a *partial Boolean operation* in the usual way, that is, it is a mapping from a subset of $\{0,1\}^k$ to $\{0,1\}$. We say that a partial Boolean operation f of arity k is *idempotent* if $f(0,\ldots,0) = 0$ and $f(1,\ldots,1) = 1$; and, *self-dual* if for all $(a_1,\ldots,a_k) \in \{0,1\}^k$, when $f(a_1,\ldots,a_k)$ is defined, it holds that $f(\neg a_1,\ldots,\neg a_k)$ is defined and $f(a_1,\ldots,a_k) = \neg f(\neg a_1,\ldots,\neg a_k)$.

▶ **Definition 2.2.** A partial Boolean operation $f : \{0,1\}^k \to \{0,1\}$ is *balanced* if there exist integer values $\alpha_1,\ldots,\alpha_k$, called the *coefficients* of f, such that

- $\sum_i^k \alpha_i = 1$,
- $(x_1,\ldots,x_k)$ is in the domain of f if and only if $\sum_{i\in[k]} \alpha_i x_i \in \{0,1\}$, and
- $f(x_1,\ldots,x_k) = \sum_{i\in[k]} \alpha_i x_i$ for all tuples in its domain.

A *relation* over the set D is a subset of D^k; here, k is a natural number called the *arity* of the relation. Throughout, we assume that each relation is over a finite set D. A *Boolean relation* is a relation over $\{0,1\}$.

▶ **Definition 2.3.** For each $k \geq 1$, we use k-OR to denote the relation $\{0,1\}^k \setminus \{(0,\ldots,0)\}$.

A *constraint language over D* is a finite set of relations over D; a *Boolean constraint language* is a constraint language over $\{0,1\}$. For a Boolean constraint language Γ, we define CSP(Γ) as follows.

CSP(Γ)	**Parameter:** The number of variables $\lvert V\rvert$.
Input: A tuple $(\mathcal{C}, V)$, where $\mathcal{C}$ is a finite set of constraints, V is a finite set of variables, and each constraint is a pair $R(x_1,\ldots,x_k)$ for $R \in \Gamma$ and $x_1,\ldots,x_k \in V$.	
Question: Does there exist a *satisfying assignment*, that is, an assignment $f\colon V \to \{0,1\}$ such that for each constraint $R(x_1,\ldots,x_k) \in \mathcal{C}$ it holds that $(f(x_1),\ldots,f(x_k)) \in R$?	

Let $f : \{0,1\}^k \to \{0,1\}$ be a partial Boolean operation, and let $T \subseteq \{0,1\}^n$ be a Boolean relation. We say that T is *preserved* by f when, for any tuples $t^1 = (t^1_1, \ldots, t^1_n), \ldots, t^k = (t^k_1, \ldots, t^k_n) \in T$, if all entries of the tuple $(f(t^1_1, \ldots, t^k_1), \ldots, f(t^1_n, \ldots, t^k_n))$ are defined, then this tuple is in T. We say that a Boolean constraint language Γ is *preserved* by f if each relation in Γ is preserved by f. We say that a Boolean relation is *balanced* if it is preserved by all balanced operations, and that a Boolean constraint language is *balanced* if each relation therein is balanced.

Define an *alternating operation* to be a balanced operation $f : \{0,1\}^k \to \{0,1\}$ such that k is odd and the coefficients alternate between $+1$ and -1, so that $\alpha_1 = +1$, $\alpha_2 = -1$, $\alpha_3 = +1$, …, $\alpha_k = +1$. We have the following.

▶ **Proposition 2.4 (★).** *A relation is balanced if and only if it is preserved by all alternating operations.*

We will use the following straightforwardly verified fact tacitly, throughout.

▶ **Observation 2.5.** *Each balanced operation is idempotent and self-dual.*

For $b \in \{0,1\}$, let $u_b : \{0,1\} \to \{0,1\}$ be the unary operation defined by $u_b(0) = u_b(1) = b$; let $\mathsf{major} : \{0,1\}^3 \to \{0,1\}$ to be the operation defined by $\mathsf{major}(x,y,z) = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$; and, let $\mathsf{minor} : \{0,1\}^3 \to \{0,1\}$ to be the operation defined by $\mathsf{minor}(x,y,z) = x \oplus y \oplus z$, where $\oplus$ denotes exclusive OR. We say that a Boolean constraint language Γ is *tractable* if it is preserved by one of the six following operations: u_0, u_1, $\wedge$, $\vee$, minor, major; we say that Γ is *intractable* otherwise. It is known that, in terms of classical complexity, the problem $\mathsf{CSP}(\Gamma)$ is polynomial-time decidable when Γ is tractable, and that the problem $\mathsf{CSP}(\Gamma)$ is NP-complete when Γ is intractable (see [4] for a proof; in particular, refer there to the proof of Theorem 3.21).

Constraint Satisfaction and Definability

▶ **Assumption 2.6.** *By default, we assume in the sequel that the operations, relations, and constraint languages under discussion are Boolean, and that said operations and relations are of positive arity. We nonetheless sometimes describe them as being Boolean, for emphasis.*

▶ **Definition 2.7.** Let us say that a Boolean relation T of arity m is *cone-definable* from a Boolean relation U of arity n if there exists a tuple $(y_1, \ldots, y_n)$ where:

- for each $j \in [n]$, it holds that y_j is an element of $\{0,1\} \cup \{x_1, \ldots, x_m\} \cup \{\neg x_1, \ldots, \neg x_m\}$;
- for each $i \in [m]$, there exists $j \in [n]$ such that $y_j \in \{x_i, \neg x_i\}$; and,
- for each $f : \{x_1, \ldots, x_m\} \to \{0,1\}$, it holds that $(f(x_1), \ldots, f(x_m)) \in T$ if and only if $(\hat{f}(y_1), \ldots, \hat{f}(y_n)) \in U$. Here, $\hat{f}$ denotes the natural extension of f where $\hat{f}(0) = 0$, $\hat{f}(1) = 1$, and $\hat{f}(\neg x_i) = \neg f(x_i)$.

(The prefix *cone* indicates the allowing of **co**nstants and **ne**gation.)

▶ **Example 2.8.** Let $R = \{(0,0),(0,1)\}$ and let $S = \{(0,1),(1,1)\}$. We have that R is cone-definable from S via the tuple $(\neg x_2, \neg x_1)$; also, S is cone-definable from R via the same tuple.

When Γ is a constraint language over D, we use Γ^* to denote the expansion of Γ where each element of D appears as a relation, that is, we define Γ^* as $\Gamma \cup \{\{(d)\} \mid d \in D\}$.

The following is a key property of cone-definability; it states that relations that are cone-definable from a constraint language Γ may be simulated by the constraint language, and thus used to prove hardness results for $\mathsf{CSP}(\Gamma)$.

▶ **Proposition 2.9 (★).** *Suppose that* Γ *is an intractable constraint language, and that* Δ *is a constraint language such that each relation in* Δ *is cone-definable from a relation in* Γ. *Then, there exists a linear-parameter transformation from* $\mathsf{CSP}(\Gamma^* \cup \Delta)$ *to* $\mathsf{CSP}(\Gamma)$.

3 Trivial versus non-trivial sparsification

It is well known that k-CNF-SAT allows no non-trivial sparsification, for each $k \geq 3$ [7]. This means that we cannot efficiently reduce the number of clauses in such a formula to $\mathcal{O}(n^{k-\varepsilon})$. The k-OR relation is special, in the sense that there is exactly one k-tuple that is not contained in the relation. We show in this section that when considering k-ary relations for which there is more than one k-tuple not contained in the relation, a non-trivial sparsification is always possible. In particular, the number of constraints of any input can efficiently be reduced to $\mathcal{O}(n^{k-1})$. Using Lemmas 3.4 and 3.6, we can completely classify the constraint languages that allow a non-trivial sparsification.

▶ **Theorem 3.1 (★).** *Let* Γ *be an intractable constraint language. Let* k *be the maximum arity of any relation* $R \in \Gamma$. *The following dichotomy holds.*

- *If for all* $R \in \Gamma$ *it holds that* $|R| \neq 2^k - 1$, *then* $\mathsf{CSP}(\Gamma)$ *has a kernel with* $\mathcal{O}(n^{k-1})$ *constraints that can be stored in* $\mathcal{O}(n^{k-1} \log n)$ *bits.*
- *If there exists* $R \in \Gamma$ *with* $|R| = 2^k - 1$, *then* $\mathsf{CSP}(\Gamma)$ *has no generalized kernel of bitsize* $\mathcal{O}(n^{k-\varepsilon})$ *for any* $\varepsilon > 0$, *unless* $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.

To obtain the kernels given in this section, we will heavily rely on the following procedure based on representing constraints by polynomials.

▶ **Definition 3.2.** Let R be a k-ary Boolean relation. We say that a polynomial p_u *captures* an unsatisfying assignment $u \in \{0,1\}^k \setminus R$ with respect to R over $\mathbb{Z}/q_u\mathbb{Z}$ or over $\mathbb{Z}$, if the following two conditions hold (over $\mathbb{Z}/q_u\mathbb{Z}$, or over $\mathbb{Z}$ respectively).

$$p_u(x_1, \ldots, x_k) \equiv 0 \text{ for all } (x_1, \ldots, x_k) \in R, \text{ and} \tag{1}$$

$$p_u(u_1, \ldots, u_k) \not\equiv 0. \tag{2}$$

The following Theorem is a generalization of Theorem 16 in [15]. The main improvement is that we now allow the usage of different polynomials, over different rings, for each $u \notin R$. Previously, all polynomials had to be given over the same ring, and each constraint was captured by a single polynomial.

▶ **Theorem 3.3 (★).** *Let* $R \subseteq \{0,1\}^k$ *be a fixed* k*-ary relation, such that for every* $u \in \{0,1\}^k \setminus R$ *there exists a polynomial* p_u *or degree at most* d *(and optionally, a prime power* q_u*) that captures* u *with respect to* R *(over* $\mathbb{Z}/q_u\mathbb{Z}$ *or over* $\mathbb{Z}$*). Then there exists a polynomial-time algorithm that, given a set of constraints* $\mathcal{C}$ *over* $\{R\}$ *over* n *variables, outputs* $\mathcal{C}' \subseteq \mathcal{C}$ *with* $|\mathcal{C}'| = \mathcal{O}(n^d)$, *such that any Boolean assignment satisfies all constraints in* $\mathcal{C}$ *if and only if it satisfies all constraints in* $\mathcal{C}'$.

The next lemma states that any k-ary relation R with $|\{0,1\}^k \setminus R| > 1$ admits a non-trivial sparsification. To prove the lemma, we show that such relations can be represented by polynomials of degree at most $k-1$, such that the sparsification can be obtained using Theorem 3.3. Since relations with $|R| = 2^k$ have a sparsification of size $\mathcal{O}(1)$, as constraints over such languages are satisfied by any assignment, it will follow that k-ary relations with $|\{0,1\}^k \setminus R| \neq 1$ always allow a non-trivial sparsification.

