



Best-Case and Worst-Case Sparsifiability of Boolean CSPs

Hubie Chen¹ · Bart M. P. Jansen² · Astrid Pieterse³ 

Received: 1 December 2018 / Accepted: 9 December 2019 / Published online: 10 January 2020
© Springer Science+Business Media, LLC, part of Springer Nature 2020

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—and identify constraint languages giving the best-possible and worst-possible behavior for worst-case sparsifiability. Two algorithmic results are presented. The first result essentially shows that for any arity k , the only constraint type for which no nontrivial sparsification is possible has exactly one falsifying assignment, and corresponds to logical OR (up to negations). Our second result concerns linear sparsification, that is, a reduction to an equivalent instance with $O(n)$ 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 ring, 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.

Keywords Constraint satisfaction problems · Kernelization · Sparsification · Lower bounds

An extended abstract of this work, with the same title, appeared in the Proceedings of the 13th International Symposium on Parameterized and Exact Computation (IPEC 2018). This work was supported by NWO Gravitation grant “Networks”. This work was done while the third author was a PhD student at Eindhoven University of Technology.

Extended author information available on the last page of the article

1 Introduction

1.1 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 (finite) *constraint language*, which is a set of (finite) 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 [25], which classifies each CSP over the Boolean domain as polynomial-time solvable or NP-complete.

Continuing a recent line of investigation [12, 14, 19], 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, 22], 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 $\text{NP} \not\subseteq \text{coNP/poly}$ (which we tacitly assume throughout this introduction), Dell and van Melkebeek [7] proved a strong lower bound: For any 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 $O(n^{d-\varepsilon})$. In fact, there cannot even be an algorithm that transforms such φ into small equivalent instances ψ of an *arbitrary* decision problem. Since an instance of d -CNF-SAT has at most $2^d n^d \in O(n^d)$ distinct clauses, it can *trivially* be sparsified to $O(n^d)$ clauses by removing duplicates, and can be compressed to size $O(n^d)$ 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, 17] 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 $O(n^{d-1})$ clauses that preserves the answer,

resulting in a compression of bitsize $O(n^{d-1} \log n)$. The first proof of this result was based on an ad-hoc application of a theorem of Lovász [23]. 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 $O(n^d)$. 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 $O(n)$ constraints for each constant d . This prompted a detailed investigation into linear sparsifications for CSPs by Lagerkvist and Wahlström [19], 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.

1.2 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 (Sect. 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 $\text{CSP}(\Gamma)$ 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 constraint 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 (Sect. 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 $s_1, \dots, s_{2n}, s_{2n+1} \in R$ for $n \geq 1$ such that $s_1 - s_2 + s_3 \cdots - s_{2n} + s_{2n+1} = u \in \{0, 1\}^k \setminus R$. (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-vector u that is outside R . For example, the binary OR relation $2\text{-OR} = \{0, 1\}^2 \setminus \{(0, 0)\}$ is *not* balanced, since $(0, 1) - (1, 1) + (1, 0) = (0, 0) \notin 2\text{-OR}$, 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 $O(n)$. Hence when *all* relations in a constraint language Γ are balanced—we call such a constraint language *balanced*—then $\text{CSP}(\Gamma)$ has a sparsification with $O(n)$ 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 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. In Sect. 7 we compare our universal-algebraic approach to the approach proposed by Lagerkvist and Wahlström [19].

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 (Sect. 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 $R_{a \rightarrow b} = \{(0, 0), (0, 1), (1, 1)\}$ corresponding to the truth value of $a \rightarrow b$ is not. We prove that if a symmetric Boolean relation R is not balanced, then it can implement (Definition 2.10) 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 $\text{CSP}(\Gamma)$ is NP-complete, then $\text{CSP}(\Gamma)$ does *not* admit a sparsification of size $O(n^{2-\varepsilon})$ 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 (Sect. 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 $\text{CSP}(\Gamma)$ has a sparsification of size $O(n^k)$, but no sparsification of size $O(n^{k-\varepsilon})$ 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.

2 Preliminaries

For a positive integer n , define $[n] := \{1, 2, \dots, n\}$. We use \mathbb{N} (resp. \mathbb{N}_0) to denote the positive (non-negative) integers. 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 .

2.1 Parameterized Complexity

A parameterized problem \mathcal{Q} is a subset of $\Sigma^* \times \mathbb{N}$, where Σ is a finite alphabet.

Definition 2.1 Let $\mathcal{Q}, \mathcal{Q}' \subseteq \Sigma^* \times \mathbb{N}$ be parameterized problems and let $h: \mathbb{N} \rightarrow \mathbb{N}$ 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:

1. $|x'|$ and k' are bounded by $h(k)$, and
2. $(x', k') \in \mathcal{Q}'$ if and only if $(x, k) \in \mathcal{Q}$.

The algorithm is a *kernel* for \mathcal{Q} if $\mathcal{Q}' = \mathcal{Q}$. It is a *polynomial (generalized) kernel* if $h(k)$ is a polynomial.

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 2.2 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. $k' \in O(k)$.

It is well-known [1,2] that the existence of a 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} .

2.2 Polynomials

A multivariate *polynomial* (over a ring E) is a function $p: E^k \rightarrow E$ of the type

$$p(x_1, \dots, x_k) = \sum_{(d_1, \dots, d_k) \in (\mathbb{N}_0)^k} \alpha_{d_1, \dots, d_k} \prod_{i \in [k]} (x_i)^{d_i}$$

with coefficients $\alpha_{d_1, \dots, d_k} \in E$ that are non-zero for a finite number of choices for $(d_1, \dots, d_k) \in (\mathbb{N}_0)^k$. A term of the type $\prod_{i \in [k]} (x_i)^{d_i}$ is referred to as a *monomial*. The *degree* of a monomial is simply $\sum_{i \in [k]} d_i$. The degree of a polynomial is the maximum of the degrees of all monomials in this polynomial that have a non-zero coefficient. A *linear polynomial* is a polynomial of degree one.

We utilize the following property of alternating sums of linear polynomials.

Proposition 2.3 *Let E be a ring, let $y_1, \dots, y_m \in E^k$ with $y_i = (y_{i,1}, \dots, y_{i,k})$ for all $i \in [m]$, where $m \geq 3$ is odd. Let $x = (x_1, \dots, x_k)$ be such that $x_j = y_{1,j} - y_{2,j} \dots + y_{m,j}$ for all $j \in [k]$. Let p be a linear polynomial over E . Then $p(x_1, \dots, x_k) = p(y_{1,1}, \dots, y_{1,k}) - p(y_{2,1}, \dots, y_{2,k}) \dots + p(y_{m,1}, \dots, y_{m,k})$.*

Proof Let $p(v_1, \dots, v_k) := \alpha_0 + \sum_{i \in [k]} \alpha_i v_i$. We start by proving the proposition for $m = 3$. Then

$$\begin{aligned} p(x_1, \dots, x_k) &= \alpha_0 + \sum_{i \in [k]} \alpha_i (y_{1,i} - y_{2,i} + y_{3,i}) \\ &= \alpha_0 + \sum_{i \in [k]} \alpha_i y_{1,i} - \sum_{i \in [k]} \alpha_i y_{2,i} + \sum_{i \in [k]} \alpha_i y_{3,i} \\ &= \alpha_0 + \sum_{i \in [k]} \alpha_i y_{1,i} - \alpha_0 - \sum_{i \in [k]} \alpha_i y_{2,i} + \alpha_0 + \sum_{i \in [k]} \alpha_i y_{3,i} \\ &= p(y_{1,1}, \dots, y_{1,k}) - p(y_{2,1}, \dots, y_{2,k}) + p(y_{3,1}, \dots, y_{3,k}). \end{aligned}$$

Using a simple induction, the result for $m > 3$ follows. □

2.3 Operations, Relations, and Preservation

An *operation* on a domain D is a mapping $f: D^k \rightarrow D$, 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 operation* on D in the usual way, that is, it is a mapping f' from a subset domain $\text{domain}(f') \subseteq D^k$ to D . The partial operation

is undefined outside its domain. When the domain is $\{0, 1\}$, we speak of a *Boolean (partial) operation*.

We say that a partial Boolean operation f of arity k is *idempotent* if $f(0, \dots, 0) = 0$ and $f(1, \dots, 1) = 1$; and, *self-dual* if for all $(a_1, \dots, a_k) \in \{0, 1\}^k$, when $f(a_1, \dots, a_k)$ is defined, it holds that $f(\neg a_1, \dots, \neg a_k)$ is defined and $f(a_1, \dots, a_k) = \neg f(\neg a_1, \dots, \neg a_k)$.

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

Definition 2.4 For each $k \geq 1$, we use k -OR to denote the relation $\{0, 1\}^k \setminus \{(0, \dots, 0)\}$.

A *constraint language over D* is a set of relations over D ; a *Boolean constraint language* is a constraint language over $\{0, 1\}$. All our algorithmic results will deal with finite constraint languages. For a finite constraint language Γ over a domain D , we define $\text{CSP}(\Gamma)$ as follows.

$\text{CSP}(\Gamma)$

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 of the form $R(x_1, \dots, x_k)$ for $R \in \Gamma$ and $x_1, \dots, x_k \in V$.

Question: Does there exist a *satisfying assignment*, that is, an assignment $f: V \rightarrow D$ such that for each constraint $R(x_1, \dots, x_k) \in \mathcal{C}$ it holds that $(f(x_1), \dots, f(x_k)) \in R$?

When using the framework of parameterized complexity to analyze sparsifiability of $\text{CSP}(\Gamma)$, we will consistently consider its parameterization by the number of variables $n := |V|$.

Let f be a partial operation of arity k on a domain D , and let $T \subseteq D^n$ be a relation. We say that T is *preserved* by f when for any tuples $t_1 = (t_{1,1}, \dots, t_{1,n}), \dots, t_k = (t_{k,1}, \dots, t_{k,n}) \in T$, if all entries of the tuple $(f(t_{1,1}, \dots, t_{k,1}), \dots, f(t_{1,n}, \dots, t_{k,n}))$ are defined, then this tuple is in T . We say that a constraint language Γ is *preserved* by f if each relation in Γ is preserved by f . If T is preserved by a partial operation f , we also say T has f as a *partial polymorphism*. Similarly, if Γ is preserved by f then f is a *partial polymorphism* of Γ .

2.4 Balanced and Alternating Operations

The following definitions capture the key notions for the universal-algebraic approach to sparsification via low-degree polynomials.

Definition 2.5 A partial Boolean operation $f: \{0, 1\}^k \rightarrow \{0, 1\}$ is *balanced* if there exist integer values $\alpha_1, \dots, \alpha_k$, called the *coefficients* of f , such that

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

Definition 2.6 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.

Definition 2.7 Define an *alternating operation* to be a balanced operation $\alpha_k : \{0, 1\}^k \rightarrow \{0, 1\}$ such that k is odd and the coefficients alternate between $+1$ and -1 , so that $\alpha_1 = +1, \alpha_2 = -1, \alpha_3 = +1, \dots, \alpha_k = +1$.

While alternating operations form a restricted type of balanced operations, the following proposition shows that being preserved by all balanced operations is equivalent to being preserved by all alternating operations.

Proposition 2.8 *A Boolean relation R is balanced if and only if for all odd $k \geq 1$, the relation R is preserved by the alternating operation of arity k .*

Proof 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, \dots, t_k in T such that $\alpha_1 t_1 + \dots + \alpha_k t_k$ 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 $t_i + \dots + t_i$ (α_i times); likewise, for each negative α_i , replace $\alpha_i t_i$ in the sum with $-t_i - \dots - t_i$ ($-\alpha_i$ 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 . □

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

Observation 2.9 *Each balanced operation is idempotent and self-dual.*

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

2.5 Cone-Definability

To relate the sparsification complexity of different CSPs to each other, we introduce a new kind of definability. The traditional notion of positive-primitive definability (Definition 2.15) is not suitable for this purpose, since it allows an arbitrary number of existentially quantified new variables. This means that when reducing an instance (V, \mathcal{C}) of $\text{CSP}(\Gamma)$ to an equivalent instance (V', \mathcal{C}') of $\text{CSP}(\Gamma')$ knowing only that each relation $R \in \Gamma$ can be pp-defined from a relation $R' \in \Gamma'$, the number

of variables in V' may be as large as $\Omega(|\mathcal{C}| + |V|)$; the pp-definition may require fresh variables to be introduced for each constraint. For that reason, such a transformation does not transfer lower bounds on sparsification complexity. The notion of quantifier-free pp-expression (cf. [19, Section 2.2]) prevents such a blow-up in the number of variables, and indeed if all relations in Γ can be qfpp-defined from relations in Γ' , this leads to a transformation showing that $\text{CSP}(\Gamma')$ is at least as hard to sparsify as $\text{CSP}(\Gamma)$. We can work with a slightly more permissive notion of definability, though, since the number of variables is allowed to increase by a constant factor. We can therefore permit to introduce variables for each of the two constants, and to introduce a negated version of each variable, leading to the following notion of **cone**-definition.

Definition 2.10 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, \dots, y_n) where:

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

Example 2.11 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.

Lagerkvist and Wahlström introduced the notion of *sign-symmetric* constraint languages [21], which are languages that are closed under variable negation. When working with sign-symmetric intractable Boolean constraint languages, cone-definitions give no additional expressive power over qfpp-definitions; the use of cone-definitions will allow us to deal also with languages that are not sign-symmetric. To give lower bounds on the sparsification complexity of $\text{CSP}(\Gamma)$, it turns out that we are primarily interested in whether or not Γ can define k -OR for some $k \geq 2$. Since this means that only one tuple has to be excluded from the relation, to cone-define k -OR it does not help to take a conjunction of cone-definitions. To keep the notation as light as possible, the notion of cone-definition therefore does not allow a definition of T in terms of a conjunction of terms over U , although one can naturally extend the given notion of cone-definition to allow for conjunction in such a way that Proposition 2.17 below holds.