▶ **Lemma 3.4 (★).** *Let R be a k-ary relation with $|R| < 2^k - 1$. Let $\mathcal{C}$ be a set of constraints over $\{R\}$, using n variables. Then there exists a polynomial-time algorithm that outputs $\mathcal{C}' \subseteq \mathcal{C}$ with $|\mathcal{C}'| = \mathcal{O}(n^{k-1})$, such that a Boolean assignment satisfies all constraints in $\mathcal{C}'$ if and only if it satisfies all constraints in $\mathcal{C}$.*

Proof sketch. We will show that for every $u \in \{0,1\}^k \setminus R$, there exists a degree-$(k-1)$ polynomial p_u over the integers that captures u, such that the result follows from Theorem 3.3. We will prove the existence of such a polynomial by induction on k. For $k = 1$, the lemma statement implies that $R = \emptyset$. Thereby, for any $u \notin R$, we simply choose $p_u(x_1) := 1$. This polynomial satisfies the requirements, and has degree 0. Let $k > 1$ and let $u = (u_1, \ldots, u_k) \in \{0,1\}^k \setminus R$. Since $|R| < 2^k - 1$, we can choose $w = (w_1, \ldots, w_k)$ such that $w \in \{0,1\}^k \setminus R$ and $w \neq u$. We distinguish two cases, depending on whether u and w agree on some position.

Suppose $u_i \neq w_i$ for all i, and assume for concreteness that $u = (0, \ldots, 0)$ and $w = (1, \ldots, 1)$. Then the polynomial $p_u(x_1, \ldots, x_k) := \prod_{i=1}^{k-1}(i - \sum_{j=1}^{k} x_j)$ suffices: $p_u(0, \ldots, 0) = \prod_{j=1}^{k-1} j \neq 0$, while for any $(x_1, \ldots, x_k) \in R$, it holds that $\sum_{i=1}^{k} x_i \in [k-1]$ and thereby $p_u(x_1, \ldots, x_k) = 0$; the product has a 0-term. Other values of u and w are handled similarly.

Now suppose $u_i = w_i$ for some $i \in [k]$, and assume for concreteness that $u_1 = w_1 = 1$. Define $R' := \{(x_2, \ldots, x_k) \mid (1, x_2, \ldots, x_k) \in R\}$ and let $u' := (u_2, \ldots, u_k)$. Since $(u_2, \ldots, u_k)$ and $(w_2, \ldots, w_k)$ are distinct tuples not in R', by induction there is a polynomial $p_{u'}$ of degree $k-2$ that captures u' with respect to R'. Then the polynomial $p_u(x_1, \ldots, x_k) := x_1 \cdot p_{u'}(x_2, \ldots, x_k)$ has degree $k-1$ and captures u with respect to R. ◀

To show the other part of the dichotomy, we will need the following theorem.

▶ **Theorem 3.5 (★).** *Let Γ be an intractable constraint language, and let $k \geq 1$. If there exists $R \in \Gamma$ such that R cone-defines k-OR, then $\mathsf{CSP}(\Gamma)$ does not have a generalized kernel of size $\mathcal{O}(n^{k-\varepsilon})$, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.*

The next lemma formalizes the idea that any k-ary relation with $|\{0,1\}^k \setminus R| = 1$ is equivalent to k-OR, up to negation of variables. The proof of the dichotomy given in Theorem 3.1 will follow from Lemma 3.4, together with the next lemma and Theorem 3.5.

▶ **Lemma 3.6 (★).** *Let R be a k-ary relation with $|R| = 2^k - 1$. Then R cone-defines k-OR.*

4 From balanced operations to linear sparsification

The main result of this section is the following theorem, which we will prove later in this section.

▶ **Theorem 4.1.** *Let Γ be a balanced constraint language. Then $\mathsf{CSP}(\Gamma)$ has a kernel with $\mathcal{O}(n)$ constraints that are a subset of the original constraints. The kernel can be stored using $\mathcal{O}(n \log n)$ bits.*

To prove the theorem, we will use two additional technical lemmas. To state them, we introduce some notions from linear algebra. Given a set $S = \{s_1, \ldots, s_n\}$ of k-ary vectors in $\mathbb{Z}^k$, we define $\text{span}_{\mathbb{Z}}(S)$ as the set of all vectors y in $\mathbb{Z}^d$ for which there exist $\alpha_1, \ldots, \alpha_n \in \mathbb{Z}$ such that $y = \sum_{i \in [n]} \alpha_i s_i$. Similarly, we define $\text{span}_q(S)$ as the set of all k-ary vectors y over $\mathbb{Z}/q\mathbb{Z}$, such that there exist $\alpha_1, \ldots, \alpha_n$ such that $y \equiv_q \sum_{i \in [n]} \alpha_i s_i$. For an $m \times n$ matrix S, we use s_i for $i \in [m]$ to denote the i'th row of S.

▶ **Lemma 4.2 (★).** *Let S be an $m \times n$ integer matrix. Let $u \in \mathbb{Z}^n$ be a row vector. If $u \in span_q(\{s_1, \ldots, s_m\})$ for all prime powers q, then $u \in span_{\mathbb{Z}}(\{s_1, \ldots, s_m\})$.*

▶ **Lemma 4.3 (★).** *Let q be a prime power. Let A be an $m \times n$ matrix over $\mathbb{Z}/q\mathbb{Z}$. Suppose there exists no constant $c \not\equiv_q 0$, such that the system $Ax \equiv_q b$ has a solution, where $b := (0, \ldots, 0, c)^T$ is the vector with c on the last position and zeros in all other positions.*

Then $a_m \in span_q(\{a_1, \ldots, a_{m-1}\})$.

Proof of Theorem 4.1. We show that for all $R \in \Gamma$, for all $u \notin R$, there exists a linear polynomial p_u that captures u with respect to R. It will then follow from Theorem 3.3, that given an instance for $\mathsf{CSP}(\Gamma)$ on n variables, we can reduce it to an equivalent instance with $\mathcal{O}(n)$ constraints. Since each constraint uses constantly many variables, we can store the resulting instance using $\mathcal{O}(n \log n)$ bits, as desired.

Suppose for contradiction that there exists $R \in \Gamma$ and $u \notin R$, such that no prime power q and polynomial p exists that satisfy conditions (1) and (2). We can view the process of finding such a linear polynomial, as solving a set of linear equations whose unknowns are the coefficients of the polynomial. We have a linear equation for each evaluation of the polynomial for which we want to enforce a certain value.

Let $R = \{r_1, \ldots, r_\ell\}$. By the non-existence of p and q, the following system has no solution for any prime power q

$$\begin{pmatrix} 1 & r_{1,1} & r_{1,2} & \ldots & r_{1,k} \\ 1 & r_{2,1} & r_{2,2} & \ldots & r_{2,k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & r_{\ell,1} & r_{\ell,2} & \ldots & r_{\ell,k} \\ 1 & u_1 & u_2 & \ldots & u_k \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_k \end{pmatrix} \equiv_q \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ c \end{pmatrix}$$

and for any $c \not\equiv_q 0$. Otherwise, it is easy to verify that q is the desired prime power and $p(x_1, \ldots, x_k) := \alpha_0 + \sum_{i=1}^{k} \alpha_i x_i$ is the desired polynomial.

The fact that no solution exists, implies that $(1, u_1, \ldots, u_k)$ is in the span of the remaining rows of the matrix, by Lemma 4.3.

But this implies that for any prime power q, there exist coefficients $\beta_1, \ldots, \beta_\ell$ over $\mathbb{Z}/q\mathbb{Z}$ such that $u \equiv_q \sum \beta_i r_i$. Furthermore, since the first column of the matrix is the all-ones column, we obtain that $\sum \beta_i \equiv_q 1$. By Lemma 4.2, it follows that there exist integer coefficients $\gamma_1, \ldots, \gamma_\ell$ such that $\sum \gamma_i = 1$ and furthermore $u = \sum \gamma_i r_i$. But it immediately follows that R is not preserved by the balanced operation given by $f(x_1, \ldots, x_\ell) := \sum \gamma_i x_i$, which is a contradiction. ◀

The kernelization result above is obtained by using the fact that when Γ is balanced, the constraints in $\mathsf{CSP}(\Gamma)$ can be replaced by linear polynomials. We show in the next theorem that this approach fails when Γ is not balanced.

▶ **Theorem 4.4 (★).** *Let R be a k-ary relation that is not balanced. Then there exists $u \in \{0,1\}^k \setminus R$ such that there exists no polynomial p_u such that p_u captures u with respect to R over $\mathbb{Z}$, and no integer q such that p_u captures u over $\mathbb{Z}/q\mathbb{Z}$.*

5 Characterization of symmetric CSPs with linear sparsification

In this section, we characterize constraint languages Γ such that $\mathsf{CSP}(\Gamma)$ has a linear sparsification, under the assumption that Γ is symmetric.

▶ **Definition 5.1.** We say a k-ary Boolean relation R is *symmetric*, if there exists $S \subseteq \{0, 1, \ldots, k\}$ such that a tuple $x = (x_1, \ldots, x_k)$ is in R if and only if $\text{weight}(x) \in S$. We call S the set of *satisfying weights* for R.

We will say that a constraint language Γ is symmetric, if it only contains symmetric relations. We will prove the following theorem at the end of this section.

▶ **Theorem 5.2.** *Let Γ be a Boolean symmetric intractable constraint language.*

- *If Γ is balanced, $\mathsf{CSP}(\Gamma)$ has a kernel with $\mathcal{O}(n)$ constraints that can be stored in $\mathcal{O}(n \log n)$ bits.*
- *If Γ is not balanced, then $\mathsf{CSP}(\Gamma)$ does not have a generalized kernel of size $\mathcal{O}(n^{2-\varepsilon})$ for any $\varepsilon > 0$, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.*

To show this, we use the following lemma.

▶ **Lemma 5.3 (★).** *Let R be a symmetric relation with satisfying weights $S \subseteq \{0, 1, \ldots, k\}$. Let $U := \{0, 1, \ldots, k\} \setminus S$. If there exist $a, b, c \in S$ and $d \in U$ such that $a - b + c = d$, then R cone-defines* 2-OR*.*

Proof sketch. We will demonstrate the result in the case that $b \leq a$, $b \leq c$, and $b \leq d$; the other cases are similar. We use the following tuple to express $x_1 \vee x_2$.

$$(\underbrace{\neg x_1, \ldots, \neg x_1}_{(a-b) \text{ copies}}, \underbrace{\neg x_2, \ldots, \neg x_2}_{(c-b) \text{ copies}}, \underbrace{1, \ldots, 1}_{b \text{ copies}}).$$

Let $f : \{x_1, x_2\} \to \{0, 1\}$, then $(\neg f(x_1), \ldots, \neg f(x_1), \neg f(x_2), \ldots, \neg f(x_2), 1, \ldots, 1)$ has weight $d \notin S$ when $f(x_1) = f(x_2) = 0$. It is easy to verify that in all other cases, the weight is either a, b, or c. In these cases the tuple is contained in R, as the weight is contained in S. ◀

We now give the main lemma that is needed to prove Theorem 5.2. It shows that if a relation is symmetric and not balanced, it must cone-define 2-OR.

▶ **Lemma 5.4.** *Let R be a symmetric relation of arity k. If R is not balanced, then R cone-defines* 2-OR*.*

Proof. Let f be a balanced operation that does not preserve R. Since f has integer coefficients, it follows that there exist (not necessarily distinct) $r_1, \ldots, r_m \in R$, such that $r_1 - r_2 + r_3 - r_4 \cdots + r_m = u$ for some $u \in \{0, 1\}^k \setminus R$. Thereby, $\text{weight}(r_1) - \text{weight}(r_2) + \text{weight}(r_3) - \text{weight}(r_4) \cdots + \text{weight}(r_m) = \text{weight}(u)$. Let S be the set of satisfying weights for R and let $U := \{0, \ldots, k\} \setminus S$. Define $s_i := \text{weight}(r_i)$ for $i \in [m]$, and $t = \text{weight}(u)$, such that $s_1 - s_2 + s_3 - s_4 \ldots + s_m = t$, and furthermore $s_i \in S$ for all i, and $t \in U$. We show that there exist $a, b, c \in S$ and $d \in U$ such that $a - b + c = d$, such that the result follows from Lemma 5.3. We do this by induction on the length of the alternating sum.

If $m = 3$, we have that $s_1 - s_2 + s_3 = t$ and define $a := s_1$, $b := s_2$, $c := s_3$, and $d := t$.

If $m > 3$, we will use the following claim.

▶ **Claim 5.5 (★).** *Let $s_1, \ldots, s_m \in S$ and $t \in U$ such that $s_1 - s_2 + s_3 - s_4 \cdots + s_m = t$. There exist distinct $i, j, \ell \in [m]$ with i, j odd and ℓ even, such that $s_i - s_\ell + s_j \in \{0, \ldots, k\}$.*

Use Claim 5.5 to find i, j, ℓ such that $s_i - s_\ell + s_j \in \{0, \ldots, k\}$. We consider two options. If $s_i - s_\ell + s_j \in U$, then define $d := s_i - s_\ell + s_j$, $a := s_i$, $b := s_\ell$, and $c := s_j$ and we are done. The other option is that $s_i - s_\ell + s_j = s \in S$. Replacing $s_i - s_j + s_k$ by s in $s_1 - s_2 + s_3 - s_4 \cdots + s_m$ gives a shorter alternating sum with result t. We obtain a, b, c, and d by the induction hypothesis.

Thereby, we have obtained $a, b, c \in S$, $d \in U$ such that $a - b + c = d$. It now follows from Lemma 5.3 that R cone-defines 2-OR. ◀

Using the lemma above, we can now prove Theorem 5.2.

Proof of Theorem 5.2. If Γ is balanced, it follows from Theorem 4.1 that $\mathsf{CSP}(\Gamma)$ has a kernel with $\mathcal{O}(n)$ constraints that can be stored in $\mathcal{O}(n \log n)$ bits. Note that the assumption that Γ is symmetric is not needed in this case.

If Γ is not balanced, it follows that Γ contains a relation R that is not balanced. It follows from Lemma 5.4, that R cone-defines the 2-OR relation. Thereby, we obtain from Theorem 3.5 that $\mathsf{CSP}(\Gamma)$ has no generalized kernel of size $\mathcal{O}(n^{2-\varepsilon})$ for any $\varepsilon > 0$, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. ◀

6 Low-arity classification

In this section, we will give a full classification of the sparsifiability for constraint languages that consist only of low-arity relations. The next results will show that in this case, if the constraint language is not balanced, it can cone-define the 2-OR relation.

▶ **Observation 6.1.** *Each relation of arity* 1 *is balanced.*

▶ **Theorem 6.2 (★).** *A relation of arity* 2 *is balanced if and only if it is not cone-interdefinable with the* 2-OR *relation.*

▶ **Theorem 6.3 (★).** *Suppose that* $U \subseteq \{0,1\}^3$ *is an arity* 3 *Boolean relation that is not balanced. Then, the* 2-OR *relation is cone-definable from* U*.*

Combining the results in this section with the results in previous sections, allows us to give a full classification of the sparsifiability of constraint languages that only contain relations of arity at most three. Observe that any k-ary relation R such that $R \neq \emptyset$ and $\{0,1\}^k \setminus R \neq \emptyset$ cone-defines the 1-OR relation. Since we assume that Γ is intractable in the next theorem, it follows that k is always defined and $k \in \{1,2,3\}$.

▶ **Theorem 6.4.** *Let* Γ *be an intractable Boolean constraint language such that each relation therein has arity* ≤ 3*. Let* $k \in \mathbb{N}$ *be the largest value for which* k*-OR can be cone-defined from a relation in* Γ*. Then* $\mathsf{CSP}(\Gamma)$ *has a kernel with* $\mathcal{O}(n^k)$ *constraints that can be encoded in* $\mathcal{O}(n^k \log k)$ *bits, but for any* $\varepsilon > 0$ *there is no kernel of size* $\mathcal{O}(n^{k-\varepsilon})$*, unless* $\mathsf{NP} \subseteq \mathsf{coNP/poly}$*.*

Proof. To show that there is a kernel with $\mathcal{O}(n^k)$ constraints, we do a case distinction on k.

- ($k = 1$) If $k = 1$, there is no relation in Γ that cone-defines the 2-OR relation. It follows from Observation 6.1 and Theorems 6.2 and 6.3 that thereby, Γ is balanced. It now follows from Theorem 4.1 that $\mathsf{CSP}(\Gamma)$ has a kernel with $\mathcal{O}(n)$ constraints that can be stored in $\mathcal{O}(n \log n)$ bits.
- ($k = 2$) If $k = 2$, there is no relation $R \in \Gamma$ with $|R| = 2^3 - 1 = 7$, as otherwise by Lemma 3.6 such a relation R would cone-define 3-OR which is a contradiction. Thereby, it follows from Theorem 3.1 that $\mathsf{CSP}(\Gamma)$ has a sparsification with $\mathcal{O}(n^{3-1}) = \mathcal{O}(n^2)$ constraints that can be encoded in $\mathcal{O}(n^2 \log n)$ bits.
- ($k = 3$) Given an instance $(\mathcal{C}, V)$, it is easy to obtain a kernel of with $\mathcal{O}(n^3)$ constraints by simply removing duplicate constraints. This kernel can be stored in $\mathcal{O}(n^3)$ bits, by storing for each relation $R \in \Gamma$ and for each tuple $(x_1, x_2, x_3) \in V^3$ whether $R(x_1, x_2, x_3) \in \mathcal{C}$. Since $|\Gamma|$ is constant and there are $\mathcal{O}(n^3)$ such tuples, this results in using $\mathcal{O}(n^3)$ bits.

It remains to prove the lower bound. By definition, there exists $R \in \Gamma$ such that R cone-defines the k-OR relation. Thereby, the result follows immediately from Theorem 3.5. Thus, $\mathsf{CSP}(\Gamma)$ has no kernel of size $\mathcal{O}(n^{k-\varepsilon})$ for any $\varepsilon > 0$, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. ◀

7 Conclusion

In this paper, we studied the best-case and worst-case sparsifiability of $\mathsf{CSP}(\Gamma)$ for intractable constraint languages. First of all, we studied which constraint languages allow for non-trivial sparsification. If k is the largest arity of any relation in Γ, $\mathsf{CSP}(\Gamma)$ has a trivial sparsification of size $\mathcal{O}(n^k)$. We introduced the notion of cone-definability and have shown that when no relation in Γ cone-defines k-OR, a non-trivial sparsification is always possible. By showing that if a relation cone-defines k-OR, it has no $\mathcal{O}(n^{k-\varepsilon})$ sparsification, we obtained a full classification of Boolean CSPs that admit a non-trivial sparsification.

Secondly, we introduced the notion of balanced constraint languages and have shown that $\mathsf{CSP}(\Gamma)$ allows a sparsification with $\mathcal{O}(n)$ constraints whenever Γ is balanced. The constructive proof of this statement can be transformed into an effective algorithm to find a series of low-degree polynomials to capture the constraints, which earlier had to be done by hand. By combining the resulting upper and lower bound framework, we fully classified the symmetric constraint languages for which $\mathsf{CSP}(\Gamma)$ allows a linear sparsification. Furthermore, we fully classified the sparsifiability of $\mathsf{CSP}(\Gamma)$ when Γ contains relations of arity at most three. To obtain these results, we combined the above results with the additional insight that low-arity or symmetric relations that are not balanced, always cone-define the 2-OR relation.

The goal of this line of research is to fully classify the sparsifiability of $\mathsf{CSP}(\Gamma)$, depending on Γ. In particular, we would like to classify those Γ for which $\mathcal{O}(n)$ sparsifiability is possible. In this paper, we have shown that Γ being balanced is a sufficient condition to obtain a linear sparsification. We conjecture that for intractable Boolean constraint languages Γ, $\mathsf{CSP}(\Gamma)$ admits no $\mathcal{O}(n)$ sparsification if Γ is not balanced. If true, this would give a full classification of the constraint languages that allow for linear sparsification.

One way to prove the conjecture, would be to show that if Γ is not balanced, it cone-defines 2-OR and therefore has no subquadratic sparsification. As we have seen, this is true for both the low-arity and the symmetric case. Unfortunately, this does not hold in general, as can be shown by an example inspired by a construction of Lagerkvist and Wahlström [17, p.169]. Let Γ consist of a single relation R, we start by defining the balanced relation that will not preserve R. Let $f(x_1, \ldots, x_5) := x_1 - x_2 + x_3 - x_4 + x_5$ and let the domain D_f of f consist of those tuples $(x_1, \ldots, x_5) \in \{0,1\}^5$ for which $x_1 - x_2 + x_3 - x_4 + x_5 \in \{0,1\}$. We now define R as a $|D_f|$-ary relation with $R = \{r_1, r_2, r_3, r_4, r_5\}$, as follows. Let $D_f = \{d_1, \ldots, d_m\}$ and let $d_i = \{d_{i,1}, \ldots, d_{i,5}\}$ for $i \in [m]$. Define $r_{i,j} := d_{j,i}$ for $i \in [5]$ and $j \in [m]$. It can be shown that R is not balanced by observing that $f(r_1, \ldots, r_5) \notin R$. However, R does not cone-define the 2-OR relation: by construction it is preserved by all alternating operations of arity three (because its outcome is never defined when applied to three distinct tuples from R), while a cone-definition of 2-OR would yield a violation by an arity-three alternating operation.

Luckily, cone-defining the 2-OR relation is not strictly necessary to obtain a non-linear lower bound. For example, a polynomial parameter transformation from 4-CNF-SAT to $\mathsf{CSP}(\Gamma)$ using at most $\mathcal{O}(n^2)$ additional variables, also shows that no $\mathcal{O}(n^{2-\varepsilon})$ sparsification exists, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. We thereby leave as an open question whether there exist non-balanced constraint languages for which $\mathsf{CSP}(\Gamma)$ allows a linear sparsification. Our suggested characterization in terms of balanced operations can be compared to that of Lagerkvist and Wahlström [17]. They suggest a weaker condition on constraint languages, and show that it is necessary and sufficient for $\mathsf{CSP}(\Gamma)$ having a Maltsev embedding over an infinite domain (which does not have immediate consequences for sparsification). Our condition of being balanced is a stronger requirement, but as we have shown, leads directly to linear sparsification.