There is a close relation between cone-definability of the 2-OR relation and having the partial Boolean operation φ_1 as a partial polymorphism, where φ_1 is defined by $\varphi_1(x, x, y) = y$, $\varphi_1(x, y, y) = x$, and $\text{domain}(\varphi_1) = \{(0, 0, 0), (0, 0, 1), (0, 1, 1), (1, 0, 0), (1, 1, 0), (1, 1, 1)\}$. The overall idea of the next proposition was first discovered in [20, Section 5.2], we state it here in terms of cone-definitions as we will use this result throughout the paper.

Proposition 2.12 *Let Γ be a Boolean constraint language. Then Γ is not preserved by φ_1 if and only if there exists $R \in \Gamma$ such that R cone-defines 2-OR.*

Proof (\Rightarrow) Suppose $R \in \Gamma$ is not preserved by φ_1 . We show that R cone-defines 2-OR. Let m be the arity of R and let $t_1, t_2, t_3 \in R$ and $u \in \{0, 1\}^m \setminus R$ such that $\varphi_1(t_1, t_2, t_3) = u$.

We cone-define the 2-OR relation on variables x_1 and x_2 via the tuple (y_1, \dots, y_m) as follows. Consider $i \in [m]$, if $t_{1,i} = t_{2,i} = t_{3,i} = u_i$ let y_1 have the constant value $u_i \in \{0, 1\}$. If $(t_{1,i}, t_{2,i}, t_{3,i}) = (1, 1, 0)$ let $y_i := x_1$, if $(t_{1,i}, t_{2,i}, t_{3,i}) = (0, 0, 1)$ let $y_i := \neg x_1$. Similarly, if $(t_{1,i}, t_{2,i}, t_{3,i}) = (0, 1, 1)$ let $y_i := x_2$, if $(t_{1,i}, t_{2,i}, t_{3,i}) = (1, 0, 0)$ let $y_i := \neg x_2$. Note that this covers all cases for the value of $(t_{1,i}, t_{2,i}, t_{3,i})$.

Let $f : \{x_1, x_2\} \rightarrow \{0, 1\}$. We show that $f(x_1) \vee f(x_2)$ if and only if $(\hat{f}(y_1), \dots, \hat{f}(y_m)) \in R$, with \hat{f} as in Definition 2.10. We do a case distinction. If $f(x_1) = 1$ and $f(x_2) = 1$, then one may verify that $(\hat{f}(y_1), \dots, \hat{f}(y_m)) = t_2 \in R$. If $f(x_1) = 0$ and $f(x_2) = 1$, then $(\hat{f}(y_1), \dots, \hat{f}(y_m)) = t_3$. If $f(x_2) = 0$ and $f(x_1) = 1$, then $(\hat{f}(y_1), \dots, \hat{f}(y_m)) = t_1 \in R$. Finally, if $f(x_1) = f(x_2) = 0$, then $(\hat{f}(y_1), \dots, \hat{f}(y_m)) = u \notin R$, as desired.

To complete the cone-definition, it remains to show that there are $i, j \in [m]$ such that $y_i \in \{x_1, \neg x_1\}$ and $y_j \in \{x_2, \neg x_2\}$. Now observe that $t_1 \neq t_2 \neq t_3$: if $t_1 = t_2$ then by definition of φ_1 we have $\varphi_1(t_1, t_2, t_3) = t_3 \in R$, and similarly if $t_2 = t_3$ then $\varphi_1(t_1, t_2, t_3) = t_1 \in R$. Since $\varphi_1(t_1, t_2, t_3) \notin R$, this cannot be. If $t_1 = t_3 \neq t_2$, then consider a coordinate $k \in [m]$ for which $t_{1,k} = t_{3,k} \neq t_{2,k}$; but then φ_1 is not defined on this coordinate. Consequently, all three tuples t_1, t_2, t_3 are distinct. Now, since the argumentation above shows that $(\hat{f}(y_1), \dots, \hat{f}(y_m))$ can evaluate to each of t_1, t_2, t_3 , depending on the values assigned to x_1 and x_2 , it follows that \hat{f} depends on both x_1 and x_2 which implies the existence of the desired positions $i, j \in [m]$.

(\Leftarrow) Suppose there exists $R \in \Gamma$ of some arity m that cone-defines 2-OR. Let (y_1, \dots, y_m) be the tuple witnessing this, with $y_i \in \{0, 1\} \cup \{x_1, \neg x_1, x_2, \neg x_2\}$. Define $f : \{x_1, x_2\} \rightarrow \{0, 1\}$ as $f(x_1) = 0$ and $f(x_2) = 1$. Let $t_1 := (\hat{f}(y_1), \dots, \hat{f}(y_m)) \in R$ where \hat{f} is the natural extension of f . Similarly, for $f'(x_1) = f'(x_2) = 1$ let $t_2 := (\hat{f}'(y_1), \dots, \hat{f}'(y_m))$ and for $f''(x_1) = 1, f''(x_2) = 0$ let $t_3 := (\hat{f}''(y_1), \dots, \hat{f}''(y_m))$. Finally, let u be the tuple witnessing that for $f'''(x_1) = f'''(x_2) = 0, (\hat{f}'''(y_1), \dots, \hat{f}'''(y_m)) = u \notin R$. We will show that $\varphi_1(t_1, t_2, t_3) = u$, thus showing that φ_1 does not preserve R .

We show that $u_i = t_{1,i} - t_{2,i} + t_{3,i}$ for each $i \in [m]$. Suppose $y_i = 0$, then by the definition above $t_{1,i} = t_{2,i} = t_{3,i} = u_i = 0$ as $\hat{f}(y_i) = 0$ for any $f : \{x_1, x_2\} \rightarrow \{0, 1\}$. Thus, $u_i = t_{1,i} - t_{2,i} + t_{3,i}$ in this case. Similarly, if $y_i = 1$ we obtain $u_i = 1 = t_{1,i} - t_{2,i} + t_{3,i}$.

Suppose $y_i = x_1$. Then by the definition above, $t_{1,i} = 0$ and $t_{2,i} = t_{3,i} = 1$. Furthermore, $u_i = 0 = 1 - 1 + 0$ as desired. For $y_i = \neg x_1$, we have $t_{1,i} = 1$ and $t_{2,i} = t_{3,i} = 0$ and $u_i = 1$, as desired. It is straightforward to verify that also when $y_i = x_2$ and $y_i = \neg x_2$ we obtain $u_i = t_{1,i} - t_{2,i} + t_{3,i}$, concluding the proof. \square

2.6 Unary Constraints and Reductions via Definability

We now give some results about the existence of linear parameter transformations. These will be used to obtain kernelization lower bounds.

Definition 2.13 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 $\Gamma \cup \{\{(d)\} \mid d \in D\}$.

Effectively, the added relations in Γ^* make it possible to enforce via a constraint that a variable must be assigned a fixed value by any satisfying assignment.

Theorem 2.14 (Follows from [3]) *Let Γ be a constraint language over a finite set D such that each unary operation $u: D \rightarrow D$ that preserves Γ is a bijection. Then, there exists a linear-parameter transformation from $\text{CSP}(\Gamma^*)$ to $\text{CSP}(\Gamma)$, parameterized by the number of variables.*

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 $\text{CSP}(\Gamma^*)$ to $\text{CSP}(\Gamma)$.

Proof (Theorem 2.14) 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 $\text{CSP}(\Gamma^*)$ with n variables to an instance of $\text{CSP}(\Gamma \cup \{=_D\})$ 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 $\text{CSP}(\Gamma)$ has $\leq n + |D|$ variables. \square

Definition 2.15 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 $\text{CSP}(\Gamma)$ and there exist pairwise distinct variables $x_1, \dots, x_k \in V$ such that, for each map $f: \{x_1, \dots, x_k\} \rightarrow \{0, 1\}$, it holds that f can be extended to a satisfying assignment of the instance if and only if $(f(x_1), \dots, f(x_k)) \in T$.

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

Proposition 2.16 *If Γ is an intractable Boolean constraint language, then every Boolean relation is pp-definable from Γ^* .*

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 while blowing up the number of variables by only a constant factor, and may thus be used to prove lower bounds on the sparsification complexity of $\text{CSP}(\Gamma)$.

Proposition 2.17 *Suppose that Γ is an intractable finite Boolean constraint language, and that Δ is a finite Boolean constraint language such that each relation in Δ is cone-definable from a relation in Γ . Then, there exists a linear-parameter transformation from $\text{CSP}(\Gamma^* \cup \Delta)$ to $\text{CSP}(\Gamma)$.*

Proof It suffices to give a linear-parameter transformation from $\text{CSP}(\Gamma^* \cup \Delta)$ to $\text{CSP}(\Gamma^*)$, by Theorem 2.14. Let (\mathcal{C}, V) be an instance of $\text{CSP}(\Gamma^* \cup \Delta)$, and let n denote $|V|$. We generate an instance (\mathcal{C}', V') of $\text{CSP}(\Gamma^*)$ as follows.

- For each variable $v \in V$, introduce a primed variable v' . By Proposition 2.16, 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.¹ 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 add the constraint $\{(b)\}(z_b)$ to \mathcal{C}' .
- For each constraint $T(v_1, \dots, v_k)$ in \mathcal{C} such that $T \in \Gamma^*$, add the constraint to \mathcal{C}' .
- For each constraint $T(v_1, \dots, v_k)$ 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 add a constraint to \mathcal{C}' that has the same effect as $T(v_1, \dots, v_k)$. In particular, assume that T is cone-definable from $U \in \Gamma$ via the tuple (y_1, \dots, y_ℓ) , and that U has arity ℓ . Add to \mathcal{C}' the constraint $U(w_1, \dots, w_\ell)$, where, for each $i \in [\ell]$, the entry w_i is defined as follows:

$$w_i = \begin{cases} v_j & \text{if } y_i = x_j, \\ v'_j & \text{if } y_i = \neg x_j, \\ z_0 & \text{if } y_i = 0, \text{ and} \\ z_1 & \text{if } y_i = 1. \end{cases}$$

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 $f: V \rightarrow \{0, 1\}$ satisfies \mathcal{C} if and only if there exists an assignment $f': V' \rightarrow \{0, 1\}$ of f that satisfies \mathcal{C}' . □

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 $O(n^{k-\epsilon})$. 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 $O(n^{k-1})$. Using Lemmas 3.9 and 3.11, the next theorem will give a complete classification of the constraint languages that admit a non-trivial sparsification. We defer its proof to the end of this section.

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

¹ It is possible [18,24] to achieve $d = 2$, although this will not be necessary for our purposes.

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

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

Definition 3.2 Let R be a k -ary Boolean relation. We say that a polynomial p_u over a ring E_u captures an unsatisfying assignment $u \in \{0, 1\}^k \setminus R$ with respect to R , if the following two conditions hold over E_u .

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

$$p_u(u_1, \dots, u_k) \neq 0. \tag{2}$$

In Theorem 3.5, we will generalize the following theorem which was proven by a subset of the current authors. We recall the required terminology. Let E be a ring. Define d -POLYNOMIAL ROOT CSP over E as the problem whose input consists of a set L of polynomial equalities over E of degree at most d , over a set of variables V . Each equality is of the form $p(x_1, \dots, x_k) = 0$ (over E). The question is whether there exists a Boolean assignment to the variables in V that satisfies all equalities in L .

Theorem 3.3 ([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.

We say a field F is *efficient* if the field operations and Gaussian elimination can be done in polynomial time in the size of a reasonable input encoding.

Theorem 3.4 ([15, Theorem 5]) *There is a polynomial-time algorithm that, given an instance (L, V) of d -POLYNOMIAL ROOT CSP over an efficient field F , outputs an equivalent instance (L', V) with at most $n^d + 1$ constraints such that $L' \subseteq L$.*

Observe that the above theorem statement in particular applies to instances of d -POLYNOMIAL ROOT CSP over \mathbb{Q} , since \mathbb{Q} is an efficient field.

We present a generalization of Theorem 3.3. 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 3.5 *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 ring $E_u \in \{\mathbb{Q}\} \cup \{\mathbb{Z}/q_u\mathbb{Z} \mid q_u > 1 \text{ and } q_u \in \mathbb{N}\}$ and polynomial p_u over E_u of degree at most d that captures u with respect to R . 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 $|\mathcal{C}'| = O(n^d)$, such that any Boolean assignment satisfies all constraints in \mathcal{C} if and only if it satisfies all constraints in \mathcal{C}' .

Proof 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 with variable set V . For each $u \in R \setminus \{0, 1\}^k$, we create an instance (L_u, V) of d -POLYNOMIAL ROOT CSP over E_u , as follows. Choose a ring $E_u \in \{\mathbb{Q}\} \cup \{\mathbb{Z}/q_u\mathbb{Z} \mid q_u > 1 \text{ and } q_u \in \mathbb{N}\}$ and a polynomial p_u over E_u such that (1) and (2) are satisfied for u . For each constraint $(x_1, \dots, x_k) \in \mathcal{C}$, add the equality $p_u(x_1, \dots, x_k) = 0$ to the set L_u ; note that these are equations over the ring E_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 property.

Claim 3.6 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 $(x_1, \dots, x_k) \in \mathcal{C}$, such that $(f(x_1), \dots, f(x_k)) \notin R$. Let $u := (f(x_1), \dots, f(x_k))$. Since $u \notin R$, the equation $p_u(x_1, \dots, x_k) = 0$ was added to $L_u \subseteq L$. However, it follows from (2) that $p_u(f(x_1), \dots, f(x_k)) \neq 0$, which contradicts the assumption that f satisfies all equalities in L . ■

For each instance (L_u, V) of d -POLYNOMIAL ROOT CSP over E_u with $E_u \neq \mathbb{Q}$, apply Theorem 3.3 to obtain an equivalent instance (L'_u, V) with $L'_u \subseteq L_u$ and $|L'_u| = O(r_u n^d)$, where r_u is the number of distinct prime divisors of q_u . Similarly, for each instance (L_u, V) of d -POLYNOMIAL ROOT CSP over E_u with $E_u = \mathbb{Q}$, apply Theorem 3.4 and obtain an equivalent instance (L'_u, V) with $L'_u \subseteq L_u$ and $|L'_u| = O(n^d)$. 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 $(x_1, \dots, x_k) \in \mathcal{C}$, add (x_1, \dots, x_k) to \mathcal{C}' if there exists $u \in \{0, 1\}^k \setminus R$ such that $p_u(x_1, \dots, x_k) = 0 \in L'$. Hereby, $\mathcal{C}' \subseteq \mathcal{C}$. The following two claims show the correctness of this sparsification procedure.