References

1. Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Kernelization lower bounds by cross-composition. *SIAM J. Discrete Math.*, 28(1):277–305, 2014. doi:10.1137/120880240.
2. Hans L. Bodlaender, Stéphan Thomassé, and Anders Yeo. Kernel bounds for disjoint cycles and disjoint paths. *Theor. Comput. Sci.*, 412(35):4570–4578, 2011. doi:10.1016/j.tcs.2011.04.039.
3. A. Bulatov, P. Jeavons, and A. Krokhin. Classifying the Complexity of Constraints using Finite Algebras. *SIAM Journal on Computing*, 34(3):720–742, 2005. doi:10.1137/S0097539700376676.
4. Hubie Chen. A rendezvous of logic, complexity, and algebra. *ACM Computing Surveys*, 42(1), 2009. doi:10.1145/1189056.1189076.
5. Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.
6. Holger Dell and Dániel Marx. Kernelization of packing problems. In *Proc. 23rd SODA*, pages 68–81, 2012. doi:10.1137/1.9781611973099.6.
7. Holger Dell and Dieter van Melkebeek. Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. *J. ACM*, 61(4):23:1–23:27, 2014. doi:10.1145/2629620.
8. Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.
9. Andrew Drucker. New limits to classical and quantum instance compression. *SIAM J. Comput.*, 44(5):1443–1479, 2015. doi:10.1137/130927115.
10. M.S. Gockenbach. *Finite-Dimensional Linear Algebra*. Discrete Mathematics and Its Applications. Taylor & Francis, 2011.
11. Bart M. P. Jansen. On sparsification for computing treewidth. *Algorithmica*, 71(3):605–635, 2015. doi:10.1007/s00453-014-9924-2.
12. Bart M. P. Jansen and Astrid Pieterse. Optimal sparsification for some binary CSPs using low-degree polynomials. In *Proc. 41st MFCS*, pages 71:1–71:14, 2016. doi:10.4230/LIPIcs.MFCS.2016.71.
13. Bart M. P. Jansen and Astrid Pieterse. Optimal data reduction for graph coloring using low-degree polynomials. In *Proc. 12th IPEC*, pages 22:1–22:12, 2017. doi:10.4230/LIPIcs.IPEC.2017.22.
14. Bart M. P. Jansen and Astrid Pieterse. Sparsification upper and lower bounds for graph problems and not-all-equal SAT. *Algorithmica*, 79(1):3–28, 2017. doi:10.1007/s00453-016-0189-9.
15. Bart M. P. Jansen and Astrid Pieterse. Optimal sparsification for some binary CSPs using low-degree polynomials. *CoRR*, abs/1606.03233, 2018. URL: http://arxiv.org/abs/1606.03233v2.
16. Stefan Kratsch, Geevarghese Philip, and Saurabh Ray. Point line cover: The easy kernel is essentially tight. *ACM Trans. Algorithms*, 12(3):40:1–40:16, 2016. doi:10.1145/2832912.
17. Victor Lagerkvist and Magnus Wahlström. Kernelization of constraint satisfaction problems: A study through universal algebra. In *Proc. 23rd CP*, pages 157–171, 2017. doi:10.1007/978-3-319-66158-2_11.
18. Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Kernelization - Preprocessing with a guarantee. In *The Multivariate Algorithmic Revolution and Beyond*, pages 129–161, 2012. doi:10.1007/978-3-642-30891-8_10.
19. László Lovász. Chromatic number of hypergraphs and linear algebra. In *Studia Scientiarum Mathematicarum Hungarica 11*, pages 113–114, 1976.

559 **20** Thomas J. Schaefer. The complexity of satisfiability problems. In *Proc. 10th STOC*, pages
560 216–226, 1978. doi:10.1145/800133.804350.

A Omitted proofs

A.1 Proofs omitted from Section 2

Proof of Proposition 2.4. It suffices to show that if a relation T is not balanced, then there exists an alternating operation that does not preserve T. Let f be a k-ary balanced operation that does not preserve T. Then there exist tuples $t^1, \ldots, t^k$ in T such that $\alpha_1 t^1 + \cdots + \alpha_k t^k$ is not in T, where the sum of the α_i is equal to 1 (and where we may assume that no α_i is equal to 0). For each positive α_i, replace $\alpha_i t^i$ in the sum with $t^i + \cdots + t^i$ (α_i times); likewise, for each negative α_i, replace $\alpha_i t^i$ in the sum with $-t^i - \cdots - t^i$ ($-\alpha_i$ times). Each tuple then has coefficient $+1$ or -1 in the sum; since the sum of coefficients is $+1$, by permuting the sum's terms, the coefficients can be made to alternate between $+1$ and -1. ◀

For the proof of Proposition 2.9, we will need the following additional theorem.

▶ **Theorem A.1.** *(follows from [3]) Let Γ be a constraint language over a finite set D such that each unary operation $u : D \to D$ that preserves Γ is a bijection. Then, there exists a linear-parameter transformation from $\mathsf{CSP}(\Gamma^*)$ to $\mathsf{CSP}(\Gamma)$.*

Note that in particular, an *intractable* Boolean constraint language can only be preserved by unary operations that are bijections. Hence for intractable Boolean Γ, there is a linear-parameter transformation from $\mathsf{CSP}(\Gamma^*)$ to $\mathsf{CSP}(\Gamma)$.

Proof of Theorem A.1. The desired transformation is the final polynomial-time reduction given in the proof of Theorem 4.7 of [3]. This reduction translates an instance of $\mathsf{CSP}(\Gamma^*)$ with n variables to an instance of $\mathsf{CSP}(\Gamma \cup \{=_D\})$ with $n + |D|$ variables; here, $=_D$ denotes the equality relation on domain D. Each constraint of the form $=_D (v, v')$ may be removed (while preserving satisfiability) by taking one of the variables v, v', and replacing each instance of that variable with the other. The resulting instance of $\mathsf{CSP}(\Gamma)$ has $\leq n + |D|$ variables. ◀

▶ **Definition A.2.** A relation $T \subseteq D^k$ is *pp-definable* (short for *primitive positive definable*) from a constraint language Γ over D if there exists an instance $(\mathcal{C}, V)$ of $\mathsf{CSP}(\Gamma)$ and there exist pairwise distinct variables $x_1, \ldots, x_k \in V$ such that, for each map $f : \{x_1, \ldots, x_k\} \to \{0, 1\}$, it holds that f can be extended to a satisfying assignment of the instance if and only if $(f(x_1), \ldots, f(x_k)) \in T$.

The following is a known fact; for an exposition, we refer the reader to Theorems 3.13 and 5.1 of [4].

▶ **Proposition A.3.** *If Γ is an intractable Boolean constraint language, then every Boolean relation is pp-definable from Γ^*.*

Proof of Proposition 2.9. It suffices to give a linear-parameter transformation from $\mathsf{CSP}(\Gamma^* \cup \Delta)$ to $\mathsf{CSP}(\Gamma^*)$, by Theorem A.1. Let $(\mathcal{C}, V)$ be an instance of $\mathsf{CSP}(\Gamma^* \cup \Delta)$, and let n denote $|V|$. We generate an instance $(\mathcal{C}', V')$ of $\mathsf{CSP}(\Gamma^*)$ as follows.

- For each variable $v \in V$, introduce a primed variable v'. By Proposition A.3, the relation $\neq$ (that is, the relation $\{(0,1),(1,0)\}$) is pp-definable from Γ^*. Fix such a pp-definition, and let d be the number of variables in the definition. For each $v \in V$, include in $\mathcal{C}'$ all constraints in the pp-definition of $\neq$, but where the variables are renamed so that v and v' are the distinguished variables, and the other variables are fresh.
  The number of variables used so far in $\mathcal{C}'$ is nd.
- For each $b \in \{0, 1\}$, introduce a variable z_b, and include the constraint $\{(b)\}(z_b)$ in $\mathcal{C}'$.

- For each constraint $T(v_1, \ldots, v_k)$ in $\mathcal{C}$ such that $T \in \Gamma^*$, include the constraint in $\mathcal{C}'$.
- For each constraint $T(v_1, \ldots, v_k)$ in $\mathcal{C}$ such that $T \in \Delta \setminus \Gamma^*$, we use the assumption that T is cone-definable from a relation in Γ to include a constraint in $\mathcal{C}'$ that has the same effect as $T(v_1, \ldots, v_k)$. In particular, assume that T is cone-definable from $U \in \Gamma$ via the tuple $(y_1, \ldots, y_\ell)$, and that U has arity ℓ. Include in $\mathcal{C}'$ the constraint $U(w_1, \ldots, w_\ell)$, where, for each $i \in [\ell]$, the entry w_i is set to v_j if $y_i = x_j$, v'_j if $y_i = \neg x_j$, z_0 if $y_i = 0$, and z_1 if $y_i = 1$.

The set V' of variables used in $\mathcal{C}'$ is the union of $V \cup \{v' \mid v \in V\} \cup \{z_0, z_1\}$ with the other variables used in the copies of the pp-definition of $\neq$. We have $|V'| = nd + 2$. It is straightforward to verify that an assignment $f : V \to \{0, 1\}$ satisfies $\mathcal{C}$ if and only if there exists an assignment $f' : V' \to \{0, 1\}$ of f that satisfies $\mathcal{C}'$. ◀

A.2 Proofs omitted from Section 3

We start by proving the main Theorem of this section, using the other lemmas in the section.

Proof of Theorem 3.1. Suppose that for all $R \in \Gamma$, it holds that $|R| \neq 2^k - 1$. We give the following kernelization procedure. Suppose we are given an instance of $\mathsf{CSP}(\Gamma)$, with set of constraints $\mathcal{C}$. We show how to define $\mathcal{C}' \subseteq \mathcal{C}$. For each constraint $R(x_1, \ldots, x_\ell) \in \mathcal{C}$ where R is a relation of arity $\ell < k$, add one such constraint to $\mathcal{C}'$ (thus removing duplicate constraints). Note that this adds at most $\mathcal{O}(n^\ell)$ constraints for each ℓ-ary relation $R \in \Gamma$.

For a k-ary relation $R \in \Gamma$, let $\mathcal{C}_R$ contain all constraints of the form $R(x_1, \ldots, x_k)$. For all k-ary relations R with $|R| < 2^k - 1$, apply Lemma 3.4 to obtain $\mathcal{C}'_R \subseteq \mathcal{C}_R$ such that $|\mathcal{C}'_R| = \mathcal{O}(n^{k-1})$ and any Boolean assignment satisfying $\mathcal{C}'_R$ also satisfies $\mathcal{C}_R$. Add $\mathcal{C}'_R$ to $\mathcal{C}'$. This concludes the definition of $\mathcal{C}'$. Note that the procedure removes constraints of the form $R(x_1, \ldots, x_k)$ with $|R| = 2^k$, as these are always satisfied. It is easy to verify that $|\mathcal{C}'| \leq |\Gamma| \cdot \mathcal{O}(n^{k-1}) = \mathcal{O}(n^{k-1})$. Since each constraint can be stored in $\mathcal{O}(\log n)$ bits, this gives a kernel of bitsize $\mathcal{O}(n^{k-1} \log n)$.

Suppose that there exists $R \in \Gamma$ with $|R| = 2^k - 1$. It follows from Lemma 3.6 that R cone-defines k-OR. Since Γ is intractable, it now follows from Theorem 3.5 that $\mathsf{CSP}(\Gamma)$ has no generalized kernel of size $\mathcal{O}(n^{k-\varepsilon})$, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. ◀

To prove Theorem 3.3, we use the following theorem, that was proven by a subset of the current authors [15]. We recall the required terminology. Define d-POLYNOMIAL ROOT CSP over $\mathbb{Z}/m\mathbb{Z}$ as the problem whose input consists of a set L of polynomial equalities over $\mathbb{Z}/m\mathbb{Z}$ of degree at most d, over a set of variables V. Each equality is of the form $p(x_1, \ldots, x_k) \equiv_m 0$. The question is whether there exists a Boolean assignment to the variables in V that satisfies all equalities in L.

▶ **Theorem A.4** ([15, Theorem 16]). *There is a polynomial-time algorithm that, given an instance (L, V) of d-POLYNOMIAL ROOT CSP over $\mathbb{Z}/m\mathbb{Z}$ for some fixed integer $m \geq 2$ with r distinct prime divisors, outputs an equivalent instance (L', V) of d-POLYNOMIAL ROOT CSP over $\mathbb{Z}/m\mathbb{Z}$ with at most $r \cdot (n^d + 1)$ constraints such that $L' \subseteq L$.*

Note that if m is a prime power, m has only one distinct prime divisor and thereby $r = 1$ in the above theorem statement.