Claim 3.7 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 $p_u(x_1, \dots, x_k) = 0 \in L'$. Thereby, $(x_1, \dots, x_k) \in \mathcal{C}'$ and since f is a satisfying assignment, $(f(x_1), \dots, f(x_k)) \in R$. It follows from property (1) that $p_u(f(x_1), \dots, f(x_k)) = 0$ as desired.

Since f satisfies all equalities in L' , it satisfies all equalities in L by the choice of L' . It follows from Claim 3.6 that thereby f satisfies all constraints in \mathcal{C} . ■

Claim 3.8 $|\mathcal{C}'| = O(n^d)$.

Proof By the construction of \mathcal{C}' , it follows that $|\mathcal{C}'| \leq |L'|$. Let r be the maximum of all r_u for $u \in \{0, 1\}^k \setminus R$. Since R is fixed, r is a constant. We know $|L'| = \sum_{u \notin R} |L'_u| \leq \sum_{u \notin R} O(r \cdot n^d) \leq 2^k \cdot O(r \cdot n^d) = O(n^d)$, as $|R| \leq 2^k$ and k is a constant. ■

Claims 3.7 and 3.8 complete the proof of Theorem 3.5. □

The next lemma states that any k -ary Boolean relation R with $|R| < 2^k - 1$ 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 3.5. Since relations with $|R| = 2^k$ have a sparsification of size $O(1)$, as constraints over such relations 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 3.9 *Let R be a k -ary Boolean 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 $|\mathcal{C}'| = O(n^{k-1})$, such that a Boolean assignment satisfies all constraints in \mathcal{C}' if and only if it satisfies all constraints in \mathcal{C} .*

Proof 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 \mathbb{Q} of degree at most $k - 1$ satisfying (1) and (2), such that the result follows from Theorem 3.5.

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 $u = (u_1, \dots, u_k) \in \{0, 1\}^k \setminus R$. Since $|R| < 2^k - 1$, we can choose $w = (w_1, \dots, w_k)$ 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 $i \in [k]$ for which $u_i = w_i$) This implies $u_i = \neg w_i$ for all i . One may note that for $u = (0, \dots, 0)$ and $w = (1, \dots, 1)$ this situation corresponds to MONOTONE k -NAE-SAT. We show that there exists a polynomial p_u such that $p_u(u_1, \dots, u_k) \neq 0$, and $p_u(x_1, \dots, x_k) = 0$ for all $(x_1, \dots, x_k) \in R$. 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 $r_i(u_i) = 0$ and $r_i(w_i) = 1$ for all $i \in [k]$. Define

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

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

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

as desired. Since $r_i(w_i) = 1$ for all i , we obtain $p_u(w_1, \dots, w_k) = \prod_{i=1}^{k-1} (i - k) \neq 0$, which is allowed since $w \notin R$. It is easy to verify that in all other cases, $\sum_{j=1}^k r_j(x_j) \in$

$\{1, 2, \dots, k - 1\}$ and thereby one of the terms of the product is zero, implying that $p_u(x_1, \dots, x_k) = 0$.

(**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

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

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 $k - 2$, such that $p_{u'}(u'_1, \dots, u'_{k-1}) \neq 0$ and $p_{u'}(x'_1, \dots, x'_{k-1}) = 0$ for all $x' \in R'$. Now define

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

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

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

Since we have shown for all $u \in \{0, 1\}^k \setminus R$ that there exists a polynomial p_u over \mathbb{Q} satisfying (1) and (2), the proof of Lemma 3.9 now follows from Theorem 3.5. \square

The reader may find it interesting to compare the preceding proof to Section 3.1 in recent work by Lagerkvist and Wahlström [21], which uncovers structural properties of relations defined via roots of low-degree polynomials using similar techniques.

To show the other part of the nontrivial versus no nontrivial sparsification dichotomy, we will need the following theorem. It combines existing ideas for kernelization lower bounds due to several authors [7,9,19] with the formalism of cone-definition.

Theorem 3.10 *Let Γ be an intractable finite Boolean constraint language, and let $k \geq 1$. If there exists $R \in \Gamma$ such that R cone-defines k -OR, then $\text{CSP}(\Gamma)$ does not have a generalized kernel of bitsize $O(n^{k-\varepsilon})$ for any $\varepsilon > 0$, unless $\text{NP} \subseteq \text{coNP/poly}$.*

Proof We do a case distinction on k .

($k = 1$) Suppose that there exists $\varepsilon > 0$ such that $\text{CSP}(\Gamma)$ has a (generalized) kernel of size $O(n^{1-\varepsilon})$ into a parameterized decision problem L . Let $\tilde{L} := \{x\#1^k \mid (x, k) \in L\}$ denote the classical (non-parameterized) problem corresponding to L , in which the parameter is written in unary and appended to the end of each input string.

Here $\#$ is an arbitrary character in the encoding alphabet, and $\#$ is a separator character that is freshly added to the alphabet.

Using this hypothetical generalized kernel, one could obtain a polynomial-time algorithm that takes as input a series of instances $(\mathcal{C}_1, V_1), \dots, (\mathcal{C}_t, V_t)$ of $\text{CSP}(\Gamma)$, and outputs in polynomial time an instance \tilde{x} of \tilde{L} such that:

- $\tilde{x} \in \tilde{L}$ if and only if (\mathcal{C}_i, V_i) is a YES-instance of $\text{CSP}(\Gamma)$ for all $i \in [t]$, and
- \tilde{x} has bitsize $O(N^{1-\varepsilon})$, where $N := \sum_{i=1}^t |V_i|$.

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

1. On input a series of instances $(\mathcal{C}_1, V_1), \dots, (\mathcal{C}_t, V_t)$ of $\text{CSP}(\Gamma)$, form a new instance $(\mathcal{C}^* := \bigcup_{i=1}^t \mathcal{C}_i, V^* := \bigcup_{i=1}^t V_i)$ of $\text{CSP}(\Gamma)$. 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^*, k^*) of L with size and parameter bounded by $O(N^{1-\varepsilon})$. Let \tilde{x} be the classical instance corresponding to (x^*, k^*) .

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 $O(|V^*|^{1-\varepsilon}) = O((m \cdot m^\alpha)^{1-\varepsilon}) = O(m^{(1+\alpha)(1-\varepsilon)})$ 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 one instance expressing the logical AND, of size at most $t_2(m) \leq O(m^{(1+\alpha)(1-\varepsilon)}) \leq C \cdot t_1(m)$ 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 $\text{NP} \subseteq \text{coNP/poly}$. Hence the lower bound for $k = 1$ follows since $\text{CSP}(\Gamma)$ is NP-complete.

($k \geq 2$) For $k \geq 2$, we prove the lower bound using a linear-parameter transformation (recall Definition 2.2). 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 2.17 that there is a linear-parameter transformation from $\text{CSP}(\Gamma^* \cup \Delta)$ to $\text{CSP}(\Gamma)$. Thus, to prove the lower bound for $\text{CSP}(\Gamma)$, it suffices to prove the desired lower bound for $\text{CSP}(\Gamma^* \cup \Delta)$.

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

Suppose we are given a graph $G = (V, E)$ on n vertices and integer $k \leq n$, forming an instance of the VERTEX COVER problem. The question is whether there is a set S of k vertices, such that each edge has at least one endpoint in S . We create an equivalent instance (\mathcal{C}, V') of $\text{CSP}(\Gamma^* \cup \Delta)$ 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 of the vertex cover is bounded by k .

Let $H_{n,k}$ be the n -ary relation given by $H_{n,k} = \{(x_1, \dots, x_n) \mid x_i \in \{0, 1\} \text{ for all } i \in [n] \text{ and } \sum_{i \in [n]} x_i = k\}$. By Proposition 2.16, we obtain that Γ^* pp-defines all Boolean relations. It follows from [19, Lemma 17] that Γ^* pp-defines $H_{n,k}$ using $O(n + k)$ constraints and $O(n + k)$ existentially quantified variables. We add the constraints from this pp-definition to \mathcal{C} , and add the existentially quantified variables to V' . 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 $O(n + k) \in O(n)$ variables and thereby this is a linear-parameter transformation from VERTEX COVER to $\text{CSP}(\Gamma^* \cup \Delta)$.

($k \geq 3$) In this case there is a trivial linear-parameter transformation from $\text{CSP}(\Delta)$ to $\text{CSP}(\Gamma^* \cup \Delta)$. It is easy to verify that $\text{CSP}(\Delta)$ 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 $O(n^{k-\varepsilon})$ for any $\varepsilon > 0$, unless $\text{NP} \subseteq \text{coNP/poly}[7]$. \square

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 3.9, together with the next lemma and Theorem 3.10.

Lemma 3.11 *If R is a Boolean k -ary relation with $|R| = 2^k - 1$, then R cone-defines k -OR.*

Proof Let $u = (u_1, \dots, u_k)$ be the unique k -tuple not contained in R . Define the tuple (y_1, \dots, 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 $f: \{x_1, \dots, x_k\} \rightarrow \{0, 1\}$. Suppose $(f(x_1), \dots, f(x_k)) \in k\text{-OR}$. We show $(\hat{f}(y_1), \dots, \hat{f}(y_k)) \in R$. It follows from $(f(x_1), \dots, f(x_k)) \in k\text{-OR}$, that 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 $(\hat{f}(y_1), \dots, \hat{f}(y_k)) \neq u$, implying $(\hat{f}(y_1), \dots, \hat{f}(y_k)) \in R$.

Suppose $(f(x_1), \dots, f(x_k)) \notin k\text{-OR}$, 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 $(\hat{f}(y_1), \dots, \hat{f}(y_k)) = u \notin R$. \square

Using the above results, we can now prove Theorem 3.1.

Proof (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 $\text{CSP}(\Gamma)$, with set of constraints \mathcal{C} . We show how to define $\mathcal{C}' \subseteq \mathcal{C}$. For each constraint $R(x_1, \dots, x_\ell) \in \mathcal{C}$ 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 $O(n^\ell)$ 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 $R(x_1, \dots, x_k)$. For all k -ary relations R with $|R| < 2^k - 1$, apply Lemma 3.9 to obtain $\mathcal{C}'_R \subseteq \mathcal{C}_R$ such that $|\mathcal{C}'_R| = O(n^{k-1})$ and any Boolean assignment satisfies all constraints in \mathcal{C}'_R if and only if it satisfies the constraints in \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 $R(x_1, \dots, x_k)$ with $|R| = 2^k$, as these are always satisfied. It is easy to verify that $|\mathcal{C}'| \leq |\Gamma| \cdot O(n^{k-1}) = O(n^{k-1})$. Since each constraint can be stored in $O(\log n)$ bits, this gives a kernel of bitsize $O(n^{k-1} \log n)$.

Suppose that there exists $R \in \Gamma$ with $|R| = 2^k - 1$. It follows from Lemma 3.11 that R cone-defines k -OR. Since Γ is intractable, it now follows from Theorem 3.10 that $\text{CSP}(\Gamma)$ has no generalized kernel of size $O(n^{k-\varepsilon})$, unless $\text{NP} \subseteq \text{coNP/poly}$. \square

4 From Balanced Operations to Linear Sparsification

The main result of this section is the following theorem, which we prove below.

Theorem 4.1 *Let Γ be a finite balanced Boolean constraint language. Then $\text{CSP}(\Gamma)$ has a kernel with $O(n)$ constraints, which are a subset of the original constraints. The kernel can be stored using $O(n \log n)$ 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 $S = \{s_1, \dots, s_n\}$ 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}^k for which there exist $\alpha_1, \dots, \alpha_n \in \mathbb{Z}$ 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 $\alpha_1, \dots, \alpha_n$ 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 $i \in [n]$ to denote the i 'th row of S .

Definition 4.2 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 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 $\text{gcd}(x, y)$ and their least common multiple as $\text{lcm}(x, y)$. The two concepts are closely related, since $\text{lcm}(x, y) = |x \cdot y| / \text{gcd}(x, y)$ for x or y non-zero. Recall that by Bézout's lemma, if $\text{gcd}(x, y) = z$ then 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. The proof of the following lemma was contributed by Emil Jeřábek.

Lemma 4.3 *Let S be an $m \times n$ integer matrix. Let $u \in \mathbb{Z}^n$ be a row vector. If $u \notin \text{span}_{\mathbb{Z}}(\{s_1, \dots, s_m\})$, then there exists a prime power q such that $u \notin \text{span}_q(\{s_1, \dots, s_m\})$. Furthermore, there is a polynomial-time algorithm that computes a (possibly composite) integer q' for which $u \notin \text{span}_{q'}(\{s_1, \dots, s_m\})$.*

Proof Suppose $u \notin \text{span}_{\mathbb{Z}}(\{s_1, \dots, s_m\})$, thus u cannot be written as a linear combination of the rows of S over \mathbb{Z} ; equivalently, 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 $u \notin \text{span}_q(\{s_1, \dots, s_m\})$.

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 (cf. [10, Theorem 368]). In particular, this implies that S' is a diagonal matrix. Furthermore, M , S , and N can be computed in polynomial time [16, Theorem 4]. Define $u' := uN$.

Claim 4.4 *If $y'S' = u'$ is solvable for y' over \mathbb{Z} , then $yS = u$ is solvable for y over \mathbb{Z} .*

Proof Consider y' such that $y'S' = u'$. One can verify that $y := y'M$ solves $yS = u$, as

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

Claim 4.5 *Let $q \in \mathbb{N}$. If $yS \equiv_q u$ is solvable for y , then $y'S' \equiv_q u'$ is solvable for y' .*

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

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

Using these two claims, our proof proceeds as follows. From our starting assumption $u \notin \text{span}_{\mathbb{Z}}(\{s_1, \dots, s_m\})$, it follows by Claim 4.4 that $y'S' = u'$ has no solution y' over \mathbb{Z} . Below, we prove that this implies there exists a prime power q such that $y'S' \equiv_q u'$ is unsolvable. Furthermore we show how to find a (possibly composite) integer q in polynomial time such that $y'S' \equiv_q u'$ is unsolvable. By Claim 4.5 this will imply that $yS \equiv_q u$ is unsolvable and complete the proof.

We inferred that $y'S' = u'$ has no solutions over \mathbb{Z} . 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 $q \nmid u'_i$. Observe that such q can be obtained in polynomial-time, for example by taking the smallest power of two that is larger than 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 $y'S' \equiv_q u'$ has no solution.
- Otherwise, there exists $i \in [n]$ such that $s'_{i,i} \nmid u'_i$. Choose a prime power q such that $q \nmid u'_i$ and $q \mid s'_{i,i}$. Such a prime power can be chosen by letting $q := p^\ell$ for a prime p that occurs $\ell \geq 1$ times in the prime factorization of $s'_{i,i}$, but less often in the prime factorization of u'_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.

It is not clear how to obtain q as described above in polynomial time, as it requires computing the prime decompositions of possibly large integers. However, observe that letting $q := s'_{i,i}$ means $u'_i \not\equiv_q 0$ while $s'_{i,i} \equiv_q 0$, implying that for this choice of q the system $y'S' \equiv_q u'$ again has no solutions. Since this q is trivial to obtain in polynomial time, that concludes the proof. □

Given that $u \notin \text{span}_{\mathbb{Z}}(\{s_1, \dots, s_m\})$, we can thus use Lemma 4.3 to obtain an integer q such that $u \notin \text{span}_q(\{s_1, \dots, s_m\})$. The next lemma shows that in this case the system $S'x \equiv_q b$, where $b = (0, \dots, 0, c)$ and S' is the matrix consisting of rows $\{s_1, \dots, s_m, u\}$, has a solution x for some $c \not\equiv_q 0$.

Lemma 4.6 *Let $q > 1$ be an integer. Let A be an $m \times n$ matrix over $\mathbb{Z}/q\mathbb{Z}$. Suppose $a_m \notin \text{span}_q(\{a_1, \dots, a_{m-1}\})$. Then there exists a constant $c \not\equiv_q 0$ for which the system $Ax \equiv_q b$ has a solution, where $b := (0, \dots, 0, c)^T$ is the vector with c on the last position and zeros in all other positions. Furthermore, x and c can be computed in polynomial time.*

Proof 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} . (The only property of Smith Normal Form we rely on is that S' is a diagonal matrix.)

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, $S' \pmod q$ remains a diagonal matrix.

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

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

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

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

Define $S := MAN$ and verify that

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

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 4.7 *Let $b := (0, \dots, 0, c)$ 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 such that $Ax \equiv_q b$. 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 4.8 *$s_m \in \text{span}_q(\{s_1, \dots, s_{m-1}\})$ if and only if $a_m \in \text{span}_q(\{a_1, \dots, a_{m-1}\})$.*

Proof Suppose $s_m \in \text{span}_q(\{s_1, \dots, s_{m-1}\})$. This implies that there exist $\alpha_1, \dots, \alpha_{m-1}$ 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 $\alpha = (\alpha_1, \dots, \alpha_{m-1})$ 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 $a_m \in \text{span}_q(\{a_1, \dots, a_{m-1}\})$.

For the other direction, suppose $a_m \in \text{span}_q(\{a_1, \dots, a_{m-1}\})$. Thus, there exists $\alpha \equiv_q (\alpha_1, \dots, \alpha_{m-1})$ 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 it follows from the definition of S in (3) that $s_m \in \text{span}_q(\{s_1, \dots, s_{m-1}\})$. ■

It follows from Claims 4.7 and 4.8, that it suffices to show that $Sx = (0, \dots, 0, c)^T$ has a solution for some $c \not\equiv_q 0$ if $s_m \notin \text{span}_q(\{s_1, \dots, s_{m-1}\})$. Observe that since S' (the first $m - 1$ rows of S) is a diagonal matrix, there must exist $i \in [m - 1]$ for which there is no α_i satisfying $s_{i,i} \cdot \alpha_i \equiv_q s_{m,i}$. Otherwise, it is easy to see that $\sum_{i \in [m-1]} \alpha_i s_i \equiv_q s_m$, contradicting that $s_m \notin \text{span}_q(\{s_1, \dots, s_{m-1}\})$. 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 $x = (0, \dots, 0, 1, 0, \dots, 0)$ be the vector with 1 in the i 'th position and zeros in all other positions. It is easy to verify that $Sx \equiv_q (0, \dots, 0, s_{m,i})^T$ and thereby the system $Sx \equiv_q b$ has a solution for $c = s_{m,i}$.

Otherwise, choose i such that $s_{i,i} \not\equiv_q 0$ and there exists no integer α_i satisfying $s_{i,i} \cdot \alpha_i \equiv_q s_{m,i}$. We consider the following two cases.

- Suppose $\text{gcd}(s_{i,i}, q) \mid s_{m,i}$ over the integers. Let $d \in \mathbb{Z}$ such that $s_{m,i} = d \cdot \text{gcd}(s_{i,i}, q)$ over the integers. It follows from Bézout's lemma that there exist integers a and b such that $\text{gcd}(s_{i,i}, q) = a \cdot s_{i,i} + b \cdot q$. Thereby,

$$s_{m,i} \equiv_q d \cdot (a \cdot s_{i,i} + b \cdot q) \equiv_q d \cdot a \cdot s_{i,i}$$

which is a contradiction with the assumption that no integer α_i exists such that $s_{i,i} \cdot \alpha_i \equiv_q s_{m,i}$.

- Suppose $\text{gcd}(s_{i,i}, q) \nmid s_{m,i}$ over the integers, let $y := q / \text{gcd}(s_{i,i}, q)$. Define $x := (0, \dots, 0, y, 0, \dots, 0)^T$ as the vector with y in position i . Then

$$Sx \equiv_q (0, \dots, 0, y \cdot s_{i,i}, 0, \dots, 0, y \cdot s_{m,i})^T.$$

Then $y \cdot s_{i,i} = s_{i,i} \cdot q / \text{gcd}(s_{i,i}, q) = \pm \text{lcm}(q, s_{i,i}) \equiv_q 0$. It remains to show that $y \cdot s_{m,i} \not\equiv_q 0$. Suppose for contradiction that $y \cdot s_{m,i} \equiv_q 0$, such that over the integers $q \cdot s_{m,i} / \text{gcd}(s_{i,i}, q) = d \cdot q$ for some $d \in \mathbb{Z}$. But then $s_{m,i} / \text{gcd}(s_{i,i}, q) = d \in \mathbb{Z}$ contradicting that $\text{gcd}(s_{i,i}, q) \nmid s_{m,i}$. It follows that $y \cdot s_{m,i} \not\equiv_q 0$. Thus, for x defined as above and $c := y \cdot s_{m,i} \not\equiv_q 0$ we obtain that $Sx = (0, \dots, 0, c)^T$, concluding the proof.

Computing the Smith Normal Form of a matrix can be done in polynomial time [16] and computing the greatest common divisor of two integers can be done in time polynomial in their binary encoding. Hereby, it is straightforward to turn the above proof in a polynomial-time algorithm that computes both x and c . \square

In the context of capturing a Boolean relation R by degree-1 polynomials, the constructive proof of Lemma 4.6 effectively shows the following: given a ring $\mathbb{Z}/q\mathbb{Z}$ over which a degree-1 polynomial exists that captures a certain tuple $u \notin R$, one can constructively find the coefficients x of a polynomial that captures u by following the steps in the proof. We will formalize this idea in Theorem 4.10, but first we prove the main sparsification result.

Proof (Theorem 4.1) We start by showing in the following claim that for all relations $R \in \Gamma$ we can find linear polynomials over an appropriate ring that capture the tuples that are not in R . We will then use this representation to obtain our kernel.

Claim 4.9 *For all relations R in a balanced Boolean constraint language Γ , for all $u \notin R$, there exists a linear polynomial p_u over a ring $E_u \in \{\mathbb{Z}/q_u\mathbb{Z} \mid q_u \text{ is a prime power}\}$ that captures u with respect to R .*

Proof Suppose for a contradiction that there exists $R \in \Gamma$ and $u \notin R$, such that no prime power q and linear polynomial p over $\mathbb{Z}/q\mathbb{Z}$ exist that capture u with respect to R . Let $R = \{r_1, \dots, r_\ell\}$. By the non-existence of p and q , the system

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

has no solution for any prime power q and $c \not\equiv_q 0$. Otherwise, it is easy to verify that q is the desired prime power and $p(x_1, \dots, x_k) := \alpha_0 + \sum_{i=1}^k \alpha_i x_i$ is the desired polynomial.

The fact that no solution exists, implies that $(1, u_1, \dots, u_k)$ is in the span of the remaining rows of the matrix, by Lemma 4.6. Thus, for any prime power q , we obtain that $(1, u_1, \dots, u_k)$ is in the span of the first ℓ rows of this matrix, over the integers modulo q . By the contrapositive of Lemma 4.3, it follows that the same must hold over \mathbb{Z} . Therefore, there exist integer coefficients $\gamma_1, \dots, \gamma_\ell$ such that $\sum \gamma_i = 1$ (since the first column consists of ones) and furthermore $u = \sum \gamma_i r_i$. But it immediately follows that $R \in \Gamma$ is not preserved by the balanced operation given by $f(x_1, \dots, x_\ell) := \sum \gamma_i x_i$, which contradicts the assumption that Γ is balanced. \blacksquare

By applying Theorem 3.5 once for each relation $R \in \Gamma$, to reduce the number of constraints involving R to $O(n)$, we then reduce any n -variable instance of $\text{CSP}(\Gamma)$ to an equivalent one on $|\Gamma| \cdot O(n) \in O(n)$ constraints. \square

The next theorem shows that given an arbitrary constraint language, it is possible to decide in polynomial time whether it is balanced. Furthermore, for balanced constraint

languages there is a polynomial-time algorithm to obtain the capturing polynomials. Here we assume that the relation R is encoded explicitly as a list of tuples contained in the relation, together with a list of tuples not contained in the relation, making the total encoding size of a Boolean k -ary relation at least 2^k .

Theorem 4.10 *There is a polynomial-time algorithm that, given a k -ary Boolean relation R encoded as a full table, decides whether R is balanced. For balanced R , the algorithm outputs a linear polynomial p_u and integer q_u for all $u \in \{0, 1\}^k \setminus R$ such that p_u captures u over $\mathbb{Z}/q_u\mathbb{Z}$.*

Proof Let $R = \{r_1, \dots, r_m\}$. Define s_i as $(1, r_{i,1}, \dots, r_{i,k})$ for $i \in [m]$. Relation R is balanced if and only if there exists no $u \in \{0, 1\}^k \setminus R$ such that $\text{ext}(u) \in \text{span}_{\mathbb{Z}}(\{s_1, \dots, s_m\})$, where $\text{ext}(u) = (1, u_1, \dots, u_k)$. Thereby, R is balanced if and only if for all $u \in \{0, 1\}^k \setminus R$, the following system has no solutions over \mathbb{Z} .

$$(\alpha_1, \dots, \alpha_m) \begin{pmatrix} 1 & r_{1,1} & r_{1,2} & \dots & r_{1,k} \\ 1 & r_{2,1} & r_{2,2} & \dots & r_{2,k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & r_{m,1} & r_{m,2} & \dots & r_{m,k} \end{pmatrix} = (1, u_1, \dots, u_k)$$

The solvability of these systems over the integers can be tested in time that is polynomial in the size of an explicit encoding of R that indicates for each tuple whether or not it is contained in R . This computation can be done, for example, by first computing the Smith Normal Form of the matrix.

Suppose R is balanced. Let $u \in \{0, 1\}^k \setminus R$, we show how to find p_u and q_u . Let s_i be defined as above. Then $\text{ext}(u) \notin \text{span}_{\mathbb{Z}}(\{s_1, \dots, s_m\})$ since R is balanced. Apply Lemma 4.3 to obtain $q_u \in \mathbb{N}$ in polynomial time such that $\text{ext}(u) \notin \text{span}_{q_u}(\{s_1, \dots, s_m\})$. It follows from Lemma 4.6 that the following system has a solution

$$\begin{pmatrix} 1 & r_{1,1} & r_{1,2} & \dots & r_{1,k} \\ 1 & r_{2,1} & r_{2,2} & \dots & r_{2,k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & r_{m,1} & r_{m,2} & \dots & r_{m,k} \\ 1 & u_1 & u_2 & \dots & u_k \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_k \end{pmatrix} \equiv_{q_u} \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ c \end{pmatrix}$$

for $(\alpha_0, \alpha_1, \dots, \alpha_k)^T$ and some $c \not\equiv_{q_u} 0$ and that we can find it in polynomial time. Let $p_u(x_1, \dots, x_k) := \alpha_0 + \sum_{i=1}^k \alpha_i x_i$. It is straightforward to verify that p_u captures u with respect to R over $\mathbb{Z}/q_u\mathbb{Z}$. □

The kernelization result of Theorem 4.1 above is obtained by using the fact that when Γ is balanced, the constraints in $\text{CSP}(\Gamma)$ can be replaced by linear polynomials. We show in the next theorem that this approach fails when Γ is not balanced.

Theorem 4.11 *Let R be a k -ary Boolean relation that is not balanced. Then there exists $u \in \{0, 1\}^k \setminus R$ for which there exists no linear polynomial p_u over any ring E that captures u with respect to R .*

Proof Suppose R is not balanced. By Proposition 2.8, this implies R is violated by an alternating operation. Let f be an alternating operation that does not preserve R , such that $f(y_1, \dots, y_m) := \sum_{i=1}^m (-1)^{i+1} y_i$ for some odd m , and for some (not necessarily distinct) $r_1, \dots, r_m \in R$ we have $f(r_1, \dots, r_m) = u$ with $u \notin R$.