Proof of Theorem 3.3. Let $\mathcal{C}$ be a set of constraints over R and let V be the set of variables used. We will create $|\{0, 1\}^k \setminus R|$ instances of d-POLYNOMIAL ROOT CSP over V. For each $u \in R \setminus \{0, 1\}^k$, we create an instance (L_u, V) of d-POLYNOMIAL ROOT CSP over $\mathbb{Z}/q_u\mathbb{Z}$, as follows. Choose q_u and p_u such that they satisfy (1) and (2) for u. For each constraint

$(x_1, \ldots, x_k) \in \mathcal{C}$, add the equality $p_u(x_1, \ldots, x_k) \equiv_{q_u} 0$ to the set L_u. Let $L := \bigcup_{u \notin R} L_u$ be the union of all created sets of equalities. From this construction, we obtain the following claim.

▶ **Claim A.5.** *Any Boolean assignment f that satisfies all equalities in L, satisfies all constraints in $\mathcal{C}$.*

Proof. Let f be a Boolean assignment that satisfies all equalities in L. Suppose f does not satisfy all equalities in $\mathcal{C}$, thus there exists $(x_1, \ldots, x_k) \in \mathcal{C}$, such that $(f(x_1), \ldots, f(x_k)) \notin R$. Let $u := (f(x_1), \ldots, f(x_k))$. Since $u \notin R$, the equation $p_u(x_1, \ldots, x_k) \equiv_{q_u} 0$ was added to $L_u \subseteq L$. However, it follows from (2) that $p_u(f(x_1), \ldots, f(x_k)) \not\equiv_{q_u} 0$, which contradicts the assumption that f satisfies all equalities in L. ⌟

For each instance (L_u, V) of d-Polynomial root CSP, apply Theorem A.4 to obtain an equivalent instance (L'_u, V) with $L'_u \subseteq L_u$ and $|L'_u| = \mathcal{O}(n^d)$. Let $L' := \bigcup L'_u$. By this definition, any Boolean assignment satisfies all equalities in L, if and only if it satisfies all equalities in L'. Construct $\mathcal{C}'$ as follows. For any $(x_1, \ldots, x_k) \in \mathcal{C}$, add $(x_1, \ldots, x_k)$ to $\mathcal{C}'$ if there exists $u \in \{0,1\}^k \setminus R$ such that $p_u(x_1, \ldots, x_k) \equiv_{q_u} 0 \in L'$. Hereby, $\mathcal{C}' \subseteq \mathcal{C}$. The following two claims show the correctness of this sparsification procedure.

▶ **Claim A.6.** *Any Boolean assignment f satisfies all constraints in $\mathcal{C}'$, if and only if it satisfies all constraints in $\mathcal{C}$.*

Proof. Since $\mathcal{C}' \subseteq \mathcal{C}$, it follows immediately that any Boolean assignment satisfying the constraints in $\mathcal{C}$ also satisfies all constraints in $\mathcal{C}'$. It remains to prove the opposite direction.

Let f be a Boolean assignment satisfying all constraints in $\mathcal{C}'$. We show that f satisfies all equalities in L'. Let $p_u(x_1, \ldots, x_k) \equiv_{q_u} 0 \in L'$. Thereby, $(x_1, \ldots, x_k) \in \mathcal{C}'$ and since f is a satisfying assignment, $(f(x_1), \ldots, f(x_k)) \in R$. It follows from property (1) that $p_u(f(x_1), \ldots, f(x_k)) \equiv_{q_u} 0$ as desired.

Since f satisfies all equalities in L', it satisfies all equalities in L by the choice of L'. It follows from Claim A.5 that thereby f satisfies all constraints in $\mathcal{C}$. ⌟

▶ **Claim A.7.** $|\mathcal{C}'| = \mathcal{O}(n^d)$.

Proof. By the construction of $\mathcal{C}'$, it follows that $|\mathcal{C}'| \leq |L'|$. We know $|L'| = \sum_{u \notin R} |L'_u| \leq \sum_{u \notin R} \mathcal{O}(n^d) \leq 2^k \mathcal{O}(n^d) = \mathcal{O}(n^d)$, as k is considered constant. ⌟

The lemma statement now follows from Claims A.6 and A.7. ◀

Proof of Lemma 3.4. We will prove this by showing that for every $u \in \{0,1\}^k \setminus R$, there exists a k-ary polynomial p_u over the integers of degree at most $k-1$ satisfying (1) and (2), such that the result follows from Theorem 3.3.

We will prove the existence of such a polynomial by induction on k. For $k = 1$, the lemma statement implies that $R = \emptyset$. Thereby, for any $u \notin R$, we simply choose $p_u(x_1) := 1$. This polynomial satisfies the requirements, and has degree 0.

Let $k > 1$ and let $u = (u_1, \ldots, u_k) \in \{0,1\}^k \setminus R$. Since $|R| < 2^k - 1$, we can choose $w = (w_1, \ldots, w_k)$ such that $w \in \{0,1\}^k \setminus R$ and $w \neq u$. Choose such w arbitrarily, we now do a case distinction.

(There exists no $i \in [k]$, such that $u_i = w_i$) This implies $u_i = \neg w_i$ for all i. One may note that for $u = (0, \ldots, 0)$ and $w = (1, \ldots, 1)$ this situation corresponds to monotone k-NAE-SAT. We show that there exists a polynomial p_u such that $p_u(u_1, \ldots, u_k) \neq 0$, and $p_u(x_1, \ldots, x_k) = 0$ for all $(x_1, \ldots, x_k) \in R$. Hereby p_u satisfies conditions (1) and (2) for u.

For $i \in [k]$, define $r_i(x) := (1-x)$ if $u_i = 1$ and $r_i(x) := x$ if $u_i = 0$. It follows immediately from this definition, that $r_i(u_i) = 0$ for all $i \in [k]$. Define

$$p_u(x_1, \ldots, x_k) := \prod_{i=1}^{k-1} \left(i - \sum_{j=1}^{k} r_j(x_j) \right).$$

By this definition, p_u has degree $k-1$. It remains to verify that p_u has the desired properties. First of all, since $\sum_{j=1}^{k} r_j(u_j) = 0$ by definition, it follows that

$$p_u(u_1, \ldots, u_k) = \prod_{i=1}^{k-1} i \neq 0,$$

as desired. By the assumption that $u_i \neq w_i$ for all i, we obtain $p_u(w_1, \ldots, w_k) = \prod_{i=1}^{k-1}(i-k) \neq 0$, which is allowed since $w \notin R$. It is easy to verify that in all other cases, $\sum_{j=1}^{k} r_j(x_j) \in \{1, 2, \ldots, k-1\}$ and thereby one of the terms of the product is zero, implying $p_u(x_1, \ldots, x_k) = 0$.

(There exists $i \in [k]$, such that $u_i = w_i$) Let u' and w' be defined as the results of removing coordinate i from u and w respectively. Note that $u' \neq w'$. Define

$$R' := \{(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_k) \mid (x_1, \ldots, x_{i-1}, u_i, x_{i+1}, x_k) \in R\}.$$

By this definition, $u', w' \notin R'$ and thereby R' is a $(k-1)$-ary relation with $|R'| < 2^{k-1} - 1$. By the induction hypothesis, there exists a polynomial $p_{u'}$ of degree at most $k-1$, such that $p_{u'}(u'_1, \ldots, u'_{k-1}) \neq 0$ and $p_{u'}(x'_1, \ldots, x'_{k-1}) = 0$ for all $x' \in R'$. Now define

$$p_u(x_1, \ldots, x_k) := (1 - x_i - u_i) \cdot p_{u'}(x_1, \ldots, x_{i-1}, x_{i+1}, x_k).$$

We show that p_u has the desired properties. By definition, p_u has the degree of $p_{u'}$ plus one. Since $p_{u'}$ has degree $k-2$ by the induction hypothesis, it follows that p_u has degree $k-1$. Let $(x_1, \ldots, x_k) \in R$. We do a case distinction on the value taken by x_i.

- $x_i \neq u_i$. In this case, $(1 - x_i - u_i) = 0$, and thereby $p_u(x_1, \ldots, x_k) = 0$, thus satisfying equation (1).
- $x_i = u_i$. Since $x = (x_1, \ldots, x_{i-1}, u_i, x_{i+1}, \ldots, x_k) \in R$, it follows that $(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_k) \in R'$. By definition of $p_{u'}$, it follows that $p_{u'}(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_k) = 0$ and thus $p_u(x_1, \ldots, x_k) = 0$, showing (1).

It remains to show that $p_u(u_1, \ldots, u_k) \neq 0$. This follows from $(1 - u_i - u_i) \in \{-1, 1\}$, and $p_{u'}(u_1, \ldots, u_{i-1}, u_{i+1}, \ldots, u_k) \neq 0$, showing that (2) holds.

Since we have shown for all $u \in \{0,1\}^k \setminus R$, that there exists a polynomial p_u satisfying (1) and (2), the result follows from Theorem 3.3. ◀

Proof of Theorem 3.5. We do a case distinction on k.

($k = 1$) Suppose that there exists $\varepsilon > 0$ such that $\mathsf{CSP}(\Gamma)$ has a (generalized) kernel of size $\mathcal{O}(n^{1-\varepsilon})$. Using this hypothetical generalized kernel, one could obtain a polynomial-time algorithm that takes as input a series of inputs $(\mathcal{C}_1, V_1), \ldots, (\mathcal{C}_t, V_t)$ of $\mathsf{CSP}(\Gamma)$, and outputs in polynomial time an instance x^* of some fixed decision problem L such that:

- $x^* \in L$ if and only if *all* $(\mathcal{C}_i, V_i)$ are YES-instances of $\mathsf{CSP}(\Gamma)$, and
- x^* has bitsize $\mathcal{O}(N^{1-\varepsilon})$, where $N := \sum_{i=1}^{t} |V_i|$.

To obtain such an AND-compression algorithm from a hypothetical generalized kernel of $\mathsf{CSP}(\Gamma)$ into a decision problem L, it suffices to do the following:

1. On input a series of instances $(\mathcal{C}_1, V_1), \ldots, (\mathcal{C}_t, V_t)$ of $\mathsf{CSP}(\Gamma)$, form a new instance $(\mathcal{C}^* := \bigcup_{i=1}^{t} \mathcal{C}_i, V^* := \bigcup_{i=1}^{t} V_i)$ of $\mathsf{CSP}(\Gamma)$. Hence we take the disjoint union of the sets of variables and the sets of constraints, and it follows that the new instance has answer YES if and only if all the inputs $(\mathcal{C}_i, V_i)$ have answer YES.
2. Run the hypothetical generalized kernel on $(\mathcal{C}^*, V^*)$, which has $|V^*| = N$ variables and is therefore reduced to an equivalent instance x^* of L with bitsize $\mathcal{O}(N^{1-\varepsilon})$.

If we apply this AND-compression scheme to a sequence of $t_1(m) := m^\alpha$ instances of m bits each (which therefore have at most m variables each), the resulting output has $\mathcal{O}(|V^*|^{1-\varepsilon}) = \mathcal{O}((m \cdot m^\alpha)^{1-\varepsilon}) = \mathcal{O}(m^{(1+\alpha)(1-\varepsilon)})$ bits. By picking α large enough that it satisfies $(1+\alpha)(1-\varepsilon) \leq \alpha$, we therefore compress a sequence of $t_1(m)$ instances of bitsize m into an equivalent instance of size at most $t_2(m) \leq \mathcal{O}(m^{(1+\alpha)(1-\varepsilon)}) \leq C \cdot t_1(m)$, for some suitable constant C. Drucker [9, Theorem 5.4] has shown that an error-free deterministic AND-compression algorithm with these parameters for an NP-complete problem into a fixed decision problem L, implies $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. Hence the lower bound for $k = 1$ follows since $\mathsf{CSP}(\Gamma)$ is NP-complete.