Suppose for a contradiction that there exists a linear polynomial p_u that captures u over a ring E_u . Let $r_i := (r_{i,1}, \dots, r_{i,k})$ for $i \in [m]$. Since $f(r_1, \dots, r_m) = u$, we have that u_i equals the following alternating sum over \mathbb{Z} :

$$u_i = r_{1,i} - r_{2,i} \cdots + r_{m,i}. \quad (4)$$

Since $r_{j,i} \in \{0, 1\}$ for all $i \in [k]$ and $j \in [m]$, Eq. (4) holds over any ring, so in particular over E_u . By Definition 3.2, $p_u(r_{i,1}, \dots, r_{i,k}) = 0$ for all $i \in [m]$. It thus follows from Eq. (4) and Proposition 2.3 that $p_u(u_1, \dots, u_k) = 0$. This contradicts the fact that p_u captures u with respect to R . Thus, there exists no linear polynomial that captures u with respect to R . \square

It is immediate from Claim 4.9 and Theorem 4.11 that a relation R is balanced if and only if every $u \notin R$ can be captured by a linear polynomial over $\mathbb{Z}/q_u\mathbb{Z}$ for some q_u . While this allows for using a different modulus for every unsatisfying assignment, it turns out that this is unnecessary and a single modulus always suffices, by the following observation.

Let p be a linear polynomial over $\mathbb{Z}/q_u\mathbb{Z}$ capturing u with respect to R . If $q = c \cdot q_u$ for some $c \in \mathbb{N}$, there exists a polynomial p' over $\mathbb{Z}/q\mathbb{Z}$ capturing u with respect to R . It is easy to obtain p' by multiplying all coefficients of p by c . If Γ is a finite balanced constraint language whose relations are captured using polynomials with moduli q_1, \dots, q_m , we may alternatively use capturing polynomials over the single modulus $\text{lcm}(q_1, \dots, q_m)$. This leads to the following corollary.

Corollary 4.12 *A finite Boolean constraint language Γ is balanced if and only if there exists a single integer $q \in \mathbb{N}$ such that for each relation $R \in \Gamma$, for each $u \in \{0, 1\}^k \setminus R$, there exist a linear polynomial over $\mathbb{Z}/q\mathbb{Z}$ that captures u with respect to R .*

5 Characterization of Symmetric CSPs with Linear Sparsification

In this section, we characterize the symmetric Boolean constraint languages Γ for which $\text{CSP}(\Gamma)$ has a linear sparsification.

Definition 5.1 We say a k -ary Boolean relation R is *symmetric*, if there exists $S \subseteq \{0, 1, \dots, k\}$ such that a tuple $x = (x_1, \dots, x_k)$ 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 Boolean 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 an intractable finite symmetric Boolean constraint language.*

- *If Γ is balanced, then $\text{CSP}(\Gamma)$ has a kernel with $O(n)$ constraints that can be stored in $O(n \log n)$ bits.*

– If Γ is not balanced, then $\text{CSP}(\Gamma)$ does not have a generalized kernel of size $O(n^{2-\varepsilon})$ for any $\varepsilon > 0$, unless $\text{NP} \subseteq \text{coNP/poly}$.

To show this, we use the following lemma. The lemma can be seen as a special case of Lemma 32 in [21], which uses a slightly different setting. We prove it here for consistency.

Lemma 5.3 *Let R be a k -ary symmetric Boolean relation with satisfying weights $S \subseteq \{0, 1, \dots, k\}$. Let $U := \{0, 1, \dots, k\} \setminus S$. 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 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 $x_1 \vee x_2$.

$$\left(\underbrace{\neg x_1, \dots, \neg x_1}_{(a-b) \text{ copies}}, \underbrace{\neg x_2, \dots, \neg x_2}_{(c-b) \text{ copies}}, \underbrace{1, \dots, 1}_b, \underbrace{0, \dots, 0}_{(k-d) \text{ copies}} \right).$$

Let $f : \{x_1, x_2\} \rightarrow \{0, 1\}$, then $(\neg f(x_1), \dots, \neg f(x_1), \neg f(x_2), \dots, \neg f(x_2), 1, \dots, 1, 0, \dots, 0)$ 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 by swapping a and c). In this case, use the tuple

$$\left(\underbrace{\neg x_1, \dots, \neg x_1}_{(d-a) \text{ copies}}, \underbrace{x_2, \dots, x_2}_{(b-a) \text{ copies}}, \underbrace{1, \dots, 1}_a, \underbrace{0, \dots, 0}_{(k-c) \text{ copies}} \right).$$

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

$$\left(\underbrace{x_1, \dots, x_1}_{(a-d) \text{ copies}}, \underbrace{x_2, \dots, x_2}_{(c-d) \text{ copies}}, \underbrace{1, \dots, 1}_d, \underbrace{0, \dots, 0}_{(k-b) \text{ copies}} \right).$$

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 $a - d + c = b \in S$. □

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 k -ary symmetric Boolean relation. 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) $r_1, \dots, r_m \in R$, such that $r_1 - r_2 + r_3 - r_4 \cdots + r_m = u$ for some $u \in \{0, 1\}^k \setminus R$ and odd $m \geq 3$. 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 $U := \{0, \dots, k\} \setminus S$. Define $s_i := \text{weight}(r_i)$ for $i \in [m]$, and $t = \text{weight}(u)$, such that $s_1 - s_2 + s_3 - s_4 \cdots + s_m = t$, 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 $s_1, \dots, s_m \in S$ 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 $s_i - s_\ell + s_j \in \{0, \dots, k\}$.*

Proof 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. ■

Use Claim 5.5 to find i, j, ℓ such that $s_i - s_\ell + s_j \in \{0, \dots, k\}$. 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 $s_i - s_\ell + s_j$ 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 (Theorem 5.2) If Γ is balanced, it follows from Theorem 4.1 that $\text{CSP}(\Gamma)$ has a kernel with $O(n)$ constraints that can be stored in $O(n \log n)$ bits. Note that the assumption that Γ is symmetric is not needed in this case.

If the symmetric constraint language Γ is not balanced, then Γ contains a symmetric 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 3.10 that $\text{CSP}(\Gamma)$ has no generalized kernel of size $O(n^{2-\epsilon})$ for any $\epsilon > 0$, unless $\text{NP} \subseteq \text{coNP/poly}$. □

The authors remark that an alternative proof of Theorem 5.2 can be obtained using a result by Lagerkvist and Wahlström [21, Lemma 47] (also refer to the discussion

after the proof of Lemma 47). It is shown that a k -ary symmetric relation is preserved by φ_1 if and only if the satisfying weights form a complete arithmetic progression, that is, there exist $a, b \in \mathbb{N}_0$ such that $S = \{a + c \cdot b \mid c \in \mathbb{N}_0, a + c \cdot b \leq k\}$. It is easy to see that if S is a complete arithmetic progression, then the relation can be captured by the polynomial $p(x) := -a + \sum_{i \in [k]} x_i$ over $\mathbb{Z}/b\mathbb{Z}$.

This leads to the same dichotomy as in Theorem 5.2. If Γ preserves φ_1 , then it follows from the above that then every $R \in \Gamma$ is captured by a linear polynomial, implying that $\text{CSP}(\Gamma)$ has a kernel with linearly many constraints by Theorem 3.5. On the other hand, if a relation does not preserve φ_1 , then it cone-defines 2-OR (Proposition 2.12) leading to a lower bound using Theorem 3.10.

6 Low-Arity Classification

In this section, we will give a full classification of the sparsifiability for Boolean constraint languages that consist only of low-arity relations. We start by providing some additional properties of cone-definability.

Observation 6.1 *If a Boolean relation T of arity m is cone-definable from a Boolean relation U of arity n , then $m \leq n$.*

Observation 6.2 *Suppose a Boolean relation T is cone-definable from a Boolean relation U , and that g is a partial operation that is idempotent and self-dual. If g preserves U , then g preserves T .*

Observation 6.3 (transitivity of cone-definability) *Suppose that T_1, T_2, T_3 are Boolean 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 6.4 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 6.1 and 6.2. 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 6.5 *If Boolean relations T, U are cone-interdefinable, then they have the same arity.*

Proposition 6.6 *Suppose that T and U are Boolean 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 .*

The next results will show that when the constraint language consists of only low-arity relations, if the constraint language is not balanced, it can cone-define the 2-OR relation.

Observation 6.7 *Each Boolean relation of arity 1 is balanced.*

Theorem 6.8 *A Boolean relation of arity 2 is balanced if and only if it is not cone-interdefinable with the 2-OR relation.*

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

(\Rightarrow) 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 2.10. 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 $g: \{0, 1\}^2 \rightarrow \{0, 1\}^2$ 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.

(\Leftarrow) We again use contraposition. Suppose R is not balanced; we will prove R is cone-interdefinable with 2-OR. Let $f: \{0, 1\}^k \rightarrow \{0, 1\}$ be a balanced partial Boolean operation of minimum arity that does not preserve R . Let $\alpha_1, \dots, \alpha_k \in \mathbb{Z}$ be the coefficients of f , as in Definition 2.5. Minimality of f implies that $\alpha_i \neq 0$ for all $i \in [k]$. Let $s_1, \dots, s_k \in R$ such that $f(s_1, \dots, s_k) = u \in \{0, 1\}^2 \setminus R$ witnesses that f does not preserve R . By Definition 2.5 we have $u = \sum_{i=1}^k \alpha_i s_i$ and $\sum_{i=1}^k \alpha_i = 1$. By minimality of f , all tuples s_1, \dots, s_k are distinct.

Claim 6.9 *The arity k of operation f is 3.*

Proof Since $s_1, \dots, s_k \in R \subseteq \{0, 1\}^2$ 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 $f(s_1) = s_1 \in R$, contradicting that $f(s_1, \dots, s_k) = f(s_1) = u \notin R$. 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 $f(s_1, \dots, s_k) = f(s_1, s_2) = \alpha_1 s_1 + \alpha_2 s_2 = u \in \{0, 1\}^2 \setminus R$, 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 non-zero 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 minimality. ■

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 show the classification of the arity-3 relations with a linear sparsification in Theorem 6.12, we first present some additional lemmas and definitions. Let $U \subseteq \{0, 1\}^n$ be a relation. We say that $w \in \{0, 1\}^n$ is a *witness* for U if $w \notin U$, and there exists a balanced operation $f: \{0, 1\}^k \rightarrow \{0, 1\}$ and tuples $t_1, \dots, t_k \in U$ such that $w = f(t_1, \dots, t_k)$. Observe that U is not balanced if and only if there exists a witness for U .

Lemma 6.10 *Suppose that $U \subseteq \{0, 1\}^n$ is a Boolean relation, and that there exist an integer c and a natural number $m > 1$ such that, for each $u \in U$, it holds that*

$$\text{weight}(u) \equiv_m c.$$

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

Proof Since w is a witness for U , there exist tuples $t_1 = (t_{1,1}, \dots, t_{1,n}), \dots, t_k = (t_{k,1}, \dots, t_{k,n})$ and a balanced operation $f: \{0, 1\}^k \rightarrow \{0, 1\}$ such that $f(t_1, \dots, t_k) = w$. Let $\alpha_1, \dots, \alpha_k$ be the coefficients of f . From $f(t_1, \dots, t_k) = w$, we obtain that $\alpha_1 \text{weight}(t_1) + \dots + \alpha_k \text{weight}(t_k) = \text{weight}(w)$. Since $\sum_{i \in [k]} \alpha_i = 1$ by definition of a balanced operation, we have

$$\alpha_1 \text{weight}(t_1) + \dots + \alpha_k \text{weight}(t_k) \equiv_m \alpha_1 c + \dots + \alpha_k c = \left(\sum_{i \in [k]} \alpha_i \right) c = c$$

and the result follows. □

We will view Boolean tuples of arity n as maps $f: [n] \rightarrow \{0, 1\}$, via the natural correspondence where such a map f represents the tuple $(f(1), \dots, f(n))$. We freely interchange between these two representations of tuples.

For $S \subseteq \mathbb{N}$, we say that $f: S \rightarrow \{0, 1\}$ is a *no-good* of $U \subseteq \{0, 1\}^n$ when:

- $S \subseteq [n]$;
- each extension $g: [n] \rightarrow \{0, 1\}$ of f is not an element of U ; and
- there exists an extension $h: [n] \rightarrow \{0, 1\}$ of f that is a witness for U .

We say that $f: S \rightarrow \{0, 1\}$ is a *min-no-good* if 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 $s_1 < \dots < s_m$ denote the elements of S ; then, we use $U \upharpoonright S$ to denote the relation $\{(h(s_1), \dots, h(s_m)) \mid h \in U\}$.

Proposition 6.11 *Let $U \subseteq \{0, 1\}^n$ be a relation, let $S \subseteq [n]$, and suppose that $f: S \rightarrow \{0, 1\}$ is a min-no-good of U . Then f is a min-no-good of $U \upharpoonright S$.*

Proof Observe that f is not in $U \upharpoonright S$; since f has an extension that is a witness for U , it follows that f is a witness for $U \upharpoonright S$. Thus, f is a no-good of $U \upharpoonright S$. In order to obtain that f is a min-no-good of $U \upharpoonright S$, it suffices to establish that, for any restriction $f^-: S^- \rightarrow \{0, 1\}$ of f , it holds that f^- is a no-good of U if and only if f^- is a no-good of $U \upharpoonright S$. This follows from what we have established concerning f and the following fact: all extensions $h: S \rightarrow \{0, 1\}$ of f^- are not in $U \upharpoonright S$ if and only if all extensions $h': [n] \rightarrow \{0, 1\}$ of f^- are not in U . □

Using these tools we are finally in position to prove Theorem 6.12.

Theorem 6.12 *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 .*

Proof Let $f: S \rightarrow \{0, 1\}$ 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 6.11), which is not possible since $U \upharpoonright S$ would have arity 1 and hence would be preserved by all balanced operations (by Observation 6.7).

For the remaining cases, by replacing U with a relation that is interdefinable with it, we may assume that $f : S \rightarrow \{0, 1\}$ 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 6.11, f is a min-no-good of $U \upharpoonright S$. By Theorem 6.8, 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.

Claim 6.13 *If there is a realization that is a good tuple, then the 2-OR relation is cone-definable from U .*

Proof We show cone-definability via a tuple of the form (x_1, x_2, y) where $y \in \{0, 1, x_1, x_2, \neg x_1, \neg x_2\}$. The right setting for y can be derived from the realization that forms a good tuple.

- choose $y = 0$ 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 $(0, 1, 0)$.

It is easy to verify that this choice of y gives the desired cone-definition. ■

Claim 6.14 *There is a realization that is a good tuple.*

Proof Proof 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 6.10 to U with $a = 2$ and $m = 3$, we obtain that for any witness w for U , it holds that $\text{weight}(w) \equiv_3 2$. 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 as f maps both $s \in S$ to 0; we have thus contradicted that f is a no-good of U . ■

Together, the two claims complete the case that $|S| = 2$.

Suppose that $|S| = 3$. Since f is both a min-no-good and a witness, mapping all $s \in S$ to 0, 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 6.10 with $a = 1$ and $m = 2$, we would obtain that $\text{weight}(f) \equiv_2 1$, 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 $(0, 1, 0), (0, 0, 1), (0, 1, 1) \in R$. □

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 6.15 *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 $\text{CSP}(\Gamma)$ has a kernel with $O(n^k)$ constraints that can be encoded in $O(n^k \log k)$ bits, but for any $\varepsilon > 0$ there is no kernel of size $O(n^{k-\varepsilon})$, unless $\text{NP} \subseteq \text{coNP/poly}$.*

Proof To show that there is a kernel with $O(n^k)$ 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 6.7 and Theorems 6.8 and 6.12 that thereby, Γ is balanced. It now follows from Theorem 4.1 that $\text{CSP}(\Gamma)$ has a kernel with $O(n)$ constraints that can be stored in $O(n \log n)$ bits.
- ($k = 2$) If $k = 2$, there is no relation $R \in \Gamma$ with $|R| = 2^3 - 1 = 7$, as otherwise by Lemma 3.11 such a relation R would cone-define 3-OR which is a contradiction. Thereby, it follows from Theorem 3.1 that $\text{CSP}(\Gamma)$ has a sparsification with $O(n^{3-1}) = O(n^2)$ constraints that can be encoded in $O(n^2 \log n)$ bits.
- ($k = 3$) Given an instance (\mathcal{C}, V) , it is easy to obtain a kernel of with $O(n^3)$ constraints by simply removing duplicate constraints. This kernel can be stored in $O(n^3)$ 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 $O(n^3)$ such tuples, this results in using $O(n^3)$ 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 3.10. Thus, $\text{CSP}(\Gamma)$ has no kernel of size $O(n^{k-\varepsilon})$ for any $\varepsilon > 0$, unless $\text{NP} \subseteq \text{coNP/poly}$. □

7 Capturing Polynomials Versus Compression via Maltsev Embeddings

In this section we compare our polynomial-based framework for linear sparsification to the framework of Lagerkvist and Wahlström [19] based on Maltsev embeddings.

7.1 Maltsev Embeddings and Definitions

To facilitate the discussion, we introduce some additional concepts. A ternary operation $f : D^3 \rightarrow D$ over a domain D is a *Maltsev operation* if it satisfies the identities $f(x, x, y) = y$ and $f(x, y, y) = x$ for all $x, y \in D$.

Definition 7.1 ([19, Definition 7]) A constraint language Γ over a domain D admits an *embedding* over the constraint language Γ' over $D' \supseteq D$ if there exists a bijection $h : \Gamma \rightarrow \Gamma'$ such that for every $R \in \Gamma$, if the arity of R is k then the arity of $h(R)$ is k and $h(R) \cap D^k = R$.

If Γ' is preserved by a Maltsev operation, then Γ is said to *admit a Maltsev embedding*. Lagerkvist and Wahlström proved [19, Theorems 10–11] that if Γ is a constraint language (over a possibly non-Boolean domain) that admits a Maltsev embedding over Γ' with a finite domain, then $\text{CSP}(\Gamma)$ has a kernel with $O(n)$ constraints. Hence constraint languages admitting Maltsev embeddings over finite domains admit linear sparsifications, just as balanced constraint languages.

In a quest to understand which CSPs can be sparsified through this route, they investigated which constraint languages admit Maltsev embeddings using universal algebra. For this purpose, they defined a *universal partial Maltsev operation* as a partial Boolean operation f such that each Boolean constraint language Γ that admits a Maltsev embedding is preserved by f .

Definition 7.2 For a k -ary operation $f: D^k \rightarrow D$ on a domain $D \supseteq \{0, 1\}$, the partial Boolean operation $f_{\mathbb{B}}$ is the restriction of f to Boolean arguments that result in a Boolean value. Hence $\text{domain}(f_{\mathbb{B}}) = \{x \in \{0, 1\}^k \mid f(x) \in \{0, 1\}\}$, and for each k -tuple x in the domain of $f_{\mathbb{B}}$ we have $f_{\mathbb{B}}(x) = f(x)$.

Definition 7.3 ([19, Definition 14]) The infinite domain D_∞ is recursively defined as follows. It contains the elements 0 and 1 along with each triple (x, y, z) where $x, y, z \in D_\infty$ with $x \neq y$ and $y \neq z$. The Maltsev operation $u: D_\infty^3 \rightarrow D_\infty$ is defined by setting $u(x, x, y) = y$, setting $u(x, y, y) = x$, and setting $u(x, y, z) = (x, y, z)$ otherwise.

We define $[[u]]$ as the set of all operations that can be defined as a term over the algebra $(D_\infty, \{u\})$. Hence an arity- k operation $f \in [[u]]$ can be defined either as a projection, so that $f(x_1, \dots, x_k) = x_i$ for some $i \in [k]$, or can be recursively defined from operations $f_1, f_2, f_3 \in [[u]]$ via $f(x_1, \dots, x_k) = u(f_1(x_1, \dots, x_k), f_2(x_1, \dots, x_k), f_3(x_1, \dots, x_k))$. We will use this recursive decomposition of operations in $[[u]]$ later in our proofs. The universal partial Maltsev operations can be characterized precisely [19, Theorems 13–15, p.165] as the operations $f_{\mathbb{B}}$ for $f \in [[u]]$.

Any Boolean constraint language that is preserved by all universal partial Maltsev operations, has [20, Theorem 28] a Maltsev embedding over D_∞ . However, since only *finite-domain* Maltsev embeddings lead to linear sparsifications, this infinite-domain embedding does not directly have algorithmic applications.

In the remainder of this section, we explore relations between balanced Boolean constraint languages (which can be sparsified using capturing polynomials) and Boolean constraint languages admitting a Maltsev embedding.

7.2 Balanced Constraint Languages Versus Maltsev Embeddings

The next theorem shows that being balanced is at least as strong of a requirement as being preserved by all universal partial Maltsev operations. In particular, any balanced relation is preserved by all universal partial Maltsev operations. This implies that if the Maltsev approach does not apply to obtain a linear sparsification for a Boolean CSP, then the polynomial-based framework does not apply either.

Via [20, Theorem 28], this theorem yields that any balanced constraint language has a Maltsev embedding, over the infinite domain D_∞ . We will strengthen this result in Theorem 7.6 by showing that any balanced constraint language has a Maltsev embedding over a finite domain. Since the proof of Theorem 7.6 does not use Theorem 7.4, this theorem follows directly from Theorem 7.6. However, we include the present theorem and the proof as we believe that the proof technique is of independent interest; in contrast to the proof of Theorem 7.6, it does not use any of the machinery established in Sect. 4 and instead provides insight into the similarities between universal Maltsev operations and balanced operations.

Theorem 7.4 *Let Γ be a finite Boolean constraint language. If there exists a universal partial Maltsev operation f such that Γ is not preserved by f , then Γ is not balanced.*

Proof We introduce a function to associate an integer value to every element of D_∞ , as follows. Let $\text{val}: D_\infty \rightarrow \mathbb{N}$ be given by $\text{val}(0) := 0$, $\text{val}(1) := 1$ and $\text{val}((d_1, d_2, d_3)) := \text{val}(d_1) - \text{val}(d_2) + \text{val}(d_3)$ for $d_1, d_2, d_3 \in D_\infty$. We start by proving the following claim.

Claim 7.5 *Let $f \in \{u\}$ have arity m . There is a sequence $\alpha_1, \dots, \alpha_m \in \mathbb{Z}$ with $\sum_{i \in [m]} \alpha_i = 1$, such that for each Boolean vector $x_1, \dots, x_m \in \{0, 1\}$:*

$$\sum_{i \in [m]} \alpha_i x_i = \text{val}(f(x_1, \dots, x_m)).$$

Proof We prove this by induction on the structure of f as given by a term over u .

(Base case) If $f(x_1, \dots, x_m) := x_j$ for $j \in [m]$, we define $\alpha_j := 1$ and $\alpha_i := 0$ for all $i \neq j$. By this definition, $\sum_{i \in [m]} \alpha_i = 1$ and $\text{val}(f(x_1, \dots, x_m)) = x_j = \sum_{i \in [m]} \alpha_i x_i$.

(Step) Let $f(x_1, \dots, x_m) = u(f_1(x_1, \dots, x_m), f_2(x_1, \dots, x_m), f_3(x_1, \dots, x_m))$. For $b \in [3]$, choose coefficients $\alpha_{b,1}, \dots, \alpha_{b,m} \in \mathbb{Z}$ with $\sum_{i \in [m]} \alpha_{b,i} = 1$ such that

$$\sum_{i \in [m]} \alpha_{b,i} x_i = \text{val}(f_b(x_1, \dots, x_m))$$

for all $x_1, \dots, x_m \in \{0, 1\}^m$. These coefficients exist by the induction hypothesis. For $i \in [m]$, define $\alpha_i := \alpha_{1,i} - \alpha_{2,i} + \alpha_{3,i}$. By this definition,

$$\sum_{i \in [m]} \alpha_i = \sum_{i \in [m]} \alpha_{1,i} - \sum_{i \in [m]} \alpha_{2,i} + \sum_{i \in [m]} \alpha_{3,i} = 1 - 1 + 1 = 1,$$

as desired. Let $x_1, \dots, x_m \in \{0, 1\}$ be given. To show that $\sum_{i \in [m]} \alpha_i x_i = \text{val}(f(x_1, \dots, x_m))$ we distinguish three cases.

Suppose $f_1(x_1, \dots, x_m) = f_2(x_1, \dots, x_m)$. Then $f(x_1, \dots, x_m) = f_3(x_1, \dots, x_m)$ by the definition of u , and thus

$$\begin{aligned} \sum_{i \in [m]} \alpha_i x_i &= \sum_{i \in [m]} \alpha_{1,i} x_i - \sum_{i \in [m]} \alpha_{2,i} x_i + \sum_{i \in [m]} \alpha_{3,i} x_i \\ &= \text{val}(f_1(x_1, \dots, x_m)) - \text{val}(f_2(x_1, \dots, x_m)) + \text{val}(f_3(x_1, \dots, x_m)) \\ &= \text{val}(f_3(x_1, \dots, x_m)) = \text{val}(f(x_1, \dots, x_m)). \end{aligned}$$

If $f_3(x_1, \dots, x_m) = f_2(x_1, \dots, x_m)$, then a symmetric argument to the case above shows that indeed $\sum_{i \in [m]} \alpha_i x_i = f(x_1, \dots, x_m)$.

Otherwise, $f_3(x_1, \dots, x_m) \neq f_2(x_1, \dots, x_m)$ and $f_1(x_1, \dots, x_m) \neq f_2(x_1, \dots, x_m)$. It follows that $f(x_1, \dots, x_m) = (f_1(x_1, \dots, x_m), f_2(x_1, \dots, x_m), f_3(x_1, \dots, x_m))$ and we obtain

$$\begin{aligned} \sum_{i \in [m]} \alpha_i x_i &= \sum_{i \in [m]} \alpha_{1,i} x_i - \sum_{i \in [m]} \alpha_{2,i} x_i + \sum_{i \in [m]} \alpha_{3,i} x_i \\ &= \text{val}(f_1(x_1, \dots, x_m)) - \text{val}(f_2(x_1, \dots, x_m)) + \text{val}(f_3(x_1, \dots, x_m)) \\ &= \text{val}((f_1(x_1, \dots, x_m), f_2(x_1, \dots, x_m), f_3(x_1, \dots, x_m))) \\ &= \text{val}(f(x_1, \dots, x_m)), \end{aligned}$$

concluding the proof of this claim. ■

Using the claim we prove Theorem 7.4. Let $h: \{0, 1\}^m \rightarrow \{0, 1\}$ be an m -ary universal partial Maltsev operation that does not preserve Γ . As described in Sect. 7.1, there exists $f \in [\{u\}]$ such that $h = f_{\mathbb{B}}$. Let $R \in \Gamma$ be a k -ary relation such that R is not preserved by h . We show that there exists a balanced operation g_f that does not preserve R . By Claim 7.5 there exist coefficients $\alpha_1, \dots, \alpha_m \in \mathbb{Z}$ such that

$$\sum_{i \in [m]} \alpha_i x_i = \text{val}(f(x_1, \dots, x_m))$$

for all $x_1, \dots, x_m \in \{0, 1\}$. Let g_f be the balanced operation with coefficients α_i .

Since $f_{\mathbb{B}}$ does not preserve R , there are $s_1, \dots, s_m \in R$ such that $f_{\mathbb{B}}(s_1, \dots, s_m)$ is well-defined and $f_{\mathbb{B}}(s_1, \dots, s_m) \notin R$. Let $s_i = (s_{i,1}, \dots, s_{i,k})$. Then since $f_{\mathbb{B}}(s_1, \dots, s_m)$ is well-defined, it evaluates to a 0/1-tuple and

$$\text{val}(f(s_{1,j}, \dots, s_{m,j})) = f(s_{1,j}, \dots, s_{m,j}),$$

for all $j \in [k]$. Since $\sum_{i \in [m]} \alpha_i x_i = \text{val}(f(x_1, \dots, x_m))$ for all $x_1, \dots, x_m \in \{0, 1\}$, it follows that g_f violates R . So g_f is a balanced operation that does not preserve Γ . □

7.3 Balanced Constraint Languages Versus Finite-Domain Maltsev Embeddings

Theorem 7.4 implies [20, Theorem 28] that any balanced constraint language has a Maltsev embedding over the infinite domain D_∞ . We show in the next theorem that in fact, every balanced constraint language allows a Maltsev embedding over a *finite*

domain. Thus, there are in fact two ways to obtain a kernel with $O(n)$ constraints for balanced constraint languages, one given by Theorem 4.1 and one via Maltsev embeddings [19, Theorems 10–11].