($k \geq 2$) For $k \geq 2$, we prove the lower bound using a linear-parameter transformation (recall Definition 2.1). Let Δ be the set of k-ary relations given by $\Delta := \{\{0,1\}^k \setminus \{u\} \mid u \in \{0,1\}^k\}$. In particular, note that Δ contains the k-OR relation. Since R cone-defines k-OR, it is easy to see that by variable negations, R cone-defines all relations in Δ. Thereby, it follows from Proposition 2.9 that there is a linear-parameter transformation from $\mathsf{CSP}(\Gamma^* \cup \Delta)$ to $\mathsf{CSP}(\Gamma)$. Thus, to prove the lower bound for $\mathsf{CSP}(\Gamma)$, it suffices to prove the desired lower bound for $\mathsf{CSP}(\Gamma^* \cup \Delta)$.

($k = 2$) If $k = 2$, we do a linear-parameter transformation from VERTEX COVER to $\mathsf{CSP}(\Gamma^* \cup \Delta)$. Since it is known that VERTEX COVER parameterized by the number of vertices n has no generalized kernel of size $\mathcal{O}(n^{2-\varepsilon})$ for any $\varepsilon > 0$, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$ [7], the result follows.

Suppose we are given a graph $G = (V, E)$ and integer k for VERTEX COVER. We create an instance $(\mathcal{C}, V')$ of $\mathsf{CSP}(\Gamma^*)$ as follows. We introduce a new variable x_v for each $v \in V$. For each edge $\{u, v\} \in E$, we add the constraint 2-OR(x_u, x_v) to $\mathcal{C}$.

At this point, any vertex cover in G corresponds to a satisfying assignment, and vice versa. It remains to ensure that the size is bounded by k. Let $H_{n,k}$ be the n-ary relation given by $H_{n,k} = \{(x_1, \ldots, x_n) \mid x_i \in \{0,1\}$ for all $i \in [n]$ and $\sum_{i \in [n]} x_i = k\}$. By Proposition A.3, we obtain that Γ^* pp-defines all boolean relations. It follows from [17, Lemma 17] that Γ^* pp-defines $H_{n,k}$ using $\mathcal{O}(n+k)$ constraints and $\mathcal{O}(n+k)$ existentially quantified variables. We add the constraints from this pp-definition to $\mathcal{C}$, and remove the existential quantification. This concludes the construction of $\mathcal{C}$. It is easy to see that $\mathcal{C}$ has a satisfying assignment if and only if G has a vertex cover of size k. Furthermore, we used $\mathcal{O}(n)$ variables and thereby this is a linear-parameter transformation from VERTEX COVER to $\mathsf{CSP}(\Gamma^* \cup \Delta)$.

($k \geq 3$) In this case there is a trivial linear-parameter transformation from $\mathsf{CSP}(\Delta)$ to $\mathsf{CSP}(\Gamma^* \cup \Delta)$. It is easy to verify that $\mathsf{CSP}(\Delta)$ is equivalent to k-CNF-SAT. The result now follows from the fact that for $k \geq 3$, k-CNF-SAT has no kernel of size $\mathcal{O}(n^{k-\varepsilon})$ for any $\varepsilon > 0$, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$ [7]. ◀

Proof of Lemma 3.6. Let $u = (u_1, \ldots, u_k)$ be the unique k-tuple not contained in R. Define the tuple $(y_1, \ldots, y_k)$ as follows. Let $y_i := x_i$ if $u_i = 0$, and let $y_i := \neg x_i$ otherwise. Clearly, this satisfies the first two conditions of cone-definability. It remains to prove the last condition. Let $f\colon \{x_1, \ldots, x_m\} \to \{0,1\}$. Suppose $(f(x_1), \ldots, f(x_k)) \in k$-OR. We show $(\hat{f}(y_1), \ldots, \hat{f}(y_k)) \in R$. Since $(f(x_1), \ldots, f(x_k)) \in k$-OR, there exists at least one $i \in [k]$ such that $f(x_i) \neq 0$. Thereby, $\hat{f}(y_i) \neq u_i$ and thus $(\hat{f}(y_1), \ldots, \hat{f}(y_k)) \neq u$, implying $(\hat{f}(y_1), \ldots, \hat{f}(y_k)) \in R$.

Suppose $(f(x_1), \ldots, f(x_k)) \notin k$-OR, implying $f(x_i) = 0$ for all $i \in [k]$. But this implies $\hat{f}(y_i) = u_i$ for all $i \in [k]$ and thus $(\hat{f}(y_1), \ldots, \hat{f}(y_k)) = u \notin R$. ◀

A.3 Proofs omitted from Section 4

To give the proofs that were omitted from Section 4, we need the following additional definitions.

▶ **Definition A.8.** We say an $m \times n$ matrix A is a *diagonal matrix*, if all entries $a_{i,j}$ with $i \neq j$ are zero. Thus, all non-zero elements occur on the diagonal.

Note that by the above definition of diagonal matrices, a matrix can be diagonal even if it is not a square matrix.

We denote the greatest common divisor of two integers x and y as $\gcd(x, y)$. Recall that by Bézout's lemma, if $\gcd(x, y) = z$ there exist integers a and b such that $ax + by = z$. We will use $x \mid y$ to indicate that x divides y (over the integers) and $x \nmid y$ to indicate that it does not.

Proof of Lemma 4.2. Suppose $u \notin \text{span}_{\mathbb{Z}}(\{s_1, \ldots, s_m\})$, thus u cannot be be written as a linear combination of the rows of S over $\mathbb{Z}$. This implies that the system $yS = u$ has no solutions for y over $\mathbb{Z}$. We will show that there exists a prime power q, such that $yS \equiv_q u$ has no solutions over $\mathbb{Z}/q\mathbb{Z}$ and thus $u \notin \text{span}_q(\{s_1, \ldots, s_m\})$.

There exist an $m \times m$ matrix M and an $n \times n$ matrix N over $\mathbb{Z}$, such that M and N are invertible over $\mathbb{Z}$ and furthermore $S' := MSN$ is in Smith Normal Form (c.f. [10, Theorem 368]). In particular, this implies that all non-zero elements of S' are on the diagonal. Define $u' := uN$.

▶ **Claim A.9.** *$yS = u$ is solvable for y over $\mathbb{Z}$, if and only if $y'S' = u'$ is solvable for y' over $\mathbb{Z}$.*

Proof. Let y be such that $yS = u$. Define $y' := yM^{-1}$. We verify that $y'S' = u'$ as follows.

$$y'S' = y'MSN = yM^{-1}MSN = ySN = uN = u'.$$

For the other direction, consider y' such that $y'S' = u'$. In this case one can verify that $y := y'M$ solves $yS = u$, as

$$yS = y'MS = y'MSNN^{-1} = y'S'N^{-1} = u'N^{-1} = uNN^{-1} = u. \quad \lrcorner$$

It remains to show that if $y'S' = u'$ is unsolvable over $\mathbb{Z}$, then there exists a prime power q such that $y'S' \equiv_q u'$ is unsolvable. By Claim A.9, this implies the lemma statement.

Suppose $y'S' = u'$ has no solutions. Since all non-zero elements of S' are on the diagonal, this implies that either there exists $i \in [n]$, such that u'_i is not divisible by $s'_{i,i}$, or $s'_{i,i}$ is zero while $u'_i \neq 0$. We finish the proof by a case distinction.

- Suppose there exists $i \in [n]$ such that $s'_{i,i} = 0$, while $u'_i \neq 0$. Choose a prime power q such that $q \nmid u'_i$. It is easy to see that thereby, $u'_i \not\equiv_q 0$. Since $s'_{i,i} \equiv_q 0$ holds trivially in this case, the system has $y'S' \equiv_q u'$ no solution.
- Otherwise, there exists $i \in [n]$ such that $s'_{i,i} \nmid u'_i$. Choose a prime power q such that $q \mid u'_i$ and $q \nmid s'_{i,i}$. Such a prime power can be chosen by letting $q := p^\ell$ for a prime p that occurs ℓ times in the prime factorization of u'_i, but less often in the prime factorization of $s'_{i,i}$. Thereby, $u'_i \not\equiv_q 0$, while $s'_{i,i} \equiv_q 0$. It again follows that the system $y'S' \equiv_q u'$ has no solutions. ◀

Proof of Lemma 4.3. Let A' be the $(m-1) \times n$ matrix consisting of the first $m-1$ rows of A. Find the Smith normal form [10] of A' over $\mathbb{Z}$, thus there exist an $(m-1) \times (m-1)$ matrix M' and an $n \times n$ matrix N, such that $S' := M'A'N$ is in Smith Normal Form and M' and N are invertible over $\mathbb{Z}$.

We show that similar properties hold over $\mathbb{Z}/q\mathbb{Z}$. Let $(M')^{-1}$, N^{-1} be the inverses of M' and N over $\mathbb{Z}$, it is easy to verify that $NN^{-1} = I \equiv_q I$ and $M'(M')^{-1} = I \equiv_q I$, such that M' and N are still invertible over $\mathbb{Z}/q\mathbb{Z}$. Furthermore, S' (mod q) remains a diagonal matrix.

Define M to be the following $m \times m$ matrix

$$M := \left(\begin{array}{c|c} M' & \mathbf{0} \\ \hline \mathbf{0} & 1 \end{array}\right),$$

then M has an inverse over $\mathbb{Z}/q\mathbb{Z}$ that is given by the following matrix

$$M^{-1} \equiv_q \left(\begin{array}{c|c} (M')^{-1} & \mathbf{0} \\ \hline \mathbf{0} & 1 \end{array}\right).$$

Define $S := MAN$ and verify that

$$S := MAN \equiv_q \left(\begin{array}{c} S' \\ \hline a_m N \end{array}\right), \tag{3}$$

meaning that the first $m-1$ rows of S are equal to the first $m-1$ rows of S', and the last row of S is given by the row vector $a_m N$.

The following two claims will be used to show that proving the lemma statement for matrix S, will give the desired result for A.

▶ **Claim A.10.** *Let $b := (0, \ldots, 0, c)$ for some constant c. The system $Sx' \equiv_q b$ has a solution, if and only if the system $Ax \equiv_q b$ has a solution.*

Proof. Let x be a solution for $Ax \equiv_q b$. Define $x' := N^{-1}x$. Then $MANx' \equiv_q MAx \equiv_q Mb$. Observe that by the definitions of M and b, $Mb \equiv_q b$, which concludes this direction of the proof.

For the other direction, let x' be a solution for $MANx' \equiv_q b$. Define $x := Nx'$. Then $M^{-1}MANx' \equiv_q M^{-1}b$ and thus $ANx' \equiv_q M^{-1}b$ and thereby $Ax \equiv_q M^{-1}b$. By the definition of M^{-1} and b, we again have $M^{-1}b \equiv_q b$. ⌟

▶ **Claim A.11.** *$s_m \in span_q(\{s_1, \ldots, s_{m-1}\})$ if and only if $a_m \in span_q(\{a_1, \ldots, a_{m-1}\})$.*

Proof. Suppose $s_m \in \text{span}_q(\{s_1, \ldots, s_{m-1}\})$. This implies that there exist $\alpha_1, \ldots, \alpha_{m-1}$ such that $\sum_{i \in [m-1]} \alpha_i s_i \equiv_q s_m \equiv_q a_m N$. Thus, $\sum_{i \in [m-1]} \alpha_i s'_i \equiv_q a_m N$, and for $\alpha = (\alpha_1, \ldots, \alpha_{m-1})$ we therefore have $\alpha S' \equiv_q a_m N$, implying $(\alpha M')A'N \equiv_q a_m N$. Since N is invertible, it follows that

$$(\alpha M')A' \equiv_q a_m$$

and thus $a_m \in \text{span}_q(\{a_1, \ldots, a_{m-1}\})$.

For the other direction, suppose $a_m \in \text{span}_q(\{a_1, \ldots, a_{m-1}\})$. Thus, there exists $\alpha \equiv_q (\alpha_1, \ldots, \alpha_{m-1})$ such that $\alpha A' \equiv_q a_m$. Let $\alpha' := \alpha(M')^{-1}$. Then

$$\alpha' S' \equiv_q \alpha' M' A' N \equiv_q \alpha A' N \equiv_q a_m N \equiv_q s_m,$$

and now it follows from the definition of S given in (3) that $s_m \in \text{span}_q(\{s_1, \ldots, s_{m-1}\})$. ⌟

It follows from Claims A.10 and A.11, that it suffices to show that if $Sx = (0, \ldots, 0, c)^T$ has no solutions for any $c \not\equiv_q 0$, then $s_m \in \text{span}_q(\{s_1, \ldots, s_{m-1}\})$. So suppose $s_m \notin \text{span}_q(\{s_1, \ldots, s_{m-1}\})$, we show that the system has a solution for some non-zero c. Observe that since S' (the first $m-1$ rows of S) is a diagonal matrix, there must exist $i \in [m-1]$ such that there is no α_i such that $s_{i,i} \cdot \alpha_i \equiv_q s_{m,i}$. Otherwise, it is easy to see that $\sum_{i \in [m-1]} \alpha_i s_i \equiv_q s_m$, contradicting that $s_m \notin \text{span}_q(\{s_1, \ldots, s_{m-1}\})$. We now do a case distinction.

Suppose there exists $i \in [m-1]$ such that $s_{i,i} \equiv_q 0$, while $s_{m,i} \not\equiv_q 0$. Let $x = (0, \ldots, 0, 1, 0, \ldots, 0)$ be the vector with 1 in the i'th position and zeros in all other positions. It is easy to verify that $Sx \equiv_q (0, \ldots, 0, s_{m,i})^T$ and thereby the system $Sx \equiv_q b$ has a solution for $c = s_{m,i}$.

Otherwise, choose i such that there exists no integer α_i such that $s_{i,i} \cdot \alpha_i \equiv_q s_{m,i}$ and $s_{i,i} \not\equiv_q 0$. It is given that q is a prime power, let $q = p^k$ for prime p. Let $0 \leq \ell < k$ be the largest integer such that $p^\ell \mid s_{i,i}$ over the integers. We consider the following two cases.

- Suppose $p^\ell \mid s_{m,i}$. Let c such that $s_{i,i} \equiv_q cp^\ell$ and choose d such that $s_{m,i} \equiv_q d \cdot p^\ell$. Note that $\gcd(c, q) = 1$ and it follows from Bézout's lemma that c has an inverse c^{-1} such that $cc^{-1} \equiv_q 1$. Then

$$s_{i,m} \equiv_q (d \cdot c^{-1}) s_{i,i},$$

  which is a contradiction with the assumption that no integer α_i exists such that $s_{i,i} \cdot \alpha_i \equiv_q s_{i,m}$.
- Suppose $p^\ell \nmid s_{m,i}$. Define $x := (0, \ldots, 0, p^{k-\ell}, 0, \ldots, 0)^{\mathrm{T}}$ as the vector with $p^{k-\ell}$ in position i. Then

$$Sx \equiv_q (0, \ldots, 0, p^{k-\ell} \cdot s_{i,i}, 0, \ldots, 0, p^{k-\ell} \cdot s_{m,i})^{\mathrm{T}}.$$

  Since $p^\ell \mid s_{i,i}$ it follows that $p^{k-\ell} \cdot s_{i,i} \equiv_q 0$. Furthermore, since $p^\ell \nmid s_{m,i}$, it follows that $p^{k-\ell} \cdot s_{m,i} \not\equiv_q 0$, and thereby the system $Sx \equiv_q b$ has a solution for $b := (0, \ldots, 0, p^{k-\ell} s_{m,i})$.

◀

Proof of Theorem 4.4. Suppose R is not balanced, let f be a balanced operation that does not preserve R. Let the coefficients of f be $\alpha_1, \ldots, \alpha_m$. Then, there exist $r_1, \ldots, r_m \in R$ and $u \in \{0,1\}^k \setminus R$ such that $\sum_{i \in [m]} \alpha_i r_i = u$. Now suppose there exists a linear polynomial p_u (and optionally, a prime power q_u) such that p_u captures u (over $\mathbb{Z}$ or $\mathbb{Z}/q_u\mathbb{Z}$ respectively). Let $r_i = (r_{i,1}, \ldots, r_{i,k})$ for $i \in [m]$. By definition, $p_u(r_{i,1}, \ldots, r_{i,k}) \equiv 0$ (over $\mathbb{Z}$ or $\mathbb{Z}/q_u\mathbb{Z}$ respectively). Since p_u is a linear polynomial, suppose $p_u(x_1, \ldots, x_k)$ is given by $p_u(x_1, \ldots, x_k) \equiv \beta_0 + \sum_{i \in [m]} \beta_i x_i$ for integer coefficients β_i. Let $u = (u_1, \ldots, u_k)$. But then,

$$\begin{aligned} p_u(u_1, \ldots, u_k) &\equiv \beta_0 + \sum_{i \in [m]} \beta_i u_i \\ &\equiv \beta_0 + \sum_{i \in [m]} \beta_i \Big(\sum_{j \in [m]} \alpha_j r_{j,i} \Big) \\ &\equiv \beta_0 + \sum_{i \in [m]} \sum_{j \in [m]} \alpha_j \beta_i r_{j,i} \end{aligned}$$

(Since $\sum_{i\in[m]} \alpha_i \equiv 1$)

$$\equiv \sum_{j\in[m]} \alpha_j\beta_0 + \sum_{j\in[m]}\sum_{i\in[m]} \alpha_j\beta_i r_{j,i}$$

$$\equiv \sum_{j\in[m]} \alpha_j\beta_0 + \alpha_j \sum_{i\in[m]} \beta_i r_{j,i}$$

$$\equiv \sum_{j\in[m]} p_u(r_{j,1},\ldots,r_{j,k}) \equiv 0,$$

But this contradicts the fact that $p_u(u_1,\ldots,u_k) \not\equiv 0$. Thereby, there exists no linear polynomial that captures u with respect to R. ◀

A.4 Proofs omitted from Section 5

Proof of Lemma 5.3. We first show the result when $b \leq a$, $b \leq c$, and $b \leq d$. In this case, we use the following tuple to express $x_1 \vee x_2$.

$$(\underbrace{\neg x_1,\ldots,\neg x_1}_{(a-b)\text{ copies}},\underbrace{\neg x_2,\ldots,\neg x_2}_{(c-b)\text{ copies}},\underbrace{1,\ldots,1}_{b\text{ copies}}).$$

Let $f : \{x_1, x_2\} \to \{0, 1\}$, then $(\neg f(x_1),\ldots,\neg f(x_1),\neg f(x_2),\ldots,\neg f(x_2),1,\ldots,1)$ has weight $(a-b)(1-f(x_1)) + (c-b)(1-f(x_2)) + b$. It is easy to verify that for $f(x_1) = f(x_2) = 0$, this implies the tuple has weight $a+c-b = d \notin S$ and thus the tuple is not in R. Otherwise, the weight is either a, b, or c. In these cases the tuple is contained in R, as the weight is contained in S.

Note that the above case applies when b is the smallest of all four integers. We now consider the remaining cases. Suppose $a \leq b$, $a \leq c$, and $a \leq d$ (the case where c is smallest is symmetric). In this case, use the tuple

$$(\underbrace{\neg x_1,\ldots,\neg x_1}_{(d-a)\text{ copies}},\underbrace{x_2,\ldots,x_2}_{(b-a)\text{ copies}},\underbrace{1,\ldots,1}_{a\text{ copies}}).$$

Consider an assignment f satisfying $x_1 \vee x_2$, verify that the weight of the above tuple under this assignment lies in $\{a, b, c\}$, and thus the tuple is contained in R. Assigning 0 to both x_1 and x_2 gives weight d, such that the tuple is not in R.

Otherwise, we have $d \leq a$, $d \leq b$, and $d \leq c$ and use the tuple

$$(\underbrace{x_1,\ldots,x_1}_{(a-d)\text{ copies}},\underbrace{x_2,\ldots,x_2}_{(c-d)\text{ copies}},\underbrace{1,\ldots,1}_{d\text{ copies}}).$$

It is again easy to verify that any assignment to x_1 and x_2 satisfies this tuple if and only if it satisfies $(x_1 \vee x_2)$, using the fact that $a - d + c = b \in S$. ◀

Proof of Claim 5.5. If there exist distinct $i, j, \ell \in [m]$ with i, j odd and ℓ even, such that $s_i \geq s_\ell \geq s_j$, then these i, j, ℓ satisfy the claim statement. Suppose these do not exist, we consider two options.

- Suppose $s_i \geq s_\ell$ for all $i, \ell \in [m]$ with i odd and ℓ even. It is easy to see that thereby, for any i, j, ℓ with i, j odd and ℓ even it holds that $s_i - s_\ell + s_j \geq 0$. Furthermore, $s_i - s_\ell + s_j \leq s_1 - s_2 + s_3 - s_4 \cdots + s_m = t$ and $t \leq k$ since $t \in U$. Thus, any distinct $i, j, \ell \in [m]$ with i, j odd and ℓ even satisfy the statement.

- Otherwise, $s_i \leq s_\ell$ for all $i, \ell \in [m]$ with i odd and ℓ even. It follows that for any i, j, ℓ with i, j odd and ℓ even $s_i - s_\ell + s_j \leq k$, as $s_i - s_\ell \leq 0$ and $s_j \leq k$. Furthermore, $s_i - s_\ell + s_j \geq s_1 - s_2 + s_3 - s_4 \cdots + s_m = t$ and $t \geq 0$ by definition. Thus, any distinct $i, j, \ell \in [m]$ with i, j odd and ℓ even satisfy the statement. ⌟

A.5 Proofs omitted from Section 6

To state the proofs that were omitted from this section, we will first give a number of relevant observations and propositions.

▶ **Observation A.12.** *If a relation T of arity m is cone-definable from a relation U of arity n, then $m \leq n$.*

▶ **Observation A.13.** *Suppose a relation T is cone-definable from a relation U, and that g is a partial operation that is idempotent and self-dual. If g preserves U, then g preserves T.*

▶ **Observation A.14.** *(transitivity of cone-definability) Suppose that T_1, T_2, T_3 are relations such that T_2 is cone-definable from T_1, and T_3 is cone-definable from T_2. Then T_3 is cone-definable from T_1.*

▶ **Definition A.15.** Let us say that two Boolean relations T, U are *cone-interdefinable* if each is cone-definable from the other.

The following two propositions are consequences of Observations A.12 and A.13. We will tacitly use them in the sequel. Morally, they show that the properties of relations that we are interested in are invariant under cone-interdefinability.

▶ **Proposition A.16.** *If relations T, U are cone-interdefinable, then they have the same arity.*

▶ **Proposition A.17.** *Suppose that T and U are relations that are cone-interdefinable, and that g is a partial operation that is idempotent and self-dual. Then, g preserves T if and only if g preserves U.*

Proof of Theorem 6.2. Let $R \subseteq \{0,1\}^2$ be a relation. We prove the two directions separately.