Theorem 7.6 *If Γ is a finite balanced Boolean constraint language, then Γ admits a Maltsev embedding over $\mathbb{Z}/q\mathbb{Z}$ for some $q \in \mathbb{N}$.*

Proof Since Γ is balanced, it follows from Corollary 4.12 that there exists $q \in \mathbb{N}$ such that for every $R \in \Gamma$, $u \notin R$ there exists a linear polynomial $p_{u,R}$ such that $p_{u,R}$ captures u with respect to R over $\mathbb{Z}/q\mathbb{Z}$. Define $D := \mathbb{Z}/q\mathbb{Z}$. Let $\varphi: D^3 \rightarrow D$ be the Maltsev operation given by $\varphi(x, y, z) = x - y + z \pmod q$.

We now show how to define a constraint language Γ' that is the Maltsev embedding of Γ . For any k -ary relation $R \in \Gamma$, we show how to define R' such that $R' \cap \{0, 1\}^k = R$. Define

$$R' := \{(x_1, \dots, x_k) \in D^k \mid p_{u,R}(x_1, \dots, x_k) \equiv_q 0 \text{ for all } u \in \{0, 1\}^k \setminus R\},$$

and let $\Gamma' := \{R' \mid R \in \Gamma\}$. We now show that $R' \cap \{0, 1\}^k = R$. Clearly, for all $(x_1, \dots, x_k) \in R$ and $u \notin R$ it holds that $p_{u,R}(x_1, \dots, x_k) \equiv_q 0$ as $p_{u,R}$ captures u with respect to R , and thereby $(x_1, \dots, x_k) \in R'$. For the other direction, consider $x = (x_1, \dots, x_k) \in \{0, 1\}^k \cap R'$. Suppose towards a contradiction that $(x_1, \dots, x_k) \notin R$, then $p_{x,R}(x_1, \dots, x_k) \not\equiv_q 0$ since $p_{x,R}$ captures x with respect to R . This contradicts that $x \in R'$.

It remains to prove that every $R' \in \Gamma'$ is preserved by φ . Towards a contradiction, suppose there are $x = (x_1, \dots, x_k)$, $y = (y_1, \dots, y_k)$, and $z = (z_1, \dots, z_k)$ in R' such that the tuple w obtained by applying φ to x , y , and z coordinate-wise is not contained in R' . Since $w \notin R'$ there exists $u \notin R$ such that $p_{u,R}(w_1, \dots, w_k) \not\equiv_q 0$. However, since $x, y, z \in R'$ it follows that

$$p_{u,R}(x_1, \dots, x_k) \equiv_q p_{u,R}(y_1, \dots, y_k) \equiv_q p_{u,R}(z_1, \dots, z_k) \equiv_q 0.$$

It then follows from Proposition 2.3 that $p_{u,R}(w_1, \dots, w_k) \equiv_q p(x_1 - y_1 + z_1, \dots, x_k - y_k + z_k) \equiv_q 0$, which is a contradiction. Thereby, Γ' is preserved by φ , as desired. □

Theorem 7.6 shows that balanced constraint languages have (finite-domain) Maltsev embeddings. In the next theorem, we prove a partial converse: constraint languages that are not balanced, do not admit Maltsev embeddings of a particular type over a (possibly infinite) domain. The particular type we consider consists of embeddings for which the constraint language Γ' into which we embed has a group structure on its domain, and the Maltsev operation that preserves Γ' is the coset generating operation of the group. Let us give the relevant definitions.

Definition 7.7 Let (D, \cdot) be a group. The *coset generating operation* of the group is the Maltsev operation $c: D^3 \rightarrow D$ defined by $c(x, y, z) = x \cdot y^{-1} \cdot z$.

Theorem 7.8 *Let Γ be a Boolean constraint language that is not balanced, and let (D, \cdot) be a group with coset generating operation c . Then there is no constraint*

language Γ' over domain $D \supseteq \{0, 1\}$ that is preserved by c and for which Γ admits an embedding over Γ' , not even if the identity element of (D, \cdot) is allowed to differ from $\{0, 1\}$.

Proof Suppose for contradiction that there exists a group (D, \cdot) with coset generating operation c and a constraint language Γ' such that Γ' is preserved by c and Γ admits an embedding over Γ' . Let $R \in \Gamma$ be a relation that is not balanced, suppose R has arity k . Since R is not balanced, take $r_1, \dots, r_m \in R$ with $r_1 - r_2 \dots + r_m = u \notin R$. Let $\widehat{R} \in \Gamma'$ be the image of R under the considered Maltsev embedding. We start by proving the following claim.

Claim 7.9 For all $i \in [k]$, the following equation holds over the group (D, \cdot) :

$$r_{1,i} \cdot r_{2,i}^{-1} \cdot \dots \cdot r_{m,i} = u_i.$$

Proof We show by induction that for all $x_1, \dots, x_m \in \{0, 1\}$ with $x_1 - x_2 \dots + x_m = y$ for $y \in \{0, 1\}$, it holds that $x_1 \cdot x_2^{-1} \cdot \dots \cdot x_m = y$ over D .

(Base case) If $m = 1$, we trivially obtain that $x_1 = y$.

(Step) Suppose $m > 1$. We start by showing that there exists $j \in [m - 1]$ such that $x_j = x_{j+1}$. Suppose not, then we are in one of the two cases below.

- $x_1 = x_3 = \dots = x_m = 0$ and $x_2 = x_4 = \dots = x_{m-1} = 1$, or
- $x_1 = x_3 = \dots = x_m = 1$ and $x_2 = x_4 = \dots = x_{m-1} = 0$.

In both cases it is easily verified that for this choice of variables, $x_1 - x_2 \dots + x_m \notin \{0, 1\}$, which is a contradiction. Therefore, there exists $j \in [m - 1]$ such that $x_j = x_{j+1}$. Suppose for ease of notation that j is even, the case when j is odd follows symmetrically. It is easy to verify that $y = x_1 - x_2 \dots + x_{j-1} - x_j + x_{j+1} - x_{j+2} \dots + x_m = x_1 - x_2 \dots + x_{j-1} - x_{j+2} \dots + x_m$, and thus it follows from the induction hypothesis that $x_1 \cdot x_2^{-1} \cdot \dots \cdot x_{j-1} \cdot x_{j+2}^{-1} \cdot \dots \cdot x_m = y$. Thereby

$$\begin{aligned} &x_1 \cdot x_2^{-1} \cdot \dots \cdot x_{j-1} \cdot x_j^{-1} \cdot x_{j+1} \cdot x_{j+2}^{-1} \cdot \dots \cdot x_m = \\ &x_1 \cdot x_2^{-1} \cdot \dots \cdot x_{j-1} \cdot (x_j^{-1} \cdot x_j) \cdot x_{j+2}^{-1} \cdot \dots \cdot x_m = \\ &x_1 \cdot x_2^{-1} \cdot \dots \cdot x_{j-1} \cdot x_{j+2}^{-1} \cdot \dots \cdot x_{m-2} = y. \end{aligned}$$

Since r_1, \dots, r_m were chosen such that $r_1 - r_2 \dots + r_m = u \in \{0, 1\}^k \setminus R$, the statement of the claim follows. ■

Recall that \widehat{R} is preserved by c . We have the following claim.

Claim 7.10 Let \widehat{R} be a k -ary relation and $m \geq 3$ be odd. If \widehat{R} is preserved by c , then it is preserved by the m -ary operation $f_m: D^m \rightarrow D$ given by $f_m(x_1, \dots, x_m) := x_1 \cdot x_2^{-1} \cdot \dots \cdot x_m$.

Proof We show this by a simple induction. If $m = 3$, the statement is true since in this case f and c are equivalent. Let $m > 3$ and let $t_1, \dots, t_m \in \widehat{R}$ be given, we show

$f(t_1, \dots, t_m) \in \widehat{R}$. Let $t' := c(t_1, t_2, t_3)$, observe that $t' \in \widehat{R}$ since \widehat{R} is preserved by c . Choose $i \in [k]$ and observe that

$$\begin{aligned} f_m(t_{1,i}, \dots, t_{m,i}) &= t_{1,i} \cdot t_{2,i}^{-1} \cdot t_{3,i} \cdot t_{4,i}^{-1} \cdots t_{m,i} \\ &= c(t_{1,i}, t_{2,i}, t_{3,i}) \cdot t_{4,i}^{-1} \cdots t_{m,i} \\ &= t' \cdot t_{4,i}^{-1} \cdots t_{m,i}. \end{aligned}$$

Thereby, $f_m(t_1, \dots, t_m) = f_{m-2}(t', t_4, \dots, t_m)$. It follows from the induction hypothesis that f_{m-2} is preserved by \widehat{R} and thus $f_{m-2}(t', t_4, \dots, t_m) \in \widehat{R}$. ■

It follows from the fact that c preserves \widehat{R} and Claim 7.10, that f_m preserves \widehat{R} . However, by Claim 7.9, it follows that $f_m(r_1, \dots, r_m) = u$ and thus $u \in \widehat{R}$, contradicting that we have given a valid Maltsev embedding of Γ , since $u \in \{0, 1\}^k$, but $u \notin R$. □

We conclude the subsection with a discussion of the implications of Theorems 7.6 and 7.8. On the one hand, Theorem 7.6 shows that balanced constraint languages admit Maltsev embeddings over finite domains. On the other hand, Theorem 7.8 shows that unbalanced constraint languages do not allow Maltsev embeddings over the coset generating operation of a group, not even an infinite group. As a corollary to these results, we therefore obtain an infinite-domain to finite-domain transformation, for embeddings via coset generating operations.

Corollary 7.11 *If Γ is a finite Boolean constraint language that has a Maltsev embedding over Γ' , where the domain D of Γ' is a (possibly infinite) group and Γ' is preserved by the coset generating operation of the group, then Γ has a Maltsev embedding over $\mathbb{Z}/q\mathbb{Z}$ for some $q \in \mathbb{N}$.*

7.4 Preservation by Balanced or Universal Partial Maltsev Operations

In this section we compare preservation by balanced operations to preservation by universal partial Maltsev operations. Theorem 7.4 implies that every balanced constraint language is preserved by all universal partial Maltsev operations. At this point, it is unknown whether the converse also holds. If the Boolean constraint language Γ is preserved by all universal partial Maltsev operations, then is it also balanced?

We have not managed to resolve this question, but we present some insights in this direction. Recall that a_i is the alternating (partial) operation of arity i , for odd $i \geq 1$. It is easy to verify that a_3 is equivalent to $u_{\mathbb{B}}$. We show the following result about a_5 .

Theorem 7.12 *Let Γ be a Boolean constraint language. If Γ is not preserved by a_5 , then there is a universal partial Maltsev operation that does not preserve Γ .*

Proof We start by considering the term $f \in [\{u\}]$ defined as follows:

$$f(x_1, \dots, x_5) := u(x_1, u(x_2, x_3, u(x_1, x_2, x_3)), u(x_5, x_4, u(x_3, x_2, x_1))).$$

The key of the proof is that the Boolean restriction $f|_{\mathbb{B}}$ of f does not preserve Γ . We start by showing how f relates to the alternating operation of arity 5.

Claim For all $x_1, \dots, x_5 \in \{0, 1\}$ such that $a_5(x_1, \dots, x_5) \in \{0, 1\}$, it holds that

$$f(x_1, \dots, x_5) = a_5(x_1, \dots, x_5).$$

Proof Suppose $a_5(x_1, \dots, x_5) \in \{0, 1\}$, we do a case distinction.

- ($u(x_5, x_4, u(x_3, x_2, x_1)) \in \{0, 1\}$) Observe that in particular, this implies that $u(x_3, x_2, x_1) \in \{0, 1\}$, implying that $x_3 = x_2$ or $x_1 = x_2$. If $x_3 = x_2$, we obtain that $u(x_2, x_3, u(x_1, x_2, x_3)) = u(x_3, x_3, u(x_1, x_3, x_3)) = x_1$ and $u(x_3, x_2, x_1) = x_1$, implying that $f(x_1, \dots, x_5) = u(x_5, x_4, u(x_3, x_2, x_1)) = u(x_5, x_4, x_1)$ and since $u(a, b, c) = a - b + c$ if $u(a, b, c) \in \{0, 1\}$, the result follows since $a_5(x_1, x_2, x_3, x_4, x_5) = x_5 - x_4 + x_1 \in \{0, 1\}$ for $x_2 = x_3$. Similarly, if $x_1 = x_2$, then $u(x_2, x_3, u(x_1, x_2, x_3)) = u(x_1, x_3, u(x_1, x_1, x_3)) = x_1$. Just like in the previous case, this implies $f(x_1, \dots, x_5) = u(x_5, x_4, u(x_3, x_2, x_1))$ and the result follows.
- ($u(x_5, x_4, u(x_3, x_2, x_1)) \notin \{0, 1\}$) We again consider two options, based on whether $u(x_3, x_2, x_1)$ is in $\{0, 1\}$ or not. Observe that if $u(x_3, x_2, x_1) \in \{0, 1\}$, the reason that $u(x_5, x_4, u(x_3, x_2, x_1)) \notin \{0, 1\}$ is that $x_5 = u(x_3, x_2, x_1)$ and $x_5 \neq x_4$. But then by definition, $x_5 = x_3 - x_2 + x_1$ and thus if $x_5 = 1$, then $x_4 = 0$ and we obtain $x_5 - x_4 + x_3 - x_2 + x_1 = 2 \notin \{0, 1\}$, which is a contradiction. Similarly, if $x_5 = 0$ we obtain that $x_5 - x_4 + x_3 - x_2 + x_1 = -1 \notin \{0, 1\}$, which is again a contradiction. We thereby conclude that $u(x_3, x_2, x_1) \notin \{0, 1\}$.