($\Leftarrow$) Proof by contraposition. Suppose that R is cone-interdefinable with 2-OR. Then in particular, R cone-defines the 2-OR relation. Let (y_1, y_2) be a tuple witnessing cone-definability as in Definition 2.7. Since 2-OR is symmetric in its two arguments, we may assume without loss of generality that y_i is either x_i or $\neg x_i$ for $i \in [2]$. Define $g\colon \{0,1\}^2 \to \{0,1\}^2$ by letting $g(i_1, i_2) := (\hat{i}_1, \hat{i}_2)$ where $\hat{i}_\ell = i_\ell$ if $y_i = x_i$ and $\hat{i}_\ell = 1 - i_\ell$ if $y_i = \neg x_i$. By definition of cone-definability we then have $g(1,0), g(0,1), g(1,1) \in R$ while $g(0,0) \notin R$. But $g(1,0) - g(1,1) + g(0,1) = g(0,0)$, showing that R is not preserved by all alternating operations and therefore is not balanced.

($\Rightarrow$) We again use contraposition. Suppose R is not balanced; we will prove R is cone-interdefinable with 2-OR. Let $f\colon \{0,1\}^k \to \{0,1\}$ be a balanced partial Boolean operation *of minimum arity* that does not preserve R. Let $\alpha_1, \ldots, \alpha_k \in \mathbb{Z}$ be the coefficients of f, as in Definition 2.2. Let $s^1, \ldots, s^k \in R$ such that $f(s^1, \ldots, s^k) = u \in \{0,1\}^2 \setminus R$ witnesses that f does not preserve R. By Definition 2.2 we have $u = \sum_{i=1}^k \alpha_i s^i$ and $\sum_{i=1}^k \alpha_i = 1$. This shows that if $\alpha_i = 0$ for some coordinate i, then that position does not influence the value of f, implying the existence of a smaller-arity balanced relation that does not preserve T. Hence our choice of f as a minimum-arity operation ensures that α_i is nonzero for all $i \in [k]$.

▶ **Claim A.18.** *The tuples $s^1, \ldots, s^k$ are all distinct.*

Proof. Suppose that $s^i = s^j$ for some distinct $i, j \in [k]$, and assume without loss of generality that $i = k-1$ and $j = k$. But then the balanced operation f' of arity $k-1$ defined by the coefficients $(\alpha_1, \ldots, \alpha_{k-2}, \alpha_{k-1} + \alpha_k)$ does not preserve T since $f'(s^1, \ldots, s^{k-1}) = f(s^1, \ldots, s^k) = u \notin R$, contradicting that f is a minimum-arity balanced operation that does not preserve R. ⌟

▶ **Claim A.19.** *The arity k of operation f is 3.*

Proof. Since $s^1, \ldots, s^k \in R \subseteq \{0,1\}^2$ are all distinct, while $u \in \{0,1\}^2 \setminus R$, we have $k \leq 3$. We cannot have $k = 1$ since that would imply $f(s^1) = s^1 \in R$ and $f(s^1, \ldots, s^k) = f(s^1) = u \notin R$. It remains to show that $k \neq 2$. So assume for a contradiction that $k = 2$. Since s^1 and s^2 are distinct, there is a position $\ell \in [2]$ such that $s^1_\ell \neq s^2_\ell$. Assume without loss of generality that $s^1_\ell = 1$ while $s^2_\ell = 0$. Since $f(s^1, \ldots, s^k) = f(s^1, s^2) = \alpha_1 s^1 + \alpha_2 s^2 = u \in \{0,1\}^2 \setminus R$, we find $u_\ell = \alpha_1 s^1_\ell + \alpha_2 s^2_\ell = \alpha_1 \cdot 1 + \alpha_2 \cdot 0 \in \{0,1\}$. Since α_1 is a nonzero integer, we must have $\alpha_1 = 1$. But since $\alpha_1 + \alpha_2 = 1$ by definition of a balanced operation, this implies $\alpha_2 = 0$, contradicting that f is a minimum-arity balanced operation that does not preserve R. ⌟

The previous two claims show that there are at least three distinct tuples in $R \subseteq \{0,1\}^2$. Since $u \in \{0,1\}^2 \setminus R$ it follows that $|R| = 3$. Hence R and 2-OR are both Boolean relations of arity two that each have three tuples. To cone-define one from the other, one may easily verify that it suffices to use the tuple (y_1, y_2), where $y_i = x_i$ if $u_i = 0$ and $y_i = \neg x_i$ otherwise. ◀

In order to prove Theorem 6.3, we first present some additional lemmas and definitions.

▶ **Lemma A.20.** *Suppose that $U \subseteq \{0,1\}^n$ is a Boolean relation, and that there exist an integer a and a natural number $m > 1$ such that, for each $u \in U$, it holds that*

$$\text{weight}(u) \equiv a \mod m.$$

Then, if w is a witness for U, it holds that $\text{weight}(w) \equiv a \mod m$.

Let $U \subseteq \{0,1\}^n$ be a relation. We say that $w \in \{0,1\}^n$ is a *witness* for U if $w \notin R$, and there exists a balanced operation $f : \{0,1\}^k \to \{0,1\}$ and tuples $t^1, \ldots, t^k \in U$ such that $w = f(t^1, \ldots, t^k)$. Observe that U is not balanced if and only if there exists a witness for U. We will view tuples of arity n as maps $f : [n] \to \{0,1\}$, via the natural correspondence where such a map f represents the tuple $(f(1), \ldots, f(n))$. We freely interchange between these two representations of tuples. We say that $f : S \to \{0,1\}$ is a *no-good* of U when: $S \subseteq [n]$; each extension $g : [n] \to \{0,1\}$ of f is not an element of U; and there exists an extension $h : [n] \to \{0,1\}$ of f that is a witness for U. We say that $f : S \to \{0,1\}$ is a *min-no-good* of f is a no-good, but no proper restriction of f is a no-good. Observe that the following are equivalent, for a relation: the relation is not balanced; it has a witness; it has a no-good; it has a min-no-good.

When $U \subseteq \{0,1\}^n$ is a relation and $S \subseteq [n]$, let $s_1 < \cdots < s_m$ denote the elements of S; then, we use $U \restriction S$ to denote the relation $\{(h(s_1), \ldots, h(s_m)) \mid h \in U\}$.

▶ **Proposition A.21.** *Let $U \subseteq \{0,1\}^n$ be a relation, let $S \subseteq [n]$, and suppose that $f : S \to \{0,1\}$ is a min-no-good of U. Then f is a min-no-good of $U \restriction S$.*

Proof. Observe that f is not in $U \restriction S$; since f has an extension that is a witness for U, it follows that f is a witness for $U \restriction S$. Thus, f is a no-good of $U \restriction S$. In order to obtain that f is a min-no-good of $U \restriction S$, it suffices to establish that, for any restriction $f^- : S^- \to \{0,1\}$ of f, it holds that f^- is a no-good of U if and only if f^- is a no-good of $U \restriction S$. This

follows from what we have established concerning f and the following fact: all extensions $h : S \to \{0,1\}$ of f^- are not in $U \upharpoonright S$ if and only if all extensions $h' : [n] \to \{0,1\}$ of f^- are not in U. ◀

Proof. Since w is a witness for U, there exist tuples $t^1 = (t^1_1, \ldots, t^1_n)$, …, $t^k = (t^k_1, \ldots, t^k_n)$ and a balanced operation $f : \{0,1\}^k \to \{0,1\}$ such that $f(t^1, \ldots, t^k) = w$. Let $\alpha_1, \ldots, \alpha_k$ be the coefficients of f. From $f(t^1, \ldots, t^k) = w$, we obtain that $\alpha_1 \text{weight}(t^1) + \cdots + \alpha_k \text{weight}(t^k) = \text{weight}(w)$. As

$$\alpha_1 \text{weight}(t^1) + \cdots + \alpha_k \text{weight}(t^k) \equiv \alpha_1 a + \cdots + \alpha_k a \equiv a \mod m,$$

the result follows. ◀

Proof of Theorem 6.3. Let $f : S \to \{0,1\}$ be a min-no-good of U.

It cannot hold that $|S| = 0$, since then U would be empty and hence preserved by all balanced operations. It also cannot hold that $|S| = 1$, since then f would be a min-no-good of $U \upharpoonright S$ (by Proposition A.21), which is not possible since $U \upharpoonright S$ would have arity 1 and hence would be preserved by all balanced operations (by Observation 6.1).

For the remaining cases, by replacing U with a relation that is interdefinable with it, we may assume that $f : S \to \{0,1\}$ maps each $s \in S$ to 0.

Suppose that $|S| = 2$, and assume for the sake of notation that $S = \{1,2\}$ (this can be obtained by replacing U with a relation that is interdefinable with it). By Proposition A.21, f is a min-no-good of $U \upharpoonright S$. By Theorem 6.2, we obtain that $U \upharpoonright S$ contains all tuples other than f, that is, we have $\{(0,1),(1,0),(1,1)\} = U \upharpoonright S$. It follows that there exists a *realization*, where we define a *realization* to be a tuple $(a_1, a_2, a_3) \in \{0,1\}^3$ such that $(0,1,a_1),(1,0,a_2),(1,1,a_3) \in U$. Let us refer to $(0,0,1)$ and $(1,1,0)$ as *bad* tuples, and to all other arity 3 tuples as *good* tuples. We claim that there exists a realization that is a good tuple; this suffices, for then the 2-OR relation is cone-definable from U via a tuple of the form (x_1, x_2, y) where $y \in \{0, 1, x_1, x_2, \neg x_1, \neg x_2\}$. In particular, let us state explicitly how y can be chosen as a function of a realization that is a good tuple: chose $y = 0$ for $(0,0,0)$; $y = 1$ for $(1,1,1)$; $y = x_1$ for $(0,1,1)$; $y = x_2$ for $(1,0,1)$; $y = \neg x_1$ for $(1,0,0)$; and, $y = \neg x_2$ for $(0,1,0)$. We prove the claim by contradiction. If there exists no realization that is a good tuple, every realization is a bad tuple; moreover, there is a unique realization, for if there were more than one, there would exist a realization that was a good tuple. We may assume (up to interdefinability of U) that the unique realization is $(1,1,0)$. Then, U is the relation $\{(0,1,1),(1,0,1),(1,1,0)\}$ containing exactly the weight 2 tuples; applying Lemma A.20 to U with $a = 2$ and $m = 3$, we obtain that for any witness w for U, it holds that $\text{weight}(w) \equiv 2 \mod 3$. This implies that f has no extension w' that is a witness, since any such extension must have $\text{weight}(w')$ equal to 0 or 1; we have thus contradicted that f is a no-good of U.

Suppose that $|S| = 3$. Since f is both a min-no-good and a witness, it follows that each of the weight 1 tuples $(1,0,0),(0,1,0),(0,0,1)$ is contained in U. We claim that U contains a weight 2 tuple; if not, then U would contain only weight 1 and weight 3 tuples, and by invoking Lemma A.20 with $a = 1$ and $m = 2$, we would obtain that $\text{weight}(f) \equiv 1 \mod 2$, a contradiction. Assume for the sake of notation that U contains the weight 2 tuple $(0,1,1)$. Then U cone-defines the 2-OR relation via the tuple $(0, x_1, x_2)$, since $(0,0,0) \notin R$ and $(0,1,0),(0,0,1),(0,1,1) \in R$. ◀