By definition, this implies $x_3 = x_1 \neq x_2$. It follows that $x_4 \neq x_5$ to ensure that $a_5(x_1, \dots, x_5) = x_5 - x_4 + x_3 - x_2 + x_1 \in \{0, 1\}$. Furthermore, observe that $x_4 = x_3$ for this same reason. Thus, $x_1 = x_3 = x_4 \neq x_2 = x_5$. It follows that $a_5(x_1, \dots, x_5) = x_1$. Furthermore, substituting this into the formula shows that $u(x_5, x_4, u(x_3, x_2, x_1)) = u(x_2, x_1, u(x_1, x_2, x_1)) = (x_2, x_1, (x_1, x_2, x_1)) = u(x_2, x_3, u(x_1, x_2, x_3))$, implying that $f(x_1, \dots, x_5) = x_1$, as desired. ■

Now let Γ be a constraint language that is not preserved by a_5 . It follows from Claim 7.4 that for any $(x_1, \dots, x_5) \in \text{domain}(a_5)$, it holds that $a_5(x_1, \dots, x_5) = f(x_1, \dots, x_5)$ and $(x_1, \dots, x_5) \in \text{domain}(f|_{\mathbb{B}})$. It follows that Γ is not preserved by $f|_{\mathbb{B}}$, which is a universal partial Maltsev operation [19, Theorem 15]. □

Theorem 7.12, together with the observation that a_3 is equivalent to $u|_{\mathbb{B}}$, have the following consequence. If a constraint language Γ is unbalanced because some alternating operation of arity at most five does not preserve it, then Γ does not admit a Maltsev embedding, not even over an infinite domain. We leave it for future work to determine whether there exist Boolean constraint languages that admit finite-domain Maltsev embeddings, but are not balanced. Are there constraint languages for which the Maltsev framework yields linear kernelizations, but the polynomial-based framework does not?

8 Conclusion

In this paper we analyzed the best-case and worst-case sparsifiability of $\text{CSP}(\Gamma)$ for intractable finite Boolean constraint languages Γ . First of all, we characterized those Boolean CSPs for which a nontrivial sparsification is possible, based on the number of non-satisfying assignments. Then we presented our key structural contribution: the notion of balanced constraint languages. We have shown that $\text{CSP}(\Gamma)$ allows a sparsification with $O(n)$ constraints whenever Γ is balanced. The constructive proof of this statement yields a polynomial-time algorithm to find a series of degree-1 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 $\text{CSP}(\Gamma)$ allows a linear sparsification. Furthermore, we fully classified the sparsifiability of $\text{CSP}(\Gamma)$ when Γ contains relations of arity at most three, based on the arity of the largest OR that can be cone-defined from Γ . As we explain in the next paragraph, it follows from results of Lagerkvist and Wahlström [19,20] that for constraint languages of arbitrary arity, the exponent of the best sparsification size does not always match the arity of the largest OR cone-definable from Γ . Hence the type of characterization we presented is inherently limited to low-arity constraint languages. It may be possible to extend our characterization to languages of arity at most four, however.

In their work, [20, Theorem 32] Lagerkvist and Wahlström show the following. For every integer $d \geq 3$ and for every finite set P of partial polymorphisms for which the set of Boolean relations preserved by P can pp-define all Boolean relations, there is a polynomial-parameter transformation [1,2] from d -CNF-SAT instances with n variables, to equivalent instances of $\text{CSP}(\Gamma)$ on $O(n^c)$ variables. Here $c \in \mathbb{N}$ depends only on P and Γ is a finite Boolean constraint language whose relations are preserved by all operations in P . Assuming $\text{NP} \not\subseteq \text{coNP/poly}$, the problem d -CNF-SAT has no kernel [7] of bitsize $O(n^{d-\varepsilon})$ for any $\varepsilon > 0$. Via the cited transformation, this implies $\text{CSP}(\Gamma)$ has no sparsification of bitsize $O(n^{d/c-\varepsilon})$. Hence knowing that the constraint language is preserved by any finite set of partial polymorphisms does not guarantee any polynomial compressibility. Note that any constraint language that can cone-define k -OR for some $k \geq 2$ can also cone-define 2-OR, while Proposition 2.12 shows that being able to cone-define 2-OR is equivalent to being violated by the partial operation φ_1 . A constraint language preserved by the single partial polymorphism φ_1 therefore does not cone-define k -OR for any $k \geq 2$. Using the transformation and incompressibility results mentioned above, we find (assuming $\text{NP} \not\subseteq \text{coNP/poly}$) that for any $d' \in \mathbb{R}$ there is a finite Boolean constraint language $\Gamma_{d'}$ that does not cone-define 2-OR or larger, but for which $\text{CSP}(\Gamma_{d'})$ does not have a sparsification of bitsize $O(n^{d'})$. A general characterization of optimal sparsification size by the arity of the largest cone-definable OR is therefore impossible.

We move to a discussion of future work. The ultimate goal of this line of research is to fully classify the sparsifiability of $\text{CSP}(\Gamma)$, depending on Γ . In particular, we would like to classify those Γ for which $O(n)$ sparsifiability is possible. In this paper, we have shown that Γ being balanced is a sufficient condition to obtain a linear sparsification; it is tempting to conjecture that this condition is also necessary. The simplest example of a Boolean constraint language for which we currently do not understand whether

or not it has a linear sparsification, consists of a single Boolean relation R^* of arity nine. Relation R^* has five satisfying assignments $s_1, \dots, s_5 \in \{0, 1\}^9$:

$$R^* = \left\{ \begin{array}{l} s_1 = (1, 0, 0, 1, 1, 1, 0, 0, 1), \\ s_2 = (0, 0, 0, 0, 1, 0, 0, 1, 1), \\ s_3 = (0, 1, 0, 1, 1, 0, 1, 1, 0), \\ s_4 = (0, 0, 0, 1, 0, 1, 1, 0, 0), \\ s_5 = (0, 0, 1, 0, 0, 1, 1, 1, 1) \end{array} \right\}$$

Relation R^* is not balanced, since $s_1 - s_2 + s_3 - s_4 + s_5 = (1, 1, 1, 1, 1, 1, 1, 1, 1) \notin R^*$. By Theorem 7.12, it follows that R^* is violated by some universal partial Maltsev operation. Hence neither our polynomial framework for compression, nor the Maltsev-based approach yields a linear sparsification for $\text{CSP}(\{R^*\})$. On the other hand, no superlinear lower bound is currently known. Resolving the sparsification complexity of $\text{CSP}(\{R^*\})$ is the first obstacle in a general classification of linearly-compressible Boolean CSPs.

Note that the matrix consisting of the five satisfying assignments of R^* has a very succinct description: there is one column for each vector $x \in \{0, 1\}^5$ for which $x[1] - x[2] + x[3] - x[4] + x[5] = 1$, except for the all-ones column. The presence of these columns ensures that φ_1 preserves R^* , for the simple reason that $\varphi_1(s_i, s_j, s_k)$ for $i, j, k \in [5]$ is only defined when $i = j$ or $j = k$, in which case the output tuple equals one of the input tuples. In other cases, there is an index ℓ for which $s_i[\ell] - s_j[\ell] + s_k[\ell] \in \{-1, 2\}$, making the output undefined. Hence, using Proposition 2.12, relation R^* does not cone-define 2-OR.

Observe that $\text{CSP}(\{R^*\})$ is NP-complete [25] by Schaefer's dichotomy theorem: R^* does not have the constantly-1 operation as a polymorphism, nor the constantly-0 operation; the tuples s_1, s_2 show that R^* is not preserved by the binary AND operation, nor by the binary OR operation; and the tuples s_1, s_2, s_3 show that R^* is not preserved by the ternary majority or minority operations.

Resolving the sparsification complexity of $\text{CSP}(\{R^*\})$, and subsequently obtaining a complete characterization of the Boolean CSPs that admit a linear compression, form the main open problems of this work. Other directions include the investigation of constraint languages of larger arity and the characterization of the CSPs that admit sparsifications with a quadratic, or even larger polynomial number, of constraints.


Acknowledgements We would like to thank Emil Jeřábek for the proof of Lemma 4.3. We thank Andrei Bulatov for insightful discussions, and thank Magnus Wahlström for sharing his ideas and an initial version of the proof of Theorem 7.12. We thank the anonymous referees of Algorithmica for the suggestion that a balanced constraint language has a Maltsev embedding over a single integer ring $\mathbb{Z}/q\mathbb{Z}$, and for comments that improved the presentation of the paper.

References

1. Bodlaender, H.L., Jansen, B.M.P., Kratsch, S.: Kernelization lower bounds by cross-composition. *SIAM J. Discrete Math.* **28**(1), 277–305 (2014). <https://doi.org/10.1137/120880240>
2. Bodlaender, H.L., Thomassé, S., Yeo, A.: Kernel bounds for disjoint cycles and disjoint paths. *Theor. Comput. Sci.* **412**(35), 4570–4578 (2011). <https://doi.org/10.1016/j.tcs.2011.04.039>

3. Bulatov, A., Jeavons, P., Krokhin, A.: Classifying the complexity of constraints using finite algebras. *SIAM J. Comput.* **34**(3), 720–742 (2005). <https://doi.org/10.1137/S0097539700376676>
4. Chen, H.A.: A rendezvous of logic, complexity, and algebra. *ACM Comput. Surv.* (2009). <https://doi.org/10.1145/1189056.1189076>
5. Cygan, M., Fomin, F.V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: *Parameterized Algorithms*. Springer, Berlin (2015). <https://doi.org/10.1007/978-3-319-21275-3>
6. Dell, H., Marx, D.: Kernelization of packing problems. In: *Proceedings of 23rd SODA*, pp. 68–81 (2012). <https://doi.org/10.1137/1.9781611973099.6>
7. Dell, H., van Melkebeek, D.: Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. *J. ACM* **61**(4), 23:1–23:27 (2014). <https://doi.org/10.1145/2629620>
8. Downey, R.G., Fellows, M.R.: *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, Berlin (2013). <https://doi.org/10.1007/978-1-4471-5559-1>
9. Drucker, A.: New limits to classical and quantum instance compression. *SIAM J. Comput.* **44**(5), 1443–1479 (2015). <https://doi.org/10.1137/130927115>
10. Gockenbach, M.: *Finite-Dimensional Linear Algebra*. Discrete Mathematics and Its Applications. Taylor & Francis, Abingdon (2011)
11. Jansen, B.M.P.: On sparsification for computing treewidth. *Algorithmica* **71**(3), 605–635 (2015). <https://doi.org/10.1007/s00453-014-9924-2>
12. Jansen, B.M.P., Pieterse, A.: Optimal sparsification for some binary CSPs using low-degree polynomials. In: *Proceedings of 41st MFCS*, pp. 71:1–71:14 (2016). <https://doi.org/10.4230/LIPIcs.MFCS.2016.71>
13. Jansen, B.M.P., Pieterse, A.: Optimal data reduction for graph coloring using low-degree polynomials. In: *Proceedings of 12th IPEC*, pp. 22:1–22:12 (2017). <https://doi.org/10.4230/LIPIcs.IPEC.2017.22>
14. Jansen, B.M.P., Pieterse, A.: Sparsification upper and lower bounds for graph problems and not-all-equal SAT. *Algorithmica* **79**(1), 3–28 (2017). <https://doi.org/10.1007/s00453-016-0189-9>
15. Jansen, B.M.P., Pieterse, A.: Optimal sparsification for some binary CSPs using low-degree polynomials. *CoRR abs/1606.03233* (2018). [arXiv:1606.03233v2](https://arxiv.org/abs/1606.03233v2)
16. Kannan, R., Bachem, A.: Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix. *SIAM J. Comput.* **8**(4), 499–507 (1979). <https://doi.org/10.1137/0208040>
17. Kratsch, S., Philip, G., Ray, S.: Point line cover: the easy kernel is essentially tight. *ACM Trans. Algorithms* **12**(3), 40:1–40:16 (2016). <https://doi.org/10.1145/2832912>
18. Lagerkvist, V.: Weak bases of Boolean co-clones. *Inf. Process. Lett.* **114**(9), 462–468 (2014). <https://doi.org/10.1016/j.ipl.2014.03.011>
19. Lagerkvist, V., Wahlström, M.: Kernelization of constraint satisfaction problems: a study through universal algebra. In: *Proceedings of 23rd CP*, pp. 157–171 (2017). https://doi.org/10.1007/978-3-319-66158-2_11
20. Lagerkvist, V., Wahlström, M.: Kernelization of constraint satisfaction problems: a study through universal algebra. *CoRR abs/1706.05941* (2017). [arXiv:1706.05941](https://arxiv.org/abs/1706.05941)
21. Lagerkvist, V., Wahlström, M.: Which NP-hard SAT and CSP problems admit exponentially improved algorithms? *CoRR abs/1801.09488* (2018). [arXiv:1801.09488v1](https://arxiv.org/abs/1801.09488v1)
22. Lokshtanov, D., Misra, N., Saurabh, S.: Kernelization—preprocessing with a guarantee. In: *The Multivariate Algorithmic Revolution and Beyond*, pp. 129–161 (2012). https://doi.org/10.1007/978-3-642-30891-8_10
23. Lovász, L.: Chromatic number of hypergraphs and linear algebra. In: *Studia Scientiarum Mathematicarum Hungarica* 11, pp. 113–114 (1976). <http://real-j.mtak.hu/5461/>
24. Nordh, G., Zanuttini, B.: Frozen Boolean partial co-clones. In: *Proceedings of 39th International Symposium on Multiple-Valued Logic*, pp. 120–125. IEEE Computer Society (2009). <https://doi.org/10.1109/ISMVL.2009.10>
25. Schaefer, T.J.: The complexity of satisfiability problems. In: *Proceedings of 10th STOC*, pp. 216–226 (1978). <https://doi.org/10.1145/800133.804350>

Affiliations

Hubie Chen¹ · Bart M. P. Jansen² · Astrid Pieterse³ 

✉ Astrid Pieterse
astrid.pieterse@informatik.hu-berlin.de

Hubie Chen
hubie@dcs.bbk.ac.uk

Bart M. P. Jansen
b.m.p.jansen@tue.nl

- ¹ Birkbeck, University of London, Malet Street, Bloomsbury, London WC1E 7HX, UK
- ² Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
- ³ Institut für Informatik, Humboldt-Universität zu Berlin, Unter den Linden 6, 10099 Berlin, Germany